

# יסודות מדעי המחשב 1

## בשפת C#

תמר בניה וד"ר מיכל ארמוני – ראשי צוות הכתיבה

יעל בילצ'יק

נעה גרדוביץ

עדי גרין

אתי מנשה (סעיפי התבניות)

הילה קדמן (נספח)

ייעוץ: ד"ר דוד גינת

עריכה: לירון ברגר

תשס"ח 2007

אוניברסיטת תל-אביב החוג להוראת המדעים

מטה מל"מ המרכז הישראלי להוראת המדעים ע"ש עמוס דה-שליט

משרד החינוך האגף לתכנון ולפיתוח תכניות לימודים



# יסודות מדעי המחשב 1 בשפת C#

תמר בניה וד"ר מיכל ארמוני – ראשי צוות הכתיבה

יעל בילצ'יק

נעה גרדוביץ

עדי גרין

אתי מנשה (סעיפי תבניות)

הילה קדמן (נספח)

ייעוץ: ד"ר דוד גינת

עריכה: לירון ברגר

כל הזכויות שמורות © 2007

השראה הוצאה לאור, ת"ד 19022, חיפה 31190

טל': 04-8254752, פקס: 1534-8254752

E-Mail: [books@hashraa.co.il](mailto:books@hashraa.co.il)

[www.hashraa.co.il](http://www.hashraa.co.il)



**השראה הוצאה לאור**

מהדורה שנייה 2007

עיצוב העטיפה: טל גרין

אין לשכפל, להעתיק, לצלם, לתרגם, להקליט, לאחסן במאגר מידע כלשהו, לשדר או לקלוט בכל דרך או בכל אמצעי אלקטרוני, אופטי או מכני (לרבות צילום, הקלטה, אינטרנט, מחשב ודואר אלקטרוני), כל חלק שהוא מהחומר שבספר זה. שימוש מסחרי מכל סוג בחומר הכלול בספר זה אסור בהחלט, אלא ברשות מפורשת בכתב מהמוציא לאור ומהגורמים המפורטים להלן.



כל הזכויות שמורות  
משרד החינוך

מסת"ב 965-90844-5-5 ISBN

## פתח דבר

יחידות הלימוד "יסודות מדעי המחשב 1 ו-2" מיועדות להקניית מושגי יסוד ועקרונות שעליהם מושתת תחום מדעי המחשב. פרקי יחידות הלימוד משלבים שני ערוצים – ערוץ תיאורטי וערוץ יישומי. הערוץ התיאורטי מתמקד בחשיבה אלגוריתמית ובפיתוח וניתוח של אלגוריתמים וכוללת התייחסות למושג עצמים. הערוץ היישומי כולל יישום של האלגוריתמים בשפת התכנות C#, שפה מונחית עצמים.

ספר זה כולל את היחידה "יסודות מדעי המחשב 1". היחידה מציגה בעיות ראשונות ואת פתרונותיהן המיועדים לביצוע למחשב. הבעיות נקראות בעיות אלגוריתמיות, ופתרונותיהן – אלגוריתמים. האלגוריתמים מיושמים בתוכניות מחשב. במהלך הלימוד מוצגים המרכיבים הבסיסיים של אלגוריתמים ושל תוכניות מחשב. ההצגה משלבת פיתוח וניתוח של אלגוריתמים, וכוללת התייחסות ראשונית למושג עצמים. פיתוח האלגוריתמים נעשה בשלבים, תוך שימת דגש על ניתוח הבעיה ועל התייחסות להיבטים של נכונות ושל יעילות. כמו כן, מושם דגש על מבנים תבניתיים בפתרונות אלגוריתמיים, והם נקראים תבניות. פירוט מלא של התבניות מופיע באתר הספר: [www.tau.ac.il/~csedu/yesodot.html](http://www.tau.ac.il/~csedu/yesodot.html).

ספר זה פותח על בסיס ספר הלימוד "יסודות מדעי המחשב 1" שפותח במכון ויצמן למדע בסוף שנות ה-90. בספר הקודם נעשה היישום של האלגוריתמים בשפת התכנות Pascal, שפה פרוצדורלית. בספר זה נעשה היישום בשפת C#, שפה מונחית עצמים. העקרונות האלגוריתמיים והנושאים בשמות הפרקים הראשונים בספר זה זהים לאלה שפותחו בידי מכון ויצמן. בספר "יסודות מדעי המחשב 2" מורחב המבט על עצמים, על אלגוריתמים ועל תבניות. לספר זה מצורף מדריך מעבדה מקוון, המופיע באתר הספר המצוין לעיל.

**תודות.** ספר זה פותח בתמיכת מפמ"ר מדעי המחשב במשרד החינוך ד"ר אבי כהן וחברי שתי ועדות המקצוע האחרונות להוראת מדעי המחשב – הועדה בראשות פרופ' עמיהוד אמיר והועדה (הנוכחית) בראשות פרופ' יהודית גל-עזר. תודתנו נתונה להם על תמיכתם ועל הערותיהם. בנוסף, לאורך הספר משולבת התייחסות מפורשת לתבניות בפיתוח ובניתוח של אלגוריתמים. ההתייחסות מבוססת על הספר "תבניות במדעי המחשב" שפיתחו חברי הקבוצה להוראת מדעי-המחשב בחוג להוראת-המדעים באוניברסיטת תל-אביב בשנת 2001. ארנה מילר, אחת מחברות הקבוצה, אף חקרה את הנושא של הוראה מכוונת תבניות, ושיתפה את חברות צוות הכתיבה בניסיונה. תודתנו נתונה לה על כך.

# תוכן עניינים

<b>פרק 1 – מבוא</b>	<b>1</b>
1.1 מהו מחשב?	1
1.2 חומרה	2
1.3 תוכנה	5
1.4 התפתחות המחשבים ומדעי המחשב	8
התפתחות הנדסית וטכנולוגית – חומרה	8
התפתחות הנדסית וטכנולוגית – תוכנה	9
סיכום	11
שאלות נוספות	11
<b>פרק 2 – פתרון בעיות אלגוריתמיות</b>	<b>13</b>
2.1 אלגוריתמים	13
2.2 תבניות	21
סיכום	22
שאלות נוספות	22
<b>פרק 3 – מודל חישוב בסיסי</b>	<b>25</b>
3.1 צעדים ראשוניים: הוראת פלט, הוראת קלט ומשתנים	25
3.2 הוראת השְׁמָה	33
3.3 טבלת מעקב	39
3.4 החלפה בין ערכי משתנים	44
3.5 טיפוסים	46
3.6 קבועים	51
סיכום	52
סיכום מרכיבי שפת C# שנלמדו בפרק 3	54
שאלות נוספות	57
תבניות – פרק 3	61
החלפת ערכים בין שני משתנים, היפוך סדר האיברים בסדרה, ממוצע של סדרת מספרים, הזזה מעגלית בסדרה	
<b>פרק 4 – הרחבה בפיתוח אלגוריתמים</b>	<b>63</b>
4.1 מבט נוסף אל התהליך של פיתוח אלגוריתם ויישומו	63
המחלקה המתמטית	67
4.2 פעולות חלוקה בשלמים	68
עוד על פעולת השארית	73
המרת ערך שלם לממשי	75
פירוק מספר דו-ספרתי לספרותיו	77

80	4.3. הטיפוס התווי .....
84	המרה מתו המייצג ספרה לערך מספרי מתאים .....
85	4.4. בחירה אקראית .....
88	סיכום .....
89	סיכום מרכיבי שפת C# שנלמדו בפרק 4 .....
90	שאלות נוספות .....
92	תבניות – פרק 4 .....
	חלוקת כמות פריטים לקבוצות בגודל נתון, פירוק מספר חיובי לספרותיו, בניית מספר
95	<b>פרק 5 – ביצוע מותנה .....</b>
95	5.1. הוראה לביצוע-בתנאי .....
95	הוראה לביצוע-בתנאי במבנה אם... אחרת... ..
101	הוראה לביצוע-בתנאי במבנה אם... ..
106	התניית ביצוע של שתי הוראות או יותר .....
108	ביטויים בוליאניים הכוללים תווים .....
111	5.2. תנאי מורכב .....
112	הקשר /אם... ..
116	הקשר /א... ..
122	תנאים מורכבים מעורבים .....
122	5.3. קינון של הוראה לביצוע-בתנאי .....
129	5.4. הוראת שרשרת לביצוע-בתנאי .....
133	5.5. הוראת בחירה .....
138	סיכום .....
139	סיכום מרכיבי שפת C# שנלמדו בפרק 5 .....
140	שאלות נוספות .....
143	תבניות – פרק 5 .....
	מציאת מקסימום ומינימום בסדרה, סידור ערכים בסדרה, ערכים עוקבים, זוגיות מספר, מחלק של מספר
147	<b>פרק 6 – נכונות אלגוריתמים .....</b>
154	סיכום .....
155	<b>פרק 7 – ביצוע-חוזר .....</b>
155	7.1. ביצוע-חוזר מספר פעמים ידוע מראש .....
169	7.2. מציאת מקסימום או מינימום .....
172	7.3. מציאת ערך נלווה למקסימום או למינימום .....
174	7.4. ביצוע-חוזר-בתנאי .....
174	ביצוע-חוזר בשימוש בזקיף .....
180	ביצוע-חוזר עם תנאי כניסה כלשהו .....

187	ביצוע-חוזר אינסופי
191	7.5. משתנים מטיפוס בוליאני
196	7.6. הקשר הלוגי (not)
198	7.7. קינון הוראות לביצוע-חוזר
202	סיכום
204	סיכום מרכיבי שפת C# שנלמדו בפרק 7
205	תבניות – פרק 7
	מנייה וצבירה, ממוצע של סדרת מספרים, מציאת מקסימום או מינימום בסדרה, מציאת ערך נלווה למקסימום או למינימום בסדרה, איסוף בקיזוז, פירוק מספר חיובי לספרותיו, בניית מספר, האם כל הערכים בסדרה מקיימים תנאי?, האם קיים ערך בסדרה המקיים תנאי?, מציאת כל הערכים בסדרה המקיימים תנאי, מעבר על זוגות סמוכים בסדרה
213	פרק 8 – יעילות של אלגוריתמים
222	סיכום
223	אינדקס

## תוכן יסודות 2

פרק 9 – המחלקה מחרוזת (String)

פרק 10 – מערכים

פרק 11 – מחלקות ועצמים: הרחבה והעמקה

פרק 12 – תבניות אלגוריתמיות (מערך דו-ממדי, מיון, חיפוש ומיזוג)

פרק 13 – פתרון בעיות

## פתח דבר למורה

ספר זה פותח על בסיס ספר הלימוד "יסודות מדעי המחשב" שפותח במכון ויצמן למדע בסוף שנות ה-90, אך הוא שונה ממנו בכמה אספקטים. ראשית, בספר הקודם אלגוריתמים יושמו בשפה הפרוצדורלית Pascal, בעוד שבספר זה הם מיושמים בשפת C#, שפה מונחית עצמים. עם זאת, העקרונות האלגוריתמיים והנושאים בשמונת הפרקים הראשונים בספר זה זהים לאלה שפותחו בידי מכון ויצמן. כלומר, גם הספר הזה שם דגש על פתרון בעיות אלגוריתמיות ומתמקד בחשיבה אלגוריתמית. אמנם אין ניסיון להסתיר את מאפייניה הייחודיים של שפת C#, כשפה מונחית עצמים, וכך המילה מחלקה והשימוש במחלקות-קיימות מופיע החל מפרק 3 ובו מוצגות תוכניות ראשונות. לאורך כל פרקי הספר של "יסודות 1" כתובות תוכניות המכילות רק מחלקה אחת, והיא מכילה פעולה אחת – הפעולה הראשית. בספר "יסודות 2" מורחב המבט על עצמים ועל אלגוריתמים החל מפרק 9 הדין במחרוזות, דרך פרק 11 המהווה הרחבה והעמקה בנושא העצמים, וכלה בפרק 13 המציג פתרון תרגילי בגרות ברוח התכנות מונחה העצמים.

בנוסף, לאורך הספר משולבת התייחסות מפורשת לתבניות בפיתוח אלגוריתמים ובניתוחם. בסופם של כמה מפרקי הספר יש סעיף העוסק בתבניות הרלבנטיות לאותו פרק. סעיפים אלה מציגים את התבניות השונות ומפנים לאתר הספר הכולל הסבר מפורט ושאלות נוספות. אמנם לכאורה יש הפרדה בין החומר השוטף של הפרק לדפי התבניות, אך הכוונה היא שהם יילמדו באופן משולב: מתוך הפרק עצמו, יש הפניות אל דפי התבניות, בכל פעם שבעיה או שאלה מציפה תבנית מסוימת. ההתייחסות לתבניות מבוססת על הספר "תבניות במדעי המחשב" שפותח בשנת 2001 בידי חברי קבוצת הוראת מדעי המחשב בחוג להוראת המדעים באוניברסיטת תל-אביב.

את הספר מלווה אתר ובו מדריך מעבדה מקוון וקבצי התכניות המופיעות בספר לנוחיותם של תלמידים ושל מורים. כמו כן, בסוף הספר הוספנו אינדקס מפורט, אשר באמצעותו תלמיד יכול לנווט הלוך וחזור בתוך הספרים, ולמצוא חומרים במהירות לפי מילות מפתח.



# פרק 1 – מבוא

בפרק מבוא קצר זה נסביר מעט על המחשב, על תפקידיו, על מבנהו ועל אופן השימוש בו לצורך פתרון בעיות מסוגים שונים. ייתכן כי לאלה מביניכם הרגילים בשימוש במחשב, ואולי אף בתכנות, חלק מהמושגים יהיו מוכרים. בכל זאת, סביר שקריאת הפרק תאיר כמה מהמושגים האלה ואת הקשרים ביניהם באור מעט שונה, וסביר שכמה מהמושגים המוצגים בפרק יהיו חדשים גם עבור התלמידים שרגילים בשימוש במחשב.

מחשבים מצויים במקומות רבים: הם מנחים מטוסים, אוניות וספינות חלל; הם מפקחים על מיליוני טלפונים המחוברים ברשת ופזורים על פני מדינות שונות; הם משמשים לחיזוי מזג האוויר, לכתבה ולהדפסה של מסמכים, לבקרה על מערכות ייצור אוטומטיות, להלחנת מוסיקה, למשחקים ולדברים רבים נוספים.

ללא המחשב לא היה מתאפשר מבצע הנחתת אדם על הירח. המבצע דרש פתרון בעיות חישוביות קשות ומסובכות לקביעת מסלול חללית בהשפעת הירח, כדור הארץ והשמש. ביצוע החישובים האלה ללא מחשב היה מצריך עבודת צוות גדולה במשך עשרות שנים. בנוסף, כל שינוי במועד ההמראה או במסלול הטיסה חייב פתרון מחדש של בעיות חישוביות אלו. המבצע לא היה מתאפשר ללא מחשב.

## 1.1 מהו מחשב?

**מחשב** (computer) הוא מכונה אלקטרונית הקולטת נתונים, מעבדת אותם ופולטת מידע הנוצר בתהליך העיבוד. הנתונים שקולט המחשב נקראים **קלט** (input); המידע שפולט המחשב נקרא **פלט** (output); העיבוד המבוצע במחשב מונחה על ידי אוסף הוראות הנקרא **תוכנית מחשב** (computer program).

הנה כמה דוגמאות להמחשת ההגדרות האלו:

- ◆ כשאנו מגיעים לקנות כרטיסים לסרט, הקופאי מקיש בלוח המקשים של המחשב שעל שולחנו את מספר הכרטיסים שאנו מבקשים לקנות. המחשב מעבד נתון זה בבדיקת המקומות הפנויים, בהקצאת מקומות כמבוקש ובסימון המקומות שהוקצו כתפוסים. המדפסת מדפיסה כרטיסים שעליהם מסומנים המקומות שהוקצו. במקרה זה הקלט הוא מספר הכרטיסים המבוקש, והפלט הוא הכרטיסים שעליהם מקומות מסומנים.
- ◆ אמן אנימציה מציין כקלט על צג מחשב שתי נקודות לתנועה של דמות – נקודת התחלה ונקודת סיום. המחשב מעבד נתונים אלה יחד עם נתונים נוספים על הדמות, ומציג על הצג כפלט את תנועת הדמות מנקודת ההתחלה לנקודת הסיום.
- ◆ מנהל חשבונות מקיש כקלט שם של עובד ואת מספר הימים שעבד בחודש האחרון. המחשב מעבד נתונים אלה יחד עם נתונים אחרים השמורים בו (כמו דרגת העובד), ומדפיס כפלט את המשכורת שמגיעה לעובד עבור החודש האחרון.
- ◆ מדען, המשתמש במחשב לצורך חישוב מסלול התעופה של טיל, נותן כקלט למחשב את נקודת שיגור הטיל, את זמן השיגור, את מהירות הטיל ונתונים על הרוחות במסלול מעופו הצפוי של הטיל. המחשב מעבד נתונים אלה בעזרת נוסחאות לחישוב תעופת טיל ובעזרת מידע השמור בו על כדור הארץ, ונותן כפלט את המקום שאמור הטיל לנחות בו.

## שאלה 1.1

הביאו דוגמאות מסביבת הבית ובית הספר לשימוש במחשב, וציינו עבור כל דוגמה את הקלט ואת הפלט.

ייחודה של המכונה "מחשב" לעומת מכונות אחרות הוא במגוון השימושים הרחב שלה. בעוד שבמכונת כביסה אנו משתמשים כדי לכבס, במקרר כדי לשמור על מזון ובמכונת כדי להגיע ממקום למקום, הרי שבמחשב משתמשים בקשת רחבה של משימות. זה נובע מכך שהמחשב מבצע פעילויות שונות של עיבוד ואחסון מידע, פעילויות המונחות כולן על ידי תוכניות מחשב מתאימות.

אמנם, קיימים כיום מחשבים אשר מסוגלים לבצע בצורה מוגבלת תפקודים אנושיים, כמו ראייה, שמיעה, דיבור ותנועה, אבל המחשב אינו כל יכול. הוא בסך הכול מכונה. הוא איננו יכול למשל לדמיין, לכאוב או לשמוח. למעשה, גם אם נגביל את הדיון למשימות שאינן מערבות רגשות, ישנן משימות רבות שאינן ניתנות לביצוע על ידי מחשב.

יתרון בולט של מחשב הוא בכך שהוא מבצע פעולות חישוב ועיבוד מידע במהירות עצומה ובדיוק רב. למשל, מחשב יכול לקלוט אלף מספרים בני חמש ספרות כל אחד, ולחשב תוך שבריר שנייה את סכומם. מחשב יכול לקלוט טקסט בן אלף מילים, ולחשב תוך שבריר שנייה את האות השכיחה ביותר בטקסט.

יתרון חשוב נוסף של מחשב הוא שניתן לאחסן בזיכרוננו מיליוני נתונים. למשל, במחשב של משרד הפנים מאוחסנים נתונים על כל אחד מתושבי המדינה (שם, מספר זהות, תאריך לידה, מצב משפחתי ועוד). במחשב של בנק מאוחסנים פרטים אישיים על כל לקוח של הבנק, והנתונים על התנועות בכל חשבונות הבנק.

אמנם המחשב מבצע פעולות חישוב ועיבוד בדיוק רב, אך מאחר שהוא לא יותר ממכונה, המבצעת הוראות, לא מובטח שתמיד ייתן המחשב כפלט תוצאת עיבוד נכונה. פלט לא נכון יכול להתקבל כתוצאה מקלט שגוי או מתוכנית מחשב שגויה.

**?** מהי תוכנית מחשב שגויה?

תוכנית מחשב נכתבת על ידי בני אדם. ייתכן שההוראות הכלולות בה אינן מורות על העיבוד הדרוש. במקרה כזה התוכנית שגויה, וייתכן כי כאשר המחשב פועל על פיה הוא ייתן כפלט תוצאת עיבוד לא נכונה. למשל, ייתכן כי מתכנת יכתוב תוכנית לחישוב ממוצע של אלף מספרים, ובה יורה על סיכום המספרים אך יטעה וישכח להורות על חלוקת הסכום המחושב ב-1000. ברור כי עיבוד שיתבצע על פי תוכנית זו לא יבצע חישוב ממוצע כנדרש וייתן פלט שגוי.

אנו מבחינים בין שני צדדים של מחשב: חומרה ותוכנה.

**חומרה (hardware)** היא הרכיבים הפיסיים שמרכיבים את המחשב.

**תוכנה (software)** היא אוסף תוכניות המחשב.

## 1.2 חומרה

בסעיף זה נתאר על קצה המזלג את הרכיבים הפיסיים של המחשב. במהלך לימוד היחידה "יסודות מדעי המחשב" תעבדו בוודאי על מחשבים אישיים (personal computer – pc). מחשבים אישיים הם קטנים יחסית בממדיהם ומיועדים לשרת בו זמנית משתמש אחד בלבד.

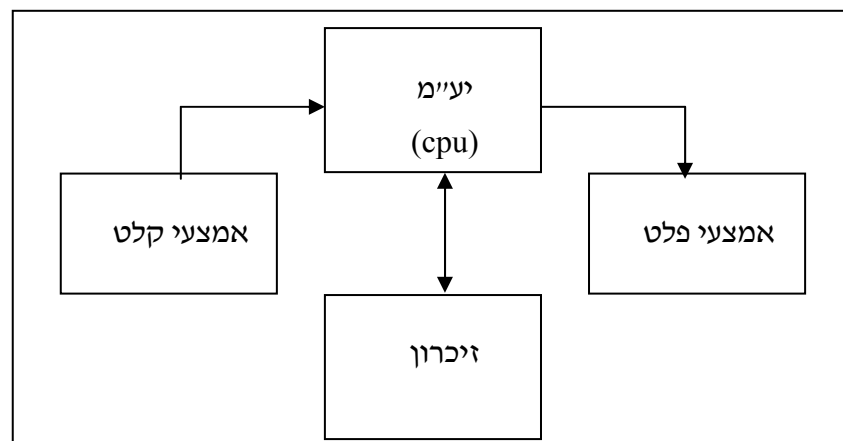
קיימים גם מחשבים גדולים יותר, שיכולים לשרת בו זמנית משתמשים רבים. עם זאת, המבנה הכללי של מחשב אינו תלוי בגודלו.

למחשב כמה יחידות בסיסיות :

- ♦ **יחידת עיבוד מרכזית** (central processing unit), ובקיצור **יע"מ** (CPU) היא היחידה שאחראית על עיבוד מידע, מבצעת חישובים ומנהלת את כל התהליכים המתבצעים במחשב.
- ♦ **זיכרון** (memory) נשמרים תוכניות המחשב, המידע שבו משתמש המחשב ותוצאות ביניים של תהליכי עיבוד.
- ♦ **דרך אמצעי קלט** (input devices) מתבצעת קליטת נתונים.
- ♦ **דרך אמצעי פלט** (output devices) ניתן הפלט של תוצאות העיבוד.

איור 1.1 מתאר את היחידות הבסיסיות של מחשב :

**מחשב**



איור 1.1 – יחידות הבסיסיות של מחשב

נרחיב מעט על היחידות הבסיסיות :

**יחידת העיבוד המרכזית** היא ה"מוח של המחשב". היא מבצעת את ההוראות הכתובות בתוכניות המחשב, ביניהן הוראות לביצוע פעולות חישוב, פעולות השוואה ועוד. במהלך ביצוע פעולותיה פונה יחידת העיבוד המרכזית אל הזיכרון ואל אמצעי הקלט והפלט.

**הזיכרון** מורכב מאוסף גדול מאוד של תאים. לכל תא בזיכרון מותאם מספר סידורי, הנקרא מען (כתובת, address). יחידת העיבוד המרכזית משתמשת במידע השמור בזיכרון ולעתים משנה אותו, תוך פנייה לתאי הזיכרון על ידי המען שלהם. יש שני סוגי פניות לזיכרון: כתיבת מידע בזיכרון וקריאת מידע מהזיכרון. תהליך הכתיבה בזיכרון גורם למחיקת מידע ולשמירת מידע חדש במקומו. לעומתו, תהליך הקריאה מהזיכרון גורם להעברת המידע מהזיכרון אל יחידת העיבוד המרכזית, אך איננו גורם למחיקתו. הוא מזכיר את תהליך הקריאה שאנו מבצעים: כאשר אנו קוראים ספר, השורות אינן נעלמות מדפי הספר, אלא רק "מועתקות" לזיכרוננו.

המידע השמור בזיכרון מיוצג על ידי סיביות. **סיבית** (bit) – קיצור של **ספרה בינארית** (binary digit) – היא אחת מן הספרות 0 או 1, כמו שספרה עשרונית היא אחת מן הספרות 0, ..., 9.

תא בזיכרון המחשב מכיל סדרה של סיביות, בדרך כלל 16, 32 או 64 סיביות, על פי תכנון החומרה של המחשב. מפתיע ומרשים שכל המשימות המורכבות המתבצעות על ידי מחשב הן בסופו של דבר אוסף של פעולות על סדרות המורכבות מהספרות 0 ו-1! המחשב מפרש סדרות של סיביות כמספרים או כאותיות, לעתים סדרות של סיביות מפורשות כהוראות לביצוע או כטיפוסי מידע אחרים.

למעשה, יחידת העיבוד המרכזית, ה"מוח" של המחשב, מבצעת בסך הכול הוראות פשוטות כגון "קרא את סדרות הסיביות שנמצאות בתאי הזיכרון שמעניהם הם 13 ו-37, התייחס לכל סדרת סיביות כאל מספר, חבר אותם וכתוב את התוצאה בתא שמענו 116".

זיכרון המחשב מורכב למעשה משני חלקים נפרדים:

**זיכרון ראשי (main memory)** משמש לשמירת תוכניות בזמן ביצוען, ולשמירת נתונים ותוצאות ביניים של תוכניות שמתבצעות.

**זיכרון משני (secondary memory)** משמש לאחסון לזמן בלתי מוגבל של מידע ושל תוכניות מחשב.

הזיכרון הראשי קטן משמעותית מן הזיכרון המשני. מהירות הקריאה ממנו והכתיבה אליו גדולה הרבה יותר ממהירותן של פעולות אלו בזיכרון המשני. הזיכרון הראשי ממוקם צמוד ליחידת העיבוד המרכזית, ואילו הזיכרון המשני נמצא באמצעי אחסון חיצוניים כמו דיסק ודיסק קשיח. במהלך ביצועה של תוכנית מחשב, יחידת העיבוד המרכזית מעתיקה את התוכנית מהזיכרון המשני אל הזיכרון הראשי. במובנים רבים, ניתן להתייחס לזיכרון הראשי כאל זיכרון לטווח קצר, ואל הזיכרון המשני כאל זיכרון לטווח ארוך.

## שאלה 1.2

הביאו דוגמה מכיתת בית הספר לחפץ שניתן לדמות את השימוש בו לשימוש בזיכרון ראשי, ולעומתה דוגמה לחפץ שניתן לדמות את השימוש בו לשימוש בזיכרון משני.

**אמצעי קלט** משמשים להעברת נתונים אל המחשב מן המשתמשים בו. למשל, בדרך כלל מחוברים למחשב אישי עכבר ולוח מקשים. שניהם אמצעי קלט המשמשים להעברת נתונים. גם סורק דיגיטלי, מצלמה דיגיטלית או מיקרופון מחוברים למחשב הם אמצעי קלט.

**אמצעי פלט** משמשים להעברת מידע מן המחשב אל המשתמשים בו. למשל, בדרך כלל מחוברים למחשב אישי מסך ומדפסת. גם מקרן או רמקול מחוברים למחשב הם אמצעי פלט.

## שאלה 1.3

מחשבון נועד לבצע פעולות חשבון. מהו אמצעי הקלט למחשבון? מהו אמצעי הפלט למחשבון?

## שאלה 1.4

המחשב קולט מידע, מעבד אותו ונותן כפלט את תוצאת העיבוד. גם מוח האדם קולט מידע, מעבד אותו ופולט את תוצאת העיבוד.

א. הביאו דוגמאות לאיברים קולטי מידע ולאיברים פולטי מידע בגוף האדם.

ב. הביאו דוגמאות למידע השמור בזיכרון האדם.

## 1.3 תוכנה

תוכנה היא אוסף תוכניות מחשב. תוכניות מחשב מנחות את העיבוד המתבצע על ידי החומרה. קיימות תוכניות לביצוע חישובים מתמטיים, לניהול מאגרי מידע, לבקרה על תהליכים, להדמיית מערכות, לעיבוד תמלילים, למשחקים וליישומים שונים ורבים נוספים.

בנוסף לתוכניות המיועדות ליישומים שונים, יש בכל מחשב תוכנית מיוחדת הנקראת מערכת הפעלה, והיא מהווה את הקשר בין התוכנה לחומרה:

**מערכת הפעלה** היא תוכנית המנהלת את שאר התוכניות ומקצה לשימושן את משאבי החומרה השונים (יע"מ, זיכרון, אמצעי קלט ואמצעי פלט).

כל תוכנית מחשב (או בקיצור, תוכנית) נכתבת על ידי מתכנת בשפת תכנות. נסביר את שני המושגים האלה:

**שפת תכנות** (programming language) היא למעשה אוסף של כל הכללים הקובעים כיצד נכתבות ההוראות בתוכנית מחשב, ומה המשמעות של כל הוראה.

**מתכנת** (programmer) הוא אדם הכותב תוכניות בשפת מחשב. עבודתו של מתכנת – תהליך התכנות – כולל ניתוח של משימות המיועדות לביצוע במחשב, כתיבת מתכון לביצוע המשימה, ויישומו של המתכון בשפת מחשב.

כזכור, הפעולות המתבצעות ביחידת העיבוד המרכזית הן פעולות על סיביות. לכן, בעצם, השפה שבאמצעותה ניתן לתקשר עם מחשב, או השפה שבה ניתן לתת לו הוראות שיוכל "להבין", צריכה להיות שפה מאוד פשוטה, הכוללת הוראות לביצוע פעולות על סיביות. שפה כזאת נקראת שפת מכונה:

**שפת מכונה** (machine language) היא שפת תכנות הכוללת הוראות לביצוע פעולות פשוטות מאוד. כל הוראה בשפת מכונה מורה על ביצוע פעולות על סדרות של סיביות (למשל, חיבור שתי סדרות). למעשה, גם ההוראות של שפת מכונה נכתבות בקוד המבוסס על סיביות, ולכן תוכנית בשפת מכונה היא בעצם סדרה ארוכה של סיביות, כלומר, רצף ארוך של 0 ו-1.

לכל סוג מחשב שפת מכונה משלו.

התוכניות הראשונות שנכתבו עבור מחשבים (בסוף שנות ה-40 של המאה העשרים) נכתבו בשפת מכונה. תהליך הכתיבה של תוכניות אלו היה מסורבל מאוד ולא נוח, בגלל החסרונות הרבים שיש לכתיבה בשפת מכונה.

מהם החסרונות של כתיבה בשפת מכונה ?

♦ מאחר שתוכנית בשפת מכונה היא רצף ארוך של 0 ו-1, קשה מאוד לכתוב אותה וקשה עוד יותר לקרוא אותה, לעקוב אחר מהלך ביצועה ולהבין את מטרתה. חשוב להבין שתהליך התכנות לא מסתיים בדרך כלל עם כתיבת התוכנית: לעתים מתגלות שגיאות בתוכנית וצריך לתקנה, לפעמים צריך לעדכן אותה כדי להתאימה לדרישות חדשות של המשימה שהיא

מבצעת. לא תמיד התיקונים והעדכונים מתבצעים על ידי הכותב המקורי, ולכן לנוחות הקריאה וההבנה של תוכנית נתונה יש חשיבות רבה.

♦ לכל סוג מחשב יש שפת מכונה שמתאימה בדרך כלל רק לו. לכן לא ניתן לקחת תוכנית שנכתבה בשפת מכונה של מחשב מסוג אחד, ולהריץ אותה כמו שהיא במחשב מסוג אחר. מעבר בין סוגי מחשבים דורש כתיבה מחודשת של התוכנית.

חסרונות אלה הביאו לפיתוחן של שפות נוחות יותר לכתיבה, לקריאה ולשימוש. שפות כאלו פותחו החל מאמצע שנות ה-50 של המאה העשרים, והן נקראות שפות עיליות. ביניהן למשל השפות פסקל (pascal), ג'אווה (Java), ו-C.

**שפה עילית** (high level language) היא שפת תכנות, אשר ההוראות בה דומות למשפטים בשפה טבעית (כמו אנגלית) או לנוסחאות מתמטיות. למרות הדמיון לשפה טבעית, ההוראות אינן נכתבות בכתיבה חופשית, אלא על פי כללים מוגדרים, שנקראים **כללי התחביר** (syntax) של השפה.

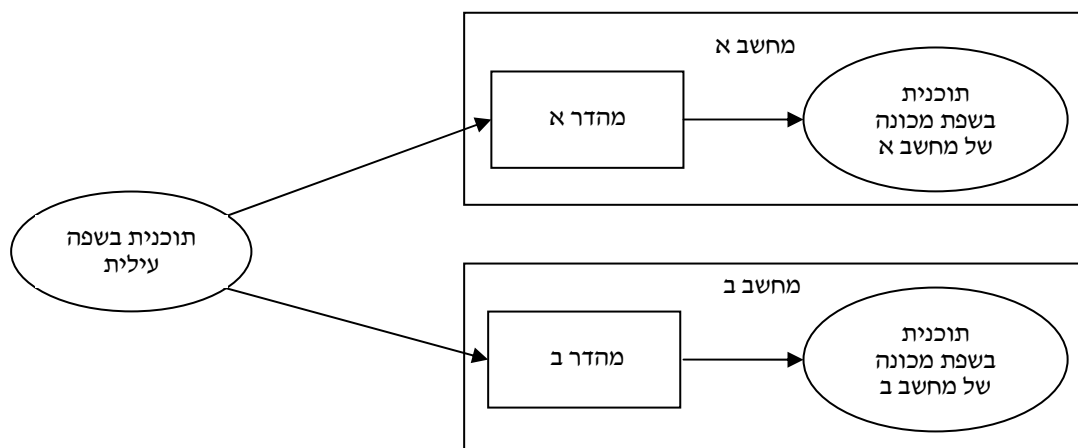
? אם המחשב "מבין" רק שפת מכונה, כיצד ניתן לגרום לו "להבין" תוכנית הכתובה בשפה עילית?

תוכנית הכתובה בשפה עילית עוברת תהליך של תרגום לשפת מכונה. התרגום נעשה על ידי תוכנית מחשב מיוחדת, הנקראת מהדר:

**מהדר** (compiler) היא תוכנית המתרגמת משפה עילית לשפת מכונה. תהליך התרגום נקרא **הידור** או **קומפילציה** (compilation). הקלט של המהדר הוא תוכנית בשפה עילית והפלט שלו הוא תוכנית בשפת מכונה, שהיא התרגום של תוכנית הקלט.

אם כך, כדי לבצע תוכנית מחשב הכתובה בשפה עילית צריכים להתבצע שני שלבים. קודם כל מתבצע שלב ההידור, במהלכו התוכנית בשפה העילית עוברת תרגום לשפת מכונה. רק אחר כך יכול להתבצע שלב ההרצה, במהלכו מתבצעת המשימה עצמה, כלומר, המחשב מבצע את ההוראות הכתובות בשפת מכונה, ונותן את תוצאת העיבוד כפלט.

לכל זוג של שפה עילית ושפת מכונה דרוש מהדר נפרד שיבצע את התרגום ביניהן. אם בכוונתנו להריץ במחשב מסוים תוכניות הכתובות בכמה שפות עיליות, עלינו לדאוג שבמחשב יהיה מותקן מהדר מתאים לכל אחת מהשפות העיליות האלו. גם ההיפך נכון: אם בכוונתנו להריץ תוכניות בשפה עילית מסוימת בכמה מחשבים מסוגים שונים, עלינו לדאוג שיהיו ברשותנו מהדר עבור כל סוג מחשב. איור 1.2 מדגים את הקשרים האלה.



איור 1.2 – הידור של תוכנית בשפה עילית במחשבים מסוגים שונים

אם כך, שפה עילית איננה רק יותר נוחה לקריאה ולכתיבה, אלא היא גם מגשרת על פני ההבדלים בין סוגים שונים של מחשבים. בעזרת מהדר מתאים ניתן לתרגם כל תוכנית בשפה עילית לתוכנית בשפת מכונה של מחשב זה או אחר.

תהליך ההידור מורכב משני שלבים :

1. בדיקת תחביר התוכנית בשפה העילית
2. תרגום התוכנית בשפה העילית לתוכנית בשפת מכונה.

בשלב 1, נערכת בדיקה כי התוכנית בשפה העילית עומדת בכללי התחביר של השפה שבה נכתבה. אם כתיבת התוכנית לא נעשתה בהתאם לכללי התחביר, יש בה **שגיאות תחביר** (syntax errors). הפלט של המהדר אחרי שלב 1 הוא פירוט שגיאות התחביר שמצא. לפני שניתן יהיה לתרגם את התוכנית יש לתקן את כל שגיאות התחביר שבה, כלומר, לעבור בהצלחה את שלב 1. למשל, תוכנית בשפת C# חייבת להכיל בתוכה לפחות פעם אחת את המילה class. אם ננסה לתרגם תוכנית בשפת C# שאינה מכילה את המילה class המהדר יודיע על שגיאת תחביר.

כאשר שלב 1 מסתיים בהצלחה, ואין בתוכנית שגיאות תחביר, מתבצע השלב השני ובו התוכנית מיתרגמת לשפת מכונה, ומתקבלת תוכנית שיכולה לרוץ במחשב.

גם במהלך הריצה של התוכנית עלולות להתגלות שגיאות שיגרמו לעצירת הריצה לפני סיומה המיועד, או להודעות שגיאה. אלו הן **שגיאות ריצה** (run-time errors), שאינן יכולות להתגלות בזמן ההידור. למשל, הניסיון לחלק ערך השמור בזיכרון המחשב ב-0 יגרום לשגיאת ריצה ולהדפסת הודעה מתאימה.

תהליך איתור שגיאות ריצה ותיקונן נקרא **ניפוי** (debugging). מקורו של המונח באנגלית בשלבים המוקדמים של שימוש במחשבים, כאשר מחשבים היו כה גדולים עד כי מחשב אחד מילא אולם שלם. מחשב מסוים חדל לפעול, ולאחר זמן התגלה בין רכיביו חרק גדול שנתקע שם והפריע לפעולתו התקינה של המחשב. מאז נוהגים לקרוא לשגיאה בתוכנית באג (bug – חרק באנגלית).

מאז תחילת פיתוח השפות העיליות, באמצע שנות ה-50 של המאה העשרים, פותח מספר גדול מאוד של שפות. עובדה זו מעוררת את השאלות הבאות:

- ◆ מדוע יש צורך בכל כך הרבה שפות?
- ◆ האם לא עדיפה שפה אחת אחידה שבה יכתבו כל התוכניות, ואשר אותה יוכל כל אחד ללמוד בקלות?
- ◆ מה מבדיל בין השפות השונות?
- ◆ מי משתמש באילו שפות ולאילו מטרות?

לריבוי השפות שתי סיבות עיקריות. סיבה אחת קשורה להתפתחות שפות תכנות כתגובה לצרכים המתעוררים בשטחים חדשים ושונים של יישומים. לכל שפה מאפיינים ייחודיים משלה: יש שפות המיועדות בעיקר לחישובים מדעיים, יש אחרות המתאימות יותר לעיבוד נתונים מנהלי (הפקת משכורות, הנהלת חשבונות וכו'). הסיבה השנייה היא התקדמות המחקר המדעי העוסק בשפות תכנות ומסייע בשיפור השפות.

את שפות התכנות ניתן לחלק לקבוצות על פי העקרונות המנחים את הכתיבה בשפות אלו. בספר זה ללימוד היחידה "יסודות מדעי המחשב" נשתמש בשפת C#, השייכת לקבוצת השפות הקרויות **מונחות עצמים** (object oriented). קבוצות אחרות, שאליהן לא נתייחס ביחידה זו, הן קבוצת השפות **הפּרוּצדורליות** (כמו פסקל או C), קבוצת השפות **הפונקציונליות** (כמו scheme) וקבוצת השפות **הלוגיות** (כמו פרולוג).

## 1.4 התפתחות המחשבים ומדעי המחשב

ההתפתחות הקשורה למחשבים נעשתה בשני מסלולים: ההתפתחות ההנדסית והטכנולוגית שאפשרה בניית מחשבים משוכללים יותר ויותר, וההתפתחות המדעית שניסתה להתמודד בצורה מדויקת, אפילו פורמלית לעתים, עם שאלות הקשורות לפתרון בעיות באמצעות מחשב. בסעיף זה ננסה לתת סקירה קצרה של שני מסלולי ההתפתחות, שכמובן אינם מנותקים זה מזה.

### התפתחות הנדסית וטכנולוגית – חומרה

ציון דרך חשוב בהתפתחות ההנדסית הוא באמצע המאה ה-17. אז פיתח המתמטיקאי הצרפתי בליז פסקל (Pascal) – על שמו נקראת שפת התכנות פסקל – מכונת חיבור וחסור. את המכונה בנה עבור אביו, כדי לסכם סכומי כסף לצורך גביית מסים. פסקל בנה את המכונה כדי לעזור לאנשים שביצעו בדרך כלל את החישובים הדרושים, ובמיוחד כדי להקטין את מספר הטעויות שנגרמו על ידי החישובים האנושיים. המכונה של פסקל מהווה ציון דרך חשוב משום שזו הייתה הפעם הראשונה ששולב מרכיב אוטומטי, באמצעות מכונה, בפעולת חישוב.

המכונה של פסקל זכתה לשיפורים כמה עשרות שנים מאוחר יותר. המדען הגרמני וילהלם לייבניץ (Leibnitz) בנה אף הוא מכונת חישוב. הוא העתיק את מנגנוני החיבור והחסור של מכונתו מהמכונה של פסקל, אך הוסיף גם חלק שמבצע כפל וחילוק. לייבניץ בנה את המכונה שלו משום שלטענתו המדע אמנם לא יכול להתקיים ללא חישוב, אך חישוב הוא פעולה חוזרת על עצמה, משעממת ולא יצירתית, וצריך להעביר את ביצועה למכונות.

בתחילת המאה ה-19, ב-1801, פיתח גם הצרפתי ז'וזף ז'אקאר (Jacquard) מכונה לביצוע אוטומטי של משימות. בניגוד למכונות של פסקל ולייבניץ אלו לא היו משימות חישוב מספריות, אלא משימות אריגה. מכונתו של ז'אקאר היתה למעשה נול אריגה מתוחכם, שיכול היה לארוג במגוון של דוגמאות. הדוגמאות השונות תוארו על ידי כרטיסים מנוקבים, לכל דוגמת אריגה תבנית ניקובים משלה. מנגנון בקרה מיוחד בתוך המכונה חש את הנקבים בכרטיס, ובהתאם לכך פיקח על פעולות המכונה, כגון בחירת חוטים.

התכנון של מה שנחשב היום המחשב הראשון הגיע כשלושים שנה מאוחר יותר. ב-1833, תוכננה לראשונה מכונה כללית יותר, שיכולה לבצע משימות מסוגים שונים. המתמטיקאי האנגלי צ'רלס בבג' (Babbage) תכנן את "המכונה האנליטית" שלו, מכונה שהיתה אמורה לבצע תוכניות שונות מסוגים שונים למטרות שונות. התוכניות היו אמורות להיות מקודדות, בדומה למכונה של ז'אקאר, בעזרת כרטיסים מנוקבים. התכנון של בבג', שכלל צירים, ידיות, גלגלי שיניים ורכיבים מכניים אחרים, לא מומש אף פעם. בניית חלקי המכונה דרשה דיוק טכני רב מדי, שלא ניתן היה להשגה באותו זמן. אבל, אף על פי שלא נבנתה, המכונה האנליטית היא בעלת חשיבות גדולה. למעשה, הרעיונות הגלומים בה הם הבסיס למבנה המחשבים של ימינו ולאופן פעולתם. אפילו בעצם נכתבה אז תוכנית עבור המכונה הלא-בנויה, תוכנית הראויה בהחלט להיחשב כתוכנית המחשב הראשונה בהיסטוריה! את התוכנית כתבה עדה לאבלייס (Lovelace) – על שמה נקראת שפת Ada.

לאחר מותו של בבג' מרכז הפעילות בבניית מכונות חישוב נדד לארצות הברית. העיסוק המוגבר בכך בארצות הברית נבע בין השאר מהצורך שהתעורר מהשטח: ב-1880 נערך בארצות הברית מפקד אוכלוסין, המפקד הבא נועד ל-1890, וב-1886, ארבע שנים לפני המפקד הבא, ושש שנים אחרי המפקד הקודם, עדיין לא סיימו לסכם את תוצאותיו, ומועד סיום הסיכום לא נראה באופק... הרמן הולרית (Hollerith), מהנדס שעבד כפקיד בלשכת מפקד התושבים, גילה יוזמה וב-



1886 הוא הציע כמה מכשירים שפיתח כדי לפתור את הבעיה. סיכום המפקד של 1890 כבר נעשה בעזרת פיתוחו של הולריק וארץ לא יותר מחודש! גם פיתוחו של הולריק היה מבוסס על כרטיסים מנוקבים. בעקבות הצלחתו ראה הולריק כי טוב והקים חברה למכונות חישוב. מאוחר יותר, ב-1928, הרחיבה אותה חברה את פעילותה והפכה לחברה בינלאומית למכונות עסקיות, ושינתה את שמה בהתאם ל-International Business Machines, או בקיצור, IBM, המוכרת לנו היטב גם היום.

בשנת 1937 החל המדען האמריקני הווארד אייקן (Aiken), יחד עם חברת IBM, לבנות את המחשב האלקטרו-מכני הראשון. למעשה, אייקן הגשים את חלומו של בבג: התכנון שלו התבסס על רעיונותיו של בבג, ונעזר במכשירים החשמליים והאלקטרו-מכניים אשר בתקופתו של אייקן כבר היו זמינים. בניית המחשב הושלמה ב-1944. שמו היה MARK I וגודלו היה כגודל אולם התעמלות! למעשה, במקביל לבניית MARK I, במהלך מלחמת העולם השנייה, בנו גם הבריטים מחשב, בשם אניגמה (Enigma), שבעזרתו פיצחו צפנים של הגרמנים. אלא שעקב תפקידו הרגיש נשמר קיומו של Enigma בסוד במשך זמן רב. זמן קצר אחר-כך, ב-1946, כבר הושלמה בנייתו של מחשב מהיר בהרבה מ-MARK I. הוא נקרא ENIAC, ותוכן על ידי האמריקנים אקרט (Eckert) ומוצ'לי (Mauchly). הוא היה הרבה יותר מהיר משום שלא התבסס בכלל על תנועות מכניות, ולכן נחשב המחשב האלקטרוני הראשון. אורכו היה שלושים מטרים, רוחבו מטר אחד, גובהו שלושה מטרים, והוא הכיל 18,000 שפופרות ואקום. צריכת החשמל של ה-ENIAC, שהוצב בעיר פילדלפיה, הייתה כה גבוהה, עד שנהגו אז לומר שבכל פעם שהופעל התעממו האורות בעיר כולה!

אותם מחשבים ראשונים השתמשו בסרטי נייר מנוקבים. עבור כל עיבוד קודדו על סרט כזה גם התוכנית לביצוע וגם נתוני הקלט עבורה. כלומר, אם ביצעו אותה תוכנית כמה פעמים, היה צריך להזין למחשב את סרט הנייר שעליו קודדה בכל פעם ופעם. ב-1946 הציע המדען ההונגרי-אמריקני ג'ון פון-נוימן (von Neuman) לשמור את התוכניות בזיכרון המחשב. כתוצאה מכך, מהירות תהליכי העיבוד השתפרה משמעותית. המחשב התעשייתי הראשון שפעל לפי עקרון זה נבנה ב-1951, ונקרא UNIVAC. עקרון זה של פון-נוימן משמש למעשה גם במחשבים של ימינו.

מכאן החלה האצה בקצב ההתפתחות הטכנולוגית. בשנות ה-50 וה-60 של המאה העשרים הוחלפו שפופרות הריק בטרנזיסטורים, ושינוי זה הביא להקטנה בממדי המחשבים, להקטנה בצריכת ההספק החשמלי שלהם ולהגדלה במהירות פעולתם. בנוסף למחשבים הגדולים (mainframes) שהיו עד אז, נבנו גם מחשבים בינוניים (שגודלם כגודל כונית ספרים) שנקראו מחשבי מידי ומחשבי מיני (midi/mini computers). זמן לא רב אחר-כך קטנו המחשבים אף יותר: בסוף שנות ה-60 ובתחילת שנות ה-70 פותחה טכנולוגיית המעגל המשולב (integrated circuit) ובעקבות כך ירד מחיר המחשבים, מהירותם עלתה ונבנו אפילו מחשבים זעירים, שגודלם לא עלה על גודל קופסת גפרורים – המיקרו-מחשבים (micro computers). משום שהיו כה קטנים וזולים יחסית, החלו להשתמש בהם הרבה לצרכים מגוונים למשל כבקרים במערכות אלקטרוניות שונות. המחשב האישי, המוכר לנו היום, נבנה לראשונה בסוף שנות ה-70 והוא היה מבוסס על מיקרו-מחשב. קפיצת הדרך הזאת, שנעשתה במשך תקופה קצרה יחסית, היא כמעט בלתי נתפסת: היום יש מיקרו מחשבים שכושר העיבוד שלהם עולה בהרבה על זה של אותם מחשבי הענק הראשונים.

## **התפתחות הנדסית וטכנולוגית – תוכנה**

במקביל להתפתחות הטכנולוגית של החומרה, המכונות עצמן, חלה התפתחות גם בתחום התוכנה. המחשבים הראשונים תוכנתו בשפת מכונה. כפי שהזכרנו, כתיבת תוכניות כאלו הייתה כרוכה באי-נוחות רבה. אי-נוחות זו הביאה באמצע שנות ה-50 לפיתוחה של שפת התכנות העילית

הראשונה, פורטרן (Fortran). משום שבאותה תקופה הייתה עלייה בביקוש לתוכניות מחשב המבצעות חישובים מתמטיים, פורטרן הותאמה לכתיבה של חישובים כאלה. אבל היא לא הייתה נוחה לכתיבת תוכניות לניהול מאגרי מידע (ניהול כוח אדם, ניהול מלאי וכו'), ומשום כך פותחה בעקבותיה, בתחילת שנות ה-60, שפת קובול (Cobol). באותה תקופה פותחה שפה נוספת, ליספ (Lisp) שהתאימה לצרכים אחרים. ממנה נגזרה מאוחר יותר שפת לוגו (LOGO) המשמשת בדרך כלל כשפה לימודית לפיתוח הרגלי חשיבה בפתרון בעיות.

מאז פותחו שפות עיליות רבות לצרכים שונים ומגוונים. למשל, שפת בייסיק (Basic) פותחה באמצע שנות ה-60, למטרות לימודיות. פסקל (Pascal) פותחה אף היא כשפה לימודית, בתחילת שנות ה-70, ובאותה תקופה פותחה גם שפת C, שנועדה לכתיבת מערכות הפעלה. ב-1970 פותחה שפת פרולוג (PROLOG) שגישת התכנות בה מבוססת על כללים לוגיים. עם עליית הצורך בכתיבת תוכניות גדולות מאוד ומורכבות מאוד, פותחו בשנות ה-80 שפות שנועדו במיוחד לפיתוח תוכניות גדולות, כגון Modula ו-Ada. בעקבותיהן פותחו שפות נוספות שהתמקדו בפיתוח תוכניות גדולות, והתבססו על מה שקרוי תכנות מונחה-עצמים. בין אלו ניתן למצוא את C++, את SmallTalk, את Ada95, את Eiffel ואת Java.

## התפתחות מדעית

כאמור, במקביל להתפתחות ההנדסית והטכנולוגית, הן בתחום החומרה והן בתחום התוכנה, חלה גם התפתחות מדעית. החל מאמצע שנות ה-30 של המאה העשרים (עוד לפני שנבנה המחשב הראשון!), נעשתה עבודה תיאורטית חשובה, שהניחה את הבסיס לתחום המדעי הקרוי היום מדעי המחשב. עבודה זו נעשתה על ידי מתמטיקאים, שניסו להגדיר בצורה מתמטית מהו תהליך של חישוב, ולנתח בצורה מתמטית, פורמלית ומדויקת, את מגבלותיהם של תהליכי חישוב, ובפרט את מגבלותיהן של מכוונות המבצעות תהליכי חישוב. בין המתמטיקאים האלה ניתן למנות את אלן טיורינג (Turing) האנגלי, שהיה מעורב מאוחר יותר בפרויקט האניגמה, את קורט גדל (Gödel) הגרמני, את אנדריי מרקוב (Markov) הרוסי, ואת האמריקנים אלונזו צ'רץ' (Church), אמיל פוסט (Post) וסטיבן קלין (Kleene). חוקרים אלה ידעו להצביע כבר אז על כך שיש בעיות חישוביות שלא יצליחו לעולם להיפתר על ידי מכונה חישובית, וזאת כאמור עוד לפני שנבנה המחשב הראשון.

מאז חלה התפתחות מדעית עצומה, כאשר לעתים ההתפתחויות הטכנולוגיות הניעו וזירזו התפתחויות מדעיות ולעתים דווקא ההתפתחויות המדעיות גרמו להתפתחות טכנולוגית. כיום קיימת מחלקה למדעי המחשב כמעט בכל מוסד אקדמי, והפעילות המחקרית במדעי המחשב היא רבה ומגוונת: תורת החישוביות העוסקת באפיון של בעיות שניתנות או לא ניתנות לפתרון; תורת הסיבוכיות העוסקת באפיון של בעיות על פי כמות המשאבים (זמן וזיכרון) הנדרשים לפתרון; קריפטוגרפיה העוסקת בהצפנות מסוגים שונים; חישוב מקבילי ומבוזר, העוסק בפתרון בעיות שנועדו להתבצע במערכות שבהן כמה מחשבים עובדים ביחד לפתרון משימה אחת; תורת התקשורת העוסקת באפיונים של רשתות תקשורת (כמו האינטרנט) ופתרון בעיות הקשורות לרשתות תקשורת; בינה מלאכותית העוסקת במערכות שנועדו לדמות פעילות אנושית, ועוד תחומים רבים נוספים.

במסגרת לימודי מדעי המחשב בבית הספר התיכון ניתן כמובן להציג רק מקצת מתחומי הפעילות השונים במדעי המחשב. בכל זאת תוכנית הלימודים התיכונית במדעי המחשב נוגעת במגוון רחב למדי של נושאים, גם בחלק מאלה שהוזכרו לעיל.

## סיכום

בפרק זה תיארונו בקצרה מהו מחשב, מהן היחידות הבסיסיות שמהן הוא בנוי, וכיצד נכתבות ומתבצעות תוכניות מחשב. תיארונו גם את ההתפתחות ההנדסית והטכנולוגית של המחשבים ואת ההתפתחות המדעית של תחום מדעי המחשב.

**מחשב** הוא מכונה אלקטרונית הקולטת נתונים, מעבדת אותם ופולטת מידע שנוצר בתהליך העיבוד.

הנתונים שקולט המחשב נקראים **קלט**.

המידע שפולט המחשב נקרא **פלט**.

העיבוד המבוצע במחשב מונחה על ידי קבוצת הוראות הנקראת **תוכנית מחשב**.

הרכיבים הפיסיים של המחשב נקראים **חומרה** ואוסף תוכניות המחשב נקרא **תוכנה**. החומרה מחולקת לכמה רכיבים בסיסיים: יחידת עיבוד מרכזי, זיכרון, אמצעי קלט ואמצעי פלט. תוכנה של מחשב כוללת בין השאר את מערכת ההפעלה, את המהדרים ותוכניות ליישומים שונים.

**יחידת העיבוד המרכזית** מנהלת את כל התהליכים המתבצעים במחשב.

**הזיכרון** שומר מידע, תוכניות, ותוצאות ביניים של תהליכי עיבוד. הזיכרון מתחלק ל**זיכרון משני** ול**זיכרון ראשי**. המידע השמור בזיכרון מיוצג באמצעות סיביות (**סיבית** – אחת מן הספרות 0 או 1).

**אמצעי הקלט** אחראים על קליטת נתונים ו**אמצעי הפלט** אחראים על פליטת מידע.

תוכנית מחשב נכתבת על ידי **מתכנת** בשפת תכנות. **לשפת תכנות** כללים המכתיבים את אופן כתיבת ההוראות בתוכנית. כיום נכתבות תוכניות מחשב בשפה עילית.

**בשפה עילית** המשפטים דומים למשפטים בשפה טבעית, כמו אנגלית. **שפת מכונה** היא השפה אותה "מבין" המחשב, וההוראות בה מקודדות על ידי סיביות.

לפני ביצוע תוכנית בשפה עילית עליה לעבור שני שלבים: הידור והרצה. **הידור** (קומפילציה) הוא התהליך של תרגום תוכנית בשפה עילית לשפת מכונה. תהליך זה מתבצע על ידי **מהדר** (קומפילר).

**שגיאות תחביר** מתגלות בשלב ההידור. **שגיאות ריצה** מתגלות בזמן ההרצה. תהליך איתור שגיאות ריצה ותיקונן נקרא **ניפוי שגיאות**.

## שאלות נוספות

1. מדוע אי-אפשר לכתוב תוכנית למחשב בשפה טבעית, כמו אנגלית או עברית?
2. מדוע שפת תכנות עילית נקראת בשם זה?
3. למה מתכוונים כאשר אומרים כי מחשב מבין שפת מכונה?
4. כמה מהדרים דרושים להידור תוכנית בשפת C, תוכנית בשפת פסקל, ותוכנית בשפת בייסיק בשלושה מחשבים מסוגים שונים?

## פרק 2 – פתרון בעיות אלגוריתמיות

פרק זה עורך הכרה עם הנושא שבו נעסוק למעשה במהלך כל לימוד היחידה: פתרון בעיות אלגוריתמיות. נכיר את המושג אלגוריתם, מושג מרכזי וחשוב במדעי המחשב, שמשמעותו למעשה פתרון לבעיה, פתרון שאפשר אחר כך לתרגמו לתוכנית מחשב. לאחר מכן נערוך היכרות ראשונית עם מושג התבנית. גם תבנית היא אלגוריתם, כלומר פתרון לבעיה, אך היא יכולה גם לשמש כתת-פתרון לבעיות רבות בעלות מאפיינים משותפים.

### 2.1 אלגוריתמים

המושג **אלגוריתם** שנתמקד בו בפרק זה, הוא מושג מרכזי במדעי המחשב. בפרק זה נכיר ונפתח אלגוריתמים ראשוניים, אשר אינם מיועדים לביצוע במחשב. בפרק הבא ובפרקים הבאים אחריו נפתח אלגוריתמים המיועדים ליישום בתוכניות מחשב, לביצוע במחשב.

קבוצת ההוראות שבדוגמה הבאה היא אלגוריתם:

1. הריג עשר כוסות מים
2. הוסף קורט מל
3. הוסף ג'י קילו פגיתים למים היוגורטים
4. הכא את המים לריגה נוספת
5. כשל את הפגיתים למשך 20 דקות על אש קטנה
6. סן את הפגיתים

האלגוריתם שבדוגמה זו הוא מתכון לבישול חצי קילו פתיתים.

הנה דוגמה נוספת לאלגוריתם:

1. בגר מספר שלם חיובי
2. גבר את מספר המספר
3. גלק את הגולדאכ 3-2
4. כגלב את שארית הגולדאכ

#### שאלה 2.1

מהי תוצאת ביצוע האלגוריתם שלעיל עבור המספר 1977?

באופן כללי ניתן לומר כי אלגוריתם הוא מתכון אך לא דווקא לבישול:

**אלגוריתם** הוא מתכון לביצוע משימה. אלגוריתם מורכב תמיד מקבוצת הוראות חד-משמעיות ואפשריות לביצוע, אשר **סדר ביצוען** מוגדר היטב.

המילים המודגשות בהגדרה שלעיל הן שלושת המאפיינים החשובים לאלגוריתם: חד-משמעיות, אפשריות לביצוע וסדר ביצוע מוגדר היטב.

**חד-משמעיות:** הוראה המופיעה באלגוריתם חייבת להיות חד-משמעית. כלומר, כל ביצוע שלה צריך להסתיים תמיד באותה תוצאה. כך, למשל, ההוראה השנייה בדוגמה האחרונה, **גבר את מספר המספר**, היא חד-משמעית. לעומתה, ההוראה **ג'י קצב הצידה** אינה חד-משמעית: אנשים שונים יזוזו לפיה למקומות שונים, וייתכן גם כי אותו אדם יזוז לפיה אחרת בפעמים שונות.

**אפשר לבצע:** הוראה באלגוריתם צריכה להיות אפשרית לביצוע, ובפרט, עליה להתאים למבצע המיועד שלה. למשל טבח יכול לבצע את ההוראה *הכג 10 כ/סל מ/ס*, אך אינו יכול לבצע את ההוראה *הכג אל המטוס*.

**סדר ביצוע:** סדר ביצוע הוראות האלגוריתם הוא לפי סדר הופעתן, אם לא נאמר אחרת. הביצוע של האלגוריתם מסתיים כאשר אין יותר הוראות. כאשר ביצוע של הוראה מסתיים, ממשיכים להוראה הבאה.

צורת הכתיבה שאנו כותבים בה אלגוריתמים נקראת כתיבה בפסאודו-קוד. **פסאודו-קוד** (קוד מדומה, pseudo-code) של אלגוריתם הוא ייצוג או כתיבה של האלגוריתם בדרך דמוית שפת תכנות, כלומר, במילים ובמשפטים בשפה חופשית אבל ברורה וחד-משמעית.

המונח "אלגוריתם" נגזר ככל הנראה משמו של המתמטיקאי מוחמד אל-חואריזמי, שהשתבש לאל-גואריזמי. אל-חואריזמי חי במאה ה-9 לספירה, באזור חואריזם, אשר נמצא היום באוזבקיסטן. אל-חואריזמי היה הראשון שניסח את הכללים המשמשים אותנו עד היום לביצוע ארבע פעולות החשבון הבסיסיות. במאה ה-14 החל המונח "אלגוריתם" להיות שגור בפי המתמטיקאים, ככינוי ל"מתכון מתמטי". מתכונים מתמטיים כאלו הם למשל אלגוריתם להכפלת שני מספרים ("כפל ארוך"), אלגוריתם להעלאה בחזקה של מספר אחד באחר, אלגוריתם למציאת המחלק המשותף הגדול ביותר של שני מספרים שלמים חיוביים (האלגוריתם של אוקלידס).

אנו נתמקד באלגוריתמים כפתרון לבעיות אלגוריתמיות:

**בעיה אלגוריתמית** היא בעיה אשר נתונות בה נקודת מוצא ומטרה ונדרש אלגוריתם שלאחר ביצועו מגיעים מנקודת המוצא אל המטרה.

כלומר, בעיה אלגוריתמית מגדירה למעשה משימה: הגעה מנקודת המוצא הנתונה אל המטרה הנתונה. אלגוריתם הפותר את הבעיה הוא מתכון לביצוע המשימה הזאת. הנה דוגמה לבעיה אלגוריתמית:

## חצי'ה 1

**מטרת הבעיה ופתרונה:** הצגת בעיה אלגוריתמית ראשונה, והצגת אלגוריתם לפתרונה.

שייט נמצא על גדת נהר ועמו כרוב, כבש וזאב. הוא רוצה לעבור לגדה השנייה בסירה קטנה ולהעביר את השלושה. הסירה יכולה להכיל בו-זמנית רק את השייט ועוד אחד מבין שלושת הפריטים שאיתו. השייט אינו יכול להשאיר את הזאב ואת הכבש ביחד או את הכבש ואת הכרוב ביחד ללא השגחתו. פתחו אלגוריתם שינחה את השייט כיצד להעביר את הכבש, את הזאב ואת הכרוב מהגדה האחת אל האחרת.

בעיה זו מגדירה משימה של חציית נהר באילוצים שונים. נקודת המוצא היא המצב שהשייט, הזאב, הכבש והכרוב נמצאים על גדה אחת (גדה א) של הנהר. המטרה היא המצב שהשייט, הזאב, הכבש והכרוב נמצאים על הגדה האחרת (גדה ב).

נציג אלגוריתם אשר מורה כיצד לבצע את המשימה המתוארת בבעיה. בצד הוראות האלגוריתם מוצג מעקב אחר מהלך ביצועו.

## אלגוריתם לפתרון בעיה 1

גדה ב	גדה א	
	שייט, כרוב, כבש, זאב	(נקודת המוצא) ←
שייט, כבש	כרוב, זאב	1. הפלא מגדה א לגדה ב עם הכבש
כבש	שייט, כרוב, זאב	2. הפלא מגדה ב לגדה א לבד
שייט, כבש, זאב	כרוב	3. הפלא מגדה א לגדה ב עם הזאב
זאב	שייט, כרוב, כבש	4. הפלא מגדה ב לגדה א עם הכבש
שייט, כרוב, זאב	כבש	5. הפלא מגדה א לגדה ב עם הכוב
כרוב, זאב	שייט, כבש	6. הפלא מגדה ב לגדה א לבד
שייט, כרוב, כבש, זאב		7. הפלא מגדה א לגדה ב עם הכבש

האם זהו האלגוריתם היחיד הפותר את בעיה 1?

לא. למשל גם האלגוריתם הבא הוא פתרון לבעיה 1:

1. הפלא מגדה א לגדה ב עם הכבש
2. הפלא מגדה ב לגדה א לבד
3. הפלא מגדה א לגדה ב עם הכוב
4. הפלא מגדה ב לגדה א עם הכבש
5. הפלא מגדה א לגדה ב עם הזאב
6. הפלא מגדה ב לגדה א לבד
7. הפלא מגדה א לגדה ב עם הכבש

**שימו ♥ :** במקרים רבים קיים יותר מאלגוריתם אחד הפותר בעיה אלגוריתמית נתונה.

## 2.2 שאלה

עקבו באמצעות טבלה מתאימה אחר מהלך ביצוע האלגוריתם הנוסף לפתרון בעיה 1.

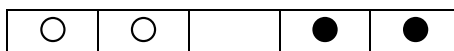
### סוף פתרון בעיה 1

בעיה 1 היא בעיה ספציפית העוסקת בפריטים מסוימים (כרוב, כבש, זאב). אפשר להרחיב אותה לבעיה כללית שנתונים בה שלושה פריטים כלשהם, הנמצאים בגדה א, ויש להעבירם לגדה ב, בשמירה על האילוצים שבבעיה 1. את הבעיה הכללית אפשר לפתור באמצעות אותם אלגוריתמים שפותרים את בעיה 1, אך צריך יהיה להחליף בהם כל התייחסות ספציפית לכרוב, לכבש ולזאב ב"פריט 1", ב"פריט 2", וב"פריט 3" בהתאמה. כך אפשר יהיה להשתמש באותו אלגוריתם כדי לפתור את הבעיה עבור כל שלושה פריטים המתאימים לאילוצים. למשל, עבור כלב, חתול ועכבר, או עבור אריה, פרה וחציר. גם במקרה של כלב, חתול ועכבר, לא ניתן להשאיר את פריט 1 (כלב) ואת פריט 2 (חתול) יחד ללא השגחה, וגם לא את פריט 2 (חתול) ואת פריט 3 (עכבר). כך גם לגבי המקרה שפריט 1 הוא אריה, פריט 2 הוא פרה ופריט 3 הוא חציר.

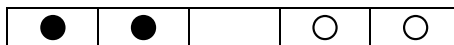
במדעי המחשב עוסקים בדרך כלל בבעיות אלגוריתמיות, אשר לכל אחת מהן אפשרויות רבות לנקודת מוצא. אלגוריתם הפותר בעיה אלגוריתמית שעבורה כמה נקודות מוצא, צריך להיות כללי ולפתור את הבעיה עבור כולן.

### שאלה 2.3

על לוח בן חמש משבצות מונחות שתי אבני משחק מצבע אחד ושתי אבני משחק מצבע אחר בנקודת המוצא הבאה:



המטרה היא להביא את אבני המשחק למצב:



הפעולות המותרות הן: העברת אבן משחק למשבצת סמוכה פנויה, והקפצת אבן משחק מצבע אחד מעל לאבן משחק מצבע אחר אל משבצת פנויה.

פתחו אלגוריתם להעברת אבני המשחק מנקודת המוצא אל מצב המטרה.

**שימו ♥ :** בבעיה זו מספר רב של אפשרויות לנקודת המוצא, כיוון שכל זוג צבעים שונים של אבני המשחק מכתוב למעשה נקודות מוצא שונות. האלגוריתם שתפתחו יתאים לכל זוג צבעים, ולכן יהיה כללי.

בפתרון בעיה 1 פיתחנו אלגוריתם, אשר בכל ביצוע שלו מבוצעות כל הוראותיו, זו אחר זו, לפי סדר הופעתן. לעתים יש צורך באלגוריתמים אשר ביצוע חלק מההוראות בהם הוא מותנה. נראה זאת בבעיה הבאה.

## מטרה 2

**מטרת הבעיה ופתרונה:** הצגת אלגוריתם שיש בו הוראה לביצוע-בתנאי.

שני מקלים, מכל א ומכל ב, מכילים מספר שונה של תפוזים. סכום מספרי התפוזים בשני המכלים הוא זוגי.

פתחו אלגוריתם להעברת תפוזים בין המכלים, כך שתתבצע רק העברה אחת של מספר תפוזים, ולאחר ההעברה יכילו המכלים מספר שווה של תפוזים.

**שימו ♥ :** יש אפשרויות רבות לנקודת המוצא, כי יש אפשרויות רבות לזוגות מספרי תפוזים שונים זה מזה שסכומם זוגי.

### שאלה 2.4

באלגוריתם לפתרון יהיה צריך לציין כמה תפוזים יש להעביר בין המכלים כדי שמספרי התפוזים בשני המכלים יהיו שווים. לפניכם זוגות של מספרי תפוזים. בכל זוג המספר השמאלי מציין את מספר התפוזים במכל א, והמספר הימני מציין את מספר התפוזים במכל ב. עבור כל זוג, חשבו כמה תפוזים יש להעביר מהמכל המלא יותר למכל הפחות מלא.

א. 100 160

ב. 971 935

ג. 1 1001

בעזרת התשובה לשאלה 2.4 ניתן להיווכח שמספר התפוזים להעברה הוא חצי מההפרש בין מספרי התפוזים שבמכלים. צריך לחשב מספר זה לפני ההעברה, ולכן ההוראה הראשונה

באלגוריתם לפתרון הבעיה תהיה:  
 1. **יש** **אם** **מספר** **הגפולים** **אחריה**: **גבי** **מהפריש** **בין** **מספרי** **הגפולים**  
**שבמילים**

אם במכל א יש יותר תפוזים, צריך לבצע את ההוראה הבאה:  
**העבר** **גפולים** **ממכל** **א** **למכל** **ב** **על** **פי** **המספר** **המחולק**

ואם במכל ב יש יותר תפוזים, צריך לבצע את ההוראה הבאה:  
**העבר** **גפולים** **ממכל** **ב** **למכל** **א** **על** **פי** **המספר** **המחולק**

**?** כיצד נבחר איזו משתי ההוראות לבצע?

בחירת ההוראה המתאימה לביצוע תיקבע על פי תנאי. התנאי המתאים הוא:

**במכל** **א** **יש** **יותר** **גפולים** **מאשר** **במכל** **ב**

אם התנאי יתקיים, תתבצע ההוראה הראשונה. אחרת, תתבצע ההוראה השנייה. באלגוריתם לפתרון הבעיה ננסח את ההתניה הזאת על ידי הוראה לביצוע-בתנאי באופן הבא:

## אלגוריתם לפתרון בעיה 2

1. **יש** **אם** **מספר** **הגפולים** **אחריה**: **גבי** **מהפריש** **בין** **מספרי** **הגפולים**  
**שבמילים**

2. **אם** **במכל** **א** **יש** **יותר** **גפולים** **מאשר** **במכל** **ב**

2.1. **העבר** **גפולים** **ממכל** **א** **למכל** **ב** **על** **פי** **המספר** **המחולק**

3. **אחרת**

3.1. **העבר** **גפולים** **ממכל** **ב** **למכל** **א** **על** **פי** **המספר** **המחולק**

סוף פתרון בעיה 2

ההוראה	אם ...
	.
	.
	אחרת
	.
	.
	.
היא הוראה לביצוע-בתנאי, המורה על ביצוע קבוצת הוראות אחת או קבוצת הוראות אחרת על פי תנאי.	
הוראה לביצוע-בתנאי היא הוראת בקרה. הוראת בקרה היא הוראה שמשפיעה על מהלך ביצוע ההוראות באלגוריתם.	

## שאלה 2.5

על השולחן מונחות שלוש מעטפות בשורה, בכל מעטפה יש פתק ועליו רשום מספר. במעטפה אחת יש פתק שעליו רשום המספר 0, ובשתי המעטפות האחרות יש פתקים שרשומים עליהם מספרים שונים מ-0. המעטפה שבה נמצא הפתק שעליו רשום 0 איננה המעטפה האמצעית בשורה. א. יש אינסוף אפשרויות לנקודת המוצא, כי על שניים מהפתקים רשומים מספרים כלשהם שונים מ-0. תארו חמש אפשרויות שונות של נקודת המוצא.



ב. פתחו אלגוריתם שמטרתו היא לשים באמצע, בין שתי המעטפות האחרות, את המעטפה עם הפתק שעליו רשום 0. הפעולות שבהן יש להשתמש לביצוע המשימה הן: קריאת המספר הרשום על המעטפה, והחלפת מקומות בין מעטפות שכנות.

בפתרון בעיה 2 ראינו אלגוריתם שכלל ביצוע הוראות בתנאי. לעתים יש צורך באלגוריתמים אשר ביצוע חלק מההוראות בהם חוזר כמה פעמים. נראה זאת בבעיה הבאה.

### קצ'ה 3

**מטרת הבעיה ופתרונה:** הצגת אלגוריתם שבו קיימת הוראה לביצוע-חוזר.

על השולחן יש קלפים המסודרים בשורה. מספר הקלפים בשורה הוא אי-זוגי, והקלף האמצעי הוא לבן. כל הקלפים משמאל לקלף הלבן הם שחורים וכל הקלפים שממימין אדומים. נתונות גם שתי סימניות: סימניה-1 המוצבת על הקלף שבקצה השמאלי וסימניה-2 המוצבת על הקלף שבקצה הימני.

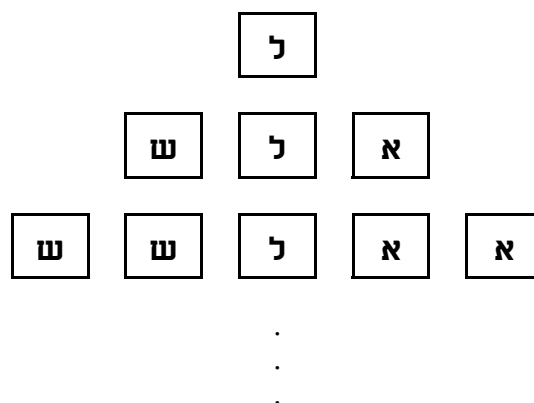
פתחו אלגוריתם אשר מסדר את שורת הקלפים מחדש כך שכל הקלפים האדומים יהיו משמאל לקלף הלבן וכל השחורים מימינו.

הפעולות המותרות לביצוע הן:

- ♦ הצבת סימניה מימין או משמאל לקלף שעליו היא מוצבת. הצבת סימניה כוללת קריאת צבע הקלף שעליו היא מוצבת.
- ♦ החלפה זה בזה של מקומות הקלפים שעליהם מוצבות הסימניות.

נקודת המוצא של הבעיה היא המצב שמונחות על השולחן שורת הקלפים ושתי הסימניות. המטרה היא הסידור החדש, שמוחלפים בו מקומותיהם של הקלפים האדומים ושל הקלפים השחורים.

**שימו ♥:** יש אינסוף אפשרויות לנקודת המוצא כי מספר הקלפים בשורה יכול להיות כל מספר אי-זוגי, למשל (א) מסמן קלף אדום, ל מסמן קלף לבן ו-ש מסמן קלף שחור):



האלגוריתם המבוקש אינו צריך להיות תלוי באורך שורת הקלפים הנתונה, כלומר ביצועו צריך להשיג את המטרה לכל נקודת מוצא אפשרית.

כדי לפתח אלגוריתם לפתרון הבעיה, נחשוב תחילה על מקרה פרטי של הבעיה. נניח כי בשורה שבעה קלפים. כלומר, נקודת המוצא היא:



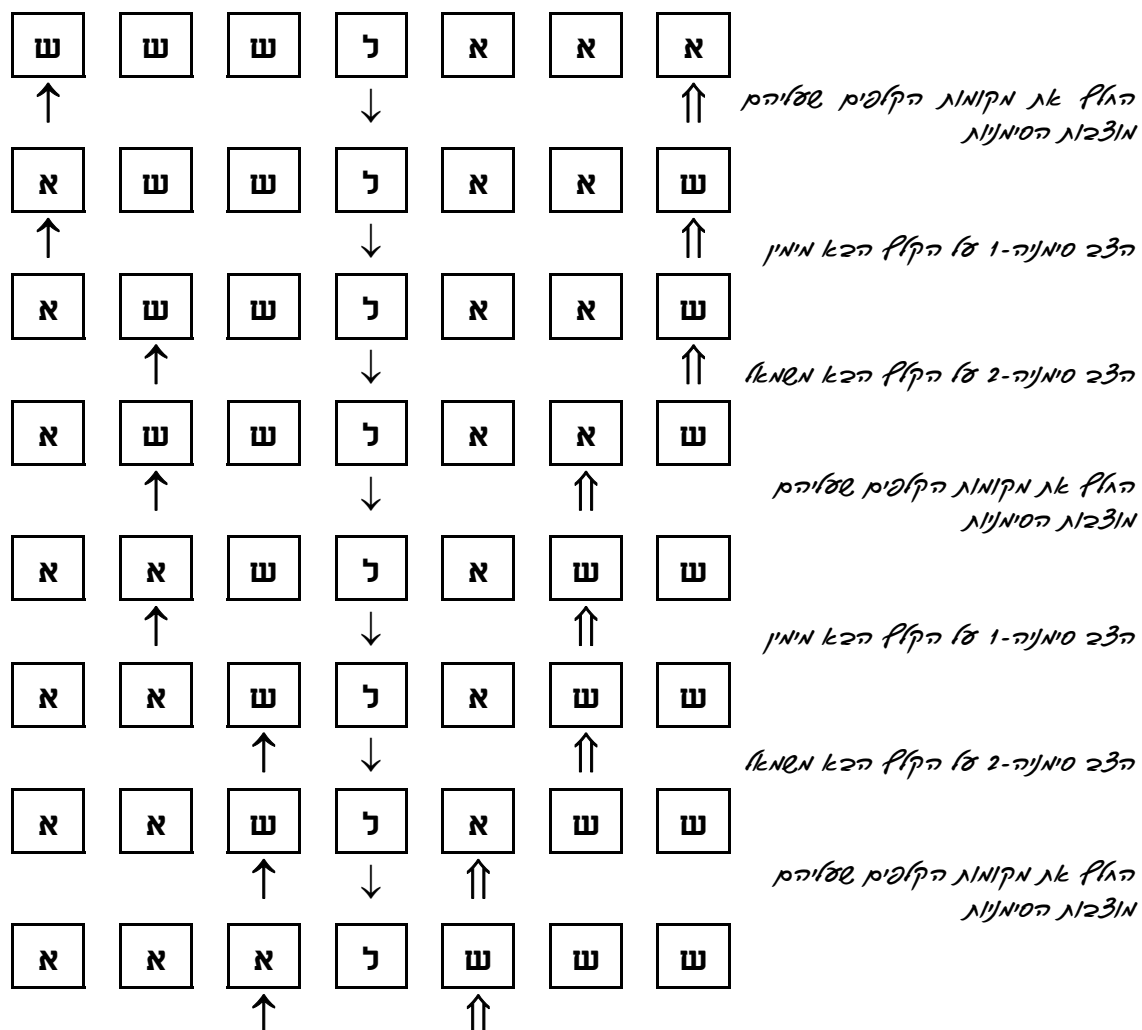
ונקודת הסיום היא:



את המקרה הפרטי הזה ניתן לפתור באופן הבא:

ראשית צריך להחליף את זוג הקלפים שבקצות השורה, ולקדם את הסימניות פנימה (כלומר, לקדם את הסימניה השמאלית מקום אחד ימינה ואת הימנית מקום אחד שמאלה). אחר כך צריך להחליף את זוג הקלפים שעליהם מצביעות הסימניות כעת, ושוב לקדם את הסימניות פנימה. לבסוף נותר להחליף את זוג הקלפים שעדיין לא הוחלפו.

הנה תיאור מפורט של אלגוריתם הפותר את המקרה הפרטי, יחד עם מעקב אחר ביצועו (סימניה-1 מסומנת בחץ דק וסימניה-2 מסומנת בחץ עבה):



פתרון המקרה הפרטי של הבעיה מלמד כי כדי להשיג את המטרה יש לבצע כמה פעמים את התת-משימה הבאה :

החלפת מקומות שהקלפים עליהם מוצבות הסימניות וקידום הסימניות פנימה

תת-משימה זו תבוצע על ידי קבוצת ההוראות הבאה :

1. האלף אק מקומות הקלפים שצויהם מוצבות הסימניות
2. הצב סימניה-1 על הקלף הכא מימין
3. הצב סימניה-2 על הקלף הכא משמאל

האלגוריתם הדרוש צריך להורות על ביצוע-חוזר של התת-משימה (כלומר, שלוש ההוראות). מספר החזרות תלוי במספר הקלפים בשורה. מאחר שיש לכתוב אלגוריתם כללי, שיתאים לשורת קלפים בכל אורך אי-זוגי, לא יהיה זה מספיק לכתוב אלגוריתם המטפל באורך מסוים. בנוסף, אפילו אם היה ידוע מראש אורך מסוים, למשל 401, אין זה סביר שנכתוב אלגוריתם ובו כתובות שלוש ההוראות שוב ושוב 401 פעמים.

**?** כיצד ננסח אלגוריתם שיתאים לכל שורת קלפים באורך אי-זוגי ?

כדי לנסח את האלגוריתם המבוקש, נשתמש בהוראה לביצוע-חוזר-בתנאי, שתורה לחזור על ביצוע קבוצת שלוש ההוראות שתיארנו כל עוד מתקיים התנאי :

הקלף שצויה מוצבות הסימניות אינו לבן

האלגוריתם הבא פותר את הבעיה, והוא כולל הוראה לביצוע-חוזר-בתנאי :

### אלגוריתם לפתרון בעיה 3

1. כל עוד הקלף שצויה מוצבות הסימניות אינו לבן בצב:
  - 1.1. האלף אק מקומות הקלפים שצויהם מוצבות הסימניות
  - 1.2. הצב סימניה-1 על הקלף הכא מימין
  - 1.3. הצב סימניה-2 על הקלף הכא משמאל

סוף פתרון בעיה 3

ההוראה	כל עוד ... בצב:
.	.
.	.
.	.
היא הוראה לביצוע-חוזר-בתנאי, המורה לחזור על ביצוע של קבוצת הוראות כל עוד מתקיים תנאי מסוים.	
בדומה להוראה לביצוע-בתנאי גם הוראה לביצוע-חוזר-בתנאי היא הוראת בקרה.	

### שאלה 2.6

תארו את מצבי ביצוע האלגוריתם לפתרון בעיה 3 עבור כל אחת מן האפשרויות הבאות של נקודת המוצא.

- א. שורת קלפים באורך 9.
- ב. שורת קלפים באורך 3.

## שאלה 2.7

שנו את האלגוריתם שבפתרון בעיה 3 כך שבתום הביצוע תושג מטרה אחרת: הקלפים שמשני צידי הקלף הלבן יהיו מסודרים לסירוגין לפי צבעים (כלומר, ש, א, ש, ..., ל, .... א, ש, א).

## 2.2 תבניות

בפתרון בעיה 3 בסעיף הקודם נכלל ביצוע-חוזר של הפעולה "החלף את מקומות הקלפים שעליהם מוצבות הסימניות". פעולת ההחלפה משולבת בבדיקה חוזרת ונשנית של מקום הסימניות – "האם הן מוצבות על קלף שאינו לבן". **פעולת ההחלפה, ופעולת הבדיקה החוזרת של ערך** (במקרה זה, צבע) הן פעולות שימושיות בפתרונות של בעיות אלגוריתמיות נוספות רבות. למשל, הבעיה של מיון שורת מספרים היא בעיה אלגוריתמית, ובפתרונה ניתן להשתמש שוב ושוב ב**החלפה** בין מספרים סמוכים בשורה, עד אשר השורה תהיה ממוינת. גם הבעיה של חיפוש מספר מסוים בשורת מספרים היא בעיה אלגוריתמית. ניתן לפתור אותה במעבר שיטתי על פני המספרים שבשורה משמאל לימין, תוך **בדיקה חוזרת של הערך** של המספר הבא בשורה (עד מציאת המספר הרצוי או עד ההגעה לסוף השורה).

לעתים קרובות, פתרונות לבעיות אלגוריתמיות כוללים פעולות, אשר חוזרות שוב ושוב בפתרונות שונים. לכל פעולה כזו יש מבנה אלגוריתמי בסיסי המתאר אופן ביצוע של משימה מנקודת מוצא למטרה. כיוון שפעולה זו משמשת שוב ושוב בפתרונות שונים נוח לקרוא לה בשם, ולהתייחס אל המבנה האלגוריתמי שלה כאל **תבנית** אלגוריתמית לביצוע משימה. **תבניות** מורכבות מהמרכיבים הבאים:

- **שם התבנית**, המבטא בצורה מאוד תמציתית משימה לביצוע או את דרך ביצועה (למשל החלפת ערכים).
- **נקודת מוצא**, המציינת את המצב ההתחלתי הנתון של המשימה לביצוע.
- **מטרה**, המתארת את המצב הסופי, את הפלט הדרוש, או את הערך שיש להחזיר בתום הביצוע.
- **אלגוריתם**, המתאר מתכון לביצוע המשימה. האלגוריתם הוא לב התבנית.

מחקרים שונים בתחום הוראת מדעי המחשב מצביעים על כך שפותרים מנוסים של בעיות אלגוריתמיות שומרים בזיכרונם פתרונות קודמים על פי התבנית שבבסיסם. בבואם לפתור בעיה אלגוריתמית הם מסוגלים לזהות קשר בינה ובין בעיות אחרות שכבר פתרו, על פי התבניות שמשמשות בפתרון הבעיות. משום כך, השימוש בתבניות מסייע לתהליך הפיתוח של אלגוריתמים. לכן בספר הלימוד אנו נתייחס גם לתבניות, נציג אותן ונדון בהן.

ספר הלימוד משלב תבניות בצורה מודרגת. כבר בפרק הבא מוצגות תבניות ראשונות, ובפרקים שאחריו מוצגות עוד ועוד תבניות בהתאם לבעיות האלגוריתמיות שבהם. כל פרק שנוספות בו תבניות חדשות, הן מוצגות בסוף הפרק, מודגמות, מתורגלות, ומקושרות לבעיות אלגוריתמיות שהוצגו בפרק. לפעמים תבנית חדשה אף מוצגת כסעיף של פרק. לעתים מורחב בפרק חדש המבט על תבנית שכבר הוצגה בפרק קודם.

הרחבה בנושא התבניות ושאלות שבפתרון יש שימוש בתבניות ניתן למצוא באתר הספר ברשת האינטרנט.

## סיכום

בפרק זה הכרנו את המושגים **אלגוריתם** ו**בעיה אלגוריתמית**.

**אלגוריתם** הוא מתכון לביצוע משימה, המורכב מקבוצת הוראות חד-משמעיות ואפשריות לביצוע אשר סדר ביצוען מוגדר היטב.

**בעיה אלגוריתמית** היא בעיה שבה מתוארות נקודת מוצא ומטרה המגדירות משימה, ונדרש אלגוריתם הפותר את המשימה, כלומר מגיע מנקודת המוצא אל המטרה.

בבעיות אלגוריתמיות ייתכנו אפשרויות רבות, לפעמים אינסוף, לנקודת המוצא.

בפרק זה ובפרקים הבאים אנו משתמשים ב**פסאודו-קוד** לכתובת אלגוריתמים, כלומר, במילים ובמשפטים בשפה חופשית אך ברורה וחד-משמעית.

**סדר ביצוע ההוראות** של אלגוריתם הוא על פי סדר הופעתן, אם לא נאמר אחרת.

כאשר יש צורך להתנות ביצוע של הוראות בקיום תנאי מסוים, האלגוריתם יכול **הוראה לביצוע-בתנאי (אם... אחרת...)**.

כאשר יש צורך בביצוע-חוזר של הוראות, האלגוריתם יכול **הוראה לביצוע-חוזר (כ... עד...)**.

הוראה לביצוע-בתנאי והוראה לביצוע-חוזר הן **הוראות בקרה** המנחות את אופן הביצוע של הוראות האלגוריתם.

## שאלות נוספות

1. בצעו את האלגוריתם הבא ותארו את מהלך ביצועו. מהי התוצאה המוכרזת בסוף הביצוע?

1. הכפל ב-2 את של הארבע

2. הוסף 5 לארבע

3. הכפל את הארבע ב-50

4. הוסף את גילך (כמספר שלם) לארבע

5. הגסר 250 מן הארבע

6. גלק ב-100 את הארבע

7. הכיז על הארבע

2. על השולחן מונחים שלושה קלפים בשורה. על כל קלף רשום מספר. נתון האלגוריתם הבא:

1. השווה את המספר של הקלף השמאלי למספר של הקלף האמצעי

2. אם המספר של הקלף השמאלי גדול מהמספר של הקלף האמצעי

2.1. הגלף את מקומותיהם

3. השווה את המספר של הקלף האמצעי הנכבי למספר של הקלף הימני

4. אם המספר של הקלף האמצעי הנכבי גדול מהמספר של הקלף הימני

4.1. הגלף את מקומותיהם

א. תארו את מהלך ביצוע האלגוריתם עבור נקודת המוצא הבאה:

24

2

15

ב. תארו את מהלך ביצוע האלגוריתם עבור נקודת המוצא הבאה :

13	24	2
----	----	---

ג. תארו את מהלך ביצוע האלגוריתם עבור נקודת המוצא הבאה :

13	2	15
----	---	----

ד. מהי הבעיה האלגוריתמית שהאלגוריתם פותר?

ה. שנו את האלגוריתם כך ששיגי את המטרה הבאה : הקלף שרשום עליו המספר הקטן ביותר יהיה בקצה הימני של שורת הקלפים.

3. מפעל מסמן כל מוצר שלו בסימן המורכב מאות גדולה באלף-בית האנגלי ומספרה. כלומר סדרת הסימנים של המפעל היא הסדרה  $A_0, A_1, \dots, A_9, B_0, B_1, \dots, Z_0, \dots, Z_9$  הוא הסימן הראשון בסדרה ו- $Z_9$  הוא הסימן האחרון בסדרה.

א. מהם הסימנים שעוקבים לסימנים  $A_1, B_9, Z_0$ ?

ב. פתחו אלגוריתם לקריאת סימן שאינו הסימן האחרון בסדרה ולכתיבת הסימן הבא אחריו. הפעולות המותרות הן : קריאת סימן (המורכב מאות ומספרה), כתיבת אות, כתיבת ספרה.

4. בכיתת תלמידים יש ילד אחד ששערו ג'ינג'י וילדים רבים שצבע שערם אינו ג'ינג'י. התלמידים מסודרים בטור. הילד הג'ינג'י אינו עומד בראש הטור.

פתחו אלגוריתם אשר מטרתו היא שהילד הג'ינג'י יעמוד בראש הטור. הפעולות המותרות הן : עמידה מול התלמיד שבראש הטור, התקדמות לתלמיד הבא בטור, החלפת מקומות בין התלמיד שאתה עומד מולו ובין התלמיד שבראש הטור.

5. על השולחן מונחים קלפים שחורים ואדומים בשורה. בשורה שלושה קלפים לפחות. הקלפים השחורים נמצאים משמאל לקלפים האדומים, וידוע כי מספר הקלפים האדומים קטן ממספר הקלפים השחורים.

נתונות שתי סימניות : סימניה-1 המוצבת על הקלף שבקצה השמאלי, וסימניה-2 המוצבת על הקלף שבקצה הימני.

הפעולות המותרות הן : הצבת סימניה מימין או משמאל לקלף שעליו היא מוצבת, והחלפה של מקומות הקלפים שעליהם מוצבות הסימניות זה בזה.

א. ציינו שלוש אפשרויות שונות לשורת הקלפים.

ב. נתון האלגוריתם הבא אשר מטרתו היא שכל הקלפים האדומים יהיו משמאל לכל הקלפים השחורים. השלימו את האלגוריתם :

1.  $\text{כא עז} \underline{\hspace{10em}} \text{עז} :$

1.1. האלף אג מקומות הקלפים שעליהם מוצב/ה הסימניות

1.2. הצב סימניה-1 על הקלף הכא מימין

1.3. הצב סימניה-2 על הקלף הכא משמאל

## פרק 3 – מודל חישוב בסיסי

בפרק 1 הכרנו את המכונה מחשב, ובפרק 2 ראינו אלגוריתמים ראשוניים, אשר לא נועדו לביצוע במחשב. בפרק זה נכיר אלגוריתמים המיועדים לביצוע במחשב, ונראה כיצד נכתוב אותם בתוכניות בשפת התכנות C#.

באמצעות בעיות אלגוריתמיות שונות נציג את מרכיביהם הבסיסיים של אלגוריתמים ואת יישומם בשפת התכנות C#. בכל בעיה יתואר פלט דרוש עבור קלט נתון. הקלט הוא נקודת המוצא של הבעיה, והפלט הוא המטרה. כל אלגוריתם שיפותח יתואר בצורה מילולית, בדומה לתיאורים שהוצגו בפרק 2, ולאחר מכן יוצג יישומו באמצעות תוכנית בשפת התכנות C#.

בפרק זה נכיר את האמצעי לשמירת נתונים במחשב, את ההוראות לביצוע קלט ופלט וכיצד נבצע חישובים ונשמור את תוצאותיהם. לאחר מכן נעקוב אחר מהלך ביצוע של אלגוריתם.

### 3.1 צעדים ראשוניים: הוראת פלט, הוראת קלט ומשתנים

המחשב מציג הודעות באמצעות אמצעי פלט. בפרק 1 הזכרנו את שני אמצעי הפלט הנפוצים: מדפסת וצג. בפתרון הבעיה הבאה נראה אלגוריתם להצגת מילים כפלט. האלגוריתם מיושם בתוכנית בשפת C#, התוכנית הראשונה בפרק.

#### הצגה 1

מטרת הבעיה ופתרונה: הצגת משפט כפלט

פתחו אלגוריתם שהפלט שלו הוא המילים Hello World, וישמו את האלגוריתם בתוכנית מחשב בשפת C#.

**האלגוריתם** לפתרון הבעיה יהיה האלגוריתם הפשוט הבא, הכולל הוראת פלט אחת:

1. הצג כפלט את המילים Hello World

**היישום** בשפת C# של הוראת הפלט (הצגת הפלט על המסך) ייראה כך:

```
System.Console.WriteLine("Hello World");
```

נבחן ממה מורכבת הוראה זו:



כיוון שבתוכנית מתבצעות פעולות רבות של מחלקות השייכות למרחב השמות System, (כגון פעולות קלט פלט של המחלקה Console), נהוג לקצר את כתיבת הפעולות באמצעות הכרזה בתחילת התוכנית שהיא משתמשת במחלקות הנמצאות במרחב שמות זה. ההכרזה נכתבת כך:

```
using System;
```

וכעת ניתן לכתוב את הוראת הפלט בצורה מקוצרת כך:

```
Console.WriteLine("Hello World");
```

כעת ניצור מהמשפט שכתבנו תוכנית מלאה בשפת C#:

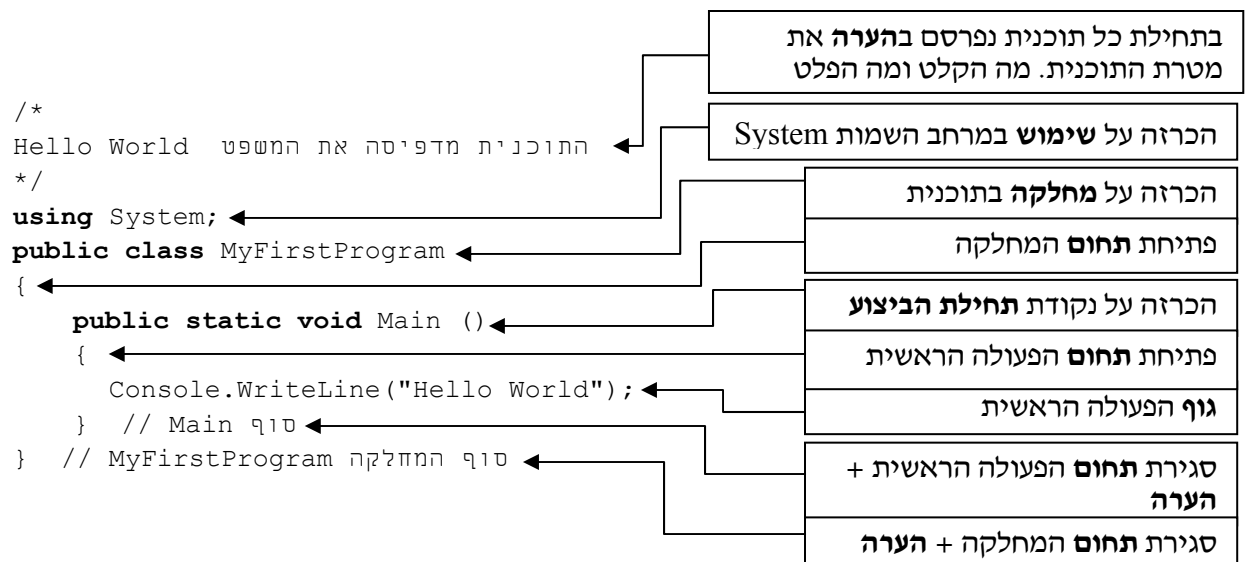
```
using System;
public class MyFirstProgram
{
    public static void Main ()
    {
        Console.WriteLine("Hello World");
    }
}
```



**תוצאת** ההרצה של התוכנית תהיה הצגת המילים Hello World על המסך.

### סוף פתרון בעיה 1

ננסה להבין כמה מהחלקים שהוספנו לתוכנית ונוסיף הערות להבהרה:



## מחלקה – class

כל תוכנית בשפה מורכבת ממחלקות שונות. לכל מחלקה תפקיד ואחריות משלה.

תוכנית זו היא תוכנית פשוטה ביותר, ולכן מכילה מחלקה אחת בלבד, שהיא התוכנית כולה.

כל מחלקה מוגדרת באמצעות המילה `class`. בתוכנית זו מוכרז כי המחלקה `MyFirstProgram` היא ציבורית (`public`), משמע, פתוחה לשימוש לכל המעוניין. בשלב הראשון נכתוב תוכניות המכילות מחלקה אחת בלבד ונצהיר שמחלקה זו היא ציבורית.

בשפת C# מקובל כי שם מחלקה מתחיל תמיד באות גדולה, ואם שם המחלקה מורכב מכמה מילים, הן נכתבות צמודות זו לזו, והאות הראשונה בכל מילה היא גדולה.

## Main

לכל תוכנית יש נקודת התחלה אחת בלבד. שורת הכותרת `Main` מציינת נקודה זו.

את המשמעות המדויקת של שורה זו על כל מרכיביה תבינו בהמשך לימודיכם.



המחלקה אשר מכילה את נקודת תחילת התוכנית (מכילה את שורת ה-Main) היא המחלקה הראשית בתוכנית. שם המחלקה הראשית הוא למעשה שם התוכנית. בדוגמה זו, שם המחלקה הראשית הוא MyFirstProgram.

## גוף הפעולה הראשית

בגוף הפעולה הראשית נכתוב את רצף ההוראות שהוא תרגום המקודד את האלגוריתם הפותר את הבעיה לשפת התכנות.

כל הוראה נכתבת בשורה נפרדת המסתיימת בסימן ;.

בגוף תוכנית זו נכללת הוראה יחידה המציגה על המסך את הכיתוב Hello World.

אנו מבצעים זאת על ידי קריאה לפעולה WriteLine אשר מציגה שורה על המסך. שימו לב כיצד פנינו לפעולה זו: Console.WriteLine. זהו שמה המקוצר של הפעולה, המעיד על כך שהפעולה WriteLine היא פעולת פלט, השייכת למחלקה האחראית לקלט ולפלט. השם המלא הוא System.Console.WriteLine והוא מעיד על כך שהמחלקה Console, שייכת למרחב השמות של System (הכוללת מחלקות שאחראיות לפעולות מערכת כלליות).

## תחום

את רצף ההוראות, המהוות את גוף הפעולה הראשית, יש לתחום בין פותח מסולסל לסוגר מסולסל (הסימנים {...}).

בדומה לתחום הפעולה הראשית, יש לתחום בין פותח מסולסל לסוגר מסולסל את כל ההוראות השייכות למחלקה. אם כך, בתוכנית זו הוגדרו שני תחומים של הוראות: האחד למחלקה התחומה בסימנים { } החיצוניים, והשני לפעולה הראשית Main התחומה בסימנים { } הפנימיים.

## הערה

פעמים רבות נרצה לכתוב הערות בתוכנית, אשר נועדות לקורא התוכנית (תיעוד). נציג כאן שתי דרכים לכתוב הערות:

◆ הערה אשר מתפרשת על פני כמה שורות ניתן לרשום בין הסימנים /\* ... \*/ תוכן ההערה ... למשל שלוש השורות הראשונות בדוגמה זו הן הערה המבהירה לקורא מהי מטרת התוכנית.

◆ הערה אשר מתפרשת על פני שורה בודדת ניתן לרשום אחרי הסימנים //

למשל ההערות המופיעות בשתי השורות האחרונות בדוגמה זו מסייעות לקורא לשייך את הסוגריים המסולסלים לתחומים השונים.

אין חובה לכלול הערות בתוכנית, אך הן תורמות תרומה משמעותית לקריאות התוכנית ולכן חשוב להוסיפן.

---

### **שאלה 3.1**

פתחו אלגוריתם המציג על המסך את שמכם, וישמו אותו בשפת C#. למשל, אם השם הוא יאיר, הפלט יהיה: Yair

### **שאלה 3.2**

פתחו אלגוריתם המציג על המסך את שמכם מוקף במסגרת של כוכביות, וישמו אותו בשפת C#.

למשל אם השם הוא יאיר הפלט יהיה :

```
*****  
*Yair*  
*****
```

**הדרכה:** פעולת WriteLine מציגה שורה אחת על המסך. בתוכנית זו עליכם להציג 3 שורות.

כזכור, המחשב קולט נתונים באמצעות אמצעי קלט. אמצעי קלט נפוץ הוא לוח המקשים – המקלדת.  
בפתרון הבעיה הבאה, נראה אלגוריתם הקולט נתונים ומציגם כפלט.

## הצ'יה 2

**מטרת הבעיה ופתרונה:** שימוש במשתנים, הוראות קלט ופלט למשתנים, תוכנית הכוללת כמה משפטים ומעקב ראשון אחרי ביצוע של תוכנית.

פתחו אלגוריתם שהקלט שלו הוא שני מספרים שלמים (המופרדים ברווח), והפלט שלו הוא ההודעה: "שני המספרים שנקלטו הם: " ומתחתיה שני המספרים. ישמו את האלגוריתם בתוכנית בשפת C#.

**שימו ♥:** מספר הקלטים האפשריים בבעיה זו הוא רב (ולמעשה אינסופי) כיוון שקלט יכול להיות כל זוג מספרים, למשל: 7 ו-5, או 20 ו-2. האלגוריתם המבוקש צריך לתת את הפלט הנכון עבור כל זוג מספרים שהוא, כלומר, הוא צריך להיות אלגוריתם כללי.

### חלוקה לתת-משימות

את המשימה שיש לפתור בבעיה ניתן לחלק לשלוש תת-משימות:

1. קליטת שני מספרים שלמים
  2. הצגת ההודעה "שני המספרים שנקלטו הם:"
  3. הצגת שני המספרים שנקלטו בשורה חדשה
- ❓ היכן ישמור המחשב את הנתונים הנקלטים?

הנתונים הנקלטים נשמרים בתאי הזיכרון המכונים **"תאי משתנים"** או בקיצור **"משתנים"**.

