C#

סדנת לימוד

קרא בהקדמה על קבצי קוד המקור

עורך ראשי: **זהר עמיהוד**

בודק מקצועי: **ארז קורן**

תרגום: **איציק מנשרוף**

עריכה ועיצוב: **ענבל אילני, שרה עמיהוד**

עיצוב עטיפה: **שרון רז**

- **טלפון: 09-9564716 1-700-7000-44**
- **פקס: 09-9571582**
- **דואר אלקטרוני: info@hod-ami.co.il**
- **אתר באינטרנט: www.hod-ami.co.il**

C#

סדנת לימוד

Bradley L. Jones

SAMS

הוצאת הוד-עמי
ספרי מחשבים

Teach Yourself C#.NET In 21 Days

By Bradley L. Jones

Editor: **Z. Amihud**

הוצאת הוד-עמי

לספרי מחשבים בע"מ

ת.ד. 6108 הרצליה 46160

טלפון: 09-9564716 פקס: 09-9571582

info@hod-ami.co.il

תרגום מורשה של ההוצאה המקורית באנגלית כמפורט למעלה, שהודפסה על ידי
SAMS Publishing, Copyright © 2002.

הודפס בישראל 2002

HOD-AMI Ltd.
P.O.B. 6108, Herzliya
ISRAEL, 2002

ISBN 965-361-325-1 מסת"ב

תוכן עניינים מקוצר

תוכן העניינים

נספחים ואינדקס באנגלית – התחלה מסוף הספר

מבוא

ברוך בואך אל **סדנת לימוד #C**. הספר מחולק ל-21 פרקים. לכל קורא דרוש זמן שונה לקרוא כל פרק, אך אם אתה מצפה ללמוד #C על ידי קריאה בלבד, צפויה לך אכזבה גדולה. לימוד של פרק אחד כולל קריאה, הקלדת קוד או חלקי קוד, פתרון שאלוני הבוחן והתנסות בפתרון התרגילים.

השאלות והתרגילים מיועדים לעזור לך לבסס את הבנתך בחומר הנלמד בכל פרק. בסיום קריאת כל פרק תוכל לענות על השאלות בבוחן. אם אינך יכול לעשות זאת, רצוי שתחזור ותעיין שוב בחלקים שונים של הפרק.

התרגילים נותנים לך הזדמנות ליישם את מה שלמדת. הם מתמקדים בדרך כלל בהבנת קוד, איתור בעיות שכיחות בקוד או בכתיבת קוד על סמך החומר שנלמד בפרק.

נספח א' מספק פתרונות לשאלוני הבוחן ולרוב התרגילים בספר. נסה לענות על השאלות, ולפתור את התרגילים בכוחות עצמך בטרם תבדוק את הפתרונות שבנספח.

הספר מכיל טיפים, הערות ואזהרות. הטיפים מספקים עצות שימושיות וההערות מספקות מידע נוסף שעשוי לעניין אותך. האזהרות מתריעות בפני בעיה או סוגיה שכיחה שאתה עשוי להיתקל בה.

למי מיועד הספר

ספר זה מניח שיש ברשותך מהדר #C וסביבת הריצה היא .NET.

הנחה נוספת היא שאתה מתכנת מתחיל. גם אם אין לך רקע בתכנות, תוכל ללמוד הרבה מהספר. עם זאת, אולי תגלה שבמספר תחומים קצב ההתקדמות שלך איטי יותר מכפי שרצית. אם יש לך רקע תכנות באחת מהשפות C++, Java או שפה מוכוונת עצמים אחרת, לא תתקשה ללמוד את החומר המוצג בספר. גם רקע בשפות כגון C ו- Visual Basic יספיק לך. מתכנת מנוסה ב- Visual Basic עם ידע בסיסי באחת מהשפות שצוינו לעיל עשוי אף לגלות שהספר קל ומהיר ללימוד.

הספר אינו מניח שאתה משתמש בסביבות הפיתוח #Visual C או Visual Studio.NET. באפשרותך להשתמש בכלים של Microsoft או בכלים אחרים. על כך תלמד בספר.