**משתנה** (variable) הוא תא זיכרון אשר במהלך ביצוע אלגוריתם ניתן לשמור בו ערך ולקרוא את הערך השמור בו. למידע השמור בתוך המשתנה קוראים **ערך המשתנה**. פנייה למשתנה נעשית באמצעות שמו, שהוא **שם המשתנה**.

בבעיה זו הקלט הוא שני נתונים (שני מספרים), ולכן נגדיר שני משתנים שנקרא להם: **num1** ו-**num2**. בכל אחד מהמשתנים יישמר **ערך אחד בלבד**. כאשר אנו בוחרים שמות למשתנים כדאי לבחור שמות משמעותיים, שיעידו על תפקידיהם של המשתנים. בחירת שמות משמעותיים מסייעת לקריאות התוכנית ולבהירותה, בדיוק כמו כתיבת הערות בתוכנית.

בשפת C# מקובל להתחיל שם של משתנה באות קטנה. אם שמו מורכב מכמה מילים הן נכתבות צמודות זו לזו ללא רווחים; החל מהמילה השנייה תיכתב כל אחת מהן באות גדולה בתחילתה.

**האלגוריתם** לפתרון הבעיה ישתמש בשני המשתנים שבחרנו:

1. קלוט שני מספרים שלמים במשתנים num1 ו-num2

2. הצג כפולט אג ההודעה: "שני המספרים שנקלטו הם:"  
3. הצג כפולט בשורה גדישה אג ערך המשתנה num1 ולא ערך המשתנה num2

## יישום האלגוריתם

ב-C# יש להצהיר על כל משתנה לפני השימוש בו. הצהרה נעשית בכתיבת **טיפוס** המשתנה ושמו של המשתנה.

**טיפוס** (type) הוא סוג של ערכים. למשל, כל המספרים השלמים הם מטיפוס שלם וכל המספרים הממשיים הם מטיפוס ממשי.

כיוון שבחרנו במספרים שלמים, נגדיר כי שני המשתנים הם מטיפוס שלם. זאת נעשה בשימוש במילה `int` (קיצורה של המילה האנגלית `integer`, שמשמעותה מספר שלם), למשל כך:

```
int num1;
```

בהצהרת משתנים בתוכנית C# ניתן להצהיר על כמה משתנים מאותו טיפוס ברשימת שמות אחת שלפניה יופיע שם הטיפוס ואחריה הסימן נקודה פסיק. שמות המשתנים יופרדו בפסיקים. נצהיר על משתנים באותה השורה רק כאשר יש להם תפקיד דומה וניתן להסביר את תפקידם בהערת תיעוד אחת משותפת.

למשל נוכל להצהיר על שני המשתנים כך:

```
int num1, num2; // שני משתנים לשמירת מספרים שלמים הנקלטים מהמשתמש
```

הצהרה על משתנה יכולה להיות בכל מקום בתוך תחום הפעולה, לפני ההתייחסות הראשונה אליו. עם זאת, כדי שהתוכנית תהיה בהירה וקריאה מקובל לרכז את כל הצהרות המשתנים ביחד בתחילת התחום.

## הוראה 1 (קלט)

כדי ליישם את הוראה 1 עלינו ללמוד כיצד לקלוט.

קליטת מספר שלם נעשית באמצעות ההוראה:

```
int num = int.Parse(Console.ReadLine());
```

הוראה זו קולטת ערך שלם ומכניסה אותו לתוך המשתנה ששמו כתוב בצד שמאל. למעשה, זו הוראה מורכבת למדי, המפעילה כמה פעולות. הפעולה הראשונה שמופעלת היא `ReadLine` של המחלקה `Console`, הקוראת שורה מהמקלדת. השורה הזאת מועברת למחלקה האחראית למספרים השלמים `int`. מחלקה זו בתורה מפעילה על השורה שקיבלה את הפעולה `Parse` והיא מפרשת את רצף הסימנים שנקרא מהמקלדת ומתרגמת אותו לערך שלם.

בעת הרצת התוכנית, כאשר תתבצע פעולה זו, **תעצור** התוכנית ותחכה לקלט מתאים מהמשתמש. נהוג להוסיף הנחיה למשתמש, מעין הדרכה מדוע נעצרה התוכנית ולמה היא מצפה. את ההנחיה אפשר להציג על המסך באמצעות הוראת פלט שנכתבת לפני הוראת הקלט. בהוראת פלט כזו נעדיף שלא לעבור לשורה הבאה בסיום הדפסת ההודעה על המסך. לשם כך לא נשתמש בפעולה `WriteLine` שהשתמשנו עד כה, אלא בפעולה `Write`. גם היא פעולת פלט, בדומה ל-`WriteLine`, אך היא אינה גורמת למעבר לשורה הבאה במסך.

לכן, את הוראה 1 ניישם במשפטי הקלט הבאים:

```
Console.Write("Enter first number: ");
num1 = int.Parse(Console.ReadLine());
Console.Write("Enter second number: ");
num2 = int.Parse(Console.ReadLine());
```

## הוראה 2 (פלט ערכי המשתנים)

גם הפעם ברצוננו להדפיס הודעת פלט על המסך ונרצה לשלב בה את ערכי המשתנים. לשם כך נשתמש בהוראת הדפסה, ונסמן בגוף ההודעה להדפסה היכן ישתלבו ערכי המשתנים. הסימונים הם רשימה של מספרים בסדר עולה, החל מ-0, עטופים בסוגריים מסולסלים {}. לאחר ההודעה להדפסה (התחומה בגרשיים כפולים), נוסיף סימן פסיק (,) ואחריו תופיע רשימת המשתנים שאת ערכם ברצוננו לשלב במשפט הפלט, והם מופרדים בפסיקים. כך:

```
Console.WriteLine("The two numbers are: {0} {1} ", num1, num2);
```

**שימו ♥:** ניתן לשלב בהודעה ערכים של משתנים רבים, אך מספר המשתנים ברשימה חייב להיות שווה למספר הסימונים בהודעת ההדפסה.

אם ברצוננו להציג על המסך רק ערך של משתנה יחיד, ללא שילובו בהודעה כלשהי, ניתן לעשות זאת בצורה פשוטה יותר, באמצעות אחת מהוראות הפלט Write או WriteLine, למשל כך:

```
Console.WriteLine(num1);
```

נשלים את גוף התוכנית לכדי תוכנית מלאה, בדומה למה שהודגם בפתרון בעיה 1:

```
/*
    התוכנית קולטת שני מספרים שלמים
    ומציגה את ערכם כפלט
*/
using System;
public class ReadWrite
{
    public static void Main ()
    {
        // הצהרה על משתנים בתוכנית
        int num1, num2; //המשתמש הנקלטים מהמשתמש
        // הוראות התוכנית
        Console.Write("Enter first number: ");
        num1 = int.Parse(Console.ReadLine());
        Console.Write("Enter second number: ");
        num2 = int.Parse(Console.ReadLine());
        Console.WriteLine("The two numbers are: {0} {1}", num1, num2);
    } // Main
} // class ReadWrite
```

שימו ♥ כי בתוכנית שילבנו הערות. בתחילה שילבנו הערה המבהירה מה תפקיד התוכנית כולה, ואחר כך שילבנו הערות המבהירות לקורא מה תפקיד כל חלק בתוכנית.

שילוב ההערות אינו בגדר חובה, אך מומלץ ביותר כדי שהתוכנית תהיה ברורה לכל אדם הקורא אותה.

נעקוב אחר הרצת התוכנית ReadWrite עבור קלט כלשהו.

כאשר ניתנת למחשב סדרה של נתונים הוא קולט אותם משמאל לימין. למשל, אם הקלט עבור התוכנית ReadWrite הוא המספרים: 20 10, אז המספר השמאלי 10 מוקלד ראשון, והמספר הימני 20 מוקלד שני.

הדו-שיח בין המשתמש למחשב בעת ביצוע התוכנית יראה כך (לאחר נתון שהמשתמש מקליד, עליו להקיש על המקש <Enter>):  
המחשב יציג כפלט:

Enter first number:

המשתמש יקליד את הערך 10 שיופיע אף הוא על המסך, בהמשך השורה:

Enter first number: 10

המחשב יציג כפלט:

Enter second number:

המשתמש יקליד את הערך 10 שיופיע אף הוא על המסך, בהמשך השורה:

Enter second number: 20

המחשב יציג כפלט:

The two numbers are: 10 20

## סוף פתרון בעיה 2

כדי להבין היטב את תפקידם של משתנים בתוכנית ואת השפעת הפעולות הקשורות אליהם, נעקוב שוב אחרי ביצוע התוכנית שבפתרון בעיה 2, אך הפעם נתמקד בזיכרון.

המשפט הראשון שמתייחס למשתני התוכנית הוא משפט ההצהרה על המשתנים:

```
int num1, num2;
```

כתוצאה מביצוע המשפט הזה מוקצים עבור num1 ועבור num2 תאי זיכרון שתוכנם אינו ידוע. אפשר לצייר זאת כך:

num1:  num2:

המשפטים הבאים בתוכנית הם משפטי הקלט (בצירוף הנחיות):

```
Console.Write("Enter first number: ");  
num1 = int.Parse(Console.ReadLine());  
Console.Write("Enter second number: ");  
num2 = int.Parse(Console.ReadLine());
```

תוצאת הביצוע של משפט קלט היא שמירת ערך במשתנה.

למשל, אם בתגובה להודעה Enter first number הקליד המשתמש את המספר 10, הרי אחרי ביצוע משפט הקלט הראשון ערכו של num1 יהיה 10. אם בתגובה להודעה Enter second number הקליד המשתמש את המספר 20, אז אחרי ביצוע משפט הקלט השני ערכו של num2 יהיה 20. נוכל לצייר זאת כך:

num1:  num2:

לאחר שמירת הנתונים בזיכרון יבצע המחשב את המשפט הבא, האחרון בתוכנית, ויצג כפלט את ההודעה, בצירוף הערכים השמורים בתוך המשתנים:

The two numbers are: 10 20

אין לפעולת הפלט כל השפעה על ערכם של המשתנים.

בתום ביצוע המשפט האחרון בתוכנית "ישוחררו" תאי הזיכרון שהוקצו בתחילת הרצת התוכנית, כלומר תאי זיכרון אלה כבר לא שייכים לתוכנית ואסור לה להשתמש בהם יותר. אם נבקש להריץ שוב את התוכנית, יוקצו שוב תאי זיכרון, ואלה אינם בהכרח אותם תאי הזיכרון שהוקצו בהרצה הקודמת.

**שימו ♥ :** מאחר שהגדרנו את שני המשתנים מטיפוס `int`, הם יכולים להכיל מספרים שלמים בלבד.

### שאלה 3.3

א. תארו את הדו-שיח בין המשתמש למחשב בעת ביצוע התוכנית `ReadWrite` (שבפתרון בעיה 2), עבור הקלט 10 5. זכרו שהמספר 5 מוקלד ראשון.

ב. נתייחס להרצת התוכנית `ReadWrite` עבור הקלט 10 5. מה יהיו הערכים השמורים במשתנים `num1` ו-`num2` בתחילת הביצוע? מה יהיו הערכים השמורים בהם לאחר כל משפט קלט? ולאחר ביצוע משפט הפלט האחרון?

ג. תארו את הדו-שיח בין המשתמש למחשב בעת ביצוע התוכנית `ReadWrite` עבור הקלט 5 10.

ד. ענו על אותן שאלות מסעיף ב, אך הפעם התייחסו לביצוע התוכנית `ReadWrite` עבור הקלט 5 10.

### שאלה 3.4

בחרו שני משתנים, הראשון לשמירת מספר הבנות בכיתה והשני לשמירת מספר הבנים בכיתה. לכל משתנה בחרו שם מתאים והצהירו עליו ב-C#, תוך ציון טיפוסו, ותוך תיעוד תפקידו בהערה מתאימה.

**שימו ♥ :** פיתוח ויישום אלגוריתם יעשה תמיד על פי השלבים הבאים, כפי שנעשה בפתרון בעיה 2:

1. בחינת דוגמאות קלט שונות והבנת הקשר הדרוש בין הקלט לפלט.
2. חלוקת המשימה לתת-משימות.
3. בחירת משתנים – תפקיד, שם וטיפוס לכל משתנה.
4. כתיבת האלגוריתם.
5. יישום האלגוריתם על ידי תוכנית.

אנו מקפידים על פיתוח ועל יישום של אלגוריתם בשלבים. אמנם בפרק זה האלגוריתמים לפיתוח הם קצרים, אך חשוב כבר עכשיו להבחין בשלבים השונים. ככל שנתקדם יותר בחומר הלימוד נפתח אלגוריתמים מורכבים יותר, וחשיבות הפיתוח בשלבים תתברר יותר ויותר.

לאחר כתיבת התוכנית חשוב לבצע מעקב לבדיקתה, כפי שבצענו בפתרון בעיה 2. מעקב זה מסייע בבדיקת נכונות התוכנית, ולעתים נמצא בעזרתו שגיאות שנצטרך לתקן.

### שאלה 3.5

פתחו וישמו בשלבים אלגוריתם שיקבל כקלט שלושה מספרים שלמים, והפלט שלו יהיה המספר השני שנקלט. למשל, עבור הקלט: 3 5 9 הפלט הדרוש הוא 5.

### שאלה 3.6

נתונה התוכנית הבאה:

```
using System;
public class InOut
{
    public static void Main ()
    {
        int num1;
        int num2;
```

```

Console.WriteLine("Enter first number: ");
num1 = int.Parse(Console.ReadLine());
Console.WriteLine("Enter second number: ");
num2 = int.Parse(Console.ReadLine());
Console.WriteLine(num1);
Console.WriteLine(num2);
Console.WriteLine("Enter another one: ");
num2 = int.Parse(Console.ReadLine());
Console.WriteLine(num1);
Console.WriteLine(num2);
}
}

```

- א. תארו את הדו-שיח בין המשתמש למחשב בעת ביצוע התוכנית InOut, כאשר הקלט עבור שני משפטי הקלט הראשונים הוא 3 2, והקלט עבור משפט הקלט השלישי הוא 5.
- ב. תארו את הדו-שיח בין המשתמש למחשב בעת ביצוע התוכנית InOut, כאשר הקלט עבור שני משפטי הקלט הראשונים הוא 5 2, והקלט עבור משפט הקלט השלישי הוא 3.

### שאלה 3.7

פתחו וישמו בשלבים אלגוריתם שהקלט שלו הוא שלושה מספרים שלמים, והפלט שלו הוא שתי שורות, ובכל שורה שניים ממספרי הקלט: בשורה הראשונה המספר השני והמספר השלישי מסודרים לפי סדר קליטתם, ובשורה השנייה המספר הראשון והמספר השני מסודרים בסדר הפוך לסדר קליטתם.

### שאלה 3.8

פתחו וישמו בשלבים אלגוריתם שהקלט שלו הוא שלושה מספרים שלמים המהווים סדרה. הפלט שלו הוא סדרה של שישה מספרים שבה משוכפל כל אחד מנתוני הקלט כמספר הפעמים המתאים למקומו הסידורי בסדרת הקלט. למשל, עבור הקלט: 8 3 6, הפלט הוא:

8 3 3 6 6 6

הנתון הראשון מופיע פעם אחת, השני פעמיים והשלישי שלוש פעמים.

## 3.2 הוראת השמה

בסעיף הקודם הכרנו אלגוריתמים (ותוכניות) למחשב שקולטים נתונים ונותנים פלט. אך קלט ופלט הם רק מרכיב אחד של אלגוריתמים למחשב. אלגוריתמים למחשב מיועדים בדרך כלל לביצוע חישובים ועיבודים שונים, בנוסף לקלט ולפלט. בסעיף זה נראה אלגוריתמים ראשונים המבצעים חישובים.

### קצ'ה 3

**מטרת הבעיה ופתרונה:** הצגת הוראת השמה.

פתחו וישמו בשלבים אלגוריתם שהקלט שלו הוא שני מספרים שלמים חיוביים, המציינים אורך ורוחב של מלבן, והפלט שלו הוא שטחו והיקפו של המלבן.

בבעיה זו, כמו בבעיות הקודמות שפתרנו, מספר הקלטים הוא רב. אם האורך והרוחב של המלבן יכולים להיות כל זוג מספרים שלמים חיוביים, הרי יש בעצם אינסוף קלטים אפשריים.

## בדיקת דוגמאות

זכור, לפני שאנו ניגשים לכתובת האלגוריתם כדאי לוודא שאנו מבינים את המשימה על ידי כך שנבחן את הפלט עבור דוגמאות קלט מגוונות:

### שאלה 3.9

ציינו את הפלט עבור כל אחד מהקלטים הבאים:

א. 5 10

ב. 12 3

## חלוקה לתת-משימות

מהן התת-משימות של האלגוריתם?

תחילה יש לקלוט את הנתונים, אחר כך יש לחשב את השטח ואת ההיקף, ולבסוף יש להציג כפלט את תוצאת החישוב. נתאר זאת בחלוקה הבאה לתת-משימות:

1. קליטת שני מספרים שלמים המייצגים אורך ורוחב של מלבן
2. חישוב שטח המלבן
3. חישוב היקף המלבן
4. הצגת תוצאת החישוב

## בחירת משתנים

כדי לשמור את אורכו ואת רוחבו של המלבן נשתמש בשני משתנים מטיפוס שלם, שנקרא להם length ו-width בהתאמה (זכרו! אנו מקפידים על בחירת שמות משמעותיים). בנוסף, נבחר את המשתנים area ו-perimeter מטיפוס שלם, לשמירת תוצאות חישובי השטח וההיקף.

בסך הכול בחרנו ארבעה משתנים מטיפוס שלם:

- length – ישמור את אורך המלבן.
- width – ישמור את רוחב המלבן.
- area – ישמור את שטח המלבן.
- perimeter – ישמור את היקף המלבן.

## האלגוריתם

את תת-משימה 1 נבצע על ידי הוראת קלט מתאימה:

קלט אורך ורוחב של מלבן 2-length 2-width

? לאחר קליטת הנתונים עלינו לחשב שטח והיקף. כיצד נבצע את החישובים?

חישוב שטח מתקבל באמצעות הביטוי:  $length * width$   
חישוב היקף מתקבל באמצעות הביטוי:  $(width + length) * 2$

שימו לב: C# מציינים פעולת כפל באמצעות התו \*

? היכן נשמור את תוצאות החישובים?

המחשב מסוגל לבצע חישוב של ביטוי חשבוני ולשים (לשמור) את תוצאת החישוב במשתנה. הביטוי החשבוני יכול לכלול ערכים המצוינים במפורש (למשל המספר 2) או ערכים השמורים במשתנים. הדרך להורות למחשב לשמור את תוצאת הביטוי במשתנה היא באמצעות הוראת השמה.



אם כך, נכלול באלגוריתם הוראות השמה לשמירת תוצאות החישוב במשתנים area ו-perimeter.

האלגוריתם לפתרון הבעיה יכלול שתי הוראות השמה ויהיה:

1. קלט אורך ורוחב של מלבן ב-length ו-width
2. גשב את שטח המלבן על ידי  $\text{length} * \text{width}$  והשם (שמור) את התוצאה ב-area
3. גשב את היקף המלבן על ידי  $2 * (\text{width} + \text{length})$  והשם (שמור) את התוצאה ב-perimeter
4. הצג כפלט את ערך area ואם ערך perimeter

## יישום האלגוריתם

קוד ניגש ליישום האלגוריתם בתוכנית C#.

יישום הוראות ההשמה יהיה על ידי שני המשפטים הבאים:

```
area = length * width;
perimeter = (width + length) * 2;
```

אלה הם משפטי השמה. כל משפט מורה על ביצוע חישוב ועל השמת תוצאתו במשתנה.

ביצוע המשפט הראשון, יביא לחישוב מכפלת ערכו של width בערכו של length, ולהשמת התוצאה במשתנה area.

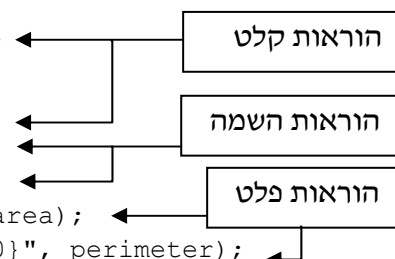
ביצוע המשפט השני, יביא לחישוב הסכום של ערכו של width בערכו של length ולהכפלתו ב-2, והשמת התוצאה במשתנה perimeter.

הנה יישום האלגוריתם כולו בשפת C#:

```
Console.Write("Enter length: ");
length = int.Parse(Console.ReadLine());
Console.Write("Enter width:");
width = int.Parse(Console.ReadLine());
area = length * width;
perimeter = (width + length) * 2;
Console.WriteLine("The area is: {0}", area);
Console.WriteLine("The perimeter is: {0}", perimeter);
```

נשלים את משפטי התוכנית לתוכנית מלאה:

```
/*
 * התוכנית מחשבת את שטחו ואת היקפו של מלבן
 */
using System;
public class Rectangle
{
    public static void Main()
    {
        int length, width, area, perimeter;
        Console.Write("Enter length: ");
        length = int.Parse(Console.ReadLine());
        Console.Write("Enter width:");
        width = int.Parse(Console.ReadLine());
        area = length * width;
        perimeter = (width + length) * 2;
        Console.WriteLine("The area is: {0}", area);
        Console.WriteLine("The perimeter is: {0}", perimeter);
    } // Main
} // class Rectangle
```



## מעקב

נעקוב עתה אחר מהלך הרצת התוכנית Rectangle עבור הקלט 3 5 :

בתחילת ההרצה ערכי כל המשתנים אינם ידועים.

בעקבות ביצוע משפט הקלט הראשון תוצג ההודעה "Enter length: " ואז ימתין המחשב לקלט מן המשתמש. המשתמש יקיש 5 ו-Enter. תוצאת הוראת הקלט הראשונה תהיה שמירת הערך 5 במשתנה length.

לאחר מכן תוצג ההודעה "Enter width: " ושוב ימתין המחשב לקלט. המשתמש יקיש 3 ו-Enter. תוצאת הוראת הקלט השנייה תהיה שמירת הערך 3 במשתנה width.

בביצוע משפט ההשמה הראשון יחושב הערך 15, שהוא ערך הביטוי  $5*3$ , המופיע בצד ימין של המשפט. ערך זה יישמר במשתנה area, ששמו כתוב בצד שמאל של המשפט.

בביצוע משפט ההשמה השני יחושב הערך 16, שהוא ערך הביטוי  $(3+5)*2$ , המופיע בצד ימין של המשפט. ערך זה יישמר במשתנה perimeter, ששמו כתוב בצד שמאל של המשפט.

**שימו ♥ :** ערכיהם של length ושל width לא ישתנו בעקבות ביצוע משפטי ההשמה! משפט השמה משפיע רק על המשתנה שבצד שמאל של המשפט. משתנים המעורבים בביטוי שבצד ימין של המשפט אינם מושפעים ממנו.

בעקבות ביצוע שני משפטי הפלט האחרונים יתקבל הפלט :

```
The area is: 15
The perimeter is: 16
```

### סוף פתרון בעיה 3

בפתרון בעיה 3 הכרנו את פעולת ההשמה :

בפעולת **השמה**, המחשב מבצע חישוב כלשהו וְשֵׁם את התוצאה בתוך משתנה.

פעולת ההשמה מיושמת ב-C# כך: `y = expression;` משפט זה מורה על השמת ערך הביטוי expression בתוך המשתנה y.

הביטוי expression יכול להיות ביטוי פשוט (למשל מספר או שם של משתנה) או ביטוי המורכב מפעולות חשבוניות שונות.

דוגמאות :

♦ `a = 7;` פירושו: "השם את הערך 7 במשתנה ששמו a". לאחר ביצוע פעולה זו, ערכו של a יהיה 7.

♦ `a = b;` פירושו: "השם את ערכו של המשתנה b בתוך המשתנה a". למשל, אם ערכו של b לפני ביצוע המשפט הוא 3, אז לאחר ביצוע המשפט ערכו של a יהיה 3. ערכו של b לא ישתנה אלא יישאר 3.

♦ `a = b * c;` פירושו: "הכפל את ערכיהם של b ושל c והשם את התוצאה ב-a". למשל, אם לפני ביצוע המשפט ערכיהם של b ושל c הם 9 ו-5 בהתאמה, אז לאחר ביצוע המשפט יהיה ערכו של a שווה ל-45 (וערכיהם של b ושל c לא ישתנו).

**שימו ♥:** כיוון שמשתנה יכול להכיל ערך אחד בלבד בכל רגע, אז בעת ביצוע המשפט `y = expression`, ערכו הקודם של `y` הולך לאיבוד, כלומר ערכו של הביטוי `expression` "דורך" על מה שהיה בתוך `y` לפני ביצוע המשפט. אבל כאמור משפט ההשמה אינו משפיע על אף משתנה חוץ מ-`y`.

**שימו ♥:** בביטוי חשבוני מתקיים סדר הקדימויות המוכר של פעולות החשבון. כלומר, לסוגריים עדיפות גבוהה ביותר, עדיפות נמוכה יותר לכפל ולחילוק, ועדיפות נמוכה ביותר לחיבור ולחיסור.

### שאלה 3.10

- נניח שהקלט בעת ביצוע התוכנית `Rectangle` הוא: 10 7.
- מה יהיו ערכי המשתנים `length` ו-`width` לאחר ביצוע משפטי הקלט?
  - מה יהיו ערכי כל המשתנים לאחר ביצוע משפט ההשמה השני?
  - תארו את הדו-שיח בין המחשב למשתמש במהלך הרצת התוכנית.

### שאלה 3.11

- כתבו משפטי השמה לביצוע הפעולות הבאות:
- איפוס המשתנה `a` (איפוס משמעותו השמת הערך 0).
  - השמת תוצאת החישוב  $3 * (729 - 511)$  במשתנה `a`.
  - השמת כפליים מערכו של המשתנה `b` במשתנה `a`.
  - השמת סכום ערכי המשתנים `x` ו-`y` במשתנה `w`.

### שאלה 3.12

נתון קטע תוכנית ובו המשפטים הבאים:

```
Console.Write("Enter first number: ");
a = int.Parse(Console.ReadLine());
Console.Write("Enter second number: ");
b = int.Parse(Console.ReadLine());
c = a + b;
Console.WriteLine(c);
```

תנו שתי דוגמאות קלט שונות שהפלט עבורן הוא 5.

### שאלה 3.13

כתבו סדרה של ארבעה משפטי השמה המבצעים, לפי הסדר הבא:

- השמת הערך 3 במשתנה `a`.
- השמת תוצאת החישוב של הביטוי  $3 * 9$  במשתנה `b`.
- השמת סכום ערכי `a` ו-`b` במשתנה `c`.
- השמת מכפלת ערכי `a` ו-`c` במשתנה `d`.

מהם ערכי ארבעת המשתנים `a`, `b`, `c` ו-`d` בתום ביצוע סדרת המשפטים?

### שאלה 3.14

נתונים משפטי התוכנית הבאים:

```
Console.Write("Enter first number: ");
num1 = int.Parse(Console.ReadLine());
Console.Write("Enter second number: ");
num2 = int.Parse(Console.ReadLine());
Console.Write("Enter third number: ");
num3 = int.Parse(Console.ReadLine());
```

```
diff1 = num1 * num2 - num3;
diff1 = num2 * num3 - num1;
Console.WriteLine(diff1);
Console.WriteLine(diff2);
```

נניח שהקלט במהלך ההרצה הוא: 3 2 3.

א. מה יהיו ערכי המשתנים num1, num2 ו-num3 לאחר ביצוע משפטי הקלט?

ב. מה יהיו ערכי המשתנים diff1 ו-diff2 לאחר ביצוע משפטי ההשמה?

ג. תארו את הדו-שיח בין המחשב למשתמש במהלך משפטי התוכנית.

### שאלה 3.15

פתחו וישמו בשלבים אלגוריתם שהקלט שלו הוא מספר חיובי שלם, המציין אורך צלע של קובייה, והפלט שלו הוא נפח הקובייה ושטח הפנים שלה.

**הדרכה:** אם צלע הקובייה היא  $a$ , הנפח יחושב כ- $a^3$  ושטח הפנים יחושב כ- $6a^2$ .

**שימו ♥:** גם פעולת קלט כוללת למעשה השמה. למשל במשפט:

```
length = int.Parse(Console.ReadLine());
```

מתבצעת פעולת השמה של המספר הנקלט לתוך המשתנה length.

כזכור, כאשר אנו מצהירים על משתנה, מוקצה לו מקום בזיכרון, אך ערכו אינו ידוע. במקרים מסוימים נרצה לתת למשתנה ערך מיד בעת ההצהרה עליו. זאת כדאי לעשות כאשר ידוע לנו כבר בשלב ההצהרה מה צריך להיות ערכו ההתחלתי של משתנה, ואין הוא תלוי בהוראה שצריכה להתבצע מאוחר יותר, כמו הוראת קלט. מתן ערך התחלתי למשתנה נקרא אתחול.

**אתחול של משתנה משמעותו מתן ערך התחלתי למשתנה.**

הסיבה שלא כדאי לנו לדחות אתחול של משתנה, וכדאי לאתחל משתנה מיד כאשר ידוע לנו הערך המתאים לאתחול, היא שאם לא נעשה זאת, אנו עלולים לשכוח לעשות זאת מאוחר יותר. ערכו של המשתנה יישאר לא ידוע, וכאשר ננסה להשתמש בערכו – למשל להדפיסו או לשלבו בביטוי חשבוני, אין לדעת מה יהיה ערכו.

שפת C# מאפשרת לנו לשלב הצהרה והשמה באופן הזה:

```
int x = 3;
```

תוצאת ביצוע ההוראה היא הקצאת מקום בזיכרון עבור המשתנה x והשמת הערך 3 בתוכו. ההוראה הזאת שקולה לרצף ההוראות

```
int x;
```

```
x = 3;
```

גם הוראה כזאת ניתן לכתוב בשפת C#:

```
int x = num1 * num2;
```

וזאת בתנאי ש-num1 ו-num2 הם משתנים מטיפוס שלם, שהוצהרו קודם לכן וערכם ידוע.

אם כך, בעת ההצהרה יכול להופיע ביטוי גם בצד ימין של ההשמה, בתנאי שכל הערכים הכלולים בו כבר ידועים.

בהמשך הפרק נשתמש לעתים במונחים ערך התחלתי ומצב התחלתי:

**הערך ההתחלתי** של משתנה ביחס לתוכנית מסוימת או לקטע תוכנית מסוים, הוא הערך שיש במשתנה מיד לפני ביצוע ההוראה הראשונה בתוכנית או בקטע התוכנית.

**מצב התחלתי** של תוכנית או של קטע תוכנית כולל את ערכם ההתחלתי של כל המשתנים ביחס לאותה תוכנית או לקטע התוכנית.

למשל, אם השורה הראשונה בתוכנית היא `int x = 3;`, אז ערכו ההתחלתי של `x` ביחס לתוכנית הוא כמובן 3. אם השורה הראשונה היא `int x;`, אז ערכו ההתחלתי של `x` אינו ידוע. כאשר אנו בוחנים את ערכו של משתנה ביחס לקטע תוכנית, עלינו לבדוק מהו הערך האחרון שהושם לתוכן, לפני קטע התוכנית. זה יהיה ערכו ההתחלתי של המשתנה ביחס לקטע התוכנית. אם לא התבצעה שום השמה למשתנה לפני קטע התוכנית, הרי ערכו ההתחלתי ביחס לקטע התוכנית אינו ידוע.

### 3.3 טבלת מעקב

בסעיף הקודם ראינו הוראות השמה ראשונות פשוטות ובחנו לראשונה מהלך של השמת ערכים במשתנים בעת התקדמות הביצוע של אלגוריתם מהוראה להוראה.

בסעיף זה נראה הוראות השמה מורכבות יותר, ונציג דרך למעקב שיטתי אחר מהלך ביצוע של אלגוריתם.

## 4 פצ'ה

**מטרת הפצ'ה ופתרונה:** הצגת הוראת השמה אשר בה הערך החדש המושם במשתנה תלוי בערך הנוכחי של המשתנה, והצגת מעקב אחר מהלך ביצוע באמצעות טבלת מעקב.

נתבונן בשתי סדרות המספרים הבאות: 12 4 2 ו-30 10 5.

בשתי הסדרות האיבר הראשון (משמאל) הוא הקטן ביותר, האיבר השני גדול פי שניים מהאיבר הראשון, והאיבר השלישי גדול פי שלושה מהאיבר השני.

פתחו וישמו בשלבים אלגוריתם אשר הקלט שלו הוא מספר שלם חיובי המציין איבר ראשון בסדרה (כדוגמת הסדרות המתוארות) והפלט שלו הוא האיבר השני והשלישי בסדרה בשורות נפרדות.

השתמשו באלגוריתם במשתנה אחד בלבד!

### בדיקת דוגמאות

#### שאלה 3.16

מהו הפלט עבור הקלט 10, ומהו הפלט עבור הקלט 2 ?

### חלוקה לתת-משימות

נחלק לתת-משימות באופן הבא:

1. קליטת מספר המייצג איבר ראשון בסדרה.
2. חישוב האיבר השני בסדרה והצגתו כפלט.
3. חישוב האיבר השלישי בסדרה והצגתו כפלט.

## בחירת משתנים

כיוון שהוטלה המגבלה של שימוש במשתנה אחד, ברור שהמשתנה שנבחר ישמור בכל פעם אחד מאיברי הסדרה. נשתמש במשתנה מסוג שלם ונקרא לו **element**.

## האלגוריתם

את התת-משימה הראשונה נבצע כמובן באמצעות קליטת האיבר הראשון בתוך **element**.

❓ כיצד נחשב באמצעות משתנה אחד בלבד את האיבר השני ואחר כך את האיבר השלישי בסדרה?

בביטוי שבהוראת השמה (כלומר, בצד ימין של ההוראה) אפשר לכלול גם את המשתנה אשר בו מושמת תוצאת החישוב של הביטוי (המשתנה המופיע בצד שמאל של ההשמה). במקרה כזה ערכו של המשתנה לפני ביצוע ההשמה משמש בחישוב ערכו החדש. לדוגמה:

גשג אג ערכו של הביטוי  $2 * \text{num}$  / והשם אג הוגזאה ב-**num**

משמעות הוראה זו היא שערכו של **num** לאחר ביצוע ההוראה יהיה פי 2 מערכו לפני ביצוע ההוראה. אם ערכו של **num** לפני ביצוע ההוראה הוא 5 אז ערכו לאחר הביצוע יהיה 10.

נשתמש בהוראת השמה כזו, שבה מופיע **num** בשני צדי משפט ההשמה. נחשב את האיבר השני על ידי הכפלה ב-2 של הערך השמור ב-**element**, נשים את התוצאה חזרה ב-**element** ונציג כפלט את ערכו. אחר כך נחשב את האיבר השלישי בסדרה באמצעות הכפלה ב-3 של הערך השמור ב-**element**, נשים את התוצאה חזרה ב-**element** ולבסוף נציג שוב את ערכו.

הנה האלגוריתם לפתרון הבעיה:

1. קלט מספר ב-**element** // קליטת איבר ראשון בסדרה
2. גשג:  $2 * \text{element}$  / והשם אג הוגזאה ב-**element** // חישוב האיבר השני בסדרה
3. גשג כפלט אג ערכו של **element**
4. גשג:  $3 * \text{element}$  / והשם אג הוגזאה ב-**element** // חישוב האיבר השלישי בסדרה
5. גשג כפלט אג ערכו של **element**

## יישום האלגוריתם

הנה התוכנית ליישום האלגוריתם שלעיל, ובה יש הערות כדי להבהיר את מטרתו של כל משפט. את מספרי השורות הוספנו לשם נוחות, ונשתמש בהם עוד מעט. הם אינם חלק מהוראות התוכנית!

```
/*
התוכנית נותנת כפלט את איבריה של סדרה בת שלושה איברים
*/
using System;
public class Sequence
{
    public static void Main ()
    {
        int element;
        1. Console.WriteLine("Enter first element: ");
        2. element = int.Parse(Console.ReadLine());
        3. element = 2 * element; // חישוב האיבר השני
        4. Console.WriteLine("The second element is: {0}", element);
        5. element = 3 * element; // חישוב האיבר השלישי
        6. Console.WriteLine("The third element is: {0}", element);
    }
}
```

```

    } // Main
} // class Sequence

```

## מעקב

נעקוב אחר מהלך ביצוע התוכנית עבור דוגמת הקלט 5 :  
 אחרי ביצוע משפט הקלט יהיה ערכו של element שווה ל-5.  
 אחרי ביצוע משפט ההשמה בשורה 3 יהיה ערכו של element שווה ל-10.  
 אחרי ביצוע משפט ההשמה בשורה 5 יהיה ערכו של element שווה ל-30.

הדו-שיח בין המחשב למשתמש יהיה:

```

Enter first element      : מחשב:
5                        : משתמש:
The second element is: 10 : מחשב:
The third element is: 30  : מחשב:

```

## סוף פתרון בעיה 4

בפתרון בעיה 4 ראינו נקודה חשובה לגבי הוראות השמה:

משתנה יכול להופיע משני צדי הוראת השמה, כלומר, גם בתפקיד המשתנה שבו מתבצעת ההשמה, וגם כמשתנה שערכו משמש בביטוי שבצד ימין של ההשמה. בהוראה כזו ערכו החדש של המשתנה תלוי בערכו לפני ביצוע ההוראה.

למשל, ההוראה `counter = counter + 1;` תוסיף 1 לערכו של המשתנה `counter`. אם למשל לפני פעולת ההשמה היה ערכו של `counter` 5, אז אחרי ביצועה יהיה ערכו 6.

## שאלה 3.17

הניחו שערכי המשתנים a ו-b לפני כל אחד ממשפטי ההשמה הבאים הם 3 ו-5, בהתאמה. מהו ערכו של a לאחר ביצוע כל משפט?

- א. `a = 1;`
- ב. `a = a + 1;`
- ג. `a = 2 * a + 3;`
- ד. `a = 2 * a + (a - 3);`
- ה. `a = b;`
- ו. `a = a * b;`
- ז. `a = a + a * b;`

שימו ♥: ערכו של b לא משתנה בעקבות אף אחת מההוראות האלו!

## שאלה 3.18

כתבו משפטי השמה לביצוע ההוראות הבאות. את תוצאת החישוב יש לשמור במשתנה a:

- א. הכפלת ערכו של המשתנה a ב-2.
- ב. החסרת ערך המשתנה b מן המשתנה a.
- ג. הכפלת ערכו של המשתנה a בסכום ערכי המשתנים b ו-c.

### שאלה 3.19

נתון קטע התוכנית הבא :

```
Console.Write("Enter number: ");
a = int.Parse(Console.ReadLine());
Console.Write("Enter number: ");
b = int.Parse(Console.ReadLine());
a = a + b;
Console.WriteLine(a);
a = a - b;
Console.WriteLine(a);
```

פלט התוכנית הוא שני מספרים. הביאו שלוש דוגמאות קלט שונות, אשר עבור כל אחת מהן יהיה ההפרש בין שני מספרי הפלט שווה ל-9.

### שאלה 3.20

האם המשפט `int x = 3 * x;` הוא משפט חוקי? נמקו את תשובתכם.

במהלך ביצוע התוכנית Sequence השתנה שוב ושוב ערכו של element. אמנם הצלחנו לבצע מעקב אחר ערכי המשתנה וגם מעקב אחר הודעות הפלט, אבל שיטת המעקב שהשתמשנו בה עלולה להיות מסורבלת עבור תוכניות ארוכות יותר ומורכבות יותר.

נוכל לעקוב אחר התוכנית באופן שיטתי באמצעות **טבלת מעקב**. בטבלת מעקב נעקוב אחר השינויים בערכי המשתנים ואחר הפלט הקורים במהלך ביצוע משפטי התוכנית.

נדגים את השימוש בטבלת מעקב אחר מהלך ביצוע התוכנית Sequence עבור הקלט 3 :

מספר שורה	המשפט לביצוע	element	פלט
1	Console.Write("Enter first element: ");	?	Enter first element
2	element = int.Parse(Console.ReadLine());	3	
3	element = 2 * element;	6	
4	Console.WriteLine("The second element is: {0}", element);	6	The second element is: 6
5	element = 3 * element;	18	
6	Console.WriteLine("The third element is: {0}", element);	18	The third element is: 18

**טבלת מעקב** משמשת למעקב אחרי ביצוע של תוכנית שלמה או של קטע תוכנית, או של אלגוריתם. בטבלה יש שורה עבור כל הוראה לביצוע, עמודה עבור כל משתנה ועמודה עבור הפלט.

### מבנה אפשרי לטבלת מעקב:

◆ עמודות הטבלה :

- העמודה השמאלית ביותר – מספרי השורות של הוראות התוכנית (לפי מספריהן בתוכנית).
- העמודה השנייה משמאל – "המשפט לביצוע", כלומר, הוראות התוכנית עצמן.
- העמודות שבמרכז הטבלה (מהשלישית משמאל ועד השנייה מימין) – עמודה עבור כל משתנה של התוכנית.
- העמודה הימנית ביותר – "פלט".



#### ♦ שורות הטבלה :

- בעמודת "המשפט לביצוע" יהיו הוראות התוכנית זו אחר זו. נכתוב בטבלה רק הוראות שמשפיעות על ערכם של משתנים. למשל לא נכלול הצהרה על משתנה אם אינה כוללת אתחול.
- בעמודות המשתנים: אם בעקבות ההוראה המתאימה בשורה קיבל המשתנה ערך חדש, נכתוב את ערכו החדש, אחרת נכתוב את ערכו הנוכחי. משתנה שערכו אינו ידוע יסומן בסימן '!'.
- בעמודת הפלט: אם ההוראה המתאימה בשורה היא הוראת פלט, כגון `Console.WriteLine` או `Console.Write`, נכתוב את הפלט המתאים, אחרת המשבצת המתאימה בטבלה תישאר ריקה.

ניתן להגמיש מעט את מבנה הטבלה. למשל ניתן לאחד את שתי העמודות השמאליות ולכתוב את מספר השורה יחד עם המשפט שנמצא בשורה זו. אבל חשוב שתהיה בטבלה שורה לכל הוראה של התוכנית, וחשוב שתהיה עמודה לכל משתנה ועמודה עבור הפלט.

#### שאלה 3.21

בנו טבלת מעקב אחר מהלך ביצוע התוכנית Sequence עבור הקלט 5, וטבלת מעקב עבור הקלט 1.

#### שאלה 3.22

נתון קטע התוכנית הבא :

```
1. c = 0;
2. a = (a + 5) * a;
3. b = b + 2 * a;
4. Console.WriteLine(a);
5. Console.WriteLine(b);
6. Console.WriteLine(c);
```

הניחו שהערכים ההתחלתיים של  $a$ ,  $b$  ו- $c$  (כלומר ערכיהם לפני תחילת ביצוע קטע התוכנית) הם 1, 2 ו-3 בהתאמה.

מלאו את טבלת המעקב עבור קטע התוכנית הנתון :

מספר השורה	המשפט לביצוע	a	b	c	פלט
		1	2	3	
1					
2					
3					
4					
5					
6					

#### שאלה 3.23

בנו טבלת מעקב אחר מהלך הביצוע של קטע התוכנית הבא עבור הקלט 2 3 :

```
1. Console.Write("Enter first number: ");
2. num1 = int.Parse(Console.ReadLine());
3. Console.Write("Enter second number: ");
4. num2 = int.Parse(Console.ReadLine());
5. sum = num1 + num2;
```

```

6. sum = sum + sum;
7. sum = sum + sum;
8. Console.WriteLine(sum);

```

מהי מטרת קטע התוכנית? (כלומר, מה הוא מבצע עבור שני מספרים שלמים כלשהם?)

### שאלה 3.24

כתבו קטע תוכנית ובו משפטי השמה אשר מכפילים את ערכו של המשתנה  $a$  ב-4, ולחיסור פעמיים של ערך המשתנה  $c$  מערכו של המשתנה  $b$ . בכל אחד מן הביטויים של משפטי ההשמה, השתמשו אך ורק בפעולת חיבור אחת או בפעולת חיסור אחת. בסוף קטע התוכנית יוצגו ערכי שלושת המשתנים.

כעת, בחרו ערכים התחלתיים כלשהם למשתנים  $a$ ,  $b$  ו- $c$ , ובנו טבלת מעקב אחר מהלך הביצוע של קטע התוכנית שכתבתם עבור ערכים אלה.

## 3.4 החלפה בין ערכי משתנים

בסעיף זה נראה כיצד לפתור בעיה אלגוריתמית בסיסית, ובפתרונה נוכל להיעזר בעתיד, בתוך אלגוריתמים אחרים. כלומר פתרון יהווה תבנית שבה נוכל להשתמש שוב ושוב בהקשרים שונים.

### קצ'ה 5

**מטרת הבעיה ופתרונה:** חידוד השימוש במשתנים, והצגת שימוש במשתנה עזר.

במשתנים  $a$  ו- $b$  יש ערכים התחלתיים כלשהם. כתבו אלגוריתם ובו הוראות השמה, המבצע החלפה של ערכי המשתנים. כלומר, לאחר ביצוע האלגוריתם יהיה ערכו של  $a$  שווה לערכו ההתחלתי של  $b$  וערכו של  $b$  יהיה שווה לערכו ההתחלתי של  $a$ .

הנה הצעה לפתרון:

1. השם  $a$  -  $b$  אג ערכו  $b$
2. השם  $b$  -  $a$  אג ערכו  $a$

ואחרי יישום כקטע תוכנית מחשב:

1.  $a = b;$
2.  $b = a;$

האם קטע זה משיג את המטרה?

נעקוב אחר מהלך ביצוע קטע התוכנית כאשר ערכו ההתחלתי של  $a$  הוא 5 וערכו ההתחלתי של  $b$  הוא 7:

מספר השורה	המשפט לביצוע	a	b
		5	7
1	$a = b;$	7	7
2	$b = a;$	7	7

לא השגנו את המטרה!

למעשה, "איבדנו" את ערכו של  $a$  בעקבות ביצוע שורה מספר 1.

? מה עלינו לעשות כדי למנוע את איבוד ערכו ההתחלתי של המשתנה a?

נגדיר משתנה נוסף, temp (מלשון temporary, שפירושו זמני, כלומר משתנה זמני), אשר ישמש לשמירת ערכו ההתחלתי של a במהלך ביצוע ההחלפה.

נשמור תחילה את ערכו ההתחלתי של a במשתנה הזמני temp. אחר כך נשים את ערכו של b ב-a, ולבסוף נשים ב-b את הערך השמור ב-temp (הלוא הוא ערכו ההתחלתי של a).

**האלגוריתם** לפתרון הבעיה יהיה:

1. השם temp-2 אגזיכיל a
2. השם a-2 אגזיכיל b
3. השם b-2 אגזיכיל temp

ואחרי יישומו, נקבל את קטע התוכנית הבא:

```
1. temp = a;  
2. a = b;  
3. b = temp;
```

לשם המחשה נוכל לדמיין מצב שבו שמנו את הסוכר בכלי של המלח ואת המלח בכלי של הסוכר, על מנת להחליף ביניהם נהיה חייבים להשתמש בכלי עזר!

### סוף פתרון בעיה 5

בבעיה זו למדנו שכדי להחליף ערכים של שני משתנים יש להשתמש במשתנה נוסף, שיעזור בביצוע ההחלפה, משתנה כזה נקרא משתנה עזר:

**משתנה עזר** הוא משתנה שנועד לסייע בביצוע משימה כלשהי.

### שאלה 3.25

א. בנו שתי טבלאות מעקב אחר מהלכי ביצוע שני הפתרונות שהוצעו לבעיה 5, עבור הערכים ההתחלתיים 1 ו-2 במשתנים a ו-b: טבלה אחת עבור הפתרון השגוי של הבעיה וטבלה אחת עבור הפתרון הנכון של הבעיה.

- ב. הביאו שתי דוגמאות קלט שונות אשר עבור כל אחת מהן יהיה פלט הפתרון השגוי 5.
- ג. האם תיתכן דוגמת קלט שעבורה יהיה פלט הפתרון השגוי 6? 5? נמקו.

### שאלה 3.26

נתון קטע התוכנית הבא:

```
1. Console.WriteLine("Enter number: ");  
2. a = int.Parse(Console.ReadLine());  
3. Console.WriteLine("Enter number: ");  
4. b = int.Parse(Console.ReadLine());  
5. Console.WriteLine("Enter number: ");  
6. c = int.Parse(Console.ReadLine());  
7. temp = a;  
8. a = b;  
9. b = c;  
10. c = temp;  
11. Console.WriteLine(a);  
12. Console.WriteLine(b);  
13. Console.WriteLine(c);
```

- א. בנו טבלת מעקב אחר מהלך הביצוע של קטע התוכנית עבור הקלט 1 2 3.
- ב. הביאו דוגמת קלט שהפלט המתקבל עבורה הוא 1 2 3.
- ג. מהי מטרת קטע התוכנית?
- ד. האם ניתן להשיג את מטרת קטע התוכנית ללא משפטי השמה כלל? אם כן, כיצד?

בעיה 5 עסקה בהחלפה של ערכי משתנים. להעמקה בתבנית **החלפת ערכים בין שני משתנים** פנו לסעיף התבניות המופיע בסוף הפרק.

### שאלה 3.27

מטרת קטע התוכנית הבא היא הצגת נתוני הקלט בסדר **הפוך** לסדר קליטתם:

```
1. Console.Write("Enter number: ");
2. a = int.Parse(Console.ReadLine());
3. Console.Write("Enter number: ");
4. b = int.Parse(Console.ReadLine());
5. Console.Write("Enter number: ");
6. c = int.Parse(Console.ReadLine());
7. a = c;
8. c = a;
9. Console.Write(a);
10. Console.Write(b);
11. Console.Write(c);
```

- א. הביאו שתי דוגמאות קלט שונות שעבור כל אחת מהן יוצג הפלט הדרוש.
- ב. הביאו דוגמת קלט שעבורה **לא** יוצג הפלט הדרוש.
- ג. תקנו את הקטע בלי לשנות את משפטי הקלט והפלט, כלומר, השלימו רק את השורות החסרות בקטע התוכנית שלהלן:

```
Console.Write("Enter number: ");
a = int.Parse(Console.ReadLine());
Console.Write("Enter number: ");
b = int.Parse(Console.ReadLine());
Console.Write("Enter number: ");
c = int.Parse(Console.ReadLine());
```

השלימו כאן

```
Console.WriteLine(a);
Console.WriteLine(b);
Console.WriteLine(c);
```

שאלה 3.27 עסקה בהיפוך סדרת איברים. להעמקה בתבנית **היפוך סדר האיברים בסדרה** פנו לסעיף התבניות המופיע בסוף הפרק.

## 3.5 טיפוסים

בפתרון הבעיות שהוצגו עד עתה השתמשנו במספרים שלמים. יש בעיות שכדי לפתור אותן יש להשתמש במספרים שאינם בהכרח שלמים. בסעיף זה נראה דוגמאות לעיבוד מספרים ממשיים (באנגלית real), שהם ערכים מטיפוס ממשי. בסעיף 3.1 נתנו כבר הגדרה למונח טיפוס. עתה נרחיב אותה באופן הבא:

**טיפוס של ערך (data type)** מגדיר קבוצת ערכים ואת הפעולות שניתן לבצע על הערכים האלה.

כלומר יחד עם הערכים השייכים לטיפוס מסוים יש לציין גם את הפעולות המותרות עליהם. בסעיף זה נראה עיבודים עם שני טיפוסים ערכים: טיפוס שלם, המוכר לנו כבר, וטיפוס ממשי.

ערכים **מטיפוס שלם** הם המספרים השלמים, למשל: 3, 0, 700, -511.  
ערכים **מטיפוס ממשי** הם מספרים ממשיים למשל: 5.0, 17.2, 73.1.  
ערך מטיפוס ממשי כולל תמיד חלק שלם ושבר, למשל, במספר 3.5 החלק השלם הוא 3 והשבר הוא 0.5. במספר 5.0 החלק השלם הוא 5 והשבר הוא 0.

ניתן לבצע את הפעולות החשבוניות המוכרות לנו (חיבור, חיסור, כפל וחילוק) הן על ערכים מטיפוס שלם והן על ערכים מטיפוס ממשי. השימוש בפעולות אלו על ערכים מספריים יוצר ביטוי חשבוני. הזכרנו בסעיף 3.2 שביטוי חשבוני מורכב מפעולות חשבון בין ערכים מספריים מפורשים או בין משתנים השומרים ערכים מספריים. ביטוי חשבוני אף הוא מטיפוס מסוים, שלם או ממשי:

**טיפוס של ביטוי חשבוני** נקבע על פי טיפוס הערכים המפורשים, על פי המשתנים שבו, ועל פי הפעולות שבו. אם נכלל בביטוי לפחות ערך אחד או משתנה מטיפוס ממשי, או שנכללת בו פעולה אשר תוצאתה עשויה להיות מספר לא שלם, אז הביטוי הוא מטיפוס ממשי. רק אם כל המרכיבים של הביטוי הם מטיפוס שלם אז הביטוי הוא מטיפוס שלם, למשל:  $7+6$ ,  $5*3$ , או  $num2-num1$ , כאשר  $num1$  ו- $num2$  הוגדרו כמשתנים שלמים.

**שימו ⚡:** קבוצת הערכים מטיפוס שלם היא בעצם תת-קבוצה של קבוצת הערכים מטיפוס ממשי. בהמשך הסעיף נדגים ונסביר את החשיבות שבהגדרת קבוצת הערכים השלמים כטיפוס נפרד.

## 6 חזרה

**מטרת הבעיה ופתרונה:** הצגת שימוש במשתנים משני הטיפוסים: שלם וממשי. היכרות עם תבנית חישוב ממוצע.

פתחו וישמו בשלבים אלגוריתם שהקלט שלו הוא ארבעה מספרים שלמים, והפלט שלו הוא ממוצע המספרים.

### בדיקת דוגמאות

נוודא כי הבעיה מובנת לנו, בכך שנבחן את הפלט עבור שתי דוגמאות קלט שונות:

- ♦ עבור הקלט 40 30 20 10 הפלט הוא 25.
- ♦ עבור הקלט 14 13 12 11 הפלט הוא 12.5.

### חלוקה לתת-משימות

נבצע את החלוקה הבאה לתת-משימות:

1. קליטת ארבעה מספרים שלמים.

2. חישוב סכום ארבעת המספרים.
3. חלוקת הסכום ב-4.
4. הצגה של תוצאת החילוק כפלט.

## בחירת משתנים

אילו טיפוסים מתאימים לבעיה זו?

נגדיר ארבעה משתנים שבהם ייקלטו נתוני הקלט: num1, num2, num3 ו-num4. כיוון שידוע שנתוני הקלט הם מספרים שלמים, משתנים אלה יהיו מטיפוס שלם. נגדיר משתנה נוסף sum, והוא ישמור את סכום ארבעת נתוני הקלט. גם משתנה זה יהיה מטיפוס שלם.

נותר לנו להגדיר את המשתנה אשר ישמור את ממוצע ארבעת המספרים. נקרא לו average. ראינו בדוגמת הקלט/פלט השנייה שבחנו שייתכן שמשתנה זה ישמור ערך אשר איננו מספר שלם. לכן נגדיר את average כמשתנה מטיפוס ממשי.

ובסך הכול נקבל:

num1, num2, num3, num4 – מטיפוס שלם, ישמרו את נתוני הקלט.

sum – מטיפוס שלם, ישמור את סכום נתוני הקלט.

average – מטיפוס ממשי, ישמור את ממוצע נתוני הקלט.

**האלגוריתם** לפתרון הבעיה יהיה:

1. קלט ארבעה מספרים - num1, num2, num3, num4.
2. שלב אל סכום ארבעת המספרים והשם אל הגזאה - sum.
3. שלב אל מחוצה המספרים, של ידוי sum/4, והשם אל הגזאה - average.
4. הצג כפלט אל ערכו של average.

## יישום האלגוריתם

כדי להגדיר בשפת C# משתנה מטיפוס ממשי, יש להצהיר עליו כעל משתנה מטיפוס double, כך:

```
double average;
```

היישום של הוראה 3 משתמש במשפט השמה הכולל ערכים שלמים וממשיים. כיצד מתפרשת הוראה כזאת ב-C#?

במשתנה מטיפוס ממשי ניתן לבצע השמה של ביטוי מטיפוס ממשי או מטיפוס שלם. אם הביטוי הוא מטיפוס שלם ערכו מומר לממשי לפני ביצוע ההשמה.

למשל, נניח ש-x משתנה מטיפוס ממשי, ונתבונן במשפט ההשמה:  $x = 3 + 4$ . הביטוי בצד ימין של ההשמה הוא ביטוי חשבוני שמורכב רק מערכים שלמים (3 ו-4). לכן הביטוי הזה הוא מטיפוס שלם וערכו 7. לפני ביצוע ההשמה ערכו מומר לערך הממשי המקביל 7.0, ובעקבות ההשמה ערכו של x הוא 7.0.

את הוראה 3 היינו רוצים ליישם במשפט השמה כזה:

```
average = sum / 4;
```

מאחר שכל המרכיבים של הביטוי sum/4 הם שלמים, הרי הטיפוס של הביטוי כולו הוא שלם (בפרק הבא נלמד מה משמעות פעולת החלוקה עבור ערכי הטיפוס השלם). אבל אנו מעוניינים לחשב את הביטוי כמספר ממשי ולבצע השמה של תוצאת הביטוי (הממשית) בתוך משתנה מטיפוס ממשי. למשל, אם ערכו של sum הוא 10, אנו מעוניינים לשים את הערך הממשי 2.5 במשתנה average.

נוכל להשיג זאת אם נבקש להתייחס באופן זמני אל אחד ממרכיבי הביטוי כאל ממשי. בכך נגרום לערך הביטוי כולו להיחפז לממשי. כלומר אנו מבקשים להמיר את ערך המשתנה sum לממשי, רק לצורך חישוב הביטוי. פעולת ההמרה (casting) ב-C# מתבצעת כך: כתיבת שם הטיפוס (במקרה זה double) בתוך סוגריים, משמאל למשתנה שרוצים להמיר (במקרה זה sum). נרחיב עוד על המרה בפרק הבא.

לכן הדרך הנכונה ליישם ב-C# את הוראה 3 היא במשפט ההשמה הבא:

```
average = (double) sum / 4;
```

הנה התוכנית השלמה:

```
/*
התוכנית מחשבת ממוצע של ארבעה ערכים
*/
using System;
public class FourNumbersAverage
{
    public static void Main ()
    {
        int num1, num2, num3, num4;    // ארבעת נתוני הקלט
        int sum;                        // סכום נתוני הקלט
        double average;                // ממוצע נתוני הקלט
        1. Console.WriteLine("Enter first number: ");
        2. num1 = int.Parse(Console.ReadLine());
        3. Console.WriteLine("Enter second number: ");
        4. num2 = int.Parse(Console.ReadLine());
        5. Console.WriteLine("Enter third number: ");
        6. num3 = int.Parse(Console.ReadLine());
        7. Console.WriteLine("Enter fourth number: ");
        8. num4 = int.Parse(Console.ReadLine());
        9. sum = num1 + num2 + num3 + num4;
        10. average = (double) sum / 4;
        11. Console.WriteLine("The average is: {0}", average);
    } // Main
} // class FourNumbersAverage
```

בנה טבלת מעקב אחר ביצוע מהלך התוכנית עבור הקלט 1 2 3 5:

מספר שורה	המשפט לביצוע	num1	num2	num3	num4	sum	average	פלט
1	Console.WriteLine("...");	?	?	?	?	?	?	Enter first number:
2	num1 = int.Parse(...);	1	?	?	?	?	?	
3	Console.WriteLine("...");	1	?	?	?	?	?	Enter second number:
4	num2 = int.Parse(...);	1	2	?	?	?	?	
5	Console.WriteLine("...");	1	2	?	?	?	?	Enter third number:
6	num3 = int.Parse(...);	1	2	3	?	?	?	
7	Console.WriteLine("...");	1	2	3	?	?	?	Enter fourth

								number:
8	num4 = <b>int</b> .Parse(...);	1	2	3	5	?	?	
9	sum = num1 + ...	1	2	3	5	11		
10	average = ...	1	2	3	5	11	2.75	
11	Console.WriteLine("...");	1	2	3	5	11	2.75	The average is 2.75

הפלט המתקבל בשורה האחרונה הוא: The average is 2.75

## סוף פתרון קציה 6

בפתרון שהוצג לבעיה 6 למדנו כמה עובדות חשובות על משתנים ועל טיפוסים משתנים:

מרכיב חשוב בהגדרת הייעוד של משתנה הוא הגדרת סוג הערכים שיישמרו במשתנה. סוג ערכים זה נקבע באמצעות טיפוס המשתנה. חשוב להתאים את טיפוס המשתנה לתפקידו.

משתנה שומר ערכים מטיפוס (סוג) אחד בלבד.

יש להצהיר בנפרד על משתנים מטיפוסים שונים.

בשפת C# מצהירים על משתנה מטיפוס שלם באמצעות המילה `int` ועל משתנה מטיפוס ממשי באמצעות המילה `double`.

במשפט השמה ניתן לשים במשתנה מטיפוס שלם רק ערך מטיפוס שלם, ואילו במשתנה מטיפוס ממשי ניתן לשים גם ערך מטיפוס שלם וגם ערך מטיפוס ממשי.

ניתן לקלוט ערך בתוך משתנה ממשי בדומה לקליטת ערך בתוך משתנה שלם. במקום להשתמש בהוראה `int.Parse` נשתמש בהוראה `double.Parse` למשל, כך:

```
double x;
Console.Write("Enter a real number: ");
x = double.Parse(Console.ReadLine());
```

### שאלה 3.28

בנו טבלת מעקב אחר ביצוע התוכנית `FourNumbersAverage` עבור הקלט 1 8 6 5.

### שאלה 3.29

פתחו בשלבים אלגוריתם שהקלט שלו הוא שני מספרים ממשיים והפלט שלו הוא שורה שמופיעות בה תוצאות החילוק ב-4 של כל אחד משני המספרים, ושורה שבה מופיע סכום תוצאות החילוק. ישמו את האלגוריתם בשפת C#.

למשל, עבור הקלט 2.84 1.6 הפלט הוא: 0.71 0.4

1.11

שימו ♥ לבחירת טיפוסים המשתנים!

### שאלה 3.30

פתחו בשלבים אלגוריתם שהקלט שלו הוא מחיריהם של שלושה מוצרים בשקלים. הפלט שלו הוא מחיר כולל המתקבל מסכום שלושת המחירים בתוספת מס בשיעור 20%.



### כדאי לדעת – סיבות לאבחנה בין שלם לממשי:

אמנם המספרים השלמים הם תת-קבוצה של המספרים הממשיים, אבל כאשר אנו יודעים כי משתנה מסוים עתיד להכיל רק ערכים שלמים, רצוי להגדירו מטיפוס שלם ולא מטיפוס ממשי. יש לכך שלוש סיבות:

- ◆ בכך שנצהיר על טיפוסו המדויק של משתנה נסייע בהבנת תפקידו ובכך נהפוך את התוכנית לבהירה ולקריאה יותר.
- ◆ ערכים מטיפוס שלם וערכים מטיפוס ממשי מיוצגים בזיכרון המחשב באופן שונה. משום כך, ביצוע פעולות חישוב על ערכים מטיפוס שלם הן פשוטות ומהירות יותר מביצוע אותן פעולות על ערכים מטיפוס ממשי.
- ◆ הגדרת הטיפוסים משמשת את המהדר (הקומפיילר) לבדיקת ההתאמה של משפטי השמה. המהדר בודק אם טיפוס הביטוי המחושב בצד ימין מתאים לטיפוס הביטוי שמבצעים בו את ההשמה. משום כך, הצהרה מדויקת על טיפוסו של משתנה יכולה לסייע לנו באיתור שגיאות.

## 3.6 קבועים

בסעיף זה נכיר הרגל תכנותי שמסייע ליצור תוכניות בהירות, קריאות ועמידות בפני שגיאות.

כיתה י"ב 3 מתכוננת לסיום לימודיה בבית הספר התיכון. בכיתה 36 תלמידים. הגזבר של הכיתה מקבל הצעות מחיר ממארגני אירועים עבור סעופים שונים בתכנון אירועי חגיגות הסיום. למשל, עלות רכישת חולצות עם הדפס שנבחר על ידי הכיתה, עלות הדפסת ספר מחזור, עלות צריבת דיסק עם השיר שהקליטו לכבוד המסיבה ועוד ועוד... מארגני האירועים נותנים הצעת מחיר לתלמיד יחיד, וכדי לחשב את עלותה עבור הכיתה כולה יש להכפיל במספר התלמידים.

אחת התלמידות בכיתה כתבה לגזבר תוכנית שתסייע לו בחישוב עלות הסעופים השונים. הגזבר יוכל לתת לתוכנית כקלט את ההצעות שקיבל עבור הסעופים השונים, והתוכנית תיתן כפלט את העלות של כל אחד מהסעופים עבור הכיתה כולה ואת העלות הכוללת של כל הסעופים.

הנה שלוש ההוראות הראשונות בקטע התוכנית. הקטע מחשב את העלות עבור כל אחד מהסעופים לכיתה כולה:

```
totalShirtPrice = shirtPrice * 36;  
totalBookPrice = bookPrice * 36;  
totalDiskPrice = diskPrice * 36;
```

מאחר שבבית ספר זה נוהגים התלמידים לחגוג את סיום לימודיהם באופן מוגזם משהו, יש בקטע התוכנית הזה עוד שורות רבות...

קטע התוכנית הזה מתאים כמובן רק עבור כיתה י"ב 3, משום שהוא מתייחס באופן ישיר למספר התלמידים בכיתה (36). אם תרצה גם כיתה י"ב 6, ובה 37 תלמידים, להשתמש בתוכנית, תצטרך התלמידה שכתבה את התוכנית לעבור עליה ולשנות בכל מקום את הערך 36 ל-37. אמנם סביר שמספר הסעופים אינו באמת גדול מאוד, ולכן שינוי זה לא יגזול זמן רב, אך עדיין ייתכן כי תטעה ותשכח לשנות את אחד הערכים באחד המקומות. בכך התוכנית תהפוך לשגויה, ותיתן פלט שגוי.

בתוכנית גדולה ומורכבת (לדוגמה: מערכת לניווט לוויינים), הצורך לשנות ערך שמופיע באופן מפורש במקומות רבים בתוכנית, מאות פעמים או אפילו אלפים, הוא בעייתי מאוד, ויש להימנע ממנו במידת האפשר כדי להפוך את התוכנית לעמידה יותר.

הדרך להימנע מכך, היא לתת לערך זה שם, ובכל מקום בתוכנית להשתמש בשמו ולא בערכו. רק במקום אחד בתוכנית נקשר בין השם לערך. אם יהיה צורך בשינוי הערך, השינוי יתבצע רק במקום אחד. בשפת C# נוכל לעשות זאת על ידי שימוש בקבוע.

הצהרה על קבוע בשפת C# דומה להגדרת משתנה. גם עבור קבוע מוקצה מקום בזיכרון ובו נשמר ערכו. אלא שלא כמו עבור משתנה, ערכו של קבוע לא ניתן לשינוי במהלך התוכנית. למשל בתחילת התוכנית שמחשבת את עלות חגיגות הסיום נוכל לכתוב את המשפט הבא:

```
const int NUM_OF_STUDENTS = 36; // מספר התלמידים בכיתה
```

מעתה נקפיד להשתמש בקבועים בתוכניות שנכתוב, כפי שמודגם כבר בתוכנית הראשונה בפרק הבא, פרק 4.

**קבוע (constant)** הוא תא זיכרון, אשר ערכו ההתחלתי לא ניתן לשינוי לאחר שאותחל.

כדי להצהיר על קבוע נכתוב את המילה `const` בתחילת שורת ההצהרה, למשל:

```
const int x = 5;
```

נהוג לכתוב את שם הקבוע באותיות גדולות. אם שם הקבוע כולל יותר ממילה אחת, נהוג להפריד את המילים בקו תחתון.

## סיכום

בפרק זה פיתחנו אלגוריתמים בסיסיים לביצוע במחשב, ויישמו אותם בתוכנית מחשב בשפת C#. הכרנו את אבני הבניין של אלגוריתמים למחשב: משתנים, אשר בהם נשמרים נתונים ותוצאות חישוב, הוראות קלט והוראות פלט, אשר מורות על קליטה של נתונים והצגה של נתונים כפלט, והוראות השמה, אשר מורות על ביצוע חישובים ועל שמירת תוצאותיהם במשתנים.

**משתנה (variable)** הוא תא זיכרון אשר במהלך ביצוע אלגוריתם ניתן לשמור בו ערך ולקרוא את הערך השמור בו. הערך השמור במשתנה נקרא **ערך המשתנה**. פנייה למשתנה מתבצעת באמצעות שם הניתן לו על ידי מפתח האלגוריתם, שם זה הוא **שם המשתנה**.

במשתנה נשמרים ערכים מטיפוסים אשר מתאימים לתפקידו של המשתנה. הטיפוס של הערכים הנשמרים במשתנה הוא **טיפוס המשתנה**.

**טיפוס של ערך (data type)** מגדיר אוסף של ערכים אפשריים ואת הפעולות שניתן לבצע עליהם. בפרק זה הכרנו ערכים מטיפוס שלם (כלומר, מספרים שלמים) וערכים מטיפוס ממשי (כלומר, מספרים ממשיים). כל טיפוס מיוצג בזיכרון המחשב בצורה שונה.

**בחירת טיפוס של משתנה** נעשית כחלק מהגדרת הייעוד של המשתנה. סוג הערכים נקבע על פי נתונים שייקלטו במשתנה או על פי ערכים (של פעולות חישוב) שיושמו במשתנה, למשל כאשר יש לשמור במשתנה תוצאת ממוצע של שני מספרים, יהיה מתאים להגדירו כמשתנה מטיפוס ממשי.

כדי לעדכן את ערכו של משתנה נשתמש ב**פעולת השמה**. בפעולה זו מחושב תחילה ערכו של הביטוי הנמצא בצד ימין של הוראת ההשמה. תוצאת החישוב נשמרת במשתנה הרשום בצד שמאל של הוראת ההשמה. הביטוי לחישוב יכול להיות ביטוי פשוט כגון ערך מפורש או שם של

משתנה, או יכול להיות ביטוי המורכב מפעולות חשבוניות שונות בין ערכים ובין משתנים. משפט השמה משפיע **רק** על ערכו של המשתנה שישמור את תוצאת החישוב, ולא על משתנים אחרים המעורבים בביטוי המחושב. המשתנה שישמור את תוצאת החישוב יכול בעצמו להופיע כחלק מהביטוי המחושב. במקרה זה ערכו החדש תלוי בערכו הישן.

מתן ערך התחלתי למשתנה נקרא **אתחול**.

קליטת נתונים נעשית על ידי **הוראת קלט**, המבצעת השמה של נתון שנקרא מהקלט בתוך משתנה. הצגת נתונים נעשית באמצעות **הוראת פלט**, ובאמצעותה ניתן להציג הודעות, ערכי משתנים וערכי ביטויים כפלט.

ביטוי אשר מורכב מפעולות חשבון בין ערכים ובין משתנים מטיפוס שלם או ממשי נקרא **ביטוי חשבוני**. **טיפוס של ביטוי חשבוני** נקבע על פי טיפוס הערכים והמשתנים שבו, ועל פי הפעולות שבו.

**פתרון בעיה אלגוריתמית** נעשה בשלבים :

1. בחינת **דוגמאות קלט** שונות והבנת הקשר בין הקלט לפלט.
2. חלוקה של משימות האלגוריתם ל**תת-משימות**.
3. בחירת **משתנים** – תפקיד, שם וטיפוס לכל משתנה.
4. כתיבת **האלגוריתם**.
5. כתיבת התוכנית **ליישום** האלגוריתם בשפת התכנות.

לאחר כתיבת התוכנית כדאי לבצע **מעקב** אחר מהלך ביצועה עבור דוגמאות קלט מגוונות, כדי להשתכנע בנכונותה.

מעקב מסודר אחר מהלך ביצוע של אלגוריתם או של תוכנית נעשה באמצעות **טבלת מעקב**. בטבלת מעקב מפורטים השינויים בערכי המשתנים, ומפורט הפלט בעקבות ביצוע כל אחת ואחת מהוראות האלגוריתם או התוכנית.

**ערך התחלתי** של משתנה ביחס לתוכנית או לקטע תוכנית הוא הערך השמור בו מיד לפני תחילת ביצוע אותה תוכנית או אותו קטע תוכנית. **מצב התחלתי** של תוכנית או של קטע תוכנית מתאר את ערכם ההתחלתי של כל המשתנים לפני תחילת הביצוע.

המצב ההתחלתי מתואר בשורה הראשונה של טבלת המעקב. עבור תוכנית שלמה הערכים ההתחלתיים של המשתנים שלא אותחלו עם הצהרתם אינם ידועים (נסמן בסימן שאלה (?)). עבור קטע תוכנית נקבל מראש את הערכים ההתחלתיים של כל המשתנים ונכתוב אותם בשורה הראשונה של טבלת המעקב.

בכל אלגוריתם או תוכנית כדאי לכלול **תיעוד** כדי להסבירם לקורא. יש לצרף לכותרת האלגוריתם או התוכנית הערה המתארת את המטרה, כלומר, את המשימה שהפתרון מיועד למלא. את שמות המשתנים נבחר על פי תפקידיהם, ונוסיף הערות המתארות את תפקידיהם. ההערות מיועדות לקורא בלבד.

במהלך הפרק הצגנו שאלות רבות ומגוונות, שאפשר לחלק לשני סוגים :

♦ שאלות **פיתוח** ויישום של אלגוריתם.

♦ שאלות **ניתוח** אלגוריתם או קטע תוכנית נתון.

שאלות הפיתוח והיישום דורשות פיתוח מלא בשלבים או פיתוח חלקי (כלומר עד שלב החלוקה לתת-משימות או עד שלב בחירת המשתנים או עד שלב כתיבת האלגוריתם, ללא יישום). שאלות הניתוח דורשות מעקב אחר מהלך ביצוע עבור קלט נתון, הבאת דוגמת קלט עבור פלט נתון, תיאור מטרת קטע תוכנית או משימות ניתוח אחרות. ההתנסות בשני הסוגים של השאלות מפתחת את היכולת להבין אלגוריתמים ולפתחם, ובעקבות כך מפתחת את היכולת לפתרון בעיות.

בפרק הבא נפתור בעיות מורכבות יותר מהבעיות שהוצגו בפרק זה ונרחיב בפירוט את השלבים השונים של תהליך פיתוח אלגוריתם ויישומם בתוכנית. בפרקים הבאים אחר כך יוצגו בעיות מורכבות יותר ויותר. כדי להתמודד עם בעיות מורכבות חשובה לא רק היכולת לכתוב תוכנית מחשב, אלא חשובות גם היכולת להתקדם בשלבים והיכולת לנתח ולהבין אלגוריתם נתון (או קטע תוכנית).

## סיכום מרכיבי שפת C# שנלמדו בפרק 3

בחלק זה נפרט את כללי שפת C# שלמדנו בפרק 3. פרט להוראות הקלט, הכללים המוצגים כאן הם הכללים של שפת C# הסטנדרטית, ואינם הכללים של סביבת עבודה מסוימת כגון Visual C#. איננו יכולים להניח שכל המשתמשים בספר זה עובדים באותה הסביבה. לכן לאורך הספר כולו אנו מציגים את כללי שפת C# הסטנדרטית. עם זאת, מאחר שהוראות הקלט מהמקלדת בשפת C# הן מורכבות למדי, בחרנו לחרוג מכלל הסטנדרטיות בנקודה זו.

### מבנה תוכנית בשפת C#

♦ תוכנית בשפת C# היא אוסף של **מחלקות** (מחלקה אחת או יותר). אחת המחלקות היא המחלקה הראשית. **המחלקה הראשית** מכילה את הפעולה הראשית (Main), שממנה ביצוע התוכנית מתחיל ובתחומה משפטי התוכנית נכתבים באופן הבא:

```
public class TheNameOfTheProgram
{
    public static void Main ()
    {
        גוף התוכנית
    } // Main סוף
} // המחלקה הראשית סוף
```

♦ כל הוראה בתוכנית מסתיימת בסימן ; (נקודה-פסיק).

### הערות

בין המשפטים השונים של התוכנית מופיעות **הערות** (comments). ההערות מיועדות למתכנת ולמשתמש בתוכנית, אך לא למחשב. הן עוזרות בקריאת התוכנית ובהבנתה. הערה המתחילה בסימן // נמשכת עד סופה של השורה. הערה התחומה בין הסימנים /\* ו-\*/ יכולה להתפרש על פני כמה שורות. למשל:

```
/* תוכנית לחישוב ממוצע */
// משתנה השומר את הממוצע
```

### שמות

♦ המתכנת בוחר את **שמות** מרכיבי התוכנית שהגדיר (שמות המשתנים, שמות המחלקות ובפרט שם המחלקה הראשית, כלומר שם התוכנית). אנו נוהגים לכנות משתנים בשמות משמעותיים, ולעתים נצמיד שתי מילים או יותר כדי ליצור שם משמעותי וברור יותר.

- ◆ על פי **המוסכמות** המקובלות במתן שמות, לא נהוג לקצר שמות. למשל אם המשתנה מתעתד להכיל בתוכו ממוצע נעדיף לקרוא לו `average` ולא `avg`. כך גם בשמות של מחלקות.
- ◆ שם מחלקה יתחיל באות גדולה, שם משתנה יתחיל באות קטנה. שאר האותיות יהיו קטנות, אלא אם השם מורכב מכמה מילים שהוצמדו זו לזו. במקרה זה נשתמש באות גדולה בראש כל מילה פרט למילה הראשונה. למשל `ReadWrite, sum, numOfChildren` וכדומה. הקפדה על כללים אלה יוצרת תוכנית ברורה וקריאה יותר.
- ◆ קיימת ב-`C#` קבוצת שמות מיוחדת הנקראת **מילים שמורות** (`reserved words`). לשמות אלה יש משמעות מוגדרת ב-`C#`, ואסור להשתמש בהם לשמות אחרים. למשל אסור להשתמש במילה `class` כשם של משתנה כי היא מילה שמורה עבור מחלקה. בתוכניות המופיעות בספר זה, מילה שמורה כתובה באותיות מודגשות.
- ◆ ההצהרות, השמות ומרכיבי ההוראות בתוכנית חייבים להיות מופרדים בתו רווח אחד לפחות. למשל, לא נוכל לכתוב: `publicclassMyProgram`.

## ערכים ונתונים בתוכנית C#

### טיפוסים

- ◆ הערכים המופיעים בתוכנית וערכי המשתנים והקבועים מסווגים ל**טיפוסים** (`types`). הטיפוסים שהכרנו עד עכשיו הם שלם (`int`) וממשי (`double`).
- ◆ **ערך מטיפוס שלם** המופיע בתוכנית `C#` הוא מספר שלם כשלשמאלו יכול להופיע הסימן פלוס (+) או הסימן מינוס (-). אם המספר שלילי יש לכתוב את הסימן מינוס. כתיבת הסימן פלוס אינה הכרחית (זוהי ברירת המחדל, כלומר אם לא כתוב אף סימן, המספר מפורש כמספר חיובי). אלה לדוגמה ערכים חוקיים מטיפוס שלם בשפת `C#`: `156, +156, -3, 0`.
- ◆ **ערך מטיפוס ממשי** המופיע בתוכנית `C#` מורכב מארבעה חלקים:
  1. סימן + או הסימן - (כתיבת הסימן + אינה הכרחית כאשר המספר חיובי).
  2. סדרה לא ריקה של ספרות המייצגת את החלק השלם של המספר.
  3. נקודה עשרונית.
  4. סדרה לא ריקה של ספרות המייצגת את השבר של המספר.
 אלה לדוגמה ערכים חוקיים מטיפוס ממשי בשפת `C#`: `1.53, -0.2, 7.0, +5.3`. הערכים 5 או 3 אינם ערכים ממשיים חוקיים.
- ◆ מספרים שלמים יכולים להיות מיוצגים בשפת `C#` בערכים שטיפוסם שלם או בערכים שטיפוסם ממשי. המספר השלם שלוש למשל יכול להיות מיוצג בערך 3 שטיפוסו שלם, וגם בערך 3.0 שטיפוסו ממשי.

### משתנים

- ◆ **הצהרה על משתנים** תתבצע באמצעות הצהרה על הטיפוס ועל שמו של המשתנה. המילה `int` משמשת להצהרה על משתנה מטיפוס שלם, והמילה `double` משמשת להצהרה על משתנה מטיפוס ממשי.
 

לדוגמה:

```
int num;
```

- ◆ ניתן להצהיר על כמה משתנים באותה השורה אם תפקידם דומה, בהפרדה בפסיקים. לדוגמה:
 

```
double num1, num2;
```

## קבועים

**קבוע** הוא תא זיכרון שערכו לא יכול להשתנות לאחר שנקבע. ההצהרה על קבוע נעשית בדומה להצהרת משתנה, אך מקדימה אותה המילה `const` למשל:

```
const int MY_CONSTANT_INTEGER = 3;
```

## הוראות ביצוע של תוכנית בשפת C#

### קלט

♦ ליישום **הוראת קלט של ערך שלם** נשתמש בפעולה `.int.Parse(Console.ReadLine())` **ההוראת קלט של ערך ממשי** נשתמש בפעולה `.double.Parse(Console.ReadLine())` **ההוראת הקלט כוללת שם של משתנה שיישמר בו הערך הנקלט**. למשל המבנה הכללי של **ההוראת קלט של ערך שלם הוא**:

```
int.Parse(Console.ReadLine()) = שם משתנה
```

♦ **ביצוע פעולת קלט גורם לעצירת התוכנית עד לקליטת ערך מתאים**. לאחר קליטתו הוא נשמר בתוך המשתנה.

♦ **נזכור לכתוב לפני פעולת הקלט פעולת פלט המנחה את המשתמש לגבי הקלט שהתוכנית מצפה לקבל, למשל**:

```
Console.Write("Enter a positive integer number: ");  
x = int.Parse(Console.ReadLine());
```

### פלט

♦ **ההוראת פלט מיושמת ב-C# באמצעות הפעולה** `Console.Write` **(שאינה גורמת למעבר לשורה הבאה בפלט) או הפעולה** `Console.WriteLine` **(שגורמת למעבר לשורה הבאה לאחר הצגת הפלט המבוקש)**. למשל:

```
Console.Write("a message");  
Console.WriteLine(x);
```

**וכאשר x משתנה שם הפעולה המלא הוא** `System.Console.WriteLine`. **כדי שנוכל לכתוב בצורה המקוצרת נסיף בראש התוכנית את ההוראה הבאה**:

```
using System;
```

♦ **ניתן לצרף פריטים נוספים להודעה שברצוננו להציג**. בתוך ההודעה נסמן את המקום שאמורים להשתלב בו הפריטים הנוספים ולאחר ההודעה נצרף את רשימת הפריטים, למשל:

```
Console.WriteLine("The sum of {0} and {1} is: {2}", num1, num2, sum);
```

### השמה

♦ **המבנה הכללי של משפט השמה ב-C# הוא**:

```
ביטוי = משתנה
```

♦ **הביטוי** המופיע מימין לסימן =, הוא ערך מפורש, משתנה או ביטוי מורכב. אם הביטוי הוא ביטוי חשבוני סדר הקדימויות של פעולות החשבון זהה לסדר הקדימויות המקובל במתמטיקה.

♦ במשתנה מטיפוס ממשי אפשר לשים ערך מטיפוס שלם או ממשי. במשתנה מטיפוס שלם אפשר לשים רק ערכים שלמים.

## שאלות נוספות

### שאלות נוספות לסעיף 3.1

1. כתבו תוכנית להדפסת האות L מכוכביות באופן הבא:

```
*
*
* * *
```

2. נתון קטע התוכנית הבא:

```
left = int.Parse(Console.ReadLine());
right = int.Parse(Console.ReadLine());
Console.WriteLine("{0} {1}", right, left);
```

נניח שנתוני הקלט שהוקלדו הם 10 8:

א. מה יהיו ערכי המשתנים לאחר ביצוע משפטי הקלט:

ב. מה יהיה הפלט?

3. נתון קטע התוכנית הבא:

```
num1 = int.Parse(Console.ReadLine());
num2 = int.Parse(Console.ReadLine());
num3 = int.Parse(Console.ReadLine());
Console.WriteLine("{0} {1}", num2, num3);
Console.WriteLine("{0} {1} {2}", num3, num1, num3);
```

תנו דוגמת קלט שהפלט עבורה הוא:

```
5 9
9 5 9
```

4. פתחו בשלבים אלגוריתם אשר הקלט שלו הוא שלושה מספרים שלמים, והפלט שלו הוא שלוש שורות של מספרים: בשורה הראשונה יופיע נתון הקלט השלישי, בשורה השנייה יופיע נתון הקלט השלישי והשני, ובשורה השלישית יופיע נתון הקלט השלישי השני והראשון. ישמו את האלגוריתם בשפת C#.

### שאלות נוספות לסעיף 3.2

1. כתבו משפטי השמה לביצוע הפעולות הבאות:

א. השמה במשתנה a של סכום ערכי המשתנים c ו-b.

ב. השמה במשתנה c של ההפרש בין פעמיים ערכו של המשתנה d ובין ערך המשתנה b.

ג. השמה במשתנה e של סכום ערכו של המשתנה a וחמש פעמים ערכו של המשתנה f.

2. מחיר כרטיס כניסה לבריכת השחייה העירונית הוא 20 ₪ למבוגר ו-12 ₪ לילד. פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר מבוגרים ומספר ילדים, והפלט שלו הוא הסכום לגבייה עבור הכרטיסים. ישמו את האלגוריתם בשפת C#.

### שאלות נוספות לסעיף 3.3

1. נתון משפט ההשמה הבא:  $a = a + b + c + d$ ;  
א. כתבו במקום משפט השמה זה סדרה של משפטי השמה אשר הביטוי בצד ימין של כל אחד מהם כולל סימן חיבור אחד בלבד. בסיום ביצוע סדרת המשפטים ערכו של  $a$  יהיה זהה לערכו המתקבל לאחר ביצוע המשפט הנתון. בצעו את המשימה ללא הוספת משתנים.  
ב. בנו טבלת מעקב אחר מהלך הביצוע של סדרת משפטי ההשמה שכתבתם בסעיף א עבור הערכים ההתחלתיים 1, 5, 2 ו-4 במשתנים  $a, b, c$  ו- $d$  בהתאמה.

2. כתבו משפטי השמה לביצוע הפעולות הבאות:

א. הקטנת ערכו של  $a$  ב-5.

ב. הגדלת ערכו של  $b$  פי 3.

ג. הקטנת ערכו של המשתנה  $a$  בערכו של המשתנה  $b$ .

3. עבור כל זוג משפטי השמה נתון כתבו משפט השמה אחד שמשיג אותה מטרם:

<p>א. <math>n = m + 5</math>; <math>n = n - 2</math>;</p>	<p>ב. <math>a = (a + 2) * 3</math>; <math>a = a * 4 - 9</math>;</p>
<p>ג. <math>v = v + w</math>; <math>v = v * 5</math>;</p>	<p>ד. <math>x = x - 2</math>; <math>x = x - 3</math>;</p>

### שאלות נוספות לסעיף 3.4

1. נתון קטע התוכנית הבא:

```
x = int.Parse(Console.ReadLine());
y = int.Parse(Console.ReadLine());
x = x + y;
y = x - y;
x = x - y;
Console.WriteLine("{0} {1}", x, y);
```

א. מהו פלט קטע התוכנית עבור הקלט 5 13? היעזרו בטבלת מעקב למציאת הפלט.

ב. נסחו את הבעיה האלגוריתמית שקטע תוכנית זה פותר.

ג. כתבו קטע תוכנית אחר לפתרון הבעיה.

2. מטרת סדרת המשפטים הבאה היא כי אחרי ביצועה יהיו במשתנים  $a, b, c, d$  ו- $e$  ערכי המשתנים  $a, b, c, d$  בהתאמה.

```
b = a;
c = b;
d = c;
e = d;
```

א. תנו דוגמה של ערכים התחלתיים עבור המשתנים אשר המטרה לא מושגת עבורם.

ב. מה מתבצע בסדרת המשפטים הנתונה? האם היא גורמת ל"אובדן" ערכי משתנים?



ג. כתבו סדרת משפטים שעבורה תושג המטרה.

### שאלות נוספות לסעיף 3.5

1. במשרדי הממשלה עבור כל טופס שפקיד ממלא הוא מקבל שכר של 6.3 ₪. פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר הטפסים שעל הפקיד למלא, והפלט שלו הוא השכר שהפקיד יקבל. ישמו את האלגוריתם בשפת C#. למשל עבור הקלט 55 הפלט הדרוש הוא 346.5.

2. פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר חיובי שלם המציין אורך צלע של ריבוע, והפלט שלו הוא שטח העיגול החסום בריבוע. ישמו את האלגוריתם בשפת C#.

3. עקב איחור בעונת הגשמים החליטו יצרני המטריות על מבצע מכירות בהנחה. יש לפתח וליישם אלגוריתם אשר הקלט שלו הוא מחיר מטרייה, אחוז ההנחה למטרייה, ומספר מטריות מבוקש, והפלט שלו הוא הסכום הכולל לתשלום.

נבחר את המשתנה הבא מטיפוס שלם:

**num** – ישמור את מספר המטריות המבוקש

ואת המשתנים הבאים מטיפוס ממשי:

**price** – ישמור את המחיר של המטרייה

**discount** – ישמור את אחוז ההנחה

**newPrice** – ישמור את המחיר של מטרייה אחת לאחר הנחה

**total** – ישמור את הסכום הכולל לתשלום

פותר אלגוריתם שהוראותיו מיושמות במשפטי התוכנית הבאים:

```
Console.Write("How many umbrellas?: ");
num = int.Parse(Console.ReadLine());
Console.Write("Enter the price of one umbrella: ");
price = double.Parse(Console.ReadLine());
Console.Write("Enter discount percentage per umbrella: ");
discount = double.Parse(Console.ReadLine());
newPrice = _____;
total = _____;
Console.WriteLine("Total sum is {0}", total);
```

השלימו את משפטי התוכנית.

### שאלות מסכמות לפרק 3

1. ניתן להמיר ערך של טמפרטורה המיוצג במעלות פרנהייט (F) לייצוג במעלות צלזיוס (C) על ידי הנוסחה:  $C = 5/9 (F - 32)$ .

פתחו בשלבים אלגוריתם אשר הקלט שלו הוא טמפרטורה הנתונה במעלות פרנהייט, והפלט שלו הוא ערך הטמפרטורה במעלות צלזיוס. ישמו את האלגוריתם בשפת C#.

2. פתחו בשלבים (אין צורך ליישם בתוכנית) אלגוריתם אשר הקלט שלו הוא אורכי שני הניצבים והיתר במשולש ישר זווית, והפלט שלו הוא היקף המשולש ושטח המשולש.

3. פתחו בשלבים אלגוריתם שהקלט שלו הוא שלושה מספרים שלמים, והפלט שלו הוא כל הסיידורים האפשריים של שלושת המספרים. הניחו כי שלושת המספרים שונים זה מזה. ישמו את האלגוריתם בשפת C#.

**הדרכה:** חלקו את הפלט לשלושה חלקים: הסיידורים שמתחילים בנתון הקלט הראשון, הסיידורים שמתחילים בנתון הקלט השני, הסיידורים שמתחילים בנתון הקלט השלישי.

למשל, עבור הקלט: 1 8 30 הפלט המתאים הוא:

```
1 8 30
1 30 8
8 1 30
8 30 1
30 1 8
30 8 1
```

4. נתון קטע התוכנית הבא:

```
a = int.Parse(Console.ReadLine());
b = int.Parse(Console.ReadLine());
a = a + b;
Console.WriteLine(a);
a = a - 2 * b;
Console.WriteLine(a);
a = a + b;
Console.WriteLine(a);
```

א. מהו פלט קטע התוכנית עבור הקלט 2 3? היעזרו בטבלת מעקב כדי לענות על השאלה.

ב. תנו דוגמת קלט אשר הפלט עבורה הוא 1 2 3.

ג. נסחו במילים את היחס שמוגדר בקטע התוכנית בין הפלט לקלט.

5. הזזה מעגלית של סדרת ערכים משמעותה העברת הערך האחרון בסדרה לתחילתה. למשל לאחר ביצוע הזזה מעגלית על הסדרה 1 2 3 4 מתקבלת הסדרה: 4 1 2 3. הזזה מעגלית היא תבנית שיכולה לשמש בפתרון בעיות אלגוריתמיות שונות.

נתון קטע התוכנית הבא שהקלט שלו הוא שלושה מספרים ומטרתו היא לתת כפלט את תוצאת ההזזה המעגלית על סדרת נתוני הקלט:

```
Console.Write("Enter first element: ");
x = int.Parse(Console.ReadLine());
Console.Write("Enter second element: ");
y = int.Parse(Console.ReadLine());
Console.Write("Enter third element: ");
z = int.Parse(Console.ReadLine());
x = y;
y = z;
z = x;
Console.WriteLine("{0} {1} {2}", x, y, z);
```

קטע התוכנית שגוי.

א. הסבירו מדוע הקטע שגוי.

- ב. תקנו את התוכנית על ידי שינוי החלק של משפטי ההשמה (מבלי לשנות את משפט הפלט).  
ג. תקנו את קטע התוכנית על ידי ביטול משפטי ההשמה ועל ידי שינוי משפט הפלט.

שאלה 5 עסקה בהזזה מעגלית של סדרת איברים. העמקה בתבנית **הזזה מעגלית בסדרה** נמצאת בסעיף הבא.

## תבניות – פרק 3

פירוט מלא של התבניות ושל שאלות שבפתרון יש שימוש בתבניות ניתן למצוא באתר הספר ברשת האינטרנט.

### החלפת ערכים בין שני משתנים

שם התבנית: **החלפת ערכים בין שני משתנים**  
נקודת מוצא: שני ערכים במשתנים element1 ו-element2  
מטרה: החלפת הערכים ההתחלתיים בין שני המשתנים אלגוריתם:  
1. השם temp-2 אג ערכו של element1  
2. השם element1-2 אג ערכו של element2  
3. השם element2-2 אג ערכו של temp

### היפוך סדר האיברים בסדרה

שם התבנית: **היפוך סדר האיברים בסדרה**  
נקודת מוצא: שני ערכים במשתנים element1 ו-element2  
מטרה: היפוך הערכים בין שני המשתנים אלגוריתם:  
החלף את ערכי element1 ו-element2

### ממוצע של סדרת מספרים

שם התבנית: **ממוצע של סדרת מספרים**  
נקודת מוצא: שני מספרים ב-num1 ו-num2  
מטרה: חישוב הממוצע של שני המספרים אלגוריתם:  
השם sum-2 אג ערכו של הביטוי  $num1 + num2$   
השם average-2 אג ערכו של הביטוי  $sum / 2$

## הזזה מעגלית בסידרה

שם התבנית : הזזה מעגלית שמאלה בסדרה

נקודת מוצא : שלושה ערכים במשתנים element1, element2 ו-element3

מטרה : הזזה מעגלית שמאלה של שלושת המשתנים

אלגוריתם :

החלף את ערכי המשתנים element1 ו-element2

החלף את ערכי המשתנים element2 ו-element3

שם התבנית : הזזה מעגלית ימינה בסדרה

נקודת מוצא : שלושה ערכים במשתנים element1, element2 ו-element3

מטרה : הזזה מעגלית ימינה של שלושת המשתנים

אלגוריתם :

החלף את ערכי המשתנים element2 ו-element3

החלף את ערכי המשתנים element1 ו-element2

## תבניות – פרק 3

### החלפת ערכים בין שני משתנים

נתבונן בבעיה הבאה :

בחנות למוצרי חשמל הוחלפו בטעות מחיריהם של הדיסקמן והווקמן המוצעים למכירה במחיר מבצע. מחירו האמיתי של הווקמן נשמר במחשב החנות במשתנה diskman ומחיר הדיסקמן נשמר במשתנה walkman. עזרו לבעל החנות לתקן את הטעות על ידי השלמת האלגוריתם :

1. השם 2- \_\_\_\_\_ אג ערכו של \_\_\_\_\_
2. השם 2- walkman אג ערכו של \_\_\_\_\_
3. השם 2- \_\_\_\_\_ אג ערכו של \_\_\_\_\_

בעיה זו יש להחליף את ערכיהם של diskman ושל walkman. כדי להחליף בין ערכי המשתנים נצטרך להשתמש במשתנה עזר temp, שישמור את ערכו ההתחלתי של walkman, ולכן האלגוריתם ייראה כך :

1. השם 2- temp אג ערכו של walkman
2. השם 2- walkman אג ערכו של diskman
3. השם 2- diskman אג ערכו של temp

בפתרון בעיה זו השתמשנו בתבנית של החלפה בין שני ערכים, בדומה לאלגוריתם שבפתרון בעיה 5 בפרק 3. נתבונן בשני האלגוריתמים הללו :

1. השם 2- temp אג ערכו של walkman	1. השם 2- temp אג ערכו של a
2. השם 2- walkman אג ערכו של diskman	2. השם 2- a אג ערכו של b
3. השם 2- diskman אג ערכו של temp	3. השם 2- b אג ערכו של temp

נשים לב, כי אם נקביל את המשתנים a ו-b למשתנים walkman ו-diskman, בהתאמה, נקבל שני אלגוריתמים זהים. תבנית זו, **החלפת ערכים בין שני משתנים** מופיעה באלגוריתמים רבים ולרוב משמשת כתבנית בסיס של פעולות סידור ערכים, למשל עבור מיון ערכים בסדרה.

לתבנית זו של **החלפת ערכים בין שני משתנים** ולכל התבניות שתוגדרנה בהמשך יש כמה מרכיבים המאפיינים אותן.

נגדיר באופן כללי את **מאפייניה של תבנית** עם הסבר קצר לכל מאפיין:

**שם התבנית:** השם מבטא בצורה תמציתית את המשימה לביצוע או את דרך ביצועה.

**נקודת מוצא:** נקודת המוצא מציינת את המצב התחילי הנתון של המשימה לביצוע, כלומר: שמות המשתנים ובהקשרים מסוימים גם טיפוסיהם. התבניות מתפתחות עם ההתקדמות בחומר הלימוד ולכן ייתכן כי לתבנית אחת תהיינה נקודות מוצא שונות, למשל, פעם נקודת המוצא תהיה שני מספרים, ופעם אחרת שלושה מספרים.

**מטרה:** המטרה מתארת את המצב הסופי, הפלט הדרוש או ערך שיש להחזיר עם תום הביצוע.

**אלגוריתם:** האלגוריתם מתאר מתכונת לביצוע המשימה. האלגוריתם הוא לב התבנית. לעיתים בתוך האלגוריתם יהיה שימוש בתבנית אחרת.

**יישום ב-C#:** יישום האלגוריתם בשפת C#.

נציג את התבנית **החלפת ערכים בין שני משתנים**, על פי המאפיינים שהכרנו:

**שם התבנית:** החלפת ערכים בין שני משתנים

**נקודת מוצא:** שני ערכים במשתנים element1 ו-element2

**מטרה:** החלפת הערכים ההתחלתיים בין שני המשתנים

**אלגוריתם:**

1. השם temp-2 אג ערכו element1

2. השם element1-2 אג ערכו element2

3. השם element2-2 אג ערכו temp

**יישום ב-C#:**

```
temp = element1;
element1 = element2;
element2 = temp;
```

**שימו ♥:** ביישום התבנית נעשה שימוש במשתנה עזר temp כדי להבטיח שלא יאבד אף אחד מהערכים ההתחלתיים של שני המשתנים.

בעל החנות של מוצרי החשמל יכול להשתמש בתבנית שתיארנו. נוכל לכתוב

**החלף את ערכי המשתנים diskman ו-walkman**

מאחר שזו תבנית מוכרת לנו, הרי השימוש בתבנית כהוראה באלגוריתם (במקרה זה, אלגוריתם בן שורה אחת), מסביר מהן הפעולות שיש לבצע. האלגוריתם שנכתב הוא קצר יותר, אבל גם ברור יותר. אמנם הוראה אחת בו מייצגת כעת כמה הוראות, אבל היא מסבירה היטב את תפקידן של אותן הוראות.

## שאלה 1

ישמו בשפת C# את השימוש של בעל החנות למוצרי חשמל בתבנית.

## שאלה 2

איתמר ויאיר משחקים במשחק הקלפים "טאקי". בתחילת המשחק קיבל כל אחד מהם מספר קלפים זהה. המנצח במשחק הוא השחקן שאין בידו קלפים. במהלך המשחק, כאשר איתמר הבחין כי מספר הקלפים שבידו גדול בהרבה ממספר הקלפים שבידי יאיר החליט להשתמש בקלף "החלף קלפים", שמשמעותו החלפת הקלפים בין שני השחקנים.

נתון אלגוריתם, שהקלט שלו הוא מספר הקלפים שיש לאיתמר וליאיר לפני ביצוע ההחלפה והפלט שלו הוא מספר הקלפים של כל אחד מהם לאחר ביצוע ההחלפה וכן מספר הקלפים שאיתמר הצליח להיפטר במסגרת ההחלפה:

1. קאוט מספר קלפים של איתמר ב-itamar
2. קאוט מספר קלפים של יאיר ב-yair
3. השם ב-temp אג ערכו של itamar
4. השם ב-itamar אג ערכו של yair
5. השם ב-yair אג ערכו של temp
6. הצג כפאט אג ההודעה "מספר הקלפים של איתמר לאגרי ההגלפה" ואג ערכו של itamar
7. הצג כפאט אג ההודעה "מספר הקלפים של יאיר לאגרי ההגלפה" ואג ערכו של yair
8. השם ב-diff אג ערכו של הכיטוי הגלבוני yair - itamar
9. הצג כפאט אג ההודעה "מספר הקלפים שלהם איתמר הצויא להיפטר במסגרת ההגלפה" ואג ערכו של diff

א. ציינו מהן השורות באלגוריתם המממשות את התבנית **החלפת ערכים בין שני משתנים**.

ב. מהי נקודת המוצא של התבנית בשימוש זה?

ג. כתבו אלגוריתם שקול לאלגוריתם הנתון תוך שימוש בתבנית **החלפת ערכים בין שני משתנים**.

ד. כתבו אלגוריתם שקול לאלגוריתם הנתון המבצע אותה מטרה **ללא** שימוש בתבנית **החלפת ערכים בין שני משתנים**.

### שאלה 3

דרך נוספת להחליף ערכים בין שני משתנים היא להשים בתחילה את ערכו של element2 ב-temp. השלימו את קטע התוכנית המתאים להצעה זאת:

```
temp = element2;
```

---

---

---



## היפוך סדר האיברים בסדרה

נתבונן בשתי הבעיות האלגוריתמיות הבאות:

**בעיה 1:** כתבו אלגוריתם, שהקלט שלו הוא שני מספרים שלמים הנקלטים למשתנים big ו-small, כך שהמספר הגדול נקלט ל-small והמספר הקטן נקלט ל-big. על האלגוריתם להפוך את ערכי המשתנים, כך שבמשתנה big יישמר הערך הגדול ובמשתנה small יישמר הערך הקטן, ולהציג כפלט את הערכים לאחר ההיפוך.

**בעיה 2:** כל הספרים בספריה העירונית מקוטלגים על פי מספרים קטלוגיים. על מדף כלשהו בספריה מסודרים כל הספרים בסדר עולה על פי מספריהם הקטלוגיים פרט לשני ספרים שהחליפו בטעות את מקומם זה בזה. כתבו אלגוריתם, שמטרתו לדאוג לכך שכל הספרים במדף יהיו מסודרים בסדר עולה על פי מספריהם הקטלוגיים של הספרים. הקלט של האלגוריתם הוא המקומות של הספרים שאינם מונחים במקומותיהם ומספריהם הקטלוגיים, בהתאמה. על האלגוריתם להפוך את סדר הספרים ולהציג כפלט את המקומות והמספרים הקטלוגיים של הספרים לאחר ההיפוך.

אנו רואים כי בשתי הבעיות האלגוריתמיות עלינו להפוך סדר של שני ערכים. זהו ליבה של התבנית **היפוך סדר האיברים בסדרה**, במקרה זה של סדרה בת שני איברים. היפוך סדר האיברים בסדרה שימושי בהקשרים בהם הסדרה נתונה בסדר כלשהו ויש צורך להפכו. למשל, כאשר מעוניינים להפוך סדרה המסודרת בסדר עולה לאותה סדרה המסודרת בסדר יורד.

**שימו ♥:** עבור סדרה בת שני איברים, האלגוריתמים של התבנית **החלפת ערכים בין שני משתנים** ושל התבנית **היפוך סדר האיברים בסדרה** הם זהים, כי כדי להפוך סדר של שני ערכים בלבד יש למעשה להחליף בין שני ערכי המשתנים. חשוב לציין שזהו מקרה פרטי, ולפעולת ההיפוך יש משמעות עבור סדרה שבה יותר משני איברים. בהמשך נראה דוגמאות להיפוך סדרה שבה לפחות 3 איברים.

נגדיר את מאפייני התבנית:

**שם התבנית:** היפוך סדר האיברים בסדרה

**נקודת מוצא:** שני ערכים במשתנים element1 ו-element2

**מטרה:** היפוך הערכים בין שני המשתנים

**אלגוריתם:**

**החלף את ערכי** element1 ו-element2

הנה הפתרונות של שתי הבעיות האלגוריתמיות :

פתרון בעיה 2	פתרון בעיה 1
1. קאוט מקום ספר ראשון ב-place1	1. קאוט מספר גזאל-small
2. קאוט מספר קטלוגי של ספר ראשון ב-num1	2. קאוט מספר קטן-big
3. קאוט מקום ספר שני ב-place2	3. הפוך את סדר האיברים בסדרה small, big
4. קאוט מספר קטלוגי של ספר שני ב-num2	4. הצג כפאט "הערך הגזאל", big
5. הפוך את סדר האיברים בסדרה num1, num2	5. הצג כפאט "הערך הקטן", small
6. הצג כפאט "מיקום הספר", num1, "ב-"	
place1	
7. הצג כפאט "מיקום הספר", num2, "ב-"	
place2	

#### שאלה 4

ישמו כל אחד מן האלגוריתמים כקטע תוכנית בשפת C#.

שימו לב כי כדי ליישם את התבנית **היפוך סדר האיברים בסדרה** עליכם להשתמש ביישום של התבנית **החלפת ערכים בין שני משתנים**.

#### שאלה 5

נתונים 3 ערכים במשתנים element1, element2, ו-element3. המורה ביקשה מהתלמידים להציע אלגוריתם עבור היפוך סדר הערכים בסדרת המשתנים.

אוהד הציע את האלגוריתם הבא :

1. הפוך את סדר האיברים בסדרה element1, element3

2. הפוך את סדר האיברים בסדרה element3, element1

האלגוריתם של אוהד שגוי.

א. תנו דוגמה לערכים ב-element1, element2, ו-element3 שעבורה ניתן לראות כי האלגוריתם שגוי.

ב. תנו שתי דוגמאות שונות לערכים ב-element1, element2, ו-element3, שעבורן לא ניתן לראות כי האלגוריתם שגוי. רשמו מהו המאפיין של כל אחת מהדוגמאות.

ג. הסבירו במלים מדוע האלגוריתם שאוהד הציע שגוי.

ד. תקנו את האלגוריתם.

ה. ישמו את האלגוריתם כקטע תוכנית בשפת C#.

## ממוצע של סדרת מספרים

נתבונן בשלוש הבעיות האלגוריתמיות הבאות:

**בעיה 1:** ציון שנתי של מקצוע בתעודה נקבע על פי הממוצע של הציונים במחצית א' ובמחצית ב'. כתבו אלגוריתם, שהקלט שלו הוא ציוניו של אלי במחצית א' ובמחצית ב' במקצוע מדעי המחשב והפלט שלו הוא הציון השנתי של אלי במדעי המחשב.

**בעיה 2:** הציון הבית-ספרי של תלמיד נקבע על פי הממוצע של הציון השנתי וציון המתכונת. כתבו אלגוריתם, שהקלט שלו הוא ציונו של אלי בבחינת המתכונת במדעי המחשב והפלט שלו הוא ציונו הבית-ספרי של אלי במדעי המחשב.

**בעיה 3:** ציון סופי של תלמיד במקצוע נקבע על פי הממוצע של הציון הבית-ספרי והציון בבחינת הבגרות. כתבו אלגוריתם, שהקלט שלו הוא ציונו של אלי בבחינת הבגרות במדעי המחשב והפלט שלו הוא ציונו הסופי של אלי במדעי המחשב.

בשלוש הבעיות האלגוריתמיות יש שימוש בתבנית ממוצע. ממוצע הוא מדד סטטיסטי וחשובו הוא אחד החישובים הבסיסיים עבור סדרת ערכים מספריים. כדי לחשב ממוצע של סדרה יש לחשב תחילה את הסכום הכולל של הסדרה ולאחר מכן לחלק במספר הערכים בסדרה.

נגדיר את מאפייני התבנית **ממוצע של סדרת מספרים**, עבור סדרה בת שני מספרים:

**שם התבנית:** ממוצע של סדרת מספרים

**נקודת מוצא:** שני מספרים ב-`num1` ו-`num2`

**מטרה:** חישוב הממוצע של שני המספרים

**אלגוריתם:**

השם `sum` - סכום של שני המספרים `num1 + num2`

השם `average` - ממוצע של שני המספרים `sum / 2`

**יישום ב-C#:**

```
sum = num1 + num2;  
average = (double) sum / 2;
```

נציג עתה את הפתרון של שלוש הבעיות האלגוריתמיות ברצף:

1. קאוס ציון ממוצע א' ב-`semester1`

2. קאוס ציון ממוצע ב' ב-`semester2`

3. **גש** ממוצע של סדרת המספרים semester1, semester2 **והשם** **א** **הערך**  
המיושם ב-yearGrade
4. **הצג** כפולט "ציון השנה של **א** **ב**מדי **המגש**", yearGrade
5. **קולט** ציון בינה מכלול ב-finalExam
6. **גש** ממוצע של סדרת המספרים yearGrade, finalExam **והשם** **א** **הערך**  
המיושם ב-schoolGrade
7. **הצג** כפולט "ציון הבית-ספרי של **א** **ב**מדי **המגש**", schoolGrade
8. **קולט** ציון בינה בגרות ב-matriculation
9. **גש** ממוצע של סדרת המספרים schoolGrade, matriculation **והשם** **א** **הערך**  
המיושם ב-finalGrade
10. **הצג** כפולט "ציון הסופי של **א** **ב**מדי **המגש**", finalGrade

## שאלה 6

ישמו את האלגוריתם כקטע תוכנית בשפת C#.

## שאלה 7

עידן טוען כי ניתן לכתוב את היישום ב-C# עבור התבנית **ממוצע של סדרת מספרים** בת שני מספרים ממשיים בהוראה אחת ואין צורך לפצל לשתי הוראות, ולכן הציע את היישום הבא:

```
average = num1 + num2 / 2;
```

חגית טוענת כי היישום שעידן הציע שגוי.

- א. תנו דוגמה לערכי  $\text{num1}$  ו- $\text{num2}$ , עבורה האלגוריתם שעידן הציע נותן פלט נכון.  
 ב. תנו דוגמה לערכי  $\text{num1}$  ו- $\text{num2}$ , עבורה ניתן לראות כי טענתה של חגית נכונה.  
 ג. תקנו את היישום שעידן הציע.

## שאלה 8

לפניכם שימוש בתבנית:

אשר ממוצע של סדרת המספרים num1, num2, num3 והצג כפולט את הערך

28/1/87

- ב. כתבו את היישום ב-C# עבור השימוש המתואר.

## שאלה 9

בפינת החי "צבי הנינג'ה" ישנם 3 צבים. האחראי על פינת החי מעוניין לבצע חישובים סטטיסטיים על התפתחות הצבים. לצורך זה הוא שוקל את הצבים אחת לחודש ורושם את ממוצע משקלי שלושת הצבים.

נתון קטע התוכנית הבא שהקלט שלו הוא משקלי שלושת הצבים בחודש ינואר והפלט שלו אמור להיות ממוצע משקליהם :

```
weight1 = double.Parse(Console.ReadLine());  
weight2 = double.Parse(Console.ReadLine());  
weight3 = double.Parse(Console.ReadLine());  
sum = weight1 + weight2;  
average = sum / 2;  
sum = average + weight3;  
average = sum / 2;  
Console.WriteLine(average);
```

קטע התוכנית שגוי.

א. תנו דוגמת קלט שעבורה ניתן לראות כי קטע התוכנית אינו משיג את המטרה.

ב. תנו דוגמת קלט שעבורה קטע התוכנית משיג את המטרה.

ג. בקטע התוכנית ישנו שימוש כפול בתבנית: עבור כל אחד מהשימושים, ציינו את ההוראות המתאימות לו ותארו את השימוש בתבנית.

ד. תקנו את קטע התוכנית.

---

## הזזה מעגלית בסדרה

**הזזה מעגלית בסדרה** היא תבנית הנחוצה לעיבוד אשר בו יש להזיז באופן אחיד את כל הערכים בסדרה, תוך הקפדה על כך שערכו של אף ערך לא יאבד. ניתן להזיז את ערכי הסדרה שמאלה או ימינה.

**הזזה מעגלית שמאלה** מתבצעת על ידי שמירת ערכו של המשתנה השמאלי ביותר בסדרה במשתנה זמני ולאחר מכן, השמה של כל ערך של משתנה למשתנה שמשמאלו. כלומר, השמת ערכו של המשתנה השני במשתנה הראשון, השמת ערכו של המשתנה השלישי במשתנה השני, וכך הלאה. לבסוף השמת ערכו של המשתנה הזמני במשתנה האחרון.

**הזזה מעגלית ימינה** מתבצעת על ידי שמירת ערכו של המשתנה הימני ביותר בסדרה במשתנה זמני ולאחר מכן, השמה של כל ערך של משתנה למשתנה שמימינו. כלומר, השמת ערכו של המשתנה הלפני אחרון במשתנה האחרון, וכך הלאה, ולבסוף השמת ערכו של המשתנה הזמני במשתנה הראשון.

**שימו ♥ :** עבור סדרה בת שני איברים האלגוריתמים של התבנית **החלפת ערכים בין שני**

**משתנים** ושל התבנית **הזזה מעגלית בסדרה** הם זהים, כי כדי להזיז מעגלית שמאלה או ימינה שני ערכים בלבד יש למעשה להחליף בין שני ערכי המשתנים. חשוב לציין שזהו מקרה פרטי, ולפעולת ההזזה המעגלית יש משמעות עבור סדרה שבה יותר משני איברים. נציג הזזה מעגלית של סדרה שבה 3 איברים, ובהמשך נראה דוגמאות להזזה מעגלית של סדרה שבה לפחות 3 איברים.

נפריד את מאפייני התבנית **הזזה מעגלית בסדרה** לשתי תת-תבניות: ראשית נציג את מאפייני התבנית **הזזה מעגלית שמאלה בסדרה** ואחר כך נציג את מאפייני התבנית **הזזה מעגלית ימינה בסדרה**.

**שם התבנית:** הזזה מעגלית שמאלה בסדרה

**נקודת מוצא:** שלושה ערכים במשתנים element1, element2 ו-element3

**מטרה:** הזזה מעגלית שמאלה של שלושת המשתנים

**אלגוריתם:**

החלף את ערכי המשתנים element1 ו-element2

החלף את ערכי המשתנים element2 ו-element3

**שם התבנית:** הזזה מעגלית ימינה בסדרה

**נקודת מוצא:** שלושה ערכים במשתנים element1, element2 ו-element3

**מטרה:** הזזה מעגלית ימינה של שלושת המשתנים

**אלגוריתם:**

החלף את ערכי המשתנים element2 ו-element3

החלף את ערכי המשתנים element1 ו-element2

#### שאלה 10

נתונה סדרה של שלושה ערכים: element1, element2, element3.

לפניכם שימוש בתבנית של הזזה מעגלית שמאלה עבור הסדרה:

הזז מעגלית שמאלה את איברי הסדרה element1, element2, element3

א. כתבו אלגוריתם המתאים לתבנית, שאינו משתמש בתבנית החלפת ערכים, ויישמו אותו על ידי כתיבת קטע תוכנית בשפת C#.

ב. כתבו שימוש בתבנית של **הזזה מעגלית ימינה** עבור שלושת הערכים הנתונים, לאחר מכן כתבו אלגוריתם המתאים לתבנית, שאינו משתמש בתבנית החלפת ערכים, ויישמו אותו על ידי כתיבת קטע תוכנית בשפת C#.

#### שאלה 11

במשחק הכיסאות המוזיקליים משתתפים שלושה ילדים היושבים על שלושה כסאות. בעת הפעלת המוזיקה כל ילד זז מעגלית שמאלה ומתיישב בכסא שמשמאלו.

נתון האלגוריתם הבא שהקלט שלו הוא 3 מספרים שלמים המייצגים את מספרי הילדים, והפלט שלו הוא מספרי הילדים לאחר שתי הזזות מעגליות שמאלה:

1. קלט שלושה מספרים שלמים כ-child1, child2, child3

2. הזז מעגלית שמאלה את איברי הסדרה child1, child2, child3

3. הזז מעגלית שמאלה את איברי הסדרה child1, child2, child3

4. הצג כפלט את הערכים של child1, child2, child3

א. מה יהיה הפלט עבור הקלט 6 3 8?

ב. תנו דוגמת קלט שעבורה הפלט יהיה 9 4 2.

ג. באלגוריתם נעשה שימוש כפול בתבנית **הזזה מעגלית שמאלה בסדרה**. כתבו אלגוריתם השקול לאלגוריתם הנתון תוך שימוש **יחיד** בתבנית אחרת.

ד. ישמו את האלגוריתם שכתבתם בסעיף ג' כקטע תוכנית בשפת C#.

## שאלה 12

ביום ספורט היתולי התחרו 4 קבוצות a, b, c, d בארבע תחנות שונות. נקבע כי המעבר בין התחנות ייעשה בצורה מעגלית ימינה. נתון האלגוריתם החלקי הבא, המתאר את החלפת הקבוצות:

1. השם temp-2 אג ערכו של d
2. השם d-2 אג ערכו של \_\_\_\_\_
3. השם c-2 אג ערכו של b
4. השם b-2- \_\_\_\_\_ אג ערכו של a
5. השם a-2- \_\_\_\_\_ אג ערכו של \_\_\_\_\_

א. השלימו את האלגוריתם.

ב. לאחר שכל הקבוצות סיימו את הסיבוב הראשון בתחרות קבעו מארגני יום הספורט שהמעבר בין התחנות בסיבוב השני ייעשה בצורה מעגלית שמאלה. כתבו אלגוריתם שיתאר את החלפת הקבוצות בסיבוב השני.

## שאלה 13

נתון האלגוריתם הבא, שהקלט שלו הוא ארבעה מספרים שלמים:

1. קאוט 4 מספרים שלמים element1, element2, element3, element4-2
2. הזז מעגלית שמאלה את איברי הסדרה element1, element2, element3
3. הזז מעגלית ימינה את איברי הסדרה element1, element3, element4
4. הצג כקאוט אג הערכים element1, element2, element3, element4

א. מה יהיה הפלט עבור הקלט 2 15 7 -6?

ב. תנו דוגמת קלט שעבורה הפלט יהיה 4 6 19 10.

ג. מהי מטרת האלגוריתם?

ד. כתבו אלגוריתם שקול לאלגוריתם הנתון תוך שימוש כפול בתבנית היפוך סדר האיברים בסדרה בת שני ערכים.



## פרק 4 – הרחבה בפיתוח אלגוריתמים

בפרק הקודם הצגנו את התהליך של פיתוח ויישום אלגוריתם בשלבים. בפרק זה נרחיב בכמה מהשלבים: בניית הבעיה, בפירוק המשימה לתת-משימות, ובבדיקת הפתרון עבור דוגמאות קלט שונות. הבעיות שיוצגו בפרק זה יהיו מורכבות יותר מהבעיות שהוצגו בפרק הקודם, וכך, ככל שנתקדם בפרקי הלימוד, הבעיות יהיו מורכבות יותר ויותר, וחשיבות הפתרון בשלבים תהיה משמעותית יותר ויותר.

באמצעות הבעיות שיוצגו בפרק זה ופתרון, נכיר גם פעולות נוספות על ערכים מטיפוס שלם, טיפוס חדש שערכיו הם תווים, ואת המחלקה האחראית לפעולות מתמטיות בשפת C#.

### 4.1 מבט נוסף אל התהליך של פיתוח אלגוריתם ויישום

כזכור, תהליך של פיתוח אלגוריתם ויישום בתוכנית מחשב כולל את השלבים הבאים:

1. ניתוח ראשוני של הבעיה בעזרת דוגמאות
2. פירוק הבעיה לתת-משימות
3. בחירת משתנים, הגדרת תפקידיהם וטיפוסי הערכים שיישמרו בהם
4. כתיבת האלגוריתם
5. יישום האלגוריתם בתוכנית מחשב
6. בדיקת נכונות עבור דוגמאות קלט מגוונות בטבלת מעקב
7. כתיבת התוכנית המלאה, ובדיקתה בהרצה על דוגמאות קלט נוספות

בעת פיתוח האלגוריתם אנו נשפר ונשנה אותו. לעתים בעת העבודה על שלב, מתברר כי יש לתקן שלב קודם (לדוגמה, בעת כתיבת האלגוריתם, מסתבר כי יש להוסיף משתנה). במקרה כזה נחזור אחורה לשלב הקודם ונתקן לפני שנמשיך לשלב הבא. כמו כן, בבדיקת התוכנית עלולות להתגלות שגיאות, ותיקונן עשוי להביא לבחירת משתנים נוספים ולשינוי הוראות האלגוריתם. שיפור של אלגוריתם ושל תוכנית או תיקונים תוך חזרה משלב מתקדם לשלב קודם הוא תהליך טבעי ומקובל.

**שימו** ♥ לשני השלבים האחרונים:

בשלב 6 נבדקת נכונות התוכנית באמצעות מעקב "ידיני" אחר מהלך הביצוע עבור מספר מצומצם של דוגמאות קלט; זאת על מנת לאמת ולקבל ביטחון ראשוני שהתוכנית אכן משיגה את מטרתה המיועדת.

בשלב 7 מורצת התוכנית המלאה במחשב. ההרצה מאפשרת בדיקה מלאה יותר של התוכנית, כיוון שניתן להריץ את התוכנית במהירות עבור דוגמאות קלט רבות ומגוונות.

### הצ'יה 1

**מטרת הבעיה הבאה ופתרונה:** הדגמה מפורטת של פיתוח ויישום בשלבים של אלגוריתם.

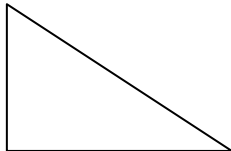
טלי ואודי מעוניינים לבנות גלגלת שתחבר בין החלונות שלהם. הגלגלת מורכבת מחבל כפול באורך של המרחק שבין החלונות. על טלי למדוד את המרחק שבין החלונות כדי לדעת מה אורך החבל שיזדקק לו. טלי בדקה ומצאה שכדי להגיע מהחלון שלה לחלון של אודי עליה לצעוד ישר מספר צעדים, לפנות ימינה ולצעוד מספר צעדים נוספים. טלי גם בדקה ומצאה שאורך כל צעד שלה הוא 42 ס"מ. (0.42 מ').

פתחו אלגוריתם שהקלט שלו הוא מספר הצעדים שעל טלי לצעוד ישר, ומספר הצעדים שעל טלי לצעוד לאחר שפנתה ימינה, והפלט שלו הוא אורך החבל המבוקש. ישמו את האלגוריתם בתוכנית מחשב בשפת C#.

### ניתוח הבעיה בעזרת דוגמאות

נבדוק עבור מספר דוגמאות מה הפלט הרצוי עבור הקלט:

אם טלי צועדת 30 צעדים ישר, ואחר כך 21 צעדים ימינה, ניתן לתאר זאת באמצעות משולש ישר זווית, שניצב אחד שלו הוא 30 צעדים, והשני 21 צעדים. החלון של טלי הוא הקדקוד התחתון הימני של המשולש, והחלון של אודי הוא הקדקוד העליון של המשולש.



המרחק המבוקש הוא בדיוק אורך היתר במשולש זה. לכן, נשתמש במשפט פיתגורס, האומר כי אורך יתר במשולש ישר זווית הוא שורש סכום ריבועי שני הניצבים, כלומר:  $c = \sqrt{a^2 + b^2}$ . לכן, אם טלי צועדת 30 צעדים ישר, ואחר כך 21 צעדים ימינה, ניתן לחשב את אורך החבל המבוקש באופן הבא:  $\sqrt{(30 \cdot 0.42)^2 + (21 \cdot 0.42)^2} = 15.38$ .

לבסוף נכפיל את התוצאה ב-2 (כיוון שהחבל הינו כפול) והערך המתקבל הוא הפלט הנדרש.

ומה אם הנתון השני בקלט הוא 0? כלומר, טלי אינה צריכה לצעוד כלל ימינה, משום שהחלון של אודי נמצא בדיוק מול החלון שלה? במקרה זה, ברור כי המרחק המבוקש ניתן לחישוב ישירות לפי מספר הצעדים הנתון בקלט, אבל מסתבר שהנוסחה הכללית שמצאנו תתאים גם למקרה הפרטי הזה:

$$2 \cdot \sqrt{(30 \cdot 0.42)^2 + (0 \cdot 0.42)^2} = 2 \cdot \sqrt{(30 \cdot 0.42)^2} = 2 \cdot 30 \cdot 0.42$$

תהליך בדיקת הפלט עבור דוגמאות שונות עוזר לנו להבין את מהות הבעיה, להבין מה נדרש מאתנו, לחשוב על התהליכים הדרושים לפתרון הבעיה, ובכך מכוון אותנו לקראת השלבים הבאים של תכנון ושל כתיבת האלגוריתם עצמו.

### פירוק הבעיה לתת-משימות

נפרק את הבעיה לשלוש תת-משימות:

1. קליטת שני מספרים חיוביים שלמים
2. חישוב אורך החבל הנדרש
3. הצגה של אורך החבל כפלט

התת-משימה הראשונה והשלישית הן פשוטות. התת-משימה השנייה היא התת-משימה העיקרית והמורכבת יותר. ניתן לפרק גם אותה לתת-משימות ובכך לפרט יותר את הרעיון לפתרון הבעיה:

- 2.1 הכפלת המספר הראשון ב-0.42, והעלאת התוצאה בריבוע
- 2.2 הכפלת המספר השני ב-0.42, והעלאת התוצאה בריבוע
- 2.3 חיבור שני הערכים שהתקבלו והוצאת שורש מהסכום
- 2.4 הכפלת הערך שהתקבל ב-2

כל הפעולות המפורטות כאן הן פעולות חישוב פשוטות לביצוע, שאינן דורשות ניתוח נוסף. כעת, לאחר פירוק הבעיה לתת-משימות, ברור לנו הרעיון לפתרון הבעיה.

## בחירת משתנים

שלושת המשתנים הראשונים שנבחר דרושים עבור פעולות הקלט והפלט:

**stepForward** – שלם, ישמור את מספר הצעדים קדימה

**stepRight** – שלם, ישמור את מספר הצעדים ימינה

**ropeLength** – ממשי, ישמור את אורך החבל הנדרש הסופי

נשתמש בעוד שני משתנים לשמירת ערכי הביניים של החישובים השונים:

**side1** – ממשי, ישמור את ריבוע האורך של ניצב אחד במשולש

**side2** – ממשי, ישמור את ריבוע האורך של ניצב שני במשולש

## האלגוריתם

לפי החלוקה לתת-משימות ותוך שימוש במשתנים שבחרנו, נקבל את האלגוריתם הבא:

1. קלט מספר שלם גיוכי **stepForward**
2. קלט מספר שלם גיוכי **stepRight**
3. גשג אג מכפול **stepForward** כ-0.42 והשלם כ-**side1**
4. גשג אג מכפול **stepRight** כ-0.42 והשלם כ-**side2**
5. העלה אג **side1** בריבוע
6. העלה אג **side2** בריבוע
7. גשג אג סכום הערכים שהגקבל
8. גשג אג השורש הריבועי של הערך שהגקבל
9. הכפול אג הערך שהגקבל כ-2 והשלם כ-**ropeLength**
10. הצג כפול אג ערכו של **ropeLength**

## יישום האלגוריתם

הוראה 8 באלגוריתם כוללת חישוב שורש ריבועי. כיצד מבצעים זאת בשפת C#:

את הוראות 5 עד 8 ניתן לבטא בביטוי:



הפעולה **Sqrt** מקבלת בתוך הסוגריים מספר מטיפוס ממשי, ומחזירה מספר ממשי השווה לשורש הריבועי של המספר שקיבלה.

פעולה זו שייכת למחלקה **Math**. מחלקה זו נמצאת במרחב השמות **System** ולכן ההכרזה על השימוש ב-**System** מאפשר להשתמש בה ישירות.

את הערך 0.42 נגדיר כקבוע, כדי ליצור תוכנית קריאה ועמידה בפני שינויים.

הנה היישום של הוראות האלגוריתם:

```
1. Console.WriteLine("Enter number of steps forward: ");
2. stepForward = int.Parse(Console.ReadLine());
3. Console.WriteLine("Enter number of steps to the right: ");
4. stepRight = int.Parse(Console.ReadLine());
5. side1 = stepForward * STEP_SIZE;
6. side2 = stepRight * STEP_SIZE;
7. ropeLength = 2 * Math.Sqrt((side1 * side1) + (side2 * side2));
```

```
8. Console.WriteLine("The length of the rope is: {0}", ropeLength);
```

## מעקב

נעקוב אחר מהלך ביצוע משפטי התוכנית עבור הקלט 21 30 :

שורה לביצוע	stepForward	stepRight	side1	side2	ropeLength	פלט
1	?	?	?	?	?	Enter number of steps forward:
2	30	?	?	?	?	
3	30	?	?	?	?	Enter number of steps to the right:
4	30	21	?	?	?	
5	30	21	12.6	?	?	
6	30	21	12.6	8.82	?	
7	30	21	12.6	8.82	30.76	
8	30	21	12.6	8.82	30.76	The length of the rope is: 30.76

על פי המעקב הזה התוכנית מבצעת את מטרותיה.

## התוכנית המלאה

```
/* קלט: מספר הצעדים בין הבתים של טלי ושל אודי */
/* פלט: אורך החבל הנדרש */
using System;
public class Rope
{
    public static void Main()
    {
        // הגדרת משתנים
        int stepForward, stepRight; // המספרים הניתנים כקלט
        double side1, side2;        // ערכי הניצבים של המשולש
        double ropeLength;          // אורך החבל
        const double STEP_SIZE = 0.42; // גודל צעד
        // קלט
        Console.Write("Enter number of steps forward: ");
        stepForward = int.Parse(Console.ReadLine());
        Console.Write("Enter number of steps to the right: ");
        stepRight = int.Parse(Console.ReadLine());
        // חישוב אורך החבל
        side1 = stepForward * STEP_SIZE;
        side2 = stepRight * STEP_SIZE;
        ropeLength = 2 * Math.Sqrt((side1 * side1) + (side2 * side2));
        // פלט
        Console.WriteLine("The rope length is: {0}", ropeLength);
    } // Main
} // class Rope
```

## סוף פתרון בעיה 1

פתרון הבעיות הבאות בפרק, בעיות מורכבות יותר מבעיה 1, יסתמך על התהליך שהצגנו. מעתה ואילך נניח שכאשר נדרשים פיתוח אלגוריתם ויישומו בשפת תכנות, נדרש פיתוח בשלבים, ולכן לא נציין זאת במפורש.

## שאלה 4.1

דן, בן וכן קיבלו כל אחד דמי חנוכה מהוריהם. כל אחד קיבל סכום הגדול מ-20 ₪. שלושת החברים החליטו לאחד את כל סכום הכסף לקופה אחת ולקנות יחדיו כדורסל שמחירו 50 ₪. ביתרת הכסף יקנו מסטיקים שעלותם 1 ₪ כל אחד. פתחו אלגוריתם המקבל כקלט את דמי החנוכה שקיבל כל אחד מהחברים, ומציג כפלט את כמות המסטיקים שאפשר לקנות. ישמו את האלגוריתם בתוכנית בשפת C#.

## המחלקה המתמטית

בפתרון בעיה 1 חישבנו שורש ריבועי על ידי שימוש בפעולה `Math.Sqrt`.

זהו שמה המלא של הפעולה `Sqrt` השייכת למחלקה המתמטית `Math`. כאמור, לפעולה אנו מעבירים ערך (פרמטר) שהוא מטיפוס ממשי, והיא מחזירה את השורש הריבועי של המספר שניתן לה. טיפוס הערך המוחזר הוא ממשי.

המחלקה המתמטית `Math` מכילה עוד פעולות מתמטיות רבות אשר נתונות לשימושנו, וכמוה יש מחלקות נוספות שנוכל להשתמש בהן לפעולות עזר מסוגים שונים.

### פעולות שימושיות מהמחלקה המתמטית `Math`

דוגמה		טיפוס ערך מוחזר	טיפוס פרמטרים	תיאור הפעולה	הפעולה
הערך המוחזר	הפעולה				
63	<code>Math.Abs(63)</code>	שלם	שלם	ערך מוחלט	<code>Abs(num)</code>
12.7	<code>Math.Abs(-12.7)</code>	ממשי	ממשי		
2.5	<code>Math.Sqrt(6.25)</code>	ממשי	ממשי	שורש ריבועי	<code>Sqrt(num)</code>
9.0	<code>Math.Pow(3,2)</code>	ממשי	ממשי, ממשי	חזקה $num1^{num2}$	<code>Pow(num1, num2)</code>
3	<code>Math.Min(3,8)</code>	שלם	שלם, שלם	הקטן מבין השניים	<code>Min(num1, num2)</code>
8.0	<code>Math.Min(8.0, 8.8)</code>	ממשי	ממשי, ממשי		
8	<code>Math.Max(3,8)</code>	שלם	שלם, שלם	הגדול מבין השניים	<code>Max(num1, num2)</code>
8.8	<code>Math.Max(8.0, 8.8)</code>	ממשי	ממשי, ממשי		
8	<code>Math.Round(7.9)</code>	שלם	ממשי	עיגול מספר ממשי	<code>Round(num)</code>

בנוסף לפעולות המתוארות בטבלה זו שייכות למחלקה `Math` פעולות רבות נוספות. אתם מוזמנים לחפש ברשת, או בדפי ההדרכה של מיקרוסופט (MSDN), ולעיין בממשק של המחלקה המתמטית בשפת C# (`Math`) ולמצוא פעולות נוספות השייכות למחלקה זו.

עד כה הכרנו כמה מחלקות שימושיות: המחלקה `Console` האחראית לפעולות הקלט והפלט; המחלקה `Math` האחראית לפעולות מתמטיות. בהמשך לימודיכם תפגשו עוד מחלקות רבות המשמשות לצרכים שונים.

#### שאלה 4.2

פתחו אלגוריתם אשר הקלט שלו הוא גובהי שני תלמידים, נתונים במספרים ממשיים, והפלט שלו הוא הערך המוחלט של הפרשי הגבהים שלהם. ישמו את האלגוריתם בתוכנית בשפת C#. שימו לב לבחירת דוגמאות קלט מגוונות.

#### שאלה 4.3

שנו את התוכנית שכתבתם כפתרון לשאלה 4.2 כך שפלט התוכנית יהיה גובה התלמיד הנמוך מבין השניים.

#### שאלה 4.4

פתחו אלגוריתם אשר הקלט שלו הוא אורך ורוחב צלעות של מלבן (מספרים שלמים) והפלט שלו הוא שטח המלבן ואורך אלכסון המלבן.

## 4.2 פעולות חלוקה בשלמים

בפרק 3 אמרנו כי הגדרת טיפוס כוללת גם את פירוט הפעולות הניתנות לביצוע על ערכי הטיפוס. בפרק 3 הצגנו את הטיפוסים שלם וממשי, ורשימת הפעולות שהצגנו עבור כל אחד מהם כללה את הפעולות החשבוניות המוכרות. בסעיף זה נכיר פעולות חשבוניות המוגדרות רק עבור ערכים מטיפוס שלם.

### קצ'ה 2

**מטרת הבעיה הבאה ופתרונה:** שימוש בפעולות לחישוב מנה ושארית בחלוקת מספרים שלמים.

ליונתן אוסף מטופח של גולות. הוא שומר את כל האוסף בקופסה, בקבוצות של 20, כלומר, כל קבוצה של 20 גולות ארוזה בשקית נפרדת. הגולות הנותרות מפוזרות בתחתית הקופסה. פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר הגולות שיש ליונתן, והפלט הוא מספר השקיות שיש בקופסה ומספר הגולות המפוזרות בתחתית. ישמו את האלגוריתם בתוכנית מחשב בשפת C#.

למשל, עבור הקלט 135, הפלט הדרוש הוא: 6 15 כלומר, 6 שקיות ו-15 גולות פזורות. (משום ש- $6 \cdot 20 + 15 = 135$ ).

### ניתוח הבעיה בעזרת דוגמאות

נבדוק את הפלט הרצוי עבור כמה דוגמאות קלט:

קלט	פלט
13	0 שקיות ו-13 פזורות
60	3 שקיות ו-0 פזורות
64	3 שקיות ו-4 פזורות
82	השלימו: _____ שקיות ו-_____ פזורות

על פי הדוגמאות אנו רואים כי מספר השקיות הוא מספר הפעמים שנכנס המספר 20 במספר הגולות הכולל, שנקלט מהקלט. הגולות הפזורות הן השארית, אלה שנותרו אחרי שאספנו קבוצות של 20 גולות ככל שיכולנו.

כלומר עלינו לבצע כאן שתי פעולות, כל אחת מהן מתבצעת על מספרים שלמים ומחזירה מספר שלם. הפעולה הראשונה מקבלת מספר  $x$  (במקרה שלנו, המספר הכולל של הגולות) ומספר  $y$  (במקרה שלנו, מספר הגולות בכל שקית, כלומר 20) ומחזירה את מספר הפעמים שהמספר  $y$  נכנס במספר  $x$ . הפעולה השנייה משלימה אותה, במובן מסוים: היא מקבלת  $x$  ו- $y$  (כמו קודם) ומחזירה את מה שנותר אחרי שמחסירים מ- $x$  כפולות שלמות של  $y$  ככל שניתן.

אלו הן פעולות המוגדרות על מספרים שלמים, כלומר על ערכים מטיפוס שלם: הפעולה הראשונה מחשבת את המנה של  $x$  חלקי  $y$ . הפעולה השנייה היא פעולת השארית (modulus).

בעברית מנה היא תוצאה של פעולת חלוקה. קיים קשר בין הפעולות האלו לפעולת החלוקה הרגילה. כאמור, אם נבצע חלוקה רגילה של מספר הגולות  $x$  במספר 20, לא נקבל בהכרח מספר שלם. למשל, אם יש 105 גולות,  $105/20=5.25$ . אבל אם ננסה למלא מתוך 105 גולות כמה שיותר שקיות של 20 גולות, נצליח למלא בדיוק 5 שקיות. זהו בדיוק החלק השלם של תוצאת החלוקה  $105/20$ .

שתי פעולות אלו נקראות פעולות חלוקה בשלמים:

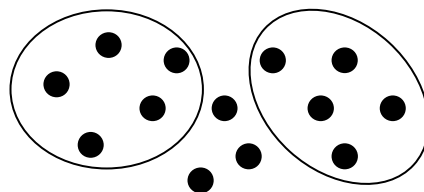
**פעולות החלוקה בשלמים** מוגדרות רק על מספרים שלמים, וגם תוצאת הפעלתן היא מספר שלם:

**מנת החלוקה** של מספר שלם  $x$  במספר שלם  $y$  שווה לחלק השלם של  $x/y$ , ובמילים אחרות, שווה למספר הפעמים שהערך  $y$  נכנס בערך  $x$ .

**שארית החלוקה** (modulus) של מספר שלם  $x$  במספר שלם  $y$  מבטאת את השארית הנוותרת לאחר חלוקה בשלמים של  $x$  ב- $y$ , ובמילים אחרות, מבטאת את מה שנותר אחרי שמפחיתים מ- $x$  כפולות שלמות של  $y$  ככל שניתן.

השוני בין פעולות חלוקה בשלמים ובין פעולת החלוקה במספרים ממשיים מדגים את האבחנה בין הטיפוסים שלם לממשי.

למשל מנת החלוקה של 13 ב-5 שווה ל-2, משום ש-5 נכנס ב-13 פעמיים (ואכן  $13/5=2.6$ ) והחלק השלם הוא 2. לעומת זאת, שארית החלוקה של 13 ב-5 שווה ל-3, משום שאחרי שנפחית מ-13 שתי כפולות של 5 (כלומר 10), נקבל 3. נדגים זאת בעזרת האיור הבא, המציג חלוקת 13 נקודות לקבוצות של 5:



בחלוקה של 13 לקבוצות של 5 אנו אכן מקבלים כי מנת החלוקה היא 2, כיוון שהתקבלו שתי קבוצות שלמות בגודל 5 כל אחת. שארית החלוקה היא 3, כיוון שנשארו שלוש נקודות שלא שייכות לאף קבוצה.

את בעיית החלוקה הזו ניתן לראות באופן נוסף: אם נרצה לחלק 13 סוכריות באופן שווה ל-5 ילדים, כמה סוכריות יקבל כל ילד? כמה יישארו לנו? והתשובה תהיה זהה – כל ילד יקבל 2 סוכריות, ולנו יישארו 3.

## פירוק הבעיה לתת-משימות

נפרק את הבעיה לארבע תת-משימות :

1. קליטת מספר הגולות
2. חישוב מספר השקיות
3. חישוב מספר הגולות הפזורות
4. הצגה של מספר השקיות ושל הגולות הפזורות כפלט.

הערכים בעיבוד הנתונים הם מספרים שלמים. אין לנו כאן עניין במספרים ממשיים, ולכן הפעולה שנרצה להשתמש בה לביצוע התת-משימה השנייה והשלישית איננה פעולת חלוקה רגילה, כיוון שפעולה כזאת תיתן תוצאה מטיפוס ממשי. למשל תוצאה של  $64/20$  תיתן 3.2, אך מספר זה אינו נותן לנו את מספר השקיות המלאות ואת מספר הגולות הפזורות. הפעולות המתאימות הן פעולות החלוקה בשלמים שהוצגו לעיל.

את מספר השקיות המלאות נחשב כמנה של חלוקת מספר הגולות ב-20; את מספר הגולות הפזורות נחשב כשארית של אותה חלוקה. שתי פעולות אלה הן שתי פנים של תבנית שימושית: חלוקת כמות פריטים לקבוצות בגודל נתון.

## בחירת משתנים

- amount – שלם, ישמור את המספר הניתן כקלט, מספר הגולות.
- bags – שלם, ישמור את מספר השקיות המכילות 20 גולות כל אחת.
- remainder – שלם, ישמור את מספר הגולות שנשארו פזורות.

## האלגוריתם

1. קלט את מספר הגולות amount
2. גש את מספר השקיות המלאות ביישוב מנה הגולות של amount ב-20 והשם את הגולות ב-bags
3. גש את מספר הגולות שנותרו פזורות ביישוב שארית הגולות של amount ב-20 והשם את הגולות ב-remainder
4. הצג כפלט את הערך bags ואת הערך remainder

## יישום האלגוריתם

בשפת C#, הפעולה לחישוב שארית החלוקה של הערך a בערך b נכתבת בצורה  $a \% b$ , כאשר גם a וגם b הם מטיפוס שלם.

למשל:  $12 \% 4 = 0$ ,  $3 \% 5 = 3$ ,  $7 \% 3 = 1$

בשפת C#, הפעולה לחישוב מנה החלוקה של הערך a בערך b נכתבת בצורה  $a / b$ , בתנאי שגם a וגם b הם מטיפוס שלם.

למשל:  $12 / 4 = 3$ ,  $13 / 5 = 2$ ,  $7 / 3 = 2$ .



שימו לב שאותו סימן משמש ב-C# הן לפעולת חלוקה במספרים ממשיים והן לפעולת חישוב של מנת החלוקה בשלמים. אם כך, איך נבדיל בין פעולת חישוב של מנת החלוקה בשלמים לבין פעולת חלוקה בממשיים? איך נדע אם הביטוי  $13/5$  ערכו שווה ל-2 או ל-2.6?

סוג החלוקה נקבע על פי חוק פשוט:

חלוקת שלם בשלם תתפרש **תמיד** כפעולת חלוקה בשלמים.  
כל אחת מהחלוקות הבאות מתפרשת כפעולת חלוקה של מספרים ממשיים:

שלם/ממשי  
ממשי/שלם  
ממשי/ממשי

למשל  $13/5=2$ ,  $13.0/5=2.6$  ו- $13/5.0=2.6$ .

ניישם את האלגוריתם בשימוש בפעולות החלוקה בשלמים בשפה, ובהגדרת קבוע (PER\_BAG) עבור מספר הגולות בשקית:

```
1. Console.Write("Enter amount of marbles: ");
2. amount = int.Parse(Console.ReadLine());
3. bags = amount / PER_BAG;
4. remainder = amount % PER_BAG;
5. Console.WriteLine("There are {0} bags, {1} are left over", bags, remainder);
```

## מעקב

נעקוב אחר מהלך ביצוע משפטי התוכנית עבור הקלט 92:

שורה לביצוע	number	bags	remainder	פלט
1. Console.Write(...)	?	?	?	Enter amount of marbles:
2. amount = int.Parse...	92	?	?	
3. bags = amount / PER_BAG	92	4	?	
4. remainder = amount % PER_BAG	92	4	12	
5. Console.WriteLine(...)	92	4	12	There are 4 bags, 12 are left over

## התוכנית המלאה

```
/*
התוכנית קולטת את מספר הגולות שיש ליונתן
ונותנת כפלט את מספר השקיות המלאות, ואת מספר הגולות שנותרו פזורות
*/
using System;
public class MarblesInBox
{
    public static void Main()
    {
        // הגדרת משתנים
        int amount; // כמות הגולות
        int bags; // מספר שקיות
        int remainder; // מספר גולות פזורות
    }
}
```

```

const int PER_BAG = 20;          // מספר הגולות בשקית אחת
// קלט
Console.WriteLine("Enter amount of marbles: ");
amount = int.Parse(Console.ReadLine());
// חישוב מספר השקיות
bags = amount / PER_BAG;
// חישוב מספר הגולות הפזורות
remainder = amount % PER_BAG;
// פלט
Console.WriteLine("There are {0} bags, {1} are left over",
                  bags, remainder);

} // Main
} // class MarblesInBox

```

## סוף כתרון בעיה 2

### שאלה 4.5

בנו טבלת מעקב אחר מהלך ביצוע התוכנית MarblesInBox עבור הקלט 123.

### שאלה 4.6

יונתן בעל הגולות החליט לשדרג את אופן האחסון של האוסף ושינה את פקודות התוכנית שבפתרון בעיה 2 כך שגם מספר הגולות בשקית יהיה חלק מהקלט, ולא קבוע:

```

Console.WriteLine("Enter amount of marbles: ");
amount = int.Parse(Console.ReadLine());
Console.WriteLine("Enter capacity: ");
capacity = int.Parse(Console.ReadLine());
bags = amount / capacity;
remainder = amount % capacity;
Console.WriteLine("There are {0} bags, {1} are left over",
                  bags, remainder);

```

מה יהיה פלט הפתרון המשודרג לבעיית אחסון הגולות של יונתן עבור הקלט 30 125? מה יהיה הפלט עבור הקלט 50 200?

### שאלה 4.7

השלימו את תוצאות הפעולות הבאות של חלוקה בשלמים:

- א.  $7/3 = \underline{\hspace{2cm}}$
- ב.  $7\%3 = \underline{\hspace{2cm}}$
- ג.  $20/4 = \underline{\hspace{2cm}}$
- ד.  $20\%4 = \underline{\hspace{2cm}}$
- ה.  $3/7 = \underline{\hspace{2cm}}$
- ו.  $3\%7 = \underline{\hspace{2cm}}$

### שאלה 4.8

נתון קטע התוכנית הבא, שהקלט שלו הוא שני מספרים שלמים, והמשתנים בו הם מטיפוס שלם:

```

Console.WriteLine("Enter first number: ");
num1 = int.Parse(Console.ReadLine());
Console.WriteLine("Enter second number: ");
num2 = int.Parse(Console.ReadLine());
a = num1 / num2;

```

```
b = num1 % num2;
Console.WriteLine("a: {0} b: {1}", a, b);
```

א. רשמו את פלט קטע התוכנית עבור כל אחד מהקלטים הבאים (משמאל לימין):

1. 5 2

2. 5 4

3. 5 5

ב. תנו דוגמה לקלט עבורו הפלט יהיה a: 2 b: 2

להעמקה בתבנית **חלוקת כמות פריטים לקבוצות בגודל נתון** פנו לסעיף התבניות המופיע בסוף הפרק.

## עוד על פעולת השארית

? בשאלה 4.7, בסעיף ד, התקבל הערך 0. מה משמעות הביטוי  $4 \% 20 = 0$ ? באופן כללי, מתי פעולת שארית נותנת תוצאה שווה ל-0?

התוצאה 0 מצביעה על כך שהמספר הראשון (במקרה זה 20) מתחלק במספר השני (במקרה זה 4) **ללא שארית**. כלומר, המספר 4 הוא אחד המחלקים של 20.

אם כך כיצד נבדוק אם מספר נתון הוא זוגי? מספר זוגי הוא מספר המתחלק ב-2 ללא שארית. לכן, נוכל לחשב את שארית החלוקה של המספר הנתון ב-2. אם המספר אכן זוגי השארית שנקבל תהיה שווה ל-0.

**שימו ♥:** שארית החלוקה של ערך x בערך y **אינה** שווה לחלק הלא שלם המתקבל מפעולת החלוקה בממשיים של x ב-y. למשל, אם נחשב  $13/5$  בפעולת חלוקה ממשיית נקבל 2.6. לעומת זאת, שארית החלוקה  $13 \% 5$  היא 3, ולא 6.

? אם נחלק מספר ב-2 ונבדוק את השארית. אילו מספרים יכולים להתקבל כתוצאה? נבדוק את המקרים הבאים:

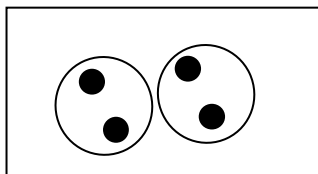
$$4 \% 2 = 0$$

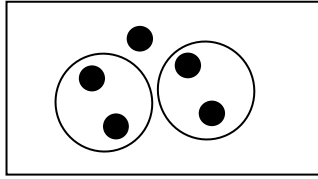
$$5 \% 2 = 1$$

$$6 \% 2 = 0$$

$$7 \% 2 = 1$$

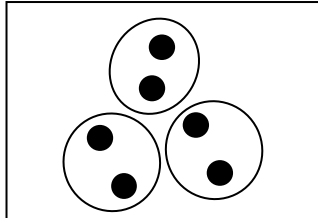
המספרים האפשריים כתוצאה הם אך ורק 0 ו-1. לא ייתכן כי נקבל תוצאה הגדולה מ-1. נראה זאת שוב בעזרת איור: כאשר אנו מחלקים מספר כלשהו של נקודות לקבוצות של שתי נקודות כל אחת, ובודקים כמה נקודות לא נכללות באף קבוצה, ייתכן כי לא נשארת אף נקודה ללא קבוצה, למשל, עבור  $4 \% 2$ :





ייתכן כי נשארת נקודה בודדת ללא קבוצה, למשל, עבור  
2 % 5 :

לא ייתכן שיישארו שלוש נקודות מחוץ לקבוצות, משום שאז ניתן לקחת שתיים מהן וליצור קבוצה חדשה: אם נגדיל את מספר הנקודות ב-1 ונחשב את 2 % 6, נראה כי נוצרה קבוצה חדשה מהנקודה הבודדת שנותרה קודם ומהנקודה החדשה:



אם כך, השארית של חלוקת מספר כלשהו ב-2 לא יכולה להיות גדולה מ-1.  
מה השארית של חלוקת מספר כלשהו ב-3? באותו אופן בדיוק ניתן להראות כי השארית של חלוקת מספר כלשהו ב-3 לא יכולה להיות גדולה מ-2.  
ובאופן כללי:

הערכים האפשריים לשארית של חלוקת מספר כלשהו ב- $n$  הם בין 0 ל- $(n-1)$ .

#### שאלה 4.9

השלימו:

- א. הערכים האפשריים לשארית החלוקה של מספר כלשהו ב-5 הם \_\_\_\_\_.
- ב. הערכים האפשריים לשארית החלוקה של מספר כלשהו ב-6 הם \_\_\_\_\_.

#### שאלה 4.10

נתון קטע התוכנית הבא אשר הקלט שלו הוא פרק זמן הנתון בשעות, השמור במשתנה `hours`, והפלט שלו הוא מספר היממות השלמות והשעות הנותרות בפרק הזמן הנתון. השלימו את קטע התוכנית.

```
// קלט
hours = int.Parse(Console.ReadLine());
days = _____;
hoursLeft = _____;
// פלט
Console.WriteLine("There are {0} days and {1} hours", days,
hoursLeft);
```

#### שאלה 4.11

פתחו אלגוריתם שהקלט שלו הוא פרק זמן הנתון בדקות, והפלט שלו הוא מרכיבי השעות והדקות בפרק זמן זה. ישמו אותו בשפת התכנות C#.  
למשל, עבור הקלט 90 הפלט יהיה: 1 שעות, 30 דקות.  
עבור הקלט 30 הפלט יהיה: 0 שעות, 30 דקות.  
עבור הקלט 300 הפלט יהיה: 5 שעות, 0 דקות.

#### שאלה 4.12

טייס חלל יוצא ביום ראשון לטיסה ממושכת בחלל. עבור כל יום טיסה, יש לצייד את הטייס ב-3.8 ליטרים של מים.

פתחו אלגוריתם שהקלט שלו הוא מספר ימי הטיסה בחלל, והפלט שלו הוא מספר השבתות שייעדר הטייס מביתו, וכמות המים שיש לציידו לקראת הטיסה. ישמו את האלגוריתם בשפת התכנות C#.

#### שאלה 4.13

פקיד חרוץ ממלא טופס במשך 10 דקות. עבור כל טופס שהפקיד ממלא הוא מקבל שכר של 6.3 ₪. הפקיד עובד ללא הפוגה.

פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר הטפסים שעל הפקיד למלא, והפלט שלו הוא משך העבודה, בשעות ובדקות, והשכר שהפקיד יקבל. ישמו את האלגוריתם בשפת C#. למשל עבור הקלט 55 יהיה הפלט: משך העבודה הוא: 9 שעות ו-10 דקות והשכר הוא: 346.5.

#### שאלה 4.14

בפס ייצור במפעל לייצור משאיות מרכיבים גלגלים במשאיות. פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר הגלגלים הכולל בפס הייצור ומספר גלגלים למשאית, והפלט שלו הוא מספר המשאיות שיורכבו להן גלגלים. למשל עבור הקלט 6 1000, שפירושו 1000 גלגלים בסך הכול בפס הייצור, ו-6 גלגלים למשאית, יהיה הפלט 166.

## המרת ערך שלם לממשי

ראובן החליט, בנדיבות לבו, כי את דמי החנוכה שהוא מקבל מסבתו, הוא יחלק באופן שווה בין החברים שיגיעו למסיבת החנוכה שלו.

למשל, אם יקבל ראובן מסבתו 25 ₪, ולמסיבה יגיעו 10 חברים, יקבל כל אחד מהחברים 2.5 ₪. ראובן כתב את ההוראות הבאות כדי לחשב את הפלט הנדרש:

```
int money, friends;
double gift;
1. Console.Write("Enter sum of money: ");
2. money = int.Parse(Console.ReadLine());
3. Console.Write("Enter number of friends: ");
4. friends = int.Parse(Console.ReadLine());
5. gift = money / friends;
6. Console.WriteLine("Every friend will get: {0} shekels", gift);
```

נעקוב אחר מהלך ביצוע משפטי התוכנית עבור הקלט 25 10:

שורה לביצוע	money	friends	gift	פלט
1. Console.Write(...)	?	?	?	Enter sum of money:
2. money = int.Parse(...)	25	?	?	
3. Console.Write(...)	25	?	?	Enter number of friends:
4. money = int.Parse(...)	25	10	?	
5. gift = money / friends	25	10	2	
6. Console.WriteLine(...)	25	10	2	Every friend will get: 2 shekels

מדוע הפלט שגוי? ניזכר בהגדרת פעולת החילוק (/) בשפת C#: כאשר מחלקים שלם בשלם, פעולת החילוק מתפרשת **תמיד** כחלוקה בשלמים.

נתבונן בשורה השלישית: בביטוי `money / friends` מתבצעת חלוקה של שני ערכים מטיפוס שלם. אמנם תוצאת החלוקה מושמת במשתנה מטיפוס ממשי, אך דבר זה נעשה רק לאחר ביצוע פעולת החלוקה, וזו הייתה פעולת חלוקה בשלמים ולא פעולת חלוקה בממשיים, כפי שנדרש. עבור הקלט 10 25, מחושבת מנת החלוקה של 25 ב-10, והתוצאה המתקבלת היא שלמה (2). תוצאה זו מוצבת במשתנה `gift` המכיל כעת את הערך הממשי 2.0 (הצורה הממשית של המספר 2). אם כך, התשובה שקיבלנו היא שגויה, מכיוון שמחלקת 25 ב-10 ציפינו במקרה זה לקבל 2.5 ולא 2.0.

כדי לפתור את הבעיה יש להודיע כי **אף על פי** שהמשתנה `money` הוא משתנה מטיפוס שלם השומר ערך שלם, יש להתייחס זמנית אל ערכו כאל ערך ממשי. בכך החלוקה הופכת להיות חלוקה של ערך ממשי בערך שלם – לכן היא חלוקה בממשיים, כפי שנדרש. דבר זה מתבצע בשפת C# באמצעות פעולת **המרה** (casting).

לפני הביטוי שאת ערכו אנו מעוניינים לפרש זמנית כממשי ולא כשלם (במקרה זה, הביטוי `money`), נוסיף את הוראת ההמרה שמשמעותה: המר, רק לצורך החישוב הנוכחי, את ערכו של ביטוי זה לערך ממשי. ההוראה מיושמת בכתיבת שם הטיפוס שנרצה להמיר אליו, עטוף בסוגריים, לפני הביטוי המומר:

```
gift = (double)money / friends;
```

**שימו ♥:** המשתנים המעורבים בביטוי המומר (במקרה זה, המשתנה `money`) **אינם** הופכים מרגע זה למשתנים מטיפוס ממשי. הם נשארים משתנים מטיפוס שלם בדיוק כפי שהיו עד כה. ההמרה גורמת לכך שרק **בעת חישוב הביטוי**, ערכם נראה כממשי ולא כשלם.

בהתאם לכך, קטע הקוד הנכון הוא:

```
int money, friends;
double gift;
Console.WriteLine("Enter sum of money: ");
money = int.Parse(Console.ReadLine());
Console.WriteLine("Enter number of friends: ");
friends = int.Parse(Console.ReadLine());
gift = (double)money / friends;
Console.WriteLine("Every friend will get: {0} shekels", gift);
```

#### שאלה 4.15

לפניכם סדרת הוראות. השלימו בטבלה את ערכי המשתנים לאחר כל הוראה.

	num	fnum
<code>int num, x = 22, y = 5;</code>		
<code>double fnum;</code>		
<code>num = x / y;</code>		
<code>fnum = x / y;</code>		
<code>fnum = (double)x / y;</code>		
<code>num = x % y;</code>		

#### שאלה 4.16

אהרון מעוניין לדעת את ממוצע ציוניו במקצועות היסטוריה, תנ"ך וספרות. פתחו אלגוריתם המקבל כקלט את שלושת הציונים במקצועות אלו (מספרים שלמים בין 0 ל-100), ומציג כפלט את הממוצע של שלושת הציונים. ישמו את האלגוריתם בשפת התכנות C#.

#### שאלה 4.17

דינה מעוניינת לדעת את הציון הכולל שלה במדעי המחשב. ידוע כי שתי היחידות הראשונות מהוות 33% מהציון, היחידה השלישית מהווה 17% מהציון, ושתי היחידות האחרונות מהוות 50% מהציון הכללי. פתחו אלגוריתם המקבל כקלט את שלושת ציוניה של דינה במדעי המחשב (בשתי היחידות הראשונות, ביחידה השלישית ובשתי היחידות האחרונות) כמספרים שלמים בין 0 ל-100, ומציג כפלט את ציונה הכולל. ישמו את האלגוריתם בשפת התכנות C#.

### פירוק מספר דו-ספרתי לספרותיו

בסעיף זה נכיר שתי תבניות שימושיות. התבנית של פירוק מספר לספרותיו תודגם בבעיה 3. תבנית שמסלימה אותה במובן מסוים היא בניית מספר מִסְפָּרוֹת, ואליה נתייחס בשאלה 4.19.

### חצייה 3

**מטרת הבעיה הבאה ופתרונה:** פיתוח ויישום בשלבים של אלגוריתם ושימוש בפעולות חלוקה בשלמים לצורך פירוק מספר דו-ספרתי לספרותיו.

פתחו אלגוריתם שהקלט שלו הוא מספר דו-ספרתי, חיובי או שלילי, והפלט שלו הוא סכום ספרות המספר. ישמו את האלגוריתם בתוכנית מחשב בשפת C#.

### ניתוח הבעיה בעזרת דוגמאות

נבדוק עבור מספר דוגמאות מה הפלט הרצוי עבור הקלט:

קלט	פלט
71	8
-53	8
49	13

גם במקרה זה, תהליך בדיקת הפלט עבור דוגמאות שונות מסייע לנו בהבנת משמעות הבעיה ובזיהוי התהליכים הדרושים לפתרונה. למשל עבור הקלט 71 והקלט 53 קיבלנו את אותו הפלט. כעת אנו רואים כי ייתכנו קלטים שונים אשר מתאים להם אותו פלט. לקלט -53 ולקלט 53 מתאים אותו פלט, וכך אנו רואים כי אין חשיבות לסימנו של המספר שבקלט. אלו הן אבחנות חשובות לצורך כתיבת אלגוריתם נכון לפתרון הבעיה.

### פירוק הבעיה לתת-משימות

את הבעיה הזו נפרק לשלוש תת-משימות:

1. קליטת מספר דו-ספרתי חיובי או שלילי
2. פירוק המספר לספרותיו וסיכום הספרות
3. הצגה של סכום הספרות כפלט

ניתן לפרק גם את התת-משימה השנייה לתת-משימות ובכך לפרט יותר את הרעיון לפתרון הבעיה:

- 2.1 פירוק המספר לספרותיו
- 2.2 סיכום הספרות

כיצד נפרק את המספר לספרותיו?  
ידוע לנו כי המספר הוא דו-ספרתי, ולכן פירוקו לספרותיו פירושו חישוב ספרת העשרות וספרת האחדות. נפרק אם כן את תת-משימה 2.1, פירוק המספר לספרותיו, כך:

2.1.1. חישוב ספרת העשרות של המספר

2.1.2. חישוב ספרת האחדות של המספר

❓ כיצד נבצע פעולות אלו?

נוכל להיעזר בפעולות חלוקה בשלמים, של המספר הדו-ספרתי הנתון במספר 10:

שארית החלוקה של מספר שלם **כלשהו** ב-10 נותנת תמיד את **ספרת האחדות** של המספר.  
מנת החלוקה של מספר שלם **דו-ספרתי** חיובי ב-10 נותנת תמיד את **ספרת העשרות** של המספר.

למשל  $10 \times 24\% = 2$  ו-  $24/10 = 2$ .

**שימו** ♥: חישוב מנת החלוקה של מספר דו-ספרתי ב-10 נותנת את ספרת העשרות של המספר, אם הוא חיובי, אך לא אם הוא שלילי! למשל  $-20/10 = -2$ , ולא 2!

לכן יש לטפל באופן שונה במספר שלילי.

בדוגמאות ראינו כי הפלט עבור מספר חיובי ועבור המספר הנגדי שלו צריך להיות זהה. לכן, כל שעלינו לעשות עבור מספר שלילי, הוא "להיפטר" מהסימן -, כלומר להפוך את המספר לחיובי, עוד לפני פירוקו לספרותיו.

❓ כיצד נהפוך מספר שלילי לחיובי?

נוכל לעשות זאת על ידי חישוב ערכו המוחלט של המספר לפני פירוקו לספרותיו.

אם כן, בסך הכול, תת-משימה 2.1, פירוק המספר לספרותיו תורכב משלוש תת-משימות:

2.1.1. חישוב ערכו המוחלט של המספר

2.1.2. חישוב ספרת העשרות של המספר

2.1.3. חישוב ספרת האחדות של המספר

כעת, לאחר פירוק הבעיה לתת-משימות ברור לנו הרעיון לפתרון הבעיה וקל לכתוב את האלגוריתם עצמו.

## בחירת משתנים

שני המשתנים הראשונים שנבחר דרושים לנו עבור פעולות הקלט והפלט:

**num** – שלם, ישמור את המספר הניתן כקלט

**sum** – שלם, ישמור את סכום ספרות המספר

המשתנים לשמירת החישובים השונים:

**absNum** – שלם, ישמור את ערכו המוחלט של המספר

**tens** – שלם, ישמור את ספרת העשרות של המספר

**units** – שלם, ישמור את ספרת האחדות של המספר



## האלגוריתם

האלגוריתם, לפי החלוקה לתת-משימות ובשימוש במשתנים שבחרנו, יהיה:

1. קלט מספר שלם דו-ספרתי  $num$ -2
2. שלב אגרוגה המואט של המספר והשם  $absNum$ -2
3. שלב אגרוגה האוקה של  $absNum$  ב-10 והשם  $tens$ -2
4. שלב אגרוגה האוקה של  $absNum$  ב-10 והשם  $units$ -2
5. שלב אגרוגה הסכום של  $tens$  ושל  $units$  והשם  $sum$ -2
6. הצג כקלט אגרוגה של  $sum$

## יישום האלגוריתם

כזכור, לצורך חישוב ערכו המוחלט של מספר, נוכל להשתמש בפעולה לחישוב הערך המוחלט ששייכת למחלקה המתמטית, ומתוארת בטבלת הפעולות. המשפט המתאים ליישום הוראה 2 באלגוריתם הוא:

```
absNum = Math.Abs(num);
```

## התוכנית המלאה

```
/*
קלט: מספר דו-ספרתי שלם חיובי או שלילי
פלט: סכום ספרות המספר
*/
using System;
public class DigitSum
{
    public static void Main()
    {
        // הגדרת משתנים
        int num; // המספר הניתן כקלט
        int absNum; // ערכו המוחלט של המספר
        int tens; // ספרת העשרות
        int units; // ספרת האחדות
        int sum; // סכום הספרות
        // קלט
        1. Console.Write("Enter a two digit number: ");
        2. num = int.Parse(Console.ReadLine());
        // קבלת ערכו המוחלט של המספר
        3. absNum = Math.Abs(num);
        // פירוק המספר לספרותיו
        4. tens = absNum / 10;
        5. units = absNum % 10;
        // סכום ספרות המספר
        6. sum = tens + units;
        // פלט
        7. Console.WriteLine("The sum of the digits is: {0}", sum);
    } // Main
} // class DigitSum
```

## מעקב

נעקוב אחר מהלך ביצוע משפטי התוכנית עבור הקלט 17-:

שורה לביצוע	num	absNum	tens	units	sum	פלט
1	?	?	?	?	?	Enter a two digit number:
2	-17	?	?	?	?	
3	-17	17	?	?	?	
4	-17	17	1	?	?	
5	-17	17	1	7	?	
6	-17	17	1	7	8	
7	-17	17	1	7	8	The sum of the digits is: 8

על פי המעקב הזה התוכנית ביצעה את מטרותיה.

אך יש עוד לבדוק את התוכנית עבור **דוגמאות קלט מגוונות** (כלומר, בעלות מאפיינים שונים אשר מבטאים את מגוון הקלטים האפשריים) כדי להשתכנע שאכן היא מבצעת את מטרותיה עבור כל קלט אפשרי.

בטבלה זו בדקנו דוגמה לקלט שלילי. כדאי לבדוק אם התוכנית מבצעת את מטרותיה גם עבור קלט חיובי.

### שאלה 4.18

בנו טבלת מעקב אחר ביצוע משפטי התוכנית DigitSum עבור הקלט 53.

סוף פתרון בציה 3

להעמקה בתבנית **פירוק מספר חיובי לספרותיו** פנו לסעיף התבניות המופיע בסוף הפרק.

### שאלה 4.19

שנו את התוכנית DigitSum (התוכנית מפתרון בעיה 3) כך שהקלט יהיה המספר שהתקבל והפלט יהיה המספר בסדר ספרות הפוך. למשל עבור הקלט 43 הפלט הנדרש הוא 34.

**הדרכה:** בדיכס שני משתנים המבטאים את ספרת האחדות (units) ואת ספרת העשרות (tens) של המספר. חשבו על ביטוי חשבוני אשר משתמש בשני ערכים אלה ונותן את המספר הנדרש.

להעמקה בתבנית **בניית מספר** פנו לסעיף התבניות המופיע בסוף הפרק.

### שאלה 4.20

שנו את התוכנית DigitSum (התוכנית מפתרון בעיה 3) כך שהפלט יהיה **המרחק** בין ספרות המספר. למשל עבור הקלט 41, הפלט הנדרש הוא 3.

**הדרכה:** היעזרו בפעולה לחישוב ערך מוחלט.

## 4.3 הטיפוס התווי

עד עתה עסקנו רק במספרים מטיפוס שלם או מטיפוס ממשי. בפרק זה נלמד כי עיבודים ניתן לבצע לא רק על מספרים, אלא גם על תווים. תו הוא כל סימן אשר ניתן להציג על מסך המחשב.

כגון: אותיות ('a', 'b', 'A', ...) , סימנים מתמטיים ('>', '+', '\*', '/', '%', ...) , ספרות ('0', '1', '2', ...) , סימני תחביר ('!', ',', ':', ...) , רווח (' '), וכן הלאה.

לצורך הטיפול הפנימי של המחשב בתווים מותאם לכל תו מספר שלם, על פי קוד סטנדרטי, השומר על העקרון: לתווים עוקבים מותאם ערך מספרי עוקב. למשל אם לתו 'b' מותאם מספר כלשהו, לתו 'c' מותאם המספר העוקב לו. למעשה, כל התווים מסודרים בתת-קבוצות המכילות תווים עוקבים: האותיות האנגליות הגדולות (A..Z) מסודרות בתת קבוצה אחת והאותיות הקטנות (a..z) מסודרות בתת קבוצה אחרת, אותיות עבריות בתת-קבוצה נפרדת, וכך גם הספרות.

בהמשך לימודינו נלמד כיצד ניתן לעבד מילים ואף משפטים שלמים, אך בסעיף זה נתמקד כאמור בתווים בודדים.

**ערך מטיפוס תווי מצוין בשפת C# בין גרשיים בודדים. למשל 'Z'.**

## הצ'יה 4

**מטרת הבעיה הבאה ופתרונה:** הצגת הטיפוס התווי

פתחו אלגוריתם אשר הקלט שלו הוא אות אנגלית גדולה והפלט שלו הוא האות האנגלית הקטנה המתאימה. ישמו את האלגוריתם בתוכנית מחשב בשפת C#.

## ניתוח הבעיה בעזרת דוגמאות

הדוגמאות הבאות מבהירות את הנדרש בבעיה 4:

קלט	פלט
'A'	'a'
'D'	'd'
'V'	'v'

## פירוק הבעיה לתת-משימות

- קליטת אות אנגלית גדולה
- חישוב האות האנגלית הקטנה המתאימה
- הצגת האות האנגלית הקטנה המתאימה כפלט

**?** כיצד נחשב את האות האנגלית הקטנה המתאימה?

אנו יודעים כי לכל תו מתאים מספר, ולתווים עוקבים מתאימים מספרים עוקבים. הנה דוגמה אפשרית למספור כזה:

'a' – 97	'A' – 65
'b' – 98	'B' – 66
'c' – 99	'C' – 67
'd' – 100	'D' – 68
...	...

מכיוון שמספור האותיות הגדולות מתחילות ממספר מסוים, וממשיך משם בסדר עולה רציף, וכך גם לגבי האותיות הקטנות, הרי **ההפרש** בין המספר המייצג אות גדולה למספר המייצג אות קטנה

המתאימה לה הוא קבוע. כלומר, אם למשל, ההפרש בין 'a' ל-'A' הוא 32 (כפי שמתקיים בדוגמה שלעיל כי  $32 = 65 - 97$ ), זהו גם ההפרש בין 'b' ל-'B' ( $32 = 66 - 98$ ), בין 'c' ל-'C', וכן הלאה.

אם כך, כל שיש לעשות הוא להוסיף למספר המייצג את האות האנגלית הגדולה שקלטנו את ההפרש הקבוע שבין אות קטנה והאות הגדולה המתאימה לה. את ההפרש הקבוע הזה נוכל לחשב עבור זוג אותיות כלשהו למשל עבור האותיות 'a' ו-'A'.

הנה הפירוק של תת-משימה 2:

2.1. חישוב ההפרש הקבוע בין אות קטנה לאות הגדולה המתאימה לה, על ידי

הפחתת המספר המייצג את 'A' מהמספר המייצג את 'a'.

2.2. הוספת ההפרש שחושב למספר המייצג את האות הגדולה שנקלטה

## בחירת משתנים

נשתמש במשתנים הבאים:

**capitalLetter** – מטיפוס תווי, ישמור את התו הניתן כקלט, כלומר את האות הגדולה

**smallLetter** – מטיפוס תווי, ישמור את תו הפלט, כלומר את האות הקטנה המתאימה

## האלגוריתם

הנה האלגוריתם המתאים לפירוק לתת-משימות ולמשתנים שנבחרו:

- קלט את גדולה ב-capitalLetter
- אם ההפרש בין המספר המייצג את 'a' למספר המייצג את 'A' הוא 0, אז  
ההפרש שחושב ל-capitalLetter והשם את הגדולה ב-smallLetter
- הצג כקלט את ערך smallLetter

## יישום האלגוריתם

האלגוריתם משתמש במשתנים מטיפוס תווי, ולכן צריך כמובן להצהיר עליהם.

**משתנה מטיפוס תווי** מוצהר באמצעות המילה **char** (קיצור של המילה **character**, ומשמעותה תו באנגלית).

למשל:

```
char capitalLetter;
```

יישום הוראה 1 באלגוריתם צריך להיות משפט קלט, הקולט תו. פעולת הקלט של ערכים מטיפוס תווים היא הפעולה **char.Parse**, והשימוש בה דומה לשימוש בפעולות קלט אחרות שראינו. לכן, היישום של הוראה 1 יהיה:

```
Console.WriteLine("Enter a capital letter: ");  
capitalLetter = char.Parse(Console.ReadLine());
```

כיצד נבצע את החישובים הנדרשים בהוראה 2?

בשפת **C#** תו מזוהה עם הערך המספרי המתאים לו. לכן ניתן לשלב תווים בביטויים חשבוניים, והפעולות שכלולות בביטוי יפעלו על הערכים המספריים המתאימים לתווים. לכן, הביטוי 'A'-'a' נותן לנו את ההפרש שאנו זקוקים לו. אם כך, הביטוי הבא מבטא את החישובים בהוראה 2:

```
capitalLetter + 'a' - 'A';
```

אבל מאחר שבשפת C# לכל תו יש ערך מספרי, הרי כאשר אנו מבצעים פעולות חשבוניות על ערכים מטיפוס תווי, התוצאה היא מטיפוס שלם. ליישום הוראה 2, נרצה להשים את ערך הביטוי בתוך משתנה מטיפוס תווי, `smallLetter`, ולכן עלינו להשתמש בהמרה (`casting`), הפעם כדי להמיר את ערך הביטוי מטיפוס שלם לטיפוס תווי.

נשתמש בפעולת ההמרה באותו האופן שהשתמשנו בה בסעיף הקודם. כאן נמיר ערך ביטוי מטיפוס שלם להיות ערך מטיפוס תווי. שימו לב שההמרה צריכה לחול על כל הביטוי, ולכן נעטוף אותו בסוגריים.

אם כך, היישום ב-C# של הוראה 2 הוא:

```
smallLetter = (char)(capitalLetter + 'a' - 'A');
```

**שימו ♥:** הוראת ההמרה לטיפוס תווי מבוטאת על ידי זוג הסוגריים השמאלי. זוג הסוגריים הימני מגדיר את התחום שמופעלת עליו ההמרה.

**שימו ♥:** בפתרון זה אנו מתייחסים הן למשתנים מטיפוס תווי (`smallLetter`), `capitalLetter`) והן לערכים תוויים, כלומר ערכים קבועים מטיפוס תווי ('A', 'a'). הערכים התוויים מוקפים בגרשיים. ההתייחסות למשתנים היא דרך שמותיהם, בדיוק כפי שנעשה עבור משתנים מטיפוס מספרי.

פלט של תווים נעשה ב-C# בעזרת הוראות הפלט המוכרות לנו.

ננישם את האלגוריתם תוך שימוש בהוראות קלט ופלט עבור תווים:

```
1. Console.WriteLine("Enter capital letter: ");
2. capitalLetter = char.Parse(Console.ReadLine());
3. smallLetter = (char)(capitalLetter + 'a' - 'A');
4. Console.WriteLine("The corresponding small letter is: {0}",
    smallLetter);
```

**שימו ♥:** הוראות אלו אינן מסתמכות על ידיעת ערך ההפרש עצמו, או על ידיעה מפורשת של הערך המספרי המותאם לכל תו. בכך אנו מסייעים לתוכנית להיות עמידה יותר: גם אם גרסאות מאוחרות יותר של C# ישתמשו בקידוד אחר לתווים, הרי כל עוד יישמר העיקרון שתווים עוקבים מתאימים למספרים עוקבים, התוכנית תישאר נכונה.

## התוכנית המלאה

```
/* התוכנית קולטת אות אנגלית גדולה
   ומציגה כפלט את האות האנגלית הקטנה המתאימה לה */
using System;
public class CapitalLetterToSmall
{
    public static void Main()
    {
        // הגדרת משתנים
        char capitalLetter; // תו הקלט - אות אנגלית גדולה
        char smallLetter;   // תו הפלט - אות אנגלית קטנה
        // קלט
        Console.WriteLine("Enter a capital letter: ");
        capitalLetter = char.Parse(Console.ReadLine());
        // חישוב האות הקטנה
        smallLetter = (char)(capitalLetter + 'a' - 'A');
        // פלט
```

```

        Console.WriteLine("The corresponding small letter is: {0}",
            smallLetter);
    } // Main
} // class CapitalLetterToSmall

```

#### סוף פתרון קציה 4

#### שאלה 4.21

שנו את התוכנית `CapitalLetterToSmall` כך שתקבל כקלט אות אנגלית קטנה, ותציג כפלט את האות האנגלית הגדולה המתאימה לה.  
**רמז:** שימו לב לסדר פעולות חשבון! בדקו עצמכם בזהירות!

#### שאלה 4.22

פתחו אלגוריתם הקולט תו ומציג כפלט את התו הבא אחריו בסדר הקידוד המספרי. ישמו את האלגוריתם כתוכנית בשפת `C#`.  
**שימו ♥:** מה התו אחרי התו 'a'? מה התו אחרי התו 'S'? מה התו אחרי התו 'z'?  
 בחנו דוגמאות קלט מגוונות ככל שניתן.

### המרה מתו המייצג ספרה לערך מספרי מתאים

כפי שנאמר בתחילת הפרק, ערכי הטיפוס התווי כוללים כל סימן אשר ניתן להציג על מסך המחשב. בין סימנים אלה נכללות גם הספרות. למשל ערך מטיפוס תו יכול להיות '4' או '7'. לעתים, בהינתן תו המייצג ספרה, נרצה לדעת את ערכה המספרי של הספרה (כלומר להתאים לתו '4' את הערך השלם 4).

הנה דוגמה אפשרית למספור הפנימי עבור התווים המייצגים ספרות:

'0'	– 48
'1'	– 49
'2'	– 50
'3'	– 51
...	

התווים המייצגים ספרות ('0', '1', '2' וכו') הם תווים עוקבים, ולכן גם המספור שלהם עוקב.  
**?** כיצד ניתן לנצל את העובדה שמספור התווים המייצגים ספרות הוא עוקב כדי לחשב, עבור תו המייצג ספרה, את ערכה המספרי של הספרה?

נשים לב כי התו '1' מרוחק תו אחד מהתו '0'. בדומה, התו '2' מרוחק שני תווים מהתו '0', התו '3' מרוחק שלושה תווים מהתו '0', וכך הלאה. לכן נוכל להשתמש בפעולת חיסור כדי לחשב את הערך הדרוש. למשל אם נפחית מהמספר המתאים לתו '3' את המספר המתאים לתו '0', נקבל בדיוק 3 ( $51 - 48 = 3$ ).

גם הפעם אין אנו משתמשים באופן ישיר בערכי הקידוד עצמם, אלא רק מסתמכים על כך שהקידוד נותן ערכים עוקבים לספרות עוקבות.

לכן, אם במשתנה `charDigit` (מטיפוס תווי) שמור תו המייצג ספרה, אז לאחר החישוב שלהלן יכיל המשתנה `digit`, מטיפוס שלם, את ערכה המספרי של הספרה המיוצגת על ידי התו שב-`charDigit`.

```
digit = charDigit - '0';
```

#### שאלה 4.23

פתחו אלגוריתם המקבל כקלט תו המייצג ספרה. (יש עשרה תווים אפשריים לקלט זה: '0', '1', '2' ... '9'). פלט האלגוריתם יהיה מורכב משלושה מספרים: הספרה המיוצגת על ידי תו הקלט (שנקרא לה לשם קיצור ספרת הקלט), ספרת הקלט לאחר הכפלתה ב-10, וספרת הקלט לאחר הכפלתה ב-15. ישמו את האלגוריתם כתוכנית בשפת C#.

**הדרכה:** חשבו מהו המספר המתקבל אחרי הכפלת ספרת הקלט ב-10.

#### שאלה 4.24

פתחו אלגוריתם המקבל כקלט שני תווים, כל אחד מהם מייצג ספרה. פלט האלגוריתם יהיה תוצאת החיבור בין שני המספרים המתאימים. למשל, אם הקלט הוא '5' '9' הפלט יהיה 14. ישמו את האלגוריתם כתוכנית בשפת C#.

**שימו ♥:** כדי לקלוט מספר תווים מבלי להקיש Enter לאחר הקלדת כל תו, נוכל להשתמש בהוראת הקלט Read באופן הבא:

```
ch1 = (char) Console.Read();  
ch2 = (char) Console.Read();
```

בשפת C# קיימות מחלקות מוכנות המיועדות לשימוש המתכנת. על אחת מהן למדנו בתחילת פרק זה, המחלקה Math ובה פעולות מתמטיות. בסעיף הבא נלמד על מחלקה נוספת בשם Random שיכולה לשמש אותנו ליצירת מספרים אקראיים.

## 4.4 בחירה אקראית

הגרלת מספרים אקראיים היא פעולה שימושית מאוד במדעי המחשב. למשל, בתחום בדיקות התוכנה, השימוש במספרים אקראיים מאפשר בדיקת תוכנה על אוסף קלטים גדול ואקראי שנוצר אוטומטית. גם בתחום ההצפנה מספרים אקראיים הם שימושיים מאוד. בסעיף זה נכיר מחלקה המוגדרת בשפת C# ומאפשרת עבודה נוחה עם מספרים אקראיים.

### כיצד מגרילים מספר שלם ?

כדי ליישם הוראת הגרלה בשפת C# נזדקק למחלקת **מספרים אקראיים**: Random. בעזרת עצם מהמחלקה Random ניתן לבצע הגרלות של ערכים בכל תחום שנבחר. כדי שנוכל להשתמש בעצם מהמחלקה עלינו להצהיר עליו וליצור אותו באופן הבא:

```
Random rnd = new Random();
```

הוראה זו מצהירה על עצם מהמחלקה Random, בשם rnd, יוצרת אותו ומקצה לו מקום בזיכרון. כעת, כשיש בידינו עצם מהמחלקה Random אנו יכולים להגריל בעזרתו מספר שלם על ידי הפעלת הפעולה Next של העצם. השימוש בפעולה זו דומה לשימוש בפעולות של המחלקה Math שנלמדו בפרק זה. פעולה זו מקבלת בסוגריים פרמטר יחיד, שהוא ערך שלם המתאר את טווח המספרים להגרלה. אם נכתוב Next(n) יוגרל מספר שלם בתחום שבין 0 ל-(n-1). מאחר שזו פעולה של העצם, יש להשתמש בסימון הנקודה כדי להפעילה. למשל בעקבות ביצוע ההוראות:

```
int num;  
num = rnd.Next(6);
```

יושם במשתנה num מספר אקראי כלשהו בין 0 ל-5.

לפעולה זו גרסה נוספת, המקבלת שני פרמטרים, ערכים שלמים, המתארים את גבולות ההגרלה: הפרמטר הראשון מבטא את הגבול התחתון של תחום ההגרלה והפרמטר השני מבטא את הגבול העליון, כך: ערך הביטוי  $\text{rnd.Next}(n, m)$  הוא מספר שלם אקראי בתחום שבין  $n$  ל- $(m-1)$ .  
**שימו ♥:** כאשר משתמשים בפעולות של המחלקה `Math` מפעילים אותן ישירות מהמחלקה `Math` (למשל: `Math.Sqrt(x)`), ואילו כאשר משתמשים בפעולה `Next` של `Random`, צריך ליצור תחילה עצם מסוג `Random` ועליו להפעיל את הפעולה `Next`. הבדל זה נובע מכך שהפעולות במחלקה `Math` מוגדרות כפעולות סטטיות השייכות למחלקה ולא לעצם. פרק 11 מרחיב בנושא זה.

## פצ'ה 5

**מטרת הבעיה הבאה ופתרונה:** הצגת מחלקת מספרים אקראיים ואופן השימוש בה.

פתחו אלגוריתם אשר ידמה הטלה של 2 קוביות, כלומר יגריל שני ערכים בתחום 1-6 ויציג אותם כפלט. ישמו את האלגוריתם בשפת התכנות `C#`.

בבעיה זו אנו מתבקשים לדמות הטלת שתי קוביות, כלומר להגריל שני ערכים בתחום 1-6.

### פירוק הבעיה לתת-משימות

נפרק את הבעיה באופן הבא:

1. הגרלת 2 מספרים בין 1 ל-6.
2. הצגה של המספרים שהוגרלו כפלט.

### בחירת משתנים

- `die1` – מספר שלם שערכו בין 1-6 ובו המספר הראשון שיוגרל.
- `die2` – מספר שלם שערכו בין 1-6 ובו המספר השני שיוגרל.

### האלגוריתם

1. הגריל מספר בין 1 ל-6 והשם `die1`.
2. הגריל מספר בין 1 ל-6 והשם `die2`.
3. הצג כפלט: "ערך שגו הקוביות הוא " `die1`, `die2`.

### יישום האלגוריתם

אם נעביר לפעולה `Next` את הערך 6, נקבל מספר אקראי בין 0 ל-5. אם נעביר לפעולה `Next` את הערך 7, נקבל מספר אקראי בין 0 ל-6. אם כך:

**?** כיצד נדמה הטלת קובייה? כלומר כיצד נגריל מספר בין 1 ל-6?

נעביר לפעולה `Next` את הערך 6, על מנת לקבל מספר אקראי בין 0 ל-5. כדי "להמיר" את הערך שהתקבל לערך אפשרי עבור הטלת קובייה, נוסיף לו את הערך 1. כך יהיה בידינו מספר אקראי בין 1 ל-6. אפשרות נוספת היא להעביר לפעולה `Next` את הערכים: 1 ו-7 כך שיוגרל מספר בין 1 ל-6.

בתוכנית זו עלינו לבצע שתי הגרלות, ואכן לאחר שיצרנו עצם מהמחלקה `Random`, אנו יכולים להשתמש בו שוב ושוב להגרלות נוספות (אפילו מתוך תחומים שונים).



המחלקה Random שייכת למרחב השמות System לכן ההכרזה על השימוש ב-System מאפשרת לתוכנית להשתמש ישירות במחלקה Random.

## התוכנית המלאה

```
/*
התוכנית מדמה שתי הטלות קובייה ומדפיסה את ערכן
*/
using System;
public class TwoDice
{
    public static void Main ()
    {
        // הגדרת משתנים
        int die1;          // תוצאת הטלת קובייה אחת
        int die2;          // תוצאת הטלת קובייה שנייה
        Random rnd = new Random(); // Random מסוג
        die1 = rnd.Next(1,7); // הטלת קובייה ראשונה
        die2 = rnd.Next(1,7); // הטלת קובייה שנייה
        // הדפסות
        Console.WriteLine("The first one {0}: ", die1);
        Console.WriteLine("The second one {0}: ", die2);
    } // Main
} // class TwoDice
```

## סוף פתרון בעיה 5

נסכם את המושגים החדשים שהוצגו בפתרון בעיה 5:

בשפת C# מוגדרת מחלקה למספרים אקראיים הנקראת Random. באמצעות עצם מהמחלקה ניתן להגדיל מספרים אקראיים בתחום מבוקש.

לפני ביצוע הגרלות יש להצהיר על עצם מהמחלקה Random וליצור אותו, למשל כך:

```
Random rnd = new Random();
```

הפעולה Next של עצם מהמחלקה Random מקבלת כפרמטר מספר שלם חיובי n ומחזירה ערך אקראי שלם בתחום 0 עד n-1, או מקבלת כפרמטרים שני מספרים שלמים חיוביים m, n, ומחזירה איבר אקראי בתחום n עד m-1. מאחר שזו פעולה של עצם, יש להפעילה באמצעות סימון הנקודה, למשל כך:

```
rnd.Next(101)
```

```
rnd.Next(1,7)
```

באמצעות עצם מהמחלקה Random, ניתן לבצע הגרלות חוזרות ונשנות, גם מתחומים שונים.

## שאלה 4.25

בהינתן העצם rnd מהמחלקה Random,

א. מהו טווח הערכים האפשריים למשתנה השלם num בעקבות ביצוע ההוראה הבאה?

```
num = rnd.Next(6) + 1;
```

ב. מהו טווח הערכים האפשריים שיוצגו על המסך בעקבות ביצוע ההוראה הבאה?  
`Console.WriteLine(rnd.Next(10));`

ג. מהו טווח הערכים האפשריים שיוצגו על המסך בעקבות ביצוע ההוראה הבאה?  
`Console.WriteLine(rnd.Next(11,111));`

#### שאלה 4.26

בהינתן העצם `rndNum` מהמחלקה `Random`, מהו טווח הערכים האפשריים עבור כל אחד מהביטויים הבאים?

- א. `rndNum.Next(100)`
- ב. `rndNum.Next(1,100)`
- ג. `rndNum.Next(101)`
- ד. `rndNum.Next(101,201)`

#### שאלה 4.27

בכל אחד מהסעיפים הבאים תארו ביטוי, המתייחס לעצם `rndNum` מהמחלקה `Random`, כך שערך הביטוי הוא בתחום המבוקש.

- א. מספר אקראי שלם בין 0 ל-9
- ב. מספר אקראי שלם בין 10 ל-100
- ג. מספר אקראי שלם בין 100 ל-500
- ד. מספר אקראי שלם וזוגי בין 0 ל-100
- ה. מספר אקראי שלם ותלת-ספרתי

#### שאלה 4.28

א. נסו לחשוב על נוסחה כללית כיצד נגריל מספר בטווח הערכים בין שני ערכים שלמים כלשהם  $x$  ו- $y$ , כאשר ידוע ש- $x$  הוא הערך הקטן מבין השניים.

ב. כתבו קטע תוכנית בשפת `C#` שיקלוט שני מספרים שלמים  $x$  ו- $y$  (כאשר  $x < y$ ), ויגריל מספר בתחום שבין  $x$  ל- $y$ . השתמשו בנוסחה שכתבתם בסעיף א.

## סיכום

### פיתוח ויישום אלגוריתמים

הדגמנו בפרק זה את שלבי התהליך של פיתוח ויישום אלגוריתם, באמצעות בעיות ברמות שונות של קושי ושל מורכבות. למרות הבדלי הרמות, הקפדנו על שלבי התהליך בפתרון כל הבעיות: פיתוח ראשוני של הבעיה תוך בחינת הפלט עבור דוגמאות קלט מגוונות, ניסוח רעיון לפתרון על ידי פירוק לתת-משימות, בחירת משתנים, כתיבת אלגוריתם, יישום האלגוריתם במשפטי תוכנית, בדיקת מהלך ביצוע התוכנית באמצעות טבלת מעקב וכתיבת התוכנית המלאה והרצתה.

גם בשאר פרקי הספר, בעת פתרון בעיות אלגוריתמיות, נשתדל להקפיד על פיתוח אלגוריתם בשלבים ועל יישומו על פי שלבים אלה.

ייתכן כי בפתרון בעיות בהמשך, נאחד לפעמים חלק מן השלבים. נעשה זאת רק לאחר רכישת מיומנות מספקת בפתרון מפורט בשלבים. בכל מקרה נקפיד תמיד על ביצוע שלבי הבדיקה – הן הבדיקה ה"ידנית" והן הבדיקה באמצעות ההרצה.

## פעולות חלוקה בשלמים

על ערכים מטיפוס שלם מוגדרות שתי פעולות: **מנת חלוקה ושארית חלוקה** (modulus):  
**מנת החלוקה** של מספר שלם  $x$  במספר שלם  $y$  שווה לחלק השלם של  $x/y$ , ובמילים אחרות, למספר הפעמים שהערך  $y$  נכנס בערך  $x$ .

**שארית החלוקה** של מספר שלם  $x$  במספר שלם  $y$  מבטאת את השארית הנותרת לאחר חלוקה בשלמים של  $x$  ב- $y$ , ובמילים אחרות, את מה שנותר אחרי שמפחיתים מ- $x$  כפולות שלמות של  $y$  ככל שניתן.

**תוצאת שארית החלוקה** של מספר שלם כלשהו ב- $n$ , יכולה להיות כל מספר שלם בתחום בין 0 ל- $(n-1)$ .

פעולות אלו מוגדרות עבור ערכים שלמים בלבד!

ניתן להיעזר בפעולות חלוקה בשלמים כדי לבצע **פירוק מספר דו-ספרתי לספרותיו**.

## הטיפוס התווי

לכל תו מותאם ערך מספרי השמור לו.

לתווים עוקבים יש ערכים מספריים עוקבים.

## סיכום מרכיבי שפת C# שנלמדו בפרק 4

### פעולות חלוקה בשלמים

פעולת **מנת החלוקה** בשלמים מסומנת בשפת C# בסימן  $/$ .

כאשר הפעולה / מופעלת על שני ערכים שלמים היא מפורשת בשפת C# כפעולת חלוקה בשלמים.

בכל מקרה אחר (שלם וממשי, ממשי ושלם, ממשי וממשי) הפעולה / מפורשת בשפת C# כפעולת חלוקה על ערכים ממשיים.

פעולת **שארית החלוקה** בשלמים מסומנת בשפת C# בסימן  $\%$ .

כדי לבצע פעולת חלוקה ממשיית בין שני ערכים שלמים יש לבצע **המרה** (casting) של המחלק או של המחולק לערך ממשי.

המרה של ערך שלם  $x$  לערך ממשי נעשית בשפת C# כך:  $(double) x$ .

לפעולת ההמרה אין השפעה על טיפוס המשתנה המעורב בביטוי המומר. היא רק מנחה להתייחס אל ערכו כאילו היה מטיפוס ממשי, באופן זמני, רק לצורך החישוב הנוכחי.

## הטיפוס התווי

ערך מטיפוס תווי מצוין ב-C# בין **גרשיים** בודדים, למשל כך: 'a'.

**הצהרה** על משתנה מטיפוס תווי נעשית באמצעות המילה `char`.

ניתן לשלב תווים בביטויים חשבוניים. בזמן חישוב הביטוי נלקחים הערכים המספריים המותאמים לכל תו ותו.

הטיפוס של ביטוי חשבוני הכולל תווים הוא מספרי. במידת הצורך, ניתן להמיר את ערכו של ביטוי כזה לערך מטיפוס תווי באמצעות פעולת ההמרה, למשל: `(char) ('a' + 3)`.

## בחירה אקראית

בשפת C# ניתן להגדיל מספר אקראי באמצעות המחלקה `Random`.

כדי להשתמש במחלקה `Random` בתוך תוכנית C# צריך להכריז על שימוש במרחב השמות `:System`

```
using System;
```

**לפני ביצוע הגרלות יש להצהיר על עצם מהמחלקה וליצור אותו.** פעולת היצירה `new` מקצה לו מקום בזיכרון:

```
Random העצם r = new Random();
```

**הגרלת מספר אקראי שלם בתחום שבין 0 ל-n-1 מתבצעת באמצעות הפעלת הפעולה `Next(n)` של עצם מהמחלקה `Random`.** הגרלת מספר שלם בתחום שבין n ל-m-1 מתבצעת על ידי הפעלת הפעולה `Next(n,m)` של עצם מהמחלקה `Random`. הפעולה מופעלת באמצעות סימון הנקודה ומחזירה ערך אקראי שלם בתחום המבוקש.

באמצעות עצם מהמחלקה `Random`, שהוקצה עבורו מקום בזיכרון, ניתן לבצע הגרלות חוזרות ונשנות, גם מתחומים שונים.

## שאלות נוספות

### שאלות נוספות לסעיף 4.1

1. פתחו אלגוריתם אשר הקלט שלו הוא מקדמים של משוואה ריבועית:  $a, b$  ו-  $c$  והפלט הוא שני הפתרונות האפשריים של המשוואה הריבועית. הניחו כי הקלט תקין ולמשוואה הריבועית אכן קיימים שני פתרונות. ישמו את האלגוריתם בשפת C#. להזכירכם, הנה הנוסחה לחישוב פתרונותיה של משוואה ריבועית:

$$X_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

### שאלות נוספות לסעיף 4.2

1. פתחו אלגוריתם אשר הקלט שלו הוא פרק זמן הנתון בימים והפלט שלו הוא מספר השבועות השלמים הכלולים בפרק הזמן הנתון. לדוגמה עבור הקלט 18 הפלט הוא 2. ישמו את האלגוריתם כתוכנית בשפת C#.

2. נהגי מוניות שירות יוצאים לדרכם רק כאשר כל המושבים במונית תפוסים. פתחו אלגוריתם אשר הקלט שלו הוא מספר הנוסעים הממתנים למונית בתחנה ומספר המושבים במונית, והפלט שלו הוא מספר המוניות שניתן למלא במלואן ומספר הנוסעים שייותרו בתחנה (הניחו כי לכל המוניות בתחנה יש מספר מושבים זהה). למשל:  
עבור הקלט 7 25 יהיה הפלט: ניתן למלא 3 מוניות, 4 נוסעים יותרו בתחנה.  
עבור הקלט 7 35 יהיה הפלט: ניתן למלא 5 מוניות, 0 נוסעים יותרו בתחנה.

3. נתון קטע התוכנית הבא: (num1 ו-num2 הם משתנים מטיפוס שלם)

```
a = num1 / 2;  
b = num1 % 2;  
c = num2 / 10;  
d = num2 % 10;  
e = num1 / num2;  
f = num1 % num2;
```

- א. ציינו ערך התחלתי של num1 שעבורו ערכיהם הסופיים של a ושל b יהיו 3 ו-0 בהתאמה.  
ב. ציינו ערך התחלתי של num2 שעבורו ערכיהם הסופיים של c ושל d יהיו 6 ו-3 בהתאמה.  
ג. תנו שתי דוגמאות לערכים התחלתיים של num1 ושל num2 שעבורם ערכיהם הסופיים של e ושל f יהיו 3 ו-2, בהתאמה.

## שאלות מסכמות לפרק 4

1. במכולת של חנניה מוכרים מסטיקים בקבוצות על פי גודל האריזות הקיימות בחנות, והמסטיקים הנותרים נמכרים בודדים. שווי כל מסטיק הוא 0.2 ₪.  
יש לפתח אלגוריתם שהקלט שלו הוא מספר המסטיקים הנמצא במכולת וגודל האריזות הקיימות בחנות (כל האריזות באותו הגודל). הפלט של האלגוריתם הוא השווי הכולל של המסטיקים בחבילות השלמות ושווי המסטיקים שנותרים לא ארוזים.  
למשל, עבור הקלט 7 100 הפלט הדרוש הוא 0.4 19.6.  
בחנו את הפלט עבור דוגמאות קלט מייצגות, והציגו את חלוקת המשימה המתוארת לתת-משימות.
2. חנניה מהמכולת מעוניין באלגוריתם אשר יעזור לו להחזיר עודף במטבעות בצורה היעילה ביותר, כלומר במספר המטבעות הקטן ביותר.  
בהנחה כי חנניה יכול להחזיר עודף אך ורק במטבעות של 1 ₪, של 5 ₪ ושל 10 ₪, כתבו אלגוריתם המקבל כקלט את הסכום שחנניה צריך להחזיר כעודף (מספר שלם), ומציג כפלט:  
א. את מספר המטבעות שיחזיר חנניה מכל סוג.  
ב. את מספר המטבעות הכולל שיחזיר חנניה.  
למשל עבור הקלט 18 יתקבל הפלט:  
א. 1 ₪ : 3, 5 ₪ : 1, 10 ₪ : 1  
ב. 5 מטבעות.  
ישמו את האלגוריתם בשפת C#.
3. כתבו קטע תוכנית בשפת C# המגריל מספר תלת-ספרתי. התוכנית תדפיס את ספרות המספר כל אחת בשורה נפרדת. מאות עשרות ואחדות

## תבניות – פרק 4

פירוט מלא של התבניות ושל שאלות שבפתרון יש שימוש בתבניות ניתן למצוא באתר הספר ברשת האינטרנט.

### חלוקת כמות פריטים לקבוצות בגודל נתון

שם התבנית: מנת החלוקה לקבוצות של כמות פריטים
נקודת מוצא: שני מספרים שלמים חיוביים; quantity (כמות הפריטים) ו-num (מספר הפריטים בקבוצה)
מטרה: מספר הקבוצות המלאות מחלוקה של quantity הפריטים לקבוצות בגודל num
אלגוריתם:
השם groups-2 אל מנת האלוקה של quantity-2 num

שם התבנית: שארית החלוקה לקבוצות של כמות פריטים
נקודת מוצא: שני מספרים שלמים חיוביים; quantity (כמות הפריטים) ו-num (מספר הפריטים בקבוצה)
מטרה: מספר הפריטים העודף בחלוקה של quantity הפריטים לקבוצות בגודל num
אלגוריתם:
השם remainder-2 שארית האלוקה של quantity-2 num

### פירוק מספר חיובי לספרותיו

שם התבנית: ספרת האחדות של מספר
נקודת מוצא: מספר דו-ספרתי חיובי num
מטרה: חישוב ספרת האחדות של num
אלגוריתם:
השם units-2 שארית האלוקה של num-2 10

שם התבנית: ספרת העשרות של מספר
נקודת מוצא: מספר דו-ספרתי חיובי num
מטרה: חישוב ספרת העשרות של num
אלגוריתם:
השם tens-2 אל מנת האלוקה של num-2 10

## בניית מספר

שם התבנית: בניית מספר

נקודת מוצא: שתי ספרות left ו-right

מטרה: בניית מספר דו-ספרתי מהספרות הנתונות

אלגוריתם:

השם ב-num אג הערך של הביטוי  $left * 10 + right$

## תבניות – פרק 4

### חלוקת כמות פריטים לקבוצות בגודל נתון

נתבונן בבעיה הבאה :

במפעל זכוכית אורזים 8 כוסות בקופסת קרטון. מחיר קופסת קרטון הוא 1.5 ₪. נתון אלגוריתם חלקי שהקלט שלו הוא מספר הכוסות המיועדות לאריזה, והפלט שלו הוא : מספר הקופסאות המלאות (בכוסות) שניתן לארוז, המחיר הכולל של הקרטון הדרוש לאריזה ומספר הכוסות שנותרו בתפזורת. השלימו את האלגוריתם :

1. קאוט מספר כוסות המיועדות לאריזה כ-glasses
2. גשב אג \_\_\_\_\_ על ידי \_\_\_\_\_ והשם כ-boxes
3. גשב אג \_\_\_\_\_ על ידי \_\_\_\_\_ והשם כ-price
4. גשב אג \_\_\_\_\_ על ידי \_\_\_\_\_ והשם כ-left
5. הצג כפאוט אג הערך boxes, אג הערך price ואג הערך left

בבעיה זו יש לחשב את המספר המירבי של קופסאות מלאות על ידי חישוב מנת החלוקה של מספר הכוסות ב-8. את חישוב מספר הכוסות שנותרו בתפזורת נבצע באמצעות חישוב שארית החלוקה של מספר הכוסות המיועדות לאריזה ב-8. התבנית של חלוקת כמות פריטים לקבוצות בגודל נתון, באמצעות חישובי מנה ושארית, כפי שמשמשת בפתרון בעיה זו, דומה לחישובים שהוצגו בפתרון בעיה 2 בפרק הלימוד. תבנית זו היא בסיסית ביותר ושימושית בחישובים רבים של מספרים שלמים. התבנית משמשת הן כתבנית עיקרית בבעיות חישוב של מספרים שלמים והן כתבנית המשולבת בחישובים מורכבים שונים.

נתבונן בשני האלגוריתמים הללו :

1. קאוט מספר הגז/אג כ-amount	1. קאוט מספר כוסות המיועדות לאריזה כ-glasses
2. גשב אג מספר השקיות המלאות על ידי מנה האוקה של amount כ-20 והשם אג הגז/אג כ-bags	2. גשב אג מספר הקופסאות המלאות על ידי מנה האוקה של glasses כ-8 והשם אג הגז/אג כ-boxes
3. גשב אג מספר הגז/אג שנותרו כ-amount כ-20	3. גשב אג מנה הקופסאות על ידי 1.5 * boxes והשם כ-price
4. גשב אג שארית האוקה של amount כ-20 והשם אג הגז/אג כ-remainder	4. גשב אג מספר הכוסות שנותרו כ-amount כ-8 והשם left כ-2
5. הצג כפאוט אג הערך bags ואג הערך remainder	5. הצג כפאוט אג הערך boxes, אג הערך price ואג הערך left



אנו רואים כי בשני האלגוריתמים מתבצעת חלוקת כמות פריטים לקבוצות: בבעיה הראשונה כמות הפריטים היא מספר הכוסות המיועדות לאריזה וגודל כל קבוצה (כוסות בקופסת קרטון) הוא 8 ובבעיה השנייה כמות הפריטים היא מספר הגוגואים וגודל כל קבוצה (מספר גוגואים בשקית) הוא 20.

נפריד את מאפייני התבנית **חלוקת כמות פריטים לקבוצות בגודל נתון** לשתי תת-תבניות: ראשית, נציג את מאפייני התבנית **מנת החלוקה של כמות פריטים לקבוצות** ואחר כך נציג את מאפייני התבנית **שארית החלוקה של כמות פריטים לקבוצות**.

**שם התבנית:** מנת החלוקה לקבוצות של כמות פריטים

**נקודת מוצא:** שני מספרים שלמים חיוביים, quantity (כמות הפריטים) ו-num (מספר הפריטים בקבוצה)

**מטרה:** מספר הקבוצות המלאות מחלוקה של quantity הפריטים לקבוצות בגודל num

**אלגוריתם:**

השם groups אף לא האלוקה של quantity ב-num

**יישום ב-C#:**

```
groups = quantity / num;
```

**שם התבנית:** שארית החלוקה לקבוצות של כמות פריטים

**נקודת מוצא:** שני מספרים שלמים חיוביים, quantity (כמות הפריטים) ו-num (מספר הפריטים בקבוצה)

**מטרה:** מספר הפריטים העודף בחלוקה של quantity הפריטים לקבוצות בגודל num

**אלגוריתם:**

השם remainder אף לא שארית האלוקה של quantity ב-num

**יישום ב-C#:**

```
remainder = quantity % num;
```

## שאלה 1

ישמו את האלגוריתם לפתרון בעיית הכוסות בשפת C#.

## שאלה 2

א. מהם הערכים האפשריים עבור מספר הכוסות שעשויות להישאר בתפזורת לאחר החלוקה לקופסאות מלאות?

ב. אחד העובדים במפעל הזכוכית הציע הצעת ייעול לאריזת הכוסות, כדי להקטין את מספר הכוסות שיישארו בתפזורת: חלוקת מספר הכוסות הנותרות לקופסאות קרטון קטנות בגודל 2, שעלות כל אחת מהן היא 0.5 ₪.

1. מהם הערכים האפשריים עבור מספר הכוסות שעשויות להישאר בתפזורת לאחר החלוקה לשני סוגי הקופסאות?

2. הרחיבו את האלגוריתם על פי הצעת הייעול של העובד, כך שהפלט של האלגוריתם יהיה המחיר הכולל של הקרטון עבור שני סוגי הקופסאות ומספר הכוסות שיישארו בתפזורת לאחר החלוקה.

3. ישמו את האלגוריתם שכתבתם בסעיף ב בשפת C#.