דרישות

סעיף זה מספק הדרכה כללית אודות הצרכים להפעלת C#:

- **מערכת הפעלה:** Windows XP Professional, או Windows 2000 (בגרסת Server או Professional), או Windows NT 4.0 עם Service Pack 6a (בגרסת Server או Professional). ההתקנה של Visual Studio.NET תטפל בהתקנת כל חבילות השירות הדרושות, תעדכן קבצי גישה לנתונים וגירסה 6 של Internet Explorer.
- **חומרה:** Pentium II 450Mhz ומעלה או מחשב שווה ערך, זיכרון בהתאם למערכת ההפעלה שבשימוש, כרטיס Video בעל יכולת של 800x600, ו-256 צבעים וכונן דיסק ובו לפחות 2GB. לצרכי התקנה דרוש כונן תקליטורים.

באשר לדוגמאות אחרות, ההנחה היא שיש לך גישה לשרת IIS-5 המותקן במחשב שמותקנת בו מערכת Visual Studio או מערכת NET Framework SDK. עדיף לוודא שבמחשב הפיתוח העיקרי שלך מותקן שרת Web מסוג IIS-5, כפי שב- Windows 2000 או Windows NT. דרישה עיקרית אחרת עבור הדוגמאות שבחציו השני של הספר היא גישה למסד נתונים, במיוחד ל- SQL Server 2000 או SQL Server 7. אם אין ברשותך SQL Server תוכל להשתמש ב- Microsoft Data Engine (MSDE), הפועל בדומה ל- SQL Server. אם מסדי נתונים אלה אינם ברשותך, מסד נתונים Access יספיק גם הוא, אך יהיה עליך לשנות כמה מהדוגמאות העוסקות במסדי נתונים כדי שיפעלו.

לפרטים נוספים ראה:

```
http://msdn.microsoft.com/vstudio/productinfo/sysreq.asp
```

לקבלת גרסת התנסות של Visual Studio.NET ניתן לפנות לחברת Microsoft ישראל, מחלקת שירות לקוחות, בטלפון: 09-9525400. קבלת גרסת התוכנה היא על פי תנאי חברת Microsoft ישראל בלבד ועל פי שיקול דעתה.

תמיכה מקוונת

אף אחד אינו מושלם ובכלל זה המחבר, היועץ המקצועי והעורך הראשי. אם תשלב זאת עם העובדה שמדובר בשפת תכנות, עליך גם לצפות לבעיות שתרצה למצוא להן תשובות. מלבד הבדיקות שנערכו לפני מסירת הספר לדפוס, הספר נבחן גם על ידי מספר אנשים ברמה הטכנית וברמת הפיתוח, הן במהדורה המקורית באנגלית והן במהדורה העברית. עם זאת, גם לאחר כל הבחינות והבדיקות ייתכן שתגלה שגיאות. תיקונים לספר (errata) תוכל למצוא באתר הוד-עמי בדף הספר:

```
www.hod-ami.co.il
```

עבדנו קשה כדי להפוך ספר זה לכלי יעיל ללימוד NET ותוספת רבת ערך לספריית הפיתוח שלך. אם תרצה שנקדיש יותר זמן (או פחות) לנושא מסוים, או אם עולות בדעתך הצעות כיצד לשפר ספר זה, נא צור קשר עם הוצאת הוד-עמי:

כתובת: support@hod-ami.co.il **Subject**: ציין את המספר 59363

אנו ננסה לשלב את הצעותיך בספרים עתידיים ובגירסה הבאה של ספר זה.

תוכל גם לעיין באתר הוצאת SAMS:

www.sampublishing.com

בנוסף לכך, המחבר יצר אתר המיועד לתמיכה בספר:

www.TeachYourselfCSharp.com

קבצי קוד המקור

את קוד המקור של כל הדוגמאות בספר תמצא באתר בכתובת: **www.hod-ami.co.il**, בקובץ EXE אותו תוריד/תפתח למחשב שלך. כמה מהדוגמאות ארוכות וכך תוכל לחסוך בהקלדה.

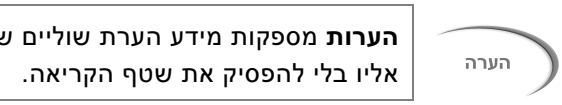
אתר את הספר בקטגוריה **תכנות** בתת-קטגוריה **.NET** או **C#**, לחץ על הקישור **לפרטים נוספים**. מתחת לתמונת הספר, לחץ על הקישור **קוד מקור** ותיפתח תיבת דו-שיח. רצוי לבחור בשמירת הקובץ לדיסק ורק אחר כך לפתוח אותו.