---

## פירוק מספר חיובי לספרותיו

התבנית של הפרדת הספרות של מספר, ובפרט הספרה הימנית ביותר – ספרת האחדות, היא שימושית בהקשרים רבים במדעי המחשב, למשל, לצורך סיווג מספרים על פי ספרת האחדות שלהם.

לעת עתה נתמקד בתבנית של פירוק מספר דו-ספרתי חיובי לספרותיו, בהדרגה נרחיב עבור מספר תלת-ספרתי חיובי ובהמשך נראה תבנית כללית עבור מספר חיובי כלשהו. במספר דו-ספרתי ישנן שתי ספרות: הפרדת ספרת האחדות נעשית על ידי חישוב שארית החלוקה של המספר ב-10 והפרדת ספרת העשרות נעשית על ידי חישוב של מנת החלוקה של המספר ב-10.

נפריד את מאפייני התבנית **פירוק מספר חיובי לספרותיו** לשתי תת-תבניות: ראשית נציג את מאפייני התבנית **ספרת האחדות של מספר** ואחר כך נציג את מאפייני התבנית **ספרת העשרות של מספר**. כפי שצוין קודם, נקודת המוצא של התבנית היא מספר דו-ספרתי חיובי.

<b>שם התבנית:</b> ספרת האחדות של מספר
<b>נקודת מוצא:</b> מספר דו-ספרתי חיובי num
<b>מטרה:</b> חישוב ספרת האחדות של num
<b>אלגוריתם:</b>
השם units אחראי האלוקה של num ב-10
<b>יישום ב-C#:</b>
<pre>units = num % 10;</pre>

<b>שם התבנית:</b> ספרת העשרות של מספר
<b>נקודת מוצא:</b> מספר דו-ספרתי חיובי num
<b>מטרה:</b> חישוב ספרת העשרות של num
<b>אלגוריתם:</b>
השם tens אחראי האלוקה של num ב-10
<b>יישום ב-C#:</b>
<pre>tens = num / 10;</pre>

### שאלה 3

נתון מספר דו-ספרתי חיובי  $num$  ועליו הופעלו שתי התבניות :

1.  $num$  /הצג אותה כפלט

2.  $num$  /הצג אותה כפלט

א. תנו שתי דוגמאות שונות לערכים אפשריים ל- $num$  שעבורם הערך הראשון שיוצג כפלט הוא 5.

ב. תנו שתי דוגמאות שונות לערכים אפשריים ל- $num$  שעבורם הערך השני שיוצג כפלט הוא 8.

ג. רשמו את כל זוגות הערכים שעשויים להתקבל בפלט אם ידוע שהמספר  $num$  גדול מ-40,

מתחלק ב-5 ללא שארית ואינו זוגי.

ד. רשמו את כל הערכים האפשריים ל- $num$  כך שעבור סדרת ההוראות הבאה יוצג הערך 3 :

1.  $num$  /הצג אותה כפלט

2.  $num$  /הצג אותה כפלט

3.  $num$  /הצג אותה כפלט

ה. נתונה סדרת ההוראות הבאה :

1.  $num$  /הצג אותה כפלט

2.  $num$  /הצג אותה כפלט

3.  $num$  /הצג אותה כפלט

1. רשמו את כל הערכים האפשריים ל- $num$  עבורם יוצג כפלט הערך 1.

2. ישמו את ההוראה בשפת C#.

### שאלה 4

עינת הציעה ליישם את התבנית **פירוק מספר לספרותיו** עבור מספר דו-ספרתי חיובי  $num$

באמצעות התבנית **חלוקת כמות פריטים לקבוצות בגודל נתון**.

א. מהי כמות הפריטים? הסבירו.

ב. מהו גודל הקבוצות? הסבירו.

ג. השלימו את השימוש בתבניות על פי הצעתה של עינת :

1.  $num$  /הצג אותה כפלט

2.  $num$  /הצג אותה כפלט

### שאלה 5

המורה ביקשה מתלמידיה לכתוב אלגוריתם שהקלט שלו הוא מספר תלת-ספרתי  $num$  והפלט

שלו הוא שלוש ספרותיו של  $num$ , על פי הרעיון הבא, המתבסס על פירוק המספר לשני מספרים :

ספרת האחדות, ומנת החלוקה של  $num$  ב-10. לאחר מכן יש לבצע פירוק נוסף של המנה

שחושבה, כמספר דו-ספרתי.

א. השלימו את האלגוריתם, ונסו להשתמש לשם כך בתבניות.

1. קאוט מספר גא-ספרי ב-num
  2. גלב את ספרי האצול עז ידי \_\_\_\_\_ והלם ב-units
  3. גלב את המספר די-ספרי המתקבל מהספר ספרי האצול עז ידי \_\_\_\_\_ והלם ב-doubleDigit
  4. גלב את ספרי העשרות של num עז ידי \_\_\_\_\_ והלם ב-tens
  5. גלב את ספרי המאות של num עז ידי \_\_\_\_\_ והלם ב-hundreds
  6. הצג כפולט את הערך units, את הערך tens ואת הערך hundreds
- ב. ישמו את האלגוריתם בשפת C#.
- ג. אלון הציע דרך אחרת לפירוק המספר. לפניכם האלגוריתם:

1. קאוט מספר גא-ספרי ב-num
  2. גלב את ספרי המאות של num עז ידי גישוב מנת החלוקה של num פריטים ל-100 קבוצות והלם ב-hundreds
  3. גלב את המספר די-ספרי המתקבל מהספר ספרי המאות עז ידי גישוב שארית החלוקה של num פריטים ל-100 קבוצות והלם ב-doubleDigit
  4. גלב את ספרי העשרות של num עז ידי גישוב ספרת העשרות של doubleDigit והלם ב-tens
  5. גלב את ספרי האצול של num עז ידי גישוב ספרת האחדות של doubleDigit והלם ב-units
  6. הצג כפולט את הערך units, את הערך tens ואת הערך hundreds
- ד. הסבירו את הרעיון לפתרון עליו מתבסס אלון. בהסבר התייחסו לשימוש בתבנית **חלוקת כמות פריטים לקבוצות בגודל נתון**.
- ה. ישמו את האלגוריתם בשפת C#.

## שאלה 6

- בבניין משרדים בן 9 קומות מסומן כל חדר באמצעות קוד של מספר תלת-ספרתי: ספרת המאות מציינת את מספר הקומה בה נמצא החדר וספרות האחדות והעשרות מציינות את מספר החדר בקומה.
- א. כתבו אלגוריתם, שהקלט שלו הוא קוד חדר והפלט שלו הוא מספר הקומה בה נמצא החדר ומספר החדר בקומה.
- ב. ציינו באילו תבניות השתמשתם לפתרון הבעיה.
- ג. ישמו את האלגוריתם בשפת C#.

## בניית מספר

התבנית של בניית מספר היא שימושית בהקשרים רבים. אולם, הבנייה של מרכיב ממספר מרכיבים מצומצמים יותר שימושית גם בהקשרים אחרים במדעי המחשב, למשל: בניית מילה מאותיות בודדות, בניית משפט ממילים בודדות, בניית צורה גרפית מקווים בודדים וכדומה.

לעת עתה נתמקד בתבנית של הרכבת מספר דו-ספרתי משתי ספרות, בהדרגה נרחיב עבור בניית מספר תלת-ספרתי ובהמשך נראה תבנית כללית עבור מספר חיובי כלשהו.

כדי להרכיב מספר משתי ספרות נכפיל את הספרה המיועדת להיות ספרת העשרות ב-10 ונחבר לתוצאה את הספרה המיועדת להיות ספרת האחדות.

**שם התבנית:** בניית מספר

**נקודת מוצא:** שתי ספרות left ו-right

**מטרה:** בניית מספר דו-ספרתי מהספרות הנתונות

**אלגוריתם:**

השם num-2 את הערך של הביטוי  $left * 10 + right$

**יישום ב-C#:**

```
num = left * 10 + right;
```

**שימו ♥:** כאשר אנו משתמשים בתבנית **בניית מספר** חשוב להקפיד על תיאור חד-משמעי: נסכים כי הספרה הראשונה שנציין תהיה ספרת העשרות והשנייה ספרת האחדות. למשל, הכוונה בשימוש בנה מספר מ-1 ו-3 היא לבנות את המספר 13 (ולא את המספר 31). בדומה, בשימוש בתבנית לשלוש ספרות נציין קודם את ספרת המאות, אחר כך את ספרת העשרות ולבסוף את ספרת האחדות.

### שאלה 7

נתון אלגוריתם חלקי, שהקלט שלו הוא מספר דו-ספרתי חיובי num, והפלט שלו הוא מספר דו-ספרתי חדש, שערך ספרת האחדות בו גדולה ב-1 מספרת האחדות של המספר הנתון num, וערך ספרת העשרות בו קטנה ב-1 מספרת העשרות של המספר הנתון num. הניחו שבמספר הנקלט ספרת האחדות שונה מ-9 וספרת העשרות שונה מ-0. א. השלימו את האלגוריתם:

1. קלוט מספר דו-ספרתי ג'א'2- num

2. גשג את ספרת האגד'א'ל י'ד' \_\_\_\_\_ /השם 2-units

3. גשג את ספרת העשר'א'ל num י'ד' \_\_\_\_\_ /השם 2-tens

4.          וְשֵׁב אֶת עֵרִכָּה לֵאל סִפְרָה הָאֲדֻמָּה לֵאל הָאֲסֵפֶר הָגִדֵּשׁ עַל יָדֵי \_\_\_\_\_ וְהֵשֶׁם

newUnits-ב

5.          וְשֵׁב אֶת עֵרִכָּה לֵאל סִפְרָה הָעֲשָׂרִים לֵאל הָאֲסֵפֶר הָגִדֵּשׁ עַל יָדֵי \_\_\_\_\_ וְהֵשֶׁם

newTens-ב

6. בְּנֵה מִסְפָּר מ- \_\_\_\_\_ ו- \_\_\_\_\_ וְהֵצֵג כִּפְּלוֹט אֶת הָעֵרֶךְ שֶׁהִתְקַבֵּל

א. ציינו מהן התבניות שהשתמשתם בהן בהשלמת האלגוריתם.

ב. מהו הפלט עבור הקלט 67?

ג. מהו הקלט שהפלט עבורו הוא 45?

ד. מהו הערך הקטן ביותר שיוצג כפלט? עבור איזה ערך של num יוצג כפלט ערך זה?

ה. מהו הערך הגדול ביותר שיוצג כפלט? עבור איזה ערך של num יוצג כפלט ערך זה?

ו. ישמו את האלגוריתם בשפת C#.

ז. כתבו את האלגוריתם ללא שימוש בתבנית בניית מספר.

## שאלה 8

א. כתבו אלגוריתם, שהקלט שלו הוא מספר תלת-ספרתי num והפלט שלו הוא כל המספרים

הדו-ספרתיים שניתן להרכיב משלוש הספרות של num.

ב. ציינו את מספר הפעמים בהם השתמשתם בתבנית בניית מספר.

ג. עבור אילו ערכים של num יוצג כפלט ערך יחיד (החוזר על עצמו)?

ד. ישמו את האלגוריתם בשפת C#.

## שאלה 9

נתון אלגוריתם חלקי שהקלט שלו הוא 3 ספרות והפלט שלו הוא מספר תלת-ספרתי המורכב

משלוש הספרות:

1. קְלוֹט סִפְרָה מֵאֹת-ב hundreds, סִפְרָה עֲשָׂרִים-ב tens, סִפְרָה אֲדֻמָּה-ב units

2. בְּנֵה מִסְפָּר מ- hundreds tens-ו וְהֵשֶׁם ב- temp אֶת הָעֵרֶךְ שֶׁהִתְקַבֵּל

3. בְּנֵה מִסְפָּר מ- \_\_\_\_\_ ו- \_\_\_\_\_ וְהֵשֶׁם ב- num אֶת הָאֲסֵפֶר הָאֲדֻמָּה-סִפְרָה

שֶׁהִתְקַבֵּל

4. הֵצֵג כִּפְּלוֹט אֶת עֵרִכּוֹ לֵאל num

א. השלימו את האלגוריתם.

ב. הסבירו את תפקידו של המשתנה temp.

ג. ישמו את האלגוריתם בשפת C#.

ד. ילנה הציעה תבנית לאלגוריתם של בניית מספר משלוש ספרות ללא שימוש כפול בתבנית

בניית מספר עבור שתי ספרות. לפניכם האלגוריתם החלקי:

1. קאוט ספריג מאלג-ב hundreds, ספריג עשרות-ב tens, ספריג אגדלג-ב units

2. השם ב-num את \_\_\_\_\_ \*10 + \_\_\_\_\_ \*100 + \_\_\_\_\_

3. הצג כפוט אל ערכו של num

השלימו את האלגוריתם.

ה. נתון שימוש בתבנית :

בנה מספר מ-digit, digit ו-digit והצג כפוט אל הערך שהקבל

מה יוצג כפלט עבור הערך 8 ב-digit?

## שאלה 10

נתון אלגוריתם חלקי, שהקלט שלו הוא מספר ארבע-ספרתי חיובי num :

1. קאוט מספר ארבע-ספרתי ג'וי ב-num

2. גשג אל ספריג האגדלג ע' ידי \_\_\_\_\_ והשם ב-units

3. גשג אל ספריג העשרות של num ע' ידי \_\_\_\_\_ והשם ב-tens

4. גשג אל ספריג המאלג של num ע' ידי \_\_\_\_\_ והשם ב-hundreds

5. גשג אל ספריג האלפים של num ע' ידי ג'שג מנת החלוקה של \_\_\_\_\_ פריטים

ל- \_\_\_\_\_ קבוצות והשם ב-thousands

6. בנה מספר מ-tens ו-units והכפל אלג ב-100

7. בנה מספר מ-thousands ו-hundreds

8. הצג כפוט אל סכום המספרים ג'שג

א. השלימו את האלגוריתם.

ב. מהו הפלט עבור הקלט 5243?

ג. מהו הקלט עבור הפלט 1197?

ד. מהי מטרת האלגוריתם?

ה. ניתן לכתוב אלגוריתם המבצע אותה מטרה ללא שימוש בתבנית בניית מספר עבור שתי ספרות. לפניכם האלגוריתם החלקי :

1. קאוט מספר ארבע-ספרתי ג'וי ב-num

2. גשג אל שגי הספרות השמאליות ע' ידי \_\_\_\_\_ והשם ב-left

3. גשג אל שגי הספרות הימניות ע' ידי \_\_\_\_\_ והשם ב-right

4. הצג כפוט אל הערך של \_\_\_\_\_ \* 100 + \_\_\_\_\_

השלימו את האלגוריתם.



אחד השימושים של התבנית **בניית מספר** הוא לצורך היפוך ספרותיו של מספר נתון. נראה זאת בשתי השאלות הבאות :

---

### שאלה 11

נתון אלגוריתם חלקי, שהקלט שלו הוא מספר דו-ספרתי חיובי num, והפלט שלו הוא מספר דו-ספרתי שסדר ספרותיו הפוך מסדר הספרות במספר הנקלט num :  
א. השלימו את האלגוריתם :

1. קאוט מספר דו-ספרתי גיוכי ב-num
  2. גשב אג ספרג האגדאג עג יגזי \_\_\_\_\_ /הגסג ב-units
  3. גשב אג ספרג העגגאג גג num עג יגזי \_\_\_\_\_ /הגסג ב-tens
  4. בנה מספר מ- \_\_\_\_\_ -/ \_\_\_\_\_ /הגסג ב-reverseNum
  5. הגג כפאט אג עגכג גג reverseNum
- ב. כתבו את כל הערכים האפשריים ל-num עבורם הפלט יהיה זהה לקלט.
- ג. ישמו את האלגוריתם בשפת C#.

### שאלה 12

נתון אלגוריתם חלקי שהקלט שלו הוא מספר תלת-ספרתי חיובי num, והפלט שלו הוא מספר תלת-ספרתי המורכב משלוש הספרות של num אך בסדר הפוך.

1. קאוט מספר גאג-ספרג גיוכי ב-num
  2. גשב אג ספרג האגדאג עג יגזי \_\_\_\_\_ /הגסג ב-units
  3. גשב אג ספרג העגגאג גג num עג יגזי \_\_\_\_\_ /הגסג ב-tens
  4. גשב אג ספרג האגאג גג num עג יגזי \_\_\_\_\_ /הגסג ב-hundreds
  5. בנה מספר מ- \_\_\_\_\_ , \_\_\_\_\_ -/ \_\_\_\_\_ /הגסג ב-reverseNum
  6. הגג כפאט אג עגכג גג reverseNum
- א. השלימו את האלגוריתם.
- ב. ישמו את האלגוריתם בשפת C#.
-

## פרק 5 – ביצוע מותנה

בשני הפרקים הקודמים ראינו אלגוריתמים שבמהלך ביצועם מתבצעת כל אחת מהוראות האלגוריתם. בפרק זה נכיר אלגוריתמים אשר במהלך ביצועם לא מתבצעות תמיד כל הוראות האלגוריתם. אלגוריתמים אלה כוללים הוראות המורות על ביצוע קבוצת הוראות אחת או על קבוצת הוראות אחרת, בהתאם לקיומו או לאי-קיומו של תנאי. הוראות אלו נקראות **הוראות לביצוע-בתנאי**. קיומו של התנאי תלוי בקלט לאלגוריתם.

למשל כאשר נערכות בחירות בין שני מועמדים, משווים את מספרי הקולות לכל מועמד. המועמד שצבר יותר קולות הוא המנצח בבחירות. הקלט של אלגוריתם להכרזת המנצח יהיה מספר הקולות אשר צבר כל מועמד. אם יתקיים התנאי שהנתון הראשון בקלט גדול מן השני אז תתבצע באלגוריתם הוראה להכרזת המועמד הראשון כמנצח. אחרת תתבצע הוראה להכרזת המועמד השני כמנצח.

### 5.1 הוראה לביצוע-בתנאי

*הוראה לביצוע-בתנאי במבנה אם... אז...*

#### קצ'ה 1

**מטרת הבעיה ופתרונה:** הצגת אלגוריתם הכולל הוראה לביצוע-בתנאי.

פְּלִינְדְרוֹם (palindrome) הוא מילה, מספר או משפט שניתן לקרוא משני הכיוונים, משמאל לימין ומימין לשמאל, ולקבל אותה תוצאה. למשל השם ישי הוא שם פלינדרומי, וכן המילים זוז, שמש, הסוסה. המילה הפלינדרומית הארוכה ביותר בעברית שיש לה משמעות היא "ולכשתשכלו". המשפט: "ילד כותב בתוך דלי" גם הוא פלינדרומי. המספרים 17371 ו-4994 הם דוגמאות למספרים פלינדרומים. פתחו אלגוריתם אשר הקלט שלו הוא מספר שלם חיובי תלת-ספרתי, והפלט שלו הוא הודעה אם המספר הנתון הוא פלינדרום. ישמו את האלגוריתם בתוכנית מחשב בשפת C#.

#### ניתוח הבעיה בעזרת דוגמאות

דוגמאות למספרים שלמים חיוביים תלת-ספרתיים פלינדרומים: 777 424 787  
דוגמאות למספרים שלמים חיוביים תלת-ספרתיים שאינם פלינדרומים: 192 234 778

#### 5.1 שאלה

הגדירו כלל פשוט המתאר מתי מספר הוא פלינדרום ומתי אינו פלינדרום.  
בדוגמאות הפשוטות שבחנו התברר שמספר הוא פלינדרומי רק כאשר ספרת האחדות שווה לספרת המאות.

#### פירוק הבעיה לתת-משימות

על פי ניתוח הבעיה נפרק את המשימה העומדת לפנינו באופן הבא:

1. קליטת מספר שלם חיובי תלת-ספרתי.

2. מציאת ספרת האחדות.

3. מציאת ספרת המאות.
4. השוואת הספרות. אם הספרות שוות נציג הודעה שהמספר פלינדרומי, אחרת נציג הודעה שהמספר אינו פלינדרומי.

### בחירת משתנים

**num** – שלם, ישמור את המספר התלת-ספרתי הנקלט.

**units** – שלם, ישמור את ספרת האחדות.

**hundreds** – שלם, ישמור את ספרת המאות.

### האלגוריתם

את תת-משימה 2 ו-3 למדנו לבצע בפרק הקודם. על מנת למצוא את ספרת האחדות של מספר כלשהו, נחשב את תוצאת שארית החילוק של המספר ב-10. על מנת למצוא את ספרת המאות של מספר תלת-ספרתי, נחשב את תוצאת החילוק של המספר ב-100.

כיצד ננסח באלגוריתם את תת-משימה 4?

ננסח תנאי אשר על פי קיומו ניתן לקבוע אם המספר הוא פלינדרומי. התנאי יהיה: ספרת האחדות שווה לספרת המאות.

אם התנאי מתקיים יש להודיע: המספר פלינדרומי.

אם התנאי לא מתקיים יש להודיע: המספר אינו פלינדרומי.

הוראה זו מבטאת את הרעיון של בחירה באחת מבין שתי אפשרויות לביצוע על פי תנאי. נציג אותה במבנה הבא:

*אם ספרת האגרות שווה לספרת המאות*

*הצג כפלט: המספר פלינדרומי*

*אגרת*

*הצג כפלט: המספר אינו פלינדרומי*

הוראה במבנה זה נקראת **הוראה לביצוע-בתנאי**. הוראה לביצוע-בתנאי היא הוראת **בקרה**, משום שהיא משפיעה על מהלך הביצוע של האלגוריתם, כלומר קובעת אם יבוצעו הוראות אלו או אחרות.

### יישום האלגוריתם

ההוראה לביצוע-בתנאי המופיעה באלגוריתם מיושמת בשפת C# בהוראת **if...else...**, ואופן כתיבתה דומה לצורת הכתיבה העברית **אם...אגרת...**

אבל כיצד כותבים ב-C# תנאי כמו זה המופיע באלגוריתם: **ספרת האגרות שווה לספרת המאות**? פעולת ההשוואה נכתבת בעזרת סימן הפעולה **==** (שני סימני שוויון רצופים).

לכן, את תת-משימה 4 באלגוריתם ניתן ליישם בשפת C# כך:

```
if (units == hundreds)
{
    Console.WriteLine("The number {0} is a palindrome", num);
}
else
{
    Console.WriteLine("The number {0} is not a palindrome", num);
}
```

## התוכנית המלאה

```

/*
הקלט: מספר שלם חיובי תלת-ספרתי
הפלט: הודעה אם המספר הוא פלינדרום
*/
using System;
public class Palindrome
{
    public static void Main ()
    {
        // הצהרה על משתנים בתוכנית
        int num;           // מספר שלם חיובי תלת-ספרתי
        int units;         // ספרת האחדות
        int hundreds;     // ספרת המאות
        // קליטת המשתנים
1. Console.WriteLine("Enter a 3 digit number: ");
2. num = int.Parse(Console.ReadLine());
        // פירוק ספרת האחדות וספרת המאות
3. units = num % 10;
4. hundreds = num / 100;
        // ההוראה לביצוע-בתנאי
5. if (units == hundreds)
    {
        // הצגת הודעה: המספר פלינדרומי
5.1. Console.WriteLine("{0} is a palindrome", num);
    }
6. else
    {
        // הצגת הודעה: המספר אינו פלינדרומי
6.1. Console.WriteLine("{0} is not a palindrome", num);
    }
    } // Main
} // class Palindrome

```

## המעקב

נבדוק את התוכנית Palindrome באמצעות מעקב אחר ביצועה עבור דוגמאות קלט שונות.

עבור הקלט 363, יתקיים התנאי *ספרת האחדות שווה לספרת המאות*, ותתקבל טבלת המעקב הבאה:

מספר השורה	המשפט לביצוע	num	units	hundreds	units == hundreds	פלט
1	Console.WriteLine("Enter a 3 digit number: ");	?	?	?		Enter a 3 digit number:
2	num = int.Parse(...);	363	?	?		
3	units = num % 10;	363	3	?		
4	hundreds = num / 100;	363	3	3		
5	if (units == hundreds)	363	3	3	אמת	
5.1	Console.WriteLine("{0} is a palindrome", num);	363	3	3		363 is a palindrome

עבור הקלט 366, לא מתקיים התנאי, ומתקבלת טבלת המעקב הבאה:

מספר השורה	המשפט לביצוע	num	units	hundreds	units == hundreds	פלט
1	Console.Write("Enter a 3 digit number: ");	?	?	?		Enter a 3 digit number:
2	num = <b>int</b> .Parse(...);	366	?	?		
3	units = num % 10;	366	6	?		
4	hundreds = num / 100;	366	6	3		
5	<b>if</b> (units == hundreds)	366	6	3	שקר	
6.1	Console.WriteLine("{0} is not a palindrome", num);	366	6	3		366 is not a palindrome

**שימו** ♥ להבדל בין עמודת "המשפט לביצוע" בשתי הטבלאות. בטבלה הראשונה מופיע המשפט:

```
Console.WriteLine("{0} is a palindrome", num);
```

זאת מכיוון שהתנאי מתקיים ולכן מתבצע תחום ה-**if** של משפט ה-**if**.

לעומת זאת, בטבלה השנייה מופיע המשפט:

```
Console.WriteLine("{0} is not a palindrome", num);
```

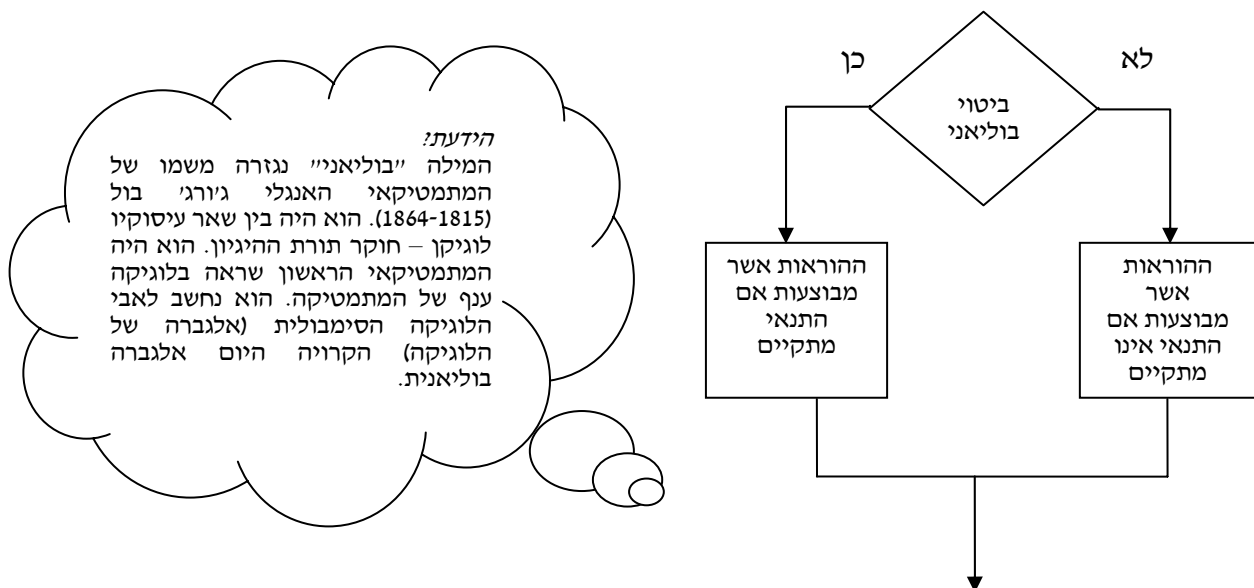
זאת מכיוון שהתנאי אינו מתקיים ולכן מתבצע תחום ה-**else** של המשפט.

### סוף פתרון בעיה 1

הוראה לביצוע-בתנאי כוללת בתוכה כמובן תנאי, שמכוון את המשך הביצוע. באלגוריתם שנתנו לפתרון בעיה 1 התנאי ששילבנו בהוראה הוא **האם 366 הוא מספר פלינדרום**.

התנאי העומד בבסיסה של ההוראה ביצוע-בתנאי מיוצג בביטוי **בוליאני**. כמו ביטוי חשבוני, גם לביטוי בוליאני יש ערך. אלא שערך זה אינו ערך מספרי. ערכו של ביטוי בוליאני יכול להיות אחד משניים – אמת (**true**) או שקר (**false**). אם התנאי שמייצג הביטוי הבוליאני מתקיים אז ערכו של הביטוי הבוליאני הוא **true**. אם התנאי שמייצג הביטוי הבוליאני אינו מתקיים אז ערכו של הביטוי הבוליאני הוא **false**.

ניתן להמחיש את המשמעות של ההוראה לביצוע-בתנאי באמצעות תרשים הזרימה הבא:



במהלך הספר לא נציג אלגוריתמים באמצעות תרשימים. הצגה זו ארוכה מאוד וכן שונה מאוד מן המראה של תוכנית מחשב. אם השימוש בתרשימים מסייע לכם, כדאי לכם להשתמש בהם מדי פעם במהלך פיתוח אלגוריתם.

לעתים ננסח את התנאי באופן מילולי ולעתים נעדיף לנסח אותו בעזרת סימנים. ניתן לבחור בכל צורת ניסוח, כל עוד התנאי המתקבל הוא ברור וחד-משמעי. למשל עבור המשימה: השוואת ערכי המשתנים a ו-b והשמת הערך הקטן מביניהם במשתנה min, ניתן לנסח את התנאי במילים או בעזרת סמלים:

$a < b$  אס  $a$  ערכו  $b$  אס  $b < a$  או  $a < b$  אס

נכיר ביטויים בוליאניים המציינים השוואה של ערכים, של משתנים ושל ביטויים חשבוניים. למשל:  $a + b > 0$ ,  $a < b - 1$ ,  $units == hundreds$ .

לסיכום – נציג את אופן הכתיבה האלגוריתמית של הוראה לביצוע-בתנאי ואת יישומה בשפת C#:

הוראה לביצוע-בתנאי נכתבת בצורה זו:

אס <ביטוי בוליאני>

<דבר 1 הוראה>

אגה

<דבר 2 הוראה>

ביטוי בוליאני מייצג תנאי, שערכו יכול להיות true (אמיתי) או false (שקרי).

ביצוע של הוראה לביצוע-בתנאי מתחיל תמיד בחישוב ערכו של הביטוי הבוליאני. אם ערכו אמיתי, תתבצע סדרת ההוראות הראשונה, אחרת תתבצע סדרת ההוראות השנייה.

בשפת C# הוראה לביצוע-בתנאי מיושמת במשפט if.

מבנה משפט if ב-C# הוא:

if (ביטוי בוליאני)

{

ההוראות אשר יבוצעו אם התנאי מתקיים

}

else

{

ההוראות אשר יבוצעו אם התנאי אינו מתקיים

}

החלק שנמצא בתוך זוג הסוגריים המסולסלים {...} הראשון נקרא תחום ה-if ובו נמצאות ההוראות אשר יבוצעו אם התנאי מתקיים. החלק שנמצא בתוך זוג הסוגריים המסולסלים השני נקרא תחום ה-else ובו נמצאות ההוראות אשר מבוצעות אם התנאי אינו מתקיים.

שימו ♥: במקרים שיש הוראה אחת לביצוע בתחום ה-if, אפשר להשמיט את הסוגריים המסולסלים של התחום. גם במקרים שיש הוראה אחת לביצוע בתחום ה-else, דין הסוגריים זהה וניתן להשמיטם.

הביטוי הבוליאני שהופיע במשפט ה-`if` שראינו בפתרון בעיה 1 השתמש בסימן ההשוואה `==`. בשפת C# קיימים סימני השוואה נוספים. בטבלה הבאה מובאים סימני ההשוואה בשפת C#.

סימן ההשוואה ב-C#	סימן ההשוואה המקובל במתמטיקה	משמעות סימן ההשוואה	דוגמה ב-C#	דוגמה במתמטיקה
<code>==</code>	<code>=</code>	שווה	<code>x == 5</code>	<code>x = 5</code>
<code>!=</code>	<code>≠</code>	שונה	<code>x != y</code>	<code>x ≠ y</code>
<code>&lt;</code>	<code>&lt;</code>	קטן	<code>x &lt; 2</code>	<code>x &lt; 2</code>
<code>&lt;=</code>	<code>≤</code>	קטן או שווה	<code>x &lt;= 1</code>	<code>x ≤ 2</code>
<code>&gt;</code>	<code>&gt;</code>	גדול	<code>y &gt; 0</code>	<code>y &gt; 0</code>
<code>&gt;=</code>	<code>≥</code>	גדול או שווה	<code>y &gt;= 8</code>	<code>y ≥ 8</code>

סימני ההשוואה בשפת C#

**שימו ⚡ :** הסימנים המתמטיים `<`, `>`, `=`, `≠` כתובים ב-C# בצורה שונה במקצת. בשפת C# ההשוואה שבתוך התנאי מתבצעת על ידי סימן הפעולה `==` ולא על ידי הסימן `=`. הסימן `=` שמור ב-C# להשמה.

## 5.2 שאלה

נסחו הוראה לביצוע-בתנאי עבור כל אחת מן המשימות הבאות :

- השוואת ערכי המשתנים `a` ו-`b` והצגת הודעה אם הערכים שווים או שונים.
- השוואת ערכי המשתנים `a` ו-`b` והפחתת ערכו של המשתנה הקטן מהמשתנה הגדול. אם שני המשתנים שווים יש להפחית את ערכו של `b` מערכו של `a`.

## 5.3 שאלה

כתבו כל אחד מן התנאים המילוליים הבאים כביטוי בוליאני המשתמש בסימני ההשוואה של C# :

תנאי	ביטוי בוליאני
ערך המשתנה <code>a</code> שווה ל-0.	
ערך המשתנה <code>a</code> שווה לערך המשתנה <code>b</code> .	
ערך המשתנה <code>a</code> שווה לכפליים ערכו של המשתנה <code>b</code> .	
ערך המשתנה <code>a</code> שונה מערך המשתנה <code>b</code> .	
סכום ערכי המשתנים <code>a</code> ו- <code>b</code> קטן או שווה ל-10.	

## 5.4 שאלה

נניח שערכי המשתנים `a` ו-`b` הם 1 ו-2 בהתאמה. ציינו עבור כל אחד מן הביטויים הבוליאניים הבאים אם ערכו `true` או `false`.

ביטוי בוליאני	ערך
<code>a == 1</code>	
<code>a == b</code>	
<code>3a == b + 1</code>	
<code>2a &lt;= b</code>	
<code>2a != b</code>	

## שאלה 5.5

במשתנים girafaHeight ו-girafHeight שמורים הגבהים של ג'ירף ושל ג'ירפה. כתבו משפט if מתאים לביצוע המשימות הבאות:

א. הצגת הודעה אם הג'ירפה גבוהה מ-1.70 מ', או לא.

ב. הצגת הודעה אם הג'ירף גבוה יותר או אינו גבוה יותר מהג'ירפה.

כפי שראינו בפתרון בעיה 1: בטבלת מעקב אחר מהלך ביצוע של תוכנית הכוללת משפט if, נוח להוסיף עמודה עבור הביטוי הבוליאני, ולציין בה את ערך הביטוי. בטבלה שבפתרון בעיה 1 כתבנו "אמת" או "שקר". מעתה נכתוב true או false.

**שימו ♥ :** כאשר מופיע בתוכנית משפט if (הוראה לביצוע-בתנאי), יש לבדוק את מהלך הביצוע עבור דוגמאות קלט מגוונות. על הדוגמאות לכלול קלט שיביא לכך שערכו של הביטוי הבוליאני שבמשפט יהיה true וכן קלט שיביא לכך שערכו של הביטוי הבוליאני שבמשפט יהיה false. קלטים כאלו נקראים קלטים מייצגים.

לא מספיק לבדוק את מהלך הביצוע רק עבור אחד משני המקרים, כיוון שמקרה אחד אינו מעיד על האחר. עלינו לבדוק את שניהם, כפי שמדגימה השאלה הבאה.

## שאלה 5.6

מטרתו של משפט ה-if הבא היא השמת הערך הגדול מבין המשתנים a ו-b במשתנה max. חישוב מקסימום היא תבנית שימושית מאוד, שמשמשת בפתרון בעיות רבות.

```
if (a > b)
{
    max = a;
}
else
{
    max = b;
}
```

בחרו שתי דוגמאות של קלטים מייצגים לבדיקת המשפט.

באחת יהיה ערכו ההתחלתי של a גדול מערכו ההתחלתי של b, ובשנייה יהיה ערכו ההתחלתי של a קטן מערכו ההתחלתי של b.

בדקו את מהלך ביצוע המשפט באמצעות טבלת מעקב עבור כל אחת מן הדוגמאות.

מה יהיה מהלך ביצוע המשפט עבור המקרה שבו הערכים ההתחלתיים של a ו-b שווים?

## הוראה לביצוע-בתנאי במבנה אם...אז...

לעתים ברצוננו לבצע חלק של אלגוריתם כאשר תנאי מסוים מתקיים, ואיננו רוצים לבצע מאומה כאשר התנאי אינו מתקיים. נראה זאת בפתרון הבעיה הבאה.

## תשובה 2

**מטרת הבעיה ופתרונה:** הצגת הוראה לביצוע-בתנאי במבנה אם...אז... (ללא החלק אז...).

תלמידי כיתות י' בבית הספר טסים לירח. יש להזמין מספר מתאים של חלליות כך שלכל תלמיד יהיה מקום ישיבה ושמספר החלליות יהיה קטן ככל האפשר.



פתחו וישמו אלגוריתם להזמנת חלליות לירח כך שהקלט שלו הוא מספר התלמידים ומספר המושבים בחללית, והפלט שלו הוא מספר החלליות שיש להזמין.

### ניתוח הבעיה בעזרת דוגמאות

ייתכן כי מספר התלמידים הוא כפולה של מספר המושבים בחללית. במקרה כזה בכל החלליות שיוזמנו יהיו כל המושבים תפוסים.

לעומת זאת, ייתכן כי מספר התלמידים אינו כפולה של מספר המושבים בחללית. במקרה כזה תוזמן גם חללית אחת אשר רק חלק מהמושבים בה יהיו תפוסים.

### שאלה 5.7

בחרו שתי דוגמאות קלט מייצגות וציינו את הפלט עבור כל אחת מהן.

?

כיצד נחשב את מספר החלליות הדרוש?

יש לחשב את המנה ואת השארית של חלוקת מספר התלמידים במספר המושבים בחללית. מנת החלוקה שווה למספר החלליות המלאות. שארית החלוקה תקבע אם יש צורך בחללית נוספת. אם שארית החלוקה היא 0 הרי מספר התלמידים הוא כפולה של מספר המושבים בחללית. אחרת יש להזמין חללית נוספת שתפוסתה תהיה חלקית ושווה לשארית.

### פירוק הבעיה לתת-משימות

1. קליטת מספר התלמידים ומספר המושבים
2. חישוב מספר החלליות המלאות
3. חישוב מספר התלמידים שישארו לאחר מילוי החלליות המלאות
4. אם צריך חללית נוספת, חישוב מספר החלליות הכולל
5. הצגת הפלט: מספר החלליות

### בחירת משתנים

נבחר את המשתנים הבאים מטיפוס שלם:

`studentsNum` – ישמור את מספר התלמידים.  
`seatsPerShip` – ישמור את מספר המושבים בחללית.  
`shipsNum` – ישמור את מספר החלליות שיש להזמין.  
`leftoverNum` – ישמור את מספר התלמידים אשר ייוותרו לאחר מילוי החלליות המלאות.

### האלגוריתם

כיצד ננסח באלגוריתם את תת-משימה 4?

ניתן לתאר תת-משימה זו באופן הבא:

אם מספר התלמידים  $\geq$  מספר המושבים בחללית  
הפלט הוא `shipsNum + 1`  
אחרת הפלט הוא `shipsNum`

האלגוריתם לפתרון הבעיה כולל הוראה לביצוע-בתנאי במבנה **אם... (ללא החלק אחר...)**:

המשמעות של הוראה לביצוע-בתנאי במבנה **אם... היא שאם התנאי שבהוראה מתקיים יבוצעו ההוראות המתאימות, אחרת לא יבוצע דבר.**

## יישום האלגוריתם

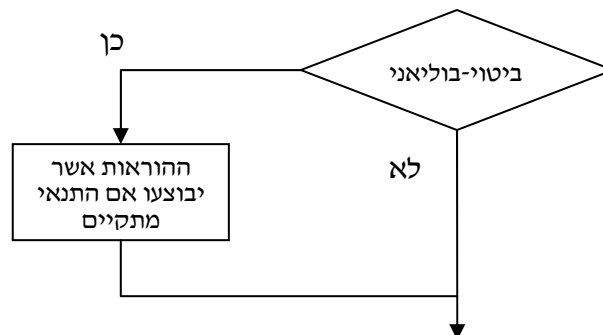
```
/*
קלט: מספר התלמידים והמושבים
פלט: מספר החלליות להזמנה
*/
using System;
public class SpaceshipOrder
{
    public static void Main ()
    {
        // הצהרה על משתנים בתוכנית
        int studentsNum; // מספר התלמידים
        int seatsPerShip; // מספר המושבים בחללית
        int shipsNum; // מספר החלליות שיש להזמין
        int leftoverNum; // מספר התלמידים שיישארו
        // לאחר מילוי החלליות המלאות
        // קליטת המשתנים
1. Console.WriteLine("Enter number of students:");
2. studentsNum = int.Parse(Console.ReadLine());
3. Console.WriteLine("Enter number of seats in a spaceship: ");
4. seatsPerShip = int.Parse(Console.ReadLine());
        // חישוב מספר החלליות המלאות
5. shipsNum = studentsNum / seatsPerShip;
        // חישוב מספר התלמידים שיוותרו
6. leftoverNum = studentsNum % seatsPerShip;
        // ההוראה לביצוע-בתנאי
7. if (leftoverNum > 0) // דרושה חללית נוספת שתפוסתה תהיה חלקית.
    {
7.1. shipsNum = shipsNum + 1;
    }
        Console.WriteLine("The number of spaceships is {0}",
                                                                    shipsNum);
    } // Main
} // class SpaceshipOrder
```

## שאלה 5.8

עקבו אחר ביצוע התוכנית באמצעות שתי טבלאות מעקב. האחת, כאשר התנאי מתקיים, והשנייה, כאשר אין התנאי מתקיים. לדוגמה: עבור הקלט 120 תלמידים ו-40 מושבים בכל חללית התנאי לא יתקיים, ואילו עבור הקלט 120 תלמידים ו-50 מושבים בכל חללית התנאי יתקיים.

**סוף פתרון בעיה 2**

ניתן להמחיש הוראה לביצוע-בתנאי במבנה **if**... באמצעות תרשים הזרימה הבא:



בשפת C# מבנה משפט ה-`if` ליישום הוראה לביצוע-בתנאי במבנה **if**... הוא:

```

if (ביטוי בוליאני)
{
    ההוראות אשר יבוצעו אם התנאי מתקיים
}
  
```

**שימו** ♥: במקרים שיש הוראה אחת לביצוע, אפשר להשמיט את הסוגריים המסולסלים של תחום ה-`if`.

**שימו** ♥: הערה שמלווה את התנאי (דרושה חללית...) מבהירה ביטוי בוליאני שמשמעותו איננה ברורה מיד עם קריאתו. נקרא להערה זו **תיאור משמעות של קיום תנאי**.

**תיאור משמעות של קיום תנאי** הוא תיעוד המסביר את תפקידו של ביטוי בוליאני בהוראה לביצוע-בתנאי. תיאור משמעות קיום תנאי מסייע לנו בקריאת תוכנית ובהבנתה.

בתוכניות שנפתח נשתדל לצרף תיאורי משמעות של קיום או של אי-קיום תנאי במשפטי `if` אשר כדאי להבהירם.

## שאלה 5.9

בנו טבלת מעקב אחר ביצוע התוכנית `SpaceshipOrder` לפתרון בעיה 2 עבור הקלט: 180 תלמידים ו-60 מושבים.

## שאלה 5.10

ביטוי בוליאני עשוי לכלול השוואה בין ביטויים חשבוניים שאינם פשוטים. למשל,  $(a + b) == (c + d)$  או  $(a / 2) == 0$ .

כתבו משפט `if` אשר בו:

- אם ערכו של  $a$  גדול מפעמיים ערכו של  $b$ , מוכפל ערכו של  $c$  ב-2.
- אם ערכו של  $c$  קטן מסכום ערכי  $a$  ו- $b$ , מופחת מ- $c$  ערכו של  $a$ .
- אם ערכו של  $a$  הוא כפולה של 10, מושם ב- $b$  ערכו של  $a$ .
- אם מכפלת ערכי  $a$  ו- $b$  גדולה מסכום ערכי  $b$  ו- $c$ , סימנו של הערך הנתון ב- $c$  מתהפך.

### שאלה 5.11

המשתנה  $a$  מכיל מספר שלם חיובי קטן מ-100. השלימו את תיאור המשמעות של קיום תנאי בכל אחד מן המשפטים הבאים, כלומר הסבירו את תפקידו של כל תנאי ומה הוא בודק:

א. \_\_\_\_\_ ב. \_\_\_\_\_

```
// _____ // _____  
if (a == (a % 10)) if ((a / 10) > 5)  
    Console.WriteLine("A"); Console.WriteLine("Pass");
```

### שאלה 5.12

נתון קטע התוכנית הבא, שהמשתנים בו הם מטיפוס שלם:

```
max = a;  
if (b > a)  
    max = b;
```

א. בנו טבלאות מעקב אחר מהלך ביצוע קטע התוכנית עבור הערכים ההתחלתיים 30 ו-30 ועבור הערכים ההתחלתיים 40 ו-70 במשתנים  $a$  ו- $b$  בהתאמה.  
ב. מהי מטרת קטע התוכנית?  
ג. נתון קטע התוכנית הבא:

```
if (a >= b)  
    max = a;  
else  
    max = b;
```

האם יש הבדל בין מטרת קטע התוכנית הנתון בסעיף זה ובין מטרת קטע התוכנית שהוצג בתחילת השאלה? הסבירו את תשובתכם.

להעמקה בתבנית **מציאת מקסימום** פנו לסעיף התבניות המופיע בסוף הפרק.

### שאלה 5.13

פתחו אלגוריתם אשר הקלט שלו הוא שני ציונים של תלמיד, שערכם הוא מספר שלם בין 0 ל-100 והפלט שלו הוא מספר המציין כמה מן הציונים גבוהים מ-80.  
א. מהם ערכי הפלט האפשריים?  
ב. ישמו את האלגוריתם בשפת C#.

## התניית ביצוע של שתי הוראות או יותר

עד כה ראינו הוראות פשוטות לביצוע-בתנאי: אם התנאי התקיים התבצעה הוראה יחידה. גם במקרה שהתנאי לא התקיים התבצעה הוראה יחידה. בבעיה הבאה נציג הוראה לביצוע-בתנאי שתחומיה השונים כוללים יותר מהוראה אחת, ומורים על ביצוע כמה תת-משימות.

### קצ'ה 3

**מטרת הבעיה ופתרונה:** הצגת הוראה לביצוע-בתנאי שתחומיה כוללים מספר הוראות.

פתחו אלגוריתם המתאר משחק שנקרא "שש אש". הקלט של האלגוריתם הוא מספר שלם  $x$ . אם  $x$  מתחלק ב-6 יזכה המשתתף ב- $x-6$  ש, אך אם  $x$  אינו מתחלק ב-6, יפסיד המשתתף  $x-10$  ש. פלט האלגוריתם הוא הודעת סכום הזכייה או ההפסד מוקף בכוכביות. ישמו את האלגוריתם בשפת C#.

### פירוק הבעיה לתת-משימות

- קליטת המספר
- חישוב סכום הזכייה או ההפסד
- הצגת הסכום מוקף בכוכביות בצירוף הודעה מתאימה

### בחירת משתנים

נבחר את המשתנים הבאים מטיפוס שלם:

**num** – לשמירת המספר הניתן כקלט

**sum** – לשמירת סכום הזכייה או ההפסד

### האלגוריתם

- קלט מספר  $num$
- אם ערכו של  $num$  מתחלק ב-6 לא שארית
  - שש אש מכפול  $num$  ב-6 והשם  $sum$
  - הצג הודעה על זכייה בסכום  $sum$  המוקף בכוכביות
- אחר
  - שש אש מכפול  $num$  ב-10 והשם  $sum$
  - הצג הודעה על הפסד של הסכום  $sum$  המוקף בכוכביות

### יישום האלגוריתם

```
/*
קלט: מספר שלם
פלט: הודעה על זכייה אם המספר מתחלק ב-6, או על הפסד אם המספר אינו
מתחלק ב-6. ההודעה תכלול את סכום הזכייה או ההפסד
*/
using System;
public class SheshEsh
{
    public static void Main ()
```

```

{
    // הצהרה על משתנים בתוכנית
    int num; //מספר שנקלט
    int sum; //סכום הזכייה או ההפסד
    // קליטת המשתנים
1. Console.WriteLine("Enter a number: ");
2. num = int.Parse(Console.ReadLine());
    //הוראה לביצוע-בתנאי: חישוב סכום הזכייה או ההפסד
3. if (num % 6 == 0)
    { // הזכייה
3.1. sum = num * 6;
3.2. Console.WriteLine("*****");
3.3. Console.WriteLine("* You won {0} shekels ", sum);
3.4. Console.WriteLine("*****");
    }
4. else
    { // ההפסד
4.1. sum = num * 10;
4.2. Console.WriteLine("*****");
4.3. Console.WriteLine("* You lost {0} shekels ", sum);
4.4. Console.WriteLine("*****");
    }
} // Main
} // class SheshEsh

```

## מעקב

נעקוב אחר מהלך ביצוע התוכנית SheshEsh עבור הקלט 12:

	המשפט לביצוע	num	sum	num%6==0	פלט
1	Console.WriteLine("Enter a number: ");	?	?		Enter a number:
2	num = int.Parse(Console.ReadLine());	12	?		
3	if (num % 6 == 0)	12	?	true	
3.1	sum = num * 6;	12	72		
3.2	Console.WriteLine("***...")	12	72		*****
3.3	Console.WriteLine("You Won {0} shekels", sum);	12	72		* You won 72 shekels *
3.4	Console.WriteLine("***...")	12	72		*****

## סוף פתרון בעיה 3

### שאלה 5.14

בנו טבלת מעקב אחר ביצוע התוכנית SheshEsh מפתרון בעיה 3 עבור הקלט 10.

### שאלה 5.15

במשתנים  $x$  ו- $y$  שמורים שני ערכים מטיפוס שלם. מטרת התוכנית היא להציג כפלט את המספר הגדול ראשון ואחריו את המספר הקטן. השלימו את משפט ה-if הבא באמצעות ביטוי בוליאני מתאים.

```

if ( _____ )
{

```

```

temp = x;
x = y;
y = temp;
}
Console.WriteLine(x, y);

```

### שאלה 5.16

פתחו אלגוריתם שהקלט שלו הוא שני מספרים חיוביים. המספר הראשון מציין את משקלו של החתול גארפילד בק"ג והמספר השני מציין את משקלו של הכלב סנופי. אם משקלו של גארפילד גדול ממשקלו של סנופי, האלגוריתם צריך לחשב כמה ק"ג שוקל גארפילד יותר מסנופי ולהציג הודעה מתאימה הכוללת את הערך שחושב. אחרת האלגוריתם צריך לחשב כמה ק"ג שוקל סנופי יותר מגארפילד ולהציג הודעה מתאימה הכוללת את הערך שחושב.

למשל פלט מתאים עבור הקלט 15 10 הוא:

Snoopy is heavier than Garfield, the difference is 5 kg

פלט מתאים עבור הקלט 13 17 הוא:

Garfield is heavier than Snoopy, the difference is 4 kg

ישמו את האלגוריתם בשפת C#.

### שאלה 5.17

פתחו אלגוריתם שהקלט שלו הוא שני מספרים שלמים. אם המספר הראשון גדול מהשני, האלגוריתם מחשב את סכומם ומציג אותו כפלט. אחרת הוא מחשב את מכפלת המספרים ומציג אותה כפלט. ישמו את האלגוריתם בשפת C#.

## ביטויים בוליאניים הכוללים תווים

הביטויים הבוליאניים שראינו עד כה כללו פעולות השוואה על מספרים. עם זאת המספרים השלמים או המספרים הממשיים אינם הטיפוסים היחידים שערכיהם ניתנים להשוואה. בגלל ההתאמה של תווים למספרים שלמים, גם ערכיו של הטיפוס התווי ניתנים להשוואה, כפי שמדגימה הבעיה הבאה.

### תוצאה 4

**מטרת הבעיה ופתרונה:** הצגת פעולות השוואה על טיפוס תווי.

בסדרות של סימנים, הכוללות מספר סופי של סימנים, לכל סימן יש סימן עוקב מלבד לאחרון. עבור סדרות כאלה נהוג להגדיר את הסימן הראשון כ"עוקב מעגלית" לסימן האחרון. פתחו וישמו אלגוריתם אשר הקלט שלו הוא אות מן האותיות הגדולות של הא"ב האנגלי (A..Z), והפלט שלו הוא האות העוקבת "בצורה מעגלית" והודעה מתאימה. עבור אות קלט השונה מהאות Z יהיה הפלט האות הבאה בא"ב והודעה המכריזה שזו האות העוקבת. עבור האות Z הפלט יהיה האות A והודעה המכריזה על חזרה להתחלה.

### פירוק הבעיה לתת-משימות

1. קליטת אות

2. חישוב האות העוקבת "בצורה מעגלית"

### 3. הצגת האות העוקבת בצירוף הודעה מתאימה

#### בחירת משתנים

נבחר את המשתנים הבאים מטיפוס תווי:

**letter** – ישמור את האות הניתנת כקלט

**nextLetter** – ישמור את האות העוקבת "בצורה מעגלית" לאות הקלט

#### האלגוריתם

1. קלוט את letter-2
2. אם ערכו של letter הוא 'Z'
  - 2.1. השם את האות 'A' ב nextLetter
  - 2.2. הצג כקלט את ההודעה "Back to start:" ואם ערכו של nextLetter
3. אחר
- 3.1. שם את האות העוקבת/אות ה/אות letter-2 והשם ב nextLetter
- 3.2. הצג כקלט את ההודעה "The next letter is:" ואם ערכו של nextLetter

#### יישום האלגוריתם

כיצד נבצע את ההשוואה בשורה 2 של האלגוריתם? ניתן להשוות בין ערכים מטיפוס תווי, בדיוק כשם שניתן להשוות בין ערכים מטיפוס שלם או ממשי, בשימוש באופרטור ההשוואה הרגיל של השפה. לכן הביטוי הבוליאני המתאים הוא `letter == 'Z'`.

#### התוכנית המלאה

```
/*
קלט: אות אנגלית גדולה
פלט: הודעה הכוללת את האות העוקבת "בצורה-מעגלית" לאות הנתונה
*/
using System;
public class NextLetterInCircle
{
    public static void Main ()
    {
        // הצהרה על משתנים בתוכנית
        char letter; // אות הקלט
        char nextLetter; // האות העוקבת
        // קליטת המשתנים
        1. Console.WriteLine("Enter a letter from the ABC: ");
        2. letter = char.Parse(Console.ReadLine());
        // ההוראה לביצוע-בתנאי: חישוב האות העוקבת
        3. if (letter == 'Z')
        {
            // המקרה המיוחד - האות האחרונה
            3.1. nextLetter = 'A';
            3.2. Console.WriteLine("Back to start: {0}", nextLetter);
        } // if
        4. else
        {
            // האות איננה האחרונה
            4.1. nextLetter = (char) (letter + 1); //המרת טיפוס (casting)
            4.2. Console.WriteLine("The next letter is: {0}", nextLetter);
        } // else
    }
}
```



```

    } // Main
} // class NextLetterInCircle

```

## מעקב

נעקוב אחר מהלך ביצוע התוכנית NextLetterInCircle עבור הקלט 'C':

	המשפט לביצוע	letter	nextLetter	letter=='Z'	פלט
1	Console.Write("Enter a letter ... ");	?	?		Enter a letter ...
2	letter = char.Parse(Console.ReadLine());	'C'	?		
3	if (letter == 'Z')	'C'	?	false	
4.1	nextLetter = (char)(letter + 1);	'C'	'D'		
4.2	Console.WriteLine("The next letter is: {0}", nextLetter);	'C'	'D'		The next letter is: D

## סוף פתרון בעיה 4

### שאלה 5.18

בנו טבלת מעקב אחר ביצוע התוכנית NextLetterInCircle לפתרון בעיה 4 עבור הקלט 'Z'.

כזכור, קבוצת התווים של המחשב כוללת גם את הספרות '0','1',...,'9'. כלומר ניתן להתייחס אל ספרה כאל ערך מטיפוס תווי. ספרות עוקבות מסודרות כתווים עוקבים בקבוצת התווים. השאלה הבאה מתייחסת לספרות כאל תווים.

### שאלה 5.19

נתון קטע התוכנית הבא:

```

char digit;
char x;
Console.Write("Enter a digit between 0 and 9: ");
digit = char.Parse(Console.ReadLine());
if (digit == '0')
    x = '9';
else
    x = (char)(digit - 1);
Console.WriteLine(x);

```

הקלט לקטע התוכנית הוא ספרה בין '0' ל-'9'.

א. מהו הפלט עבור הקלט '5', ומהו הפלט עבור הקלט '0'?

ב. מהו הקלט אשר הפלט עבורו יהיה 8?

ג. מהי מטרת קטע התוכנית?

מאחר שערכי הטיפוס התווי ניתנים להשוואה, ניתן להשוות ערכים מטיפוס תווי גם באמצעות הסימנים <, >, <=, >=, ולא רק באמצעות הסימנים == ו-!=.

למשל נניח שערכי המשתנים מטיפוס תווי let1 ו-let2 הם 'E' ו-'T' בהתאמה. אז ערכו של כל אחד מן הביטויים הבוליאניים הבאים הוא true: 'A' < 'B', '7' > '3', 'E' >= let1 ו-let1 <= let2.

ערכיו של הטיפוס התווי ניתנים להשוואה. לכן ניתן להשתמש בכל פעולות ההשוואה על תווים בדומה לשימושן על ערכים מספריים.

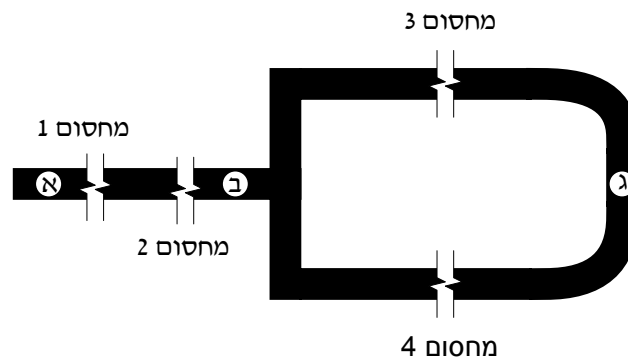
### שאלה 5.20

פתחו אלגוריתם אשר הקלט שלו הוא שתי אותיות שונות מן הא"ב האנגלי, והוא מציג את אותיות הקלט פעמיים: בשורה אחת ב"סדר עולה" ובשורה הבאה ב"סדר יורד". "סדר עולה" פירושו: האות שמופיעה קודם בא"ב האנגלי תוצג משמאל, והאות האחרת תוצג מימינה. "סדר יורד" הוא סדר הפוך ל"סדר עולה". ישמו את האלגוריתם בשפת C#.

להעמקה בתבנית **סידור ערכים בסדרה** פנו לסעיף התבניות המופיע בסוף הפרק.

## 5.2 תנאי מורכב

בסעיף זה נכיר תנאים מורכבים. תנאים מורכבים הם תנאים הבנויים מקישור של תנאים פשוטים יותר. נבחן את השרטוט הבא המתאר מערכת כבישים:



איור 5.1 – מערכת כבישים

במערכת הכבישים המתוארת באיור 5.1 ניתן להגיע מנקודה א לנקודה ג. כדי לעשות זאת יש להגיע מנקודה א לנקודה ב, ומנקודה ב לנקודה ג. על הכבישים נמצאים מחסומים. כדי להגיע מנקודה א לנקודה ב יש לעבור במחסום 1 ובמחסום 2. כלומר, רק אם אפשר לעבור בשני המחסומים, אפשר להגיע מנקודה א לנקודה ב. ניתן לתאר זאת כך:

אם מחסום 1 מורם ואם מחסום 2 מורם  
ניתן להגיע מנקודה א לנקודה ב  
אחרת  
לא ניתן להגיע מנקודה א לנקודה ב

תנאי המעבר מנקודה א לנקודה ב מתואר בתנאי מורכב, שהוא קישור התנאי מחסום 1 מורם אל התנאי מחסום 2 מורם באמצעות המילה **ואם**.

נתבונן כעת באפשרות להגיע מנקודה ב לנקודה ג. כדי לעשות זאת יש לעבור במחסום 3 או במחסום 4. כלומר, אם אפשר לעבור באחד המחסומים (או בשניהם) אפשר להגיע מנקודה ב לנקודה ג. ניתן לתאר זאת כך:

אם מחסום 3 מורם או מחסום 4 מורם  
ניתן להגיע מנקודה ב לנקודה ג  
אחרת  
לא ניתן להגיע מנקודה ב לנקודה ג

תנאי המעבר מנקודה ב לנקודה ג מתואר על ידי תנאי מורכב, שהוא קישור התנאי  $3 \wedge M$  אל התנאי  $4 \wedge M$  באמצעות המילה  $!A$ .

**א-ו-א** הם קשרים לוגיים, המאפשרים ליצור מביטויים בוליאניים פשוטים ביטויים בוליאניים מורכבים.

## הקשר א-ו-א

ראשית נתמקד בקשר הראשון מבין השניים שהודגמו בניתוח של איור 5.1.

### פעולה 5

**מטרת הבעיה ופתרונה:** הצגת תנאי מורכב הכולל את הקשר א-ו-א.

סדרת מספרים נקראת "סדרה עולה ממש" אם ערכו של כל איבר בסדרה קטן ממש מערכו של האיבר הבא אחריו. כלומר ערכו של האיבר הראשון בסדרה קטן ממש מערכו של האיבר השני, ערכו של האיבר השני קטן ממש מערכו של האיבר השלישי, וכן הלאה. למשל סדרת המספרים 10 7 2 1 היא סדרה עולה ממש, וסדרת המספרים 7 3 1 איננה סדרה עולה ממש. פתחו אלגוריתם אשר הקלט שלו הוא סדרה של שלושה מספרים שלמים, והפלט שלו הוא הודעה האם סדרת המספרים היא סדרה עולה ממש. אם הסדרה עולה ממש, יש לצרף להודעה את סדרת ההפרשים שבין איברי הסדרה המקורית. כלומר את ההפרש בין המספר השני לראשון, ואת ההפרש בין המספר השלישי לשני. ישמו את האלגוריתם בשפת התכנות C#.

## פירוק הבעיה לתת-משימות

- קליטה של שלושת איברי הסדרה.
- בדיקה אם הסדרה עולה ממש והצגת הודעה מתאימה.

## בחירת משתנים

נבחר שלושה משתנים מטיפוס שלם לשמירת שלושת איברי הסדרה:  
 $num1, num2, num3$  – ישמרו את שלושת מספרי הסדרה הנתונה.

## האלגוריתם

? בכתובת האלגוריתם עלינו לנסח תנאי אשר יתקיים כאשר הסדרה הנתונה עולה ממש, ולא יתקיים כאשר הסדרה איננה עולה ממש. מה יהיה התנאי המתאים?

התנאי אשר יתקיים כשהסדרה עולה ממש יהיה:

*המספר השלישי גדול מהמספר השני ואם המספר השני גדול מהמספר הראשון*

ניתן לכתוב זאת גם כך:

$num3 > num2$  א-ו  $num2 > num1$

**שימו:** ♥ אופן הכתיבה הבא אינו חוקי:  $num3 > num2 > num1$

תנאי זה הוא תנאי מורכב, הכולל קישור באמצעות המילה א-ו בין שני התנאים.

נציג אלגוריתם לפתרון הבעיה, תוך שימוש בתנאי שניסחנו :

1. קלט שלושה איברי סדרה כ- $num1$ , כ- $num2$  וכ- $num3$
2. אם  $num3 > num2$  וגם  $num2 > num1$
- 2.1. הציג כקלט הודעה כי הסדרה עולה ממש
- 2.2. הציג כקלט את ערך ההפרש  $num2 - num1$  / את ערך ההפרש  $num3 - num2$
3. אחרת
- 3.1. הציג כקלט הודעה כי הסדרה אינה עולה ממש

**שימו ♥ :** באלגוריתם מופיעה הוראת פלט שכוללת ערכי הפרשים בין משתנים. לא בחרנו משתנים לשמירת ההפרשים, אלא כללנו בהוראת הפלט ביטויים המבטאים את ההפרשים.

### יישום האלגוריתם

ב-C#, הקשר /גם נכתב באמצעות הסימן &&.

```
/*
קלט: 3 מספרים שלמים
פלט: הודעה אם סדרת המספרים היא סדרה עולה ממש
*/
using System;
public class CheckSequence
{
    public static void Main ()
    {
        // הצהרה על משתנים בתוכנית
        int num1, num2, num3;
        // קליטת המשתנים
1. Console.WriteLine("Enter first number: ");
2. num1 = int.Parse(Console.ReadLine());
3. Console.WriteLine("Enter second number: ");
4. num2 = int.Parse(Console.ReadLine());
5. Console.WriteLine("Enter third number: ");
6. num3 = int.Parse(Console.ReadLine());
7. if ((num3 > num2) && (num2 > num1))
    {
        // הסדרה עולה ממש
7.1. Console.WriteLine("The sequence of numbers is " +
                        "strongly increasing");
7.2. Console.WriteLine("The sequence of differences " +
                        "is {0} {1}", (num2 - num1), (num3 - num2));
    }
8. else
8.1. Console.WriteLine("The sequence of numbers is not " +
                        "strongly increasing");
    } // Main
} // class CheckSequence
```

## מעקב

נעקוב אחר מהלך ביצוע התוכנית עבור הקלט 1 2 4 :

מספר שורה	המשפט לביצוע	num1	num2	num3	Num3>num2	num2>num1	פלט
1	Console.Write("Enter first number: ");	?	?	?			Enter first number:
2	num1 = <b>int</b> .Parse(Console.ReadLine());	1	?	?			
3	Console.Write("Enter second number: ");	1	?	?			Enter second number:
4	num2 = <b>int</b> .Parse(Console.ReadLine());	1	2	?			
5	Console.Write("Enter third number: ");	1	2	?			Enter third number:
6	num3 = <b>int</b> .Parse(Console.ReadLine());	1	2	4			
7	<b>if</b> ((num3 > num2) && (num2 > num1))	1	2	4	<b>true</b>	<b>true</b>	
7.1	Console.WriteLine("The sequence of numbers is strongly increasing");	1	2	4			The sequence of numbers is strongly increasing
7.2	Console.WriteLine("The sequence of differences is ... ");	1	2	4			The sequence of differences is 1 2

## סוף פתרון קציה 5

**תנאי מורכב** הוא תנאי המורכב מקישור בין תנאים פשוטים (ביטויים בולאניים פשוטים). עד כה ראינו דרך אחת ליצור תנאי מורכב על ידי קישור תנאים פשוטים: באמצעות הקשר **/גם** (and). אשר משמעותו היא שהתנאי המורכב מתקיים רק כאשר שני התנאים הפשוטים מתקיימים ביחד בו-זמנית.

הטבלה הבאה מתארת את ערכיו של ביטוי בוליאני המורכב מקשר **/גם** בין שני ביטויים בולאניים. הטבלה נקראת **טבלת אמת של קשר /גם**:

ביטוי 1 \ ביטוי 2	false	true
false	false	false
true	false	true

כפי שניתן לראות בטבלה, ערכו של הביטוי הבוליאני המורכב הוא **true** (אמיתי, נכון) רק כאשר ערך שני הביטויים 1 ו-2, הוא **true**. בכל מקרה אחר, ערכו של הביטוי הבוליאני המורכב הוא **false** (שקרי, לא נכון).

הקשר /גס מיושם ב-C# בסימן הפעולה &&. יש להקיף בסוגריים את הביטויים הבוליאניים המקושרים ב-&&.

בטבלת מעקב אחר מהלך ביצוע תוכנית שבה משפט if הכולל ביטוי בוליאני מורכב, נקצה עמודה לכל אחד מן הביטויים הבוליאניים הפשוטים המרכיבים את הביטוי הבוליאני המורכב.

### שאלה 5.21

בתוכנית CheckSequence לפתרון בעיה 5 מופיע הביטוי הבוליאני המורכב הבא :

`((num3 > num2) && (num2 > num1))`

השלימו את הטבלה הבאה :

num1	num2	num3	num3 > num2	num2 > num1	(num3 > num2) && (num2 > num1)	פלט
1	2	2				
1	1	2				
1	1	1				
0	1	2				

### שאלה 5.22

השלימו את התנאי המורכב המתאים בכל אחת מן ההוראות הבאות :

- א. אם האור הראשונה שווה ארבעה גס  
האילה בן ארבע האותיות היא פלינדום
- ב. אם שנת הלידה שלך גדולה מ-1985 גס  
שנת הלידה שלך היא בין 1985 ל-1995
- ג. אם let >= 'A' גס  
let מיצג אור אנליט גדולה

### שאלה 5.23

כתבו את התנאים הבאים המנוסחים במילים, כביטויים בוליאניים :

- א. ערכו של המשתנה x גדול מ-0 וקטן מ-50.
- ב. ערכו של המשתנה let אינו התו 'a' ואינו התו 'z'.
- ג. הערך המוחלט של הפרשי ערכי המשתנים x ו-y גדול מערכו של x וקטן מערכו של y.

### שאלה 5.24

ביטוי בוליאני מורכב יכול לכלול יותר משני ביטויים בוליאניים פשוטים.

במשתנים temp1, temp2 ו-temp3 ערכים כלשהם.

- א. כתבו ביטוי בוליאני המבטא כי ערכי המשתנים temp1, temp2 ו-temp3 שונים זה מזה. האם נחוצים יותר משני ביטויים בוליאניים פשוטים לכתיבת הביטוי?
- ב. כתבו ביטוי בוליאני המבטא כי ערכי המשתנים temp1, temp2 ו-temp3 שווים זה לזה. האם נחוצים יותר משני ביטויים בוליאניים פשוטים לכתיבת הביטוי?

### שאלה 5.25

עבור כל אחד מן הביטויים הבוליאניים המורכבים הבאים, תנו דוגמה לערך של המשתנה `num` או `let` אשר עבורו יהיה ערכו של הביטוי הבוליאני `true`, ותנו דוגמה לערך אשר עבורו יהיה ערכו של הביטוי הבוליאני `false`.

ביטוי	true	false
<code>(num &gt;= 0) &amp;&amp; (num &lt;= 5)</code>		
<code>(num &gt; 0) &amp;&amp; (num != 1)</code>		
<code>(num &gt; 2) &amp;&amp; ((num % 2) == 0)</code>		
<code>(let != 'z') &amp;&amp; (let &gt; 'x')</code>		

### שאלה 5.26

פתחו אלגוריתם אשר הקלט שלו הוא שלושה תווים, והפלט שלו הוא התו העוקב לגדול מבין התווים, אם שלושת התווים הם תווים עוקבים ונתונים בסדר עולה. למשל עבור הקלט B C D יהיה הפלט E, ועבור הקלט A C D יהיה הפלט ריק (כלומר לא יוצג דבר, כיוון שהתווים אינם תווים עוקבים). ישמו את האלגוריתם בשפת התכנות C#. כתבו את חישוב התו העוקב כביטוי במשפט הפלט.

להעמקה בתבנית ערכים עוקבים? פנו לסעיף התבניות המופיע בסוף הפרק.

## הקשר //

בניתוח של איור 5.1 הזכרנו קשר נוסף – הקשר `//`. נדון כעת בשימוש בקשר זה בכתיבת אלגוריתמים וביישומם.

## הצ'י 6

מטרת הבעיה ופתרונה: הצגת תנאי מורכב הכולל את הקשר `//`.

פתחו אלגוריתם שהקלט שלו הוא שני מספרים שלמים חיוביים `do` ספרתיים, והפלט שלו הוא הודעה אם שני המספרים מורכבים מאותן ספרות. למשל, עבור הקלט: 19 91, הפלט יהיה הודעה שהמספרים מורכבים מאותן ספרות, ועבור הקלט 19 81, הפלט יהיה הודעה שהמספרים אינם מורכבים מאותן ספרות. ישמו את האלגוריתם בשפת התכנות C#.

### ניתוח הבעיה בעזרת דוגמאות

נבחן מספר דוגמאות קלט מגוונות אשר כוללות את המספר 91: עבור כל אחד מן הקלטים: 91 91, 91 19, 19 91, תוצג כפלט הודעה שהמספרים מורכבים מאותן ספרות. עבור כל קלט אחר הכולל את המספר 91, מלבד שלושת הקלטים האלה, תוצג כפלט הודעה שהמספרים אינם מורכבים מאותן ספרות.

מהתבוננות בשלושת הקלטים המתוארים ניתן לראות שתוצג כפלט הודעה שהמספרים מורכבים מאותן ספרות אם ורק אם המספר הנוסף ל-91 הוא 91 עצמו או שהוא 19, כלומר, המספר המתקבל מ-91 על ידי היפוך סדר הספרות.

### פירוק הבעיה לתת-משימות

נוכל לנסח רעיון ראשוני לפתרון: ראשית נפרק את המספר הראשון לספרותיו ונבנה את המספר המתקבל ממנו לאחר היפוך בסדר הספרות. כעת ניתן לבדוק אם המספר השני שווה למספר הראשון או למספר ההפוך לראשון. אם המספר השני שווה לאחד משניהם אז תוצג כפלט הודעה שהמספרים מורכבים מאותן ספרות.

נפרק לתת-משימות על פי הרעיון שהצענו:

1. קליטת שני מספרים שלמים חיוביים דו-ספרתיים
2. פירוק המספר הראשון לספרותיו
3. הרכבת מספר חדש שמתקבל מהמספר הראשון לאחר היפוך סדר הספרות
4. השוואת המספר השני למספר הראשון ולמספר החדש
5. הצגת הודעת פלט מתאימה

### בחירת משתנים

נבחר משתנים מטיפוס שלם על פי התת-משימות המתוארות:

- `num1` – לשמירת המספר הראשון
- `num2` – לשמירת המספר השני
- `tens` – לשמירת ספרת העשרות של `num1`
- `units` – לשמירת ספרת האחדות של `num1`
- `invNum1` – לשמירת המספר ההפוך בסדר ספרותיו ל-`num1`

### האלגוריתם

? כיצד נבנה את המספר ההפוך למספר הראשון בסדר ספרותיו?  
זוהי תבנית **בניית מספר** דו-ספרתי, המוכרת לנו מפרק 4: ספרת העשרות של המספר החדש היא ספרת האחדות של המספר הראשון. ספרת האחדות של המספר החדש היא ספרת העשרות של המספר הראשון. לכן יש להכפיל את ערכו של `units` ב-10 ולחבר למכפלה את ערכו של `tens`.

? בכתובת האלגוריתם עלינו לנסח תנאי אשר יתקיים כאשר `num2` יהיה שווה לאחד המספרים הנתונים ב-`num1` וב-`invNum1`. מהו התנאי המתאים?  
התנאי הוא: `num2 == num1` **או** `num2 == invNum1`  
תנאי זה הוא תנאי מורכב הכולל קישור בין שני תנאים באמצעות המילה **או**.

? האם ייתכן שיתקיימו בו-זמנית שני התנאים הפשוטים הכלולים בתנאי המורכב שניסחנו?  
כן, זה ייתכן. למשל אם הקלט הוא 11 11. במקרה כזה המספר השני שווה גם למספר הראשון, וגם למספר שמתקבל מהמספר הראשון בהיפוך סדר הספרות. גם במקרה כזה, אנחנו מעוניינים כמובן שהתנאי המורכב יתקיים. כלומר התנאי המורכב מתקיים כאשר `num2` שווה לאחד המספרים הנתונים ב-`num1` וב-`invNum1`, או לשניהם.



## יישום האלגוריתם

ב-C#, הקשר `//` נכתב באמצעות הסימן `||` (שני קווים אנכיים רצופים).

```
/*
קלט: 2 מספרים חיוביים שלמים
פלט: הודעה אם המספרים מורכבים מאותן ספרות
*/
using System;
public class DigitEquality {
    public static void Main()
    {
        // הצהרה על משתנים בתוכנית
        int num1;           // מספר ראשון
        int num2;           // מספר שני
        int tens;           // ספרת העשרות של המספר הראשון
        int units;          // ספרת האחדות של המספר הראשון
        int invNum1;        // המספר השני הפוך
        // קליטת המשתנים
1. Console.WriteLine("Enter first number: ");
2. num1 = int.Parse(Console.ReadLine());
3. Console.WriteLine("Enter second number: ");
4. num2 = int.Parse(Console.ReadLine());
        // חישוב ספרת האחדות והעשרות וחישוב המספר ההפוך
5. tens = num1 / 10;
6. units = num1 % 10;
7. invNum1 = units * 10 + tens;
        // בדיקה אם המספר השני שווה למספר הראשון או למספר ההפוך לו
8. if ((num2 == num1) || (num2 == invNum1))
8.1. Console.WriteLine("The numbers have the same digits");
9. else
9.1. Console.WriteLine("The numbers don't have the " +
        " same digits");

    } // Main
} // class DigitEquality
```

### סוף פתרון בעיה 6

דרך נוספת ליצור תנאי מורכב היא לקשר תנאים פשוטים באמצעות הקשר `//` (or). משמעותו היא שהתנאי המורכב מתקיים כאשר לפחות אחד משני התנאים הפשוטים מתקיים. הטבלה הבאה מתארת את ערכיו של ביטוי בוליאני המורכב מקשר `//` בין שני ביטויים בוליאניים. הטבלה נקראת טבלת אמת של קשר `//`:

ביטוי 1 \ ביטוי 2	false	true
false	false	true
true	true	true

כפי שניתן לראות בטבלה, ערכו של הביטוי הבוליאני המורכב הוא **true** כאשר לפחות ערך אחד משני הביטויים 1 או 2, הוא **true**. רק אם ערכי שני הביטויים הם **false** אז ערכו של הביטוי הבוליאני המורכב הוא **false**.

הקשר **!** מיושם ב-C# בסימן הפעולה **||**. יש להקיף בסוגריים את הביטויים הבוליאניים המקושרים ב-**||**.

## שאלה 5.27

בתוכנית DigitEquality לפתרון בעיה 6 מופיע הביטוי הבוליאני המורכב הבא:

```
(num1 == num2) || (num2 == invNum1)
```

ציינו עבור כל אחד מן הקלטים הבאים: מהו ערכו של הביטוי הפשוט השמאלי, מהו ערכו של הביטוי הפשוט הימני, מהו ערכו של הביטוי המורכב, ומהו הפלט במהלך ביצוע התוכנית.

קלט	num1 == num2	num2 == invNum1	(num1 == num2)    (invNum1 == num2)	פלט
25 56				
25 25				
25 52				
55 55				

## שאלה 5.28

השלימו את התנאי המורכב המתאים בכל אחת מן ההוראות הבאות:

- א. **אם האות הראשונה היא A אז** \_\_\_\_\_
  - ב. **אם זילך נמוך מ-18 אז** \_\_\_\_\_
  - ג. **אם  $num > 10$  אז** \_\_\_\_\_
- \_\_\_\_\_ **שומר ערך שהוא גדול מ-10 או קטן מ-5**

## שאלה 5.29

כתבו את התנאים הבאים המנוסחים במילים כביטויים בוליאניים.

- א. ערכו של המשתנה  $x$  חיובי או ערכו של המשתנה  $y$  הוא התו 'A'.
- ב. ערכו של המשתנה  $x$  קטן מ-1 או גדול מ-7.
- ג. ערכו של המשתנה  $x$  זוגי או מתחלק ב-3 ללא שארית.

להעמקה בתבנית **זוגיות מספר** פנו לסעיף התבניות המופיע בסוף הפרק.

להעמקה בתבנית **מחלק של?** פנו לסעיף התבניות המופיע בסוף הפרק.

## שאלה 5.30

במשתנים side1, side2 ו-side3 שמורים אורכי שלוש צלעות של משולש.

- א. כתבו ביטוי בוליאני המבטא שהמשולש שווה שוקיים (במשולש שווה שוקיים לפחות שתי צלעות שוות).

ב. כתבו ביטוי בוליאני המבטא שהמשולש ישר זווית (במשולש ישר זווית סכום ריבועי שני הניצבים שווה לריבוע היתר – משפט פיתגורס).

### שאלה 5.31

עבור כל אחד מן הביטויים הבוליאניים המורכבים הבאים, תנו דוגמה לערך של המשתנה `num` אשר עבורו יהיה ערכו של הביטוי הבוליאני `true`, ותנו דוגמה לערך אשר עבורו יהיה ערכו של הביטוי הבוליאני `false`.

ביטוי	ערכו של הביטוי false	ערכו של הביטוי true
<code>num != 0</code>	<code>num =</code>	<code>num =</code>
<code>(num &lt; 2)    (num &gt; 2)</code>	<code>num =</code>	<code>num =</code>
<code>(num &gt; 0)    (num == -5)</code>	<code>num =</code>	<code>num =</code>
<code>((num%2) == 0)    (num &lt; 0)</code>	<code>num =</code>	<code>num =</code>

### שאלה 5.32

לפעמים ניתן לצמצם ביטוי בוליאני מורכב לביטוי פשוט. בהנחה ש-`num` מייצג מספר שלם צמצמו כל אחד מן הביטויים המורכבים לביטוי פשוט (כלומר ללא שימוש בקשרים):

א. `(num < -1) || (num > 1)`  
 ב. `(num > -1) && (num < 1)`

### שאלה 5.33 (מבגרות 2003)

לפניכם קטע תוכנית:

```
a = int.Parse(Console.ReadLine());
b = int.Parse(Console.ReadLine());
if ( a < b || a < 100 )
    Console.WriteLine("The expression value: true");
else
    Console.WriteLine("The expression value: false");
```

בחרו במספר שייקלט ב-`a` ובמספר שייקלט ב-`b`, כך שיתקבל הפלט

The expression value: false

נמקו את בחירתכם.

### שאלה 5.34

פתחו אלגוריתם אשר הקלט שלו הוא תו. האלגוריתם בודק אם התו הוא תו חוקי לניחוש בטופס ספורטוטו (כלומר 1, 2 או x). האלגוריתם מציג הודעה מתאימה כפלט. ישמו את האלגוריתם בשפת C#.

### שאלה 5.35

בתחרות קליעת כדור לארגז, קולעים כדור לארגז שאורכו מטר אחד. תחילתו של הארגז היא במרחק 10 מטרים מן הקולע.

יש לפתח אלגוריתם אשר הקלט שלו הוא מרחק נפילת הכדור מהקולע והפלט שלו הוא הודעה אם הכדור נכנס לארגז. אם הכדור לא נכנס לארגז, יש לצרף לפלט גם את המרחק בין מקום נפילת הכדור למרכז הארגז.

למשל, עבור הקלט 10.3 הפלט הוא: נכנס.

ועבור הקלט 12 הפלט הוא: לא נכנס, החטאת את מרכז הארגז ב-1.5 מ'.

- נתחו תחילה את הבעיה (בעזרת דוגמאות קלט מייצגות) ובחרו משתנים. לאחר מכן:
- נסחו תנאי מורכב באמצעות קשר  $\wedge$  שיתקיים כאשר הכדור נכנס לארגז.
  - נסחו תנאי מורכב באמצעות קשר  $\vee$  שיתקיים כאשר הכדור לא נכנס לארגז.
  - נסחו את התנאי שבסעיף ב כתנאי לא מורכב.
  - כתבו את האלגוריתם לפתרון הבעיה (בחרו את אחד מהתנאים שבסעיפים א-ג), וישמו את האלגוריתם בתוכנית בשפת C#.

### שאלה 5.36

נתון הלוח הבא ובו עשר משבצות הממוספרות מ-1 עד 10:

10	9	8	7	6	5	4	3	2	1

במשתנה  $x$  שמור מספר שלם בין 1 ל-10 המבטא את מקום הכלי  $X$  על הלוח. במשתנה  $y$  שמור מספר בין 1 ל-10 המבטא את מקום הכלי  $Y$  על הלוח.

א. כתבו ביטוי בוליאני שערכו `true` אם הכלי  $X$  נמצא בחצי השמאלי של הלוח (כלומר על אחת מחמש המשבצות השמאליות).

ב. השלימו את תיאור המשמעות של קיום התנאי ואת הפלט המתאים במשפט ה-`if` הבא:

```
if (Math.Abs(x - y) == 1) // _____
    Console.WriteLine("_____");
else // _____
    Console.WriteLine("_____");
```

ג. הביטוי הבוליאני הבא אמור לבטא מצבים שבהם הכלי  $X$  נמצא מימין לכלי  $Y$  על הלוח:

$$x \neq y$$

הביטוי שגוי. תקנו אותו.

ד. כתבו ביטוי בוליאני פשוט (לא מורכב) שערכו `true` אם הכלי  $Y$  נמצא על משבצת שחורה. שימו לב שבביטוי עליכם לבטא את המשותף לחמש המשבצות השחורות.

ה. השלימו את תיאורי המשמעות של קיום התנאי ושל אי-קיומו ואת הפלט המתאים במשפט ה-`if` הבא:

```
if (((x-y) % 2) == 0) // _____
    Console.WriteLine("_____");
else // _____
    Console.WriteLine("_____");
```

ו. הביטוי הבוליאני הבא אמור לבטא מצבים שבהם שני הכלים נמצאים על משבצות בצבעים שונים:

$$((x * y) \% 2 == 0)$$

הביטוי נכון רק עבור חלק מן המקרים האפשריים. עבור אילו מקרים הביטוי נכון, ועבור אילו מקרים איננו נכון? כתבו ביטוי שיהיה נכון עבור כל המקרים האפשריים.

## תנאים מורכבים מעורבים

בעזרת הקשרים /**אם** ו-**א**/ ניתן ליצור ביטויים מורכבים אף יותר מאלו שראינו עד כה, אשר מערבים את שני הקשרים.

### שאלה 5.37

עבור כל אחד מן הביטויים הבוליאניים המורכבים הבאים, תנו דוגמה לערך של המשתנה num אשר עבורו יהיה ערכו של הביטוי הבוליאני true, ותנו דוגמה לערך אשר עבורו יהיה ערכו של הביטוי הבוליאני false.

ביטוי	ערכו של הביטוי false	ערכו של הביטוי true
$((num \neq 0) \ \&\& \ (num \geq 8)) \    \ (num == 3)$	num =	num =
$((num < 2) \    \ (num > 2)) \ \&\& \ ((num < 7) \    \ (num != 1))$	num =	num =
$(num == 0) \ \&\& \ ((num > 0) \    \ (num == -5))$	num =	num =

בדומה לקדימות המוגדרת ביחס לפעולות חשבוניות (כגון פעולת כפל קודמת לפעולת חיבור), מוגדרת גם קדימות ביחס לקשרים בוליאניים: הקשר && קודם לקשר ||. בכל זאת, לשם בהירות התוכנית ולשם קריאותה עדיף להשתמש בסוגריים כדי להבהיר את טווח הפעולה של כל קשר.

### שאלה 5.38 (מבגרות 2003)

נתון הביטוי הבוליאני:  $(z > x) \ || \ (x < 0) \ \&\& \ (z - y > 9)$   
מהו הערך של הביטוי עבור הנתונים:  $z = 13, y = 5, x = -2$ ? פרטו את כל שלבי החישוב.

## 5.3 קינון של הוראה לביצוע-בתנאי

בסעיף זה נראה כי לעתים נוח ליצור הוראה מורכבת לביצוע-בתנאי. זוהי הוראה לביצוע-בתנאי שאחת (או יותר) מההוראות היא עצמה הוראה לביצוע-בתנאי.

### קצ'ה 7

מטרת הבעיה ופתרונה: הצגת הוראה מקוננת לביצוע-בתנאי.

באולימפיאדת החיות מתקיימת תחרות ריצה למרחק 100 מטר. צבי אשר רץ 100 מטר בזמן של 10 שניות או פחות נחשב לצבי מהיר. צב שרץ 100 מטר בזמן של 10 דקות או פחות נחשב לצב מהיר.

פתחו אלגוריתם אשר הקלט שלו הוא סוג החיה, T (Turtle) עבור צב ו-D (Deer) עבור צבי, ומספר המציין זמן ריצה בשניות. האלגוריתם יציג הודעה אם החיה שנקלטה מהירה או לא. ישמו את האלגוריתם בשפת C#.

## פירוק הבעיה לתת-משימות

1. קליטת שם החיה
2. קליטת תוצאת הריצה בשניות
3. הצגת הודעה אם החיה מהירה או לא

## בחירת משתנים

**animalType** – משתנה מטיפוס תווי שישמור את שם החיה  
**scoreInSeconds** – משתנה מטיפוס שלם שישמור את תוצאת הריצה בשניות

## האלגוריתם

❓ כיצד נחליט אם החיה מהירה?

יש לבדוק תחילה אם החיה היא צבי או צב ואז להשוות את תוצאת הריצה שלה לזמן המגדיר צבי מהיר או צב מהיר, בהתאמה.

נכתוב זאת בצורה הבאה:

```
אם score <= 10 // 'D' צבי
השווה את גודל המידע ל-10 שניות
אחרת //
השווה את גודל המידע ל-600 שניות
```

תיארנו כאן מבנה של **אם...אחרת...** שהוא מבנה של ביצוע-בתנאי. אבל כדי לדעת אם הצבי מהיר ואם הצב מהיר יש לכלול בכל אחד מחלקי ההוראה לביצוע-בתנאי הוראה נוספת לביצוע-בתנאי. המבנה המתקבל הוא של **אם...אחרת...** בתוך **אם...אחרת...**, כלומר קינון של הוראות לביצוע-בתנאי.

נציג אלגוריתם לפתרון הבעיה, תוך שימוש בתנאי שניסחנו:

1. קלוט את שם החיה ב-**animalType**
2. קלוט את גודל המידע של החיה ב-**scoreInSeconds**
3. **אם animalType == 'D' // צבי**
  - 3.1. **אם scoreInSeconds <= 10**
    - 3.1.1. הצג הודעה כי הצבי מהיר
    - 3.2. אחרת
    - 3.2.1. הצג הודעה כי הצבי אינו מהיר
  4. אחרת // צב
- 4.1. **אם scoreInSeconds <= 600**
  - 4.1.1. הצג הודעה כי הצב מהיר
  - 4.2. אחרת
  - 4.2.1. הצג הודעה כי הצב אינו מהיר

שימו לב לאופן הזחת השורות (הזזתן פנימה, אינדנטציה) בכתיבה המקוננת של ההוראות לביצוע-בתנאי. אמנם אין השפה מחייבת זאת, אך מומלץ מאוד להקפיד על כך, משום שכך נוח לשייך כל **אחרת...ל-אם** המתאים. בכך אנו הופכים את התוכנית לקריאה ולברורה יותר.

## יישום האלגוריתם

```
/*
קלט: תו המזהה חיה ותוצאת ריצתה למאה מטרים
פלט: הודעה המציינת אם החיה מהירה או לא
*/
using System;
public class AnimalOlympics
{
    public static void Main()
    {
        // הצהרה על משתנים בתוכנית
        char animalType;
        int scoreInSeconds;
        const int DEER_LIMIT = 10;
        const int TURTLE_LIMIT = 600;
        // קליטת המשתנים
1. Console.WriteLine("Enter the animal type - D for a deer and " +
                        "T for a turtle: ");
2. animalType = char.Parse(Console.ReadLine());
3. Console.WriteLine("Enter the animal score in seconds: ");
4. scoreInSeconds = int.Parse(Console.ReadLine());
5. if (animalType == 'D')    // אם צבי
    {
5.1.     if (scoreInSeconds <= DEER_LIMIT) // האם מהיר לפי ההגדרה
                                                // המתאימה לצבי
5.1.1.         Console.WriteLine("The deer is fast");
5.2.     else
5.2.1.         Console.WriteLine("The deer is not fast");
    } // if animalType
6.     else // צב
    {
6.1.         if (scoreInSeconds <= TURTLE_LIMIT) // האם מהיר לפי
                                                // ההגדרה המתאימה לצב
6.1.1.             Console.WriteLine("The turtle is fast");
6.2.         else
6.2.1.             Console.WriteLine("The turtle is not fast");
    } // else animalType
    } // Main
} // class AnimalOlympics
```

### שאלה 5.39

בנו טבלת מעקב אחר ביצוע התוכנית AnimalOlympics עבור הקלט T 700 ועבור הקלט D 5. חשבו לאילו תחומים במבנה המקוון של משפטי ה-if בתוכנית, ינותב מהלך הביצוע עבור כל סוג קלט אפשרי?

### סוף פתרון בעיה 7

לצורך פתרון בעיה 7 היה מתאים להשתמש במבנה מקוון של הוראות לביצוע-בתנאי. ההוראה לביצוע-בתנאי שתלויה בסוג החיה (ובתוכנית תלויה במשתנה animalType) היא הוראה לביצוע-בתנאי **חיצונית** (בתוכנית היא יושמה במשפט if חיצוני). הוראה זו מכילה שתי

הוראות לביצוע-בתנאי **פנימיות** (בתוכנית – שני משפטי `if` פנימיים): האחת מוכלת בתחום ה-`סק` של ההוראה החיצונית והשנייה מוכלת בתחום ה-`אגרא` של ההוראה החיצונית. ההוראות הפנימיות הן אלו שביצוען תלוי בתוצאת הריצה (ובתוכנית תלוי בערך המשתנה `scoreInSeconds`).

לעתים, כאשר יש לנתב את מהלך הביצוע של אלגוריתם לאחת מבין שלוש או יותר אפשרויות התלויות במספר תנאים, מתאים להשתמש במבנה מקונן של הוראות לביצוע-בתנאי.

**מבנה מקונן (nesting) של ביצוע-בתנאי** (משפטי `if`) כולל הוראה לביצוע-בתנאי, אשר אחת או יותר מבין ההוראות הפנימיות היא בעצמה הוראה לביצוע-בתנאי. למשל המבנה הבא הוא מבנה מקונן של ביצוע-בתנאי:

<b>ב-#C:</b> <pre> if ( . . . ) {     . . .     if ( . . . )     { . . . }     else     { . . . }     . . . } else {     . . .     if ( . . . )     { . . . }     else     { . . . }     . . . } </pre>	<b>בכתיבה אלגוריתמית:</b> <pre> ...סק ... ...סק ... אגרא ... ... אגרא ... ...סק ... אגרא ... ... </pre>
--	--

בכתיבת מבנה מקונן נקפיד על הזחות מתאימות. באלגוריתם ייכתב כל `אגרא` מתחת ל-`סק` המתאים לו. ב-`#C` ייכתב כל `else` מתחת ל-`if` המתאים לו.

**שימו ♥:** ההגדרה של קינון הוראות לביצוע-בתנאי היא כללית למדי, ולכן ייתכנו מבנים מקוננים בצורות שונות. כל אחת מההוראות הפנימיות של הוראה לביצוע-בתנאי יכולה להיות בעצמה הוראה לביצוע-בתנאי. לכן ייתכן למשל משפט `if` שתחום ה-`if` שלו מכיל שתי הוראות לביצוע-בתנאי.

בנוסף, הוראה פנימית לביצוע-בתנאי יכולה להיות הוראה לביצוע-בתנאי מכל סוג שהוא – במבנה `אס... אגרא... אס...`, במבנה `אס... אס...` או אפילו הוראה לביצוע-בתנאי מקוננת בעצמה.

לכן מהרגע שקינון מצטרף למשחק, המבנים המתקבלים יכולים להיות מורכבים מאוד. משום כך, חשוב לזכור ולהקפיד על הערות המתארות משמעות של קיום תנאי ושל אי-קיומו. הערות אלו מסייעות לנו להבין מבנה מקונן של הוראות לביצוע-בתנאי. הערה המתארת משמעות של **קיום** תנאי במשפט `if` מבהירה בשפה ברורה וקריאה את הסיבה לניתוב מהלך הביצוע של משפט ה-`if` אל תחום ה-`if` שלו. בדומה, הערה המתארת משמעות של **אי-קיום** תנאי מבהירה בשפה ברורה וקריאה את הסיבה לניתוב מהלך הביצוע של משפט ה-`if` אל תחום ה-`else` שלו.



חשוב לתעד באופן הזה משפטי `if` שאינם מקוננים, כפי שעשינו בתוכניות שהוצגו בפרק עד כה. חשוב עוד יותר לתעד מבנים מקוננים.

הנה שתי דוגמאות למבנים מקוננים שונים: בכל אחד משני המבנים הבאים מחושב יחס (קטן מ, גדול מ, שווה ל) בין ערכי שני משתנים (`var1` ו-`var2`).

### מבנה I:

```
if (var1 > var2)
    Console.WriteLine("var1 > var2");
else
{
    if (var1 == var2)
        Console.WriteLine("var1 = var2");
    else
        Console.WriteLine("var1 < var2");
}
```

### מבנה II:

```
if (var1 >= var2)
{
    if (var1 == var2)
        Console.WriteLine("var1 = var2");
    else
        Console.WriteLine("var1 > var2");
}
else
    Console.WriteLine("var1 < var2");
```

שני המבנים כוללים הוראה אחת והיא הוראה לביצוע-בתנאי. במבנה I תחום ה-`else` של הוראה זו מכיל הוראה אחת וגם היא הוראה לביצוע-בתנאי במבנה `אם... אחר... גם במבנה II` תחום ה-`if` מכיל הוראה לביצוע-בתנאי במבנה `אם... אחר... .`

כאמור, הקפדה על הזחות מתאימות למשפטי `if` בתוכנית בשפת `C#` מיועדת לשמירה על קריאותה. המהדר של שפת `C#` אינו מייחס חשיבות להזחות. הוא אינו משייך תחום `else` למשפט ה-`if` המתאים לו על פי ההזחות. השייך נקבע על פי הכלל הבא, שיודגם מיד:

### כלל השיוך של `else` ל-`if` מתאים:

`else` תמיד משויך ל-`if` אשר נמצא בתוכנית לפניו וקרוב אליו ביותר בתנאים הבאים:

1. ל-`if` זה לא משויך `else` אחר קרוב יותר.
2. `if` זה איננו כלול בתחום סוגריים מסולסלים אחר שמסתיים עוד לפני ה-`else`.

כלל זה הוא מורכב למדי. הנה כמה דוגמאות שיסייעו בהבנתו:

- ♦ במבנה I שלעיל כל `else` משויך ל-`if` שנמצא לפניו וקרוב אליו ביותר.
- ♦ במבנה II שלעיל ה-`else` הראשון שייך ל-`if` אשר נמצא בתוכנית לפניו וקרוב אליו ביותר. לעומת זאת ה-`else` השני אינו משויך ל-`if` שנמצא לפניו וקרוב אליו ביותר, אלא משויך ל-`if` הראשון (החיצוני). זאת מכיוון שה-`if` הקרוב אליו ביותר כבר שויך ל-`else` אחר. לכן ה-`else` השני משויך ל-`if` הבא הקרוב ביותר, הלוא הוא ה-`if` הראשון.
- ♦ גם במבנה המקוון הבא ה-`else` איננו משויך ל-`if` שנמצא לפניו וקרוב ביותר אליו (ה-`if` השני). הפעם, מכיוון ש-`if` זה כלול בתחום סוגריים מסולסלים שמסתיים עוד לפני ה-`else`. לכן ה-`else` משויך ל-`if` הראשון (החיצוני) ולא ל-`if` השני (הפנימי).

```

if (var1 >= var2)
{
    if (var1 == var2)
        Console.WriteLine("var1 = var2");
}
else
    Console.WriteLine("var1 < var2");

```

#### שאלה 5.40

בכל אחד מהסעיפים הבאים נתון קטע תוכנית. כמו כן, בכל סעיף נתונות כמה אפשרויות לערכים התחלתיים של משתני הקטע. תארו את הפלטים של הקטעים הבאים, עבור כל אחד מהערכים ההתחלתיים של משתני הקטע.

א.

```

if (num > 0)
    Console.WriteLine("+");
else
{
    if (num == 0)
        Console.WriteLine("0");
    else
        Console.WriteLine("-");
}

```

פלט	ערך התחלתי של num
	0
	10
	-10

ב.

```

if (num1 > 0)
{
    if (num2 > 0)
        Console.WriteLine("+");
    else
        Console.WriteLine("+");
}
else
    Console.WriteLine("-");

```

פלט	ערך התחלתי של num2	ערך התחלתי של num1
	0	-1
	5	-5
	5	-1

ג.

```

if (num1 < 0)
{
    if (num2 < 0)
        Console.WriteLine(num1 * num2);
}
else
{
    Console.WriteLine(num1);
    if (num2 < 0)
        Console.WriteLine(num1 + num2);
}

```

פלט	ערך התחלתי של num2	ערך התחלתי של num1
	1	-1
	-1	-1
	0	0

### שאלה 5.41

הקלט בכל אחד מקטעי התוכניות הבאים הוא אות מן הא"ב האנגלי. מטרת כל אחד מקטעי התוכניות היא להציג כפלט הודעה אם האות הנקלטת היא האות N, אחת האותיות שקודמות ל-N בא"ב האנגלי או אחת האותיות שמופיעות אחרי N בא"ב האנגלי. השלימו את קטעי התוכניות:

א. קטע תוכנית א

```
Console.Write("Enter a letter: ");
letter = char.Parse(Console.ReadLine());
if (letter == 'N')
{
    .
    .
    .
}
```

ב. קטע תוכנית ב

```
Console.Write("Enter a letter: ");
letter = char.Parse(Console.ReadLine());
if (letter != 'N')
{
    .
    .
    .
}
```

---

כפי שמראה התרגיל הבא, ניתן לעתים להמיר הוראה מקוננת לביצוע-בתנאי בהוראה לביצוע-בתנאי שאינה מקוננת אך כוללת תנאים מורכבים. עם זאת במקרים מסוימים השימוש בקינון הופך את התוכנית לקריאה ולברורה יותר.

---

### שאלה 5.42

המירו את הפתרון לבעיה 7 AnimalOlympics לפתרון הכולל תנאים מורכבים, ואינו כולל קינון.

### שאלה 5.43

קבוצת חייזרים ממאדים או מנוגה תגיע לבקר בכדור הארץ. יש לכתוב אלגוריתם המברך אותם, על פי כללי הטקס המסובכים הנהוגים בחלל.

שמה של קבוצת החייזרים ממאדים הוא תו כלשהו. לעומתה, שמה של קבוצת החייזרים מנוגה הוא מספר כלשהו. פתחו אלגוריתם הקולט מהיכן הגיעה הקבוצה: V: (Venus) מנוגה ו-M: (Mars) ממאדים. אם הקבוצה היא מנוגה, יש לקלוט את שמה (מספר) ואם הוא גדול מ-10 להציג את ההודעה "hello" ואחרת להציג את ההודעה "hi". אם הקבוצה היא ממאדים יש להציג את ההודעה "have a nice day".

ישמו את האלגוריתם בשפת C#.

האם ניתן לכתוב אלגוריתם לשאלה הנעזר בתנאים מורכבים ואינו משתמש בקינון של הוראות לביצוע-בתנאי, כפי שנעשה בתרגיל 5.42? אם כן, הראו כיצד. אם לא, הסבירו מדוע.

### שאלה 5.44

נתון קטע התוכנית הבא, כאשר let1, let2 ו-let3 הם משתנים מטיפוס תווי השומרים אותיות מן הא"ב האנגלי:

```
if ((let1 == let2) || (let2 == let3) || (let1 == let3))
{
    if ((let1 == let2) && (let2 == let3))
        Console.WriteLine("1");
}
```

```

else
    Console.WriteLine("2");
}
else
    Console.WriteLine("3");

```

א. הביאו דוגמת קלט שהפלט עבורה הוא 1, דוגמת קלט שהפלט עבורה הוא 2, ודוגמת קלט שהפלט עבורה הוא 3.

ב. צרפו תיאורי משמעות של קיום התנאים ושל אי-קיומם לתחום ה-`if` ולתחום ה-`else` שבמבנה המקוץ.

ג. מהי מטרת קטע התוכנית?

#### שאלה 5.45

נתונים שני קטעי התוכניות הבאים, ובשניהם `num` הוא מטיפוס שלם:

<pre> א. if (num &lt; 0)     Console.WriteLine("-1"); else {     if (num == 0)         Console.WriteLine("0");     else //2_____         Console.WriteLine("1"); } </pre>	<pre> ב. if (num &lt; 0)     Console.WriteLine("-1"); if (num == 0)     Console.WriteLine("0"); else //1_____     Console.WriteLine("1"); </pre>
---	--

א. צרפו תיאורי משמעות לאי-קיום תנאי במקומות המסומנים

ב. האם שני קטעי התוכניות שקולים (כלומר, עבור כל קלט נתון יתקבל בשניהם אותו הפלט)? אם כן, הסבירו מדוע. אם לא, הביאו דוגמת קלט אשר עבורה יתקבלו פלטים שונים.

## 5.4 הוראת שרשרת לביצוע-בתנאי

לעתים נוח לכתוב הוראה לביצוע-בתנאי המתאימה לשרשרת של תנאים, שצריכים להיבדק זה אחר זה. בסעיף זה נתמקד בהוראות כאלו, במבנה `אם... אז... אחרת...`

### תצורה 8

מטרת הבעיה ופתרונה: הצגת הוראת שרשרת לביצוע-בתנאי במבנה `אם... אז... אחרת...`

פתחו וישמו אלגוריתם שהקלט שלו הוא מספר המציין ציון במבחן והפלט שלו הוא הציון המילולי בתעודה, על פי המפתח הבא: כל ציון בין 90 ל-100 במבחן זוכה לציון A בתעודה. כל ציון בין 80 ל-89 במבחן זוכה לציון B בתעודה. כל ציון בין 70 ל-79 במבחן מקבל את הציון C בתעודה. כל ציון בין 60 ל-69 במבחן מקבל את הציון D בתעודה וכל ציון נמוך מ-60 במבחן מקבל את הציון F בתעודה.

## פירוק הבעיה לתת-משימות

1. קליטת הציון במבחן
2. חישוב הציון המילולי בתעודה
3. הצגה כפלט של הודעה הכוללת את הציון המילולי בתעודה

## בחירת משתנים

score – משתנה מטיפוס שלם לשמירת ציון המבחן  
grade – משתנה מטיפוס תווי לשמירת הציון המילולי בתעודה

## האלגוריתם

? בכתובת האלגוריתם עלינו לנסח תנאי אשר יחשב את ציון התעודה המתאים לציון המבחן.  
מה יהיה התנאי הנחוץ?

הציון המילולי בתעודה מוכתב על ידי כמה תנאים :

1. ציון התעודה הוא A אם מתקיים התנאי : הציון במבחן גדול או שווה ל-90.
2. ציון התעודה הוא B אם מתקיים התנאי : הציון במבחן קטן מ-90 אבל גדול או שווה ל-80.  
זהו תנאי מורכב, שמשמעותו : תנאי 1 לא מתקיים וגם ציון המבחן גדול או שווה ל-80.
3. ציון התעודה הוא C אם מתקיים התנאי : הציון במבחן קטן מ-80 אבל גדול או שווה ל-70.  
זהו תנאי מורכב, שמשמעותו : תנאים 1 ו-2 לא מתקיימים וגם ציון המבחן גדול או שווה ל-70.
4. ציון התעודה הוא D אם מתקיים התנאי : הציון במבחן קטן מ-70 אבל גדול או שווה ל-60.  
זהו תנאי מורכב, שמשמעותו : תנאים 1, 2 ו-3 לא מתקיימים וגם ציון המבחן גדול או שווה ל-60.
5. ציון התעודה הוא D אם מתקיים התנאי : הציון במבחן קטן מ-60.

נוכל לבטא את ההתניה הזאת בשרשרת מורכבת של משפטי תנאי מקוננים, או בשימוש בתנאים מורכבים (ראו תרגילים בהמשך). לחילופין ניתן להיעזר בהוראה במבנה **אם... אז... אחרת...** שיוצרת מבנה ברור ופשוט למדי.

נציג אלגוריתם לפתרון הבעיה, שמשמש בהוראה לביטוי שרשרת התנאים שניסחנו :

1. קלוט את ציון המבחן score-2
2. אם  $\text{score} \geq 90$
- 2.1. השם grade-2 הוא 'A'
3. אחרת אם  $\text{score} \geq 80$
- 3.1. השם grade-2 הוא 'B'
4. אחרת אם  $\text{score} \geq 70$
- 4.1. השם grade-2 הוא 'C'
5. אחרת אם  $\text{score} \geq 60$
- 5.1. השם grade-2 הוא 'D'
6. אחרת
- 6.1. השם grade-2 הוא 'F'

## יישום האלגוריתם

/\*

קלט: ציון במבחן, בין 0 ל-100

```

פלט: ציון תעודה מילולי מתאים
*/
using System;
public class TermGrade
{
    public static void Main()
    {
        // הצהרה על משתנים בתוכנית
        int score;
        char grade;
        // קליטת המשתנים
1. Console.Write("Enter test score: ");
2. score = int.Parse(Console.ReadLine());
3. if (score >= 90)
3.1.     grade = 'A';
4. else if (score >= 80)
4.1.     grade = 'B';
5. else if (score >= 70)
5.1.     grade = 'C';
6. else if (score >= 60)
6.1.     grade = 'D';
7. else
7.1.     grade = 'F';
8. Console.WriteLine("Grade = {0}", grade);
    } // Main
} // class TermGrade

```

## מעקב

נעקוב אחר מהלך ביצוע התוכנית TermGrade עבור הקלט 76:

מספר שורה	המשפט לביצוע	score	grade	score>=?	פלט
1	Console.Write("Enter test score: ");	?	?		Enter test score:
2	score = int.Parse(Console.ReadLine());	76	?		
3	if (score >= 90)	76	?	false	
4	else if (score >= 80)	76	?	false	
5	else if (score >= 70)	76	?	true	
5.1	grade = 'C';	76	C		
8	Console.WriteLine("Grade = {0}", grade);				Grade = C

## סוף פתרון בעיה 8

בפתרון בעיה 8 מהלך הביצוע של האלגוריתם נקבע על פי שרשרת של תנאים. לצורך כך, השתמשנו במבנה **אם...אז...אז...** של הוראות לביצוע-בתנאי.

**הוראת שרשרת לביצוע-בתנאי** כוללת סדרה של תנאים ושל הוראות לביצוע עבור כל תנאי. משמעות ההוראה היא כי התנאים נבדקים לפי הסדר עד שאחד מהם מתקיים. ברגע שתנאי מסוים מתקיים, מתבצעות ההוראות שהוגדרו עבורו, ושאר התנאים לא נבדקים.

הוראת שרשרת לביצוע-בתנאי (בצד ימין בכתיבה אלגוריתמית ובצד שמאל ב-C#):

<code>if &lt;תנאי 1&gt;</code>	<code>אם &lt;תנאי 1&gt;</code>
<code>{ . . . }</code>	<code>&lt;קבוצת הוראות 1&gt;</code>
<code>else if &lt;תנאי 2&gt;</code>	<code>אחרת אם &lt;תנאי 2&gt;</code>
<code>{ . . . }</code>	<code>&lt;קבוצת הוראות 2&gt;</code>
<code>.</code>	<code>.</code>
<code>.</code>	<code>.</code>
<code>else if &lt;תנאי k&gt;</code>	<code>אחרת אם &lt;תנאי k&gt;</code>
<code>{ . . . }</code>	<code>&lt;קבוצת הוראות k&gt;</code>
<code>else</code>	<code>אחרת</code>
<code>{ . . . }</code>	<code>&lt;קבוצת הוראות k+1&gt;</code>

#### שאלה 5.46

פתחו אלגוריתם המחשב את דרגת החוכמה של תוכים, הנקבעת באופן הבא: אם התוכי יודע לומר יותר מ-10 מילים הוא תוכי חכם מאוד. אם התוכי יודע לומר בין 6 ל-10 מילים הוא תוכי חכם. אם התוכי יודע לומר בין מילה אחת ל-5 מילים הוא תוכי ממוצע. אם התוכי לא יודע לומר אף מילה הוא תוכי שתקן. האלגוריתם מקבל כקלט את מספר המילים השונות שתוכי יודע להגיד, ומציג הודעה המציינת את דרגת החוכמה של התוכי. ישמו את האלגוריתם בשפת C#.

#### שאלה 5.47

פתחו אלגוריתם אשר הקלט שלו הוא מקדמים של משוואה ריבועית,  $a$ ,  $b$  ו- $c$ , והפלט הוא מספר הפתרונות הממשיים של המשוואה הריבועית (0 פתרונות, פתרון אחד או 2 פתרונות) והפתרונות עצמם. ישמו את האלגוריתם בשפת C#.

$$X_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad \text{להזכירכם, הנה הנוסחה לחישוב פתרונותיה של משוואה ריבועית:}$$

#### שאלה 5.48

נתונים שני קטעי התוכניות הבאים אשר בכל אחד מהם `num` הוא מטיפוס שלם. האם שני קטעי התוכנית שקולים (כלומר, עבור כל קלט נתון יתקבל בשניהם אותו הפלט?) אם כן, הסבירו מדוע. אם לא, הביאו דוגמת קלט אשר עבורה יתקבלו פלטים שונים.

א.	ב.
<pre> if (num &lt; 0)     Console.WriteLine("-1"); else {     if (num == 0)         Console.WriteLine("0");     else         Console.WriteLine("1"); } </pre>	<pre> if (num &lt; 0)     Console.WriteLine("-1"); else if (num == 0)     Console.WriteLine("0"); else     Console.WriteLine("1"); </pre>

במובן מסוים, המבנה `אם... אחרת אם... אחרת` אינו חיוני. כפי שמראים התרגילים הבאים ניתן להסתדר גם בלעדיו. אבל השימוש בו מקל לעתים על הכתיבה, ויכול להפוך את התוכנית לקריאה יותר.

#### שאלה 5.49

המירו את האלגוריתם שבפתרון בעיה 8, לאלגוריתם שמשמש בהוראה לביצוע-בתנאי במבנה מקונן במקום בהוראת שרשרת לביצוע-בתנאי.

#### שאלה 5.50

המירו את האלגוריתם שבפתרון בעיה 8, לאלגוריתם שמשמש בתנאים מורכבים במקום בהוראת שרשרת לביצוע-בתנאי.

## 5.5 הוראת בחירה

במקרים רבים ערכו של ביטוי מסוים קובע את המשך המשימה. כלומר יש לבצע תת-משימה שונה עבור כל ערך אפשרי לביטוי. בכתובת אלגוריתם מתאים למשימה כזאת נוח להשתמש בהוראת בחירה שנכיר בסעיף זה.

### הצ'יף 9

מטרת הבעיה ופתרונה: הצגת הוראת בחירה המיושמת ב-C# במשפט `switch`.

פתחו וישמו אלגוריתם אשר מדמה מחשבון פשוט מאוד. הקלט הוא מספר ממשי, אחריו אחד מסימני הפעולה +, -, \* או /, ואחריו מספר ממשי נוסף. הפלט הוא תוצאת החישוב של הפעולה על שני המספרים הנתונים.

למשל, עבור הקלט:  $3.4 + 1.1$  יהיה הפלט  $2.3$   $1.1 + 3.4 = 2.3$

עבור הקלט:  $4.8 / 1.2$  יהיה הפלט  $4$   $4.8 / 1.2 = 4$

### פירוק הבעיה לתת-משימות

- קליטת נתוני הקלט
- ביצוע החישוב
- הצגת תוצאת החישוב כפלט.

### בחירת משתנים

- `num1` – משתנה מטיפוס ממשי, לשמירת המספר הראשון הניתן כקלט.
- `num2` – משתנה מטיפוס ממשי, לשמירת המספר השני הניתן כקלט.
- `operator` – משתנה מטיפוס תווי, לשמירת סימן הפעולה הניתן כקלט.
- `result` – משתנה מטיפוס ממשי, לשמירת תוצאת החישוב.

### האלגוריתם

? התת-משימה המשמעותית היא התת-משימה השנייה. בתת-משימה זו יש לבצע אחד מארבעה חישובים: חיבור, חיסור, כפל או חילוק, לפי סימן הפעולה הנקלט. כיצד ננסח זאת?

דרך אחת לניסוח החישוב היא באמצעות המבנה המקונן הבא:

אם סימן הפעולה הוא '+'  
אז אל שני המספרים



אגרת

אם סימן הפעולה הוא '+'

גסר את המספר השני מן הראשון

אגרת

אם סימן הפעולה הוא '\*'

הכפל את שני המספרים

אגרת

גזק את המספר הראשון בשני

דרך אחרת לניסוח החישוב היא באמצעות הוראת שרשרת:

אם סימן הפעולה הוא '+'

גבר את שני המספרים

אגרת אם סימן הפעולה הוא '-'

גסר את המספר השני מן הראשון

אגרת אם סימן הפעולה הוא '\*'

הכפל את שני המספרים

אגרת

גזק את המספר הראשון בשני

במקרה זה, שני המבנים הם מסורבלים יחסית. נוח יותר לכתוב הוראה המפרטת חלופות שונות בהתאם לערכו של סימן הפעולה, ועבור כל חלופה מנוסחת הוראת פעולה מתאימה.

ננסח את הפירוט של החלופות השונות באמצעות הוראת הבחירה הבאה:

**3.3 עזר הערך המתאים לסימן הפעולה:**

':+' גבר את שני המספרים

':-' גסר את המספר השני מן הראשון

':\*': הכפל את שני המספרים

':/': גזק את המספר הראשון בשני

**שימו ♥ :** הפעולות חיבור, חיסור וכפל ניתנות לביצוע עבור כל זוג מספרים ממשיים, אך בפעולת חילוק יש להיזהר – אין לחלק ב-0. אם המחלק הוא 0, אין לבצע חלוקה, אלא רק להציג כפלט הודעת שגיאה על ניסיון חלוקה ב-0. נכלול זאת באלגוריתם המלא.

## יישום האלגוריתם

הוראת בחירה מיושמת ב-C# במשפט `switch`, כפי שניתן לראות בתוכנית המלאה ליישום האלגוריתם שכתבנו:

```
/*
    התוכנית מדמה מחשבון לפעולות +, -, *, /
*/
using System;
public class Calculator
{
    public static void Main()
    {
        // הצהרה על משתנים בתוכנית
        double num1; // operand 1
        double num2; // operand 2
```

```

char operator;        // operator
double result;
// קליטת המשתנים
1. Console.WriteLine("Enter the first number: ");
2. num1 = double.Parse(Console.ReadLine());
3. Console.WriteLine("Enter the operator: ");
4. operator = char.Parse(Console.ReadLine());
5. Console.WriteLine("Enter the second number: ");
6. num2 = double.Parse(Console.ReadLine());
7. switch (operator)
{
7.1.     case '+':
7.1.1.         result = num1 + num2;
7.1.2.         Console.WriteLine("{0} + {1} = {2}", num1, num2,
                                result);
           break;
7.2.     case '-':
7.2.1.         result = num1 - num2;
7.2.2.         Console.WriteLine("{0} - {1} = {2}", num1, num2,
                                result);
           break;
7.3.     case '*':
7.3.1.         result = num1 * num2;
7.3.2.         Console.WriteLine("{0} * {1} = {2}", num1, num2,
                                result);
           break;
7.4.     case '/':
7.4.1.         if (num2 != 0)
           {
7.4.1.1.             result = num1 / num2;
7.4.1.2.             Console.WriteLine("{0} / {1} = {2}", num1, num2,
                                    result);
           }
7.4.2.         else
7.4.2.1.             Console.WriteLine("Division by 0");
           break;
7.5.     default: Console.WriteLine("Illegal operator");
           break;
} // switch
} // Main
} // class Calculator

```

אם כך, בהוראת בחירה בוחנים ערך של ביטוי (במקרה זה, הביטוי הפשוט operator), ומשווים אותו לחלופות השונות. ביישום בשפת C# מקדימה המילה case את החלופות השונות.

**שימו ⚡:** ה-case המתאים לחלופה שנבחרה, מהווה למעשה את נקודת הכניסה למשפט ה-switch. ממנו מתחילות להתבצע כל ההוראות שבתוך המשפט, זו אחר זו.

בסיום כל case שמפורטות בו פעולות לביצוע עלינו להוסיף את ההוראה break אשר מורה על יציאה ממשפט ה-switch ועל סיום ביצועו. אם ברצוננו לבצע אותן פעולות עבור case שונים, ניתן לכתוב את ה-case השונים בזה אחר זה, ורק עבור האחרון לכתוב את הפעולות לביצוע, ואחריהן להוסיף את ההוראה break. תרגיל 5.52 מדגים זאת.

קבוצת ההוראות הצמודה למילה `default` מתאימה לטיפול בערכים שאינם מטופלים באף חלופה, כלומר כאשר ערכו של הביטוי אינו שווה לאף אחד מהערכים המפורטים במשפט. אין חובה לכלול אפשרות `default` במשפט ה-`switch`. במקרה כזה, אם אין התאמה בין ערך הביטוי לאף אחד מהערכים המפורטים במשפט לא מתבצע דבר. גם בסיום חלק ה-`default` עלינו להוסיף את ההוראה `break`.

## המעקב

המעקב אחר ביצוע משפט `switch` דומה למעקב אחר ביצוע משפט `if`. כלומר בשורה המתאימה בטבלת המעקב נרשום את המשפט: `switch case` לביצוע, ובשורה שאחריה נרשום את המשפט הנבחר לביצוע. הנה טבלת המעקב אחר מהלך ביצוע התוכנית Calculator עבור הקלט `5 / 2`:

מספר שורה	המשפט לביצוע	num1	num2	operator	result	פלט
1	<code>Console.WriteLine("Enter the first number: ");</code>	?	?	?	?	Enter the first number
2	<code>num1 = double.Parse(Console.ReadLine());</code>	5	?	?	?	
3	<code>Console.WriteLine("Enter the operator: ");</code>	5	?	?	?	Enter the operator
4	<code>op = char.Parse(Console.ReadLine());</code>	5	?	'/'	?	
5	<code>Console.WriteLine("Enter the second number: ");</code>	5	?	'/'	?	Enter the second number
6	<code>num1 = double.Parse(Console.ReadLine());</code>	5	2	'/'	?	
7	<code>switch case: /</code>	5	2	'/'	?	
7.4.1	<code>if (num2 != 0)</code>	5	2	'/'	?	
7.4.1.1	<code>result = num1 / num2;</code>	5	2	'/'	2.5	
7.4.1.2	<code>Console.WriteLine("{0} / {1}={2}", num1, num2, result);</code>	5	2	'/'	2.5	5/2=2.5

## סוף פתרון תרגיל 9

### שאלה 5.51

בנו טבלת מעקב אחר ביצוע התוכנית Calculator עבור הקלט `5.3 * 0`.

כאשר עלינו לנתב את מהלך הביצוע על פי בחירה באחת מכמה אפשרויות של ערכים בדידים מתאים יותר להשתמש בהוראת בחירה מאשר בהוראת שרשרת לביצוע-בתנאי.

המבנה הכללי של הוראת בחירה הוא:

**עבור הערך המתאים ל ביטוי עצום:**

ערך 1: הוראה-1

ערך 2: הוראה-2

...

ערך k: הוראה-k

בביצוע ההוראה מחושב תחילה ערכו של הביטוי ולפיו מבוצעת החלופה המתאימה (שיכולה להיות הוראה מורכבת בפני עצמה).

הוראת בחירה מיושמת ב-C# במשפט **switch**

המבנה הכללי של משפט **switch** הוא:

```
switch (ביטוי)
{
    case ערך: משפט; break;
    case ערך: משפט; break;
    .
    .
    .
    default: משפט; break;
}
```

תפקיד **משפט ה-break** הוא לגרום לסיום ביצוע הוראת ה-switch.

תפקיד ה-**default** (ברירת מחדל) הוא לספק הוראות שיתבצעו במקרה שערכו של הביטוי אינו שווה לאף אחת מן החלופות. אין חובה לכלול טיפול ברירת מחדל.

## שאלה 5.52

בקטע התוכנית הבא יש שימוש ב-break רק בחלק מהמקרים:

```
switch (month)
{
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
        numDays = 31;
        break;
    case 4:
    case 6:
    case 9:
    case 11:
        numDays = 30;
        break;
    case 2:
        if ( ((year%4 == 0) && (year%100 != 0)) || (year%400 == 0) )
            numDays = 29;
        else
            numDays = 28;
        break;
    default:
        Console.WriteLine("Error, Acceptable values for months " +
                           "are 1-12");
        break;
} // switch
```

א. מהי מטרת קטע התוכנית?

- ב. תנו דוגמה לקלט שעבורו יגיע ביצוע משפט ה-`switch` לחלופת ברירת המחדל.
- ג. תנו דוגמה לקלט שעבורו ערכו של המשתנה `numDays` בתום ביצוע משפט ה-`switch` יהיה 28 ודוגמה לערך שעבורו ערכו של המשתנה `numDays` בתום ביצוע משפט ה-`switch` יהיה 29.

### שאלה 5.53

פתחו וישמו אלגוריתם אשר הקלט שלו הוא שלושה מספרים שלמים המציינים יום, חודש ושנה של תאריך. הפלט הוא תיאור התאריך בצורה יותר ברורה על ידי הצגת שם החודש, במקום מספרו.

למשל, עבור הקלט 1995 10 12 יהיה הפלט 12 באוקטובר 1995.

### שאלה 5.54

במבחן יש 10 שאלות אמריקאיות, והציון עבור כל שאלה הוא 10 נקודות או 0 נקודות. כלומר ציון המבחן יכול להיות רק אחד מאחד עשר הציונים 0, 10, 20, ..., 90, 100.

פתחו אלגוריתם שהקלט שלו הוא ציון במבחן (כמספר שלם), והפלט שלו הוא הציון המילולי המתאים לו: לציון 100 במבחן מתאים הציון A, ל-90 מתאים B, ל-80 מתאים C, ל-70 מתאים D, ולכל ציון 60 ומטה מתאים F.

### שאלה 5.55

כתבו תוכנית בשפת C# שתגדיל קלף מתוך חפיסת קלפים ותדפיס את צורתו ואת ערכו. ערך של קלף הוא מספר בין 1 ל-13, וצורה של קלף היא: לב, תלתן, מעוין או עלה.

**הדרכה:** הגדילו מספר בין 1 ל-13 ומספר בין 1 ל-4. לפי התוצאה הציגו את ערך הקלף ואת צורתו (1 – לב, 2 – תלתן, 3 – מעוין, 4 – עלה).

## סיכום

בפרק זה ראינו כיצד להורות על מהלכי ביצוע שונים באלגוריתם על פי קיום תנאי או על פי אי-קיומו. הדבר נעשה באמצעות הוראה לביצוע-בתנאי.

**הוראה לביצוע-בתנאי** מורה על בחירה לפי תנאי בין ביצוע תת-משימה אחת (פשוטה או מורכבת) ובין ביצוע תת-משימה אחרת: `אם ... אז ...`, או מורה על בחירה אם לבצע או לא לבצע תת-משימה: `אם ...`.

התנאי המופיע בהוראה לביצוע-בתנאי יכול להיות תנאי פשוט או **תנאי מורכב**. תנאי מורכב בנוי מצירוף של תנאים פשוטים ושל הקשרים `ו` ו-`או`.

**ביטוי בוליאני** מבטא תנאי. אם התנאי שמייצג הביטוי הבוליאני מתקיים אז ערכו של הביטוי הבוליאני הוא `true` (אמת). אם התנאי שמייצג הביטוי הבוליאני אינו מתקיים אז ערכו של הביטוי הבוליאני הוא `false` (שקר).

כאשר ביטויים בוליאניים מקושרים בקשר `ו` ומרכיבים ביטוי בוליאני חדש, ערכו `true` כאשר **לפחות** אחד מערכי הביטויים הבוליאניים המקושרים הוא `true`. כאשר הם מקושרים בקשר `או`, ערכו של הביטוי החדש הוא `true` רק כאשר ערכי **כל** הביטויים הבוליאניים המקושרים הם `true`.

כאשר יש לנתב את מהלך הביצוע של האלגוריתם לבחירה מבין שלוש אפשרויות או יותר, ניתן לעתים להשתמש בהוראות מורכבות לביצוע-בתנאי:

**מבנה מקונון של הוראות לביצוע-בתנאי** הוא הוראה לביצוע-בתנאי שאחת (או יותר) מההוראותיה הפנימיות היא הוראה לביצוע-בתנאי בעצמה (פשוטה או מורכבת).

**הוראת שרשרת לביצוע-בתנאי** מתאימה לשרשרת של תנאים, שצריכים להיבדק זה אחר זה. התנאים נבדקים לפי הסדר עד שאחד מהם מתקיים. ברגע שתנאי מסוים מתקיים, מתבצעות ההוראות שהוגדרו בתחומו, ושאר התנאים לא נבדקים.

**בהוראת בחירה** בוחנים ערך של ביטוי, ומשווים אותו לרשימת ערכים אפשריים. לפי תוצאת ההשוואה מתבצעת קבוצת הוראות.

כאשר מופיעה באלגוריתם הוראה לביצוע-בתנאי יש לבדוק את מהלך האלגוריתם עבור **קלטים מייצגים**, כלומר, לפחות דוגמת קלט אחת שתביא לכך שהתנאי יתקיים ולפחות דוגמת קלט אחת שתביא לכך שהתנאי לא יתקיים.

**תיאורי משמעות** של קיום תנאים ושל אי-קיומם הם הערות המבהירות את התנאי. הם עוזרים לנו בקריאת תוכנית. יש לצרף תיאורים כאלה לתנאים שכדאי להבהיר את משמעותם.

## סיכום מרכיבי שפת C# שנלמדו בפרק 5

**הוראה לביצוע-בתנאי במבנה *אם...אז...*** נכתבת בשפת C# במשפט `if` באופן הבא:

```
(ביטוי בוליאני) if
{
    ההוראות אשר מבוצעות אם התנאי מתקיים
}
else
{
    ההוראות אשר מבוצעות אם התנאי אינו מתקיים
}
```

**הוראה לביצוע-בתנאי במבנה *אם...אז...*** נכתבת בשפת C# במשפט `if` באופן הבא:

```
(ביטוי בוליאני) if
{
    ההוראות אשר מבוצעות אם התנאי מתקיים
}
```

ליצירת ביטויים בוליאניים פשוטים ב-C# ניתן להשתמש בכל אחד מסימני השוואה של השפה. ניתן להשתמש בסימני השוואה כדי להשוות ערכים מכל טיפוס שערכיו ניתנים להשוואה. ניתן להשוות בין תווים.

**הקשר */*** מסומן ב-C# בסימן הפעולה `||` והקשר ***/&*** מסומן בסימן הפעולה `&&`. עדיפות הקשר `&&` גבוהה מעדיפות הקשר `||` כלומר `&&` קודם ל-`||` בסדר הפעולות. עם זאת מומלץ לסמן בבירור את טווח הפעולה של כל קשר בסוגריים, כדי ליצור ביטויים ברורים וקריאים.

כל אחת מההוראות בתוך משפט `if` ב-C# יכולה להיות בעצמה הוראה לביצוע-בתנאי מכל סוג שהוא. כך מתקבל **קינון** של הוראות לביצוע-בתנאי.

בהוראה מקוננת לביצוע-בתנאי, השיוך של `else` ל-`if` המתאים לו מתבצע על פי הכלל הבא:  
`else` תמיד משויך ל-`if` אשר נמצא בתוכנית לפניו וקרוב אליו ביותר, בתנאים הבאים:

♦ ל-`if` זה לא משויך כבר `else` אחר קרוב יותר.

♦ `if` זה איננו כלול בתחום סוגריים מסולסלים אחר שמסתיים עוד לפני ה-`else`.

**הוראות שרשרת לביצוע-בתנאי נכתבת בשפת C# באופן הבא:**

```
(ביטוי בוליאני 1)
if
{
    ההוראות אשר מבוצעות אם ערכו של ביטוי 1 הוא true
}
else if (ביטוי בוליאני 2)
{
    ההוראות אשר מבוצעות אם ערכו של ביטוי 2 הוא true
}
.
.
.
else
{
    ההוראות אשר מבוצעות אם ערכם של כל הביטויים הוא false
}
```

**הוראות בחירה נכתבת בשפת C# במשפט switch באופן הבא:**

```
(ביטוי)
switch
{
    case <קבוצת הוראות>: ערך; break;
    case <קבוצת הוראות>: ערך; break;
    .
    .
    default: <קבוצת הוראות> break;
}
```

ערכו של הביטוי מושווה לכל אחד מהערכים המפורטים בחלופות השונות. **נקודת הכניסה** למשפט switch היא המשפט הצמוד לערך שעבורו הצליחה ההשוואה. מנקודת הכניסה מתבצעות כל ההוראות, עד להוראת break...

קבוצת ההוראות הצמודה ל-default (**ברירת המחדל**) מתבצעת אם ערכו של הביטוי המשווה אינו שווה לאף אחד מהערכים המפורטים. אין חובה לכלול טיפול ברירת מחדל. אם משפט switch אינו כולל חלופת ברירת מחדל, ובמהלך הביצוע ערכו של הביטוי המשווה אינו שווה לאף אחד מהערכים המפורטים, לא תבוצע אף הוראה.

## שאלות נוספות

### שאלות נוספות לסעיף 5.1

1. נתונים שני קטעי תוכניות הבאים, אשר המשתנים בהם הם מטיפוס שלם:

קטע תוכנית 2:

```
if (num1 > num2 )
    diff = num1 - num2;
else
    diff = num2 - num1;
Console.WriteLine(diff);
```

קטע תוכנית 1:

```
if (num < 0)
    num = -num;
Console.WriteLine(num);
```

א. עבור כל קטע תוכנית, בחרו שתי דוגמאות קלט כך שעבור האחת יתקיים התנאי המופיע בקטע ועבור השנייה לא יתקיים התנאי.

ב. מהי המטרה של כל קטע תוכנית?

ג. עבור כל קטע תוכנית, כתבו קטע תוכנית אחר ללא משפט `if`, המשיג את אותה המטרה.

2. התעריף לתשלום עבור צריכת גז ביתי לחימום הוא: 10 שקלים עבור כל אחד מחמשת הליטרים הראשונים הנצרכים ו-7 שקלים עבור כל ליטר נוסף. נתון קטע תוכנית אשר הקלט שלו הוא כמות ליטרים שנצרכה הנתונה כמספר ממשי, והפלט שלו הוא התשלום הכולל והתשלום הממוצע עבור ליטר נצרך. משתני קטע התוכנית הם מטיפוס ממשי: `consumption` מייצג את כמות הליטרים שנצרכה, `payment` מייצג את התשלום הכולל ו-`average` מייצג את התשלום הממוצע עבור ליטר נצרך.

```
if (consumption <= 5)
{
    השלימו
}
else
{
    השלימו
}
```

3. אחד השימושים של מחשב הוא הצפנת הודעות. דרך אחת להצפנת הודעות היא באמצעות החלפת תווים בתווים אחרים, ובאמצעות "ריפוד" בתווים נוספים. פתחו וישמו אלגוריתם אשר הקלט שלו הוא זוג אותיות שונות מהא"ב האנגלי, והפלט שלו הוא שלושה תווים לפי החוקיות הבאה: אם אות הקלט הראשונה מופיעה בסדר האלף-בית האנגלי לפני אות הקלט השנייה אז הפלט הוא האות העוקבת באלף-בית לאות הקלט הראשונה, לאחריה הסימן '+' ולבסוף האות הקודמת באלף-בית לאות הקלט השנייה. אחרת הפלט הוא האות הקודמת באלף-בית לאות הקלט הראשונה, לאחריה הסימן '-' ולבסוף האות העוקבת באלף-בית לאות הקלט השנייה. למשל, עבור הקלט `ac` יהיה הפלט `b+b` ועבור הקלט `da` יהיה הפלט `c-b`.

## שאלות נוספות לסעיף 5.2

1. מהירות הנסיעה המותרת בכביש מהיר היא 55 קמ"ש לכל הפחות ו-100 קמ"ש לכל היותר. פתחו אלגוריתם אשר הקלט שלו הוא מהירות נסיעה של מכונית, והפלט שלו הוא הודעה אם נהג המכונית חרג מגבולות המהירות המותרת. ישמו את האלגוריתם בשפת `C#`.
2. פתחו וישמו אלגוריתם אשר הקלט שלו הוא מספר שלם חיובי קטן מ-100 והפלט שלו הוא המילה בוים אם המספר הנתון מתחלק ב-7 או כולל את הספרה 7.

## שאלות נוספות לסעיפים 5.3, 5.4 ו-5.5

1. פתחו אלגוריתם הקולט תו. אם התו שווה ל-'m', האלגוריתם קורא מהקלט שני מספרים שלמים, ומציג כפלט את המספר הגדול מביניהם. אם תו הקלט שווה ל-'m', האלגוריתם קורא תו נוסף. אם התו הנוסף שווה ל-'s', האלגוריתם מציג את ההודעה תחשוב חיובי, אחרת הוא קולט מספר ומציג כפלט את ההודעה אתה המספר שנקלט ואת המספר עצמו. ישמו את האלגוריתם בשפת `C#`.



2. בחברת ההיי-טק "הייטק" יש 10 דרגות לעובדים. כל העובדים שהם מעל לדרגה 7 הם מנהלים, כל העובדים מעל לדרגה 4 הם ראשי צוותים, כל העובדים מעל לדרגה 2 הם עובדים קבועים וכל השאר הם סטודנטים. פתחו אלגוריתם (לשימוש מחלקת משאבי האנוש של החברה) הקולט את דרגת העובד ומציג כפלט את תיאורו של העובד (מנהל, ראש צוות, עובד קבוע או סטודנט). ישמו את האלגוריתם בשפת C#.

3. עזרו לזברה לדעת איזה יום היום. פתחו אלגוריתם הקולט מספר בין 1 ל-7 ומציג את השם של היום המתאים (Sunday, Monday...). ישמו את האלגוריתם בשפת C#.

## שאלות מסכמות לפרק 5

1. בכל המשפטים שלהלן  $x$  הוא משתנה מטיפוס ממשי. כתבו עבור כל זוג ממשפטי ה-`if` הנתונים משפט `if` יחיד שקול, כלומר שביצועו שקול לביצוע שני משפטי ה-`if`, בזה אחר זה:

א.	ב.
<code>if (x &lt;= 9)</code> <code>x = x - 1;</code> <code>if (x &gt; 9)</code> <code>x = x + 1;</code>	<code>if (x &gt;= 5)</code> <code>x = x * 4;</code> <code>if (x &gt;= 20)</code> <code>x = x / 2;</code>
ג.	ד.
<code>if (x == -1)</code> <code>x = 1;</code> <code>if (x &gt; 0)</code> <code>x = -x;</code>	<code>if (x &lt; -1)</code> <code>x = -x;</code> <code>if (x &gt; 1)</code> <code>x = -x;</code>

2. על לוח שחמט מוצבים שני כלים בלבד – צריח ורץ. השורות בלוח ממוספרות מ-1 עד 8, וגם העמודות ממוספרות מ-1 עד 8:

	8	7	6	5	4	3	2	1	
1									
2									
3									
4									
5									
6									
7									
8									

צריח (Rook באנגלית) מאיים על כלי הנמצא עמו באותה שורה או באותה עמודה.  
 רץ (Bishop באנגלית) מאיים על כלי הנמצא עמו על אותו אלכסון.  
 במשתנים `rookRow` ו-`rookCol` שמורים מספרי השורה והעמודה שהצריח מוצב עליהן.  
 במשתנים `bishopRow` ו-`bishopCol` שמורים מספרי השורה והעמודה שהרץ מוצב עליהן.  
 א. השלימו את תיאור משמעות קיום התנאי במשפט `if` הבא:

```
if ((rookRow + rookCol) % 2) == 1)
    //
```

```
Console.WriteLine("The rook is on ...");
```

ב. כתבו ביטוי בוליאני שערכו `true` אם הצריח מאיים על הרץ.

ג. הביטוי הבוליאני הבא אמור לבטא מצב לוח שהרץ מאיים על הצריח:

```
(bishopRow - rookRow) == (bishopCol - rookCol)
```

הביטוי כולל רק חלק מן המקרים האפשריים. מהם המקרים הנכללים בו? מהם המקרים

שאינם נכללים בו?

הרחיבו את הביטוי כך שיכלול את כל המקרים האפשריים ורק אותם.

## תבניות – פרק 5

פירוט מלא של התבניות ושל שאלות שבפתרון יש שימוש בתבניות ניתן למצוא באתר הספר ברשת האינטרנט.

### מציאת מקסימום ומינימום בסדרה

שם התבנית: **מציאת מקסימום בסדרה**

נקודת מוצא: שני ערכים במשתנים `element1` ו-`element2`

מטרה: מציאת הערך הגדול ביותר מבין שני הערכים

אלגוריתם:

1. **השם** `max`-2 **אם** `element1` **ש**

2. **אם** `element2 > max`

2.1 **השם** `max`-2 **אם** `element2` **ש**

שם התבנית: **מציאת מינימום בסדרה**

נקודת מוצא: שני ערכים במשתנים `element1` ו-`element2`

מטרה: מציאת הערך הקטן ביותר בין שני הערכים

אלגוריתם:

1. **השם** `min`-2 **אם** `element1` **ש**

2. **אם** `element2 < min`

2.1 **השם** `min`-2 **אם** `element2` **ש**

## סידור ערכים בסדרה

שם התבנית: **סידור ערכים בסדר עולה בסדרה**  
נקודת מוצא: שני ערכים במשתנים element1 ו-element2  
מטרה: השמת הערך הקטן יותר ב-element1 והערך הגדול יותר ב-element2  
אלגוריתם:  
1. אם  $element1 > element2$   
1.2 החלף את ערכי element1 ו-element2

שם התבנית: **סידור ערכים בסדר יורד בסדרה**  
נקודת מוצא: שני ערכים במשתנים element1 ו-element2  
מטרה: השמת הערך הגדול יותר ב-element1 והערך הקטן יותר ב-element2  
אלגוריתם:  
1. אם  $element1 < element2$   
1.2 החלף את ערכי element1 ו-element2

## ערכים עוקבים

שם התבנית: **ערכים עוקבים?**  
נקודת מוצא: שני ערכים element1 ו-element2  
מטרה: קביעת הערך true אם element2 עוקב ל-element1, וקביעת הערך false אם element2 אינו עוקב ל-element1  
אלגוריתם (ביטוי בוליאני):  
(האם)  $element1 + 1 = element2$

## זוגיות מספר

שם התבנית: **מספר זוגי?**  
נקודת מוצא: מספר שלם num  
מטרה: קביעת הערך true אם num זוגי וקביעת הערך false אם num אי-זוגי  
אלגוריתם (ביטוי בוליאני):  
(האם) שארית החלוקה של num פריטים ל-2 קבוצות שווה ל-0

שם התבנית : מספר אי-זוגי?

נקודת מוצא : מספר שלם num

מטרה : קביעת הערך true אם num אי-זוגי וקביעת הערך false אם num זוגי

אלגוריתם (ביטוי בוליאני) :

(האם) שארית החלוקה של num פריטים ל-2 קבוצות שווה ל-1

## מחלק של מספר

שם התבנית : מחלק של?

נקודת מוצא : שני מספרים שלמים num1 ו-num2

מטרה : קביעת הערך true אם num2 מחלק את num1 וקביעת הערך false אם num2 אינו

מחלק את num1

אלגוריתם (ביטוי בוליאני) :

(האם) שארית החלוקה של num1 פריטים ל-num2 קבוצות שווה ל-0

## תבניות – פרק 5

### מציאת מקסימום ומינימום בסדרה

נתבונן בשתי הבעיות האלגוריתמיות הבאות:

**בעיה 1:** תלמיד רשאי לגשת לבחינת הבגרות במתמטיקה בשני מועדים שונים: מועד א' ומועד ב'. ציונו של התלמיד בבחינה נקבע על פי הציון הגבוה מבין השניים. כתבו אלגוריתם שהקלט שלו הוא ציוניה של לירון (כמספרים שלמים) בבחינת הבגרות במתמטיקה במועד א' ובמועד ב', והפלט שלו הוא הציון הקובע של לירון בבחינה.

**בעיה 2:** כתבו אלגוריתם שהקלט שלו הוא שתי אותיות גדולות ב-ABC והפלט שלו הוא האות המאוחרת על פי הסדר המילוני. הניחו שהאותיות שונות זו מזו.

אנו רואים כי בשתי הבעיות האלגוריתמיות יש למצוא את המקסימום בסדרה בת שני ערכים. בבעיה הראשונה יש למצוא את המקסימום בסדרה בת שני ערכים מספריים ובבעיה השנייה יש למצוא את המקסימום בסדרה בת שני ערכים תווים. באופן דומה ישנן בעיות אלגוריתמיות שעבורן יש למצוא את המינימום בסדרת ערכים. מציאת הערך הגדול או הקטן ביותר בסדרת ערכים הינה אחת התבניות הבסיסיות ביותר במדעי המחשב. תבנית זו שימושית הן בפני עצמה והן כמרכיב בתבניות מורכבות יותר כמו תבניות של מיון סדרת ערכים, שנכיר בהמשך לימודינו.

נתבונן בשני האלגוריתמים הללו:

1. קלט שני אותיות ב- $letter1$ ו- $letter2$	1. קלט ציונים של לירון ב- $math1$ ו- $math2$
2. השם ב- $max$ את ערכו של $letter1$	2. השם ב- $max$ את ערכו של $math1$
3. אם $letter2 > max$	3. אם $math2 > max$
3.1. השם ב- $max$ את ערכו של $letter2$	3.1. השם ב- $max$ את ערכו של $math2$
4. הצג כפלט את הערך $max$	4. הצג כפלט את הערך $max$

אנו רואים כי בשני הפתרונות נקבע באופן שרירותי כי הערך של המשתנה הראשון הוא המקסימלי ולכן נשמר ערכו במשתנה  $max$ . לאחר מכן, נבדק ערכו של המשתנה השני: אם הוא גדול יותר מהמקסימום שנקבע אז ערכו של  $max$  מוחלף בערך האיבר השני.

הבחירה השרירותית במשתנה הראשון כערך התחלתי ל- $max$  נראית לכאורה מיותרת. ואכן, ניתן לוותר עליה, ולהשתמש בהוראה במבנה **אם... אז... אחרת...**, המשווה בין שני הערכים ומשימה בהתאם לתוצאת ההשוואה את ערכו של אחד מהם ב- $max$  (בדומה למה שנעשה בשאלה 5.6). בכל

זאת, נעדיף את הפתרונות כפי שהוצגו, משום שקל יהיה להרחיבם למציאת מקסימום בסדרה שבה יותר משני איברים, כפי שנראה בשלב מאוחר יותר.

נפריד את מאפייני התבנית **מציאת מקסימום ומינימום בסדרה** לשתי תת-תבניות: ראשית, נציג את מאפייני התבנית **מציאת מקסימום בסדרה** ואחר כך נציג את מאפייני התבנית **מציאת מינימום בסדרה**.

**שם התבנית:** מציאת מקסימום בסדרה

**נקודת מוצא:** שני ערכים במשתנים element1 ו-element2

**מטרה:** מציאת הערך הגדול ביותר מבין שני הערכים

**אלגוריתם:**

1. השם max-2 אם הערך של element1

2. אם  $element2 > max$

2.1 השם max-2 אם הערך של element2

**יישום ב-C#:**

```
max = element1;
if (element2 > max)
{
    max = element2;
}
```

**שם התבנית:** מציאת מינימום בסדרה

**נקודת מוצא:** שני ערכים במשתנים element1 ו-element2

**מטרה:** מציאת הערך הקטן ביותר מבין שני הערכים

**אלגוריתם:**

1. השם min-2 אם הערך של element1

2. אם  $element2 < min$

2.1 השם min-2 אם הערך של element2

**יישום ב-C#:**

```
min = element1;
if (element2 < min)
{
    min = element2;
}
```

## שאלה 1

- א.
1. ישמו בשפת C# את האלגוריתם לחישוב הציון הקובע של לירון במתמטיקה.
  2. הרחיבו את התוכנית שכתבתם בסעיף א.1 כך שיוצג כפלט גם מועד הבחינה (א' או ב') שבו הושג הציון הגבוה יותר.
  3. איזה מועד בחינה יוצג כפלט במקרה שציוניה של לירון זהים בשני המועדים? הסבירו.
- ב.
1. ישמו בשפת C# את האלגוריתם למציאת האות המאוחרת יותר.
  2. שנו את התוכנית שכתבתם בסעיף ב.1 כך שתוצג כפלט האות המוקדמת יותר על פי סדר מילוני.

## שאלה 2

נתון אלגוריתם שהקלט שלו הוא שני מספרים ממשיים:

1. קאוט שני מספרים ממשיים ב- num1 ו- num2
  2. מצא מקסימום בסדרה num1, num2 /השם אג ערכו ב- max
  3. מצא מינימום בסדרה num1, num2 /השם אג ערכו ב- min
  4. הצג כפאוט אג ערכו של הביטוי  $\max - \min$
- א. מהו הפלט עבור הקלט 6.9 5.3?
- ב. תנו שתי דוגמאות קלט שונות שעבורן יוצג כפלט הערך 9.1.
- ג. מהי מטרת האלגוריתם?
- ד. כתבו אלגוריתם המשיג את אותה המטרה ללא שימוש בתבניות **מציאת מקסימום בסדרה** ו- **מציאת מינימום בסדרה**.

## שאלה 3

- נתון אלגוריתם שהקלט שלו הוא שני מספרים שלמים שונים והפלט שלו הוא המספר הדו-ספרתי הקטן ביותר מבין ערכי הקלט. הניחו שלפחות אחד מבין שני המספרים שבקלט הוא דו-ספרתי.
1. קאוט שני מספרים שלמים ב- num1, num2
  2. מצא מינימום בסדרה num1, num2 והצג כפאוט אג ערכו
- האלגוריתם שגוי.
- א. תנו דוגמת קלט שעבורה לא ניתן לראות כי האלגוריתם שגוי. מהו המאפיין של הדוגמה?
- ב. תנו שתי דוגמאות קלט שונות שעבורן ניתן לראות כי האלגוריתם שגוי. מהו המאפיין של כל אחת מהדוגמאות?
- ג. הסבירו במלים מדוע האלגוריתם שגוי.
- ד. תקנו את האלגוריתם.
- ה. ישמו את האלגוריתם כקטע תוכנית בשפת C#.

#### שאלה 4

נתון אלגוריתם שהקלט שלו הוא 3 מספרים שלמים השונים זה מזה :

1. קאוט 3 מספרים שונים - num1, num2, num3

2. מצא מינימום בסדרה num1, num2 /השם אג ערכו min-

3. מצא מינימום בסדרה min, num3 /השם אג ערכו min-

4. הצג כפאט אג ערכו min

א. מה יוצג כפלט עבור הקלט 57 34 816?

ב. תנו שלוש דוגמאות קלט שונות שעבורן יהיה הפלט 5. מהו המאפיין של כל אחת מדוגמאות הקלט?

ג. ישמו את האלגוריתם בשפת C#.

**שאלה 5** (שאלה זו מתאימה לאחר לימוד סעיף 5.3 – קינון של הוראה לביצוע בתנאי)

נתונה הבעיה האלגוריתמית הבאה :

כתבו אלגוריתם שהקלט שלו הוא 3 גבהים של שחקני כדורסל והפלט שלו הוא הגובה המקסימלי מבין שלושת נתוני הקלט. הניחו שהגבהים שונים זה מזה.  
זוהי למעשה בעיית מציאת מקסימום בסדרה בת 3 ערכים. אנה, בן, ענר ומשה הציעו אלגוריתמים שונים לפתרון הבעיה.

האלגוריתם של אנה :

1. קאוט 3 גבהים שגוקן הכדורסל - height1, height2, height3

2. השם max-אג ערכו height1

3. אם height2 > max

3.1. השם max-אג ערכו height2

4. אם height3 > max

4.1. השם max-אג ערכו height3

5. הצג כפאט אג ערכו max

האלגוריתם של בן :

1. קאוט 3 גבהים שגוקן הכדורסל - height1, height2, height3

2. אם height1 > height2 וגם height2 > height3

2.1. הצג כפאט אג ערכו height1

3. אם height2 > height3 וגם height3 > height1

3.1. הצג כפאט אג ערכו height2



$$4. \text{ אם } height1 > height2 \text{ וגם } height3 > height1$$

$$4.1. \text{ הצג כפולט את ערכו של } height3$$

האלגוריתם של ענר :

$$1. \text{ קולט 3 גבהים של שוקני הכדורסל ב- } height1, height2, \text{ ו- } height3$$

$$2. \text{ אם } height1 > height2 \text{ וגם } height1 > height3$$

$$2.1. \text{ הצג כפולט את ערכו של } height1$$

$$3. \text{ אם } height2 > height1 \text{ וגם } height2 > height3$$

$$3.1. \text{ הצג כפולט את ערכו של } height2$$

$$4. \text{ אם } height3 > height1 \text{ וגם } height3 > height2$$

$$4.1. \text{ הצג כפולט את ערכו של } height3$$

האלגוריתם של משה :

$$1. \text{ קולט 3 גבהים של שוקני הכדורסל ב- } height1, height2, \text{ ו- } height3$$

$$2. \text{ אם } height1 > height2$$

$$2.1. \text{ אם } height1 > height3$$

$$2.1.1. \text{ הצג כפולט את ערכו של } height1$$

$$2.2. \text{ אחרת}$$

$$2.2.1. \text{ הצג כפולט את ערכו של } height3$$

$$3. \text{ אחרת}$$

$$3.1. \text{ אם } height2 > height3$$

$$3.1.1. \text{ הצג כפולט את ערכו של } height2$$

$$3.2. \text{ אחרת}$$

$$3.2.1. \text{ הצג כפולט את ערכו של } height3$$

עבור כל אחד מהאלגוריתמים המוצעים ענו על הסעיפים הבאים :

א. אם האלגוריתם נכון :

$$1. \text{ בנו טבלת מעקב עבור הקלט } 1.98 \quad 2.05 \quad 1.94$$

$$2. \text{ הסבירו במלים את הרעיון עליו מתבסס האלגוריתם.}$$

ב. אם האלגוריתם אינו נכון :

$$1. \text{ תנו לפחות דוגמת קלט אחת שעבורה ניתן לראות כי האלגוריתם שגוי. אפיינו את דוגמת הקלט.}$$



## סידור ערכים בסדרה

**סידור ערכים בסדרה** היא תבנית של מיון ערכי סדרה נתונה. ניתן לסדר את ערכי הסדרה בסדר עולה או לסדרם בסדר יורד.

התבנית **סידור ערכים בסדר עולה בסדרה** מביאה למצב בו איברי הסדרה ממוינים בסדר עולה. מכאן נובע כי הערך המינימלי בסדרה נמצא בקצה השמאלי של הסדרה והערך המקסימלי נמצא בקצה הימני של הסדרה.

התבנית **סידור ערכים בסדר יורד בסדרה** מביאה למצב בו איברי הסדרה ממוינים בסדר יורד. מכאן נובע כי הערך המקסימלי בסדרה נמצא בקצה השמאלי של הסדרה והערך המינימלי נמצא בקצה הימני של הסדרה.

נפריד את מאפייני התבנית **סידור ערכים בסדרה** לשתי תת-תבניות: ראשית נציג את מאפייני התבנית **סידור ערכים בסדר עולה בסדרה** ואחר כך נציג את מאפייני התבנית **סידור ערכים בסדר עולה בסדרה**.

תחילה נציג את התבניות לסידור ערכים בסדרה בת שני ערכים בסדר עולה ובסדר יורד, לאחר מכן נרחיב לסדרה בת 3 ערכים ובהמשך נראה אלגוריתם כללי יותר לסידור מספר גדול יותר של ערכים בסדרה.

**שם התבנית:** סידור ערכים בסדר עולה בסדרה

**נקודת מוצא:** שני ערכים במשתנים element1 ו-element2

**מטרה:** השמת הערך הקטן יותר ב-element1 והערך הגדול יותר ב-element2

**אלגוריתם:**

1. אם  $element1 > element2$

1.2 החלף את ערכי element1 ו-element2

**שם התבנית:** סידור ערכים בסדר יורד בסדרה

**נקודת מוצא:** שני ערכים במשתנים element1 ו-element2

**מטרה:** השמת הערך הגדול יותר ב-element1 והערך הקטן יותר ב-element2

**אלגוריתם:**

1. אם  $element1 < element2$

1.2 החלף את ערכי element1 ו-element2

## שאלה 9

ישמו כל אחד מן האלגוריתמים לסידור הערכים element1 ו-element2 כקטע תוכנית בשפת C#. שימו לב כי כדי ליישם את התבניות של **סידור ערכים בסדרה** עליכם להשתמש ביישום של התבנית **החלפת ערכים בין שני משתנים**.

## שאלה 10

א. פתחו אלגוריתם שהקלט שלו הוא מספר דו-ספרתי num והפלט שלו הוא המספר הגדול ביותר שניתן להרכיב מספרות המספר הנתון.  
ב. ציינו באילו תבניות השתמשתם עבור כתיבת האלגוריתם.  
ג. ישמו את האלגוריתם בשפת C#.

## שאלה 11

נתונה סדרה של שלושה ערכים: element1, element2, element3. ידוע כי הערך השני בגודלו אינו נמצא ב-element2. נתון אלגוריתם חלקי, שמטרתו לסדר את ערכי הסדרה בסדר עולה:

1. אם  $element1 < element2$  // גיאו קיום גלוי: \_\_\_\_\_

1.1. החלף את ערכי \_\_\_\_\_ ו- \_\_\_\_\_

1.2. סדר בסדר עולה את איברי הסדרה \_\_\_\_\_, \_\_\_\_\_

2. אחרת // גיאו אי-קיום גלוי: \_\_\_\_\_

2.1. החלף את ערכי \_\_\_\_\_ ו- \_\_\_\_\_

2.2. סדר בסדר עולה את איברי הסדרה \_\_\_\_\_, \_\_\_\_\_

א. השלימו את האלגוריתם. הוסיפו תיאורי קיום ואי-קיום תנאי במקומות המסומנים.  
ב. שנו את האלגוריתם שכתבתם כך שיסדר את ערכי הסדרה בסדר יורד.  
ג. ישמו את שני האלגוריתמים כקטעי תוכניות בשפת C#.

## שאלה 12

א. כתבו אלגוריתם שהקלט שלו הוא סדרה של 3 מספרים שלמים שונים והפלט שלו הוא סידור של המספרים, כך שעבור הסידור המתקבל שני הערכים המוחלטים של הפרשי המספרים יהיו בסדר עולה.  
ב. ציינו באילו תבניות השתמשתם עבור כתיבת האלגוריתם.  
ג. ישמו את האלגוריתם בשפת C#.

## שאלה 13

נתון אלגוריתם, שהקלט שלו הוא סדרה של שלושה ערכים element1, element2, element3:

1. קלוט שלושה מספרים שונים element1, element2, element3

2. סדר בסדר עולה את איברי הסדרה element1, element2

3. סדר בסדר עולה את איברי הסדרה element2, element3
4. סדר בסדר עולה את איברי הסדרה element1, element2
5. הצג כפולט את הערכים element1, element2, element3

- א. מה יהיה הפלט עבור הקלט 6 13 8?
- ב. מהי מטרת האלגוריתם?
- ג. באלגוריתם נעשה שלוש פעמים שימוש בתבנית **סידור ערכים בסדר עולה בסדרה**.
1. תנו דוגמה לקלט שעבורו מתבצעת החלפת ערכי המשתנים שלוש פעמים.
2. תנו דוגמה לקלט שעבורו לא מתבצעת החלפת ערכי המשתנים כלל.
- ד. ישמו את האלגוריתם בשפת C#.
-

## ערכים עוקבים

שם התבנית: ערכים עוקבים?

נקודת מוצא: שני ערכים element1 ו-element2

מטרה: קביעת הערך true אם element2 עוקב ל-element1, וקביעת הערך false אם element2 אינו עוקב ל-element1

אלגוריתם (ביטוי בוליאני):

$element1 + 1 = element2$

יישום ב-C#:

$(element1 + 1 == element2)$

התבנית **ערכים עוקבים?** היא תבנית המחשבת ערך בוליאני, כלומר, true או false. משום כך, האלגוריתם שמממש את התבנית, וכמוהו גם יישומו בשפת C# כוללים למעשה ביטוי בוליאני. את הערך המחושב על ידי התבנית ניתן לשלב בביטוי בוליאני.

**שימו ♥:** הערכים element1 ו-element2 עשויים להיות מספרים שלמים, תווים וכן כל זוג ערכים בדידים שניתנים לסידור.

### שאלה 14

לפניכם שימוש בתבנית **ערכים עוקבים?**:

1. אסך 5 ערך עוקב ל-num

1.1. הצג כפלט "אמת"

2. אגרא

2.1. הצג כפלט "שקר"

א. תנו דוגמה לערך של num שעבורו יוצג כפלט "אמת".

ב. תנו שתי דוגמאות לערך של num שעבורו יוצג כפלט "שקר".

### שאלה 15

נתונים שני מספרים שלמים num1 ו-num2. עבור כל אחד מהסעיפים הבאים כתבו ביטוי בוליאני מתאים:

א. המספר הראשון אינו עוקב למספר השני.

ב. שני המספרים עוקבים (הסדר אינו משנה).

## שאלה 16

- נתונים שני תווים ch1 ו-ch2. עבור כל אחד מהסעיפים הבאים כתבו ביטוי בוליאני מתאים :
- א. שני התווים הם אותיות קטנות עוקבות ב-abc או ששני התווים הם אותיות גדולות עוקבות ב-ABC.
- ב. שני התווים הם ספרות עוקבות (בין '0' ל-'9').
-

## זוגיות מספר

בדיקת זוגיות של מספר שימושית בהקשרים רבים במדעי המחשב. למשל, נזכור כי בזיכרון המחשב נשמרים ערכים כסדרות של סיביות. אם מתייחסים לסדרת סיביות כאל מספר, אז בדיקת הזוגיות של המספר מעידה האם הסיבית הימנית ביותר במספר היא 0 או 1.

נפריד את מאפייני התבנית **זוגיות מספר** לשתי תת-תבניות: ראשית נציג את מאפייני התבנית **מספר זוגי?** ואחר כך נציג את מאפייני התבנית **מספר אי-זוגי?**. שתי התבניות מחשבות ערכים בוליאניים, בדומה לתבנית **ערכים עוקבים?**.

<b>שם התבנית:</b> מספר זוגי?
<b>נקודת מוצא:</b> מספר שלם num
<b>מטרה:</b> קביעת הערך true אם num זוגי וקביעת הערך false אם num אי-זוגי
<b>אלגוריתם (ביטוי בוליאני):</b>
שארית החלוקה של num פריטים ל-2 קבוצות שווה ל-0

<b>שם התבנית:</b> מספר אי-זוגי?
<b>נקודת מוצא:</b> מספר שלם num
<b>מטרה:</b> קביעת הערך true אם num אי-זוגי וקביעת הערך false אם num זוגי
<b>אלגוריתם (ביטוי בוליאני):</b>
שארית החלוקה של num פריטים ל-2 קבוצות שווה ל-1

**שימו ♥:** שתי התת-תבניות **מספר זוגי?** ו-**מספר אי-זוגי?** הן שתי תבניות המשלימות זו את זו. כלומר, עבור מספר שלם מסוים, חישוב הערך true בשימוש בתבנית אחת יגרור חישוב הערך false בשימוש בתבנית השנייה, ולהיפך.

### שאלה 17

נתונים שני מספרים שלמים num1 ו-num2. לפניכם מספר ביטויים בוליאניים חלקיים המחזירים ערך true אם אחד משני המספרים זוגי והאחר אי-זוגי, ו-false אחרת. השלימו את הביטויים הבוליאניים:



א. ( \_\_\_\_\_ מספר זוגי /גס \_\_\_\_\_ מספר אי-זוגי )

א/

( \_\_\_\_\_ מספר זוגי /גס \_\_\_\_\_ מספר אי-זוגי )

ב. (שארית החלוקה של num1 פריטים ל-2 קבוצות שווה ל-1) \_\_\_\_\_ (שארית החלוקה של num1 פריטים ל-2 קבוצות שווה ל-1) \_\_\_\_\_ (מספר זוגי) \_\_\_\_\_

### שאלה 18

נתון אלגוריתם שהקלט שלו הוא מספר שלם num :

1. קאוט מספר שלם כ- num

1.1. אם num מספר זוגי

1.1.1. הקטן את ערכו של num פי 2

1.2. אחרת

1.2.1. הגדל את ערכו של num פי 2

2. הצג כקאוט את ערכו של num

א. מה יהיה הפלט עבור הקלט 15? היעזרו בטבלת מעקב.

ב. תנו שתי דוגמאות קלט שונות, שעבורן יוצג כפלט הערך 10.

ג. ישמו את האלגוריתם בשפת C#.

### שאלה 19

בחברת "מודיעין אזרחי" לכל לקוח יש מספר ייחודי משלו. החברה מצפינה את מספרי לקוחותיה באופן הבא: כל ספרה אי-זוגית הופכת לזוגית על ידי הפחתה של 1. שאר הספרות נותרות ללא שינוי. למשל, עבור מספר הלקוח 921 יתקבל הקוד 820 ועבור מספר הלקוח 129 יתקבל הקוד 28 (הספרה 1 הפכה ל-0, ומשום שהייתה ספרה מובילה נעלמה).

א. כתבו אלגוריתם, שהקלט שלו הוא מספר תלת-ספרתי של לקוח והפלט שלו הוא המספר המוצפן.

ב. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם.

ג. לאחר שנה החליטה החברה להצפין שוב את מספרי לקוחותיה. הפעם בחרה בשיטת הצפנה זו: כל ספרה אי-זוגית הופכת לזוגית על ידי הוספת 1 ב"צורה מעגלית" (כלומר, 9 הופכת ל-0). שאר הספרות נשארות ללא שינוי. למשל, עבור מספר הלקוח 976 יתקבל הקוד 86 ועבור מספר הלקוח 439 יתקבל הקוד 440. הרחיבו את האלגוריתם שכתבתם גם עבור שיטת ההצפנה השנייה.

ד. ישמו את האלגוריתם בשפת C#.

## מחלק של מספר

התבנית **מחלק של?** היא הרחבה של התבנית **מספר זוגי?**. נאמר על מספר שלם  $num2$  שהוא מחלק את המספר השלם  $num1$  אם  $num1$  מתחלק ב- $num2$  ללא שארית. למעשה, התבנית **מספר זוגי?** בודקת האם 2 הוא מחלק של מספר נתון, ולכן שימוש בתבנית **מחלק של?** עבור שני מספרים  $num1$  ו- $num2$ , כאשר ערכו של  $num2$  הוא 2, שקול לשימוש בתבנית **מספר זוגי?** עבור  $num1$ . כמו **מספר זוגי?** גם **מחלק של?** היא תבנית המחשבת ערך בוליאני.

**שם התבנית:** מחלק של?

**נקודת מוצא:** שני מספרים שלמים  $num1$  ו- $num2$

**מטרה:** קביעת הערך  $true$  אם  $num2$  מחלק את  $num1$  וקביעת הערך  $false$  אם  $num2$  אינו מחלק את  $num1$

**אלגוריתם (ביטוי בוליאני):**

שארית החלוקה של  $num1$  פריטים ל- $num2$  קבוצות שווה ל-0

### שאלה 20

נתונים שני מספרים שלמים  $num1$  ו- $num2$ . עבור כל אחד מהסעיפים הבאים כתבו ביטוי בוליאני מתאים:

א. סכום שני המספרים מתחלק ב-4 ללא שארית.

ב. המספר הראשון זוגי ואינו מתחלק ב-3.

ג. המספר הראשון הוא מספר דו-ספרתי הגדול מ-50, וספרת העשרות שלו שווה לספרת האחדות של המספר השני.

### שאלה 21

נתון הביטוי הבוליאני הבא עבור מספר שלם חיובי  $num$ :

$(num \text{ מספר זוגי}) \wedge (\text{שארית החלוקה של } num \text{ פריטים ל-5 קבוצות שווה ל-0})$

א. כתבו את הביטוי הבוליאני בשפת C#.

ב. תנו דוגמה לערך של  $num$ , שעבורו יהיה ערך הביטוי  $true$ . הסבירו.

ג. תנו דוגמה לערך של  $num$ , שעבורו יהיה ערך הביטוי  $false$ . הסבירו.

ד. כתבו ביטוי בוליאני פשוט השקול לביטוי הנתון.

### שאלה 22

מורה מעוניינת לחלק את תלמידיה לזוגות לצורך מבצע התרמה שבו הם נוטלים חלק.

א. כתבו תוכנית בשפת C#, שהקלט שלה הוא מספר התלמידים בכיתה והפלט שלה הוא הודעה האם ניתן להתאים בין-זוג לכל תלמיד.

ב. מאחר שהמורה ראתה כי לא ניתן להתאים בן-זוג לכל תלמיד החליטה לחלק את הכיתה לקבוצות של שלושה תלמידים.

1. שנו את התוכנית שכתבתם בסעיף א כך שתוצג כפלט הודעה האם ניתן לחלק את הכיתה לקבוצות של שלושה תלמידים.

2. האם עתה נפתרה בעייתה של המורה? אם כן, הסבירו. אם לא, תנו דוגמת קלט שעבורה ניתן לראות כי המורה שגתה בהחלטתה.

### שאלה 23

א. כתבו אלגוריתם שהקלט שלו הוא זמן המיוצג בשעות ובדקות, והפלט שלו הוא הזמן הנותר בשעות ובדקות עד שעת חצות.

ב. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם.

ג. ישמו את האלגוריתם בשפת C#.

---

## פרק 6 – נכונות אלגוריתמים

לאורך ההיסטוריה הקצרה של מדעי המחשב יש אינספור דוגמאות של תוכניות שגויות, ולא מעט מהן הסתיימו בכי רע. אחת הדוגמאות היא סיפורה של ספינת חלל אמריקנית מסדרת מרינר שנשלחה אל כוכב הלכת ונוס. הספינה אבדה כתוצאה משגיאה בתוכנית המחשב שהופקדה על בקרת הטיסה, ומיליוני דולרים ירדו לטמיון.

חלק לא מבוטל מן המחקר במדעי המחשב מוקדש לתחום הנקרא "הוכחת נכונות של תוכניות". מטרתה של הוכחת נכונות של תוכנית היא אימות מלא לטענה שהתוכנית מציגה עבור כל קלט את הפלט הדרוש.

חוקרים פיתחו ומפתחים שיטות שונות להוכחת נכונות של תוכניות. שיטות אלה הן מתמטיות באופיין ונשענות על תיאוריות מתמטיות מורכבות למדי. חומר הלימוד של "יסודות מדעי המחשב" איננו דן בהוכחת נכונות של תוכניות, כיוון שהדבר מחייב ידע מתמטי שאיננו נרכש בבית-ספר תיכון. אך ההתייחסות לנכונות של תוכניות חשובה לאורך חומר הלימוד כולו, כבר עם פיתוח תוכניות ראשונות. לכן, אנו מציגים נושא זה כבר עתה, בצורה פשוטה ואינטואיטיבית.

אלגוריתם לפתרון בעיה אלגוריתמית נתונה הוא **נכון** אם ביצעו מביא להצגת הפלט הדרוש עבור כל קלט חוקי (כלומר כל קלט המתאים להגדרת הבעיה).

מספר האפשרויות השונות לקלטים חוקיים הוא בדרך כלל רב, ולעתים רבות אפילו אינסופי, ולכן לא מתקבל על הדעת להיווכח בנכונותו של אלגוריתם על ידי בדיקת הפלט עבור כל קלט אפשרי. בפרקי הלימוד של "יסודות מדעי המחשב" אנו בודקים את נכונותו של אלגוריתם על ידי בדיקת הפלט עבור דוגמאות שונות של קלט.

בפרק 4 הצגנו בדיקת נכונות עבור **דוגמאות קלט מגוונות**; כלומר דוגמאות קלט אשר מאפייניהן מבטאים את מגוון הקלטים האפשריים. עתה נחדד בנקודה של בדיקת נכונות, ונשתמש בדרך כלל במושג **דוגמאות קלט מייצגות**.

**דוגמאות קלט מייצגות** הן דוגמאות קלט אשר כל אחת מהן מייצגת קבוצת קלטים. **בחירה ממצה** של דוגמאות קלט מייצגות היא בחירה המבטאת חלוקה ממצה של הקלטים האפשריים לקבוצות מייצגות.

בפתרון הבעיה הבאה נמחיש בחירה ממצה של דוגמאות קלט מייצגות, ונראה שימוש בדוגמאות הקלט המייצגות כדי לזהות שגיאה בתוכנית ולתקנה.

### הצ'יה 1

**מטרת הבעיה ופתרונה:** הדגמה של חלוקת הקלטים לקבוצות אשר מאפייניהן שונים, בחירת דוגמת קלט מייצגת לכל קבוצה, ושימוש בדוגמאות הקלט המייצגות כדי לאמת את נכונותה של תוכנית וכדי לתקנה במידת הצורך.

נתונה התוכנית הבאה:

/\*

קלט: שתי אותיות אנגליות גדולות

פלט: הודעה אם האותיות עוקבות זו לזו

```

*/
using System;
public class Letters
{
    public static void Main()
    {
        char letter1, letter2;
        Console.WriteLine("Enter a capital letter: ");
        letter1 = char.Parse(Console.ReadLine());
        Console.WriteLine("Enter another capital letter: ");
        letter2 = char.Parse(Console.ReadLine());
        if (letter2 == letter1 + 1)
            Console.WriteLine("The letters are consecutive");
        else
            Console.WriteLine("The letters are not consecutive");
    } // Main
} // class Letters

```

ביצוע התוכנית יביא להצגת הודעת פלט נכונה רק עבור חלק מן הקלטים האפשריים.

חלקו את הקלטים האפשריים לקבוצות, בחרו דוגמת קלט מייצגת לכל קבוצה, ותארו עבור אילו דוגמאות תוצג הודעה נכונה ועבור אילו דוגמאות תוצג הודעה שגויה. אחר-כך, תקנו את התוכנית כך שעבור כל קלט תוצג הודעה נכונה.

**?** מכותרת התוכנית וממשפטי הפלט ניתן לראות שמטרת התוכנית היא לבדוק אם שתי האותיות הנתונות כקלט הן אותיות אנגליות עוקבות. מהי חלוקה מתאימה של הקלטים האפשריים לקבוצות?

מתאים לחלק את הקלטים האפשריים לשתי הקבוצות הבאות:

1. קלטים שהאותיות הנתונות בהם הן אותיות אנגליות עוקבות.
2. קלטים שהאותיות הנתונות בהם אינן אותיות אנגליות עוקבות.

**?** החלוקה המתוארת אכן מבטאת אבחנה בין קלטים אשר יש להם מאפיינים שונים. האם חלוקה זו מספיקה? האם לא קיימים קלטים בעלי מאפיינים שונים בקבוצות אלו?

למשל, הקלט B C שייך לקבוצה הראשונה, שהאותיות בה הן אנגליות עוקבות. גם הקלט C B שייך לקבוצה הראשונה, אך לשני קלטים אלה מאפיינים שונים. בקלט B C האות השנייה עוקבת לאות הראשונה, ואילו בקלט C B האות הראשונה עוקבת לאות השנייה.

**שימו ♥:** ניתן גם לחלק את קבוצת הקלטים השנייה באופן דומה, אך חלוקה זו איננה נחוצה: כיוון שהאותיות אינן עוקבות, הרי סדר הופעתן בקלט איננו משנה.

אם כך, נקבל את החלוקה הבאה של הקלטים האפשריים לשלוש קבוצות:

- א. האות השנייה עוקבת לאות הראשונה.
- ב. האות הראשונה עוקבת לאות השנייה.
2. שתי האותיות אינן עוקבות.

**?** חילקנו את הקלטים האפשריים חלוקה ממצה לקבוצות. מה תהיה בחירה ממצה של דוגמאות קלט מייצגות עבור הבעיה הנתונה?

בחירה ממצה של דוגמאות קלט מייצגות תהיה בחירת דוגמת קלט מכל קבוצה. למשל, קבוצת דוגמאות הקלט הבאות מבטאת בחירה ממצה: X Y (מייצגת את קבוצת הקלטים א1); E D (מייצגת את קבוצת הקלטים ב1); D O (מייצגת את קבוצת הקלטים 2)

? עבור אלו מדוגמאות הקלט המתוארות תוצג הודעת פלט נכונה?

עבור דוגמת הקלט הראשונה (X Y) והשלישית (D O) תוצג הודעת פלט נכונה. אך עבור דוגמת הקלט השנייה (E D) תוצג הודעת פלט שגויה! עבור דוגמה זו תוצג ההודעה:

The letters are not consecutive

הודעה שגויה תוצג, בעצם, עבור כל קלט מהקבוצה ב1, המיוצגת בדוגמת הקלט השנייה.

? כיצד ניתן לתקן את התוכנית, כך שתציג פלט דרוש עבור כל קלט חוקי? כלומר, כך שגם עבור קלטים מן הקבוצה השנייה תוצג הודעה מתאימה?

יש להרחיב את הביטוי הבוליאני שבמשפט ה-if. הביטוי הבוליאני שבתוכנית הנתונה כולל רק את האפשרות שהאות השנייה בקלט היא אות עוקבת לאות הראשונה בקלט. יש להרחיב ביטוי זה כך שיכלול גם את האפשרות שהאות הראשונה בקלט היא אות עוקבת לאות השנייה.

לכן, הביטוי הבוליאני שבתוכנית המתוקנת יהיה:

```
(letter2 == letter1 + 1) || (letter1 == letter2 + 1)
```

התוכנית המתוקנת תהיה:

```
/*
קלט: שתי אותיות אנגליות גדולות
פלט: הודעה אם האותיות עוקבות
*/
using System;
public class Letters
{
    public static void Main()
    {
        char letter1, letter2;
        // קלט
        Console.WriteLine("Enter a capital letter: ");
        letter1 = char.Parse(Console.ReadLine());
        Console.WriteLine("Enter another capital letter: ");
        letter2 = char.Parse(Console.ReadLine());
        if ( (letter2 == letter1 + 1) || (letter1 == letter2 + 1) )
            Console.WriteLine("The letters are consecutive");
        else
            Console.WriteLine("The letters are not consecutive");
    } // Main
} // class Letters
```

**סוף פתרון בעיה 1**

ניתן ללמוד לקח מן הפתרון לבעיה 1:

השגיאה שהופיעה בתוכנית הנתונה נובעת מכך שמפתח התוכנית לא ביצע ניתוח מלא של כל אפשרויות הקלט השונות. מפתח התוכנית לא הבחין בכך שהמשמעות של אותיות אנגליות עוקבות, איננה רק האפשרות ש"האות השנייה עוקבת לאות הראשונה", אלא גם האפשרות ש"האות הראשונה עוקבת לאות השנייה". זיהוי השגיאה ותיקונה התאפשר הודות לחלוקה ממצה של הקלטים האפשריים לקבוצות, והיא הביאה לבחירה ממצה של דוגמאות קלט מייצגות.

**שימו ♥:** הבדיקה בעזרת דוגמאות קלט מייצגות אינה **מוכיחה** נכונות, אלא מסייעת באיתור שגיאות, ומקטינה מאוד את ההסתברות לטעות, אך תמיד ייתכן (בייחוד בתוכניות גדולות מאוד) שנפספס תת-מקרה מסוים, משום שאין בידינו מתכון לקביעת חלוקה ממצה לקבוצות.

כיום בכל חברת תוכנה יש צוות בודקים אשר כל תפקידו הוא לוודא כי התוכנה עובדת כשורה עבור כל קלט אפשרי. חשיבות בדיקות אלה גבוהה ונועדה על מנת להימנע מהפסדים ולעתים גם מנזקים משמעותיים הרבה יותר, כמו פגיעה בחיי אדם (למשל בתוכנות רפואיות).

במרבית חברות התוכנה מערך הבדיקות מבוסס על סימולציה, כלומר על בחירה מתוככמת של דוגמאות קלט מייצגות ובדיקת התוכניות עליהן, בחירה שיכולה להיות מורכבת מאוד ומסובכת מאוד בתוכניות גדולות ומורכבות. עם זאת, יש גם חברות (בעיקר חברות לתכנון ולפיתוח חומרה) המשתמשות בכלי ה**הוכחה** (הנשענים על תורות מתמטיות). הסיבה שממעטים להשתמש בכלי ההוכחה היא שלכלים כאלה הקיימים כיום קשה להתמודד בצורה נוחה עם תוכניות גדולות. הסיבה שמשתמשים בהם יותר בתעשיית החומרה היא שהרבה יותר יקר לייצר מחדש רכיב חומרה אם מתגלית בו טעות אחרי שלב הייצור, מאשר לתקן ולהפיץ גרסה חדשה של תוכנה שהתגלתה בה טעות.

## שאלה 6.1

נתון הלוח הבא, ובכל משבצת בו מופיע מספר שלם:

1	2
3	4

קטע התוכנית הבא, אשר הקלט שלו הוא שניים מן המספרים המופיעים בלוח, יציג כפלט הודעה.

```
Console.Write("Enter first number: ");
num1 = int.Parse(Console.ReadLine());
Console.Write("Enter second number: ");
num2 = int.Parse(Console.ReadLine());
if (num1 == num2 + 2)
    Console.WriteLine("The numbers are in the same column");
else
    Console.WriteLine("The numbers are not in the same column");
```

א. בחרו שתי דוגמאות קלט, אשר עבור כל אחת מהן תוצג הודעה נכונה אחרת. תארו את המאפיינים של שתי קבוצות הקלטים שהדוגמאות שבחרתם שייכות אליהן.

ב. בחרו דוגמת קלט שעבורה תוצג הודעה שגויה, ותארו את המאפיין של קבוצת הקלטים שעבורם תוצג הודעה שגויה.

ג. שנו את הביטוי הבוליאני שבקטע התוכנית לביטוי בוליאני אחר, שעבורו תוצג תמיד הודעה מתאימה. נסחו את הביטוי החדש כביטוי פשוט.

## שאלה 6.2

הקלט לקטע התוכנית הבא הוא מספר שלם כלשהו. מטרת קטע התוכנית הבא היא להציג כפלט ערך שסימנו כסימן המספר שנקלט וגודלו הוא ריבוע המספר שנקלט. למשל עבור הקלט 10 הפלט הנדרש הוא 100, ועבור הקלט 10- הפלט הנדרש הוא -100.

**שימו ⚡:** בקטע התוכנית נעשה שימוש בפעולת החזקה Pow של המחלקה המתמטית Math. הפעולות המתמטיות הוצגו בפרק 4.

```
num = int.Parse(Console.ReadLine());
s = Math.Pow(num, 2);
Console.WriteLine(s);
```

קטע התוכנית שגוי.

- א. תנו דוגמת קלט מייצגת שעבורה יתקבל הפלט הדרוש.
- ב. תנו דוגמת קלט מייצגת שעבורה לא יתקבל הפלט הדרוש.
- ג. תארו את קבוצת הקלטים שעבורם לא יתקבל הפלט הדרוש.
- ד. תקנו את קטע התוכנית, כך שעבור כל קלט יתקבל הפלט הדרוש.

## שאלה 6.3

נתון קטע התוכנית הבא אשר הקלט שלו הוא ארבע אותיות אנגליות גדולות.

```
letter1 = char.Parse(Console.ReadLine());
letter2 = char.Parse(Console.ReadLine());
letter3 = char.Parse(Console.ReadLine());
letter4 = char.Parse(Console.ReadLine());
if ((letter1 == letter2) && (letter3 == letter4))
    Console.WriteLine("All letters are equal");
else
    Console.WriteLine("Not all letters are equal");
```

- א. תנו שתי דוגמאות קלט שונות אשר עבור כל אחת מהן תתקבל הודעה שגויה.
- ב. תארו את קבוצת הקלטים שעבורם תתקבל הודעה שגויה.
- ג. שנו את הביטוי הבוליאני שבקטע התוכנית כך שעבור כל קלט תתקבל הודעה נכונה.

## שאלה 6.4

לפניכם תוכנית בשפת C#. התוכנית קולטת תו כקלט. מטרת התוכנית היא לבדוק האם התו מייצג מספר. אם כן, יוצג המספר הבא אחריו כפלט. אחרת תוצג הודעה כי התו אינו מספר. הראו **שלוש** דוגמאות קלט מייצגות לתוכנית זו והסבירו עבור כל דוגמה איזו קבוצה היא מייצגת. מצאו את השגיאה בתוכנית ותקנו אותה.

```
/*
התוכנית קולטת תו
אם התו מייצג מספר, התוכנית מציגה את המספר הבא אחריו
בכל מקרה אחר מוצגת ההודעה "לא מספר"
*/
using System;
public class NextNumber
{
    public static void Main ()
    {
        // הגדרת משתנים
        char character; // תו הקלט
        // קלט
```



```

Console.Write("Enter a character: ");
character = char.Parse(Console.ReadLine());
// פלט
if (character >= '0' && character <= '9')
    Console.WriteLine( (char)(character + 1) );
else
    Console.WriteLine("Not a number");
} // Main
} // class NextNumber

```

ראינו עד כה ניתוח של תוכנית נתונה שמטרתה ברורה. אך בפיתוח תוכנה קורה לא פעם שיש לשלב תוכניות אשר אינן מתועדות כראוי ומטרתן איננה ברורה. במקרים כאלו יש לזהות קודם כל את מטרת התוכנית הנתונה.

**זיהוי מטרת תוכנית נתונה** מתבצע על ידי כך שנבחן את פלט התוכנית עבור דוגמאות קלט שונות, ונקבע את היחס בין הקלט לפלט.

נדגים זאת בעזרת שתי השאלות הבאות.

### שאלה 6.5

נתון קטע התוכנית הבא שהקלט שלו הוא מספר לא שלילי, והמשתנים בו הם מטיפוס ממשי. **שימו ♥**: בקטע התוכנית נעשה שימוש בפעולות Sqrt ו-Floor של המחלקה המתמטית Math שהוצגה בפרק 4. הפעולה Floor מקבלת מספר ממשי ומחזירה את החלק השלם שלו (למשל, ערך הביטוי Math.Floor(5.8) שווה ל-5.0)

```

num = double.Parse(Console.ReadLine());
sqrtNum = Math.Sqrt(num);
fraction = sqrtNum - Math.Floor(sqrtNum);
if (fraction > 0)
    Console.WriteLine("1");
else
    Console.WriteLine("0");

```

- א. תנו דוגמת קלט שעבורה יהיה הפלט 1.
- ב. תנו דוגמת קלט שעבורה יהיה הפלט 0.
- ג. תארו את מטרת קטע התוכנית וחלקו את הקלטים האפשריים לשתי קבוצות. תארו את הפלט עבור הקלט בכל קבוצה.

### שאלה 6.6

נתון קטע התוכנית הבא, שהקלט שלו הוא ארבעה מספרים שלמים שונים. כל המשתנים בקטע התוכנית הם מטיפוס שלם.

```

num1 = int.Parse(Console.ReadLine());
num2 = int.Parse(Console.ReadLine());
num3 = int.Parse(Console.ReadLine());
num4 = int.Parse(Console.ReadLine());
if (num1 > num2)
    max1 = num1;
else
    max1 = num2;
if (num3 > num4)
    max2 = num3;

```

```

else
    max2 = num4;
if (max1 > max2)
    Console.WriteLine(max1);
else
    Console.WriteLine(max2);

```

א. מהו הפלט עבור כל אחת מדוגמאות הקלט הבאות: 10 20 30 40 ו-50 30 40 20.

ב. תנו שלוש דוגמאות קלט שונות של מספרים חיוביים שהפלט עבורן הוא 5.

ג. מהי מטרת קטע התוכנית?

## שאלה 6.7

לפניכם תוכנית בשפת C#:

```

/* התוכנית קולטת שלושה מספרים שלמים
   */
using System;
public class What
{
    public static void Main ()
    {
        // הגדרת משתנים
        int num1, num2, num3; // משתני הקלט
        int temp;             // משתנה עזר
        // קלט
        Console.Write("Enter first number: ");
        num1 = int.Parse(Console.ReadLine());
        Console.Write("Enter second number: ");
        num2 = int.Parse(Console.ReadLine());
        Console.Write("Enter third number: ");
        num3 = int.Parse(Console.ReadLine());
        if (num1 > num2)
        {
            temp = num1;
            num1 = num2;
            num2 = temp;
        }
        if (num2 > num3)
        {
            temp = num2;
            num2 = num3;
            num3 = temp;
        }
        // פלט
        Console.WriteLine("{0},{1},{2}", num1, num2, num3);
    } // Main
} // class What

```

א. כתבו במשפט אחד מהי לדעתכם מטרת התוכנית.

ב. הציעו חלוקה ממצה של הקלטים האפשריים לקבוצות.

ג. בחרו דוגמאות קלט מייצגות על פי החלוקה שהצעתם בסעיף א', וציינו את הפלט עבור כל אחת מהן.

ד. האם התוכנית משיגה את מטרתה? אם כן, הסבירו מדוע; אם לא, תקנו אותה.

## **סיכום**

בפרק זה הרחבנו והעמקנו בנושא נכונות של אלגוריתם.

**אלגוריתם נכון** הוא אלגוריתם אשר ביצעו מביא להשגת המטרה עבור **כל** קלט חוקי (המתאים להגדרת הבעיה).

לא מתקבל על הדעת להיווכח בנכונות של אלגוריתם על ידי בניית טבלת מעקב עבור כל קלט אפשרי.

לכן אנו בוחרים בצורה ממצה דוגמאות מייצגות של קלט ובודקים את מהלך ביצוע האלגוריתם עבור דוגמאות קלט אלה.

**דוגמאות קלט מייצגות** הן דוגמאות קלט אשר כל אחת מהן מייצגת קבוצת קלטים בעלת מאפיינים שונים.

**בחירה ממצה** של דוגמאות קלט מייצגות היא בחירה המבטאת חלוקה ממצה של הקלטים האפשריים לקבוצות מייצגות.

לפעמים נתונים תוכניות או קטעי תוכניות אשר מטרתם אינה ברורה. **זיהוי מטרת תוכנית נתונה** נעשה לפי בחינת פלט התוכנית עבור דוגמאות קלט שונות, ולפי הכללת היחס בין הקלט לפלט.

## פרק 7 – ביצוע-חוזר

עד כה הכרנו בעיות אשר לשם פתרון ביצענו מספר תת-משימות שונות, באופן סדרתי. כלומר כל תת-משימה בסדרה בוצעה פעם אחת (ואם זו משימה שביצועה תלוי בתנאי, ייתכן שלא בוצעה אפילו פעם אחת). אולם יש בעיות אשר לצורך פתרון יש לבצע תת-משימה אחת, או כמה תת-משימות, יותר מפעם אחת, ואולי אף מספר רב של פעמים. בפרק זה נכיר אלגוריתמים אשר מורים על חזרה שוב ושוב על ביצוע של תת-משימה (או תת-משימות). אלגוריתמים אלה כוללים הוראה לביצוע-חוזר של קבוצת הוראות.

בסעיף 7.1 נכיר אלגוריתמים שמבנה הביצוע-החוזר בהם הוא פשוט. באלגוריתמים אלה מספר הפעמים של הביצוע-החוזר נקבע לפני תחילת ביצועו.

בסעיף 7.4 נכיר אלגוריתמים שמבנה הביצוע-החוזר בהם מורכב יותר. באלגוריתמים אלה מספר הפעמים של הביצוע-החוזר לא נקבע מראש, אלא תלוי בתנאי אשר נבדק שוב ושוב במהלך הביצוע-החוזר.

### 7.1 ביצוע-חוזר מספר פעמים ידוע מראש

#### קצ'ה 1

**מטרת הבעיה ופתרונה:** הצגת אלגוריתם הכולל הוראה לביצוע-חוזר מספר פעמים ידוע מראש.

עלינו להמיר רשימת מחירים מייצוג בדולרים לייצוג בשקלים. פתחו וישמו אלגוריתם אשר הקלט שלו הוא שער ההמרה מדולרים לשקלים ואחריו רשימה של עשרה מחירים הנתונים בדולרים. הפלט שלו הוא הערך בשקלים של כל אחד מעשרת המחירים. הפלט עבור כל מחיר צריך להינתן מיד אחרי קליטתו ולפני קליטת המחיר הבא.

#### פירוק הבעיה לתת-משימות

1. קליטת שער ההמרה מדולרים לשקלים
2. קליטת כל אחד מעשרת המחירים בדולרים, חישוב ערכו בשקלים והצגת הערך המחושב

**?** ניתן לפרט את התת-משימה השנייה. כיצד?

התת-משימה השנייה מורכבת בעצם מביצוע-חוזר, עשר פעמים, של התת-משימות הבאות:

- 2.1 קליטת מחיר בדולרים
  - 2.2 חישוב ערכו של המחיר בשקלים
  - 2.3 הצגה של הערך המחושב
- דרך אחת להורות על ביצוע החוזר עשר פעמים על התת-משימות שניסחנו, היא כמובן לכתוב אותן עשר פעמים, כך:

- 2.1 קליטת מחיר בדולרים
- 2.2 חישוב ערכו של המחיר בשקלים
- 2.3 הצגה של הערך המחושב
- 2.4 קליטת מחיר בדולרים
- 2.5 חישוב ערכו של המחיר בשקלים
- 2.6 הצגה של הערך המחושב

- 2.28. קליטת מחיר בדולרים
- 2.29. חישוב ערכו של המחיר בשקלים
- 2.30. הצגה של הערך המחושב

זהו ניסוח מסורבל כמובן. עבור רשימה של עשרה מחירים נכתבות שלושים הוראות. עבור רשימה של מאה מחירים ייכתבו 300 הוראות. ובעצם, עבור רשימות מחירים באורכים שונים ייכתבו אלגוריתמים שבהם מספר הוראות שונה. כלומר, לא רק שמדובר באלגוריתמים ארוכים מאוד, אלא שעבור שינוי קטן בהגדרת הבעיה (מספר המחירים), יש צורך לבצע שינוי משמעותי באלגוריתם.

האם ניתן להימנע מן הסרבול המתואר? האם ניתן לכתוב אלגוריתם שבו יהיה אותו מספר הוראות עבור רשימות מחירים באורכים שונים?

אכן ניתן באמצעות הוראה לביצוע-חוזר לקבוצת הוראות. בפתרון הבעיה הנוכחית שבה יש להמיר עשרה מחירים ניתן להשתמש בהוראה הבאה לביצוע-חוזר:

**כִּצְע 10 פִּעֻמִּים:**  
 קאוט מגיב כִּדִּילִימ  
 אַשְׁב אַת עֵרְכּוֹ שֶׁל הַמַּחִיר בִּשְׁקִלִים  
 הִצַּג אַת הָעֵרֶךְ הַמְחֻשָּׁב

עבור רשימה של 100 מחירים ניתן לכתוב אלגוריתם הכולל הוראה לביצוע-חוזר במבנה זהה, אלא שמספר הפעמים המצוין בו בכותרת ההוראה הוא 100 במקום 10.

**שימו** ♥ להזחה בהוראה לביצוע-חוזר. בהוראה זו (כמו בהוראה לביצוע-בתנאי) אנו מזיחים פנימה את קבוצת ההוראה לביצוע-חוזר.

## בחירת משתנים

נשתמש במשתנים הבאים מטיפוס ממשי:

rate – ישמור את שער ההמרה מדולרים לשקלים.

dollarPrice – ישמור מחיר בדולרים

shekelPrice – ישמור את ערכו בשקלים של המחיר השמור ב-dollarPrice

## האלגוריתם

1. קאוט שער המרה ב-rate
2. כִּצְע 10 פִּעֻמִּים:
  - 2.1 קאוט מגיב ב-dollarPrice
  - 2.2 אַשְׁב אַת עֵרְכּוֹ בִּשְׁקִלִים שֶׁל הַמַּחִיר הַשְּׂמֹר ב-dollarPrice וְהַשֵּׁם ב-shekelPrice
  - 2.3 הִצַּג אַת עֵרְכּוֹ שֶׁל shekelPrice

## יישום האלגוריתם

הוראה לביצוע-חוזר במבנה של **for** מיושמת ב-C# במשפט **for**. משפט **for** משתמש במשתנה **בקה**, אשר שולט בביצוע הלולאה. למשל אם ברצוננו לבצע קבוצת הוראות 10 פעמים, נכתוב משפט **for** בצורה הבאה:

```
for (i = 1; i <= 10; i++)
{
    ההוראות לביצוע
}
```

משתנה הבקה במשפט זה הוא **i**. ערכו מאותחל ב-1 (כפי שמורה משפט ההשמה **i = 1**, המהווה את הרכיב הראשון בסוגריים). אחרי שסדרת ההוראות לביצוע מתבצעת פעם אחת ערכו של משתנה הבקה גדל ב-1 (על כך מורה הרכיב השלישי בסוגריים: **i++**), והביצוע-החוזר יימשך כל עוד ערכו של **i** קטן או שווה ל-10 (כפי שמורה התנאי **i <= 10**, הרכיב השני בסוגריים).

אם כך בתחילת הביצוע של משפט ה-**for**, **i** יאותחל ב-1. אחרי שקבוצת ההוראות לביצוע תבוצע פעם אחת, ערכו יגדל ל-2. אחרי שקבוצת ההוראות תבוצע פעם שנייה, ערכו יגדל ל-3. אחרי שקבוצת ההוראות תבוצע פעם עשירית, ערכו כבר יהיה 11, ואז יסתיים הביצוע-החוזר, משום שערכו של התנאי **i <= 10** יהיה **false**.

באופן כללי, בביצוע משפט **for** מושם ערך התחלתי במשתנה הבקה לפי הרכיב הראשון במשפט. לאחר מכן מתבצעת בדיקת התנאי, המתואר ברכיב השני. אם ערכו של התנאי הוא **true** מתבצעות ההוראות לביצוע. בתום ביצוע קבוצת ההוראות גדל ערכו של משתנה הבקה לפי הרכיב השלישי. כעת מתבצעת שוב בדיקת התנאי. אם ערכו של התנאי הוא **true** מתבצעת שוב קבוצת ההוראות וכך הלאה, עד אשר ערכו של התנאי הוא **false** ואז מסתיים הביצוע.

באופן דומה, ניתן היה לבחור גם במשפט ה-**for** הבא ליישום ההוראה לביצוע-חוזר באלגוריתם:

```
for (i = 0; i < 10; i++)
{
    הוראות לביצוע
}
```

גם במקרה זה ההוראות לביצוע מתבצעות 10 פעמים: פעם אחת כאשר ערכו של **i** שווה ל-0, פעם שנייה כאשר ערכו שווה ל-1, פעם שלישית כאשר ערכו שווה ל-2, ובפעם העשירית ואחרונה כאשר ערכו של **i** שווה ל-9. כאשר ערכו של **i** גדל שוב, ומגיע ל-10, התנאי להמשך הביצוע כבר לא מתקיים, והביצוע-החוזר מסתיים.

**שימו ♥ :** ההוראה **i++** היא למעשה הוראת השמה מקוצרת. היא שקולה להוראת ההשמה: **i = i + 1**. ניתן להשתמש בהוראה זו בכל מקום בתוכנית, לאו דווקא במשפט **for**.

## התוכנית המלאה

```
/*
הקלט: שער ההמרה מדולרים לשקלים ורשימה של 10 מחירים בדולרים
הפלט: הערכים השקליים של 10 המחירים הנתונים בדולרים
*/
using System;
public class Convertor
{
    public static void Main ()
    {
        קבוע: מספר המחירים הנקראים מהקלט // HOW_MANY=10; const int
    }
}
```

```

double rate;           //שער ההמרה
double dollarPrice;    //מחיר בדולרים
double shekelPrice;    //מחיר בשקלים
int i;                 //משתנה בקרה
// קלט
1. Console.Write("Enter the rate: ");
2. rate = double.Parse(Console.ReadLine());
   // ההוראה לביצוע-חוזר
3. for (i = 1; i <= HOW_MANY; i++)
   {
3.1. Console.Write("Enter price in Dollars: ");
3.2. dollarPrice = double.Parse(Console.ReadLine());
3.3. shekelPrice = dollarPrice * rate;
3.4. Console.WriteLine("Price in Shekels is {0}",
                                                                    shekelPrice);
   } // for
   } // Main
} // Convertor

```

## מעקב

בטבלת מעקב הכוללת משפט `for` נכלול עמודה עבור משתנה הבקרה של המשפט. בתוכנית לפתרון בעיה 1, ערכו של משתנה הבקרה `i` גדל ב-1 אחרי כל ביצוע של קבוצת ההוראות הכלולה במשפט ה-`for`. ערכו בביצוע-החוזר הראשון הוא 1 וערכו בביצוע-החוזר האחרון הוא 10. בנוסף לכך, טבלת המעקב תכלול עמודה עבור התנאי הבוליאני שבכותרת המשפט.

נעקוב אחר ביצוע התוכנית עבור הקלט הבא: שער ההמרה הוא 3, ועשרת המחירים להמרה הם:  
10.1 5 18.2 120.3 200.01 50.3 60 71.05 61.03 100

כדי להימנע מהצגת טבלה ארוכה מדי, תכלול הטבלה הבאה מעקב רק אחרי עיבוד שני המחירים הראשונים והמחיר האחרון שבקלט.

	המשפט לביצוע	i	i<=10	rate	dollar	shekel	פלט
1	Console.Write("Enter the rate:");	?		?	?	?	Enter the rate
2	rate = double.Parse(Console.ReadLine());	?		3	?	?	
3	for (i = 1; i <= 10; i++)	1	true	3	?	?	
3.1	Console.Write("Enter price in Dollars: ");	1		3	?	?	Enter price ...
3.2	dollarPrice = double.Parse(Console.ReadLine());	1		3	10.1	?	
3.3	shekelPrice = dollarPrice * rate;	1		3	10.1	30.3	
3.4	Console.WriteLine("Price in Shekels is {0}",shekelPrice);	1		3	10.1	30.3	Price in Shekels is 30.3
3	for (i = 1; i <= 10; i++)	2	true	3	10.1	30.3	
3.1	Console.Write("Enter price in Dollars: ");	2		3	10.1	30.3	Enter price ...
3.2	dollarPrice = double.Parse(Console.ReadLine());	2		3	5	30.3	
3.3	shekelPrice = dollarPrice *	2		3	5	15	

	rate;						
3.4	Console.WriteLine("Price in Shekels is {0}", shekelPrice);	2		3	5	15	Price in Shekels is 15
.							
3	<b>for</b> (i = 1; i <= 10; i++)	10	<b>true</b>	3	61.03	183.09	
3.1	Console.Write("Enter price in Dollars: ");	10		3	61.03	183.09	Enter price ...
3.2	dollarPrice = <b>double</b> .Parse(Console.ReadLine());	10		3	100	183.09	
3.3	shekelPrice = dollarPrice * rate;	10		3	100	300	
3.4	Console.WriteLine("Price in Shekels is {0}", shekelPrice);	10		3	100	300	Price in Shekels is 300
3	<b>for</b> (i = 1; i <= 10; i++)	11	<b>false</b>	3	100	300	

שימו לב לשינויים שחלים בערכו של משתנה הבקרה i ולבדיקת התנאי של משתנה הבקרה.

### סוף פתרון קציה 1

נציג את המושגים החדשים שהכרנו בפתרון לבעיה 1.

באלגוריתם לפתרון הבעיה כללנו הוראה לביצוע-חוזר במבנה כזה **לולאה** כדי לציין ביצוע-חוזר של תת-משימה. הוראה זו מורה על ביצוע-חוזר של קבוצת הוראות מספר פעמים. הוראה לביצוע-חוזר היא הוראת בקרה. הוראה לביצוע-חוזר נקראת גם **לולאה** (loop), וקבוצת ההוראות לביצוע הכלולות בה נקראת **גוף הלולאה**. בדומה לכתיבה של הוראה לביצוע-בתנאי גם כתיבה של הוראה לביצוע-חוזר נעשית תוך הקפדה על **הזחה** מתאימה: קבוצת ההוראות שיש לחזור על ביצוען מוזחת פנימה.

בשפת C# מיושמת הוראה לביצוע-חוזר במבנה זה במשפט **for**.

זהו המבנה הכללי של משפט **for** בשפת C#:

```
for (שינוי משתנה הבקרה; התנאי להמשך הביצוע; אתחול משתנה הבקרה)
{
    הוראות לביצוע
}
```

**אתחול משתנה הבקרה**: הוראת השמה הקובעת ערך התחלתי למשתנה הבקרה.

**התנאי להמשך הביצוע**: ביטוי בוליאני שמהווה את התנאי השולט בביצוע-החוזר. התנאי נבדק אחרי אתחול משתנה הבקרה. כמו כן הוא נבדק שוב בכל פעם שמסתיים ביצוע של קבוצת ההוראות-לביצוע. כל עוד התנאי מתקיים הביצוע-החוזר ממשיך. כאשר ערכו של התנאי הוא **false** הביצוע-החוזר מסתיים.

**שינוי משתנה הבקרה**: השינוי שחל במשתנה הבקרה בכל פעם שמסתיים שלב ביצוע נוסף. **גוף הלולאה** תחום בסוגריים מסולסלים. במקרה שגוף הלולאה מכיל משפט בודד אפשר להשמיט את הסוגריים.



למשל כך:

```
for (i = 0; i < 10; i++)  
    Console.WriteLine("*");
```

**בטבלת מעקב** אחר תוכנית הכוללת משפט **for** או כוללים עמודה עבור משתנה הבקרה ועמודה עבור התנאי להמשך הביצוע.

משתנה הבקרה במשפט **for** הוא בדרך כלל מטיפוס שלם. מאחר שבמקרים רבים תפקידו של משתנה הבקרה הוא רק לשלוט במשפט ה-**for**, ואין בו שימוש בחלקי התוכנית האחרים, שפת C# מאפשרת להצהיר על משתנה הבקרה בתוך הוראת ה-**for**, כלומר, נוכל לכתוב:

```
for (int i = 1; i <= 10; i++)
```

כאשר מצהירים על משתנה הבקרה בתוך משפט ה-**for**, אין אליו גישה מחוץ לתחום משפט ה-**for**.

### שאלה 7.1

בנו טבלת מעקב אחר מהלך ביצוע התוכנית **Converter** (לפתרון בעיה 1) עבור הקלט שבו שער ההמרה הוא 4.5 ועשרת המחירים להמרה הם:

5.05 18.01 17.03 20.9 101 105 213.05 16.1 17.2 18.3

פרטו בטבלה רק את השורות המתאימות לעיבוד שני הקלטים הראשונים ולעיבוד הקלט האחרון (בדומה לנעשה בפתרון בעיה 1).

### שאלה 7.2

נסחו עבור כל אחת מן הבעיות האלגוריתמיות הבאות קבוצת תת-משימות לביצוע-חוזר:

א. הקלט הוא 20 מרחקים הנתונים במיילים, והפלט הוא 20 המרחקים בקילומטרים (1 מייל = 1.6 קילומטר).

ב. הקלט הוא עשר אותיות מן הא"ב האנגלי השונות מהאות Z, והפלט הוא עשר האותיות שעוקבות לאותיות הנתונות.

ג. הקלט הוא 40 זוגות של ציונים (זוג ציונים עבור כל תלמיד), והפלט הוא רשימה של ארבעים מספרים: כל מספר הוא הממוצע של זוג הציונים המתאים לו.

### שאלה 7.3

לפניכם קטע תוכנית:

```
Console.WriteLine('X');  
for (i = 0; i < 10; i++)  
    Console.WriteLine("*");  
Console.WriteLine('X');
```

מהו פלט קטע התוכנית?

### שאלה 7.4

פתחו וישמו אלגוריתם אשר הקלט שלו הוא תו, והפלט שלו הוא שכפול של התו הנקלט חמישים פעמים. למשל, עבור הקלט A יהיה הפלט AAA...AA (חמישים פעמים). בשלב החלוקה לתת-משימות הקפידו על ניסוח תת-משימה לביצוע-חוזר.

### שאלה 7.5

פתחו וישמו אלגוריתם אשר הקלט שלו הוא 20 מספרים שלמים חיוביים דו-ספרתיים, והפלט שלו הוא סכום הספרות לכל אחד מהמספרים הנתונים.

למשל, אם הקלט הוא :

11 17 99 10 20 30 10 20 30 10 20 30 10 10 20 20 30 30 88 15

הפלט המתאים הוא :

2 8 18 1 2 3 1 2 3 1 2 3 1 1 2 2 3 3 16 6

בשלב החלוקה לתת-משימות הקפידו על ניסוח תת-משימה לביצוע-חוזר, ובשלב הבדיקה הקפידו על בדיקה מסודרת באמצעות טבלת מעקב.

בפתרון בעיה 1 מספר הפעמים לביצוע-חוזר נקבע עוד לפני תחילת ביצוע התוכנית. במקרים רבים, ייתכן כי מספר הפעמים לביצוע-חוזר ידוע לפני שמתחיל ביצועה של ההוראה לביצוע-חוזר, אך לא לפני תחילת ביצוע התוכנית. כלומר הוא תלוי בקלט. מקרה כזה מודגם בבעיה הבאה :

## הצ'יה 2

מטרת הבעיה ופתרונה: הצגת ביצוע-חוזר שאורכו נקבע על פי נתון קלט.