לאחר העתקת הקובץ לדיסק הקשיח, היכנס דרך סייר Windows (או המחשב שלי) למקום בו שמור הקובץ ולחץ עליו לחיצה כפולה. הקבצים ייפתחו לתיקיה בשם HodAmiBooks\59363.

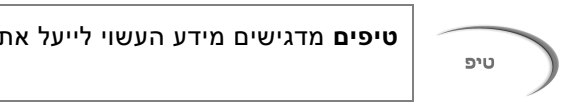
ספר זה מיועד ללימוד. עם זאת, תמצא שחלקי קוד מסוימים מתאימים לשימושים מעשיים. תוכל להשתמש בקוד המקור שבספר באופן חופשי כבסיס לתוכניות אחרות, ללמוד ממנו, או לכל דבר אחר. רכישת ספר זה מקנה לך את הזכות להשתמש בקוד בכל דרך שתמצא לנכון, עם יוצא מן הכלל אחד: אינך יכול להשתמש בו לכתיבת ספר הדרכה כלשהו עבור C#.

מוסכמות כתיבה בספר זה

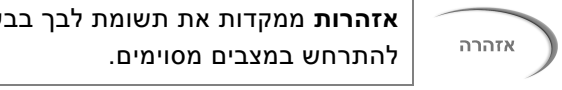
ספר זה משתמש בכמה מוסכמות כדי לסייע לך בהגדרת חשיבות המידע:



הערות מספקות מידע הערת שוליים שביכולתך לקרוא מיידית או לחזור אליו בלי להפסיק את שטף הקריאה.



טיפים מדגישים מידע העשוי לייעל את תכנות VB שלך.



אזהרות ממקדות את תשומת לבך בבעיות או בתופעות לוואי העלולות להתרחש במצבים מסוימים.

עשה	אל תעשה
תיבות עשה/אל תעשה מדגישות דרכי פעולה טובות שכדאי לך לאמץ!	ודרכי פעולה רעות שעליך לנטוש!

הסבר

סמלי הסבר מציינים הכוונה אודות קטעי קוד, כיצד ולמה נכתבו.

על המחבר

Bradley L. Jones משמש כמנהל אתר של מספר אתרים ידועים המיועדים למפתחים – כולל CodeGuru.com, Developer.com ו- Javascripts.com – וכן העורך הראשי של ערוץ EarthWeb. Bradley עסק בשפת C# יותר זמן ממרבית המפתחים, מכיון שהוזמן לחברת Microsoft לפני שחרור גרסת הביתא של התוכנה. הרקע שלו כולל ניסיון בפיתוח בשפות C, C++, PowerBuilder, SQL Server ובמספר טכנולוגיות וכלים אחרים. הוא גם חיבר ספרים שהפכו לרבי-מכר.

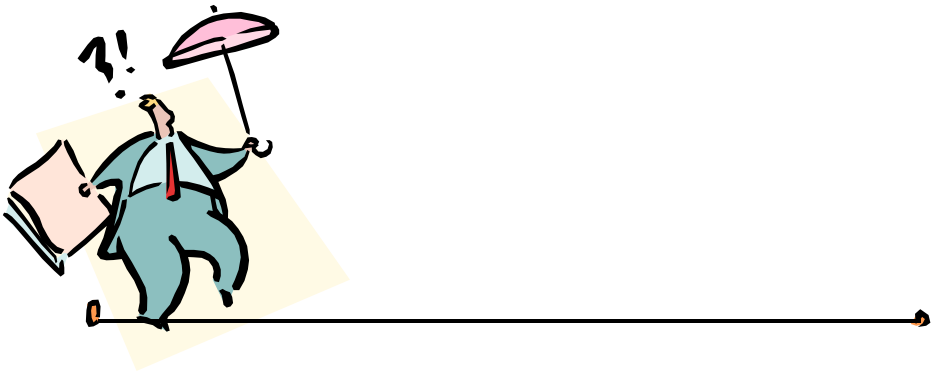
על היועץ המקצועי

ארז קוֹרֶן, בוגר יחידת המחשב בצה"ל - ממר"מ. מתמחה בעולם האינטרנט, e-Commerce וטכנולוגיות חדשניות. בעל ניסיון עשיר בפיתוח ב- C++, Java וב- C# ליישומים עתירי פלטפורמות ומשתמשים. כיום, מנהל צוות פיתוח בחברת Comverse, בתחום המסחר האלקטרוני.

הספרים שלהם יעץ **ארז קורן** בהוצאת **הוד-עמי**:

היכרות עם C# למתכנתים

Java/C++/Visual C++ למתכנתי C#



חלק 1

סקירה

ברוך הבא לספר **סדנת לימוד #C**. אם לא ברור לך מהו הידע המוקדם הנדרש כדי להפיק את המירב מספר זה, עליך לקרוא את ההקדמה. בהקדמה גם תמצא הסבר על מרכיבי הספר.

זהו הראשון מבין שלושת חלקי הספר אשר יעזרו לך לרכוש בסיס מוצק לכתיבת תוכניות #C. **בפרק 1** תערוך את היכרותך הראשונה עם תוכניות #C ותלמד איך יוצרים תוכנית #C.

בפרק 2 תלמד איך #C משתלבת ב- Microsoft .NET Framework (סביבת העבודה .NET). גם תלמד אודות העקרונות הבסיסיים של שפה מוכוונת עצמים.

בפרקים 3, 4 ו-5 תלמד על מושגי היסוד בתכנות אשר דרושים לכתיבת תוכניות #C, ובכלל זה אחסון נתונים וטיפול בהם ושליטה בזרימת תוכנית.

פרקים 6 ו-7 עוסקים במחלקות (classes) ובשיטות מחלקה (methods). מחלקות הן מושג מרכזי בתכנות מוכוון עצמים, ולכן גם מושג מרכזי בתכנות בשפת #C.

לקראת סוף חלק זה תדע רבים מהמושגים הבסיסיים הדרושים לתכנות ב-#C, ויהיו לך הכלים והידע לבנות תוכניות #C בסיסיות משלך.

חלק 1

פרק 1

התחלת העבודה

מכאן תתחיל את התהליך שבו תהיה למתכנת C# מיומן. בפרק זה תלמד:

- מדוע C# היא שפת תכנות נהדרת
- מהם השלבים בתהליך פיתוח התוכנית
- איך לכתוב, להדר ולבצע את תוכנית C# הראשונה שלך
- הודעות השגיאה המיוצרות על ידי המהדר
- סוגי היישומים שניתן ליצור ב- C#

מהי C# ?

שפת התכנות C# (קרא: C Sharp) היא שפה חדשה שפותחה על ידי Microsoft והוגשה לאיגוד יצרני המחשבים האירופאי (ECMA) לצורך תקינה. שפה חדשה זו נוצרה על ידי צוות מפתחים בניהולו של Anders Hejlsberg, מהנדס תוכנה בכיר בחברת Microsoft אשר פיתח שפות ומוצרים אחרים, ובהם Borland Turbo C++ ו- Borland Delphi. בפיתוח C# הוא לקח את החלקים הטובים בשפות הקיימות והוסיף להם שיפורים, כדי ליצור שפת תכנות טובה יותר.

#C היא שפת תכנות בעלת עוצמה רבה וגמישה בשימוש בה. ניתן להשתמש בה כמו בכל שפת תכנות אחרת ליצירת מיגוון יישומים. היא אינה מציבה גבולות, והפוטנציאל הקיים בה מוגבל אך ורק לדמיונך. #C כבר נוצלה בפרויקטים מגוונים, החל מאתרי אינטרנט דינמיים, דרך כלי פיתוח וכלה במהדרים (compilers).

#C נוצרה כשפת תכנות מוכוונת עצמים – OOP (**O**bject **O**riented **P**rogramming). קיימות שפות תכנות אחרות הכוללות תכונות מוכוונות עצמים, אך רק מעטות מהן מוכוונות עצמים באופן מושלם. בהמשך נעמוד על ההבדלים שבין #C לבין כמה משפות אחרות אלו. גם תלמד איזה סוגי יישומים ניתן ליצור. בפרק 2 תלמד מהי המשמעות של שימוש בשפה מוכוונת עצמים.

מדוע #C?

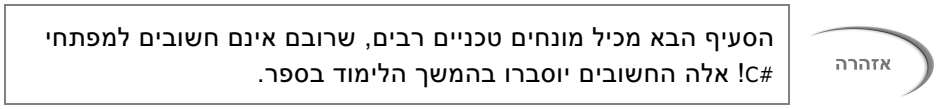
רבים האמינו שאין צורך בעוד שפת תכנות חדשה. האמינו ששפות קיימות כגון Java, C++, Microsoft Visual Basic ואחרות מציעות את כל הפונקציונליות הדרושה.