פתחו אלגוריתם שיקבל כקלט מספר שלם  $N$ , ויצג על המסך שורה של כוכביות באורך  $N$ .

### פירוק הבעיה לתת-משימות

- קליטת אורך לרשימת כוכביות
- הדפסת כוכביות לפי המספר הנקלט

### רשימת המשתנים

עד עתה השתמשנו במספרים קבועים בתנאי לסיום הביצוע-החוזר. בבעיה זו אנו נדרשים לקבל כקלט את מספר הפעמים שתבצע הלולאה, ולכן התנאי להמשך הביצוע יהיה תלוי בערכו של משתנה.

`numOfTimes` – מספר הפעמים שיש להציג כוכבית  
`i` – משתנה הבקרה של הלולאה

### האלגוריתם

- קלוט מספר שלם `numOfTimes`
- כצ `numOfTimes` כשמים:  
2.1. הצ \* \*

### יישום האלגוריתם

בפתרון בעיה 1, כאשר רצינו לבצע קבוצת משימות 10 פעמים יישמנו זאת באמצעות משפט `for` שכותרתו :

```
for(i = 1; i <= 10; i++)
```

(או באמצעות משפט שכותרתו `for(i = 1; i <= HOW_MANY; i++)`, ו-`HOW_MANY` הוגדר כקבוע שערכו 10).

כעת אנחנו רוצים לבצע את ההוראה 2.1 `numOfTimes` פעמים. לכן ניישם את ההוראה לביצוע-חוזר במשפט `for` שכותרתו :

```
for(i = 1; i <= numOfTimes; i++)
```

## התוכנית המלאה

```
/*
הקלט: מספר שלם
הפלט: שורה של כוכביות באורך הנתון כקלט
*/
using System;
public class Stars
{
    public static void Main ()
    {
        int numOfTimes;
        Console.Write("Enter a number please: ");
        numOfTimes = int.Parse(Console.ReadLine());
        for (int i = 1; i <= numOfTimes; i++)
            Console.Write('*');

        } // Main
    } // Stars
```

### סוף פתרון בעיה 2

בתוכנית Stars ראינו שימוש בנתון קלט לקביעת מספר הפעמים לביצוע לולאה. בפיתוח אלגוריתמים בהמשך נשתמש בכך פעמים רבות.

באמצעות הבעיה הבאה נכיר שני סוגים של משתנים המשמשים בפתרון בעיות רבות ותבנית העבודה איתם היא שימושית מאוד. כפי שנראה, תבנית העבודה עם משתנים כאלה משתמשת בהוראות לביצוע-חוזר.

### בעיה 3

מטרת הבעיה ופתרונה: הצגת מונה וצובר ואופן העבודה איתם.

פתחו וישמו אלגוריתם אשר הקלט שלו הוא מספר חיובי שלם, ולאחריו רשימה של מספרים ממשיים, שאורכה שווה לערך הקלט הראשון. הפלט שלו הוא ממוצע המספרים ברשימה ומספר המספרים ברשימה שערכם עולה על 50. למשל, עבור הקלט:

10 23.4 100 95 78 64.15 75 90.3 54.2 67 20

הפלט המתאים הוא: 8 66.7

משום שממוצע המספרים הוא 66.7, ו-8 מתוכם הם גדולים מ-50.

### פירוק הבעיה לתת-משימות

1. קליטת אורך הרשימה
2. קליטת רשימת הערכים, סיכום, ומניית מספר הערכים הגבוהים מ-50
3. חישוב ממוצע הערכים
4. הצגה כפלט של הממוצע שחושב ושל מספר הערכים הגבוהים מ-50

? כדי לחשב את הממוצע יש לסכם את הערכים הנתונים בקלט, ובנוסף יש למנות כמה מהערכים הנתונים בקלט הם גבוהים מ-50. ניתן לבצע הן את פעולת הסיכום והן את פעולת המנייה תוך כדי קריאת נתוני הקלט. כיצד ניתן לבטא זאת כהוראה לביצוע-חוזר?

ניתן לבצע את פעולת הסיכום בהוספת כל ערך שנקלט לסכום מצטבר, עוד לפני קריאת הערך הבא. בדומה, ניתן לבצע את פעולת המנייה בהשוואת כל ערך שנקלט ל-50, ובהגדלת מונה מתאים בכל פעם שהערך שנקלט עולה על 50. לכן, את תת-משימה 2 נוכל לבצע באמצעות ביצוע-חוזר של קבוצת ההוראות הבאה:

- 2.1. קליטת ערך
- 2.2. הוספת הערך שנקלט לסכום המצטבר
- 2.3. השוואת הערך שנקלט ל-50. אם גבוה מ-50, הגדלת ערכו של מונה למניית מספר הערכים הגבוהים מ-50.

## בחירת משתנים

מן התת-משימות לביצוע-חוזר ניתן להסיק שיש להשתמש במשתנה אשר יישמר בו ערך תורן שנקרא מהקלט, במשתנה שיצבור את סכום הערכים, ובמשתנה שימנה את מספר הערכים הגבוהים מ-50. לכן ניעזר במשתנים הבאים:

- length – מטיפוס שלם, ישמור את אורך רשימת המספרים
- num – מטיפוס ממשי, ישמור ערך תורן הנקרא מהקלט
- sum – מטיפוס ממשי, צובר שישמור את סכום הערכים
- average – מטיפוס ממשי, ישמור את ממוצע הערכים
- counterLarge – מטיפוס שלם, מונה שישמור את מספר הערכים הגבוהים מ-50

**שימו ⚡:** המשתנים average, num, sum הם מטיפוס ממשי, כיוון שערכי הקלט בבעיה הם ממשיים. לעומתם counterLarge הוא מטיפוס שלם כיוון שהוא משמש למנייה.

## יישום האלגוריתם

את ההוראה לביצוע-חוזר נוכל לנסח באופן הבא:

- כצעד length עשמים:**
1. קלט ערך ממשי ב-num
  2. הוסף את ערכו של num לסכום המצטבר השמור ב-sum
  3. אם ערכו של num גדול מ-50
  - 3.1. הגדל ב-1 את ערכו של counterLarge

**?** אילו הוראות יש להוסיף לפני ההוראה לביצוע-חוזר?

**שימו ⚡:** כדי שהצבירה והמנייה יתבצעו באופן נכון, עלינו לאתחל נכונה את הערכים של sum ושל counterLarge. על פי תפקידם של שני המשתנים באלגוריתם יש לאתחל את ערכיהם ב-0.

## התוכנית המלאה

```
/*
קלט: רשימת ערכים ממשיים
פלט: ממוצע הערכים ומספר הערכים הגבוהים מ-50
*/
using System;
public class CalcAvgAndCountLargerThan50
{
    public static void Main ()
    {
        // הצהרה על קבוע בתוכנית
        const int LIMIT = 50;
```

```
// הצהרה על משתנים בתוכנית
int length; // אורך רשימת הקלט
double num; // ערך קלט תורן
double sum = 0; // צובר
double average; // ממוצע
int counterLarge = 0; // מונה למספר הערכים הגבוהים מ-50
Console.WriteLine("Enter length of input list: ");
length = int.Parse(Console.ReadLine());
// ההוראה לביצוע-חוזר
for (int i = 1; i <= length; i++)
{
    Console.WriteLine("Enter number: ");
    num = double.Parse(Console.ReadLine());
    sum = sum + num;
    if (num > LIMIT)
        counterLarge++; // counterLarge=counterLarge+1
} // for
average = sum / length;
Console.WriteLine("Average is {0}", average);
Console.WriteLine("{0} numbers are larger than {1}",
    counterLarge, LIMIT);

} // Main
} // CalcAvgAndCountLargerThan50
```

### סוף פתרון בעיה 3

באלגוריתם לפתרון בעיה 3 משמש המשתנה **sum** כצובר של הערכים הנתונים.

**צובר** הוא משתנה אשר תפקידו לצבור ערכים. למשל ניתן להשתמש בצובר לסכימת ערכים.

המשתנה counterLarge משמש כ**מונה** של מספר הערכים הגבוהים מ-50 מבין הערכים הנתונים.

**מונה** הינו משתנה אשר תפקידו למנות את מספר הפעמים שהתרחש אירוע מסוים (למשל מספר הפעמים שנקרא נתון קלט). כיוון שהשימוש במונה הוא לספירה, מונה הוא משתנה **מטיפוס שלם**.

בתוכנית לפתרון בעיה 3 מופיעים המשפטים המבטאים את פעולות הצבירה והמנייה בגוף הלולאה לאחר משפט הקלט.

המשפט המבטא את **פעולת הצבירה** בגוף הלולאה הוא:

```
sum = sum + num;
```

המשפט המבטא את **פעולת המנייה** בגוף הלולאה הוא:

```
counterLarge++;
```

מאחר שתחזוקה של מונה או של צובר כוללת ביצוע של פעולות עדכון חוזרות ונשנות, נבצע בדרך-כלל פעולות צבירה ומנייה בגוף הלולאה.

**שימו** ♥ בשני המקרים גדל המשתנה המשמש כצובר או כמונה בערך כלשהו. במקרה של צובר הוא גדל בערך ששייך לקבוצת הערכים המצטברים. במקרה של מונה הוא גדל ב-1.

באלגוריתם שמשמשים בו בצובר או במונה יש לאתחל את הצובר או את המונה. בתוכנית לפתרון הבעיה מאותחלים הצובר sum והמונה counterLarge ב-0.

**אתחול של צובר או של מונה** הוא השמת ערך המתאים לתחילת תהליך הצבירה או המנייה. מונה מאותחל בדרך כלל ב-0. הערך ההתחלתי המתאים לצובר תלוי במהות הצבירה המתבצעת בו. למשל, צובר השומר סכום מצטבר יאותחל ב-0.

## שאלה 7.6

שנו את המשפטים הנמצאים בגוף הלולאה שבתוכנית לפתרון בעיה 3, כך שיחושב ממוצע הערכים הגבוהים מ-50.

## שאלה 7.7

לפעמים ניתן להשתמש בערכו של מונה לחישוב מספר הנתונים המאופיינים בצורה הפוכה לנתונים שמנינו. למשל, נניח שבבעיה 3 יש להציג גם את מספר הערכים **הקטנים או שווים ל-50**. דרך אחת לחישוב מספר זה היא באמצעות שימוש במונה נוסף `counterSmall` (נוסף ל-`counterLarge`), אשר ערכו יוגדל ב-1 בכל פעם שנקלוט ערך הקטן או שווה ל-50. אך בעצם אין צורך במונה נוסף. ניתן לבצע את החישוב בתום ביצוע הלולאה, באמצעות המונה `counterLarge` המופיע כבר בתוכנית. כיצד? הוסיפו לתוכנית את ההוראה או את ההוראות המתאימות.

## שאלה 7.8

ציינו עבור כל אחת מן הבעיות האלגוריתמיות הבאות אם נחוץ לפתרונה צובר, מונה או אף אחד מהשניים:

- א. קלט: רשימת מחירים, פלט: סך כל המחירים.
- ב. קלט: רשימת מחירים, פלט: מספר המחירים הגבוהים מ-100.
- ג. קלט: רשימת מחירים, פלט: מספר המחירים שמרכיב האגורות בהם שונה מ-0.
- ד. קלט: רשימת מספרים, פלט: הערך המוחלט של כל אחד מהמספרים.
- ה. קלט: רשימת מספרים, פלט: הממוצע של הערכים המוחלטים של המספרים.

## שאלה 7.9

יש לקלוט סדרה של תווים, אשר אורכה שמור במשתנה `listSize`, ולמנות את מספר התווים שהם אותיות גדולות (capital letters) בא"ב האנגלי. בחרו משתנים, כתבו הוראה לביצוע-חוזר אשר לפניה אתחול מתאים, וישמו אותה במשפט `for`.

## שאלה 7.10

מטרת קטע התוכנית הבא היא מניית מספר תווי קלט השונים מן האות A. המשתנים `counter` ו-`listSize` הם מטיפוס שלם והמשתנה `letter` הוא מטיפוס תווי.

```
int counter = _____ ;
for (int i = 1; i <= listSize; i++)
{
    Console.WriteLine("Enter a char: ");
    letter = char.Parse(Console.ReadLine());
    if (_____)
    {
        _____
    }
} // for
Console.WriteLine("There are {0} letters different than A", counter);
```

קטע התוכנית כולל לולאה.

א. כמה פעמים תתבצע הלולאה עבור הערך 10 ב-`listSize`?

- ב. כמה פעמים תתבצע הלולאה עבור הערך 1 ב-listSize?  
ג. השלימו את קטע התוכנית.

### שאלה 7.11

נתונה התוכנית הבאה:

```
/*
קלט: טור של 16 תווים מטופס ספורטוטו
פלט:
*/
using System;
public class Toto
{
    public static void Main ()
    {
        const int NUM_OF_GAMES = 16;
        int d = 0;
        char score;
        for (int i = 0; i < NUM_OF_GAMES; i++)
        {
            Console.Write("Enter the score: ");
            score = char.Parse(Console.ReadLine());
            if (score == 'X')
                d++;
        } // for
        Console.WriteLine(d);
    } // Main
} // Toto
```

קלט התוכנית הוא טור בטופס הספורטוטו. כלומר, 16 תווים שכל אחד מהם מציין תוצאת משחק (1, 2 או X).

- א. מהו הפלט עבור הקלט: 1 2 X 1 2 X 1 2 X 1 2 X 1 2 X 2?  
ב. האם התוכנית כוללת מונה או צובר? אם כן, מהו או מהם?  
ג. כמה פעמים תתבצע לולאת התוכנית?  
ד. הביאו דוגמת קלט אשר הפלט עבורה הוא 0.  
ה. הביאו דוגמת קלט אשר הפלט עבורה הוא 15.  
ו. מה מאפיין את הקלטים אשר הפלט עבורם הוא 1?  
ז. מהם ערכי הפלט האפשריים?  
ח. מהי מטרת התוכנית? בחרו שם משמעותי למשתנה d.

### שאלה 7.12

פתחו וישמו אלגוריתם אשר הקלט שלו הוא מספר המציין אורך של רשימה ואחריו רשימת מספרים ממשיים באורך הנתון. הפלט שלו הוא סכום החלקים השלמים של המספרים הממשיים הנתונים.

### שאלה 7.13

כתבו קטע תוכנית אשר מקבל כקלט מספר שלם num. התוכנית תגדיל num מספרים בתחום 10-50 ותבדוק כמה מספרים מבין המספרים שהוגרלו הם זוגיים.

### שאלה 7.14

כתבו קטע תוכנית שיגריל 10 מספרים תלת-ספרתיים. התוכנית תחשב ותדפיס את סכום המספרים האי-זוגיים בלבד מבין המספרים שהוגרלו.

בדוגמאות שראינו עד כה השתמשנו בצובר לצבירת סכום. ניתן להשתמש בצובר גם לצבירת מכפלה. כאשר צובר משמש לצבירת מכפלה הוא מאותחל בערך שונה מ-0. שימו לב לכך בשאלה הבאה:

### שאלה 7.15

פתחו וישמו אלגוריתם שיקבל כקלט רשימה של 50 מספרים ויצג כפלט את **מכפלתם** של המספרים הקטנים מ-10.

להעמקה בתבניות **מנייה** ו**צבירה** פנו לסעיף התבניות המופיע בסוף הפרק. בעיה 3 השתמשה גם בתבנית **ממוצע**. להעמקה בתבנית **ממוצע** עבור סדרה שיכולה להכיל יותר מ-3 ערכים פנו לסעיף התבניות המופיע בסוף הפרק.

## הצ'יה 4

**מטרת הבעיה ופתרונה:** עוד על הוראה לביצוע-חוזר מספר פעמים ידוע מראש: הצגת הוראה לביצוע-חוזר המטפלת בתחום של מספרים, והדגמת שימוש במשתנה הבקרה בתוך גוף הלולאה.

פתחו וישמו אלגוריתם שיקבל כקלט מספר שלם, ויצג כפלט את כל המספרים השלמים החיוביים הקטנים מהמספר הנתון. למשל עבור הקלט 5 הפלט המתאים הוא: 1 2 3 4.

### פירוק הבעיה לתת-משימות

ברצוננו לקלוט מספר שלם num ולהציג num-1 ערכים, מ-1 עד num-1. אם כך, החלוקה לתת-משימות היא ברורה למדי:

1. קליטת מספר מהקלט
  2. הצגה כפלט של כל הערכים החיוביים והשלמים שקטנים מהמספר שנקלט
- ברור כי לשם ביצוע תת-משימה 2 נזדקק להוראה לביצוע-חוזר, שתבצע num-1 פעמים.

### בחירת משתנים

ברור כי נזדקק למשתנה עבור הערך הנקרא מהקלט. סביר כי נהיה גם זקוקים למשתנה כלשהו, עבור האיבר התורן להצגה. משתנה זה יאותחל ב-1 ויקודם בסיום כל סיבוב בלולאה ב-1, עד אשר ערכו יגיע ל-num.

אבל איננו זקוקים למשתנה **נוסף** כי תיאור זה מתאים בדיוק למשתנה הבקרה של הלולאה!

לכן נקבל את רשימת המשתנים הבאה:

num – מטיפוס שלם, לשמירת הערך הנקרא מהקלט  
i – משתנה הבקרה



## האלגוריתם

כאמור תת-משימה 2 תתבצע באמצעות הוראה לביצוע-חוזר. למעשה בהוראה זו אנו מעוניינים לעבור על תחום של ערכים (מהערך 1 ועד הערך num-1) ולהציג כל ערך בתחום. כדי לבטא זאת, ננסח את ההוראה לביצוע-חוזר בצורה שונה מעט מזו שראינו באלגוריתמים קודמים בפרק.

1. קלוט מספר שלם num-2
2. עבור כל i שלם מ-1 עד num-1:
  - 2.1. הצג את ערכו של i

## יישום האלגוריתם

את הוראה 2 באלגוריתם שכתבנו ניישם במשפט for. זהו משפט for הדומה מאוד לאלה שראינו בתוכניות קודמות בפרק, רק שמשתנה הבקרה בו אינו משמש רק כדי לשלוט בביצוע הלולאה. יש התייחסות למשתנה הבקרה גם בתוך גוף הלולאה, ולא רק בשלושת הרכיבים שבכותרת הלולאה.

## התוכנית המלאה

```
/*
קלט: מספר שלם
פלט: כל המספרים השלמים והחיוביים הקטנים מהמספר הנתון
*/
using System;
public class PrintNumbers
{
    public static void Main()
    {
        int num; // ערך הנקרא מהקלט
        Console.Write("Enter a number: ");
        num = int.Parse(Console.ReadLine());
        for (int i = 1; i < num; i++)
            Console.WriteLine(i);
    } // Main
} // PrintNumbers
```

סוף פתרון בעיה 4

## שאלה 7.16

על פי הגדרת בעיה 4, יכול להינתן כנתון ראשון בקלט כל מספר שלם. תארו את מהלך ביצוע התוכנית PrintNumbers אם הערך הראשון בקלט הוא המספר השלם 0. תארו את מהלך הביצוע של התוכנית אם הערך הראשון בקלט הוא המספר השלם -2.

## שאלה 7.17

פתחו אלגוריתם שמקבל כקלט מספר חיובי שלם n, מחשב את n!, ומציג את הערך שחושב כפלט. ישמו את האלגוריתם בשפת C#. להזכירכם: n! היא מכפלת המספרים מ-1 עד n, כלומר:  $n! = 1 \cdot 2 \cdot \dots \cdot n$ .

## שאלה 7.18

פתחו אלגוריתם שמקבל כקלט מספר חיובי שלם n, ומחשב את הסכום:  $\frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{n}$ .

כזכור הרכיב השלישי במשפט `for` הוא משפט השמה המתאר את השינוי של משתנה הבקרה בתום כל סיבוב בלולאה. עד עתה כתבנו תוכניות שבהן משתנה הבקרה גדל ב-1 בתום כל סיבוב של הלולאה. אבל לעתים נרצה לקדם את משתנה הבקרה ב-2, 3 או אפילו פי 2. שפת C# מאפשרת לנו לתאר שינויים כאלה, פשוט בכתיבת משפט השמה מתאים. הנה מספר דוגמאות לכותרות משפטי `for` כאלה:

```
for(int i = small; i < big; i = i + 2)
for(int i = small; i < big; i = i * 2)
for(int i = big; i > small; i--)
```

(כפי שוודאי הבנתם, `i--` היא דרך מקוצרת לכתוב: `i = i - 1`). במקרים רבים, אכן מתאים לבצע עדכונים שונים של משתנה הבקרה, כפי שמדגימות השאלות הבאות.

### שאלה 7.19

לפניכם קטע תוכנית הכתוב ב-C#

```
for (int i = 1; i <= 50; i = i * 2)
    Console.WriteLine(i);
```

א. עקבו בעזרת טבלת מעקב אחר קטע התוכנית: מה יודפס?  
 ב. נניח כי במקום הערך 50 מופיע ערך חיובי שלם כלשהו  $N$ . תארו מה מבצע קטע התוכנית כתלות בערך  $N$ .

### שאלה 7.20

פתחו אלגוריתם אשר מקבל כקלט מספר שלם ומציג כפלט את כל המספרים הזוגיים החיוביים הקטנים מהמספר שנקרא.

## 7.2 מציאת מקסימום או מינימום

שתיים מהבעיות הבסיסיות ביותר במדעי המחשב הן חישוב הערך הגדול ביותר או הקטן ביותר ברשימה של ערכים נתונים. פתרון מהווה תבנית שימושית מאוד. בסעיף זה נערוך היכרות עם בעיות אלו ונראה כי כאשר אורך הרשימה ידוע ניתן לפתור אותן בעזרת הוראה לביצוע-חוזר מספר פעמים ידוע מראש.

### קצ'ה 5

**מטרת הבעיה ופתרונה:** הצגת אופן החישוב של הערך הגדול ביותר ברשימת ערכים נתונים שאורכה נתון אף הוא.

מנהל סניף הבנק החליט לבדוק מהו סכום הכסף הגדול ביותר השמור בחשבונות של לקוחות הסניף. פתחו אלגוריתם אשר קולט את מספר החשבונות בסניף, ולאחר מכן את סכום הכסף הנמצא בכל חשבון וחשבון (כמספר שלם). פלט האלגוריתם יהיה הסכום הגבוה ביותר. ישמו את האלגוריתם בשפת C#.

## ניתוח הבעיה בעזרת דוגמאות

הרעיון המרכזי בפתרון הבעיה הוא לזכור **בכל זמן** מהלך הביצוע, מהו הערך הגדול ביותר מבין אלה שכבר נקלטו. ערך זה ישווה תמיד לערך הבא שנקרא מהקלט, ויתעדכן בהתאם לתוצאת ההשוואה.

## פירוק הבעיה לתת-משימות

את הרעיון שתיארנו נוכל לבטא באמצעות התת-משימה הבאה ועל ביצועה יש לחזור לאחר כל קליטת ערך נוסף:

השוואת הסכום האחרון שנקלט לסכום הגדול ביותר שנקלט עד כה, ועדכון הסכום הגדול ביותר שנקרא עד כה על פי תוצאת ההשוואה

## בחירת משתנים

נזדקק למשתנה שיזכור את מספר ערכי הקלט (מספר החשבונות), ולמשתנה לשמירת הערך התורן בקלט. בנוסף נזדקק למשתנה כדי לזכור את הערך הגדול ביותר מבין אלה שנקלטו. נבחר את המשתנים הבאים מטיפוס שלם:

**howMany** – מספר הערכים ברשימת הנתונים, מספר החשבונות בסניף

**balance** – הערך התורן ברשימת ערכי הקלט

**max** – לשמירת הסכום הגדול ביותר מבין אלה שנקלטו

## האלגוריתם

**max** יאותחל בסכום הראשון ברשימה, כיוון שמיד אחרי שנקרא הסכום הראשון, הוא בוודאי הגדול ביותר מבין כל הסכומים שנקראו. כעת יש לקרוא את כל הסכומים האחרים בעזרת לולאת **for**. לאחר כל קליטה של סכום נוסף, יש להשוות את הסכום החדש שנקלט לסכום הגדול ביותר שנקרא עד כה (השמור ב-**max**). אם הסכום החדש גדול מזה השמור ב-**max**, יישמר ב-**max** הסכום החדש.

❓ כמה פעמים צריך להתבצע גוף הלולאה?

הסכום הראשון נקרא מהקלט עוד לפני הלולאה לצורך האתחול של המשתנה **max**. לכן הלולאה צריכה להתבצע רק **howMany-1** פעמים כדי לקלוט ולעבד את ערכי הקלט הנוותרים.

## יישום האלגוריתם

```
/*
קלט: מספר חשבונות בנק, ורשימת הסכומים בחשבונות אלה
פלט: הסכום הגבוה ביותר ברשימה
*/
using System;
public class FindMax
{
    public static void Main ()
    {
        int howMany;    // מספר הערכים ברשימה
        int balance;    // הסכום התורן
        int max;        // הסכום הגבוה ביותר מבין אלה שנקלטו
1.    Console.WriteLine("Enter the amount of accounts: ");
2.    howMany = int.Parse(Console.ReadLine());
    }
```

```

3. Console.WriteLine("Enter the first balance: ");
4. balance = int.Parse(Console.ReadLine());
5. max = balance; // אתחול המקסימום לסכום הראשון
6. for(int i = 2; i <= howMany; i++)
{
6.1. Console.WriteLine("Insert balance of account {0}: ", i);
6.2. balance = int.Parse(Console.ReadLine());
6.3. if (balance > max) // הסכום הנוכחי גדול מהמקסימום הנוכחי
6.3.1. max = balance;
} // for
7. Console.WriteLine("The maximum is {0}", max);
} // Main
} // FindMax

```

**שימו ♥** בתוך הלולאה אפשר להשתמש במשתנה הבקרה  $i$  כמשתנה לכל דבר.

## סוף פתרון בעיה 5

### שאלה 7.21

בנו טבלת מעקב אחר מהלך ביצוע התוכנית FindMax לפתרון בעיה 5 עבור הקלט:

5 3 2 4 6 9

כמה פעמים במהלך ביצוע התוכנית מושם ערך במשתנה  $max$ ?

### שאלה 7.22

בפתרון בעיה 5 אותחל  $max$  לערכו של הנתון הראשון ברשימה. האם הפתרון היה נכון לו  $max$  היה מאותחל בערך קבוע כלשהו, למשל 0? הערה: זכרו כי ייתכן שזהו בנק לא מוצלח במיוחד וכל חשבונות הבנק בו במשיכת יתר, כלומר בעלי ערך שלילי.

### שאלה 7.23

נתון קטע התוכנית החלקי הבא לחישוב המספר הקטן ביותר ברשימת מספרים חיוביים נתונה, אשר אורכה שמור במשתנה  $len$ .

```

min = _____
Console.WriteLine("Enter the list size: ");
len = int.Parse(Console.ReadLine());
for (int i = 1; i <= len; i++)
{
    _____
    _____
    _____
}
Console.WriteLine("The minimum is: {0}", min);

```

השלימו את קטע התוכנית (הוסיפו משתנה או משתנים במידת הצורך).  
במשפט הראשון אתחלו את  $min$  לערך קבוע כלשהו (ולא לערך הראשון ברשימה).

### שאלה 7.24

פתחו וישמו אלגוריתם שמקבל כקלט רשימה של ציונים במדעי המחשב של 40 תלמידים. הפלט הוא הציון הגבוה ביותר מבין התלמידים שנכשלו במבחן (ציון נכשל הוא ציון הנמוך מ-55). שימו לב: כאן הציון התורן הנקרא מהקלט אינו מושווה בכל מקרה למקסימום הנוכחי, אלא רק כאשר הוא קטן מ-55.

כפי שראינו בתרגילים הקודמים, יש דרכים שונות לאתחול בתבנית מציאת מקסימום (וכמובן, כך גם לגבי תבנית מציאת מינימום). המשתנה ששומר את המקסימום התורן, צריך בכל שלב לשמור את האיבר הגדול ביותר מבין האיברים שהושוו עד כה. בדרך כלל נעדיף לאתחול את המשתנה הזה בערך האיבר הראשון ברשימה. אכן אחרי שנקלט איבר אחד בלבד, ודאי שהוא הגדול מבין כל אלה שנקלטו. אך ישנם מקרים שעובדה זו אינה מספיקה ולא נוכל להשתמש באתחול כזה. כך למשל בשאלה 7.24: בשאלה זו אנו מעוניינים למצוא מקסימום רק מבין הציונים הנכשלים ולא מבין כל הציונים שבקלט. כלומר לא כל איבר שנקלט צריך להיות מושווה לצורך מציאת מקסימום. בפרט לא בטוח שהאיבר הראשון הוא ציון נכשל ולכן בכלל לא יהיה מועמד למקסימום. לכן יהיה שגוי לקבוע אותו כמקסימום התחלתי.

במקרים כאלה נוכל להשתמש בקצוות של טווח הערכים האפשריים. למשל אם מדובר בציונים, ערכם יכול לנוע מ-0 עד 100. במקרה כזה, יהיה נכון לאתחול את המקסימום התורן ב-0 (ואת המינימום התורן ב-100). כיוון שברור שהערך הראשון שנשווה למקסימום התורן ערכו הוא 0 לפחות, הרי מיד אחרי ההשוואה הראשונה המקסימום התורן יכיל את האיבר הראשון שהושווה.

**שימו ♥:** לא תמיד ידועים ערכי קצה לטווח הערכים האפשריים! (ראו את שאלה 7.22)

להעמקה בתבניות **מציאת מקסימום ומציאת מינימום** פנו לסעיף התבניות המופיע בסוף הפרק

## 7.3 מציאת ערך נלווה למקסימום או למינימום

בסעיף זה נכיר בעיות הדומות לבעיית מציאת מקסימום או מינימום, אך מעט יותר מורכבות. גם פתרונותיהן מהווים תבניות שימושיות.

### הצ'ה 6

**מטרת הבעיה ופתרונה:** הצגת אופן חישוב **מיקומו** של הערך הגדול ביותר והקטן ביותר ברשימת ערכים נתונים.

בנגן MP3 שמורים 100 שירים. לכל שיר מספר ייחודי משלו בין 1 ל-100. על צג המכשיר מוצג **מספרו** של השיר הארוך ביותר ושל השיר הקצר ביותר שנמצאים בו. עליכם לפתח אלגוריתם שיאפשר את הצגת המידע הזה. כלומר האלגוריתם יקבל כקלט את רשימת אורכי 100 השירים במכשיר, ופלט האלגוריתם יהיה **מספרו** של השיר הארוך ביותר ו**מספרו** של השיר הקצר ביותר.

כמובן שלצורך פתרון הבעיה יש למצוא את השיר הארוך ביותר ואת השיר הקצר ביותר. כלומר יש למצוא ברשימת אורכי השירים ערך מקסימלי וערך מינימלי. אבל בבעיה זו עלינו לשמור מלבד הערך המקסימלי ומלבד הערך המינימלי גם את **מיקומו** של ערכים אלה ברשימה.

### פירוק הבעיה לתת-משימות

את הרעיון המרכזי בפתרון נוכל לבטא באמצעות התת-משימה הבאה אשר על ביצועה יש לחזור לאחר כל קליטת ערך נוסף:

השוואת אורך השיר האחרון שנקלט לאורך השיר הגדול ביותר שנקלט עד כה ולאורך השיר הקצר ביותר שנקלט עד כה, ובהתאם לתוצאת ההשוואה עדכון ערך המקסימום ומיקומו ומיקומו המקסימום ועדכון ערך המינימום ומיקומו המינימום.

## בחירת משתנים

**currentSongLength** – מטיפוס ממשי, ישמור את אורך השיר הנוכחי.  
**longest** – מטיפוס ממשי, ישמור את אורך השיר הארוך ביותר שנקלט עד כה.  
**shortest** – מטיפוס ממשי, ישמור את אורך השיר הקצר ביותר שנקלט עד כה.  
**placeLongest** – מטיפוס שלם, ישמור את **מיקומו** של השיר הארוך ביותר שנקלט עד כה.  
**placeShortest** – מטיפוס שלם, ישמור את **מיקומו** של השיר הקצר ביותר שנקלט עד כה.

**שימו** ♥ לבחירת טיפוס המשתנים: המשתנים longest ו-shortest אשר שומרים את האורך המקסימלי ואת האורך המינימלי חייבים להיות מאותו טיפוס כמו הערכים ברשימת הקלט. כיוון שכאן מדובר באורכי שירים (שיכולים להיות לא שלמים), בחרנו בטיפוס ממשי. לעומתם המשתנים placeLongest ו-placeShortest שומרים מיקום ברשימה, ולכן הם מטיפוס שלם.

## התוכנית המלאה

? כיצד נדע מהו מיקומו של שיר נתון?

לשם כך נוכל להשתמש במשתנה הבקרה של משפט ה-for. כדי שהלולאה תתבצע 99 פעמים משתנה הבקרה יתקדם מ-2 עד 100, וכך ערכו של משתנה הבקרה יהיה בדיוק מספרו של השיר התורן.

**שימו** ♥ : בדיוק כפי שנאתחל את longest ואת shortest, נאתחל גם את placeLongest ואת placeShortest באופן עקבי. Longest יאותחל באורכו של השיר הראשון ולכן בהתאם placeLongest יאותחל במיקום הראשון (כלומר ב-1). כך גם לגבי האתחול של shortest ושל placeShortest.

```
/*
קלט: רשימת אורכי 100 השירים במכשיר MP3
פלט: מספרו של השיר הארוך ביותר, ומספרו של השיר הקצר ביותר
*/
using System;
public class Mp3
{
    public static void Main ()
    {
        const int NUM_OF_SONGS = 100; // מספר השירים בנגן
        double currentSongLength; // אורך השיר הנוכחי
        double shortest; // אורך השיר הקצר ביותר
        double longest; // אורך השיר הארוך ביותר
        int placeLongest, placeShortest; // מיקומם של השיר הקצר והארוך
        Console.WriteLine("Enter the length of the first song: ");
        currentSongLength = double.Parse(Console.ReadLine());
        // אתחולם של המקסימום, המינימום ומקומם
        longest = currentSongLength;
        shortest = currentSongLength;
        placeLongest = 1;
        placeShortest = 1;
        // הלולאה מתחילה מ-2 כיוון שהשיר הראשון כבר נקלט
        for(int i = 2; i <= NUM_OF_SONGS; i++)
        {
```

```

Console.WriteLine("Enter the length of song {0}: ", i);
currentSongLength = double.Parse(Console.ReadLine());
if (currentSongLength > longest)
{
    longest = currentSongLength;
    placeLongest = i;
} // if
if (currentSongLength < shortest)
{
    shortest = currentSongLength;
    placeShortest = i;
} // if
} // for
Console.WriteLine("The number of the longest song is {0}",
    placeLongest);
Console.WriteLine("The number of the shortest song is {0}",
    placeShortest);

} // Main
} // Mp3

```

## סוף פתרון בציה 6

### שאלה 7.25

פתחו אלגוריתם שיקבל כקלט מספר חיובי שלם המציין את מספר שחקני מכבי ת"א, ואחר כך יקלוט רשימה של נתוני קליעות של השחקנים: עבור כל שחקן ייקלט מספרו (המספר שעל החולצה שלו), ומספר הנקודות שקלע במהלך העונה. פלט האלגוריתם יהיה **מספרו** של השחקן שקלע הכי הרבה נקודות במהלך העונה.

**שימו ♥**: במקרה זה, אנחנו לא מתבקשים להציג את מיקומו של השחקן שקלע הכי הרבה נקודות אלא את מספרו, לכן משתנה הבקרה המצביע על ה"מיקום" של השחקן התורן בקלט אינו מתאים כי הוא אינו מספרו הסידורי של השחקן. את מספרו של השחקן יש לקרוא מהקלט.

להעמקה בתבניות **מציאת ערך נלווה למקסימום ומציאת ערך נלווה למינימום** פנו לסעיף התבניות המופיע בסוף הפרק.

## 7.4 ביצוע-חוזר-בתנאי

לצורך פתרון הבעיות שראינו עד כה אפשר היה להשתמש בהוראה לביצוע-חוזר אשר מספר הסיבובים בה נקבע **לפני** תחילת ביצוע הלולאה.

בסעיף זה נכיר בעיות אשר בפתרון אי אפשר לקבוע לפני תחילת הלולאה את מספר הפעמים לביצוע-חוזר. בפתרון בעיות אלה מהלך הלולאה נקבע בהתאם לתנאי. הלולאה מתבצעת שוב ושוב כל עוד התנאי מתקיים. הלולאה מסתיימת כאשר התנאי לא מתקיים.

## ביצוע-חוזר בשימוש בזקיף

בסעיף זה נראה בעיות דומות לאלו שראינו בסעיפים הקודמים. הדמיון מתבטא בכך שגם בבעיות אלו מתבצע עיבוד של רשימת נתוני קלט. ההבדל נעוץ באורך הרשימה המעובדת. בבעיות שהוצגו עד כה אורך הרשימה היה קבוע של התוכנית או נקרא מהקלט. בבעיות שיוצגו בסעיף זה אורך

הרשימה אינו ידוע כלל. במקום זאת האיבר האחרון בקלט הוא איבר מיוחד המסמן את סופה של רשימת ערכי הקלט.

## תצ"ה 7

**מטרת הבעיה ופתרונה:** הצגת הוראה לביצוע-חוזר-בתנאי בשימוש בזקיק.

צבי עובד בשעות אחר הצהריים בחלוקת משלוחי פרחים. הוא מעוניין להיעזר במחשב הנייד שהוא נושא עמו כדי לחשב בסיום יום העבודה את סכום הטיפים שהרוויח במהלך היום. בכל פעם שצבי מקבל טיפ הוא מקיש את הסכום שקיבל (בשקלים שלמים). בסיום יום העבודה הוא מקיש 1-. פתחו אלגוריתם המקבל קלט את רשימת הטיפים שקיבל צבי, המסתיימת בערך 1-, ומחשב את סכום הטיפים הכולל. ישמו את האלגוריתם בשפת C#.

## ניתוח הבעיה בעזרת דוגמאות

### שאלה 7.26

תארו את הפלט עבור כל אחד מן הקלטים הבאים (משמאל לימין):

א. 1- 9 7 10 12

ב. 1- 20

## פירוק הבעיה לתת-משימות

בניתוח ראשוני של הבעיה ניתן לראות שיש להשתמש בביצוע-חוזר של שתי התת-משימות:

1. קליטת מספר

2. הוספת המספר לסכום (בשימוש בצובר)

לו הקלט היה כולל בתחילתו את מספר הערכים בסדרה הנתונה, היינו מפתחים אלגוריתם דומה לאלגוריתמים שבסעיפים הקודמים, בשימוש בהוראה במבנה **עצם מספר פשוט**. אבל מספר הערכים בסדרה לא נתון בתחילת הקלט (כלומר אין אנו יודעים מראש כמה שלחיות עשה צבי באותו היום). במקום זה מופיע בסוף הקלט הערך 1- המציין "סוף קלט". כיצד נשתמש בסימון זה בהוראה לביצוע-חוזר?

נשתמש בערך 1- כדי להחליט על סיום ביצוע-חוזר של התת-משימות שתיארנו. כלומר לאחר קליטה של טיפ תורן, ישווה ערך הטיפ לערך 1-. אם הוא שונה מ-1- הרי הוא נתון קלט רגיל ולכן יש להוסיפו לסכום המצטבר. אחרת יש לסיים את הביצוע-החוזר.

## בחירת משתנים

tip – מטיפוס שלם, ישמור את הטיפ למשלוח הנוכחי.

sum – מטיפוס שלם, ישמור את סכום הטיפים.

## האלגוריתם

ננסח את הרעיון שתיארנו:

**כאשר הערך האגרון שנקלט אינו 1-:**  
הוסיף את הערך האגרון שנקלט לסכום המצטבר  
העבר את ערכו של הסכום המצטבר



בהוראה לביצוע-חוזר מסוג זה נבדק תנאי לפני כל סיבוב בלולאה. במקרה זה התנאי הוא **הענין האגיון שנקלט אינו 1-**. אם התנאי מתקיים מתבצע סיבוב נוסף בלולאה. כאשר התנאי לא מתקיים, כלומר הערך שנקלט הוא אכן 1-, מסתיים הביצוע-החוזר.

**?** בין ההוראות האלו לא נמצאת עדיין הוראה לקליטת הטיפ התורן. היכן נמקם את הוראת הקלט?

יש לבצע את קליטת הטיפ התורן לפני השוואתו לסימן 1-. לכן את הערך הראשון יש לקרוא מהקלט עוד לפני הלולאה. קליטת הערכים הבאים (עד לקליטת סימן סוף הקלט) תהיה ההוראה האחרונה בגוף הלולאה. כלומר מיד כאשר מסתיים עיבוד איבר קלט תורן, ולפני שנבדק קיום התנאי ביחס לאיבר הקלט החדש, מתבצעת קליטה של איבר הקלט החדש.

הנה האלגוריתם המלא:

```

1. אגיון אג המשלוח sum 0-2
2. קלוט מספר שלם 2-tip
3. כל ערך 1- tip ≠ 2:
3.1. הוסף 1-sum אג הענין tip
3.2. קלוט מספר שלם 2-tip
4. הצג אג ערכו של sum

```

## יישום האלגוריתם

את ההוראה במבנה **כל ערך ... 2:3**, ניישם ב-C# במשפט **while**. משפט **while** כולל תנאי בוליאני שקיומו קובע את המשך ביצוע הלולאה.

```

/* קלט: סדרת מספרים שלמים חיוביים המסתיימת ב-(-1) */
/* פלט: סכום של המספרים שנקלטו */
using System;
public class SumOfTips
{
    public static void Main ()
    {
        int tip;           // הטיפ מהמשלוח הנוכחי
        int sum;           // סכום הטיפים המצטבר
1.      sum = 0;
2.      Console.WriteLine("Enter your first tip for today." +
                           " End the list of tips with -1: ");
3.      tip = int.Parse(Console.ReadLine());
4.      while (tip != -1)
        {
4.1.      sum = sum + tip;
4.2.      Console.WriteLine("Enter the next tip." +
                           " End the list of tips with -1: ");
4.3.      tip = int.Parse(Console.ReadLine());
        } // while
5.      Console.WriteLine("You have earned {0} shekels", sum);
    } // Main
} // SumOfTips

```

## מעקב

נעקוב אחר מהלך ביצוע התוכנית SumOfTips עבור הקלט 1- 3 6 12:

בטבלת מעקב הכוללת משפט `while` או כוללים עמודה עבור התנאי הבוליאני שבכותרת המשפט.

	המשפט לביצוע	tip	sum	tip!= -1	פלט
1	<code>sum = 0;</code>	?	0		
2	<code>Console.Write("Enter your first tip...");</code>	?	0		Enter your first tip ...
3	<code>tip = int.Parse(Console.ReadLine());</code>	12	0		
4	<code>while (tip!= -1)</code>	12	0	<b>true</b>	
4.1	<code>sum = sum + tip;</code>	12	12		
4.2	<code>Console.Write("Enter the next tip...");</code>	12	12		Enter the next tip. ...
4.3	<code>tip = int.Parse(Console.ReadLine());</code>	6	12		
4	<code>while (tip != -1)</code>	6	12	<b>true</b>	
4.1	<code>sum = sum + tip;</code>	6	18		
4.2	<code>Console.Write("Enter the next tip...");</code>	6	18		Enter the next tip. ...
4.3	<code>tip = int.Parse(Console.ReadLine());</code>	3	18		
4	<code>while (tip!= -1)</code>	3	18	<b>true</b>	
4.1	<code>sum = sum + tip;</code>	3	21		
4.2	<code>Console.Write("Enter the next tip...");</code>	3	21		Enter the next tip. ...
4.3	<code>tip = int.Parse(Console.ReadLine());</code>	-1	21		
4	<code>while (tip!= -1)</code>	-1	21	<b>false</b>	
5	<code>Console.WriteLine("You have earned {0} shekels", sum);</code>	-1	21		You have earned 21 Shekels

### סוף פתרון קציה 7

נציג את המבנה החדש שהכרנו בפתרון לבעיה 7:

**הוראה לביצוע-חוזר-בתנאי** מבטאת ביצוע-חוזר של תת-משימה התלוי בקיום תנאי. הוראה כזאת נכתבת במבנה **כזו ... כזו ...**

גם הוראה כזאת נקראת **לולאה**, והתנאי להמשך הביצוע-החוזר נקרא **תנאי הכניסה ללולאה**.

הוראה לביצוע-חוזר-בתנאי מיושמת ב-`C#` במשפט `while`.

**המבנה הכללי של משפט while הוא:**

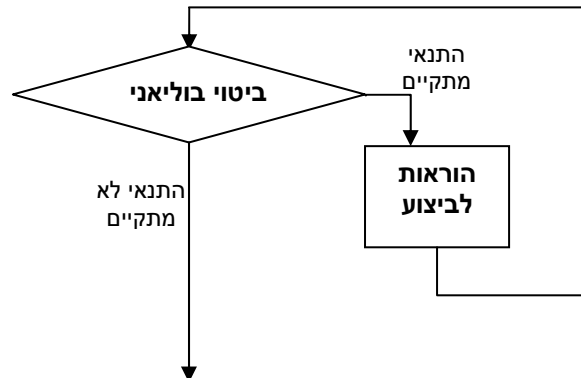
```
while (ביטוי בוליאני)
{
    גוף הלולאה: משפט או משפטים לביצוע
}
```

**ביצוע משפט while** מתחיל בחישוב ערכו של הביטוי הבוליאני. אם ערכו **true** מתבצע גוף הלולאה. בתום ביצוע גוף הלולאה מחושב הביטוי הבוליאני שוב. אם ערכו **true** מתבצע גוף הלולאה שוב. תהליך זה נמשך כל עוד ערכו של הביטוי הבוליאני הוא **true**. כאשר ערכו הוא **false** מסתיים ביצוע משפט ה-`while`.

**גוף הלולאה** תחום בסוגריים מסולסלים. במקרה שגוף הלולאה מכיל משפט בודד אפשר להשמיט את הסוגריים.

**בטבלת מעקב** אחר מהלך ביצוע תוכנית הכוללת משפט `while` אנו כוללים עמודה עבור תנאי הכניסה ללולאה.

ניתן לתאר משפט `while` באמצעות תרשים הזרימה הבא:



כאמור השוני בין בעיה 7 לבעיות הקודמות לה בפרק הוא בהגדרת הקלט. בבעיות הקודמות ניתן מראש אורך רשימת איברי הקלט, ואילו בבעיה 7 סיום רשימת הקלט מסומן בסימן מיוחד (1-). סימן כזה נקרא **זקיף**.

**זקיף** הוא נתון קלט חריג שתפקידו לסמן את סוף סדרת ערכי הקלט. הזקיף אינו חלק מסדרת הנתונים, ואין לעבד אותו כמו שאר איברי הסדרה.

למשל בפתרון בעיה 7 לא הוספנו את הערך 1- לסכום איברי הסדרה כיוון שהזקיף אינו נחשב כחלק מהסדרה!

השוני בהגדרת הקלט משפיע גם על מבנה הלולאה. הבדל ברור אחד הוא שבפתרון בעיה 7 השתמשנו בהוראה לביצוע-חוזר-בתנאי שיושמה במשפט `while`, בעוד שבפתרון הבעיות הקודמות השתמשנו בהוראות לביצוע-חוזר מספר פעמים ידוע מראש שיושמו במשפטי `for`. אך יש הבדל נוסף – האופן שמשולבות הוראות הקלט בתוך הלולאה:

בלולאות שבהן השתמשנו בפתרון בעיות קודמות, גוף הלולאה כלל הוראת קלט ומיד לאחריה הוראה לעיבוד הקלט. מבנה זה של גוף הלולאה התאים לבעיות שהיה צריך לעבד בהן כל אחד ואחד מאיברי הקלט באופן אחיד.

בבעיות דוגמת בעיה 7 עיבוד איברי הקלט אינו אחיד: העיבוד של הזקיף שונה מהעיבוד של איברים אחרים בקלט. דרוש מבנה המאפשר לקרוא איבר קלט ולעבד אותו רק אחרי שבוצעה בדיקה כי הוא אינו הזקיף. לכן בפתרון בעיות שהגדרת הקלט בהן כוללת זקיף, מתבצעת הוראת קלט ראשונה לפני הלולאה. גוף הלולאה כולל קודם כל הוראה לעיבוד הקלט, ורק אחר כך הוראת קלט.

**המבנה הכללי של הוראה לביצוע-חוזר לעיבוד רשימת קלט המסתיימת בזקיף:**

קלט *נכון*  
כל *ז* *הנכון* *הקלט* *אינו* *הזקיף* *כז*:  
*ז* *א* *הנכון* *הקלט*  
קלט *נכון*

באופן זה, הנתון החדש ייבדק תמיד **מיד** לאחר קליטתו.

### שאלה 7.27

נתון קטע התוכנית הבא. הקלט הוא רשימת תווים המהווים מילה באנגלית ובסופה הזקוף '\*'.

```
int i = 0;
char letter;
Console.WriteLine("Enter the first letter of the word: ");
letter = char.Parse(Console.ReadLine());
while (letter != '*')
{
    i = i + 1;
    Console.WriteLine("Enter the next letter of the word: ");
    letter = char.Parse(Console.ReadLine());
}
Console.WriteLine(i);
```

מהי מטרת קטע התוכנית?

### שאלה 7.28

פתחו אלגוריתם אשר הקלט שלו הוא רשימת ציונים (נתונים כמספרים שלמים) בין 0 ל-100 אשר בסופה הזקוף 101, והפלט שלו הוא מספר הציונים ברשימה הגדולים או שווים ל-60. ישמו את האלגוריתם בשפת C#.

### שאלה 7.29

אלון ובני מתחרים על תפקיד יושב ראש ועדת קישוט של הכיתה, מועמד זוכה אם **יותר** מחצי מן הבוחרים הצביעו עבורו. פתחו וישמו אלגוריתם אשר הקלט שלו הוא סדרה של התווים A ו-B (A עבור אלון, B עבור בני), המבטאת את קולות הבוחרים, ומסתיימת בזקוף '#'. הפלט שלו הוא הודעה אם אלון זכה או לא זכה ברוב קולות (כלומר ביותר מחצי מקולות הבוחרים). למשל, עבור הקלט ABBAABBAA# הפלט המתאים הוא: Alon wins, ועבור הקלט ABBABBBAA# הפלט המתאים הוא: Alon didn't win.

להעמקה בתבנית **איסוף בקיזוז** פנו לסעיף התבניות המופיע בסוף הפרק.

### שאלה 7.30

בבחירות לוועד חיות היער מועמדות שלוש חיות: העכבר שמספרו 1, האריה שמספרו 2, והנמלה שמספרה 3. פתחו אלגוריתם אשר הקלט שלו הוא רשימת קולות הבוחרים (כל קול הוא אחד מן המספרים 1, 2 או 3) ובסופה הזקוף 0. הפלט הוא הודעה המתארת עד כמה צריכות חיות היער להיזהר: אם האריה קיבל פחות מ-30% מהקולות עליהן להיזהר מאוד, אם האריה קיבל בין 31% ל-70% מהקולות עליהן להיזהר קצת, ואם האריה קיבל יותר מ-71% מהקולות הן יכולות להיות רגועות ואינן צריכות להיזהר כלל. ישמו את האלגוריתם בשפת C#.

### שאלה 7.31

פתחו אלגוריתם אשר הקלט שלו הוא סדרת תווים המהווה מילה באנגלית, ומסתיימת בזקוף '\*'. הפלט שלו הוא מילת הקלט והיא מוצפנת באופן הבא: כל אות במילה מוחלפת באות העוקבת לה "בצורה מעגלית" ב-א"ב האנגלי (כלומר, כל אות מלבד האות Z מוחלפת באות העוקבת לה, והאות Z מוחלפת באות A). למשל עבור הקלט ZEBRA\* הפלט הוא AFCSB. ישמו את האלגוריתם בשפת C#.

### שאלה 7.32

טורניר השש-בש מתחיל, אך איבדתם את הקוביות! פתחו אלגוריתם אשר מדמה הטלת שתי קוביות ומציג את תוצאות ההטלה. האלגוריתם מדמה את ההטלות עד שתוצאת ההטלה היא "שש-בש", כלומר צמד המספרים 5 ו-6 או 6 ו-5. לסיום האלגוריתם מציג כפלט את מספר ההטלות שהתבצעו עד שהתקבלה התוצאה שש-בש. ישמו את האלגוריתם בשפת C#.

## ביצוע-חוזר עם תנאי כניסה כלשהו

### חזרה 8

**מטרת הבעיה ופתרונה:** הצגת שימוש בהוראה לביצוע-חוזר-בתנאי כניסה כלשהו שאינו תלוי בזקף.

בתוכנית "הנוסע המתמיד" של חברת התעופה "שחקים" ניתן לצבור מרחקי טיסות. נוסע אשר צובר למעלה מ-3000 קילומטרים זוכה בכרטיס טיסה חינם להונולולו. פתחו וישמו אלגוריתם אשר הקלט שלו הוא רשימת מרחקי הטיסות של נוסע אשר זכה בכרטיס חינם (כלומר ידוע שכבר צבר יותר מ-3000 ק"מ). הפלט שלו הוא: מספר המרחקים המופיעים ברצף מתחילת הרשימה אשר סכומם המצטבר עולה על 3000, ומספר הקילומטרים שנשארו לנוסע מעבר ל-3000 ק"מ לצבירות הבאות. למשל עבור הקלט: 500 150 700 1000 800 הפלט המתאים הוא 5, משום שסכומם של כל חמשת המרחק ים עולה על 3000, ואחרי שמופחת מסכום המרחקים הערך 3000 נשארת יתרה של 150.

### ניתוח הבעיה בעזרת דוגמאות

### שאלה 7.33

מהו הפלט המתאים עבור כל אחד מן הקלטים הבאים:

א. 200 1000 800 600 600.

ב. 1000 1200 800 900.

**?** ברור שבפתרון הבעיה יש לצבור את המרחקים הנתונים. כלומר יש לבצע לולאה שבה ייקלט מרחק נתון ויתווסף לצובר. אך כמה פעמים יש לבצע זאת?

בניגוד לבעיות בסעיפים הקודמים שאורך רשימת איברי הקלט בהם היה נתון, או שניתן לנו סימן מיוחד לסיום הקלט (זקף), בבעיה זו יש לקלוט מרחקים ולהוסיפם לצובר כל עוד ערכו של הצובר איננו גדול מ-3000. ברגע שערכו של הצובר גדול מ-3000 אין צורך להמשיך בפעולות הקליטה והצבירה. מכאן נובע שיש לבצע את פעולות הקליטה והצבירה רק כל עוד מתקיים התנאי הבא:

המרחק הנוסף קטן או שווה ל-3000

### פירוק הבעיה לתת-משימות

כל עוד המרחק הנצבר קטן או שווה ל-3000 יש לחזור על ביצוע התת-משימות הבאות:

קליטת מרחק נתון

הוספת המרחק הנתון לצובר

הגדלה ב-1 של "מונה המרחקים"

המונה המוזכר בתת-משימה האחרונה ימנה את מספר המרחקים שנצברו בצובר. כאשר יסתיים ביצוע הלולאה, ישמור מונה זה את מספר המרחקים שנצברו עד שהסכום המצטבר עלה על 3000.

### בחירת משתנים

**dist** – מטיפוס שלם, ישמור מרחק תורן בקלט  
**sum** – מטיפוס שלם, צובר שישמור את סכום המרחקים המצטבר  
**counter** – מטיפוס שלם, מונה שישמור את מספר המרחקים שנצברו

### האלגוריתם

1. **אגף** **sum** **0-2**
2. **אגף** **counter** **0-2**
3. **כאשר** **sum** **קטן** **או** **שווה ל-3000** **בצע**:
  - 3.1. **קוט** **אגף** **המרחק** **dist-2**
  - 3.2. **הוסף** **אגף** **sum** **ל-dist**
  - 3.3. **הגדל** **1-2** **אגף** **counter**
  4. **הגדל** **אגף** **סריכו** **ל** **counter**
  5. **הגדל** **אגף** **ההפרש** **sum-3000**

### שאלה 7.34

א. ישמו את האלגוריתם בשפת C#.  
ב. בנו טבלת מעקב אחר ביצוע התוכנית (לפתרון בעיה 8) עבור הקלט:  
500 1200 800 300 300  
כמה פעמים יתבצע גוף הלולאה?

**שימו** ♥: בכל ביצוע-חוזר של הלולאה נקלט מרחק טיסה אחד, ולכן הערך השמור במשתנה counter הוא בעצם מספר הפעמים שמתבצעת הלולאה.

**סוף פתרון בעיה 8**

### שאלה 7.35

נסחו תנאי כניסה בוליאני מתאים עבור כל אחד מן התיאורים הבאים של ביצוע-חוזר:  
א. סכימת משקלי מכוניות (לצורך מעבר במעבורת) **כל עוד** הסכום אינו עולה על 100 טון.  
ב. סכימת המספרים החיוביים השלמים המתחילים ב-1 **עד אשר** הסכום גדול מהערך הנתון כקלט.  
ג. קריאת אותיות **עד אשר** נקלטות 10 אותיות A.  
**שימו** ♥: בסעיף א מתואר הביצוע-החוזר במתכונת של **כל עוד** ובסעיפים ב ו-ג מתואר הביצוע-החוזר במתכונת של **עד אשר**.

### שאלה 7.36

בכל אחד מן הסעיפים הבאים מופיע קטע תוכנית הכולל משפט **while** ובו תנאי הכניסה חסר. השלימו את תנאי הכניסה לפי מטרת קטע התוכנית, ובנו טבלת מעקב אחר ביצוע הקטע השלם.  
א. מטרת הקטע: הצגה של המספר הזוגי הקטן ביותר אשר גדול מנתון הקלט בהינתן שהקלט הוא מספר חיובי.

```
Console.WriteLine("Enter a number: ");  
num = int.Parse(Console.ReadLine());  
i = 0;
```

```
while (_____)
    i = i + 2;
Console.WriteLine(i);
```

בנו טבלת מעקב אחר ביצוע קטע התוכנית השלם עבור הקלט 9.

ב. מטרת הקטע: הצגה של מכפלת שני נתוני הקלט, בהינתן שהם מספרים שלמים חיוביים.

```
Console.Write("Enter a number: ");
x = int.Parse(Console.ReadLine());
Console.Write("Enter a number: ");
y = int.Parse(Console.ReadLine());
counter = 0;
sum = 0;
while (_____)
{
    sum = sum + x;
    counter = counter + 1;
}
Console.WriteLine(sum);
```

בנו טבלת מעקב אחר ביצוע קטע התוכנית השלם עבור הקלט 3 4.

### שאלה 7.37

נתון קטע התוכנית הבא אשר הקלט שלו הוא מספר חיובי שלם:

```
Console.Write("Enter a number: ");
limit = int.Parse(Console.ReadLine());
s = 0;
c = 0;
while (s < limit)
{
    c = c + 1;
    s = s + c;
}
Console.WriteLine(c);
```

קטע התוכנית כולל מונה  $c$  וצובר  $s$ . הצובר צובר את ערכיו של המונה במהלך הביצוע-החוזר.

א. מהו הפלט עבור הקלט 1? מהו הפלט עבור הקלט 11?

ב. ציינו שני קלטים שונים שעבורם יהיה הפלט 5. מהו מספר הפעמים לביצוע-החוזר עבור קלטים אלה?

ג. מהי מטרת קטע התוכנית?

היעזרו בטבלת מעקב כדי לענות על סעיפים א ו-ב.

### שאלה 7.38

נתון קטע התוכנית הבא אשר הקלט שלו הוא מספר חיובי שלם, ו- $TOP\_LIMIT$  הוא קבוע שערכו 100:

```
Console.Write("Enter a number: ");
num = int.Parse(Console.ReadLine());
mult = 1;
i = 0;
while ((i < num) && (mult < TOP_LIMIT))
{
    i = i + 1;
    mult = mult * i;
```

```
}  
Console.WriteLine(mult);
```

- קטע התוכנית הנתון כולל צובר `mult` אשר צובר ערך של מכפלה, והוא מאותחל בערך 1.
- א. מהו הפלט עבור הקלט 2? ומהו הפלט עבור הקלט 5?
- ב. מהו הקלט שעבורו יהיה הפלט 24? ומהו מספר הפעמים של הביצוע-החוזר עבור קלט זה?
- ג. מהי מטרת קטע התוכנית?
- היעזרו בטבלת מעקב כדי לענות על סעיפים א ו-ב.

### שאלה 7.39

פתחו אלגוריתם אשר הקלט שלו הוא מספר חיובי שלם, והפלט שלו הוא החזקה הקטנה ביותר של 2 אשר גדולה מנתון הקלט. למשל: עבור הקלט 7 הפלט הדרוש הוא 8 (כי  $2^3=8$ ), ועבור הקלט 8 הפלט הדרוש הוא 16 (כי  $2^4=16$ ). ישמו את האלגוריתם בשפת `C#`. במהלך הפיתוח הקפידו על ניסוח תת-משימות לביצוע-חוזר ועל ניסוח תנאי לביצוע-חוזר.

**שימו ♥:** באלגוריתם זה יש להשתמש בצובר של מכפלה, ולא בפעולת החזקה `Pow` המוגדרת במחלקה `Math`.

בלולאות `while` שראינו עד כה תנאי הכניסה היו דומים למדי זה לזה. הם כללו תמיד השוואה של משתנה, אשר ערכו גדל במהלך הביצוע-חוזר של הלולאה, לחסם אשר נשמר במשתנה או נקבע מפורשות. הביצוע-החוזר הסתיים כאשר ערכו של המשתנה גדל ועבר את החסם (או השתווה לו). אבל תנאי כניסה כאלה הם רק סוג אחד של תנאי כניסה ללולאה. בפתרון הבעיה הבאה נראה דוגמה לתנאי כניסה מסוג אחר.

## משימה 9

**מטרת הבעיה ופתרונה:** הצגת תבנית פירוק מספר שלם חיובי כלשהו לספרותיו.

פתחו וישמו אלגוריתם אשר הקלט שלו הוא מספר שלם חיובי והפלט שלו הוא מספר ספרות המספר. למשל: עבור הקלט 31 הפלט הדרוש הוא 2 ועבור הקלט 15568 הפלט הדרוש הוא 5.

בפרקים קודמים כבר עסקנו בפירוק של מספר שלם לספרותיו, אך שם מספר ספרותיו היה קטן וידוע. בבעיה הנתונה מספר הספרות איננו ידוע ויכול להיות גדול מאוד.

### פירוק הבעיה לתת-משימות

מה יהיו התת-משימות לביצוע-חוזר עבור מניית ספרות המספר? ומה יהיה התנאי לחזרה על ביצוע תת-משימות אלה?

נוכל "לקצץ" את ספרות המספר אחת אחת ולמנות את מספר הספרות הנקצצות. נעשה זאת באמצעות תת-משימות לביצוע-חוזר הכוללות קיצוץ ספרה מן המספר והגדלת מונה ב-1. קיצוץ ספרה יתבצע באמצעות חלוקה בשלמים של המספר ב-10. חלוקה זו תביא לקיצוץ ספרת האחדות (הספרה הימנית ביותר). למשל במקום המספר 534 יתקבל המספר 53.

אם כך, ננסח את התת-משימות הבאות לביצוע-חוזר:

1. קיצוץ ספרת האחדות
2. הגדלה ב-1 של מונה הספרות



יש לבצע תת-משימות אלו כל עוד "יש מה לקצץ", כלומר כל עוד המספר הנותר במהלך הביצוע-החוזר גדול מ-0. לכן התנאי לביצוע-חוזר יהיה: **המספר הנותר גדול מ-0**.

### בחירת משתנים

נבחר שני משתנים מטיפוס שלם:

**num** – ישמור את המספר הניתן כקלט, וספרותיו נקצצות

**digits** – מונה שישמור את מספר הספרות שנקצצו

### יישום האלגוריתם

```
/*
קלט: מספר חיובי שלם
פלט: מספר הספרות במספר הנתון
*/
using System;
public class DigitCount
{
    public static void Main ()
    {
        int digits = 0; // מספר הספרות
        int num; // המספר המעובד
        Console.Write("Enter a number: ");
        num = int.Parse(Console.ReadLine());
        while (num > 0)
        {
            num = num / 10;
            digits++;
        }
        Console.WriteLine("The number of digits is {0}", digits);
    } // Main
} // DigitCount
```

**סוף פתרון בעיה 9**

להעמקה בתבניות פירוק מספר לספרותיו ובניית מספר פנו לסעיף התבניות המופיע בסוף הפרק.

### שאלה 7.40

- א. שנו את התוכנית DigitCount לפתרון בעיה 9, כך שהפלט יכלול את **סכום** ספרות המספר, למשל עבור הקלט 153 הפלט המתאים הוא 9.
- ב. שנו את התוכנית DigitCount לפתרון בעיה 9, כך שהפלט יכלול רק את **מספר** הספרות **האי-זוגיות** במספר הנתון. למשל עבור הקלט 150 הפלט המתאים הוא 2.
- ג. שנו את התוכנית DigitCount לפתרון בעיה 9, כך שהפלט יכלול רק את **מכפלת** ספרות המספר. שימו לב לאתחול נכון של המשתנה השומר את המכפלה.

### שאלה 7.41

נתון קטע התוכנית החלקי הבא, אשר הקלט שלו הוא שני מספרים שלמים חיוביים  $x$  ו- $y$  ומטרתו היא הצגת שארית החלוקה בשלמים של  $x$  ב- $y$  (כלומר תוצאת החישוב  $x \% y$ ). בקטע התוכנית מתבצע החישוב הדרוש באמצעות פעולת **חיסור**.

השלימו את קטע התוכנית.

```
Console.Write("Enter a number: ");
x = int.Parse(Console.ReadLine());
Console.Write("Enter a number: ");
y = int.Parse(Console.ReadLine());
while (_____)
    x = x - y;
Console.WriteLine(_____);
```

#### שאלה 7.42

שנו את הקטע הנתון בשאלה הקודמת כך שיציג גם את מנת החלוקה של  $x$  ב- $y$ , (כלומר את תוצאת החישוב  $x/y$ ). יש לבצע את החישובים הדרושים באמצעות פעולות חיבור וחסר בלבד!

#### שאלה 7.43

שני תלמידים המשחקים זה נגד זה מותחים בתחילת המשחק קו באורך  $N$  סנטימטרים ( $N > 1$ ). השחקנים מחליפים תורות לסירוגין. כל שחקן מקצר בתורו את הקו לחצי מאורכו. השחקן אשר מקצר בתורו את הקו לאורך של פחות מסנטימטר אחד מנצח במשחק. למשל אם אורכו של הקו הוא 8 ס"מ, השחקן הראשון יקצר את הקו ל-4 ס"מ והשחקן השני יקצר את הקו ל-2 ס"מ, השחקן הראשון יקצר את הקו לס"מ אחד והשחקן השני יקצר את הקו ל-0.5 ס"מ וינצח במשחק. פתחו וישמו אלגוריתם אשר הקלט שלו הוא אורכו ההתחלתי של הקו, והפלט שלו הוא הודעה מיהו השחקן המנצח (הראשון או השני).

#### שאלה 7.44

במשחק אסימונים שחקן מניח 2 אסימונים בתור הראשון, 4 אסימונים בתור השני, 8 אסימונים בתור השלישי, וכך הלאה – בכל תור מוכפל מספר האסימונים. נתון קטע התוכנית הבא אשר הקלט שלו הוא מספר האסימונים ההתחלתי של השחקן, והפלט שלו צריך להיות המספר הסידורי של התור אשר בו לא ניתן להמשיך לשחק לפי השיטה המתוארת. למשל, עבור הקלט המייצג מספר אסימונים התחלתי 9, הפלט הדרוש הוא 3, כיוון שאחרי שהניח 2 אסימונים בתור הראשון ו-4 אסימונים נוספים בתור השני יוותרו לשחקן רק 3 אסימונים (ולא 8 אסימונים, כפי שנדרש לתור השלישי). קטע התוכנית שגוי.

```
int turn = 0; // המספר הסידורי של התור הנוכחי
int tokensInCurrentTurn = 0; // מספר אסימונים למשחק בתור הנוכחי
int totalPlayedTokens = 0; // סכום אסימונים כולל בתורות שהתבצעו עד כה
int startTokens; // מספר אסימונים התחלתי
Console.Write("Enter number of tokens to start with: ");
startTokens = int.Parse(Console.ReadLine());
while (totalPlayedTokens <= startTokens)
{
    turn = turn + 1;
    tokensInCurrentTurn = tokensInCurrentTurn * 2;
    totalPlayedTokens = totalPlayedTokens + tokensInCurrentTurn;
}
Console.WriteLine("The game cannot go on after {0} turns", turn);
```

ציינו מהי השגיאה ותקנו אותה.

#### שאלה 7.45

בצלחת פֶּטְרִי החיידקים מכפילים את עצמם פי 5 בכל דקה עד אשר מספרם עובר סף מסוים הנקרא "סף ההכפלה".

פתחו אלגוריתם אשר הקלט שלו הוא שני נתונים: מספר חיידקים התחלתי בצלחת וסף ההכפלה. הפלט שלו הוא מספר החיידקים שיהיו בצלחת בדקה שבה יעבור מספרם את סף ההכפלה. ישמו את האלגוריתם בשפת C#.

ראינו עד כה את המבנים `for` (או `while`) ו-`do-while`... לביצוע-חוזר של תת-משימה. לעתים נוהגים לכנות את המבנה הראשון בשם "לולאת `for`" ואת המבנה השני בשם "לולאת `while`". מתי נבחר להשתמש במבנה הראשון ומתי נעדיף את המבנה השני?

♦ כאשר אפשר לחשב לפני ביצוע לולאה את מספר הפעמים שתתבצע, נעדיף לשם הנוחות ולשם הבהירות להשתמש בלולאת `for`.

♦ כאשר אי אפשר לחשב לפני ביצוע לולאה את מספר הפעמים שתתבצע, בין שהסיום נשלט בידי זקיף ובין שנשלט בידי תנאי אחר, נשתמש בלולאת `while`. המבנה של לולאת `while` מבהיר כי ביצועה תלוי בתנאי, והתנאי עצמו קל לזיהוי.

בחירה נכונה בהוראה לביצוע-חוזר מבהירה לקורא התוכנית אם מספר הפעמים לביצוע ידוע מראש, ואם לא, על פי מה הוא נקבע. לכן בחירה נכונה של הוראה לביצוע-חוזר מסייעת לקריאות ולבהירות התוכנית.

#### שאלה 7.46

כתבו לולאה להצגת פלט של 50 כוכביות.

א. כתבו לולאת `for` להצגת הפלט הדרוש.

ב. כתבו לולאת `while` להצגת הפלט הדרוש.

ג. איזה מן הפתרונות פשוט יותר?

#### שאלה 7.47

א. כתבו לולאת `for` המציגה כפלט את 20 הכפולות החיוביות הראשונות של 5 (כלומר הפלט הוא: 5 10 15 ... 100).

ב. כתבו לולאת `for` נוספת המבצעת אותו דבר אבל בעלת כותרת שונה וגוף לולאה שונה.

ג. כתבו לולאת `while` המבצעת אותו דבר.

#### שאלה 7.48

ציינו עבור כל אחת מן הבעיות הבאות אם לפתרונה מתאים יותר להשתמש בלולאת `for` או בלולאת `while`. נמקו את תשובותיכם.

א. הקלט: מספר שלם חיובי `num`. הפלט: רשימת המספרים השלמים החיוביים מ-1 עד `num`.

ב. הקלט: מספר שלם חיובי `num`. הפלט: רשימת המספרים השלמים השליליים מ-1 עד `-num`.

ג. הקלט: מספר שלם חיובי `num`. הפלט: רשימת המספרים השלמים החיוביים מ-1 עד המספר הקטן ביותר `k` אשר עבורו מתקיים  $1+2+3+\dots+k > num$ .

שתי השאלות הבאות מתייחסות להשוואת אותיות באלף-בית האנגלי. נגדיר סדר מילוני של אותיות באופן הבא: אות אחת קטנה מאות אחרת אם האחת מופיעה לפני האחרת בסדר הא"ב. אם האות מופיעה אחרי האחרת בסדר הא"ב אז היא גדולה ממנה. (למשל B קטנה מ-E, ו-F גדולה מ-C).

#### שאלה 7.49

עבור כל אחת מן הבעיות הבאות ציינו אם לפתרונה מתאים יותר להשתמש בלולאת `for` או בלולאת `while`. נמקו בקצרה את תשובותיכם.

**שימו ⚡:** בעיות אלו מתאימות לתבניות מקסימום, מינימום, מקום המקסימום ומקום המינימום, אך לא תמיד גודל התחום שמופעלת בו התבנית ידוע מראש.

א. הקלט הוא סדרת אותיות מהאלף-בית ובסופה '\*', והפלט הוא האות הגדולה ביותר שמופיעה בקלט.

ב. הקלט הוא מספר שלם חיובי המציין אורך סדרה ואחריו סדרת אותיות מהאלף-בית באורך המצוין, והפלט הוא האות הקטנה ביותר שמופיעה בקלט.

ג. הקלט הוא כמו הקלט לסעיף ב, והפלט הוא המקום הסידורי של ההופעה הראשונה (אולי יש יותר מאחת) של האות הגדולה ביותר שמופיעה בקלט.

ד. הקלט הוא כמו הקלט לסעיף ב, והפלט הוא המקום הסידורי של ההופעה האחרונה (אולי יש יותר מאחת) של האות הקטנה ביותר שמופיעה בקלט.

ה. מהם ההבדלים ביישום האלגוריתם בין סעיף ג ל-ד?

#### שאלה 7.50

פתחו אלגוריתם אשר הקלט שלו הוא סדרת אותיות מן האלף-בית האנגלי שמסתיימת ב-\*, והפלט שלו הוא האות הגדולה ביותר מבין האותיות B עד I המופיעות בקלט.

למשל עבור הקלט \*SCHOOL הפלט המתאים הוא H. אמנם יש בקלט אותיות גדולות מ-H, למשל S, אך אותיות אלו לא כלולות בסדרת האותיות B עד I.

הניחו שבקלט מופיעה לפחות אות אחת בתחום B עד I.

ישמו את האלגוריתם בשפת C#.

### ביצוע-חוזר אינסופי

כאמור, ההוראות לביצוע-חוזר שהוצגו בשלושת הסעיפים הראשונים היו תמיד באורך ידוע מראש. כמובן עבור לולאות כאלו התשובה לשאלה "כמה פעמים יתבצע גוף הלולאה" היא פשוטה מאוד. אבל כאשר מדובר בהוראה לביצוע-חוזר-בתנאי, התשובה לשאלה זו אינה תמיד פשוטה, כפי שמדגימה הבעיה הבאה.

### קצ'ה 10

**מטרת הבעיה ופתרונה:** דיון בחישוב מספר הפעמים שמתבצעת לולאה, והצגת לולאה אינסופית.

נתונה התוכנית הבאה (מטרתה מתוארת בהערה בתחילת התוכנית):

```
/*
קלט: מספר חיובי שלם
פלט: כל המספרים האי-זוגיים החיוביים הקטנים מן המספר הנתון
*/
using System;
public class OddNumbers
{
    public static void Main ()
    {
        int limit; // המספר הנתון
        int oddN = 1; // המספר האי-זוגי הראשון
    }
}
```

```

1. Console.WriteLine("Enter a number: ");
2. limit = int.Parse(Console.ReadLine());
3. Console.WriteLine("The odd numbers that are smaller " +
                    "than the given number are: ");
4. while (oddN != limit)
    {
4.1. Console.WriteLine("{0} ", oddN);
4.2. oddN = oddN + 2;
    }
    } //Main
} // OddNumbers

```

חשבו את מספר הפעמים שתתבצע הלולאה שבתוכנית עבור קלט נתון L.

### ניתוח הבעיה בעזרת דוגמאות

כדי לבטא בצורה כללית (עבור קלט L) את מספר הפעמים שתתבצע הלולאה התוכנית נחשב את מספר הפעמים לביצוע הלולאה עבור דוגמאות קלט שונות:

עבור הקלט 1 לא יתבצע גוף הלולאה אפילו פעם אחת, כיוון שערכי limit ו-oddN שווים כבר בפעם הראשונה שמחושב תנאי הכניסה ללולאה.

עבור הקלט 5, נבחן את מספר הפעמים של ביצוע הלולאה באמצעות טבלת מעקב:

	המשפט הבא לביצוע	oddN	limit	oddN!=limit	פלט
3	Console.WriteLine("The odd Numbers ...");	1	5		The odd Numbers...
4	while (oddN != limit)	1	5	true	
4.1	Console.WriteLine("{0} ", oddN);	1	5		1
4.2	oddN = oddN + 2;	3	5		
4	while (oddN != limit)	3	5	true	
4.1	Console.WriteLine("{0} ", oddN);	3	5		3
4.2	oddN = oddN + 2;	5	5		
4	while (oddN != limit)	5	5	false	

עבור הקלט 5 יתבצע גוף הלולאה פעמיים ויוצג הפלט: 1 3

באופן דומה נוכל לראות שעבור הקלט 15 יתבצע גוף הלולאה 7 פעמים, ויוצג הפלט:

1 3 5 7 9 11 13

? ננסה להכליל על פי הדוגמאות שבחנו את מספר הפעמים שהלולאה תתבצע עבור קלט אי-זוגי שערכו L. מהו הביטוי הכללי?

עבור קלט אי-זוגי שערכו L הלולאה תתבצע  $(L-1)/2$  פעמים.

? ניסחנו ביטוי כללי של מספר הפעמים לביצוע הלולאה עבור קלטים אי-זוגיים. האם קיים ביטוי כללי דומה עבור קלטים זוגיים?

נבחן את מספר הפעמים לביצוע הלולאה עבור הקלט 2 באמצעות טבלת מעקב:

	המשפט הבא לביצוע	oddN	limit	oddN!=limit	פלט
3	Console.WriteLine("The odd Numbers ...");	1	2		The odd Numbers...
4	while (oddN != limit)	1	2	true	

4.1	Console.Write("{0} ", oddN);	1	2		1
4.2	oddN = oddN + 2;	3	2		
4	while (oddN != limit)	3	2	true	
4.1	Console.Write("{0} ", oddN);	3	2		3
4.2	oddN = oddN + 2;	5	2		
4	while (oddN != limit)	5	2	true	

קטענו את מילוי הטבלה לפני סיום ביצוע המעקב המלא. ניתן לראות שעבור הקלט 2 מוצג הפלט 3 אשר אין להציגו, כיוון שהוא מספר אי-זוגי שאיננו קטן מן הקלט!

כאשר מחושב הביטוי הבוליאני `oddN != limit` (תנאי הכניסה ללולאה) בפעם השנייה, ערכו של `oddN` הוא 3, וערכו של `limit` הוא 2, ולכן ערכו של הביטוי הבוליאני הוא `true` ומוצג הערך 3 כפלט. ערכו של `oddN` גדל ב-2 בכל סיבוב בלולאה, ולכן הלולאה תתבצע גם פעם שלישית ויוצג הפלט 5. בעצם, הלולאה תתבצע שוב ושוב וערכו של הביטוי הבוליאני יהיה `true` תמיד, כיוון שערכו של `oddN` רק ילך ויגדל. מכאן נובע שהלולאה תתבצע אינסוף פעמים.

הלולאה תתבצע אינסוף פעמים גם עבור הקלט 4 וגם עבור הקלט 6, ובעצם עבור כל קלט זוגי. זאת משום שעבור קלט זוגי יתקיים תנאי הכניסה ללולאה שוב ושוב ואף פעם לא יהיה מצב של שוויון של ערכו של `limit` (שהוא מספר זוגי) וערכו של `oddN` (שיהיה תמיד מספר אי-זוגי).

כיוון שעבור קלט זוגי הלולאה תתבצע אינסוף פעמים יוצגו כפלט מספרים אי-זוגיים שאין להציגם, ולכן לא רק שהתוכנית אינה מסתיימת, היא גם מציגה פלט שגוי!

❓ כיצד ניתן לתקן את התוכנית ולטפל בכך שהלולאה תתבצע מספר מתאים של פעמים גם עבור קלט זוגי?

אפשר לשנות את תנאי הכניסה ללולאה ל-`oddN < limit`, וכך ביצוע הלולאה יסתיים לאחר הצגת המספר האי-זוגי הגדול ביותר שעדיין קטן מהקלט.

## שאלה 7.51

תקנו את התוכנית `OddNumbers` כך שתסתיים לאחר הצגת כל המספרים האי-זוגיים החיוביים שקטנים מן המספר הנתון.

### סוף פתרון בעיה 10

התוכנית שהוצגה בבעיה 10 כללה לולאה אשר התבצעה כדרוש עבור כל קלט אי-זוגי, אך ביצועה נמשך אינסוף פעמים עבור כל קלט זוגי.

לולאה אשר מתבצעת אינסוף פעמים עבור קלט כלשהו נקראת **לולאה אינסופית**. בחומר הלימוד של "יסודות מדעי המחשב" תוכניות הכוללות לולאה אינסופית נחשבות כתוכניות שגויות.

בפיתוח אלגוריתם קורה לעתים שנכתבת לולאה אינסופית. לולאה כזו עלולה להתבצע אינסוף פעמים רק עבור חלק מן הקלטים. לכן חשוב להקפיד לבדוק לולאת אלגוריתם עבור בחירה ממצה של **דוגמאות קלט מייצגות**, כפי שעשינו בפתרון בעיה 10. בפתרון זה בדקנו תחילה את הלולאה עבור דוגמה של קלט אי-זוגי, וראינו שעבורה מושגת המטרה. אחר כך בדקנו עבור דוגמה של קלט זוגי ונוכחנו שהלולאה אינסופית.

חשבו: מה היה קורה אילו הקלדנו ערך שלילי כקלט לתוכנית?

### שאלה 7.52

ציינו עבור כל אחת מן הלולאות הבאות אם היא לולאה אינסופית.  
אם הלולאה סופית ציינו את מספר הפעמים שהיא תתבצע. היעזרו בטבלת מעקב לביצוע החישובים הדרושים.

א.	ב.
<pre>int i = 0; while (i &lt; 30)     i = i + 4;</pre>	<pre>int j = 0; while (j &lt; 50)     j = j - 10;</pre>
ג.	ד.
<pre>int j = 0; while (Math.Abs(j) &lt; 30)     j = j - 10;</pre>	<pre>int k = 0; for (int j = 1; j &lt; 10; j++)     k = k + 11;</pre>
ה. num הוא מספר שלם חיובי כלשהו	ו. num הוא מספר שלם חיובי כלשהו
<pre>while (num != 10) {     Console.WriteLine(num);     num = num + 1; }</pre>	<pre>while (num != 0) {     Console.WriteLine(num);     num = num - 1; }</pre>

### שאלה 7.53

נתון קטע התוכנית הבא אשר הקלט שלו הוא מספר שלם חיובי:

```
Console.Write("Enter a number: ");
num = int.Parse(Console.ReadLine());
while (num != 0)
{
    num = num % 3;
    Console.WriteLine(num);
}
```

הלולאה בקטע התוכנית תתבצע מספר סופי של פעמים עבור חלק מן הקלטים, ומספר אינסופי של פעמים עבור שאר הקלטים. לכן הלולאה שבקטע התוכנית היא לולאה אינסופית.  
א. תנו דוגמת קלט שעבורה תתבצע הלולאה מספר סופי של פעמים.  
מה מאפיין את הקלטים שעבורם תתבצע הלולאה מספר סופי של פעמים?  
כמה פעמים תתבצע הלולאה עבור קלטים אלה?  
ב. תנו דוגמת קלט שעבורה תתבצע הלולאה מספר אינסופי של פעמים.  
מה מאפיין את הקלטים שעבורם תתבצע הלולאה מספר אינסופי של פעמים?

### שאלה 7.54

נתון קטע התוכנית הבא אשר הקלט שלו הוא שני מספרים שלמים:

```
Console.Write("Enter a number: ");
x = int.Parse(Console.ReadLine());
Console.Write("Enter a number: ");
y = int.Parse(Console.ReadLine());
while (x != y)
{
    y = y - 1;
    x = x + 1;
}
Console.WriteLine(x);
```

א. תנו שתי דוגמאות קלט שונות שעבורן **לא** יתבצע גוף הלולאה.  
ב. תנו שתי דוגמאות קלט שונות שעבורן יתבצע גוף הלולאה **בדיוק פעם אחת**.

ג. תנו שתי דוגמאות קלט שונות שעבורן יתבצע גוף הלולאה בדיוק חמש פעמים.

ד. תנו שתי דוגמאות קלט שונות שעבורן יתבצע גוף הלולאה אינסוף פעמים.

ה. נניח שערכו של  $x$  הוא 0. עבור אילו ערכים של  $y$  יתבצע גוף הלולאה אינסוף פעמים?

## 7.5 משתנים מטיפוס בוליאני

בסעיף זה נכיר משתנים מטיפוס בוליאני, כלומר משתנים היכולים לשמור בתוכם אחד משני הערכים **true** או **false**. משתנים כאלה יכולים להשתלב בביטויים בוליאניים ולכן הם שימושיים מאוד בהוראות לביצוע-בתנאי ובהוראות לביצוע-חוזר-בתנאי, כפי שנדגים בהמשך הסעיף.

### קצ'ה 11

**מטרת הבעיה הבאה:** הצגת משתנה מטיפוס בוליאני

פתחו אלגוריתם שהקלט שלו הוא מספר שלם גדול מ-1 ולאחריו רשימה של מספרים שלמים נוספים הגדולים מ-1. סדרת המספרים מסתיימת במספר 0. אם המספר הראשון שנקרא הוא זוגי, אז הפלט הוא כל המספרים מרשימת המספרים כך שכל מספר מוצג פעמיים. אם המספר הראשון שנקרא הוא אי-זוגי אז הפלט הוא כל המספרים מרשימת המספרים וכל מספר מוצג פעם אחת בלבד.

למשל עבור הקלט: 0 7 4 2 8 הפלט המתאים הוא 2 2 4 4 7 7.

ועבור הקלט: 0 7 4 2 9 הפלט המתאים הוא 2 4 7.

### פירוק הבעיה לתת-משימות

ננסח רעיון לפירוק ראשוני לתת-משימות לפתרון הבעיה:

1. קליטת מספר ב-`num`

2. עבור כל מספר ברשימה בדיקה אם `num` זוגי. אם כן, הצגת המספר הנקלט פעמיים,

אחרת הצגת המספר הנקלט פעם אחת

נסתכל על התת-משימה השנייה: כפי שהיא מנוסחת, עבור כל מספר ברשימה אנו בודקים שוב אם `num` זוגי או לא, אף על פי שבעצם התשובה לשאלה זו תהיה זהה בכל פעם.

לכן, כדאי לבדוק רק פעם אחת אם `num` זוגי או לא.

הנה פירוק שני לתת-משימות לפתרון הבעיה:

1. קליטת מספר ב-`num`

2. אם `num` זוגי

2.1. עבור כל מספר ברשימה הצגת המספר הנקלט פעמיים

3. אחרת (אם `num` אי-זוגי)

3.1. עבור כל מספר ברשימה הצגת המספר הנקלט פעם אחת

אמנם כעת אנו בודקים רק פעם אחת אם `num` זוגי או לא, אבל אם נסתכל על התת-משימה השנייה והתת-משימה השלישית נראה ששתיהן כוללות קליטה של רשימת המספרים. כלומר, באלגוריתם שהתקבל מהפירוק הזה ולאחר מכן גם בתוכנית ליישום האלגוריתם, הוראות הקלט יופיעו פעמיים.



פירוק בהיר יותר לתת-משימות הוא הפירוק הבא:

1. קליטת מספר
  2. בדיקה אם המספר שנקלט זוגי ושמירת תוצאת הבדיקה
- 2.1. עבור כל מספר ברשימה:
    - 2.1.1. אם תוצאת הבדיקה חיובית (כלומר המספר הראשון שנקלט זוגי)
      - 2.1.1.1. הצגת המספר הנקלט פעמיים
      - 2.1.2. אחרת (כלומר המספר הראשון שנקלט אי-זוגי)
        - 2.1.2.1. הצגת המספר הנקלט פעם אחת

בפירוק זה נבדקת הזוגיות של ערך הקלט הראשון רק פעם אחת. גם המעבר על רשימת הערכים מתואר רק פעם אחת. עבור כל ערך קלט ברשימה, מספר המופעים בפלט ייקבע לפי תוצאת הבדיקה שבוצעה קודם לכן.

## בחירת משתנים

מהו טיפוס הערך המתאים לשמירת התוצאה של בדיקת הזוגיות לערך הראשון בקלט? הערכים האפשריים לתשובה לשאלה אם הערך הוא זוגי הם **כן** ו**לא**. טיפוס המתאים לערכים מסוג זה הוא טיפוס **בוליאני**. כזכור מפרקים קודמים, ערך מטיפוס בוליאני יכול להיות אחד משני הערכים: **true** או **false**.

עד כה ראינו ביטויים מטיפוס בוליאני ששימשו אותנו לתיאור תנאים. אבל ניתן גם להצהיר על משתנים מטיפוס בוליאני, ולשלב גם אותם בביטויים בוליאניים. אם נשמור את התוצאה של בדיקת הזוגיות לערך הקלט הראשון במשתנה בוליאני, נוכל להשתמש במשתנה זה כתנאי בהוראה לביצוע-בתנאי שעוברת על רשימת איברי הקלט ומעבדת אותם.

אם כך נשתמש במשתנים הבאים:

**num** – מטיפוס שלם, ישמור את ערך הקלט הראשון  
**numInList** – מטיפוס שלם, ישמור את ערך הקלט התורן מהרשימה  
**isEven** – מטיפוס בוליאני, ישמור את התוצאה של בדיקת הזוגיות ל-**num**

## האלגוריתם

מאחר שהרשימה מסתיימת בזקיף, נשתמש בהוראה לביצוע-חוזר-בתנאי התלוי בזקיף.

1. קלוט מספר ב-**num**
2. בדוק אם **num** זוגי/שמור את תוצאת הבדיקה ב-**isEven**
3. קלוט את המספר הראשון ברשימה ב-**numInList**
4. כל עוד **numList**  $\neq 0$  **כ3ע**:
  - 4.1. אם ערכו של **isEven** הוא **true**
    - 4.1.1. ה3ע פסמיים את **numInList**
    - 4.2. אגרא
    - 4.2.1. ה3ע פסם אגרא את **numInList**
    - 4.3. קלוט את המספר הכא ברשימה ב-**numInList**

## יישום האלגוריתם

הצהרה על משתנה מטיפוס בוליאני נעשית באמצעות שם הטיפוס `bool`. לכן ניתן להצהיר על המשתנה `isEven` באופן הבא:

```
bool isEven;
```

כמו בכל משתנה בתוכנית, גם במשתנים מטיפוס בוליאני אפשר לבצע השמה. הביטוי שבצד ימין של משפט ההשמה צריך להיות ביטוי בוליאני. אנו מעוניינים שהמשתנה `isEven` ישמור את תוצאת בדיקת הזוגיות של ערך הקלט הראשון, ולכן נוכל לכתוב את ההוראה הבאה לביצוע-בתנאי:

```
if (num % 2 == 0)
    isEven = true;
else
    isEven = false;
```

במקרה האחד אנחנו משימים במשתנה `isEven` את הערך הבוליאני `true` ובמקרה האחר את הערך הבוליאני `false`.

אבל מאחר שבצד ימין של משפט השמה למשתנה בוליאני אפשר לכתוב כל ביטוי בוליאני, ולא דווקא את הביטויים הפשוטים `true` או `false`, נוכל גם לכתוב את משפט ההשמה הבא:

```
isEven = (num % 2 == 0);
```

בעקבות ביצוע ההשמה שלעיל, יושם במשתנה `isEven` הערך `true` אם `num` זוגי, ואחרת יושם בו הערך `false`.

אנו מעוניינים לשלב את המשתנה `isEven` בביטוי הבוליאני בהוראה לביצוע-בתנאי. נוכל לכתוב כך:

```
if (isEven == true)
```

אבל מאחר שערכו של ביטוי בוליאני הוא `true` או `false`, וכך גם ערכו של משתנה בוליאני, הרי משתנה בוליאני הוא כבר ביטוי בוליאני בעצמו (בדיוק כמו שמשתנה שלם הוא כבר ביטוי חשבוני בעצמו). לכן נוכל גם לכתוב:

```
if (isEven)
```

## התוכנית המלאה

קלט: מספר שלם ולאחריו רשימת מספרים שלמים גדולים מ-1 שמסתיימת ב-0  
פלט: אם המספר הראשון זוגי יוצגו המספרים ברשימה פעמיים, אחרת יוצגו  
\*/ המספרים ברשימה פעם אחת בלבד

```
using System;
public class IfEven
{
    public static void Main ()
    {
        int num;                // המספר הראשון בקלט
        int numInList;           // מספר תורן ברשימת הקלט
        bool isEven;             // האם המספר הראשון זוגי
        // קליטת המספר הראשון ובדיקה האם הוא זוגי
        1. Console.WriteLine("Enter the first number: ");
        2. num = int.Parse(Console.ReadLine());
```

```

3.     isEven = (num % 2 == 0);
4.     Console.WriteLine("Enter a number. Enter 0 to end the list: ");
5.     numInList = int.Parse(Console.ReadLine());
6.     while (numInList != 0)
    {
6.1.         if (isEven)
6.1.1.             Console.WriteLine("{0} {1} ", numInList, numInList);
6.2.         else
6.2.1.             Console.WriteLine(numInList);
6.3.             Console.WriteLine("Enter the next number. Enter 0 to " +
                                     "end the list: ");
6.4.             numInList = int.Parse(Console.ReadLine());
    } // while
} // Main
} // IfEven

```

### שאלה 7.55

בנו טבלת מעקב אחר מהלך ביצוע התוכנית IfEven לפתרון בעיה 11 עבור הקלט: 4 5 7 0.

**סוף פתרון בעיה 11**

בפתרון הבעיה השתמשנו במשתנה מטיפוס **בוליאני**:

**משתנה בוליאני** שומר ערכים מטיפוס בוליאני, כלומר אחד משני הערכים **true** או **false**.  
 במשתנה בוליאני ניתן להשים ערך של ביטוי בוליאני כלשהו.  
 ניתן לשלב משתנה בוליאני בתוך תנאי בוליאני.  
 מתאים להשתמש במשתנה בוליאני כדי לזכור תוצאה של בדיקה.  
 ב-C# משמשת המילה השמורה **bool** להצהרה על משתנה בוליאני.  
 בדומה למשתנים מטיפוסים אחרים, ניתן לבצע אתחול למשתנה בוליאני בזמן ההצהרה, למשל:

```
bool boolVariable = true;
```

למשתנה בוליאני נוהגים לעתים לקרוא **דגל** (flag). כאשר ערכו של המשתנה **true** הוא משול לדגל מורם המסמן מעבר אפשרי. כאשר ערך המשתנה **false** הוא משול לדגל מורד המסמן כי אין אפשרות מעבר.

### שאלה 7.56

בקטע התוכנית הבא המשתנה **length** שומר מספר חיובי שלם המבטא אורך רשימה. הקטע קולט רשימת תוצאות הטלה של קובייה (תוצאת הטלה של קובייה היא מספר שלם בין 1 ל-6). רשימת התוצאות עלולה לכלול מספרים שגויים (כלומר שאינם בין 1 ל-6). המשתנה **valid** הוא מטיפוס בוליאני ושאר המשתנים הם מטיפוס שלם.

```

i = 1;
s = 0;
valid = true;
while (valid && (i <= length))
{
    Console.WriteLine("Enter the number on the die: ");
    die = int.Parse(Console.ReadLine());
}

```

```

    if((die >= 1) && (die <= 6))
        s = s + die;
    else
        valid = false;
        i = i + 1;
} // while
if (valid)
    Console.WriteLine(s);
else
    Console.WriteLine(die);

```

- א. תארו את הפלט עבור הקלט: 3 1 2 3.
- ב. תארו את הפלט עבור הקלט: 3 1 0 3.
- ג. מהו תפקיד המשתנה הבוליאני `valid`?
- ד. מהי מטרת קטע התוכנית?

### שאלה 7.57

פתחו אלגוריתם שהקלט שלו הוא רשימה של 20 מספרים חיוביים. הקלט נחשב חוקי אם כל המספרים בו הם זוגיים. האלגוריתם בודק אם הקלט חוקי. אם הקלט חוקי האלגוריתם מציג כפלט את סכום המספרים שבקלט, ואחרת תוצג הודעה כי הקלט איננו חוקי. ישמו את האלגוריתם בשפת `C#`.

**הדרכה:** השתמשו במשתנה בוליאני שיאותחל ב-`true`. אם יימצא בקלט מספר אי-זוגי יושם במשתנה זה הערך `false`.

הבעיה המוצגת בשאלה 7.57 מתאימה לתבנית **האם כל הערכים בסדרה מקיימים תנאי?** להעמקה בתבנית פנו לסעיף התבניות המופיע בסוף הפרק.

### שאלה 7.58

פתחו אלגוריתם שהקלט שלו הוא סיסמה, המורכבת מרשימת אותיות אנגליות המסתיימת ב-`!`. הפלט הוא הודעה המציינת אם הסיסמה תקינה. סיסמה תקינה כוללת לפחות 6 תווים, ובהם לפחות אות אחת קטנה (בתחום `a..z`) ולפחות אות אחת גדולה (בתחום `A..Z`). ישמו את האלגוריתם בשפת `C#`.

### שאלה 7.59

בקטע התוכנית הבא `length` מכיל מספר שלם חיובי. הקלט לקטע התוכנית הוא סדרה באורך `length` של מילים בנות שלוש אותיות כל אחת. מטרת קטע התוכנית היא להציג כל מילה בסדרה בסדר אותיות הפוך כל עוד אותיות המילה שונות זו מזו. ברגע שנקלטת מילה שבה לפחות שתי אותיות זהות יש לעצור את מהלך הביצוע. המשתנה `diff` הוא מטיפוס **בוליאני**, המשתנים `length` ו-`i` מטיפוס שלם והמשתנים `let1`, `let2` ו-`let3` הם מטיפוס תווי.

```

diff = _____
i = 1;
while(_____)
{
    Console.Write("Enter the first letter of the word: ");
    let1 = char.Parse(Console.ReadLine());
    Console.Write("Enter the second letter of the word: ");
    let2 = char.Parse(Console.ReadLine());

```

```

Console.WriteLine("Enter the third letter of the word: ");
let3 = char.Parse(Console.ReadLine());
if (_____)
    Console.WriteLine("{0} {1} {2}", let3, let2, let1);
else
    diff = _____
} // while

```

השלימו את קטע התוכנית.

## 7.6 הקשר הלוגי (not)

הביטויים הבוליאניים שהכרנו עד כה הם ביטויים פשוטים וביטויים מורכבים. הביטויים הבוליאניים הפשוטים שהכרנו כוללים קבועים (**true** או **false**), משתנים בוליאניים, או ביטויי השוואת ערכים בעזרת פעולות השוואה. ביטויים בוליאניים-מורכבים מתקבלים על ידי קישור של ביטויים פשוטים באמצעות קשרים לוגיים. עד כה הכרנו שני קשרים לוגיים: **or** ו-**and**.

נכיר כעת קשר לוגי נוסף, הוא הקשר **not**. כשמו כן הוא – הוא מבטא שלילה, היפוך התנאי. בניגוד לקשרים שהכרנו עד כה, הקשר **not** מופעל על ביטוי בוליאני אחד, ולא על שניים.

הנה טבלת האמת של הקשר **not**, כאשר **b** הוא ביטוי בוליאני:

b	b <b>not</b>
false	true
true	false

סדר העדיפות של הקשר **not** גבוה מסדר עדיפותם של הקשרים **and** ו-**or**. כלומר הוא קודם להם בחישוב ביטוי בוליאני.

בשפת **C#** הקשר **not** נכתב באמצעות סימן קריאה (!).

### דוגמאות

נניח ש-**a** הוא משתנה מטיפוס שלם ו-**b** הוא משתנה מטיפוס בוליאני

- ♦ ערכו של הביטוי הבוליאני **!true** הוא **false**.
- ♦ ערכו של הביטוי הבוליאני **(a == 1) !** הוא **false** כאשר ערכו של **a** שווה ל-1, ו-**!true** אחרת.
- ♦ ערכו של הביטוי הבוליאני **b !** הוא **false** כאשר ערכו של **b** הוא **true**. כאשר ערכו של **b** הוא **false**, ערכו של הביטוי **b !** הוא **true**.

### שאלה 7.60

נניח שהמשתנים **a** ו-**b** הם מטיפוס שלם. עבור כל ביטוי מן הביטויים הבוליאניים הבאים, תנו דוגמה לערכים למשתנים שעבורם יהיה ערך הביטוי **true**, ודוגמה לערכי המשתנים שעבורם יהיה ערך הביטוי **false**.

ערך הביטוי <b>true</b>	ערך הביטוי <b>false</b>	
a = b =	a = b =	א. ! (a != b)
a =	a =	ב. ! (Math.Abs(a) == a)

### שאלה 7.61

נניח שהמשתנים a ו-b הם מטיפוס שלם והמשתנה c הוא מטיפוס בוליאני. עבור כל ביטוי מן הביטויים שבמשפטי ההשמה הבאים, תנו דוגמה לערכי המשתנים שעבורם יושם ב-c הערך **true**, ודוגמה לערכי המשתנים שעבורם יושם ב-c הערך **false**.

ערך c הוא <b>true</b>	ערך c הוא <b>false</b>	
x =	x =	א. c = (x == Math.Sqrt(x))
a = b = c =	a = b = c =	ב. c = (c && (a == b))
a = b =	a = b =	ג. c = (! ((a + b) >= 5))

### שאלה 7.62

במשחק הניחושים מגריל המחשב מספר בתחום 1-100, והשחקן צריך לנחש אותו. בתום המשחק יודיע המחשב לשחקן כמה ניסיונות ניסה עד שהצליח לנחש. לפניכם קטע תוכנית ב-C#. השלימו את הקטע כך שיבצע את הנדרש:

```
Random rnd = new Random();
int num = _____;           // המספר שהמחשב יגריל/
int numOfGuesses = _____;   // מונה לספירת מספר ניחושים/
bool found = _____;
while (!found)
{
    numOfGuesses++;
    Console.WriteLine("Enter your guess: ");
    guess = int.Parse(Console.ReadLine());
    if (_____)
    {
        Console.WriteLine("Correct!! ");
        found = true;
    }
    else if (_____)
        Console.WriteLine("Your guess is too big");
    else
        Console.WriteLine("Your guess is too small");
} // while
Console.WriteLine("It took you {0} guesses", _____ );
```

### שאלה 7.63

הבעיה עוסקת בהצפנת הודעות. הודעה היא סדרת מילים, כך שכל מילה היא רצף אותיות מן הא"ב האנגלי ובין כל שתי מילים מופיע סימן קריאה (!). הצפנת הודעה מתבצעת באופן הבא: בכל מילה אי-זוגית (כלומר המילה הראשונה, השלישית, החמישית וכו') מוחלפת כל אות באות העוקבת לה בא"ב. בכל מילה זוגית (כלומר המילה השנייה, הרביעית, השישית וכו') מוחלפת כל אות באות הקודמת לה בא"ב. סימני הקריאה נותרים במקומם ללא שינוי. כלומר תחילה יש להצפין כל אות נתונה לאות העוקבת לה בא"ב ולאחר כל קריאה של סימן קריאה יש להפוך את "כיוון ההצפנה" (מעוקבת בא"ב לקודמת, או מקודמת בא"ב לעוקבת).

למשל, הצפנת ההודעה DING!DING!DONG! תהיה : EJOH!CHMF!EPOH.  
לשם הפשטות נניח שהאותיות A ו-Z אינן נכללות בהודעה.

פתחו וישמו אלגוריתם אשר הקלט שלו הוא מספר המציין אורך של הודעה (כלומר מספר התווים בהודעה) ואחריו ההודעה עצמה (הנקלטת תו אחר תו). הפלט שלו הוא ההודעה כשהיא מוצפנת באופן שתואר לעיל – הצפנת אות תופיע בפלט מיד לאחר קליטתה.

**הדרכה:** השתמשו באלגוריתם במשתנה מטיפוס בוליאני לשמירת "כיוון ההצפנה". אתחלו את ערכו של משתנה זה ל-true, ולאחר כל סימן קריאה הפכו את ערכו באמצעות הקשר  $\neg$ .

הקשר הבוליאני  $\neg$  משמש בתבנית **האם קיים ערך בסדרה המקיים תנאי?** להעמקה בתבנית זו פנו לסעיף התבניות המופיע בסוף הפרק.