ל-#C יש דימיון לשפות C ו- C++, אך היא פותחה מן היסוד. Microsoft החלה ממה שפועל טוב ב-C ו- C++ והוסיפה תכונות חדשות, שהופכות את השימוש בשפה החדשה לקל יותר. רבות מהתכונות דומות למה שניתן למצוא ב-Java. בסופו של דבר, לחברת Microsoft היו מספר יעדים כאשר בנתה את השפה. ניתן לסכם אותם כך:

- #C היא פשוטה.
- #C היא מודרנית.
- #C היא מוכוונת עצמים.

קיימות סיבות אחרות להשתמש ב-#C בנוסף לאלו שציינה Microsoft:

- #C היא חזקה וגמישה.
- #C היא שפה תמציתית.
- #C היא מודולרית.
- #C תהיה פופולרית.

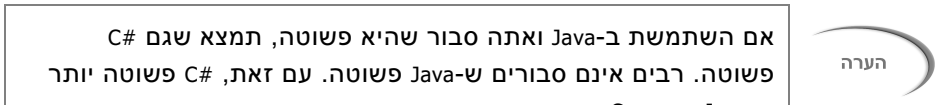


הסעיף הבא מכיל מונחים טכניים רבים, שרובם אינם חשובים למפתחי #C! אלה החשובים יוסברו בהמשך הלימוד בספר.

#C היא פשוטה

#C מסלקת כמה מהסיבוכים והמלכודות בשפות כמו Java ו- ++C, ובכלל זה שימוש במאקרו, templates, הורשה מרובה ומחלקות בסיס וירטואליות. תחומים אלה גורמים לבלבול או לבעיות פוטנציאליות עבור מתכנתי ++C. אם #C היא השפה הראשונה שאתה לומד, היה רגוע – אינך צריך לבזבז זמן על לימוד נושאים אלה!

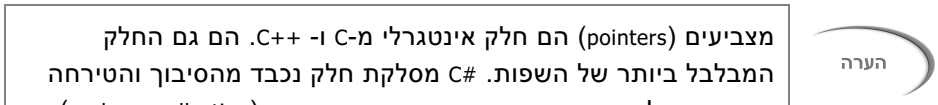
#C היא שפה פשוטה כי היא מבוססת על C ועל ++C. אם אתה מכיר שפות אלו – או אפילו Java – #C תיראה לך מוכרת למדי. משפטים, ביטויים, אופרטורים ופונקציות אחרות נלקחים ישירות משפות C ו- ++C, אך מיקצה שיפורים הופך את השפה לפשוטה יותר. כמה מהשיפורים כוללים סילוק כפילויות. תחומים אחרים של שיפור כוללים שינויי תחביר נוספים. לדוגמה, ב- ++C קיימים שלושה אופרטורים לעבודה עם חברים (members): **::**, **.** ו- **->**. הידיעה מתי להשתמש בכל אחד משלושת הסימנים האלה עשויה להיות מבלבלת מאוד ב- ++C. כל האופרטורים האלה מוחלפים ב-#C בסימן אחד – אופרטור "הנקודה" (.). עבור מתכנתים חדשים, תכונה זו ועוד רבות אחרות מסלקות בלבול רב ומהווים הקלה רבה.



אם השתמשת ב-Java ואתה סבור שהיא פשוטה, תמצא שגם #C פשוטה. רבים אינם סבורים ש-Java פשוטה. עם זאת, #C פשוטה יותר מ-Java ומ- ++C.

#C היא מודרנית

מה הופך שפה למודרנית? תכונות כמו טיפול בחריגים (exception handling), איסוף אשפה (garbage collection), סוגי נתונים הניתנים להרחבה (extensible data types) ואבטחת קוד (code security) הן תכונות שמצפים למצוא בשפה מודרנית. #C מכילה את כל אלה. אם אתה עושה את צעדיך הראשונים בתכנות, תכונות אלו אולי נשמעות לך מסובכות, אך לקראת סוף פרק 21 כבר תבין איך כולן ישימות בתוכניות #C שלך!