## 7.7 קינון הוראות לביצוע-חוזר

בהוראה לביצוע-חוזר, גוף הלולאה יכול להכיל הוראות שונות. ייתכן כי בין הוראות אלו יש גם הוראות לביצוע-חוזר. בכך מתקבל מבנה מקונן, הכולל הוראה לביצוע-חוזר בתוך הוראה לביצוע-חוזר. הדבר דומה לקינון של הוראות לביצוע-בתנאי, שהכרנו בפרק 5. בסעיף זה נראה כי מבנה מקונן של הוראות לביצוע-חוזר הוא שימושי מאוד בפתרון בעיות מסוימות.

### 8.12

**מטרת הבעיה ופתרונה:** הצגת הוראה מקוננת לביצוע-חוזר.

פתחו וישמו אלגוריתם אשר הפלט שלו הוא לוח הכפל, מוצג באופן הבא:

1	2	3	.....	10
2	4	6	.....	20
3	6	9	.....	30
.	.	.	.	.
.	.	.	.	.
10	20	.....	100	

בבעיה הנתונה אין קלט. מוגדר פלט בלבד. הפלט הוא טבלת המספרים של לוח הכפל. טבלה זו מכילה 100 מספרים, המסודרים בעשר שורות ובעשרה טורים.

### פירוק הבעיה לתת-משימות

דרך אחת להצגת לוח הכפל היא על ידי אלגוריתם ובו 10 לולאות, אשר כל אחת מהן תציג כפלט שורה של 10 מספרים.

כלומר, נקבל את הפירוק הבא לתת-משימות:

1. הצגה כפלט בשורה אחת של המספרים 1 עד 10
2. הצגה כפלט בשורה אחת של המספרים 1 עד 10, מוכפלים ב-2
3. הצגה כפלט בשורה אחת של המספרים 1 עד 10, מוכפלים ב-3

.

.

10. הצגה כפלט בשורה אחת של המספרים 1 עד 10, מוכפלים ב-10

זהו ניסוח מסורבל וארוך. בנוסף ניסוח זה אינו כללי: כדי להציג טבלה בעריכה שונה או טבלה חלקית (למשל רק את חמש השורות הראשונות) יש לכתוב אלגוריתם שונה. בנוסף קשה להכליל את האלגוריתם הזה לבעיה מעט שונה, שגודל הטבלה בה (מספר השורות ומספר העמודות) מתקבל כקלט ואינו ידוע בעת כתיבת האלגוריתם.

האם ניתן לכתוב אלגוריתם פשוט, קצר יותר, ובצורת ניסוח כללית יותר? כן!

בכל אחת מהלולאות המפורטות מתואר ביצוע-חוזר של תת-משימה. אך בעצם ניתן להתייחס לכל לולאה כאל תת-משימה כוללת יותר, אשר גם על ביצועה יש לחזור 10 פעמים. כלומר, ניתן לתאר זאת בתת-המשימה הבאה שאותה יש לבצע 10 פעמים:

הצגת השורה הבאה בתור בטבלה

כאשר הצגה של שורה בטבלה אף היא מבוטאת על ידי תת-משימה שיש לבצע 10 פעמים:

הצגת האיבר הבא בתור בשורה

### בחירת משתנים

מאחר שיש לנו צורך בשתי הוראות לביצוע-חוזר (האחת בתוך האחרת) ואורכן ידוע מראש, נזדקק לשני משתני הבקרה מטיפוס שלם:  $i$  ו- $j$ .

### האלגוריתם

1. עבור  $i$  שלם מ-1 עד 10:  
1.1. עבור  $j$  שלם מ-1 עד 10:  
1.1.1. הצג כפול  $i*j$   
1.2. עבור  $j$  לאורה הבאה

### יישום האלגוריתם

/\* התוכנית מציגה כפלט את לוח הכפל של 10\*10 \*/

using System;

public class MultTable

{

public static void Main ()

{

const int TABLE\_DIMENSION = 10;

1. Console.WriteLine("Mult Table");

2. for (int i = 1; i <= TABLE\_DIMENSION; i++)

{

2.1. for (int j = 1; j <= TABLE\_DIMENSION; j++)

{

2.1.1. Console.Write("{0,3} ", i \* j);

} // for j

2.2. Console.WriteLine(); // מעבר לשורת הפלט הבאה

} // for i

} // Main

} // MultTable

הוספת מספר כלשהו לסמן בהוראת פלט, כגון {0,3} משמעו הקצאת 3 מקומות בדיוק על המסך לצורך כתיבת הפלט



## שאלה 7.64

בנו טבלת מעקב אחר מהלך ביצוע התוכנית MultTable.

### סוף פתרון מצ'ה 12

**שימו ♥ :** הפלט מודפס באופן לא מושלם כיוון שמספרים שונים עלולים לתפוס מקום שונה על המסך. למשל, מספרים חד-ספרתיים תופס פחות מקום מאשר מספר דו-ספרתי. כדי ליצור טבלה מסודרת, שעמודותיה ממוקמות באופן אחיד, ניתן לשנות את הוראת הפלט, כך שלכל מספר יוקצו בדיוק 3 מקומות. לשם כך נוסיף לסמן שבהוראת הפלט שבשורה 2.1.1 את הערך 3 :

```
Console.WriteLine("{0,3} ", i * j);
```

**במבנה מקוון של לולאות נכללת לולאה פנימית בתוך גוף של לולאה חיצונית.**  
הלולאה הפנימית מתבצעת כולה בכל סיבוב של גוף הלולאה החיצונית.

בפתרון בעיה 12 הביצוע-החוזר של הלולאה החיצונית כלל 10 ביצועים-חוזרים של גוף הלולאה הפנימית. כיוון שהלולאה החיצונית מתבצעת 10 פעמים, הרי מספר הפעמים הכולל שיתבצע גוף הלולאה הפנימית הוא  $10 \times 10 = 100$ .

**שימו ♥ :** קינזן אינו מוגבל רק להוראות לביצוע-חוזר שמספר החזרות בהן ידוע מראש. ההוראה החיצונית יכולה להיות גם הוראה לביצוע-חוזר שתלויה בתנאי, וכך גם ההוראה הפנימית.

בכתיבת מבנה מקוון של לולאות נקפיד על **הזחה**. כיוון שהלולאה הפנימית היא חלק מגוף הלולאה החיצונית, נזיח את הלולאה הפנימית ביחד עם ההוראות של גוף הלולאה החיצונית. הדבר דומה להזחה שבמבנה המקוון של ביצוע-בתנאי.

## שאלה 7.65

נניח שבאלגוריתם דומה לאלגוריתם המתואר בבעיה 12, משתנה הבקרה של הלולאה החיצונית  $i$  יתחיל מהערך 2, ויגדל בכל ביצוע של הלולאה ב-1, עד הגיעו לערך 6. מה יהיה הפלט של אלגוריתם זה? ומה יהיה מספר הפעמים הכולל שיתבצע גוף הלולאה הפנימית שבו?

## שאלה 7.66

תארו עבור כל אחד מקטעי התוכניות הבאים את פלט קטע התוכנית ואת מספר הפעמים הכולל שמתבצע גוף הלולאה הפנימית :

א.	ב.
<pre>for (int i = 1; i &lt;= 5; i++) {     for (int j = 1; j &lt;= 5; j++)         Console.WriteLine("");     Console.WriteLine(); }</pre>	<pre>for (int i = 1; i &lt;= 5; i++) {     for (int j = 1; j &lt;= i; j++)         Console.WriteLine("");     Console.WriteLine(); }</pre>
ג.	ד.
<pre>for (int i = 1; i &lt;= 5; i++) {     for (int j = 1; j &lt;= 5; j++)         Console.WriteLine(j);     Console.WriteLine(); }</pre>	<pre>for (int i = 1; i &lt;= 5; i++) {     for (int j = 5; j &gt;= i; j--)         Console.WriteLine("");     Console.WriteLine(); }</pre>

## שאלה 7.67

עבור כל אחד מהסעיפים הבאים פתחו וישמו אלגוריתם אשר הפלט שלו הוא כפי שמופיע בתיאור הסעיף. בכל אחד מהאלגוריתמים מותר לכתוב **רק הוראה אחת** המדפיסה תו או ספרה **והוראה אחת** למעבר שורה בפלט (מכאן נובע כי עליכם להשתמש בלולאות מקוננות) :

	ד.	ג.	ב.	א.
1	*****	*****	*	
12	****	****	**	
123	***	***	***	
1234	**	**	****	
12345	*	*	*****	
	**			
	***			
	****			
	*****			
	ח.	ז.	ו.	ה.
11111	55555	12345	1	
2222	4444	1234	22	
333	333	123	333	
44	22	12	4444	
5	1	1	55555	

## שאלה 7.68

בכיתה 40 תלמידים, כל תלמיד לומד 20 מקצועות. לפניכם תוכנית המחשבת עבור כל תלמיד את ממוצע ציוניו :

```

/*
קלט: 20 הציונים עבור כל אחד מ-40 תלמידי הכיתה
פלט: הממוצע של כל תלמיד ותלמיד
*/
using System;
public class Grades
{
    public static void Main ()
    {
        const int STUDENT_NUM = 40;
        const int GRADE_NUM = 20;
        double grade;
        double sum;
        double average;
        for(int i = 1; i <= STUDENT_NUM; i++)
        {
            for(int j = 1; j <= GRADE_NUM; j++)
            {
                Console.WriteLine("Enter next student grade: ");
                grade = int.Parse(Console.ReadLine());
                sum = sum + grade;
            } // for j
        } // for i
    } // Main
} // Grades

```

המשפטים הבאים חסרים בתוכנית הנתונה, היכן צריך לשלב את המשפטים האלה כדי שהתוכנית תבצע את הנדרש?

א. `sum = 0;`

ב. `Console.WriteLine(sum / 20);`

### שאלה 7.69

פתחו אלגוריתם שמקבל כקלט מספר חיובי של `sum`, ולאחריו `sum` סדרות באורך לא ידוע של מספרים חיוביים שלמים, כאשר כל סדרה מסתיימת ב-1. האלגוריתם יציג כפלט את אורכה של הסדרה הארוכה ביותר. ישמו את האלגוריתם בשפת C#.

### שאלה 7.70

במפעל לייצור נעליים יש עובדים רבים. פתחו אלגוריתם שיקבל כקלט את מספר העובדים במפעל, ואחר כך רשימה של כל המשכורות של השנה האחרונה (12 משכורות) **לכל אחד** מעובדי המפעל. האלגוריתם יחשב וידפיס את **המשכורת האחרונה** של העובד המסכן שסכום משכורותיו כל השנה היה הנמוך ביותר. ישמו את האלגוריתם בשפת C#.

בעזרת החומר הנלמד בפרק 7 ניתן לפתור גם בעיות המשתמשות בתבניות **מציאת כל הערכים בסדרה המקיימים תנאי ומעבר על זוגות סמוכים בסדרה**. להעמקה בתבניות אלו פנו לסעיף התבניות המופיע בסוף הפרק.

## סיכום

בפרק זה למדנו כיצד להורות על ביצוע-חוזר של תת-משימה. הדבר נעשה באמצעות **הוראה לביצוע-חוזר**, והיא הוראת בקרה הנקראת גם **לולאה**.

הכרנו שני מבנים של הוראה לביצוע-חוזר: **הוראה לביצוע-חוזר מספר פעמים ידוע מראש**, ו**הוראה לביצוע-חוזר בתנאי**.

**הוראה לביצוע-חוזר באורך ידוע מראש** נכתבת בצורה:

**for (i = 0; i < 20; i++)**  
**do**

או

**for (i = 0; i < 20; i++)**  
**do**

במבנה הראשון סדרת ההוראות לביצוע מתבצעת X פעמים.

במבנה השני איבריו של התחום המוגדר נסרקים, ועבור כל אחד מהם מבוצעת סדרת ההוראות לביצוע.

**הוראה לביצוע-חוזר בתנאי** נכתבת בצורה:

**while (i < 20)**  
**do**

התנאי המתואר נקרא **תנאי הכניסה** ללולאה. כל עוד התנאי מתקיים, סדרת ההוראות לביצוע מתבצעת שוב ושוב. כאשר התנאי לא מתקיים, מסתיימת ההוראה לביצוע-החוזר.

בכתיבת ההוראה לביצוע-חוזר (מספר פעמים ידוע מראש או בתנאי) אנו מקפידים על **הזחה**: קבוצת ההוראות אשר יש לחזור על ביצוען כתובות כשהן מוזחות פנימה.

קבוצת ההוראות אשר יש לחזור על ביצוען נקראת **גוף הלולאה**.

כאשר מספר הפעמים לביצוע-חוזר אינו ידוע מראש (לפני תחילת ביצוע הלולאה), והוא תלוי בתנאי, נשתמש בהוראה לביצוע-חוזר-בתנאי. בכל מקרה אחר נשתמש בהוראה לביצוע-חוזר באורך ידוע מראש. בכך נסייע ביצירת תוכניות קריאות ובהירות.

בין ההוראות לביצוע-חוזר-בתנאי הבחנו בסוג מסוים: **הוראות לביצוע-חוזר התלויות בזקיף**.

**זקיף** הוא סימן מיוחד המציין את סוף רשימת איברי הקלט. הזקיף **אינו** נחשב כחלק מהקלט. לכן בהוראה לביצוע-חוזר-בתנאי, יש לוודא שאיבר הקלט התורן אינו הזקיף כדי שלא נעבד אותו כחלק מהקלט. זיהוי הזקיף צריך להביא לסיום הביצוע-החוזר.

הוראה לביצוע-חוזר משמשת, בין השאר לביצוע **פעולות צבירה ומנייה**. צבירה נעשית באמצעות צובר, ומנייה באמצעות מונה.

**צובר** הוא משתנה שתפקידו לצבור ערכים (למשל נתונים או תוצאות חישוב). צובר יכול לשמש לפעולות צבירה שונות (למשל, סכום מצטבר או מכפלה מצטברת).

**מונה** הינו משתנה אשר תפקידו למנות אירועים (למשל, מספר המופעים של נתונים). מאחר שהשימוש במונה הוא לצורך ספירה הרי הוא בדרך כלל מטיפוס שלם.

באלגוריתמים שיש בהם שימוש במונה או צובר יש להקפיד עבורם על **אתחול**. באתחול יושם במונה או בצובר ערך התחלתי המתאים להגדרת תפקידם.

הוראות לביצוע-חוזר משמשות גם לפתרון בעיות שנדרש למצוא בהן **איבר מקסימלי** או **איבר מינימלי** בתוך קבוצת איברים, או ערך נלווה (כמו **מיקומו**) של האיבר המקסימלי (או המינימלי) בקבוצת איברים סדורה.

כאשר אלגוריתם כולל לולאה ניתן לחשב את **מספר הפעמים שהלולאה תתבצע**. מספר זה תלוי בדרך כלל בקלט לאלגוריתם.

יתכן שעבור קלטים מסוימים תתבצע הלולאה אינסוף פעמים. לולאה כזאת נקראת **לולאה אינסופית**. אנו מחשיבים לולאה אינסופית כלולאה שגויה.

בפרק הבא נדגיש את החשיבות של פתרון בעיות באמצעות אלגוריתמים שבהם מספר הפעמים לביצוע-חוזר הוא קטן ככל האפשר.

**משתנה בוליאני** הוא משתנה שיכול לקבל אחד משני הערכים **true** ו-**false**. ניתן להשתמש במשתנים בוליאניים כחלק מביטויים בוליאניים.

**הקשר הלוגי**  $\neg$  (not) משמש ליצירת ביטויים בוליאניים המורכבים מביטויים פשוטים יותר (בדומה לקשרים  $\wedge$  (and) ו- $\vee$  (or)). הוא מופעל על ביטוי בוליאני והוא הופך את ערכו הלוגי (מ-**true** ל-**false**, ולהיפך). עדיפותו של הקשר הלוגי  $\neg$  גבוהה מעדיפותם של הקשרים  $\wedge$  ו- $\vee$ .

הוראה שנמצאת בתוך הוראה לביצוע-חוזר יכולה להיות בעצמה הוראה לביצוע-חוזר. בכך מתקבל **קינון של הוראות לביצוע-חוזר**.

## סיכום מרכיבי שפת C# שנלמדו בפרק 7

הוראה לביצוע-חוזר מספר פעמים ידוע מראש מיושמת ב-C# במשפט `for`.

המבנה הכללי של משפט `for` הוא:

```
for (שינוי משתנה הבקרה ; התנאי להמשך הביצוע ; אתחול משתנה הבקרה)
{
    הוראות לביצוע
}
```

כל עוד התנאי להמשך הביצוע (שתלוי בדרך כלל בערכו של משתנה הבקרה) מתקיים, ביצוע הלולאה ממשיך, כלומר מתבצעת סדרת ההוראות לביצוע. סדרת הוראות זו נקראת **גוף הלולאה**.

משתנה הבקרה הוא בדרך כלל מטיפוס שלם. אם אין בו שימוש מעבר לתחום הלולאה ניתן להצהיר עליו בתוך הלולאה, למשל כך:

```
for(int i = 1; i <= 10; i++)
```

הוראה לביצוע-חוזר-בתנאי מיושמת ב-C# במשפט `while`.

המבנה הכללי של משפט `while` הוא:

```
while (ביטוי בוליאני)
{
    הוראות לביצוע
}
```

**ביצוע משפט `while`** מתחיל בחישוב ערכו של הביטוי הבוליאני. אם ערכו `true` מתבצעות ההוראות שבגוף הלולאה. בתום ביצוע גוף הלולאה מחושב ערכו של הביטוי הבוליאני שוב. אם ערכו `true` מתבצעות שוב ההוראות שבגוף הלולאה, וכך הלאה כל עוד ערכו של הביטוי הבוליאני הוא `true`. כאשר ערכו `false` מסתיים ביצוע משפט ה-`while`.

הצהרה על **משתנה בוליאני** נעשית בשפת C# באמצעות המילה השמורה `bool`, למשל כך:

```
bool flag;
```

**הקשר** נכתב ב-C# באמצעות הסימן `!`, למשל כך:

```
!(x == 5)
```

## תבניות – פרק 7

פירוט מלא של התבניות ושל שאלות שבפתרון יש שימוש בתבניות ניתן למצוא באתר הספר ברשת האינטרנט.

בסעיף זה מוצגות תבניות שונות, חלקן מתייחסות לסדרת קלט שאורכה ידוע מראש וחלקן מתייחסות לסדרת קלט שאורכה אינו ידוע מראש.

### מנייה

שם התבנית: מנייה

נקודת מוצא: אורך סדרת נתוני הקלט limit, סדרת ערכי הקלט, תנאי מנייה condition  
מטרה: מנייה של ערכי הקלט המקיימים את התנאי condition, מתוך סדרת קלט שאורכה limit  
אלגוריתם:

1. אגף אל count 0-2

2. כצע limit פסמים:

2.1 קאוט ערך 2-element

2.2 אס element מקיים אל condition

2.2.1 הצד אל count 1-2

שם התבנית: מנייה

נקודת מוצא: תנאי סיום conditionToEnd, ערכי הקלט, תנאי מנייה conditionToCount  
מטרה: מנייה של ערכי הקלט המקיימים את התנאי conditionToCount. משך ביצוע המנייה תלוי בתנאי conditionToEnd.

אלגוריתם:

1. אגף אל count 0-2

2. קאוט ערך 2-element

3. כל ע/ד לא מקיים conditionToEnd כצע:

3.1 אס element מקיים אל conditionToCount

3.1.1 הצד אל count 1-2

3.2 קאוט ערך 2-element

## צבירת סכום

שם התבנית: **צבירת סכום**

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערך התחלתי של הצובר initial, ערכי הקלט,

תנאי צבירה condition

מטרה: צבירת הסכום של הערך initial ושל ערכי הקלט המקיימים את התנאי condition, מתוך

סדרת קלט שאורכה limit

אלגוריתם:

1. אגור אגור אגור initial-sum

2. בצע limit פעמים:

2.1 קוטט ערך element

2.2 אס element מקיים אגור condition

2.2.1 הוסף ל-sum אגור ערכו element

שם התבנית: **צבירת סכום**

נקודת מוצא: תנאי סיום conditionToSum, ערך התחלתי של הצובר initial, ערכי הקלט,

תנאי צבירה conditionToSum

מטרה: צבירת סכום של הערך initial ושל ערכי הקלט המקיימים את התנאי conditionToSum.

משך ביצוע הצבירה תלוי בתנאי conditionToSum.

אלגוריתם:

1. אגור אגור initial-sum

2. קוטט ערך element

3. כל עוד לא מקיים conditionToSum בצע:

3.1 אס element מקיים אגור conditionToSum

3.1.1 הוסף ל-sum אגור ערכו element

3.2 קוטט ערך element

## צבירת מכפלה

שם התבנית: **צבירת מכפלה**

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערך התחלתי של הצובר initial, ערכי הקלט,

תנאי צבירה condition

מטרה: צבירת מכפלה של הערך initial ושל ערכי הקלט המקיימים את התנאי condition, מתוך

סדרת קלט שאורכה limit

אלגוריתם:

1. אגור אגור initial-mult

2. בצע limit פעמים:

2.1 קוטט ערך element

2.2 אס element מקיים אגור condition

2.2.1 הכפל אגור mult element והשם אגור המכפלה mult

שם התבנית : צבירת מכפלה

נקודת מוצא : תנאי סיום conditionToEnd, ערך התחלתי של הצובר initial, ערכי הקלט, תנאי צבירה conditionToMult

מטרה : צבירת מכפלה של הערך initial ושל ערכי הקלט המקיימים את התנאי conditionToMult. משך ביצוע הצבירה תלוי בתנאי conditionToEnd.

אלגוריתם :

1. אגור אל mult initial-2

2. קלוט ערך element-2

3. כן עוצר לא מקיים conditionToEnd 33ע:

3.1. אס element מקיים אל conditionToMult

3.1.1. הכפל אל mult element-2 והשם אל המכפלה mult-2

3.2. קלוט ערך element-2

## ממוצע של סדרת מספרים

שם התבנית : ממוצע של סדרת מספרים

נקודת מוצא : תנאי סיום conditionToEnd, ערכי הקלט, תנאי conditionToInclude  
מטרה : חישוב ממוצע של ערכי הקלט המקיימים את התנאי conditionToInclude. ביצוע החישוב תלוי בתנאי conditionToEnd.

אלגוריתם :

1. אגור אל count 0-2

2. אגור אל sum 0-2

3. קלוט ערך element-2

4. כן עוצר לא מקיים conditionToEnd 33ע:

4.1. אס element מקיים אל conditionToInclude

4.1.1. העדף אל count 1-2

4.1.2. הוסף לערכו אל sum element

4.2. קלוט ערך element-2

5. השם ב-average אל ערכו אל הביטוי השבועי sum/count



## מציאת מקסימום או מינימום בסדרה

בשל הדמיון האלגוריתמי בין מציאת מקסימום למציאת מינימום וכדי לא לחזור בפירוט על התבניות, נפרט את התבניות באופן הבא: עבור התבנית **מציאת מקסימום** נתייחס לסדרה שאורכה ידוע מראש ואילו עבור התבנית **מציאת מינימום** נתייחס לסדרה שאורכה אינו ידוע מראש.

שם התבנית: **מציאת מקסימום בסדרה**

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערכי הקלט

מטרה: מציאת הערך הגדול ביותר בסדרת ערכי הקלט שאורכה הוא limit אלגוריתם:

1. קאנט ערך-2 max

2. בצע limit-1 פעמים:

2.1 קאנט ערך-2 element

2.2 אם  $\text{element} > \text{max}$

2.2.1 השם max-2 את הערך של element

שם התבנית: **מציאת מינימום בסדרה**

נקודת מוצא: תנאי סיום conditionToEndMinSearch, ערכי הקלט

מטרה: מציאת הערך הקטן ביותר מבין ערכי הקלט, עד אשר מתקיים התנאי conditionToEndMinSearch אלגוריתם:

1. קאנט ערך-2 min

2. קאנט ערך-2 element

3. כן עדיף לא מתקיים conditionToEndMinSearch בצע:

3.1 אם  $\text{element} < \text{min}$

3.1.1 השם min-2 את הערך של element

3.2 קאנט ערך-2 element

## מציאת ערך נלווה למקסימום או למינימום בסדרה

גם כאן בשל הדמיון האלגוריתמי בין התבניות, נפרט את התבניות באופן הבא: עבור התבנית **מציאת ערך נלווה למקסימום בסדרה** נתייחס לסדרה שאורכה ידוע מראש ואילו עבור התבנית **מציאת ערך נלווה למינימום בסדרה** נתייחס לסדרה שאורכה אינו ידוע מראש. בתבניות הבאות נמצא מיקום של איבר כערך נלווה.

שם התבנית: **מציאת ערך נלווה למקסימום בסדרה**  
נקודת מוצא: אורך סדרת נתוני הקלט limit, ערכי הקלט  
מטרה: מציאת מיקום הערך הגדול ביותר בסדרת ערכי הקלט שאורכה הוא limit אלגוריתם:

1. קאוט ערך  $\text{max}$ -2
2. אגה  $\text{placeOfMax}$  1-2
3. בצע limit-1 פעמים:
  - 3.1. קאוט ערך  $\text{element}$ -2
  - 3.2. אם  $\text{element} > \text{max}$ 
    - 3.2.1. השם  $\text{max}$ -2 אה הערך  $\text{element}$
    - 3.2.2. השם  $\text{placeOfMax}$ -2 אה מיקומו  $\text{element}$  בקלט

שם התבנית: **מציאת ערך נלווה למינימום בסדרה**  
נקודת מוצא: תנאי סיום  $\text{conditionToEnd}$ , ערכי הקלט  
מטרה: מציאת מיקום הערך הקטן ביותר מבין ערכי הקלט, עד אשר מתקיים התנאי  $\text{conditionToEnd}$   
אלגוריתם:

1. קאוט ערך  $\text{min}$ -2
2. אגה  $\text{placeOfMin}$  1-2
3. קאוט ערך  $\text{element}$ -2
4. אגה  $\text{currentPlace}$  2-2
5. כל עוד לא מתקיים  $\text{conditionToEnd}$  בצע:
  - 5.1. אם  $\text{element} < \text{min}$ 
    - 5.1.1. השם  $\text{min}$ -2 אה הערך  $\text{element}$
    - 5.1.2. השם  $\text{placeOfMin}$ -2 אה הערך  $\text{currentPlace}$
  - 5.2. העבד  $\text{currentPlace}$  1-2
  - 5.3. קאוט ערך  $\text{element}$ -2

## איסוף בקיזוז

שם התבנית : איסוף בקיזוז באמצעות מנייה

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערכי הקלט, תנאי לקיזוז conditionToDecrease  
מטרה: איסוף בקיזוז באמצעות מנייה של ערכי הקלט מתוך סדרה שאורכה limit. הקיזוז מתבצע על פי התנאי conditionToDecrease  
אלגוריתם:

1. אגף אף count 0-2
2. בצע limit פעמים:
  - 2.1 קלט ערך element-2
  - 2.2 אם element לא מקיים אף conditionToDecrease
    - 2.2.1 העדף אף count 1-2
    - 2.3 אגף
      - 2.3.1 הקטן אף count 1-2

שם התבנית : איסוף בקיזוז באמצעות צבירה

נקודת מוצא: תנאי סיום conditionToEnd, ערכי הקלט, ערך צבירה התחלתי initial, תנאי לקיזוז conditionToDecrease  
מטרה: איסוף בקיזוז באמצעות צבירת סכום של ערכי הקלט ושל הערך initial. משך הביצוע תלוי בתנאי conditionToEnd, והקיזוז מתבצע על פי התנאי conditionToDecrease  
אלגוריתם:

1. אגף אף sum 0-2 initial
2. קלט ערך element-2
3. כל עוד לא מקיים conditionToEnd בצע:
  - 3.1 אם element לא מקיים אף conditionToDecrease
    - 3.1.1 הוסף ל-sum אף ערכו של element
    - 3.2 אגף
      - 3.2.1 הפגם sum-M אף ערכו של element
      - 3.3 קלט ערך element-2

## פירוק מספר חיובי לספרותיו

שם התבנית : פירוק מספר חיובי לספרותיו

נקודת מוצא: מספר שלם חיובי num  
מטרה: הצגה כפלט של ספרותיו של num  
אלגוריתם:

1. כל עוד num שונה מ-0 בצע
2. הצג כפלט אף ספרת האחדות של num
3. הקטן אף num פי 10

## בניית מספר

שם התבנית : בניית מספר

נקודת מוצא : אורך סדרת נתוני הקלט limit, ספרות הקלט  
מטרה : בניית מספר מספרות הקלט  
אלגוריתם :

1. אגף אג num ב-0

2. בצע limit פעמים:

2.1 קאוט ספירה ב-digit

2.2 השם ב-num אג הערך של הביטוי  $num * 10 + digit$

## האם כל הערכים בסדרה מקיימים תנאי?

שם התבנית : האם כל הערכים בסדרה מקיימים תנאי?

נקודת מוצא : אורך סדרת נתוני הקלט limit, ערכי הקלט, תנאי condition  
מטרה : קביעת הערך true אם כל הערכים בסדרה מקיימים את התנאי condition וקביעת הערך false אם קיים ערך אחד בסדרה שאינו מקיים את התנאי  
אלגוריתם :

1. אגף אג all ב-true

2. אגף אג howmany ב-1

3. כל ע/צ  $howmany \leq limit$  ואם ערכו של all הוא true בצע:

3.1 קאוט ערך ב-element

3.2 הצב אג ערכו של howMany ב-1

3.3 אס element אינו מקיים אג condition

3.3.1 השם ב-all אג הערך false

## האם קיים ערך בסדרה המקיים תנאי?

שם התבנית : האם קיים ערך בסדרה המקיים תנאי?

נקודת מוצא : תנאי לסיום הסדרה toEnd, ערכי הקלט, תנאי condition  
מטרה : קביעת הערך true אם קיים ערך בסדרה המקיים את התנאי condition וקביעת הערך false אם כל הערכים בסדרה אינם מקיימים את התנאי  
אלגוריתם :

1. אגף אג found ב-true

2. קאוט ערך ב-element

3. כל ע/צ הגנאי toEnd לא מקיים ואם ערכו של found הוא false בצע:

3.1 אס element מקיים אג condition

3.1.1 השם ב-found אג הערך true

3.2 אגף

3.2.1 קאוט ערך ב-element

## מציאת כל הערכים בסדרה המקיימים תנאי

שם התבנית : מציאת כל הערכים בסדרה המקיימים תנאי  
נקודת מוצא : תנאי לסיום הסדרה toEnd, ערכי הקלט, תנאי condition  
מטרה : הצגה כפלט של כל ערכי הקלט המקיימים את התנאי condition  
אלגוריתם :

1. קאוט ערך ב-element
2. כל ערך הנאי toEnd לא מקיים בצע:
- 3.1. אם element מקיים את condition
- 3.1.1. הצג כפלט את ערכו של element
- 3.2. קאוט ערך ב-element

## מעבר על זוגות סמוכים בסדרה

שם התבנית : מעבר על זוגות סמוכים בסדרה  
נקודת מוצא : אורך סדרת נתוני הקלט limit, ערכי הקלט  
מטרה : הצגה כפלט של סכומי כל זוגות הערכים הסמוכים בסדרת הקלט שאורכה limit  
אלגוריתם :

1. קאוט ערך ב-beforeLast
2. בצע limit-1 פעמים
- 2.1. קאוט ערך ב-last
- 2.2. הצג כפלט את הערך של הביטוי beforeLast + last
- 2.3. השם ב-beforeLast את הערך של last

## תבניות – פרק 7

### מנייה וצבירה

נתבונן בשתי הבעיות האלגוריתמיות הבאות:

**בעיה 1:** עליה על מתקן "רכבת הרים" מותרת רק לילדים שגובהם עולה על 1.40 מטר וגילם עולה על 8. כתבו אלגוריתם שהקלט שלו 40 זוגות ערכים המייצגים את הגובה והגיל של 40 ילדים, והפלט שלו הוא מספר הילדים הרשאים לעלות למתקן.

**בעיה 2:** מנהל האתר הבית-ספרי "ישראלנד" מעוניין לדעת מהו מספר הכניסות לאתר ביממה. כתבו אלגוריתם שהקלט שלו הוא סדרה של קודי משתמשים, המסתיימת בזקוף 0, והפלט שלו הוא מספר הכניסות לאתר ביממה.

בשתי הבעיות האלגוריתמיות עלינו **למנות** (לספור) ערכים. בבעיה 1 יש למנות את מספר הילדים הרשאים לעלות למתקן "רכבת הרים" ובבעיה 2 יש למנות את מספר הכניסות לאתר ביממה. מנייה היא אחת התבניות השימושיות ביותר בפיתוח אלגוריתמים. השימוש בתבנית זו נעשה בכל פעם שיש למנות כמות הופעות של ערך.

נתבונן בשני האלגוריתמים הבאים, הפותרים את שתי הבעיות שלעיל:

1. אגל אג 0-2 count	1. אגל אג 0-2 count
2. קאול קוד מלמל code-2	2. 40 פסמל:
3. 0 < code פל 232:	2.1. קאול גובה 2 height-גל age-2
3.1. 1-2 count אג הלל	2.2. אג 8 > age height > 1.4 אג
3.2. קאול קוד מלמל code-2	2.2.1. 1-2 count אג הלל
4. הלל כפלט אג ערכו 0 count	3. הלל כפלט אג ערכו 0 count

בשני האלגוריתמים ערכו של המונה (count) מאותחל לפני הלולאה ל-0 ובגוף הלולאה גדל ערכו של המונה ב-1. שימו לב כי בפתרון לבעיה 1 מספר הפעמים שמבוצע גוף הלולאה ידוע מראש (40) ואילו בפתרון לבעיה 2 מספר הפעמים שמבוצע גוף הלולאה אינו ידוע מראש (סדרת נתוני הקלט מסתיימת עם קליטת הזקוף 0).

עתה נתבונן בשתי הבעיות האלגוריתמיות הבאות :

**בעיה 3 :** כתבו אלגוריתם שהקלט שלו הוא 15 מספרים ממשיים והפלט שלו הוא סכום המספרים החיוביים.

**בעיה 4 :** בתחרות הרמת משקולות משקלה של המשקולת ההתחלתית הוא 20 ק"ג. כל המתחרים בתחרות מסוגלים להרים משקולת זו. עם התקדמות התחרות נוספות עוד ועוד משקולות למשקולת ההתחלתית. כתבו אלגוריתם שהקלט שלו הוא סדרה של זוגות נתונים : הנתון הראשון הוא משקל המשקולת אותה מוסיפים למשקל המצטבר שהרים מתחרה, והנתון השני הוא התו 'y' אם המתחרה הצליח להרים את המשקל המצטבר כולל המשקולת החדשה, והתו 'n' אם לא הצליח. התו 'n' יופיע רק בזוג הנתונים האחרון. הפלט של האלגוריתם הוא המשקל המצטבר של המשקולות שהמתחרה הצליח להרים.

בשתי הבעיות האלגוריתמיות עלינו **לצבור** (בדוגמה זו, לחשב סכום) ערכים. בבעיה 3 יש לצבור מספרים חיוביים ובבעיה השנייה יש לצבור את משקלי המשקולות. תבנית הצבירה, בדומה לתבנית מנייה, הינה תבנית שימושית ביותר בפיתוח אלגוריתמים. השימוש בתבנית זו נעשה בכל פעם שיש לצבור סדרת ערכים בצורה כלשהי, למשל על ידי סכום או על ידי מכפלה.

נתבונן בשני האלגוריתמים, הפותרים את שתי הבעיות שלעיל :

<p>1. <math>\text{sum} \leftarrow 20</math></p> <p>2. קלוט משקל משקולת <math>\text{weight}</math> ו <math>\text{sum} \leftarrow \text{weight} + \text{sum}</math></p> <p>continuation</p> <p>3. <math>\text{sum} \leftarrow \text{sum} + \text{weight}</math> continuation <math>\leq 15</math></p> <p>3.1. <math>\text{sum} \leftarrow \text{sum} + \text{weight}</math></p> <p>3.2. קלוט משקל משקולת <math>\text{weight}</math> ו <math>\text{sum} \leftarrow \text{weight} + \text{sum}</math></p> <p>המשך continuation-2</p> <p>4. הצג כפולט את ערכו של <math>\text{sum}</math></p>	<p>1. <math>\text{sum} \leftarrow 0</math></p> <p>2. <math>\text{sum} \leftarrow 15</math> עזמים:</p> <p>2.1. קלוט מספר ממשי <math>\text{num}</math></p> <p>2.2. אם <math>\text{num} &gt; 0</math></p> <p>2.2.1. <math>\text{sum} \leftarrow \text{sum} + \text{num}</math></p> <p>3. הצג כפולט את ערכו של <math>\text{sum}</math></p>
---	--

בשני האלגוריתמים ערכו של הצובר (sum) מאותחל לפני הלולאה לערך תחילי, ולהבדיל ממונה, ערך זה אינו בהכרח 0. בפתרון לבעיה 3 ערכו מאותחל ב-0 ובפתרון לבעיה 4 ערכו מאותחל ב-20. בגוף הלולאה משתנה ערכו של הצובר בערך כלשהו, השונה מ-1 (שימו לב כי ייתכן שערכו של הצובר עלול לקטון במהלך הצבירה, למשל, במקרה של צבירת ערכים שליליים).

נציג כעת את מאפייני שתי התבניות. ראשית, נציג את מאפייני התבנית **מנייה** ואחר כך נציג את מאפייני התבנית **צבירה**. עבור כל אחת מהתבניות נראה אלגוריתם עבור ביצוע חוזר באורך הידוע מראש וכן אלגוריתם לביצוע חוזר בתנאי.

### שם התבנית: מנייה

**נקודת מוצא:** אורך סדרת נתוני הקלט limit, סדרת ערכי הקלט, תנאי מנייה condition

**מטרה:** מנייה של ערכי הקלט המקיימים את התנאי condition, מתוך סדרת קלט שאורכה limit

### אלגוריתם:

1. אגף אף count 0-2

2. בצע limit פעמים:

2.1 קלט ערך element-2

2.2 אם element מקיים אף condition

2.2.1 הצף אף count 1-2

### יישום ב-C#:

```
count = 0;
for (i = 1; i <= limit; i++)
{
    element = int.Parse(Console.ReadLine());
    if (condition)
    {
        count++;
    }
}
```



### שם התבנית: מנייה

**נקודת מוצא:** תנאי סיום conditionToEnd, ערכי הקלט, תנאי מנייה conditionToCount

**מטרה:** מנייה של ערכי הקלט המקיימים את התנאי conditionToCount. משך ביצוע המנייה תלוי בתנאי conditionToEnd.

### אלגוריתם:

1. אגף אף count 0-2

2. קלט ערך 2-element

3. כן עזב לא מקיים conditionToEnd **33ע:**

3.1 אף element מקיים אף conditionToCount

3.1.1 הפזל אף count 1-2

3.2 קלט ערך 2-element

### יישום ב-C#:

```
count = 0;
element = int.Parse(Console.ReadLine());
while (!conditionToEnd)
{
    if (conditionToCount)
    {
        count++;
    }
    element = int.Parse(Console.ReadLine());
}
```

ישנן שתי צורות בסיסיות של צבירה : צבירת סכום וצבירת מכפלה. אנו נראה תחילה את התבניות של צבירת סכום שמשך ביצועה ידוע מראש וצבירת סכום שמשך ביצועה תלוי בתנאי.

#### שם התבנית: צבירת סכום

**נקודת מוצא:** אורך סדרת נתוני הקלט limit, ערך תחילי של הצובר initial, ערכי הקלט, תנאי צבירה condition

**מטרה:** צבירת הסכום של הערך initial ושל סכום ערכי הקלט המקיימים את התנאי condition, מתוך סדרת קלט שאורכה limit

#### אלגוריתם:

1. אגור sum initial-

2. עבור limit פעמים:

2.1 קוטף ערך element-

2.2 אם element מקיים את condition

2.2.1 הוסיף ל-sum את ערכו element

#### יישום ב-C#:

```
sum = initial;
for (i = 1; i <= limit; i++)
{
    element = int.Parse(Console.ReadLine());
    if (condition)
    {
        sum += element;
    }
}
```

### שם התבנית: צבירת סכום

**נקודת מוצא:** תנאי סיום conditionToEnd, ערך התחלתי של הצובר initial, ערכי הקלט, תנאי צבירה conditionToSum

**מטרה:** צבירת סכום של הערך initial ושל ערכי הקלט המקיימים את התנאי conditionToSum. משך ביצוע הצבירה תלוי בתנאי conditionToEnd.

### אלגוריתם:

1. אגור initial-2 sum

2. קולט ערך-2 element

3. כל עוד לא מקיים conditionToEnd **3.3:**

3.1 אם element מקיים conditionToSum

3.1.1 הוסף ל-sum את ערכו element

3.2 קולט ערך-2 element

### יישום ב-C#:

```
sum = initial;
element = int.Parse(Console.ReadLine());
while (!conditionToEnd)
{
    if (conditionToSum)
    {
        sum += element;
    }
    element = int.Parse(Console.ReadLine());
}
```

עתה נראה את התבנית של צבירת מכפלה שמשך ביצועה ידוע מראש ושל צבירת מכפלה שמשך ביצועה תלוי בתנאי.

#### שם התבנית: צבירת מכפלה

**נקודת מוצא:** אורך סדרת נתוני הקלט limit, ערך התחלתי של הצובר initial, ערכי הקלט, תנאי צבירה condition

**מטרה:** צבירת מכפלה של הערך initial ושל ערכי הקלט המקיימים את התנאי condition, מתוך סדרת קלט שאורכה limit

#### אלגוריתם:

1. אגור initial-2 mult

2. בצע limit פעמים:

2.1 קוטף ערך element-2

2.2 אם element מקיים את condition

2.2.1 הכפל את mult element-2 והשם את המכפלה mult-2

#### יישום ב-C#:

```
mult = initial;
for (i = 1; i <= limit; i++)
{
    element = int.Parse(Console.ReadLine());
    if (condition)
    {
        mult *= element;
    }
}
```

### שם התבנית: צבירת מכפלה

**נקודת מוצא:** תנאי סיום conditionToEnd, ערך התחלתי של הצובר initial, ערכי הקלט, תנאי צבירה conditionToMult

**מטרה:** צבירת מכפלה של הערך initial ושל ערכי הקלט המקיימים את התנאי conditionToMult. משך ביצוע הצבירה תלוי בתנאי conditionToEnd.

### אלגוריתם:

1. אגור initial-2 mult

2. קלוט ערך element-2

3. כל עוד לא מקיים conditionToEnd **בצע**:

3.1 אס element מקיים את conditionToMult

3.1.1 הכפול את mult element-2 והשם את המכפלה mult-2

3.2 קלוט ערך element-2

### יישום ב-C#:

```
mult = initial;
element = int.Parse(Console.ReadLine());
while (!conditionToEnd)
{
    if (conditionToMult)
    {
        mult *= element;
    }
    element = int.Parse(Console.ReadLine());
}
```

**שימו ♥:** ערכו של initial חייב להיות שונה מ-0, שהרי מכפלת 0 בכל מספר תשאיר את ערכו של הצובר 0. ערכו של הצובר יכול להיות שלילי, וערכו של הצובר עשוי לקטון אם ערכו של element גדול מ-0 וקטן מ-1 (כלומר, הוא שבר), או אם נצברים גם ערכים שליליים.

**ועוד שימו ♥:** הבעיה האלגוריתמית אינה חייבת לכלול תנאים למנייה ולצבירה (condition, conditionToSum ו-conditionToMult). ייתכנו מקרים, כפי שאכן ראינו בבעיות האלגוריתמיות 2 ו-4, שהמנייה והצבירה מתבצעות ללא תנאי.

### שאלה 1

א. ישמו את פתרונותיהן של ארבע הבעיות האלגוריתמיות 1-4 בשפת C#.

- ב. בהתייחס לבעיה 1: שנו את התוכנית כך שתציג כפלט את מספר הילדים שאינם רשאים לעלות למתקן "רכבת הרים".
- ג. בהתייחס לבעיה 2: שנו את התוכנית כך שתציג כפלט את מספר הכניסות לאתר ביממה רק עבור המשתמשים שהקוד שלהם בין 100 ל-200.
- ד. בהתייחס לבעיה 3: הרחיבו את התוכנית כך שתציג כפלט גם את סכום המספרים השליליים.
- ה. בהתייחס לבעיה 4: הרחיבו את התוכנית כך שתציג כפלט גם את מספר המשקולות הכולל שהמתחרה הצליח להרים.

## שאלה 2

נתונה סדרת הקלט הבאה: 0 15 -6 14 -3 2

א. מה יוצג כפלט עבור כל אחד מהשימושים השונים בתבניות:

1. מנה את האיברים האי-חיוביים בסדרת הקלט שאורכה 6 והצג כפלט את הערך

שהקבל

2. מנה את האיברים הזוגיים בסדרת הקלט המסתיימת בזקוף 0 והצג כפלט את הערך

שהקבל

3. חשב את הסכום המצטבר של האיברים המתחלקים ב-3 בסדרת הקלט שאורכה 6

והצג כפלט את הערך שהקבל

4. חשב את הסכום המצטבר של 7 עם כל האיברים בסדרת הקלט המסתיימת בזקוף 0

והצג כפלט את הערך שהקבל

5. חשב את המכפלה המצטברת של האיברים האי-זוגיים בסדרת הקלט המסתיימת

בזקוף 5 והצג כפלט את הערך שהקבל

6. חשב את המכפלה המצטברת של 4 בכל האיברים בסדרת הקלט המסתיימת בזקוף

15 והצג כפלט את הערך שהקבל

ב. ישמו כל אחד מן השימושים בשפת C#.

## שאלה 3

א. בכיתה בת 41 תלמידים קיימו בחירות לנציג מועצת תלמידים. שני תלמידים הציגו את מועמדותם וכל תלמיד בכיתה הצביע עבור אחד המועמדים. נתון אלגוריתם שהקלט שלו הוא סדרה באורך 41 של המספרים 1 ו-2 המייצגים את מספרי המועמדים, והפלט שלו הוא מספרו של המועמד שזכה ברוב קולות.

1. אגף אגף count1-2-0

2. אגף אגף count2-2-0

3. כצט 41 פסמים:

3.1 קואט מספר 2 מועמד candidate-

3.2. אס candidate שווה ל-1

3.2.1. הגדל אס count1-1

3.3. אגרא

3.3.1. הגדל אס count2-1

4. אס count1 > count2

4.1. הגדל כפלט "מועמד מספר 1 זכה ברוב קולות"

5. אגרא

5.1. הגדל כפלט "מועמד מספר 2 זכה ברוב קולות"

באלגוריתם הנתון ישנו שילוב בשימוש של שני מונים : ציינו מהם ומהו תפקידם של אחד מהם.

ב. נתון אלגוריתם חלקי, השקול לאלגוריתם הנתון, אך משתמש במונה אחד. השלימו :

1. מנה את האיברים ה \_\_\_\_\_ בסדרת הקלט שאורכה 41/שס count-2 אס

הערך המקביל

2. אס \_\_\_\_\_

2.1. הגדל כפלט "מועמד מספר 1 זכה ברוב קולות"

3. אגרא

3.1. הגדל כפלט "מועמד מספר 2 זכה ברוב קולות"

#### שאלה 4

א. בסופרמרקט השכונתי בודקים מדי יום את הפדיון היומי ואת מספר הקונים. פתחו אלגוריתם שהקלט שלו הוא סדרת מספרים המייצגת את תשלומי הקונים, שבסיומה הזקף 1-, והפלט שלו הוא מספר הקונים באותו יום והודעה המציינת אם ביום זה סך הפדיון מקניות בסכומים הגבוהים מ-500 ש"ח עלה על סך הפדיון מקניות בסכומים שאינם גבוהים מ-500 ש"ח. ישמו את האלגוריתם בשפת C#.

ב. ציינו באילו תבניות השתמשתם בסעיף א וכיצד **שילבתם** ביניהן.

#### שאלה 5

תת-סדרה תחילית של המספרים הטבעיים היא סדרה מהצורה  $1, 2, 3, \dots, n$ . בניסיון לבחון את "כוחה של המכפלה לעומת הסכום" הטילה המורה על תלמידיה לסכם את ערכיהן של תת-סדרות תחיליות של המספרים הטבעיים וכן להכפיל את ערכיהן.

א. פתחו אלגוריתם שאינו מקבל קלט, והפלט שלו הוא האורך הקטן ביותר של תת-סדרה תחילית של המספרים הטבעיים, שעבורה ערכה של המכפלה המצטברת יהיה לפחות פי 100 מערכו של הסכום המצטבר. ישמו את האלגוריתם בשפת C#.

- ב. בפתרון בעיה זו יש שימוש בצובר מכפלה. מהו ערכו התחילי של צובר זה? הסבירו.
- ג. ציינו באילו תבניות נוספות השתמשתם וכיצד **שילבתם** ביניהן.

## שאלה 6

לקראת בחינות בגרות בית הספר מעוניין לתת שיעורי תגבור ל-75 תלמידים משתי כיתות שונות. שיעורי התגבור יכולים להתקיים לאחר הלימודים, בימים ראשון ושלישי. כדי להיערך הוחלט להעביר שאלון, שבעזרתו ניתן יהיה לקבוע כמה תלמידים רוצים שיעורי תגבור ומהו היום המועדף על רובם. כל תלמיד רשם בשאלון את מספר כיתתו (1 או 2) ואת מספר היום שאותו הוא מעדיף (1 או 3). תלמיד שאינו מעוניין בתגבור רשם את מספר כיתתו ואת המספר 0.

כתבו אלגוריתם, שהקלט שלו הוא הערכים שרשמו 75 התלמידים, והפלט שלו כולל את:

1. מספר התלמידים המעוניינים בשיעורי תגבור.
  2. היום בשבוע שמרבית התלמידים מעדיפים.
  3. מספר החדרים הדרושים לשיעורי התגבור (בכל חדר ילמדו 15 תלמידים לכל היותר).
  4. מספר הכיתה שבה נרשמו פחות תלמידים לשיעורי התגבור.
- א. ישמו את האלגוריתם בשפת C#.

- ב. ציינו באילו תבניות השתמשתם וכיצד **שילבתם** ביניהן.

## שאלה 7

א. פתחו אלגוריתם אשר מקבל כקלט סדרת גילאים של זוגות נשואים (גיל הבעל וגיל האישה). סוף הקלט יצוין על ידי זוג הערכים 0 0. הפלט יהיה אחוז הזוגות, שבהם גיל האישה גבוה מגילו של הבעל. ישמו את האלגוריתם בשפת C#.

- ב. ציינו באילו תבניות השתמשתם וכיצד **שילבתם** ביניהן.

## שאלה 8

א. פתחו אלגוריתם אשר מקבל כקלט רשימת מספרים שלמים תלת-ספרתיים המסתיימת במספר שאינו תלת-ספרתי, והפלט שלו הוא כמות המספרים שספרת העשרות שלהם זוגית או שסכומן של ספרות המספר הוא אי-זוגי. ישמו את האלגוריתם בשפת C#.

- ב. ציינו באילו תבניות השתמשתם וכיצד **שילבתם** ביניהן.



## ממוצע של סדרת מספרים

נתבונן בבעיה האלגוריתמית הבאה :

כתבו אלגוריתם שהקלט שלו כולל 31 מספרים המייצגים את הטמפרטורה בכל אחד מימי חודש יולי, והפלט שלו הוא ממוצע הטמפרטורות בחודש זה.

לפתרון הבעיה עלינו לחשב ממוצע של ערכים מספריים. בפרק 3 היכרנו את התבנית **ממוצע של סדרת מספרים**, עבור סדרה בת שני מספרים. עתה נרחיב את התבנית עבור סדרת מספרים באורך כלשהו. כדי לחשב ממוצע של סדרת מספרים יש לחשב תחילה את הסכום הכולל של איברי הסדרה ולאחר מכן לחלקו במספר הערכים בסדרה. התבנית של ממוצע, כפי שמשמשת בפתרון בעיה זו, דומה לחישוב הממוצע שהוצג בפתרון בעיה 3 בפרק הלימוד.

נתבונן בשני האלגוריתמים הבאים, הראשון לפתרון הבעיה שלעיל, והשני הוא האלגוריתם שניתן בפרק 7 כפתרון לבעיה 3 :

1.  $\text{avg} = \text{sum } a[0..30] / 31$

2. **כיצד**  $\text{avg}$  **נמצא**?

2.1. קואוט טמפרטורה של יום באוגוסט יולי ב-temperature

2.2.  $\text{sum}$  האוסף את ערכי של temperature

3.  $\text{avg}$  האוסף את ערכי של  $\text{sum}/31$

4.  $\text{avg}$  כפול את ערכי של average

1.  $\text{avg} = \text{sum } a[0..30] / 31$

2.  $\text{avg}$  counterLarge ב-0

3. קואוט ערך גיוני של length

4. **כיצד** length **נמצא**?

4.1. קואוט ערך ממשי ב-num

4.2. האוסף את ערכי של num האוסף את ערכי של sum

4.3.  $\text{avg}$  של num גדול מ-50

4.3.1.  $\text{avg}$  ב-1 את ערכי של counterLarge

5.  $\text{avg}$  ממחיש את ערכי של  $\text{sum}/\text{length}$  והשם ב-average

6.  $\text{avg}$  כפול את ערכי של average

7.  $\text{avg}$  כפול את ערכי של counterLarge

בשני האלגוריתמים אותחל ערכו של sum ב-0 ואז חושב הסכום המצטבר בתוך sum. בתום הביצוע החוזר מתבצעת חלוקה של sum במספר הערכים שסוכם. עבור הבעיה הנתונה מספר הערכים שסוכם הוא 31, כמספר הימים בחודש יולי. בפתרון בעיה 3 שהוצגה בפרק הלימוד מספר הערכים המסוכמים הוא Length.

נציג את מאפייני התבנית לחישוב ממוצע עבור ביצוע חוזר התלוי בתנאי. שימו לב כי האלגוריתם של התבנית משלב במקביל תבנית צבירה (כדי לסכום את הערכים) ותבנית מנייה (כדי לדעת כמה ערכים נצברו).

#### שם התבנית: ממוצע של סדרת מספרים

**נקודת מוצא:** תנאי סיום conditionToEnd, ערכי הקלט, תנאי conditionToInclude

**מטרה:** חישוב ממוצע של ערכי הקלט המקיימים את התנאי conditionToInclude. ביצוע החישוב תלוי בתנאי conditionToEnd.

#### אלגוריתם:

1. אגף אל count 0-2
2. אגף אל sum 0-2
3. קלט ערך element-2
4. כל עוד לא מקיים conditionToEnd 32:
  - 4.1. אק element מקיים אל conditionToInclude
    - 4.1.1. הגדל אל count 1-2
    - 4.1.2. הוסף לערך אל sum element
  - 4.2. קלט ערך element-2
  5. השם אל average 2-אל ערך אל היטוי הגשני / count / sum

#### יישום ב-C#:

```
count = 0;
sum = 0;
element = int.Parse(Console.ReadLine());
while (!conditionToEnd)
{
    if (conditionToInclude)
    {
        count++;
        sum += element;
    }
    element = int.Parse(Console.ReadLine());
}
average = (double) sum / count;
```

בדומה לדרך בה נהגנו עבור תבניות **מנייה** ו-**צבירה** ניתן גם כאן להציג תת-תבנית המתאימה לביצוע חוזר שאורכו ידוע מראש. במקרה של ביצוע חוזר שאורכו ידוע מראש (והוא שווה ל-limit) אין צורך במניית הערכים. ולכן האלגוריתם הוא פשוט יותר:

1. חשב את הסכום המצטבר של איברי סדרת הקלט שאורכה limit

2. חלק את הסכום המצטבר של limit-2

**שימו ♥**: עבור התבנית של ממוצע של סדרת מספרים יש להניח שבקלט יש לפחות ערך אחד המקיים את התנאי, וזאת כדי להימנע מחלוקה ב-0.

**ועוד שימו ♥**: בדומה לתבניות **מנייה** ו-**צבירה**, גם בתבנית **ממוצע** ניתן להשתמש ללא תנאי (conditionToInclude), כלומר לחשב את ממוצע כל האיברים בסדרה.

---

## שאלה 9

א. יישמו את האלגוריתם לפתרון הבעיה של הטמפרטורה הממוצעת בחודש יולי בשפת C#.   
 ב. שנו את התוכנית שכתבתם בסעיף א כך שתחשב את הטמפרטורה הממוצעת רק עבור הימים שבהם הטמפרטורה הייתה מעל 30 מעלות צלסיוס.   
 ג. מעוניינים להשוות בין הטמפרטורה הממוצעת בחודש יולי בשנה זו לבין הטמפרטורה הממוצעת בחודש יולי בשנה הקודמת, ולהציג כפלט הודעה המציינת באיזו שנה מבין השתיים הטמפרטורה הממוצעת הייתה גבוהה יותר. הקלט כולל קודם כל את הערכים הנתונים עבור חודש יולי בשנה זו, ואחר כך את הנתונים עבור חודש יולי בשנה הקודמת. השלימו את האלגוריתם החלקי:

1. חשב את הממוצע של סדרת ערכי הקלט שאורכה \_\_\_\_\_ והשם הוא C-2-

averageCurrentYear

2. חשב את הממוצע של סדרת ערכי הקלט שאורכה \_\_\_\_\_ והשם הוא C-2-

averageLastYear

3. חשב \_\_\_\_\_

3.1 חשב כפלט \_\_\_\_\_

4. חשב את \_\_\_\_\_

4.1 חשב כפלט \_\_\_\_\_

5. חשב \_\_\_\_\_

5.1 חשב כפלט \_\_\_\_\_

## שאלה 10

באוניברסיטת "בית הסטודנט" הוחלט כי בקורס אשר בו ממוצע ציוני הבחינה של הסטודנטים הוא מתחת ל-55 ייערך מבחן חוזר. נתון אלגוריתם חלקי שהקלט שלו הוא סדרת ציוני הסטודנטים בקורס מסוים, רשימה המסתיימת בזקיף 1-, והפלט שלו הוא הודעה האם יש לקיים מבחן חוזר.

1. \_\_\_\_\_

2. \_\_\_\_\_

3. קאוס ערך 2-mark

4. כול ערך 3-mark שונה מ-1-1-32:

4.1. \_\_\_\_\_

4.2. קאוס ערך 2-mark

5. השם 2-average אט ערכו של הביטוי השלילי \_\_\_\_\_

6. אס \_\_\_\_\_

6.1. \_\_\_\_\_

א. השלימו את האלגוריתם.

ב. ניתן לכתוב את האלגוריתם בעזרת תבנית. השלימו:

1. שם אט \_\_\_\_\_ סדרת ערכי הקלט המסתיימת על ידי הזקיף 1-.

2. אס \_\_\_\_\_

2.1. \_\_\_\_\_

ג. ישמו את האלגוריתם בשפת C#.

## שאלה 11

בשירות המטאורולוגי מחשבים בכל שנה את ממוצע המשקעים החודשי.

א. כתבו אלגוריתם שהקלט שלו הוא 12 מספרים המייצגים את כמות המשקעים בכל אחד מחודשי השנה, והפלט שלו הוא ממוצע המשקעים לחודש.

ב. הרחיבו את האלגוריתם כך שיציג כפלט גם את ממוצע ששת החודשים הראשונים בשנה ואת ממוצע ששת החודשים האחרונים בשנה.

ג. ישמו את האלגוריתם בשפת C#.

## שאלה 12

במפעל מסוים התבקש מנהל מחלקת משאבי אנוש לבדוק האם גילן הממוצע של הנשים במפעל נמוך מגילם הממוצע של הגברים במפעל.

א. כתבו אלגוריתם שהקלט שלו הוא 100 זוגות נתונים המייצגים את מין העובד ('m' – גבר, 'f' – אישה) ושנת הלידה של העובד, והפלט שלו הוא הודעה מתאימה של מנהל מחלקת משאבי האנוש לאחר הבדיקה.

ב. הרחיבו את האלגוריתם שכתבתם בסעיף א כך שיציג כפלט את מספר העובדים הכולל שאמור לצאת לגמלאות ב-5 השנים הקרובות (הניחו כי גברים יוצאים לפנסיה בגיל 67 ונשים בגיל 62).

ג. ישמו את האלגוריתם בשפת C#.

ד. ציינו באילו תבניות השתמשתם וכיצד **שילבתם** ביניהן.

---

## מציאת מקסימום או מינימום בסדרה

נתבונן בבעיה האלגוריתמית הבאה :

למכרז זורמות הצעות כספיות שונות. ההצעה הכספית הגבוהה ביותר היא זו הזוכה במכרז. כתבו אלגוריתם שהקלט שלו הוא סדרה המסתיימת בזקיף 0, של מספרים המייצגים את ההצעות הכספיות, והפלט שלו הוא סכומה של ההצעה הזוכה. הניחו שבקלט יש לפחות הצעה אחת וכן הניחו שקיימת הצעה יחידה שזוכה.

לפתרון הבעיה עלינו לחשב את הערך המקסימלי מבין סדרת הערכים המספריים הנקלטים. בפרק 5 הראינו את התבנית של **מציאת מקסימום ומינימום בסדרה** בת שניים או שלושה ערכים. עתה נרחיב את התבנית עבור סדרה באורך כלשהו. התבנית של **מציאת מקסימום ומינימום בסדרה**, כפי שמשמשת בפתרון בעיה זו, דומה לתבנית **מציאת מקסימום ומינימום בסדרה** שהוצגה בבעיה 5 בפרק הלימוד.

נתבונן בשני האלגוריתמים הבאים, הראשון לפתרון הבעיה שלעיל, והשני הוא האלגוריתם שניתן בפרק 7 לפתרון בעיה 5 :

1. קאוט הצגת מניי-2 max

2. קאוט הצגת מניי-2 price

3. כן עדיף  $price \neq 0$  כצג:

3.1. אס  $price > max$

3.1.1. השם  $max$ -2 אס עדיף  $price$

3.2. קאוט הצגת מניי-2 price

4. הצג כפוט אס עדיף  $max$

---

1. קאוט אס מספר השבועות-2 howMany

2. קאוט אס הסכום הראשון-2 balance

3. השם אס עדיף  $balance$ -2 max

4. כצג howMany-1 פסמים:

4.1. קאוט אס הסכום הכא-2 balance

4.2. אס  $max < balance$

4.2.1. השם אס עדיף  $balance$ -2 max

5. הצג כפוט אס עדיף  $max$

בשתי הבעיות נדרשנו למצוא את הערך הגדול ביותר בסדרת ערכים. בשלב ראשון, קבענו באופן שרירותי כי הערך של המשתנה הראשון הוא המקסימלי ולכן נשמר ערכו במשתנה max. לאחר מכן, נבדקו על פי סדר קליטתם ערכיהם של שאר נתוני הקלט. עבור כל נתון קלט שערכו גדול יותר מהמקסימום הנוכחי, הוחלף ערכו של max בערכו של נתון הקלט. באופן דומה, עם שינויים קלים, עובד האלגוריתם עבור מציאת הערך הקטן ביותר בסדרה.

נפריד את מאפייני התבנית **מציאת מקסימום ומינימום בסדרה** לשתי תת-תבניות: ראשית, נציג את מאפייני התבנית **מציאת מקסימום בסדרה** ואחר כך נציג את מאפייני התבנית **מציאת מינימום בסדרה**. עבור התבנית **מציאת מקסימום** נתייחס לביצוע חוזר באורך הידוע מראש ואילו עבור התבנית **מציאת מינימום** נתייחס לביצוע חוזר התלוי בתנאי. באופן דומה ניתן, עבור כל אחת מהתבניות, לטפל במקרה שאליו לא התייחסנו.

**שם התבנית:** מציאת מקסימום בסדרה

**נקודת מוצא:** אורך סדרת נתוני הקלט limit, ערכי הקלט

**מטרה:** מציאת הערך הגדול ביותר בסדרת ערכי הקלט שאורכה הוא limit

**אלגוריתם:**

1. קאוט ערך 2-max

2. ע32 limit - 1 עמלים:

2.1. קאוט ערך 2-element

2.2. אס element > max

2.2.1. הלס 2-max אס הערך element

**יישום ב-C#:**

```
max = int.Parse(Console.ReadLine());
for (i = 2; i <= limit; i++)
{
    element = int.Parse(Console.ReadLine());
    if (element > max)
    {
        max = element;
    }
}
```

**שם התבנית:** מציאת מינימום בסדרה

**נקודת מוצא:** תנאי סיום conditionToEndMinSearch, ערכי הקלט

**מטרה:** מציאת הערך הקטן ביותר מבין ערכי הקלט, עד אשר מתקיים התנאי conditionToEndMinSearch

**אלגוריתם:**

1. קאוט ערך min-2

2. קאוט ערך element-2

3. כל עוד לא מתקיים conditionToEndMinSearch כצעד:

3.1. אם element < min

3.1.1. השם min-2 את הערך element

3.2. קאוט ערך element-2

**יישום ב-C#:**

```
min = int.Parse(Console.ReadLine());
element = int.Parse(Console.ReadLine());
while (!conditionToEndMinSearch)
{
    if (element < min)
    {
        min = element;
    }

    element = int.Parse(Console.ReadLine());
}
```

**שימו ♥:** לפני ההוראה לביצוע חוזר כבר נקראים שני ערכים מהקלט. אנו נניח שהתנאי לסיום (conditionToEndMinSearch) יכול להתקיים עבור האיבר השני ואילך. לכן, למשל, אם תנאי הסיום מבטא קריאת זקיף, הרי שסדרת הקלט מכילה לפחות איבר אחד.

**ועוד שימו ♥:** אם בקלט יש שני ערכים או יותר השווים למקסימום (או למינימום) אז האלגוריתמים עבור התבניות **מציאת מקסימום** ו-**מציאת מינימום** מוצאים את המופע הראשון של המקסימום (או המינימום) בנתוני הקלט.



### שאלה 13

א. השלימו את השימוש בתבנית עבור בעיית המכרז:

מצא מקסימום בסדרת ערכי הקלט \_\_\_\_\_ והצג את ערכו כקלט

ב. ישמו את האלגוריתם של המכרז בשפת C#.

ג. הרחיבו את האלגוריתם כך שיוצג כפלט גם סכום ההצעה השנייה בגודלה. ישמו את האלגוריתם המתקבל בשפת C#.

### שאלה 14

מדענים העוסקים בתחום המטאורולוגיה החליטו לגלות מהי הטמפרטורה הנמוכה ביותר ומהי הטמפרטורה הגבוהה ביותר בצהרי היום בחודש נובמבר באזור הקוטב הצפוני, בו הטמפרטורה תמיד מתחת ל-0 מעלות צלסיוס. לצורך כך, המדענים מדדו את הטמפרטורה במשך 30 יום במהלך חודש נובמבר, מדי יום ביומו, בשעה 12:00 בצהריים.

נתון אלגוריתם שגוי, שהקלט שלו הוא 30 ערכי הטמפרטורות והפלט שלו הוא הערך הנמוך ביותר והערך הגדול ביותר:

1. אגף `maxTemperature` - 0

2. אגף `minTemperature` - 0

3. כצע 30 פעמים:

3.1 קאוט ערך טמפרטורה יומי - `temperature`

3.2 אם `temperature > maxTemperature`

3.2.1 השם - `maxTemperature` אגף `temperature`

3.3 אגף אם `temperature < minTemperature`

3.3.1 השם - `minTemperature` אגף `temperature`

4. הצג כקלט "הטמפרטורה הנמוכה ביותר באזור הקוטב הצפוני" היא

`minTemperature`

5. הצג כקלט "הטמפרטורה הגבוהה ביותר באזור הקוטב הצפוני" היא

`maxTemperature`

א. הסבירו במלים מדוע האלגוריתם שגוי.

ב. תקנו את האלגוריתם.

### שאלה 15

קבוצת אנשים מוגדרת כ"הומוגנית" אם טווח הגילאים של חבריה אינו עולה על 5 שנים, אחרת הקבוצה מוגדרת כ"הטרוגנית".

א. כתבו אלגוריתם שהקלט שלו הוא מספר האנשים בקבוצה,  $n$ , ולאחר מכן סדרה של  $n$  מספרים המייצגים את גילאי החברים בקבוצה. הפלט שלו הוא הודעה האם הקבוצה "הומוגנית" או "הטרוגנית".

ב. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם וכיצד **שילבתם** ביניהן.

ג. ישמו את האלגוריתם בשפת C#.

---

## מציאת ערך נלווה למקסימום או מינימום בסדרה

היכרנו את התבנית של מציאת מקסימום או מינימום בסדרה אולם לעיתים אנו מעוניינים לאו דווקא בערכים של המקסימום והמינימום אלא, למשל, במיקומם היחסי בקלט. דוגמה למקרה כזה הוא מציאת היום בחודש ינואר בו ירדה כמות משקעים מקסימלית.

נתבונן בבעיה האלגוריתמית הבאה :

נרחיב את בעיית המכרז שהוצגה בסעיף הדן בתבנית **מציאת מקסימום ומינימום בסדרה** כך שהקלט הוא סדרה של זוגות נתונים, כאשר כל זוג מכיל את הקוד של מגיש ההצעה (מספר שלם) ואת סכום ההצעה הכספית. הסדרה מסתיימת עם קליטת הזקוף 0 כאיבר שני בזוג. הפלט של האלגוריתם הוא הקוד של מגיש ההצעה הזוכה .

לפתרון הבעיה עלינו לחשב את ההצעה המקסימלית מבין סדרת ההצעות הכספיות הנתונות בקלט, אבל אנו נדרשים בנוסף לשמור גם את קוד מגיש ההצעה. התבנית של **מציאת ערך נלווה למקסימום ולמינימום בסדרה**, כפי שמשמשת בפתרון בעיה זו, דומה לתבנית **מציאת ערך נלווה למקסימום ולמינימום בסדרה** שהוצגה בבעיה 6 בפרק הלימוד.

נתבונן בשני האלגוריתמים הבאים, האחד לפתרון הבעיה שלעיל, והשני הוא האלגוריתם שניתן בפרק 7 לפתרון בעיה 6 :

1. קאוט קוד מגיש הצעה 2-winCode

2. קאוט הצעג מחיר 2-max

3. קאוט קוד נוסף של מגיש הצעה 2-code

4. קאוט הצעג מחיר נוספת 2-price

5. כול ערך  $price \neq 0$  הצעג:

5.1. אס  $price > max$

5.1.1. השג 2-max אס ערכו של price

5.1.2. השג 2-winCode אס ערכו של code

5.2. קאוט קוד מגיש הצעה 2-code

5.3. קאוט הצעג מחיר 2-price

6. הצג כקאוט אס ערכו של winCode

1. קאוט אט אורק השיי הראשון ב- currentSongLength
2. השם אט ערכו של longest- currentSongLength
3. השם אט ערכו של shortest- currentSongLength
4. השם- placeLongest אט הערך 1
5. השם- placeShortest אט הערך 1
6. **ב3ע 99 עממים :**

- 6.1. קאוט אט המספר הכא ב- currentSongLength
- 6.2. **אם**  $\text{currentSongLength} > \text{longest}$ 
  - 6.2.1. השם אט longest- currentSongLength
  - 6.2.2. השם אט מקומו של השיי הנכחי ב- placeLongest
- 6.3. **אם**  $\text{currentSongLength} < \text{shortest}$ 
  - 6.3.1. השם אט shortest- currentSongLength
  - 6.3.2. השם אט מקומו של השיי הנכחי ב- placeShortest
7. **ה3ג כפוט אט ערכו של placeLongest / אט ערכו של placeShortest**

בשתי הבעיות נדרשנו למצוא ערך נלווה לערך הגדול ביותר בסדרת ערכים. בבעיה הנתונה בתחילת הסעיף הערך הנלווה הוא קוד מגיש ההצעה ובבעיה 6 בפרק הלימוד הערך הנלווה הוא מקומו של השיר הארוך ביותר.

כדי למצוא את הערך הנלווה לערך הגדול ביותר בסדרת ערכים נתבסס על התבנית למציאת מקסימום: בכל פעם שנעדכן את ערכו של max (וכמובן גם באתחול) נשמור במשתנה נוסף את ערכו של הערך הנלווה ל-max. באופן דומה, עם שינויים קלים, מתבצע האלגוריתם עבור מציאת הערך הנלווה לערך הקטן ביותר בסדרה.

בתבנית של מציאת ערך נלווה למקסימום ולמינימום בסדרה נדגים כערך הנלווה את מקום המקסימום או המינימום, אבל כאמור, ערך נלווה יכול להיות כל ערך שהוא, כמו הקוד הנלווה להצעה, בבעיה שלעיל.

נפריד את מאפייני התבנית **מציאת ערך נלווה למקסימום ולמינימום בסדרה** לשתי תת-תבניות: ראשית, נציג את מאפייני התבנית **מציאת ערך נלווה למקסימום בסדרה** ואחר כך נציג את מאפייני התבנית **מציאת ערך נלווה למינימום בסדרה**. עבור התת-תבנית הראשונה נתייחס לביצוע חוזר באורך הידוע מראש, ואילו עבור השנייה נתייחס לביצוע חוזר התלוי בתנאי. משום הדמיון בין שתי התת-תבניות, קל לפתח אלגוריתם מתאים למקרה האחר, עבור כל אחת מהן.

**שם התבנית:** מציאת ערך נלווה למקסימום בסדרה

**נקודת מוצא:** אורך סדרת נתוני הקלט limit, ערכי הקלט

**מטרה:** מציאת מיקום הערך הגדול ביותר בסדרת ערכי הקלט שאורכה הוא limit

**אלגוריתם:**

1. קאט ערך max-2

2. אגף אל placeOfMax ב-1

3. בצע limit-1 פעמים:

3.1 קאט ערך element-2

3.2 אם element > max

3.2.1 השם max-2 אל הערך element

3.2.2 השם placeOfMax אל מקומו של element בקאט

**יישום ב-C#:**

```
max = int.Parse(Console.ReadLine());
placeOfMax = 1;
for (i = 2; i <= limit; i++)
{
    element = int.Parse(Console.ReadLine());
    if (element > max)
    {
        max = element;
        placeOfMax = i;
    }
}
```

**שם התבנית:** מציאת ערך נלווה למינימום בסדרה

**נקודת מוצא:** תנאי סיום conditionToEnd, ערכי הקלט

**מטרה:** מציאת מיקום הערך הקטן ביותר מבין ערכי הקלט, עד אשר מתקיים התנאי

conditionToEnd

**אלגוריתם:**

1. קאוט ערך min-2

2. אגוא placeOfMin-1

3. קאוט ערך element-2

4. אגוא currentPlace-2

5. כול עוצר לא מתקיים conditionToEnd כ3ע:

5.1. אס element < min

5.1.1. השם min-2 אג הערך של element

5.1.2. השם placeOfMin-2 אג הערך של currentPlace

5.2. העבא אג currentPlace-1

5.3. קאוט ערך element-2

**יישום ב-C#:**

```
min = int.Parse(Console.ReadLine());
placeOfMin = 1;
element = int.Parse(Console.ReadLine());
currentPlace = 2;
while (!conditionToEnd)
{
    if (element < min)
    {
        min = element;
        placeOfMin = currentPlace;
    }
    currentPlace++;
    element = int.Parse(Console.ReadLine());
}
```

**שימו ♥ :** לפני ההוראה לביצוע חוזר כבר נקראים שני ערכים מהקלט. אנו נניח שהתנאי לסיום (conditionToEnd) יכול להתקיים עבור האיבר השני ואילך. לכן, למשל, אם תנאי הסיום מבטא קריאת זקיף, הרי שסדרת הקלט מכילה לפחות איבר אחד.

לא פעם מופיע הערך המינימלי או המקסימלי בסדרה יותר מפעם אחת. בתבנית המוצגת מחושב מקום המופע הראשון של הערך המקסימלי או המינימלי. שינוי מזערי באלגוריתם יאפשר את חישוב מקום המופע האחרון. שאלה 16 מתייחסת לנקודה זאת. עם ההתקדמות בחומר הלימוד נתייחס גם לסוגיית מציאת כל המופעים של הערך המקסימלי או המינימלי.

---

### שאלה 16

א. ישמו את האלגוריתם המורחב של המכרז בשפת C#.

ב. עתה, הניחו שבקלט יש שתי הצעות זוכות, כלומר שהערך הגבוה ביותר של הצעה הוצע על ידי שני מגישים. שנו את האלגוריתם, ואחר כך את התוכנית שכתבתם בסעיף א, כך שיוצג כפלט הקוד של מגיש ההצעה השנייה (במקום של הראשונה).

### שאלה 17

חנוכייה אופיינית היא חנוכייה בת 9 קנים, שאחד מהם גבוה מן השאר ומשמש כשמש. חנוכייה כזו נחשבת "סימטרית" אם השמש ממוקם במרכז החנוכייה (ארבעה קנים ממוקמים מימין של השמש וארבעה ממוקמים משמאלו). חנוכייה כזו נחשבת "צידית" אם השמש ממוקם בקצה אחד שלה (שאר הקנים ממוקמים משמאלו של השמש, או מימין). בכל מקרה אחר, נחשבת החנוכייה "מיוחדת".

א. כתבו אלגוריתם שהקלט שלו הוא 9 מספרים המייצגים את גבהי הקנים של חנוכייה אופיינית, ונתונים לפי סדר מיקומם (משמאל לימין), והפלט שלו הוא הודעה המציינת את סוג החנוכייה.

ב. ישמו את האלגוריתם בשפת C#.

### שאלה 18

בשכבת כיתות י' נערך מבחן משווה במדעי המחשב. המבחן נבדק על ידי המורים, וציוני התלמידים נרשמו בטופס ריכוז בו כל תלמיד מיוצג על ידי מספר סידורי. כל מורה התבקש למסור למרכז המגמה את הציון הגבוה ביותר בכיתה ומספרו הסידורי של התלמיד שקיבל ציון זה, וכן את הציון הנמוך ביותר בכיתה ומספרו הסידורי של התלמיד שקיבל ציון זה (המורה ידוע על תלמיד אחד מכל קטגוריה, גם אם יש יותר מתלמיד אחד שקיבל את הציון הגבוה ביותר בכיתה או יותר מתלמיד אחד שקיבל את הציון הנמוך ביותר בכיתה).

א. כתבו אלגוריתם שהקלט שלו הוא מספר התלמידים n בכיתה מסוימת, ואחריו סדרה של n ציוני התלמידים בכיתה, והפלט שלו הוא הציון הגבוה ביותר בכיתה ומספרו הסידורי של התלמיד שקיבל ציון זה, והציון הנמוך ביותר בכיתה ומספרו הסידורי של התלמיד שקיבל ציון זה.

ב. הרחיבו את האלגוריתם כך שיציג כפלט גם את הציון השני בגודלו.

ג. ישמו את האלגוריתם בשפת C#.

## איסוף בקיזוז

נתבונן בבעיה האלגוריתמית הבאה :

באוניברסיטה נערך סקר לבדיקת ההשערה: "מספר הסטודנטיות הלומדות קורסים המשלבים מחשב קטן יותר ממספר הסטודנטים בקורסים אלה". כתבו אלגוריתם שהקלט שלו הוא סדרה באורך 100 שאיבריה הם התו 'f' או התו 'm', המייצגת את מינם ('m' - סטודנט, 'f' - סטודנטית) של 100 סטודנטים הלומדים קורסים המשלבים מחשב. הפלט הוא הודעה האם ההשערה נכונה או אינה נכונה ביחס לנתונים.

בפתרון הבעיה אין חשיבות למספר הסטודנטים ומספר הסטודנטיות הלומדים קורסים המשלבים מחשב. לכן אין צורך במנייה רגילה. במקרה זה מתאים יותר למנות תוך כדי קיזוז. כלומר, נגדיר מונה יחיד, שערכו יגדל בכל פעם שנקלוט סטודנט, וערכו יקטן בכל פעם שנקלוט סטודנטית. בסיום התהליך יהיה במונה ההפרש בין מספר הסטודנטים למספר הסטודנטיות. נבדוק את ערכו של המונה: אם ערכו חיובי ההשערה נכונה. אחרת, ההשערה אינה נכונה. גם עבור שאלה 7.27 בפרק הלימוד ניתן לכתוב אלגוריתם המתבסס על איסוף בקיזוז.

נתבונן בשני האלגוריתמים הבאים, האחד לפתרון הבעיה שלעיל, והשני לפתרון שאלה 7.27:

1. אגאל אג 0-2 count	1. אגאל אג 0-2 count
2. קאוט קאו בואי 2-vote	2. כצצ 100 קצמ'ים:
3. כ' ע'ז #' < vote כצצ:	2.1. קאוט מין הסטודנט 2-gender
3.1. אק vote שוה ל-'A'	2.2. אק gender שוה ל-'m'
3.1.1. אג הצצ 1-2 count	2.2.1. אג הצצ 1-2 count
3.2. אגא	2.3. אגא
3.2.1. אקטן אג 1-2 count	2.3.1. אקטן אג 1-2 count
3.3. קאוט קאו בואי 2-vote	3. אק count צצו מ-0
4. אק count צצו מ-0	3.1. הצצ כפוט "ההשערה נכונה"
4.1. הצצ כפוט "אזון זכה"	4. אגא
5. אגא	4.1. הצצ כפוט "ההשערה לא נכונה"
4.2. הצצ כפוט "אזון לא זכה"	

התבנית של **איסוף בקיזוז** מתאימה גם לקיזוז באמצעות צבירה ולא רק לקיזוז באמצעות מנייה. בהמשך נראה שאלות, שבהן יש צורך לאסוף בקיזוז באמצעות צבירה.



נפריד את מאפייני התבנית **איסוף בקיזוז** לשתי תת-תבניות: ראשית, נציג את מאפייני התבנית **איסוף בקיזוז באמצעות מנייה** ואחר כך נציג את מאפייני התבנית **איסוף בקיזוז באמצעות צבירה**. עבור התבנית הראשונה נתייחס לביצוע חוזר באורך הידוע מראש ואילו עבור השנייה נתייחס לביצוע חוזר התלוי בתנאי. כבמקרים קודמים, עבור כל אחת מהתבניות ניתן לפתח אלגוריתמים מתאימים גם למקרה האחר.

#### שם התבנית: איסוף בקיזוז באמצעות מנייה

**נקודת מוצא:**אורך סדרת נתוני הקלט limit, ערכי הקלט, תנאי לקיזוז conditionToDecrease

**מטרה:** איסוף בקיזוז באמצעות מנייה של ערכי הקלט מתוך סדרה שאורכה limit. הקיזוז מתבצע על פי התנאי conditionToDecrease

#### אלגוריתם:

1. אגור count 0-2

2. בצע limit פעמים:

2.1 קלט ערך element-2

2.2 אם element לא מקיים את conditionToDecrease

2.2.1 הגדל את count ב-1

2.3 אגור

2.3.1 הקטן את count ב-1

#### יישום ב-C#:

```
count = 0;
for (i = 1; i <= limit; i++)
{
    element = int.Parse(Console.ReadLine());
    if (!conditionToDecrease)
    {
        count++;
    }
    else
    {
        count--;
    }
}
```

**שימו ♥ :** בתבנית **איסוף בקיזוז** התנאי condition הוא הכרחי, ותפקידו, שליטה בקיזוז, שונה מתפקיד התנאים conditionToCount, conditionToSum, או conditionToMult בתבניות **מנייה ו-צבירה**.

עתה נראה את התבנית של איסוף בקיזוז באמצעות צבירה, עבור שמשך ביצועה תלוי בתנאי.

**שם התבנית:** איסוף בקיזוז באמצעות צבירה

**נקודת מוצא:** תנאי סיום conditionToEnd, ערכי הקלט, ערך צבירה התחלתי initial, תנאי לקיזוז conditionToDecrease

**מטרה:** איסוף בקיזוז באמצעות צבירה של ערכי הקלט ושל הערך initial, משך הביצוע תלוי בתנאי conditionToEnd, והקיזוז מתבצע על פי התנאי conditionToDecrease

**אלגוריתם:**

1. אגור initial-2 sum

2. קאוט ערך-2 element

3. כל עוד לא מקיים conditionToEnd **כ3ע:**

3.1. אם element לא מקיים את conditionToDecrease

3.1.1. הוסף-1 sum את ערכו של element

3.2. אגור

3.2.1. הפגה-1 sum את ערכו של element

3.3. קאוט ערך-2 element

**יישום ב-C#:**

```
sum = initial;
element = int.Parse(Console.ReadLine());
while (!conditionToEnd)
{
    if (!conditionToDecrease)
    {
        sum += element;
    }
    else
    {
        sum -= element;
    }
    element = int.Parse(Console.ReadLine());
}
```

## שאלה 19

ישמו את הפתרון לבעיית הסקר באוניברסיטה בשפת C#.

## שאלה 20

נתונה סדרת ההוראות הבאה, שיש בה שימוש בתבנית **איסוף בקיזוז** :

1. **אסוף בקיזוז** באמצעות מנייה את איברי סדרת הקלט שאורכה 8, תוך קיזוז

האיברים האי-זוגיים

2. **הצג כפולט** את הערך האחרון

א. תנו דוגמה לסדרת נתוני קלט של מספרים חיוביים שעבורה יוצג כפלט הערך 3.

ב. הסבירו בקצרה מהי מטרת ההוראה.

ג. השלימו את סדרת ההוראות הבאה, השקולה לסדרה הנתונה, אך משתמשת בתנאי לקיזוז אחר :

1. **אסוף בקיזוז** באמצעות מנייה את איברי סדרת הקלט שאורכה 8, תוך קיזוז

האיברים הזוגיים

2. **הצג כפולט** את \_\_\_\_\_

## שאלה 21

נתון אלגוריתם שהקלט שלו הוא סדרת מספרים ממשיים :

1. **אגף** את sum ב-0

2. **קאוט מספר** מ-**num**

3. **כא עזר**  $num \neq 0$  **עזר** :

3.1 **אם**  $num > 0$

3.1.1 **הוסף**  $num$  **ל**  $sum$  **אם עזר** **ל**  $num$

3.2 **אחר**

3.2.1 **הפג**  $sum$  **אם עזר** **ל**  $num$

3.3 **קאוט מספר** מ- $num$

4. **הצג כפולט** את ערכו של  $sum$

א. תנו שתי דוגמאות קלט שונות לסדרות נתוני קלט של מספרים ממשיים, שעבורן יוצג כפלט הערך 0.

ב. הסבירו בקצרה מהי מטרת האלגוריתם.

ג. כתבו הוראה השקולה לאלגוריתם תוך שימוש בתבנית. השלימו :

**אסוף בקיזוז** באמצעות צבירה ל- \_\_\_\_\_ של איברי סדרת הקלט המסתיימת

בזקיף 0, תוך קיזוז \_\_\_\_\_

## שאלה 22

א. כתבו אלגוריתם שהקלט שלו הוא יתרת לקוח של בנק בתחילת החודש, ואחריו סדרה של מספרים, המסתיימת ב-0. כל מספר מייצג את סכום הפעולה: סכום חיובי מציין הפקדת הסכום בחשבון הבנק וסכום שלילי מציין משיכת הסכום מחשבון הבנק. הפלט של האלגוריתם הוא היתרה של הלקוח בסוף החודש.

ב. ישמו את האלגוריתם בשפת C#.

## שאלה 23 (שאלה זו מתאימה לאחר לימוד סעיף 7.7 – קינון הוראות לביצוע חוזר)

מבחן במתכונת של שאלון אמריקני הועבר ל-30 סטודנטים של הקורס "דיוק בתשובות". במבחן 20 שאלות. ניקוד תשובות התלמידים התבצע באופן הבא: כל תשובה נכונה זיכתה ב-5 נקודות וכל תשובה שגויה גרמה להפחתה של 3 נקודות (בכל מקרה, לא ניתן לקבל ציון נמוך מ-0).

א. כתבו אלגוריתם, שהקלט שלו הוא 20 התשובות הנכונות ואחריהן תשובות של 30 הסטודנטים בקורס (20 תשובות לכל סטודנט). הפלט הוא מספר הסטודנטים שציונם "עובר" (לפחות 55).

ב. הרחיבו את האלגוריתם שכתבתם בסעיף א כך שיציג כפלט גם את אחוז התלמידים שקיבלו 0 במבחן.

ג. הרחיבו את האלגוריתם שכתבתם בסעיף ב כך שיציג כפלט גם את הציון הגבוה ביותר בבחינה ואת הציון הנמוך ביותר בבחינה (שאינו 0).

ד. ציינו מהן התבניות המשמשות לפתרון וכיצד שילבתם ביניהן.

ה. ישמו את האלגוריתם שכתבתם בסעיף ג בשפת C#.

## שאלה 24 (\*)

נתונים ארבעה ביטויים בוליאניים המשתמשים בתבניות, הבודקים אותו תנאי, עבור סדרת קלט נתונה:

1. *השך האקבל מ-איסוף בקיזוז באמצעות צבירה של איברי סדרת הקלט שאורכה 10, תוך קיזוז האיברים הזוגיים הלא זוגי*
2. *השך האקבל מ-מנייה של האיברים האי-זוגיים בסדרת הקלט שאורכה 10 הלא זוגי*
3. *השך האקבל מ-צבירת סכום איברי סדרת הקלט שאורכה 10 הלא זוגי*
4. *השך האקבל מ-צבירת סכום האיברים האי-זוגיים בסדרת הקלט שאורכה 10 הלא זוגי*

א. תנו דוגמה לסדרת מספרים שלמים חיוביים שעבורה הערך של כל אחד מהביטויים הבוליאניים יהיה true.

ב. נסחו במלים את התנאי שמבטאים הביטויים הבוליאניים.

ג. ישמו כל אחד מהשימושים השונים כקטעי תוכניות בשפת C#.

## פירוק מספר חיובי לספרותיו

בפרק 4 הכרנו את התבנית של פירוק מספר דו-ספרתי חיובי לספרותיו. עתה נרחיב את התבנית עבור מספר שלם חיובי באורך כלשהו. פירוק המספר מתבסס על רעיון של הפרדת ספרת האחדות מהמספר, כך שהמספר שנוותר קטן פי 10, וחוזר חלילה עד שלא נותרות ספרות במספר.

נראה את מאפייני התבנית **פירוק מספר חיובי לספרותיו**:

**שם התבנית:** פירוק מספר חיובי לספרותיו

**נקודת מוצא:** מספר שלם חיובי num

**מטרה:** הצגה כפלט של ספרותיו של num

**אלגוריתם:**

1. כל עוד num שונה מ-0 כצג

2. הצג כפלט את ספרת האחדות של num

3. הקטן את num פי 10

**יישום ב-C#:**

```
while (num != 0)
{
    Console.WriteLine(num % 10);
    num /= 10;
}
```

**שימו ♥:** באלגוריתם של התבנית הצגנו כפלט את ספרותיו של num, אולם ניתן כמובן לבצע פעולות אחרות על ספרות המספר, כגון: מנייה, צבירה, ועוד. השאלות הבאות מתייחסות לבעיות אלגוריתמיות בהן נדרשות פעולות אחרות על ספרות המספר.

### שאלה 25

- פתחו אלגוריתם שהקלט שלו הוא מספר שלם חיובי, והפלט שלו הוא מספר הספרות במספר. ישמו את האלגוריתם בשפת C#.
- הרחיבו את האלגוריתם שכתבתם בסעיף א כך שיציג כפלט גם את סכום הספרות האי-זוגיות במספר.
- הרחיבו את האלגוריתם שכתבתם בסעיף ב כך שיציג כפלט גם הודעה המציינת אם יש במספר יותר ספרות המתחלקות ב-3 ללא שארית מאשר ספרות שאינן מתחלקות ב-3. ישמו את האלגוריתם המלא בשפת C#.
- ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם המלא וכיצד **שילבתם** ביניהן.

## שאלה 26

לפניכם קטע תוכנית בשפת C#, שהקלט שלו הוא מספר שלם חיובי, והפלט שלו אמור להיות מספר הספרות במספר. קטע התוכנית שגוי.

```
num = int.Parse(Console.ReadLine());  
sum = num % 10;  
while ((num / 10) > 0)  
    sum += num % 10;  
Console.WriteLine(sum);
```

- הביאו דוגמת קלט שעבורה יתקבל הפלט הדרוש.
- הביאו דוגמת קלט שעבורה לא יתקבל הפלט הדרוש.
- הסבירו במלים מדוע קטע התוכנית שגוי.
- תקנו את קטע התוכנית כך שישגי את מטרתו עבור כל קלט חוקי.

## שאלה 27

א. כתבו אלגוריתם שהקלט שלו הוא מספר שלם וחיובי num ומספר חד-ספרתי place המייצג מקום. הפלט שלו הוא הספרה הנמצאת במקום place במספר num (מיקום הספרות מתחיל מימין). אם אין במספר place ספרות יוצג כפלט הערך 1-.

למשל, עבור הקלט 2 17489 יוצג כפלט הערך 8, ועבור הקלט 6 17489 יוצג כפלט הערך 1-.

- ישמו את האלגוריתם בשפת C#.

## שאלה 28

ספרת שורש של מספר היא ספרה בין 1 ל-9 המתקבלת מתהליך של חיבור חוזר של ספרות המספר עד אשר מתקבל מספר חד-ספרתי. למשל, ספרת השורש של המספר 30486 היא 2, כיוון שסכום ספרות המספר המקורי הוא 20 וסכום הספרות של 20 הוא 2.

א. כתבו אלגוריתם שהקלט שלו הוא מספר שלם חיובי והפלט שלו הוא ספרת השורש של המספר הנתון.

- ישמו את האלגוריתם בשפת C#.

## שאלה 29

א. כתבו אלגוריתם שהקלט שלו הוא מספר שלם חיובי והפלט שלו הוא ההפרש בין הספרה הגדולה ביותר במספר לבין הספרה הקטנה ביותר במספר.

- ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם וכיצד **שילבתם** ביניהן.

- ישמו את האלגוריתם בשפת C#.

## בניית מספר

בפרק 4 הכרנו את התבנית של בניית מספר כאשר התמקדנו בבניית מספר דו-ספרתי משתי ספרות. עתה נרחיב את התבנית לבניית מספר שלם באורך כלשהו מספרות הנקלטות בזו אחר זו. כדי להרכיב מספר מספרות צריך בכל שלב כלולאה להגדיל את המספר פי 10 ולחבר לו את ספרת הקלט החדשה. כך עד לסיום הקלט.

ניתן להרחיב את התבנית, למשל, על ידי בניית מספר ממספרים דו-ספרתיים, ואז בכל שלב יגדל המספר פי 100 ויתווסף לו המספר הדו-ספרתי התורן מהקלט.

נציג עבור התבנית **בניית מספר** אלגוריתם עבור ביצוע חוזר באורך הידוע מראש. ניתן להתאימו גם למקרים בהם סיום הבנייה תלוי בתנאי.

**שם התבנית:** בניית מספר

**נקודת מוצא:** אורך סדרת נתוני הקלט limit, ספרות הקלט

**מטרה:** בניית מספר מספרות הקלט

**אלגוריתם:**

1. אגף num 0-2

2. כצע limit פעמים:

2.1 קלט ספרה digit-2

2.2 השם num-2 את הערך של הביטוי  $num * 10 + digit$

**יישום ב-C#:**

```
num = 0;
for (i = 1; i <= limit; i++)
{
    digit = int.Parse(Console.ReadLine());
    num = num * 10 + digit;
}
```

### שאלה 30

נתון האלגוריתם הבא:

1. קלט ספרה digit-2

2. קלט מספר limit-2

3. אגף num 0-2

4. כצע limit פעמים:

4.1. השם ב-num אל הערך של הביטוי  $num * 10 + digit$

4.2. הערך כפול אל num

א. מה יהיה הפלט עבור הקלט 3 5?

ב. תנו דוגמה לקלט שעבורו הפלט יהיה 6.

ג. הסבירו בקצרה מהי מטרת האלגוריתם.

### שאלה 31

א. כתבו אלגוריתם שהקלט שלו הוא מספר שלם וחיובי num, ספרה digit ומקום place. הפלט שלו הוא המספר שמתקבל מהחלפת הספרה הנמצאת במקום place (מימין) במספר num בספרה digit.

למשל, עבור הקלט 4 6 78342 יוצג כפלט הערך 76342, ועבור הקלט 1 9 13608 יוצג כפלט הערך 13608.

ב. ישמו את האלגוריתם בשפת C#.

### שאלה 32

א. כתבו אלגוריתם שהקלט שלו הוא מספר שלם וחיובי num והפלט שלו הוא המספר המתקבל מ-num על ידי היפוך סדר ספרותיו.

ב. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם וכיצד שילבתם ביניהן.

ג. ישמו את האלגוריתם בשפת C#.



## האם כל הערכים בסדרה מקיימים תנאי?

נתבונן בבעיה האלגוריתמית הבאה:

במסגרת הפעילות של עידוד "הקורא הצעיר" הוחלט להעניק פרס לבית-ספר שבו כל תלמידיו קראו לפחות ספר קריאה אחד במהלך חופשת הקיץ. פתחו אלגוריתם שהקלט שלו הוא מספר הספרים שקרא כל אחד מתלמידי בית-ספר "אמירים" שבסיומה הזקיף 1- (עבור תלמיד שלא קרא כלל ספרים ייקלט הערך 0), והפלט שלו הוא הודעה המציינת אם הפרס יוענק לבית-הספר.

לפתרון הבעיה עלינו לעבור על כל נתוני הקלט, כלומר, לבדוק עבור כל תלמיד האם קרא ספרי קריאה במהלך החופשה או לא. אבל, אם קיים לפחות תלמיד אחד שלא קרא ספר קריאה אז לבית-הספר לא יוענק הפרס ולכן אין טעם להמשיך ולבדוק את מספר ספרי הקריאה שקראו שאר תלמידי בית-הספר. אלגוריתם זה הוא תיאור של התבנית **האם כל הערכים בסדרה מקיימים**

**תנאי?** גם עבור שאלה 7.53 בפרק הלימוד ניתן לכתוב אלגוריתם המתבסס על תבנית זו.

נתבונן בשני האלגוריתמים הבאים, הראשון לפתרון הבעיה שלעיל, והשני לפתרון שאלה 7.53:

1. אגף אג allReaders ב-true

2. קאוס מספר ספרי קריאה אלמאיד ב-books

3. כן ע/כ 1- books ≠ /אם ערכו של allReaders הוא true ב-3ע:

3.1. אם ערכו של books הוא 0

3.1.1. השם ב-allReaders אג הערך false

3.2. אגף

3.2.1. קאוס מספר ספרי קריאה אלמאיד ב-books

4. אם ערכו של allReaders הוא true

4.1. הצג כפאוס "לביא-הספר אמירים מושנק הפרס"

1. אגף אג allEven ב-true

2. אגף אג sum ב-0

3. אגף אג howMany ב-1

4. כן ע/כ 20 ≤ howMany /אם ערכו של allEven הוא true ב-3ע:

4.1. קאוס מספר גיובי ב-num

4.2. הצג אג ערכו של howMany ב-1

4.3. אם num מספר אי-זוגי

4.3.1. השם ב-allEven אג הערך false

4.4. אגף

4.4.1. הצג אג sum ב-num

5. **אם** allEven

5.1. **הצג כפולט את ערכו של** sum

6. **אגרג**

6.1. **הצג כפולט "הקולט אינו גוקי"**

משמעות התבנית **האם כל הערכים בסדרה מקיימים תנאי?** היא בדיקה של קיום תנאי עבור כל ערכי הסדרה. מתבצעת סריקה של ערכי הקלט בזה אחר זה, ואם אחד הערכים אינו מקיים את התנאי אין טעם בהמשך הסריקה. בבעיה הנתונה אם ערכו של books שווה ל-0 אז נמצא תלמיד שלא קרא אף לא ספר קריאה אחד ולכן אין צורך להמשיך בסריקה. בשאלה 7.53 אם בקלט יש מספר אי-זוגי אז אין טעם להמשיך בבדיקת שאר ערכי המספרים.

נציג את מאפייני התבנית **האם כל הערכים בסדרה מקיימים תנאי?** עבור סדרה שאורכה ידוע מראש. ניתן להתאים את האלגוריתם למקרה שבו אורך הסדרה אינו ידוע מראש, בדומה לאלגוריתם שניתן לבעיה שלעיל.

**שם התבנית:** האם כל הערכים בסדרה מקיימים תנאי?

**נקודת מוצא:** אורך סדרת נתוני הקלט limit, ערכי הקלט, תנאי condition

**מטרה:** קביעת הערך true אם כל הערכים בסדרה מקיימים את התנאי condition וקביעת הערך false אם קיים ערך אחד בסדרה שאינו מקיים את התנאי

**אלגוריתם:**

1. **אגרג** all **כ-** true

2. **אגרג** howmany **כ-** 1

3. **כל עוד**  $\text{howmany} \leq \text{limit}$  **אם** all **הוא** true **כ-33:**

3.1. **קולט ערך כ-** element

3.2. **הגדל** howMany **אם** ערכו של **כ-** 1

3.3. **אם** element **אינו מקיים את** condition

3.3.1. **השם כ-** all **אם** הערך false

**יישום ב-C#:**

```
all = true;
howmany = 1;
while ((howmany <= limit) && (all))
{
    element = int.Parse(Console.ReadLine());
    howmany++;
    if (condition) // התנאי
    {
        all = false;
    }
}
```

התבנית **האם כל הערכים בסדרה מקיימים תנאי?** מחשבת ערך בוליאני שערכו true או false. אנו משתמשים במשתנה בוליאני all שערכו התחילי הוא true, כלומר, ההנחה **התחילית** היא שכל ערכי הסדרה אכן מקיימים את התנאי. אם במהלך הסריקה אחד הערכים אינו מקיים את התנאי אז ההנחה התחילית שלנו מתבדה ולכן ערכו של all מקבל ערך false ו הסריקה מסתיימת.

---

### שאלה 33

א. רשמו הוראה השקולה לאלגוריתם של "הקורא הצעיר" תוך שימוש בתבנית החדשה. השלימו:  
**האם כל הערכים בסדרת הקלט המסתיימת ב-\_\_\_\_\_ מקיימים את התנאי'\_\_\_\_\_?**  
ב. ישמו את האלגוריתם בשפת C#.

### שאלה 34

א. כתבו אלגוריתם שהקלט שלו הוא מספר שלם וחיובי num והפלט שלו הוא ההודעה "ספרות זהות" אם כל ספרות המספר זהות וההודעה "ספרות שונות" אם לא כל הספרות זהות.  
ב. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם וכיצד **שילבתם** ביניהן.  
ג. ישמו את האלגוריתם בשפת C#.

### שאלה 35

לפניכם קטע תוכנית בשפת C#, שהקלט שלו הוא סדרה של מספרים שלמים וחיוביים. סוף הקלט מצוין על ידי המספר 1-. קטע התוכנית אמור להציג כפלט את ההודעה "כל המספרים הם כפולות של 6" אם 6 מחלק של כל המספרים בסדרת הקלט. קטע התוכנית שגוי.

```
num = int.Parse(Console.ReadLine());  
while (num != -1)  
{  
    ok = (num % 6 == 0)  
    num = int.Parse(Console.ReadLine());  
}  
if (ok)  
{  
    Console.WriteLine ("All numbers are multiples of 6");  
}
```

- א. תנו דוגמה לסדרת קלט (לפחות 5 ערכים) עבורה לא ניתן להבחין שקטע התוכנית שגוי.  
ב. תנו דוגמה לסדרת קלט (לפחות 5 ערכים) עבורה ניתן להבחין שקטע התוכנית שגוי.  
ג. הסבירו במילים מדוע קטע התוכנית שגוי.  
ד. תקנו את קטע התוכנית כך שיבצע את מטרתו עבור כל סדרת קלט.

## האם קיים ערך בסדרה המקיים תנאי?

נתבונן בשתי הבעיות האלגוריתמיות הבאות:

**בעיה 1:** כתבו אלגוריתם, שהקלט שלו הוא סדרה של 15 מספרים שלמים, והפלט שלו הוא הודעה המציינת אם קיים בסדרת הקלט מספר שלילי.

**בעיה 2:** כתבו אלגוריתם, שהקלט שלו הוא סדרה של תווים, המסתיימת בזקוף '\*', והפלט שלו הוא הודעה המציינת אם נקלטה בסדרת התווים אחת מאותיות ה-ABC או אחת מאותיות ה-abc.

לפתרון שתי הבעיות עלינו לעבור על נתוני הקלט עד למציאת ערך המקיים את התנאי. בבעיה 1 התנאי הוא האם ערך הקלט הוא מספר שלילי ובבעיה 2 התנאי הוא האם נקלטה אחת מאותיות ה-ABC או אחת מאותיות ה-abc. אם קיים ערך אחד המקיים את התנאי אז יש להפסיק את הסריקה כי ערך מתאים כבר נמצא. אלגוריתם זה הוא תיאור של התבנית **האם קיים ערך בסדרה המקיים תנאי?**

נתבונן בשני האלגוריתמים לפתרון בעיה 1 ובעיה 2 :

1. אגרו אג found-2 false
2. אגרו אג howmany-1
3. כול עוצב 15 howmany/גם ערכו של found שווה ל-1 false-2:
  - 3.1 קאוס מספר שלם-2 num
  - 3.2 הגדל אג ערכו של howmany-1
  - 3.3 אג  $num < 0$
  - 3.3.1 השם found-2 אג הערך true
4. אג found
  - 4.1 הגדל כפאוס "קיים מספר שלילי בסדרה הקאוס"
  5. אגרו
    - 5.1 הגדל כפאוס "לא קיים מספר שלילי בסדרה הקאוס"

1. אלא לא found-2 false  
2. קיוס לא ch-2  
3. כן ע'ז \*' ≠ ch זכר וכל על found חל false 23:

3.1. אס  $(ch \leq 'z' \wedge ch \geq 'a') / (ch \leq 'Z' \wedge ch \geq 'A')$

3.1.1. השם found-אם הערך true

3.2. אגרא

3.2.1. קאנט גא-כ

4. אס found

4.1. הצג כפאט "קיימא אלא-כ ABC אלא-כ abc בסדרה הקאט"

5. אגרא

5.1. הצג כפאט "לא קיימא אלא-כ ABC אלא-כ abc בסדרה הקאט"

נציג את מאפייני התבנית האם קיים ערך בסדרה המקיים תנאי? עבור סדרה שאורכה אינו ידוע מראש. ניתן להתאים את האלגוריתם למקרה שבו אורך הסדרה ידוע מראש, בדומה לאלגוריתם שניתן לבעיה 1 שלעיל.

**שם התבנית:** האם קיים ערך בסדרה המקיים תנאי?

**נקודת מוצא:** תנאי לסיום הסדרה toEnd, ערכי הקלט, תנאי condition

**מטרה:** קביעת הערך true אם קיים ערך בסדרה המקיים את התנאי condition וקביעת הערך

false אם כל הערכים בסדרה אינם מקיימים את התנאי

**אלגוריתם:**

1. אגרא אלא found-כ false

2. קאנט ערך-כ element

3. כלא עוצר הגאלי toEnd לא מקיים ואם ערכו של found הוא false כצג:

3.1. אס element מקיים אלא condition

3.1.1. השם found-אם הערך true

3.2. אגרא

3.2.1. קאנט ערך-כ element

**יישום ב-C#:**

```
found = false;
element = int.Parse(Console.ReadLine());
while (!toEnd && !found)
{
    if (condition) // element מקיים את התנאי
    {
        found = true;
    }
    element = int.Parse(Console.ReadLine());
}
```

משמעות התבנית **האם קיים ערך בסדרה המקיים תנאי?** היא בדיקה של קיום תנאי עבור ערכים בסדרה. מתבצעת סריקה של ערכי הקלט בזה אחר זה, ואם אחד הערכים אכן מקיים את התנאי אין טעם בהמשך הסריקה. התבנית מחשבת ערך בוליאני שערכו true או false. לכן, אנו משתמשים במשתנה בוליאני found שערכו התחילי הוא false. כלומר, ההנחה **התחילית** היא שאין ערך בסדרה המקיים את התנאי. אם במהלך הסריקה אחד הערכים אכן מקיים את התנאי אז ההנחה התחילית שלנו מתבדה, מאחר שמצאנו ערך המקיים את התנאי. לכן found יקבל ערך true והסריקה תסתיים.

## שאלה 36

א. כתבו הוראה השקולה לאלגוריתמים של כל אחת מהבעיות 1 ו-2 תוך שימוש בתבנית **האם קיים ערך בסדרה המקיים תנאי?** השלימו:

בעיה 1:

1. **אם קיים ערך בסדרת הקלט שאורכה \_\_\_\_\_ המקיים \_\_\_\_\_**

1.1 **הצג כפלט "קיים מספר שלילי בסדרת הקלט"**

2. **אגרה**

2.1 **הצג כפלט "לא קיים מספר שלילי בסדרת הקלט"**

בעיה 2:

1. **אם קיים ערך בסדרת הקלט שמסתיימת ב- \_\_\_\_\_ המקיים \_\_\_\_\_**

1.1 **הצג כפלט "קיימת א/ב-2 ABC א/ב-2 abc בסדרת הקלט"**

2. **אגרה**

2.1 **הצג כפלט "לא קיימת א/ב-2 ABC א/ב-2 abc בסדרת הקלט"**

ב. ישמו את האלגוריתמים לפתרון הבעיות 1 ו-2 בשפת C#.

ג. עבור כל אחד מהאלגוריתמים ניתן לכתוב אלגוריתם שקול תוך שימוש בתבנית **האם כל**

**הערכים בסדרה מקיימים תנאי?** השלימו את ההוראות הבאות עבור בעיה 1, וכתבו

הוראות מתאימות, המשתמשות בתבנית זו, לבעיה 2.

בעיה 1:

1. **אם לא כל הערכים בסדרה שאורכה \_\_\_\_\_ מקיימים את התנאי**

1.1 **הצג כפלט "קיים מספר שלילי בסדרת הקלט"**

2. **אגרה**

2.1 **הצג כפלט "לא קיים מספר שלילי בסדרת הקלט"**

### שאלה 37

א. כתבו אלגוריתם, שהקלט שלו מספר שלם חיובי והפלט שלו הוא הודעה המציינת אם קיימת במספר הספרה 0.

ב. שנו את האלגוריתם שכתבתם בסעיף א כך שיציג הודעה המציינת אם קיימת ספרה זוגית במספר. ציינו איזו תבנית **שילבתם** בפתרון הבעיה.

### שאלה 38

בכיתה י' המורה העלתה הצעה לעשות קומז'ץ לגיבוש 39 תלמידי הכיתה. ההצעה תתקבל רק אם אף תלמיד לא יתנגד להגיע לקומז'ץ. כל תלמיד התבקש לרשום על דף את האות 'y' אם הוא מסכים להצעה או את האות 'n' אם הוא מתנגד להצעה. יש לכתוב אלגוריתם, שהקלט שלו הוא סדרה של 39 תווים כאשר התו 'y' מייצג את הסכמת התלמיד להצעה והתו 'n' מייצג את התנגדות התלמיד להצעה. הפלט הוא הודעה המציינת אם ההצעה התקבלה או לא.

לפניכם שני אלגוריתמים לפתרון הבעיה:

אלגוריתם 1:

1. **אם קיים** ערך בסדרת הקלט שאורכה 39 המקיים את התנאי (answer שווה ל-

'n')

1.1 **הצג כפלט** את ההודעה "ההצעה לא התקבלה"

2. **אחר**

2.1 **הצג כפלט** את ההודעה "ההצעה התקבלה"

אלגוריתם 2:

1. **אם כל** הערכים בסדרת הקלט שאורכה 39 מקיימים את התנאי (answer שווה

ל- 'y')

1.1 **הצג כפלט** את ההודעה "ההצעה התקבלה"

2. **אחר**

2.1 **הצג כפלט** את ההודעה "ההצעה לא התקבלה"

ישמו את שני האלגוריתמים בשפת C#.

## מציאת כל הערכים בסדרה המקיימים תנאי

נתבונן בשתי הבעיות האלגוריתמיות הבאות:

**בעיה 1:** כתבו אלגוריתם, שהקלט שלו הוא סדרה של 25 זוגות מספרים שלמים, והפלט שלו הוא כל זוגות המספרים המקיימים את היחס של מספרים עוקבים.

**בעיה 2:** כתבו אלגוריתם, שהפלט שלו הוא כל המספרים מ-1 עד 100 המקיימים את הכללים של "7 בוס", כלומר: מתחלקים ב-7 ללא שארית או כוללים את הספרה 7.

לפתרון שתי הבעיות עלינו לעבור על כל הערכים בסדרה ולהציג כפלט את כל הערכים המקיימים את התנאי. בבעיה 1 התנאי הוא יחס של עוקב בין כל זוג מספרים ברשימה ובבעיה 2 התנאי הוא קיום הכללים של "7 בוס". יש דמיון מסוים בין תבנית זו לשתי התבניות האחרונות, אך במקרה זה איננו יכולים להפסיק את הסריקה לפני שנגיע אל סיום סדרת הקלט. אלגוריתם זה הוא תיאור של התבנית **מציאת כל הערכים בסדרה המקיימים תנאי**.

נתבונן בשני האלגוריתמים לפתרון בעיה 1 ובעיה 2:

1. כצג 25 פסגים:

2. קלט צג מספרים שלמים ב- $num1$  ו- $num2$

2.1. אם  $num1$  ו- $num2$  הם ערכים עוקבים  $num1$  ו- $num2$  הם ערכים

עוקבים

2.1.1. הצג כפלט את הערכים של  $num1$  ושל  $num2$

---

1. עבור כל מספר  $i$  גיבוי שלם הקטן מ-100 כצג:

1.1. אם 7 מחלק של  $i$   $i$  ספרת העשרות של  $i$  שווה ל-7  $i$  ספרת האחדות

של  $i$  שווה ל-7

1.1.1. הצג כפלט את הערך של  $i$

בתבנית **מציאת כל הערכים בסדרה המקיימים תנאי** מתבצעת סריקה של כל ערכי הקלט בזה אחר זה. בכל פעם שנמצא ערך שמקיים את התנאי מבצעים עליו פעולה כגון הצגה כפלט, מנייה, פעולה חשבונית.

נציג את מאפייני התבנית **מציאת כל הערכים בסדרה המקיימים תנאי** עבור סדרה שאורכה אינו ידוע מראש. ניתן להתאים את האלגוריתם למקרה שבו אורך הסדרה ידוע מראש, בדומה



לאלגוריתם שניתן לבעיה 1 ו-2 שלעיל. מאפייני התבנית מוגדרים לפי ביצוע פעולת קלט עבור כל איבר שנמצא מקיים את התנאי. ניתן להתאימם למקרה בו נדרשת פעולה אחרת, כפי שקורה בכמה מהשאלות הבאות.

**שם התבנית:** מציאת כל הערכים בסדרה המקיימים תנאי

**נקודת מוצא:** תנאי לסיום הסדרה toEnd, ערכי הקלט, תנאי condition

**מטרה:** הצגה כפלט של כל ערכי הקלט המקיימים את התנאי condition

**אלגוריתם:**

1. קאוט ערך 2-element

2. כן ע/ד הנאי toEnd לא מקיים 23ע:

3.1. אם element מקיים את condition

3.1.1. הצג כפלט את ערכו של element

3.2. קאוט ערך 2-element

**יישום ב-C#:**

```
element = int.Parse(Console.ReadLine());
while (!toEnd)
{
    if (condition)
    {
        Console.WriteLine(element);
    }
    element = int.Parse(Console.ReadLine());
}
```

### שאלה 39

א. רשמו הוראה השקולה לאלגוריתם לפתרון בעיה 1 תוך שימוש בתבנית. השלימו:  
מצא את כל הערכים בסדרת הקלט שאורכה \_\_\_\_\_ המקיימים את התנאי \_\_\_\_\_

ב. ישמו את האלגוריתם בשפת C#.

ג. רשמו הוראה השקולה לאלגוריתם לפתרון בעיה 2 תוך שימוש בתבנית. השלימו:  
מצא את כל הערכים בסדרת הקלט שאורכה \_\_\_\_\_ המקיימים את התנאי \_\_\_\_\_

ד. ישמו את האלגוריתם בשפת C#.

## שאלה 40

תלמידי שכבה י' בבית-ספר "היובל" השתתפו בתחרויות יום ספורט היתולי, במהלכו כל משתתף צבר נקודות. תוצאתו של תלמיד שלא השתתף היא 0. למען גיבוש הכיתות הוחלט להעניק "יום כיף" לכיתות שבהן השתתפו כל התלמידים בתחרויות של יום הספורט.

נתון אלגוריתם חלקי, שהקלט שלו הוא תוצאות התחרויות של כל אחת מ-10 הכיתות בבית-הספר. לכל כיתה נקלטת סדרת התוצאות של כל התלמידים עד לקליטת הזקיף 1-. הפלט של האלגוריתם הוא מספרי הכיתות שזכו ב"יום כיף".

1.  $i$  אינדיקס הקטן אל שווה ל-10:  $23$ :

1.1.  $prize$  אגל אגל -2-

1.2. קאול גאזאגאליז 2- result

1.3.  $3/8$   $1/8$   $23$ :

1.3.1.  $אס$

1.3.1.1.  $prize$  אגל  $העניק$  false

1.3.2.  $אגל$

1.3.2.1. \_\_\_\_\_

2.  $אס$

2.1.  $הצג כפאול$

א. השלימו את האלגוריתם.

ב. ציינו מהן התבניות המשולבות באלגוריתם.

ג. בהנהלת בית-הספר החליטו להעניק פרס נוסף "גיבושון" לכיתה שתלמידיה צברו את מירב הנקודות בתחרות. הרחיבו את האלגוריתם כך שיציג כפלט את מספר הכיתה שזכתה בפרס "גיבושון". איזו תבנית שילבתם עתה בפתרון הבעיה האלגוריתמית?

ד. ישמו את האלגוריתם בשפת C#.

## שאלה 41

א. כתבו אלגוריתם, שהקלט שלו הוא סדרה של מספרים חיוביים (סוף הסדרה מצוין ע"י מספר שלילי), והפלט שלו הוא כל המספרים המתחלקים בסכום ספרותיהם וכמות המספרים המקיימים תנאי זה.

לדוגמה: עבור סדרת הקלט 7- 18 103 83 110 550 4000: המספרים 18 550 110 ו- 4000 מתחלקים בסכום ספרותיהם, לכן הפלט הוא רשימת המספרים 18 550 110 4000 והודעה המציינת כי כמות המספרים המתחלקים בסכום ספרותיהם: 4

ב. ישמו את האלגוריתם בשפת C#.

## מעבר על זוגות סמוכים בסדרה

נתבונן בשתי הבעיות האלגוריתמיות הבאות:

**בעיה 1:** כתבו אלגוריתם, שהקלט שלו הוא סדרה של 18 מספרים שלמים, והפלט שלו הוא הודעה האם הסדרה מסודרת בסדר עולה ממש.

**בעיה 2:** כתבו אלגוריתם, שהקלט שלו הוא סדרה של מספרים שלמים חיוביים, המסתיימת עם קליטת הזקיף 0, והפלט שלו הוא סדרה שבה מכפלות כל זוגות המספרים הזוגיים הנקלטים זה אחר זה בסמיכות. לדוגמה, עבור הקלט: 0 2 4 8 9 6 5 3 תוצג כפלט הסדרה: 8 32 24, המתקבלת ממכפלת 6 ב-4, מכפלת 4 ב-8 ומכפלת 4 ב-2.

לפתרון שתי הבעיות יש לעבור על כל זוגות הערכים הסמוכים בסדרה. בבעיה 1 יש לבדוק עבור כל זוג ערכים סמוכים אם הוא מקיים את היחס של הערך הראשון (משמאל) קטן יותר מהערך השני. בבעיה 2 יש להציג כפלט את המכפלות של זוגות מספרים סמוכים שאיברייהם זוגיים. זה הוא, למעשה, תיאור של התבנית **מעבר על זוגות סמוכים בסדרה**.

נתבונן בשני האלגוריתמים הבאים לפתרון בעיות 1 ו-2:

1.  $\text{true} \leftarrow \text{ordered}$  **אגא**
2.  $\text{howmany} \leftarrow 1$  **אגא**
3.  $\text{beforeLast} \leftarrow \text{last}$  **קאוס מספרי שלם**
4.  $\text{howmany} < 18 \rightarrow \text{true} \leftarrow \text{ordered}$  **כא ע/ז**
- 4.1.  $\text{last} \leftarrow \text{last}$  **קאוס מספרי שלם**
- 4.2.  $\text{beforelast} \geq \text{last}$  **אס**
- 4.2.1.  $\text{false} \leftarrow \text{ordered}$  **השם אג הערך**
- 4.3. **אגא**
- 4.3.1.  $\text{beforeLast} \leftarrow \text{last}$  **השם אג הערך של**
5.  $\text{ordered}$  **אס**
- 5.1.  $\text{true} \leftarrow \text{howmany}$  **השם אג ההוצעה "הסדרה מסודרת בסדר עולה ממש"**
6. **אגא**
- 6.1.  $\text{true} \leftarrow \text{howmany}$  **השם אג ההוצעה "הסדרה אינה מסודרת בסדר עולה ממש"**

1.  $\text{beforeLast} \leftarrow \text{last}$  **קאוס מספרי שלם**
2.  $\text{last} \leftarrow \text{last}$  **קאוס מספרי שלם**
3.  $\text{last} < 0 \rightarrow \text{true}$  **כא ע/ז**

3.1. **אם** beforeLast הוא מספר זוגי / **אם** last הוא מספר זוגי

3.1.1. **הצג** כפלט את הערך של הביטוי  $beforeLast * last$

3.2. **השם** beforeLast-2 **הערך** של last

3.3. **קלט** מספר שלם beforeLast-2

בפתרון בעיות אלגוריתמיות רבות יש לבצע פעולות על זוגות ערכים הסמוכים בסדרה. עבור סדרת נתוני קלט באורך limit יש בסך הכול limit-1 זוגות ערכים סמוכים. למעשה ניתן להסתכל על כל זוג ערכים סמוכים כאל פריט אחד. הרעיון שבבסיסם של שני האלגוריתמים הוא שבכל שלב המשתנים beforeLast ו-last מכילים איברי זוג סמוך. המעבר לזוג הבא מתבצע כך: beforeLast מקבל את ערכו של last (כלומר, האיבר שקודם היה איבר שני בזוג הוא עכשיו איבר ראשון בזוג הסמוך) וב-last נקלט ערך נוסף. כך עד לסיום הקלט.

לשם המחשה נדגים בתבנית הצגה כפלט של סכום זוגות ערכים סמוכים בסדרה. ניתן כמובן לבצע פעולות אחרות על ערכי כל זוג, כגון מנייה, צבירה ועוד. בהמשך נדגים כמה מהאפשרויות השונות דרך שאלות.

נציג את מאפייני התבנית **מעבר על זוגות סמוכים בסדרה** עבור סדרה שאורכה ידוע מראש. אפשר לערוך התאמות למקרה בו אורך סדרת הקלט אינו ידוע מראש, כמו בבעיה 2 לעיל.

**שם התבנית:** מעבר על זוגות סמוכים בסדרה

**נקודת מוצא:** אורך סדרת נתוני הקלט limit, ערכי הקלט

**מטרה:** הצגה כפלט של סכומי כל זוגות הערכים הסמוכים בסדרת הקלט שאורכה limit

**אלגוריתם:**

1. **קלט** ערך beforeLast-2

2. **בצע** limit-1 פעמים

2.1. **קלט** ערך beforeLast-2

2.2. **הצג** כפלט את הערך של הביטוי  $beforeLast + last$

2.3. **השם** beforeLast-2 **הערך** של last

**יישום ב-C#:**

```
beforeLast = int.Parse(Console.ReadLine());  
for (i = 1; i <= limit-1; i++)  
{  
    last = int.Parse(Console.ReadLine());  
    Console.WriteLine(beforeLast + last);  
    beforeLast = last;  
}
```

**שימו ♥ :** אנו מניחים שבסדרת הקלט לפחות שני ערכים.

---

#### שאלה 42

ישמו את האלגוריתמים של שתי הבעיות הנתונות בשפת C# .

#### שאלה 43

- א. כתבו אלגוריתם, שהקלט שלו הוא 25 תווים והפלט שלו הוא כל הזוגות הסמוכים בסדרת הקלט שמכילים אותיות קטנות ב-abc עוקבות. ישמו את האלגוריתם בשפת C# .
- ב. ציינו באילו תבניות נוספות השתמשתם בכתיבת האלגוריתם וכיצד **שילבתם** ביניהן.

#### שאלה 44

- כתבו אלגוריתם שהקלט שלו הוא סדרת מספרים שלמים המסתיימת בזקיף 1-, והפלט הוא כל המספרים בקלט השווים לסכום שני המספרים הקודמים להם בסדרה (שני המספרים הראשונים לא יודפסו).
- למשל, עבור סדרת הקלט הבאה (משמאל לימין): 1- 89 6 5 13 0 13 11 2 9 4 יוצגו כפלט המספרים: 13 13 11.
- ישמו את האלגוריתם בשפת C# .

#### שאלה 45

- כתבו תוכנית בשפת C#, שהקלט שלה הוא מספר שלם וחיובי num והפלט שלה הוא הודעה האם ספרות המספר מסודרות בסדר עולה ממש.
- ציינו באילו תבניות נוספות השתמשתם בכתיבת התוכנית וכיצד **שילבתם** ביניהן.
-

## פרק 8 – יעילות של אלגוריתמים

אלגוריתמים נבחנים על פי מספר קני מידה. קנה המידה החשוב ביותר הוא נכונות. כלומר השגת המטרה (מתן הפלט הנכון) עבור כל קלט חוקי. קנה מידה נוסף הוא יעילות.

**יעילות של אלגוריתם** (תוכנית) נמדדת על פי "משאבי המחשב" הדרושים לביצוע האלגוריתם. משאבים אלה הם **גודל המקום** (בזיכרון) ו**הזמן** הדרוש לביצוע האלגוריתם.

**גודל המקום** נמדד בעיקר על פי מספר המשתנים של האלגוריתם.

**זמן-הביצוע** נקבע על פי מספר פעולות היסוד שיתבצעו במהלך הרצת האלגוריתם.

**פעולות היסוד** הן: פעולת קלט, פעולת פלט ופעולות חישוב.

למעשה, בצורה פשטנית נאמר שכל הוראה באלגוריתם כוללת פעולת יסוד אחת, ומכאן **מדידת זמן-הביצוע של אלגוריתם** נעשית על פי מספר ההוראות שיתבצעו במהלך הרצת האלגוריתם במחשב.

**שימו ♥:** המדד הוא מספר ההוראות שיתבצעו ולא מספר ההוראות באלגוריתם! מספר ההוראות באלגוריתם אינו מעיד בהכרח על מספר ההוראות שיתבצעו במהלך הרצתו. בפרט, עבור אלגוריתם הכולל לולאה, ייתכן שמספר ההוראות שבו הוא מועט, אך מספר ההוראות שיתבצעו הוא רב, כי כל הוראה בלולאה יכולה להתבצע כמה פעמים. נפרט נקודה זו במהלך הפרק.

מדידת זמן-הביצוע **לא** נעשית על פי הזמן בפועל שאורכת ריצת תוכנית המיישמת את האלגוריתם. יש לכך כמה סיבות:

1. יש מחשבים מהירים יותר ומהירים פחות. מכיוון שפעולות היסוד נמשכות זמן שונה ממחשב למחשב, זמן ההרצה בפועל של אותה תוכנית יכול להיות שונה ממחשב למחשב.

2. גם אם נריץ את התוכנית באותו מחשב כמה פעמים ייתכן שבכל פעם ההרצה תימשך פרק זמן שונה, כתלות בעומס המחשב באותו הזמן. ניתן לדמות זאת לזמן המתנה במסעדה: זמן ההמתנה קצר יותר כאשר במסעדה פחות אורחים. בדומה, זמן ההמתנה לסיום ריצת תוכנית קצר יותר כאשר יש מעט תוכניות נוספות אשר רצות במקביל במחשב.

3. גם המהדר (הקומפיילר) שהשתמשנו בו כדי להדר את התוכנית משפיע על מהירותה של תוכנית. מהדרים שונים יוצרים תוכניות שונות (בשפת מכונה) ולכל אחת מהן זמן ריצה שונה במקצת, גם אם הן מורצות באותו המחשב ובאותה השעה.

בפרק זה נתמקד במדד זמן-הביצוע של אלגוריתם ונכיר אלגוריתמים שונים לפתרון אותה הבעיה. האלגוריתמים ייבדלו זה מזה בזמן-הביצוע, לפעמים בצורה משמעותית. למדד המקום נתייחס בפרק 10.

נמחיש את ההבדל בין זמני הביצוע של שני אלגוריתמים שונים באמצעות בעיה שאנו נתקלים בה מדי פעם בחיי יום-יום, והיא גם אחת מבעיות היסוד במדעי המחשב. הבעיה היא בעיית חיפוש ברשימה ממוינת:

**קלט:** רשימה של שמות ממוינים לפי סדר הא"ב (למשל מדריך טלפונים או דף קשר).

**פלט:** הודעה מתאימה אם שם מסוים מופיע ברשימה.

ניתן לחפש את השם המבוקש בשתי דרכים :

הדרך הראשונה, היא בחיפוש **סדרתי**, כלומר מעבר על פני הרשימה, שם אחרי שם, עד שנמצא את השם המבוקש או עד שיתברר שהשם לא מופיע ברשימה. ברשימות ארוכות מאוד, למשל כמו ספר טלפונים, חיפוש כזה עלול לקחת הרבה מאוד זמן.

אם תיזכרו כיצד אתם מחפשים בספר טלפונים תיווכחו שלחיפושים ברשימות ארוכות אנו משתמשים בדרך כלל בחיפושים לא סדרתיים, כמו החיפוש ה**לא-סדרתי** הבא: נתבונן בשם המופיע באמצע הרשימה, ונשווה אותו לשם המבוקש, אם השמות זהים – סיימנו את תהליך החיפוש. אם לא, נבדוק לאן עלינו להמשיך את החיפוש: אחרי השם המבוקש או לפניו (כלומר בחציה השני של הרשימה או בחציה הראשון). אם השם המבוקש מופיע בסדר הא"ב אחרי השם האמצעי נמשיך את תהליך החיפוש רק בחצי השני (כיוון שהרשימה ממוינת, לא ייתכן שיהיה בחצי הראשון), ואם השם המבוקש מופיע בסדר הא"ב לפני השם האמצעי נמשיך את התהליך רק בחצי הראשון. תהליך החיפוש ימשיך בדיוק באותו אופן: בדיקת השם האמצעי והשוואתו לשם המבוקש, ובהתאם לתוצאת ההשוואה המשיך חיפוש במחצית התחום המתאימה, וכך הלאה עד מציאת השם או עד שהסתיים התהליך (והשם לא נמצא). חיפוש זה נקרא חיפוש **בינרי**, כיוון שהוא מבוסס על הקטנת תחום החיפוש לחצי מגודלו אחרי כל השוואה של השם המבוקש לשם נבחר (האמצעי בתחום החיפוש).

נדגים זאת :

לפנינו רשימה ממוינת של מספרים :

1, 3, 5, 6, 7, 9, 12, 13, 15, 19, 20, 24, 25, 27, 31, 35, 36

וברצוננו לחפש בתוכה את המספר 27.

בשיטת **החיפוש הסדרתי**, דרושות לנו 14 פעולות השוואה עד שנמצא את המספר המבוקש (כי 27 נמצא במקום ה-14 ברשימה).

נבדוק מה קורה בשיטת **החיפוש הבינרי** :

♦ נתבונן במספר האמצעי ברשימה, 15. כיוון שהמספר המבוקש **גדול** ממנו, נמשיך את החיפוש מימינו.

♦ כעת אנו מתמקדים ברשימת המספרים: 19, 20, 24, 25, 27, 31, 35, 36.

♦ נתבונן במספר האמצעי ברשימה זו, 25 (למעשה, אורך הרשימה זוגי ולכן שני מספרים נמצאים באמצע הרשימה; בחרנו שרירותית את הקטן מביניהם). כיוון שהמספר המבוקש **גדול** ממנו, נמשיך את החיפוש מימינו.

♦ כעת אנו מתמקדים ברשימת המספרים: 27, 31, 35, 36.

♦ נתבונן במספר האמצעי ברשימה זו, 31 (גם הפעם, זהו המספר הקטן מבין שני המספרים שבאמצע הרשימה). כיוון שהמספר המבוקש **קטן** ממנו, נמשיך את החיפוש משמאלו.

♦ כעת אנו מתמקדים ברשימת המספרים: 27.

זו רשימה בת מספר אחד, והוא שווה בדיוק למספר המבוקש, ולכן סיימנו את תהליך החיפוש.

בסך הכול ביצענו ארבע השוואות (למספרים 15, 25, 31 ו-27).

בהשוואה לחיפוש הסדרתי ההבדל הוא משמעותי!

כאשר רשימת השמות ארוכה, יש חשיבות רבה לבחירת הדרך שתאפשר את ביצוע משימת החיפוש בזמן קצר עד כמה שניתן. זמן החיפוש נקבע בעיקר על פי מספר ההשוואות המתבצעות במהלך החיפוש (השוואות של השם המבוקש לשם נבחר). לכן נעדיף לבחור בדרך אשר בה יתבצעו פחות השוואות. עבור רשימה בת 1000 שמות, מספר ההשוואות בחיפוש הסדרתי עלול להיות

קרוב ל-1000, כיוון שייתכן שהשם המבוקש נמצא בקצה הרשימה. לעומת זאת, בחיפוש בינרי באותה רשימה מספר ההשוואות לא יעלה בכל מקרה על 10 השוואות. לכן עבור בעיית החיפוש הנתונה, חיפוש בינרי יעיל הרבה יותר. בפרקים הבאים נכיר חיפוש בינרי ביתר פירוט.

יש אנשים הסוברים שמחשבים הינם כה מהירים עד שאין משמעות למושג "זמן-ביצוע של אלגוריתם (או של תוכנית)" ולהשוואה בין זמני הביצוע של אלגוריתמים שונים. לדעה זו אין כל בסיס. קיימות בעיות אשר עבורן זמן הריצה של תוכנית יכול להיות דקות רבות, שעות ואף ימים (למשל תוכניות לחיזוי מזג אויר). עבור בעיית החיפוש שהצגנו, זמן הריצה של תוכנית המבצעת חיפוש סדרתי ברשימה בת 10,000,000 שמות עלול להיות מספר דקות. לעומת זאת, זמן הריצה של תוכנית המבצעת חיפוש בינרי באותה רשימה ובאותו המחשב לא יעלה על מספר שניות.

כזכור, זמן-הביצוע של אלגוריתם נמדד על פי מספר ההוראות שיתבצעו במהלך ביצוע האלגוריתם. מספר זה תלוי בדרך כלל בקלט של האלגוריתם.

למשל, בבעיית החיפוש הקלט לאלגוריתם הוא רשימת השמות הממוינת והשם שיש לחפש. מספר ההוראות שיתבצעו תלוי באורך רשימת השמות: ככל שהרשימה תהיה ארוכה יותר ייתכנו יותר השוואות, כלומר יתבצעו יותר פעולות השוואה.

בפתרון הבעיה הבאה נראה דוגמה לאלגוריתם אשר זמן-הביצוע שלו תלוי בערכו של הקלט.

## הצ'יה 1

**מטרת הבעיה ופתרונה:** הצגת שלושה אלגוריתמים שונים לפתרון בעיה, הנבדלים זה מזה במידת יעילותם מבחינת זמן-ביצוע. באלגוריתמים היעילים יותר נעשה ניצול טוב של מאפייני קלט-פלט.

פתחו אלגוריתם אשר הקלט שלו הוא מספר שלם גדול מ-1, והפלט שלו הוא המחלקים של המספר הנתון. המחלקים של מספר שלם חיובי הם כל המספרים השלמים החיוביים המחלקים את המספר ללא שארית. לכן למשל עבור הקלט 6 הפלט הדרוש הוא: 1 2 3 6.

ישמו את האלגוריתם בשפת C#. תארו את זמן-ביצוע האלגוריתם על ידי הערכת מספר ההוראות שיתבצעו במהלך ביצוע האלגוריתם.

## בדיקת דוגמאות קלט

### שאלה 8.1

צינו מהו הפלט המתאים עבור כל אחד מהקלטים הבאים:

א. 12

ב. 29

ג. 30

תחום המספרים השלמים שמתוכו יש להציג מחלקים הוא התחום שבין 1 ובין המספר הנתון כקלט. כלומר, יש להציג כל מספר שהוא גדול או שווה ל-1 וקטן או שווה למספר הנתון ומחלק אותו ללא שארית.

**?** כיצד אפשר לחשב את המספרים הדרושים לפלט?

הנה רעיון ראשון: נסרוק את תחום המספרים בין 1 לבין המספר הנתון, ונבדוק עבור כל מספר בתחום אם הוא מחלק את המספר הנתון ללא שארית.



ננסח זאת ניסוח ראשוני:

סריקת כל המספרים החיוביים בין 1 ועד למספר הנתון כקלט, ובדיקה עבור כל מספר אם הוא מחלק את המספר הנתון. אם כן, הצגתו כפלט.

?

הרעיון המתואר מציג תת-משימה לביצוע-חוזר. מהי תת-משימה זו?  
הרעיון המתואר כולל ביצוע-חוזר של התת-משימה הבאה, עבור כל מספר בתחום שבין 1 למספר הנתון:

אם המספר מחלק את המספר הנתון, הצגתו כפלט.

ננסח אלגוריתם לביטוי הרעיון המתואר.

## בחירת משתנים

משתני האלגוריתם הם:

**num** – מטיפוס שלם, לשמירת נתון הקלט.

**i** – מטיפוס שלם, משתנה בקרה של ההוראה לביצוע-חוזר.

## האלגוריתם

### אלגוריתם 1:

1. קלט מספר שלם גיבוי  $num$ -2
2. עבור כל מספר שלם גיבוי  $i$  הקטן מ- $num$  או שווה לו כ33:
  - 2.1. אם  $i$  מחלק את  $num$  לא שארית
  - 2.1.1. הצג את ערכו של  $i$

**שימו ♥:** מתאים להשתמש כאן בהוראה לביצוע-חוזר, שמספר הסיבובים בה מחושב מראש (ממומש בשפת C# בלולאת `for`). בהוראה אנו מפרטים את תחום הערכים הנסרק באמצעות משתנה הבקרה של הלולאה ( $i$ ).

באלגוריתם נעשה שימוש במשתנה הבקרה בגוף הלולאה. גוף הלולאה כולל בדיקה אם ערכו של משתנה הבקרה  $i$  הוא מחלק של  $num$ . כיוון שבמהלך ביצוע הלולאה ערכי משתנה הבקרה משתנים מ-1 עד  $num$ , הרי בדיקת החלוקה מתבצעת עבור כל מספר שלם בתחום 1 עד  $num$ .

## יישום האלגוריתם

הנה התוכנית המיישמת את אלגוריתם 1:

```
/*
קלט: מספר שלם חיובי
פלט: כל המחלקים של המספר הנתון
*/
using System;
public class Divisors1
{
    public static void Main ()
    {
        int num; // המספר הנתון
        int i; // משתנה הבקרה
        Console.WriteLine("Enter a number: ");
        num = int.Parse(Console.ReadLine());
        Console.WriteLine("The divisors of {0} are: ", num);
        for(i=1; i <= num ; i++)
```

```

        if (num % i == 0)
            Console.WriteLine("{0}", i);
    } // Main
} // Divisors1

```

ניגש עתה לחישוב זמן-ביצוע האלגוריתם על ידי הערכת מספר ההוראות שיתבצעו במהלך הביצוע. כדי לבצע את ההערכה נתמקד במרכיב העיקרי באלגוריתם המשפיע על מספר ההוראות שיתבצעו.

**?** מהו המרכיב העיקרי באלגוריתם המעיד על מספר ההוראות שיתבצעו?

באלגוריתם אשר אינו כולל לולאה אין בעצם מרכיב בולט. מספר ההוראות שיתבצעו במהלך ביצוע האלגוריתם הוא לכל היותר מספר ההוראות באלגוריתם. לעומת זאת, באלגוריתם הכולל לולאה, הלולאה היא המרכיב העיקרי המעיד על מספר ההוראות שיתבצעו. זאת כיוון שייתכן שהלולאה תתבצע מספר רב של פעמים, ובכל ביצוע-חוזר יתבצעו שוב כל ההוראות שבגוף הלולאה.

מכיוון שייתכן שהלולאה תתבצע מספר רב של פעמים, הרי מספר ההוראות שיתבצעו במהלך ריצת האלגוריתם עשוי להיות גדול בהרבה ממספר ההוראות שכתובות באלגוריתם. למשל במהלך ביצוע אלגוריתם 1 עבור הקלט 1000, תתבצע ההוראה לביצוע-בתנאי שבגוף הלולאה 1000 פעמים. מספר זה גדול בהרבה ממספר ההוראות שבאלגוריתם.

בדרך כלל הלולאה תתבצע מספר שונה של פעמים עבור קלטים שונים. לכן כדי לתאר בצורה כללית את מספר הפעמים שלולאה תתבצע, יש לבטא מספר זה על פי הקלט.

**?** מהו מספר הפעמים שתתבצע הלולאה באלגוריתם 1 עבור קלט שערכו  $N$ ?

מספר הפעמים שהלולאה באלגוריתם 1 תתבצע עבור קלט שערכו  $N$  הוא  $N$ . הלולאה תתבצע פעם אחת עבור כל אחד מהערכים 1, 2, 3, ...,  $N$  למשתנה הבקרה  $i$ .

כיוון שלולאה היא המרכיב העיקרי המעיד על מספר הפעולות שיתבצעו באלגוריתם, נהוג להתייחס למספר זה כמדד לחישוב זמן-הביצוע. לכן נאמר שתוצאת חישוב זמן-הביצוע של אלגוריתם 1 היא שלולאת האלגוריתם תתבצע  $N$  פעמים עבור קלט שערכו  $N$ .

ננסה לראות אם ביכולתנו לפתח אלגוריתם יעיל יותר מאלגוריתם 1 מבחינת זמן-הביצוע.

**?** האם אפשר לפתח אלגוריתם שבו תהיה לולאה שתתבצע פחות מ- $N$  פעמים עבור קלט שערכו  $N$ ?

כן. ניתן לפתח אלגוריתם ובו לולאה "יעילה" יותר. נראה זאת כעת.

אחד המאפיינים של חלוקה של מספרים שלמים היא שהמחלקים של מספר שלם חיובי  $num$  אינם גדולים מ- $num/2$ , מלבד המספר  $num$  עצמו. ניתן לראות זאת בדוגמאות הקלט שנבדקו בשאלה 8.1. לכן בעצם אפשר "לחסוך" ולהריץ את משתנה הבקרה בלולאה שבאלגוריתם רק עד ל- $num/2$ . נבטא רעיון זה באלגוריתם 2, והוא אלגוריתם יעיל יותר לפתרון הבעיה.

## אלגוריתם 2:

1. קלט מספר שלם גיבוי  $num$ -2
2. עבור כל מספר שלם גיבוי  $i$  הקטן מ- $num/2$  או שווה לו כ33:
  - 2.1. אם  $i$  מחלק את  $num$  אזי שאיך
    - 2.1.1. הצג את ערכו של  $i$
  3. הצג את ערכו של  $num$

אמנם הוצאנו את הצגתו של num אל מחוץ ללולאה, אך עיקר העבודה נעשית בלולאה. באלגוריתם 2, תתבצע הלולאה רק  $N/2$  פעמים עבור קלט שערכו N. זהו שיפור משמעותי, במיוחד עבור קלטים שערכם גדול (למשל 10,000).

## יישום האלגוריתם

הנה התוכנית המיישמת את אלגוריתם 2:

```
/*
קלט: מספר שלם חיובי
פלט: כל המחלקים של המספר הנתון
*/
using System;
public class Divisors2
{
    public static void Main ()
    {
        int num; // המספר הנתון
        int i; // משתנה הבקרה
        Console.WriteLine("Enter a number: ");
        num = int.Parse(Console.ReadLine());
        Console.WriteLine("The divisors of {0} are: ", num);
        for(i = 1; i <= num / 2 ; i++)
            if (num % i == 0)
                Console.WriteLine("{0} ", i);
        Console.WriteLine(num);
    } // Main
} // Divisors2
```

שיפרנו את הפתרון הראשון. האם נוכל להמשיך ולשפר גם את הפתרון השני?

**?** האם אפשר לפתח אלגוריתם שבו תהיה לולאה שתתבצע פחות מ- $N/2$  פעמים עבור קלט שערכו N?

כן. אפשר לפתח אלגוריתם ובו לולאה "יעילה" יותר.

לכל מספר שלם k שהוא מחלק של num, יש "בן-זוג" שלם  $num/k$ , שגם מחלק את num. (שהרי  $num = k \cdot num/k$ ). אם באלגוריתם, עבור כל מחלק k שנמצא, נציג כפלט גם את  $num/k$ , לא נצטרך לעבור על כל המחלקים של num, אלא רק עד מחציתם (שהרי הצגנו כבר את "בני-הזוג").

**?** היכן נמצא קו המחצית של המחלקים? כלומר, מתי עלינו להפסיק את החיפוש כדי לא להציג מחלקים כפולים?

התשובה היא  $\sqrt{num}$ . כך נובע מהאבחנה הבאה: עבור כל זוג מחלקים k ו- $num/k$ , לפחות אחד מהם אינו גדול מ- $\sqrt{num}$ . מדוע? משום שאם שניהם גדולים מ- $\sqrt{num}$ , אז מכפלתם גדולה מ- $num$ ! אם כך, אם נסרוק את כל המספרים מ-1 עד  $\sqrt{num}$ , ונציג עבור כל מחלק גם את "בן-זוגו" המחלק, למעשה נציג את כל המחלקים של num.

**?** מהו בן זוגו של  $\sqrt{num}$ ?

זהו  $\sqrt{num}$  בעצמו, משום שמכפלתו של  $\sqrt{num}$  בעצמו שווה ל- $num$ .

לכן משום שבן-זוגו של  $\sqrt{num}$  הוא  $\sqrt{num}$  עצמו, עלינו לשים לב לא להציג את  $\sqrt{num}$  פעמיים כפלט.

לדוגמה: אם  $num=100$  אז  $\sqrt{num} = 10$ . כל מחלקיו של 100 הקטנים מ-10 הם: 2, 4, ו-5. "בני זוגם" הם: 50, 25 ו-20, בהתאמה. אם נוסיף להם את  $\sqrt{num} = 10$  נקבל את הרשימה המלאה של כל המחלקים של 100.

לכן בעצם ניתן להריץ את משתנה הבקרה בלולאה שבאלגוריתם עד ל- $\sqrt{num}$  בלבד. נבטא רעיון זה באלגוריתם 3, שהוא אלגוריתם יעיל יותר משני האלגוריתמים האחרים.

### אלגוריתם 3:

1. קאוט מספר שלם  $i$  יוליב  $num$ -2
2. עזור כל מספר שלם  $i$  הקטן מ- $\sqrt{num}$  כצד:
  - 2.1. אק  $i$  מואק אק  $num$  אלא שארי
    - 2.1.1. הצג אק ערכו  $i$
    - 2.1.2. הצג אק ערכו  $num/i$
  3. אק  $\sqrt{num}$  מואק אק  $num$  אלא שארי
    - 3.1. הצג אק ערכו  $\sqrt{num}$

כאמור, הוצאנו אל מחוץ ללולאה את הבדיקה של  $\sqrt{num}$  כדי שלא נציג את ערכו פעמיים במקרה שהוא מחלק את  $num$ . לעומת זאת, אין צורך לטפל מחוץ בלולאה ב- $num$  עצמו, בניגוד לאלגוריתם הקודם, משום שהוא בן הזוג של 1 ולכן יוצג כפלט בסיבוב הראשון בלולאה.

כעת שיפרנו את אלגוריתם 2 באופן משמעותי. עבור קלטים גדולים מאוד המספר  $\sqrt{N}$  קטן משמעותית מהמספר  $N/2$ . למשל, אם  $N$  הוא 10,000 אז באלגוריתם 2 יהיו 5000 סיבובים בלולאה, בעוד שבאלגוריתם 3 יהיו 100 סיבובים בלבד!

ישמו בעצמכם את האלגוריתם בשפת C#.

### סוף פתרון בעיה 1

נסכם את הנלמד מפתרון בעיה 1:

- ♦ באלגוריתם שאינו כולל לולאה מספר הפעולות שיתבצעו הוא לכל היותר מספר ההוראות באלגוריתם.
- ♦ באלגוריתם שכולל לולאה, מספר ההוראות שיתבצעו עשוי להיות גדול בהרבה ממספר ההוראות האלגוריתם. מספר ההוראות שיתבצעו תלוי במספר הפעמים שהלולאה תתבצע. לכן מספר הפעמים שהלולאה תתבצע משמש כמדד לזמן-ביצוע האלגוריתם.
- ♦ מספר הפעמים שהלולאה תתבצע תלוי בדרך כלל בערכו של הקלט, ומבוטא באמצעות ערכו.

בפתרון בעיה 1 פיתחנו תחילה אלגוריתם אחד, ואחר כך פיתחנו אלגוריתם שני אשר היה יעיל יותר מבחינת מספר הפעמים של ביצוע הלולאה, כלומר מבחינת זמן-הביצוע. ולבסוף מצאנו אלגוריתם יעיל יותר גם ממנו. בפיתוח האלגוריתם השני השתמשנו בעובדה שאיברי הפלט, הלא הם כל המחלקים של נתון הקלט  $N$ , אינם גדולים מ- $N/2$  מלבד  $N$  עצמו. בפיתוח האלגוריתם השלישי, היעיל מבין השלושה, השתמשנו בעובדה שאיברי הפלט, ניתנים לחלוקה לזוגות כך שמכפלת איברי כל זוג שווה ל- $N$  ובכל זוג יש איבר אחד קטן מ- $\sqrt{N}$ . העובדה האחרונה אפשרה לנו לכתוב לולאה אשר תתבצע  $\sqrt{N}$  פעמים בלבד, במקום הלולאה המקורית שמתבצעת  $N$  פעמים.

- ♦ פיתוח אלגוריתם יעיל יותר נעשה תוך ניתוח וניצול טוב של מאפייני קלט-פלט. שימוש טוב במאפייני קלט-פלט חשוב לפיתוח לולאה אשר תתבצע מספר מועט של פעמים עד כמה שאפשר. כלומר השקעת מאמץ בניתוח מעמיק של הבעיה יכולה להתבטא אחר כך באלגוריתם שהוא משמעותית יעיל יותר.
- ♦ במהלך פתרון בעיה אלגוריתמית נשתדל לפתח אלגוריתם יעיל ככל האפשר מבחינת זמן-הביצוע. כלומר אלגוריתם שהלולאות שבו יתבצעו מספר פעמים מועט (לולאות "יעילות") עד כמה שניתן.

## שאלה 8.2

ציינו עבור כל אחד מהקלטים הבאים את מספר הפעמים שתתבצע הלולאה בכל אחד משלושת האלגוריתמים שפיתחנו:

א. 1000

ב. 2000

## שאלה 8.3

בקטע התוכנית הבא מחושבת המכפלה של שני נתוני קלט חיוביים שלמים בשימוש בפעולת חיבור בלבד:

```
x = int.Parse(Console.ReadLine());
y = int.Parse(Console.ReadLine());
sum = 0;
for(i = 1; i <= x ; i++)
    sum = sum + y;
Console.WriteLine( "The product is {0}", sum);
```

ידוע שאחד מנתוני הקלט גדול באופן משמעותי מהשני, אך לא ידוע אם זה הנתון הראשון או השני. השתמשו במאפיין זה של הקלט כדי לשנות את קטע התוכנית הנתון כך שיהיה יעיל ככל שניתן.

**הדרכה:** שימו לב שיש חשיבות לבחירת המשתנה אשר על פיו נקבע מספר הפעמים שתתבצע הלולאה.

מהו מספר הפעמים שתתבצע הלולאה של קטע התוכנית הנתון, ומהו מספר הפעמים שתתבצע הלולאה של קטע התוכנית החדש?

## שאלה 8.4

פתחו אלגוריתם מבלי ליישמו שיהיה יעיל ככל האפשר, והקלט שלו הוא שני מספרים שלמים גדולים מ-1 שאינם מחלקים זה את זה, והפלט שלו הוא כל המספרים השלמים החיוביים המחלקים את שני מספרי הקלט. תארו לפי ערכו של הקלט את מספר הפעמים שתתבצע לולאת האלגוריתם.

## שאלה 8.5 (מתקדמת)

נניח שבבעיה 1 הפלט הדרוש הוא כל המספרים השלמים החיוביים הקטנים מ-N שאינם מחלקים את נתון הקלט N. מה יהיה מספר הפעמים שתתבצע הלולאה באלגוריתם לפתרון הבעיה החדשה?

## שאלה 8.6

יש לפתח אלגוריתם אשר הקלט שלו הוא מספר שלם חיובי  $N$ , והפלט שלו הוא כל המספרים השלמים החיוביים אשר קטנים מ- $N$  והשורש שלהם הוא מספר שלם. למשל עבור הקלט 50 הפלט יהיה 1 4 9 16 25 36 49.

האלגוריתם הבא, הכולל משתנים מטיפוס שלם הוא פתרון אפשרי:

```
1. קאוט מספר שלם גילוי  $num$ -2
2. עבור כל מספר שלם גילוי  $i$  שקטן  $num$ - $n$  עד 32:
   2.1. אם השורש של  $i$  הוא מספר שלם
       2.1.1. הצג את ערכו של  $i$ 
```

א. מהו מספר הפעמים שתבצע הלולאה של האלגוריתם הנתון?  
ב. ישמו בביטוי בוליאני בשפת C# את התנאי "השורש של  $i$  הוא מספר שלם"  
ג. אפשר לכתוב אלגוריתם אשר זמן-הביצוע שלו יהיה קצר בהרבה מהאלגוריתם הנתון, וזאת ב"ייצור" מספרי הפלט, באמצעות העלאה בריבוע של כל המספרים השלמים אשר ריבועם קטן מן הקלט.

האלגוריתם החלקי הבא מבוסס על הרעיון המתואר:

```
1. קאוט מספר שלם גילוי  $num$ -2
1.1. עבור כל מספר שלם גילוי  $i$  הקטן  $n$ - עד 32:
   1.1.1. הצג את ערכו של  $i^2$ 
```

השלימו את האלגוריתם ותארו את מספר הפעמים שתבצע הלולאה של אלגוריתם זה.

כל האלגוריתמים שניתחנו עד עתה בפרק כללו לולאה שמספר הביצועים שלה מחושב מראש ובה הורץ משתנה הבקרה מ-1 עד ערך כלשהו בקפיצות של 1. לכן קל היה לחשב את מספר הפעמים של ביצוע הלולאה.

בלולאות מורכבות יותר, בהן ערכו ההתחלתי של משתנה הבקרה אינו 1, והשינוי בערכו בין סיבוב לסיבוב הוא לאו דווקא 1, חישוב מספר הסיבובים עלול להיות מסובך יותר. בלולאת `while` שבה אין משתנה בקרה, חישוב זמן-הביצוע יכול להיות אף מורכב יותר. יש לספור את מספר הפעמים של ביצוע הלולאה במעקב אחר שינוי ערכי משתנים המתעדכנים בגוף הלולאה ומופיעים בתנאי הכניסה ללולאה.

## שאלה 8.7

יש לפתח וליישם אלגוריתם אשר הקלט שלו הוא שני מספרים שלמים חיוביים, כך שהמספר השני גדול מהראשון. הפלט הדרוש הוא הכפולות של המספר הראשון אשר קטנות מהמספר השני או שוות לו. למשל עבור הקלט 1000 200 הפלט יהיה 200 400 600 800 1000. משפטי התוכנית הבאים הם יישום של אלגוריתם לפתרון הבעיה:

```
x = int.Parse(Console.ReadLine());
y = int.Parse(Console.ReadLine());
i = x;
for (i = x; i <= y; i++)
    if (i % x == 0)
        Console.WriteLine(i);
```

א. כמה פעמים תבצע הלולאה עבור הקלט 1000 200?

ב. תארו בצורה כללית את מספר הפעמים שתבצע הלולאה על פי ערכי נתוני הקלט  $x$  ו- $y$ .

ג. בפתרון המוצג i גדל בקפיצות של 1. ניתן לשפר את יעילות הפתרון הנתון בשינוי הקפיצות של i לקפיצות גדולות מ-1, קפיצות אשר מתאימות למרחק בין זוג מספרי פלט עוקבים (שימו לב שהמרחק בין כל זוג מספרי פלט עוקבים הוא אחיד). כתבו פתרון יעיל יותר המבוסס על הרעיון המתואר, ותארו את מספר הפעמים שתבצע לולאת הפתרון החדש.

## שאלה 8.8

נתונה לולאת ה-for הבאה:

```
for (i = 1 ; i <= 30000 ; i++)  
    if (i % 500 == 0)  
        Console.WriteLine(i);
```

א. מהו מספר הפעמים שתבצע הלולאה?

ב. מהי מטרת הלולאה?

ג. כתבו לולאה יעילה הרבה יותר להשגת אותה המטרה. מהו מספר הפעמים שתבצע הלולאה היעילה שכתבתם? פי כמה מספר זה קטן מתשובתכם בסעיף א?

## סיכום

בפרקים הקודמים בחנו אלגוריתמים על פי קנה המידה **נכונות**. בפרק זה הכרנו קנה מידה חדש – **יעילות**.

**יעילות של אלגוריתם** נמדדת על פי "משאבי המחשב" הדרושים לביצוע האלגוריתם. משאבים אלה הם גודל המקום בזיכרון והזמן הדרוש לביצוע.

**גודל המקום** נמדד בעיקר על פי מספר המשתנים באלגוריתם.

**זמן-הביצוע** נקבע לפי מספר פעולות היסוד שיתבצעו במהלך ביצוע האלגוריתם.

**פעולות היסוד** הן: פעולות קלט, פעולות פלט ופעולות חישוב. בצורה פשטנית ניתן לומר, שכל הוראה באלגוריתם כוללת פעולת יסוד אחת, ומכאן – **מדידת זמן-הביצוע** של אלגוריתם נעשית על פי מספר ההוראות שיתבצעו במהלך ריצת האלגוריתם.

באלגוריתם אשר אינו כולל לולאה מספר ההוראות שיתבצעו במהלך הביצוע הוא לכל היותר מספר ההוראות באלגוריתם.

לעומת זאת באלגוריתם הכולל לולאה, מספר ההוראות שיתבצעו במהלך הביצוע אינו נקבע על פי מספר ההוראות באלגוריתם, אלא על פי מספר הפעמים שהלולאה תתבצע. מספר זה יכול להיות תלוי בקלט של האלגוריתם ואז הוא מבוטא באמצעות מאפייני הקלט.

כאשר נתונים שני אלגוריתמים שונים לפתרון בעיה, משווים את יעילותם מבחינת זמן-הביצוע לפי מספר הפעמים שהלולאות שבהם יתבצעו במהלך הרצת כל אלגוריתם.

בפיתוח אלגוריתם נשתדל לבחור לולאות שיתבצעו מספר פעמים מועט עד כמה שניתן, כלומר לולאות "יעילות" ככל האפשר. בחירת לולאה יעילה נעשית בניתוח ובניצול טוב של מאפייני הקשר בין הקלט לפלט.

יסודות מדעי המחשב 1

**מדריך מעבדה לסביבת העבודה**

**Visual C# Express**

**כתבה: יעל בילצ'יק (סופרין)**

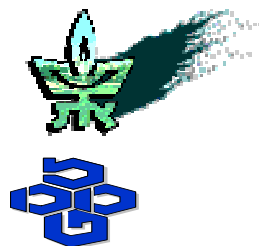
**מהדורת עיצוב**

תשס"ו 2006

אוניברסיטת תל-אביב החוג להוראת המדעים

מטה מל"מ המרכז הישראלי להוראת המדעים ע"ש עמוס דה-שליט

משרד החינוך האגף לתכנון ולפיתוח תכניות לימודים





---

אין לשכפל, להעתיק, לצלם, לתרגם או לאחסן במאגר מידע כל חלק שהוא מחומר הלימוד של ספר זה. שימוש מסחרי מכל סוג שהוא בחומר הכלול בספר זה אסור בהחלט, אלא ברשות מפורשת בכתב מהגורמים המפורטים להלן.



כל הזכויות שמורות

**אוניברסיטת תל-אביב ומשרד החינוך**

## **תוכן העניינים**

5	.....פתיחת סביבת העבודה.
5	.....יצירת פרויקט
6	.....חלונות סביבת העבודה.
7	.....כתיבת תוכנית ראשונה.
8	.....הידור תוכנית.
9	.....הרצת תוכנית.
10	.....שמירת תוכנית.
11	.....יצירת פרויקט נוסף.
11	.....הקלדת קלט.
12	.....ניפוי שגיאות.
15	.....עבודה מתקדמת עם מנפה השגיאות.
16	.....טבלת מקשי קיצור שימושיים לפעולות בסביבת העבודה:



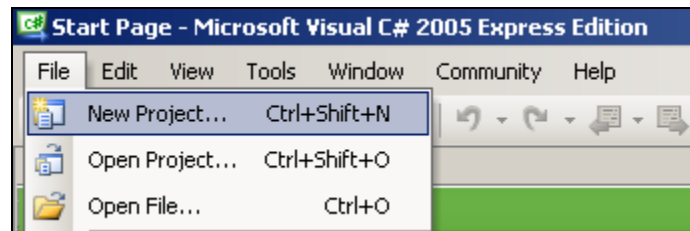
סביבת העבודה Visual C# Express מאפשרת לנו לכתוב תוכניות בשפת C#, החל מתוכניות פשוטות למדי בעלות שורות קוד בודדות, ועד מערכות מסחריות מורכבות בעלות אלפי שורות קוד. במדריך זה נלמד כיצד להשתמש בסביבה זו באופן הנוח ביותר לצרכינו כמתכנתים מתחילים.

## פתיחת סביבת העבודה

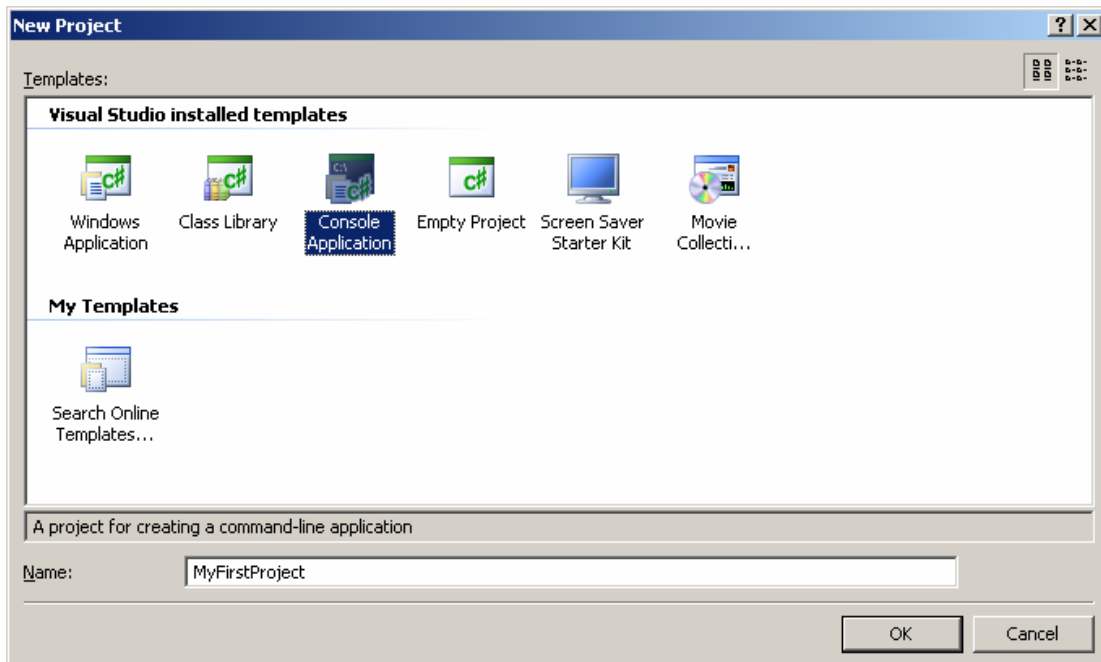
כדי לפתוח את סביבת העבודה נבחר את התוכנה Microsoft Visual C# 2005 Express Edition, דרך תפריט ההתחלה של "windows". יתקבל מסך הפתיחה של סביבת העבודה.

## יצירת פרויקט

כל תוכנית בשפת C# נמצאת בתוך פרויקט נפרד. לכן, כדי לכתוב תוכנית חדשה עלינו לפתוח פרויקט חדש. נבחר בתפריט העליון את File, ושם נבחר ב-New Project.



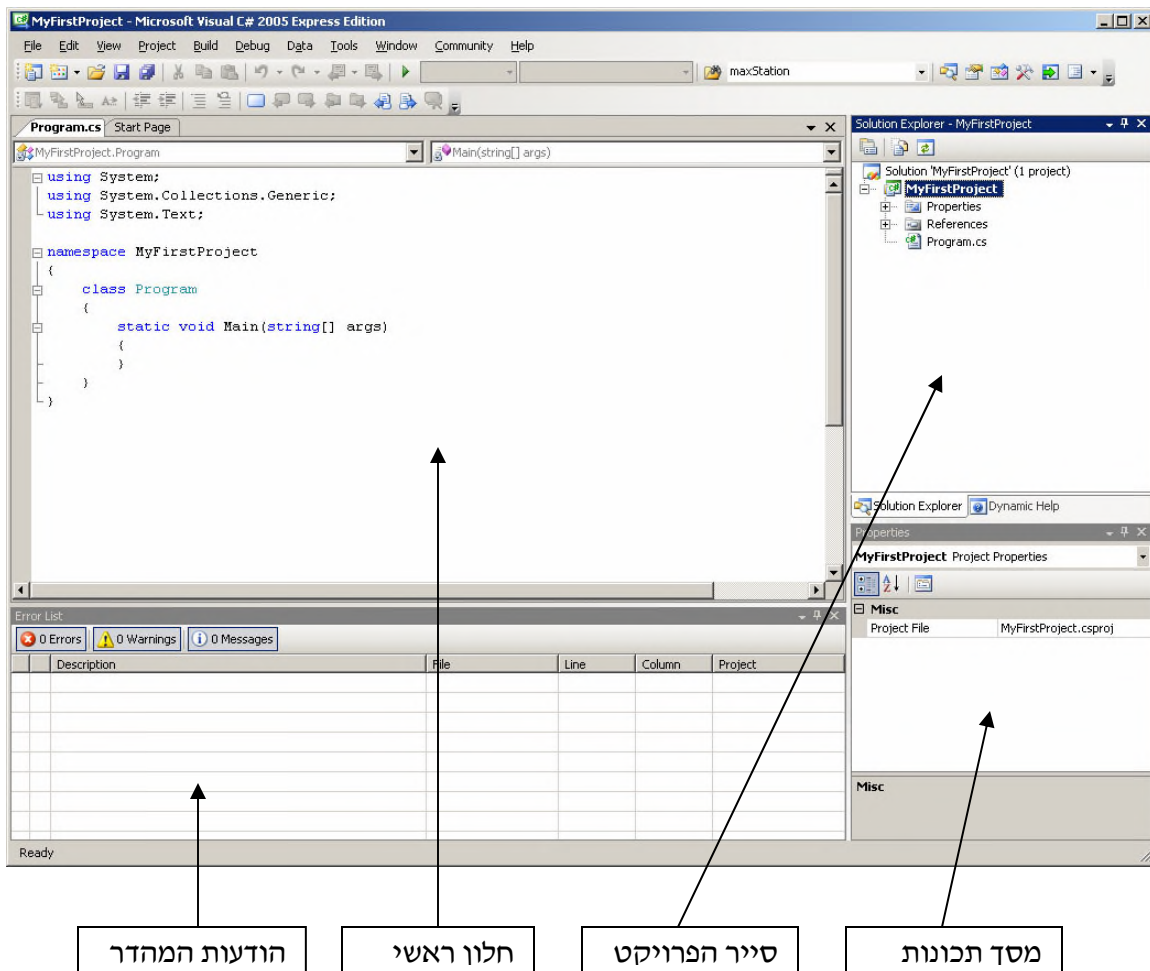
יפתח חלון המציע לנו סוגי פרויקטים שונים. נבחר ב-Console Application. בשורת השם נכתוב את השם שנקרא לפרויקט. בדוגמה זו בחרנו בשם "MyFirstProject".



לאחר הלחיצה על "OK" יתקבל מסך סביבת העבודה, כאשר בחלון המרכזי כבר מופיע בסיס של תוכנית ריקה בשפת C#.

## חלונות סביבת העבודה

נכיר את החלונות השונים בסביבת העבודה:



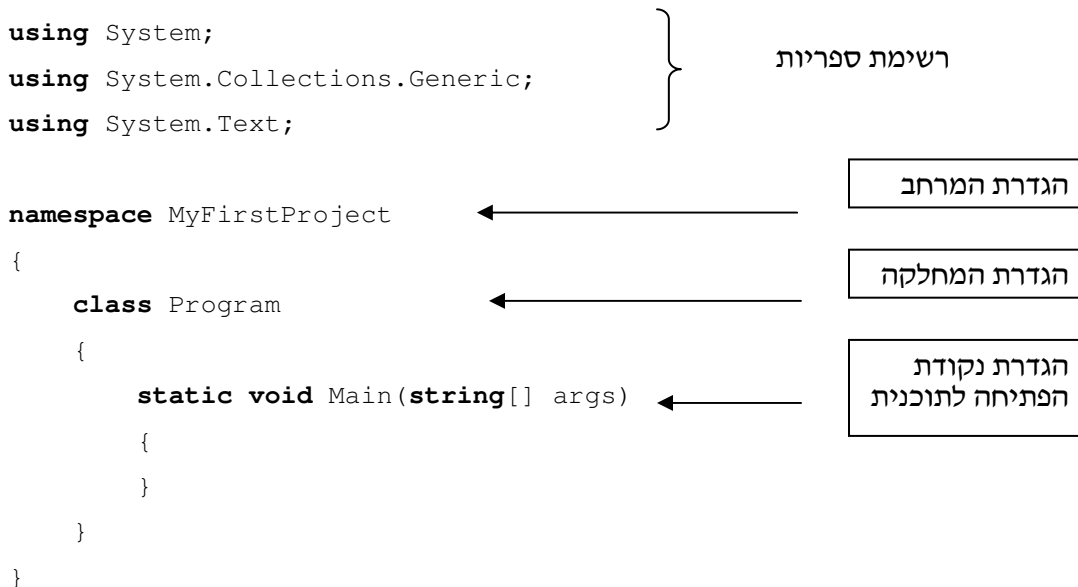
**החלון הראשי** ישרמש אותנו לכתיבת קוד התוכנית

בחלון **הודעות המהדר** נצפה בשגיאות והערות המהדר (קומפילר) לאחר כל הידור (קומפילציה).

בחלון **סייר הפרויקט** נשתמש על מנת לעבור בנוחות בין חלקי הפרויקט השונים. ניתן לראות כי הפרויקט שפתחנו קיים בתוך Solution. כאשר פתחנו פרויקט חדש נפתח עבורו Solution חדש. משמעות המילה solution היא פתרון. בסביבת העבודה, Solution שומר בתוכו פרויקט אחד או יותר, ומקובל שיהיו אלה פרויקטים שיש ביניהם קשר, והם מהווים במובן מסוים פתרון לבעיה מורכבת אחת. בהמשך נלמד כיצד ניתן ליצור כמה פרויקטים בתוך Solution אחד.

חלון מסך התכונות משמש בעיקר עבור תוכניות בהן נעשה שימוש בעזרים גרפיים ולכן לא נשתמש בו.

ניתן לסגור כל חלון אם אין בו שימוש, ולפתוח אותו שוב בעזרת התפריט View.  
נתבונן בקוד (כלומר, רצף הוראות בשפת התכנות) אשר מופיע באופן אוטומטי במסך התוכנית עם פתיחת פרויקט חדש:



לא כל ההגדרות הכרחיות לצרכינו, לכן נוכל למחוק את ההגדרות שלא נזדקק להן ולהישאר עם השלד המוכר לנו:

```
using System;

class Program
{
    static void Main()
    {
    }
}
```

## כתיבת תוכנית ראשונה

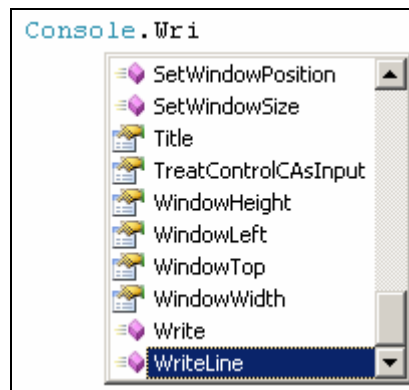
כעת אנו יכולים להקליד את התוכנית הראשונה, שתציג למסך את הפלט "Hello world".  
נשנה את שם המחלקה לשם שנרצה לקרוא לתוכנית (אין חובה לשנות את השם, אך רצוי תמיד לתת לתוכניות שמות משמעותיים, שמביעים את תפקידן), ונוסיף את פקודת ההדפסה בתוך תחום ה-  
:Main

```
using System;

class HelloWorld
{
    static void Main()
    {
        Console.WriteLine("Hello world");
    }
}
```

## שימו ♥:

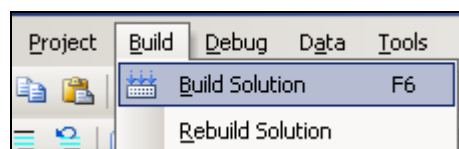
- ישנם צבעים שונים בקוד הכתוב: מילים שמורות נכתבות בכחול, מחרוזות באדום, שמות מחלקות בטורקיז.
- מייד לאחר כתיבת שם המחלקה Console נפתח חלון המציג את כל התכונות והפעולות של המחלקה בהן נוכל להשתמש. לפקודת הדפסה נבחר את הפעולה WriteLine.



- תוכלו לעבור ולבדוק תכונות ופעולות נוספות השייכות למחלקה Console.
- למשל, ההוראה `Console.ForegroundColor = ConsoleColor.Red;` תשנה את צבע הפלט לאדום.
- כדי לקבל הנחיות על כל תכונה ופעולה אפשר לעמוד עם הסמן על התכונה או הפעולה המבוקשת ולהקיש על F1.

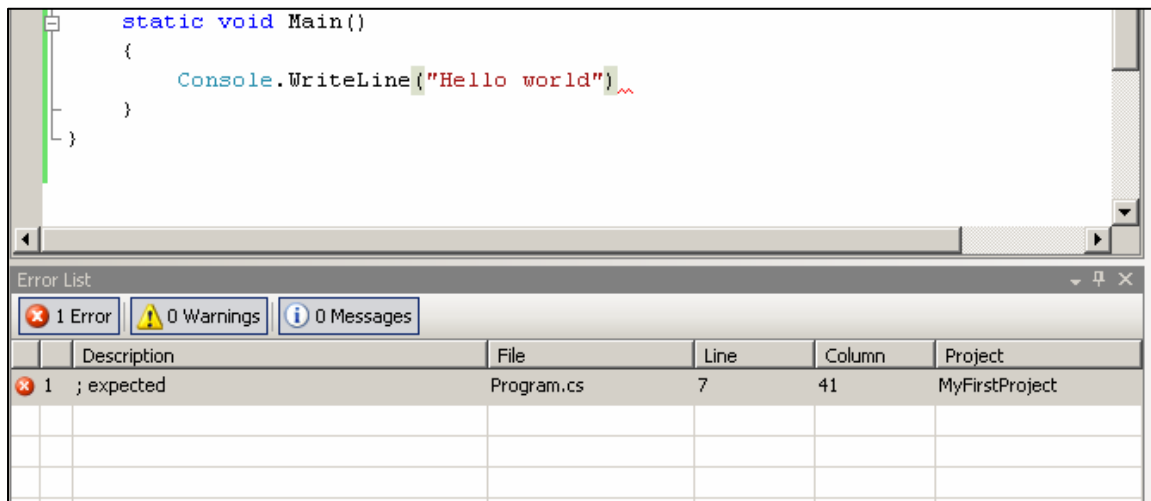
## הידור תוכנית

אחרי סיום הקלדת התוכנית, עלינו להדר (לקמפל) אותה כדי לבדוק שאין בה שגיאות תחביריות, ולהכין אותה לריצה. שלב זה יתבצע על ידי בחירת Build בתפריט, ואז בחירת Build Solution, או על ידי הקשה על המקש F6.



אם אין כלל שגיאות תחביר בתוכנית, נקבל את ההודעה Build succeeded בצד השמאלי התחתון של המסך. אם קיימות שגיאות תחביר בתוכנית נקבל הודעות מתאימות בחלון הודעות המהדר. לחיצה כפולה על ההודעה תקפיץ את הסמן למקום בו ארעה השגיאה.

למשל, השגיאה הבאה נגרמה כיוון שלא נכתב הסימן ; בסוף משפט:



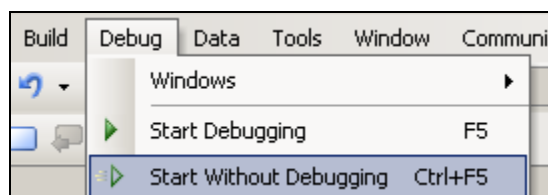
לא נוכל להריץ תוכנית לפני שנתקן את כל שגיאות התחביר. לאחר שנתקן את השגיאות, נהדר שוב את התוכנית, וכך נחזור על התהליך עד אשר לא יופיעו שגיאות בחלון הודעות המהדר, ונקבל את ההודעה Build succeeded.

**שימו ⚡:** ייתכן שלאחר הידור של תוכנית נקבל **הערות** הרצה. ההערות מסומנות בסימן צהוב (בעוד השגיאות מסומנות באדום). ניתן להריץ תוכנית שהתקבלו עבורה הערות, אך יש לתת את הדעת על הערות אלה, משום שייתכן שהן מצביעות על טעות או על בעיה אפשרית אחרת בתוכנית.

## הרצת תוכנית

לאחר שהתוכנית עברה בהצלחה את שלב ההידור, היא מוכנה להרצה.

כדי להריץ את התוכנית נבחר בתפריט Debug את Start Without Debugging או נקיש על המקשים Ctrl+F5 בו זמנית.





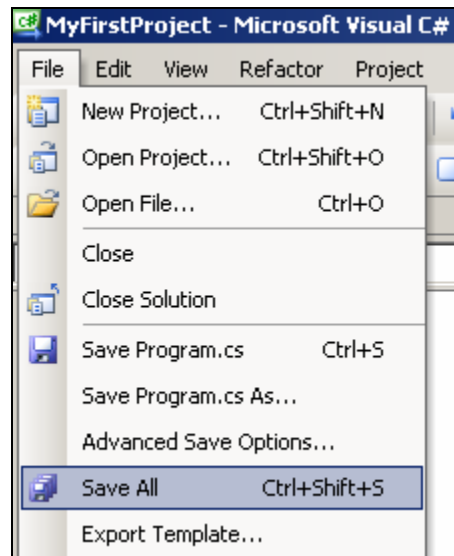
כתוצאה מכך, יפתח חלון ריצת התוכנית, ייכתב בשורה נפרדת המשפט "Hello world", וריצת התוכנית תסתיים. בסיום ריצת התוכנית חלון ההרצה ייסגר רק לאחר הקשה על מקש כלשהו.

```
Hello world
Press any key to continue . . . _
```

**שימו ♥:** אם תנסו להריץ תוכנית אשר לא עברה הידור בהצלחה, או תוכנית שעברה הידור אך לאחר מכן עברה שינוי כלשהו ויש להדרה שוב, יתבצע הידור באופן אוטומטי.

## שמירת תוכנית

לאחר שסיימנו לכתוב את התוכנית, להדר אותה (תוך תיקון שגיאות תחביר, במידת הצורך), להריץ אותה, ולבדוק שאין שגיאות לוגיות בתוכנית, נשמור את קובץ התוכנית לשימוש עתידי. בתפריט File בחרו את Save All או הקישו על המקשים Ctrl, Shift ו-S בו זמנית.



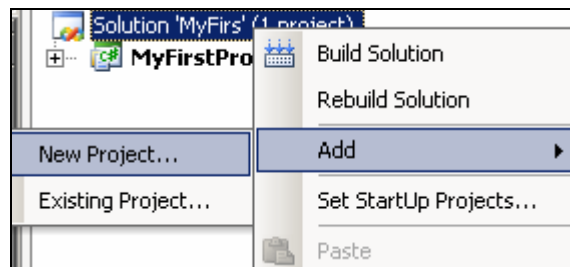
יפתח חלון בו יופיע שם הפרויקט, הכתובת על גבי הדיסק הקשיח בה יישמר הפרויקט, ושם ה-Solution.

כאשר נרצה בעתיד לפתוח תוכנית שמורה נפתח את הפרויקט שלה על ידי Open project שבתפריט File, ונבחר את ה-Solution הנדרש לפי שמו בסיומת .sln.

## יצירת פרויקט נוסף

כעת ברצוננו לכתוב תוכנית נוספת, הדורשת גם הזנת קלט. לפנינו שתי אפשרויות: לסגור את ה-Solution הקיים על ידי Close Solution שבתפריט File, ולפתוח פרויקט חדש כפי שעשינו בתחילה.

אפשרות שנייה היא להוסיף פרויקט ל-Solution קיים. לשם כך ניגש עם העכבר למילה Solution שבסייר הפרויקט, נלחץ על המקש הימני של העכבר ונבחר ב-Add. מהתפריט שיפתח נבחר את NewProject.



כעת ייפתח לנו חלון פרויקט חדש, כפי שקרה כאשר פתחנו Solution חדש.

שימו לב שלאחר הוספת הפרויקט קיימים שני פרויקטים תחת אותו ה-Solution, אך רק אחד מהם פעיל – זה המסומן בהדגשה. ניתן להחליף ולסמן פרויקט שונה כפעיל על ידי בחירתו עם העכבר, הקשה על מקש ימני ובחירת Set as StartUp Project.

כאשר אנו מוסיפים פרויקט ל-Solution, המכיל כבר פרויקטים אחרים, הרי שבסופו של דבר הפרויקטים יישמרו יחדיו, תחת אותו Solution (ואיתם גם התוכנית שכתבתם בכל אחד מהם). לכן, אם אתם כותבים כמה תוכניות בעלות אופי דומה שברצונכם לשמור יחדיו (למשל, כמו כמה תרגילים עבור עבודה אחת להגשה באותו נושא) מומלץ לשמור אותן בפרויקטים נפרדים תחת אותו ה-Solution. אך אם הינכם כותבים כמה תוכניות שאין כל קשר ביניהן, מומלץ לשמור כל אחת מהן ב-Solution נפרד עם שם משמעותי שיקל עליכם את הזיהוי של התוכנית כאשר תחפשו אותה בקבצים השמורים.

## הקלדת קלט

כידוע, זהבה גרמה נזק רב לשלושת הדובים, ולכן החליטה לפצותם בסכום כסף. את הסכום תחלק שווה בשווה בין כל השלושה. הדוב הקטן החליט שאת סכום הכסף שקיבל יחלק לארבעה חסכוניות נפרדים, ואת השארית יבזבז על צנצנת דבש איכותית. עלינו לכתוב תוכנית שתקבל כקלט את סכום הכסף שהקצתה זהבה לתשלום הפיצויים, ותציג כפלט את סכום הכסף שיוכל הדוב הקטן לבזבז על צנצנת דבש איכותית.

הנה תוכנית שנכתבה לצורך פתרון הבעיה, כפי שהוקלדה בסביבת העבודה:

```
using System;

class Bears
{
    static void Main()
    {
        int money, smallBearSum, sumForHoney;
        Console.Write("Insert the sum of money Zeahva has: ");
        money = int.Parse(Console.ReadLine());
        smallBearSum = money / 3;
        sumForHoney = smallBearSum / 4;
        Console.WriteLine("The bear junior will spend {0} shekels for
honey", sumForHoney);
    }
}
```

הקלידו גם אתם את התוכנית הזאת, והדרו אותה.

לאחר שהתוכנית עברה בהצלחה את תהליך ההידור, נריץ אותה.

בתחילה יירשם במסך ריצת התוכנית משפט הפלט

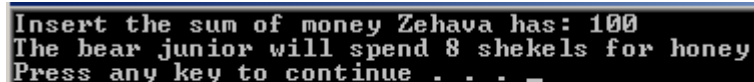
"Insert the sum of money Zeahva has: "

ולאחר מכן התוכנית תעצור פעולתה ותמתין לקלט של מספר שלם. עלינו להקליד בהמשך למשפט זה מספר שלם כלשהו ואחריו להקיש על המקש Enter. רק אז התוכנית תמשיך את ריצתה, תציג את הפלט המחושב ותסיים את פעולתה.

**שימו ♥:** אם התוכנית ממתינה לקבל מספר שלם, אך אנו נקליד קלט שאינו מספר שלם, יגרום הדבר לשגיאת ריצה. ייפתח מסך בו תופיע השאלה: האם ברצונכם לנפות את השגיאות באמצעות כלי לניפוי שגיאות? לחצו על "No". אז תופיע הודעת השגיאה על גבי מסך התוכנית, והתוכנית תפסיק ריצתה.

## ניפוי שגיאות

נתבונן במסך ריצת התוכנית לאחר ריצתה, כאשר הקלט הוא 100:



```
Insert the sum of money Zehava has: 100
The bear junior will spend 8 shekels for honey
Press any key to continue . . . _
```

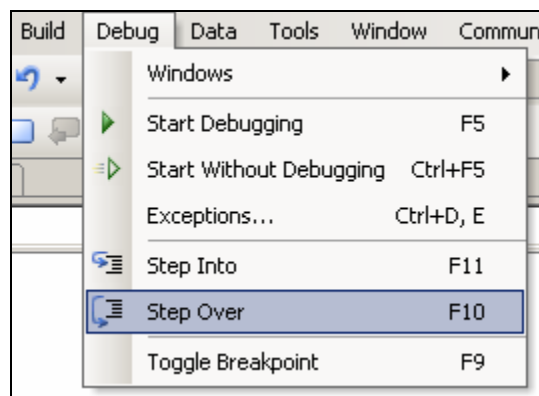
נבחן את התוצאה שהתקבלה:

אם זהבה הקצתה 100 ₪ לפיצויים, הרי כל דוב יקבל 33 ₪. הדוב הצעיר יחלק את הסכום שקיבל, 33 ₪, ל-4 קבוצות ובשארית שיקבל יקנה צנצנת דבש איכותית. שארית החילוק של 33 ב-4 היא 1, לכן הפלט צריך להיות 1. אם נתבונן במסך הפלט נראה כי הפלט הוא 8. מכאן, שיש בתוכנית שכתבנו שגיאה לוגית, שגיאת חישוב במהלך התוכנית.

בתוכנית קצרה כגון זו שכתבנו ניתן להתבונן בתוכנית ולמצוא את השגיאה בקלות יחסית. אך כאשר התוכנית גדולה ומורכבת יותר ניפוי השגיאות הופך למשימה קשה הרבה יותר. לשם כך קיים כלי המאפשר לנו לנפות שגיאות ביתר קלות, ה-Debugger.

בעזרת ה-Debugger ניתן להריץ את התוכנית באופן מבוקר, שורה אחר שורה, בכל שורה נצפה בערך המשתנים ונבדוק כי ערכם תואם לערך הצפוי.

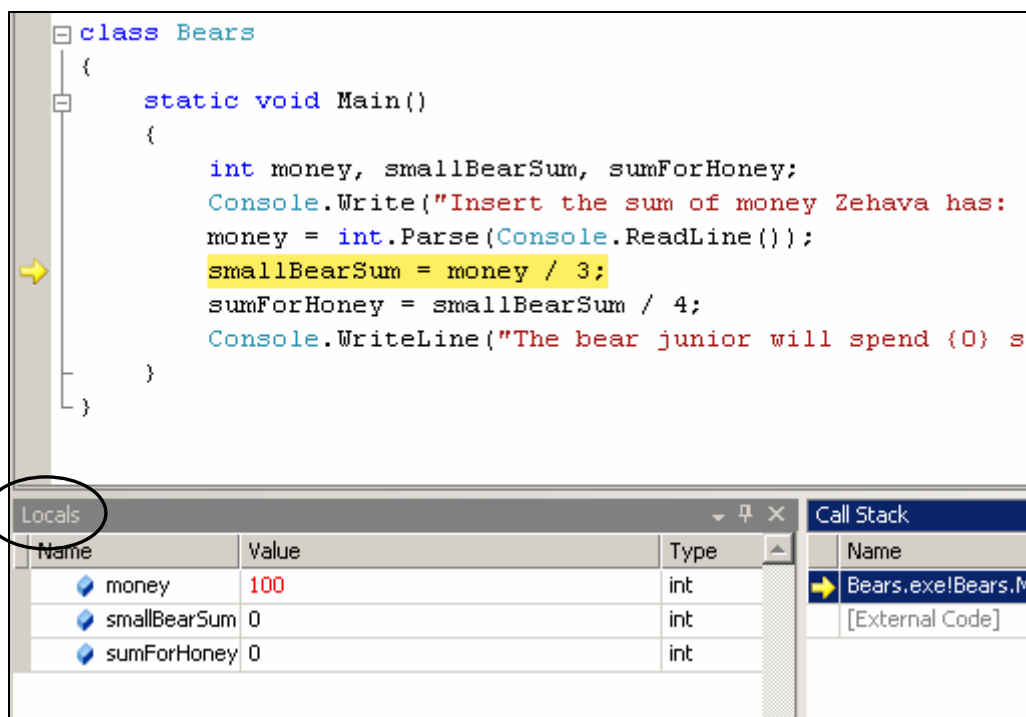
לתחילת הרצה מבוקרת של התוכנית בחרו מהתפריט Debug את Step Over או הקישו על F10.



ריצת התוכנית תחל והשורה הראשונה של התוכנית תיצבע בצהוב. כעת, כל הקשה על F10 תגרום להתקדמות התוכנית לשורה הבאה. לחצו על F10 עד אשר תתבצע שורת הפלט הראשונה וראו כי אכן נרשמה על מסך ריצת התוכנית שורת הפלט. לחיצה נוספת על F10 תגרום להמשך ריצת התוכנית אל השורה הבאה, שורת הקלט. מכאן תוכלו להמשיך את ריצת התוכנית רק לאחר שתזינו את הקלט המבוקש. הקלידו את המספר 100 במסך ריצת התוכנית והקישו Enter. כעת חזרה השליטה לתוכנית.

נתבונן במסך "Locals" הנפתח בצד שמאלי תחתון של המסך:

במסך "Locals" נוכל לצפות בערכי המשתנים בעת ריצת התוכנית. שימו לב לרגע המתואר בצילום: על פי השורה הצבועה בצהוב ניתן לדעת כי כבר נקרא הקלט 100 ממסך ריצת התוכנית, והושם במשתנה money. נתבונן במשתנה זה במסך "Locals" ונבחין כי אכן המשתנה קיבל את הערך 100. הערך צבוע בצבע אדום מכיוון ששורת התוכנית האחרונה שהתבצעה גרמה לשינוי ערך המשתנה.



השורה הבאה בתוכנית אמורה לחשב את ערך המשתנה `smallBearSum`. לפני שנמשיך את ריצת התוכנית נחשוב מה ערכו של משתנה זה אמור להיות. כפי שחישבנו קודם, כל דוב יקבל 33 ₪ ולכן זה צריך להיות ערכו החדש של המשתנה. נמשיך את ריצת התוכנית על ידי הקשה נוספת על המקש F10. התבוננו במסך "Locals" וראו כי המשתנה `smallBearSum` שינה כעת את ערכו מ-0 ל-33, כצפוי. אם כך, עד עתה התוכנית עבדה באופן תקין.

השורה הבאה אמורה לחשב את ערך המשתנה `sumForHoney`. שוב, נחשוב מה אמור להיות ערכו של משתנה זה. כפי שחישבנו קודם, שארית החלוקה של 33 ב-4 היא 1, ולכן זו התוצאה המבוקשת. הקשה נוספת על המקש F10 תמשיך את ריצת התוכנית לשורה הבאה, וערכו של המשתנה `sumForHoney` משתנה לערך 8, ולא לערך הצפוי. אם כך, בשורה זו ישנה שגיאה.

נתבונן בשורת הקוד הבעייתית:

```
sumForHoney = smallBearSum / 4;
```

כוונתנו הייתה לחשב את **שארית החילוק** של `smallBearSum` ב-4, ובמקום זאת חישבנו את **תוצאת החילוק** של `smallBearSum` ב-4. שגיאה זו אירעה כיוון שהשתמשנו בסימן / במקום בסימן %.

נתקן את השגיאה בקוד התוכנית כך שהשורה תיראה כעת כך:

```
sumForHoney = smallBearSum % 4;
```

לאחר שתיקנו את השגיאה, נרצה לבדוק האם כעת התוכנית נכונה ומציגה את הפלט הנכון.

נוכל לבצע זאת באחת משתי הדרכים הבאות:

1. נעצור את ריצת התוכנית (על ידי Stop Debugging שבתפריט Debug, או על ידי הקשה על המקשים shift ו-F5 בו זמנית), ולאחר מכן נריץ את התוכנית שוב, לאחר השינוי, מההתחלה, על ידי ריצה רגילה של התוכנית, או על ידי ריצה תוך כדי ניפוי שגיאות.

2. נגרום לשורה שתיקנו להתבצע שוב: משמאל לשורות הקוד הצבוע בצהוב (השורה הבאה לביצוע) נבחין בחץ צהוב. נגרור את החץ הצהוב חזרה לשורת הקוד המתוקנת על ידי העכבר, כך ששורה זו תתבצע שוב מחדש, ונמשיך את ריצת התוכנית על ידי המקש F10 עד לסיומה.

נחזור על תהליך ניפוי השגיאות עבור כל שורה בתוכנית, ונבצע את המעקב עד אשר נהיה בטוחים כי התוכנית מספקת פלט נכון עבור כל קלט אפשרי.

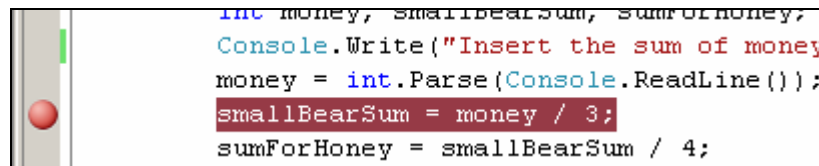
**שימו ♥:** בשלב זה שימוש במקש F11 יהיה זהה עבורנו לשימוש במקש F10. בהמשך לימודינו, כאשר נרצה להיכנס ולבדוק פעולות בקוד שנכתבו על ידינו, נשתמש במקש F11, ואילו מעבר על פניה מבלי להיכנס ולבדוק אותן יבוצע על ידי המקש F10.

## עבודה מתקדמת עם מנפה השגיאות

בהמשך לימודינו נכתוב תוכנית ארוכות, ולא נרצה לעבור על כל שורות הקוד בחיפושנו אחר שגיאה. לכן, אפשרות נוספת לעבודה עם ה-Debugger היא על ידי הרצת התוכנית באופן רציף, עד נקודה מסוימת, בה תעצור התוכנית את ריצתה. לשם כך נשתמש בנקודות עצירה, ה-Breakpoint.

למשל, נניח שאנו משוכנעים כי בתוכנית Bears קליטת נתון הקלט נעשתה כיאות ואין צורך לבדוק זאת. אם כך, אנו יכולים להציב נקודת עצירה בשורה שאחרי קליטת הקלט. לשם הצבת נקודת עצירה בשורה מסוימת נעמוד עם הסמן על השורה המבוקשת, ונבחר את Toggle Breakpoint שבתפריט Debug, או שנקיש על המקש F9. דרך נוספת להצבת נקודת עצירה היא על ידי לחיצה על המקש השמאלי של העכבר בשטח האפור שמשמאל לשורת הקוד המבוקשת.

כתוצאה מכך תופיע נקודה אדומה משמאל לשורת הקוד המבוקשת, והשורה תיצבע באדום.



```
int money, smallBearSum, sumOfMoney;
Console.WriteLine("Insert the sum of money");
money = int.Parse(Console.ReadLine());
smallBearSum = money / 3;
sumForHoney = smallBearSum / 4;
```

כעת נריץ את התוכנית, אך לא באופן הרגיל, אלא במצב ניפוי שגיאות, על ידי בחירת Start Debugging בתפריט Debug, או על ידי הקשה על המקש F5.

התוכנית תתחיל את ריצתה, תדפיס את שורת הפלט, תעצור לבקש קלט, נקליד 100 ו-Enter, וכעת, כאשר התוכנית תגיע לשורה המסומנת על ידי נקודת-עצירה, תפסיק את ריצתה ותעצור במצב Debugging המוכר לנו. ממצב זה, כפי שכבר ראינו, ניתן להמשיך להריץ שורה אחר שורה על ידי

המקש F10. אפשרות נוספת היא להוסיף נקודות-עצירה נוספות לאורך התוכנית ולרוץ מאחת לשנייה על ידי המקש F5. בכל הקשה על F5 התוכנית תרוץ מהמקום האחרון בו היא עצרה ועד נקודת-העצירה הבאה. אם לא תמצא נקודת-עצירה נוספת, תרוץ התוכנית עד לסיומה.

### טבלת מקשי קיצור שימושיים לפעולות בסביבת העבודה:

מקש קיצור	הסבר	פעולה
F6	הידור התוכנית	Build Solution
Ctrl + F5	הרצת התוכנית	Start Without Debugging
F5	הרצת תוכנית עד נקודת עצירה	Start Debugging
F10	קידום הרצת התוכנית בשורה אחת מבלי להיכנס לפעולות בקוד	Step Over
F11	קידום הרצת התוכנית בשורה אחת כך שניכנס לפעולות בקוד שנכתבו על ידינו	Step Into
F9 או מקש שמאלי של העכבר	הצבה או הסרה של נקודת עצירה על השורה בה נמצא הסמן	Toggle Breakpoint
Ctrl+E, D	הזחת התוכנית (אינדנטציה)	Format Document

# אינדקס

הערכים באינדקס שלפניכם מפנים את הקריאה לשני הכרכים של יסודות. הערכים המפנים ליסודות 2 מסומנים באות ב.

## סימנים

196 !

100 =!

70 %

113 &&

70 /

26 { }

118 ||

157 ++

100 >

100 =>

36 , 35 =

100 ==

100 <

100 =<

100 ≠

[ ] – פנייה לתו במחרוזת 11 ב, 22 ב

## א

או 111, 116, 118

אורך

מחרוזת 3 ב, 20 ב

מערך 25 ב, 27 ב

אינדקס של מערך 26 ב, 30 ב

אלגוריתם 13

אלכסון משני 123 ב

אלכסון ראשי 123 ב

אקראי

הגרלת מספר 85

אתחול 38

אתחול מערך 27 ב, 30 ב

## ב

בחירה ממצה של דוגמאות קלט 147

ביטוי בוליאני 98, 114, 118

ביצוע-חוזר 18, 155

ביצוע-חוזר-בתנאי 174

ביצוע-חוזר מספר פעמים ידוע מראש 156

זקיף 174, 178



לולאה אינסופית 187, 189  
 לולאת for 157  
 לולאת while 176  
 קינון 198, 200  
 ביצוע-מותנה 16, 95, 99  
 if 99  
 ביטוי בוליאני 98, 114, 118  
 הוראת שרשרת לביצוע-בתנאי 129  
 קינון של הוראה לביצוע-בתנאי 122  
 קשרים לוגיים  
 או 111, 116, 118  
 וגם 111, 112, 114  
 לא (not) 196  
 תנאי מורכב 111  
 בניית מספר 80, 93, 211

## ג

וגם 111, 112, 114

## ד

דוגמאות קלט מייצגות 147

## ה

הגרלת מספר 85  
 הוראת בחירה (switch) 99, 133 ב  
 הוראת פלט 25  
 הוראת קלט 29  
 הזזה מעגלית 60, 62  
 החזרת ערך 66, 67, 69 ב  
 המרת אותיות גדולות לקטנות 81, 15 ב, 22 ב  
 המרת אותיות קטנות לגדולות 22 ב  
 המרת טיפוסים 49, 75, 83  
 הערות 27  
 הפניה 3 ב, 94 ב, 105 ב  
 הפניה המועברת כפרמטר 110 ב  
 הפניה עצמית (this) 72 ב, 74 ב  
 הקצאת מקום בזיכרון 1 ב  
 הקצאת מקום עבור מערך 26 ב, 30 ב  
 הרשאות גישה  
 פרטית 64 ב  
 ציבורית 64 ב  
 השוואה – אופרטורים 100  
 השוואת מחרוזות 7 ב, 22 ב  
 השמה 30, 33

## ז

זיכרון 3  
זמן ביצוע של אלגוריתם 213  
זקיף 174, 178

## ח

חומרה 2, 8  
חזקה 67  
חיפוש בינרי 131, 214 ב  
חיפוש תו במחרוזת 11 ב  
חיפוש תת-מחרוזת 22 ב, 23 ב  
חלוקה בשלמים 68  
שארית 69, 70  
חריגה מגבולות המערך 41 ב

## ט

טבלת אמת  
או 118  
וגם 114  
לא (not) 196  
טבלת מעקב 39  
טיפוס 29, 46  
טיפוס בוליאני 191  
טיפוס המחרוזת 1 ב  
טיפוס ממשי 47, 48  
טיפוס שלם 29  
טיפוס תווי 80

## י

יחידת עיבוד מרכזית 3  
יעילות  
יעילות זמן 213  
יעילות מקום 50 ב, 52 ב  
יצירת עצם 85, 1 ב, 67 ב

## ל

לא (not) 196  
לולאות 18, 155  
ביצוע-חוזר-בתנאי 174  
ביצוע-חוזר מספר פעמים ידוע מראש 156  
זקיף 174, 178  
לולאה אינסופית 187, 189  
לולאת for 157  
לולאת while 176  
קינון 198, 200

## מ

מהדר 6

מונה 162, 203, 163 ב

מחלקה 26, 1 ב, 63 ב

הגדרת מחלקה 63 ב, 68 ב

הפניה עצמית (this) 72 ב, 74 ב

יצירת עצם 85, 1 ב, 67 ב

מחלקה מתמטית 67

מחלקת שירות 96 ב, 179 ב

עצם 1 ב, 63 ב

פעולה בונה 70 ב, 71 ב, 74 ב

פעולות גישה 75 ב, 79 ב, 84 ב

פעולות סטטיות 96 ב, 179 ב

פעולות של עצם 63 ב, 69 ב

מחרוזת 1 ב

אורך 3 ב

המרת אותיות גדולות לקטנות 81, 15 ב, 22 ב

המרת אותיות קטנות לגדולות 22 ב

השוואת מחרוזות 7 ב, 22 ב

השמה במחרוזות 12 ב

חיפוש תו במחרוזת 11 ב

חיפוש תת-מחרוזת 22 ב, 23 ב

מיקום תו במחרוזת 4 ב, 21 ב

פנייה לתו במחרוזת 4 ב

פעולות על מחרוזות 5 ב, 21 ב

שרשור מחרוזות 9 ב

מיון בועות 146 ב, 162 ב

מיון בחירה 136 ב

מיון הכנסה 139 ב, 143 ב

מיזוג 147 ב

מינימום 67

מציאת מינימום בסדרה 169, 208, 47 ב

מציאת מיקום המינימום בסדרה 172, 209, 47 ב

מיקום תו במחרוזת 4 ב, 21 ב

ממוצע 47, 61, 26 ב

מספר אקראי 85

מספר ממשי 47

מספר שלם 29

מעבר על זוגות סמוכים בסדרה 126 ב

מערך 25 ב, 26 ב

אורך 27 ב, 30 ב

איברי מערך 27 ב

אינדקס של מערך 26 ב, 30 ב

אתחול מערך 27 ב, 30 ב  
 החזרת עותק של מערך 109 ב  
 הקצאת מקום עבור מערך 26 ב, 30 ב  
 חריגה מגבולות המערך 41 ב  
 יחסים בין מערכים 170 ב  
 מערך כערך החזרה מפעולה 107 ב  
 מערך כתכונה 85 ב, 159 ב  
 מערך מונים 46 ב, 59 ב, 166 ב  
 מערך צוברים 50 ב, 60 ב  
 מערך של עצמים 90 ב, 94 ב, 159 ב  
 מציין של מערך 26 ב, 30 ב  
 סריקה של מערך 59 ב  
 מערך דו-ממדי 115 ב  
 אורך 116 ב  
 אלכסון משני 123 ב  
 אלכסון ראשי 123 ב  
 גודל 116 ב  
 מערך דו-ממדי ריבועי 123 ב  
 סריקת מערך דו-ממדי 115 ב  
 מציינים 26 ב, 30 ב  
 מקסימום 67  
 מציאת מיקום המקסימום בסדרה 172, 209  
 מציאת מקסימום בסדרה 143, 169, 208, 163 ב  
 משתנים 28  
 טיפוס בוליאני 191  
 טיפוס המחרוזת 1 ב  
 טיפוס ממשי 47, 48  
 טיפוס שלם 29  
 טיפוס תווי 80  
 משתנה מקומי (לוקאלי) 88 ב

## נ

ניפוי שגיאות 7  
 נכונות אלגוריתם 147

## ס

סריקת מערך דו-ממדי 115 ב  
 סריקת מערך חד-ממדי 59 ב

## ע

עיגול מספר ממשי 67  
 עצם 1 ב, 63 ב  
 העברת עצם כפרמטר לפעולה 101 ב, 104 ב  
 הפניה 3 ב, 94 ב, 105 ב  
 הפניה המועברת כפרמטר 110 ב

הפניה עצמית (this) 72 ב, 74 ב  
יצירת עצם 85, 1 ב, 67 ב  
פעולות של עצם 63 ב, 68 ב  
תכונות של עצם 63 ב, 69 ב  
ערך המוחזר מפעולה 67, 66 ב, 69 ב  
ערך מוחלט 67

## פ

פירוק מספר דו ספרתי לספרותיו 77, 92  
פירוק מספר לספרותיו 183, 210  
פיתוח אלגוריתם בשלבים 63  
פלט 1  
הוראת פלט 25  
פלינדרום 95  
פסאודו-קוד 14  
פעולות  
פעולה בונה 70 ב, 71 ב, 74 ב  
פעולה ראשית 26  
פעולות גישה 75 ב, 79 ב, 84 ב  
פעולות סטטיות 96 ב, 179 ב  
פעולות על מחרוזות 5 ב, 21 ב  
פעולות של עצם 63 ב, 69 ב  
פרטי (private) 64 ב  
פרמטר 67, 5 ב  
העברת עצם כפרמטר לפעולה 101 ב, 104 ב  
הפניה המועברת כפרמטר 110 ב

## צ

צובר 162, 206  
ציבורי (public) 64 ב

## ק

קבוע 51  
קינון  
הוראה לביצוע-בתנאי 122  
הוראה לביצוע-חוזר 198, 200  
קלט 1  
הוראת קלט 29

## ש

שארית חלוקה בשלמים 69, 70  
שגיאות  
ניפוי שגיאות 7  
נכונות אלגוריתם 147  
שורש ריבועי 65, 67  
שילוב תבניות 163 ב

שלילה (not) 196  
שפת מכונה 5  
שפת תכנות 5  
שרשור מחרוזות 9 ב

## ת

תבניות 21  
שילוב 163 ב  
תו 80  
תוכנה 2, 5, 9  
תחום 27  
תיעוד 27  
תכונות של עצם 63 ב, 68 ב  
תנאי 16, 95, 99  
if 99  
ביטוי בוליאני 98, 114, 118  
הוראת שרשרת לביצוע-בתנאי 129  
קינון של הוראה לביצוע-בתנאי 122  
קשרים לוגיים  
או 111, 116, 118  
וגם 111, 112, 114  
לא (not) 196  
תנאי מורכב 111  
תת משימות 64

## A

67 Abs  
114 ,112 ,111 and  
26 ב ,25 array  
68 ב ,63 attribute

## B

131 ב ,214 binary search  
193 ,191 bool  
162 ב ,146 bubble sort

## C

83 ,76 ,49 casting  
82 ,80 char  
63 ב ,1 ב ,26 class  
22 ב ,7 CompareTo  
6 compiler  
25 Console  
51 constant  
52 const  
74 ב ,71 ב ,70 constructor  
163 ב ,203 ,162 counter  
3 CPU

## D

7 debug  
70 division  
48 double

## E

22 ב ,7 Equals

## F

157 for

## I

99 ,96 ,16 if  
132 if else if  
30 ב ,26 index  
21 ב ,11 IndexOf  
143 ב ,139 insertion sort  
29 int

## L

30 ב ,27 Length (array)  
3 ב Length (string)

## **M**

26 Main

67 Math

67 Max

147 merge

69, 63 method

67 Min

70 modulus

## **N**

67, 1, 85 new

196 not

## **O**

63, 1 object

118, 126, 111 or

## **P**

67 Pow

64 private

64 public

## **R**

85 Random

69, 66 return

67 Round

## **S**

136 selection sort

67, 65 Sqrt

179, 96 static method

1 string

23, 22 Substring

99, 137 switch

25 System

## **T**

74, 72 this

22, 15 ToLower

22 ToUpper

47, 29 type

## **U**

25 using



## V

void 65 ב, 69 ב

## W

while 177

Write 29

WriteLine 25