מצביעים (pointers) הם חלק אינטגרלי מ-C ו- ++C. הם גם החלק המבלבל ביותר של השפות. #C מסלקת חלק נכבד מהסיבוך והטירחה הנגרמים על ידי מצביעים. איסוף אשפה אוטומטי (garbage collection) ובטיחות סוג (type safety) הם חלק אינטגרלי מהשפה. אם המושגים **איסוף אשפה** ו**בטיחות סוג** אינם מוכרים לך, אל דאגה. כל אלה יוסברו בפרקים שבהמשך.

C# היא מוכוונת עצמים (Object-Oriented)

תכונות המפתח של שפה מוכוונת עצמים הם כימוס, הורשה ורב-צורתיות. C# תומכת בכל אלה. **כימוס** (encapsulation) הוא אריזת פונקציונליות בחבילה אחת; **הורשה** (inheritance) היא דרך מובנית להרחבת קוד ופונקציונליות קיימים לתוך תוכניות וחבילות חדשות; **רב-צורתיות** (polymorphism) היא היכולת לעשות התאמה למה שצריך להתבצע. הסברים מפורטים על כל אחד ממושגים אלה, ותיאור מפורט יותר של שפות מוכוונות עצמים תמצא בפרק 2. בנוסף לכך, נושאים אלה נסקרים ביתר הרחבה במהלך הלימוד בספר.

C# היא חזקה וגמישה

כפי שציינתי, עם C# אתה מוגבל רק על ידי דמיונך. השפה אינה מציבה מגבלות כלשהן. ניתן להשתמש בה בפרויקטים מגוונים מאוד, כגון יצירת מעבדי תמלילים, גרפיקה, גיליונות אלקטרוניים ואפילו עבור בניית מהדרים לשפות אחרות.

C# היא שפה של מעט מילים

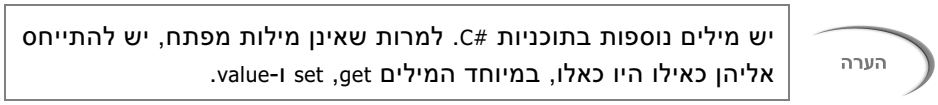
C# היא שפה המשתמשת במספר מועט של מילים. היא מכילה רק צרור מונחים הנקראים **מילות מפתח** (keywords), המשמשות כבסיס עליו נבנית הפונקציונליות שלה. טבלה 1.1 מפרטת את מילות המפתח של C#. רובן משמשות לתיאור מידע. אולי תחשוב ששפה המכילה יותר מילות מפתח תהיה בעלת עוצמה רבה יותר, אך אין זה נכון. כאשר תכתוב תוכניות ב-C#, תיווכח שמילים אלו מספיקות כדי לבצע כל משימה.

טבלה 1.1 מילות המפתח של C#

abstract	as	base	bool	break
byte	case	catch	char	checked
class	const	continue	decimal	default
delegate	do	double	else	enum
event	explicit	extern	false	finally
fixed	float	for	foreach	goto
if	implicit	in	int	interface
internal	is	lock	long	namespace
new	null	object	operator	out
override	params	private	protected	public

טבלה 1.1 המשך

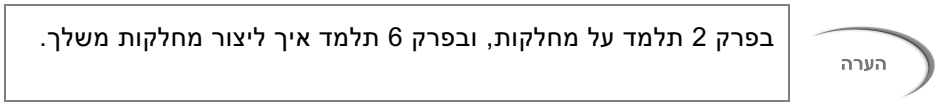
readonly	ref	return	sbyte	sealed
short	sizeof	stackalloc	static	string
struct	switch	this	throw	true
try	typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual	void
while				



יש מילים נוספות בתוכניות #C. למרות שאינן מילות מפתח, יש להתייחס אליהן כאילו היו כאלו, במיוחד המילים get, set ו-value.

#C היא מודולרית

קוד #C יכול (וצריך) להיכתב ביחידות הנקראות **מחלקות** (classes). מחלקות מכילות שגרות הנקראות **שיטות חברות** (member methods). מחלקות ושיטות ניתנות לניצול חוזר בתוכניות ויישומים אחרים. על ידי העברת קטעי מידע למחלקות ושיטות ניתן ליצור קוד מועיל לשימוש חוזר.



בפרק 2 תלמד על מחלקות, ובפרק 6 תלמד איך ליצור מחלקות משלך.

#C תהיה פופולרית

#C היא אחת משפות התכנות החדישות ביותר. בזמן שספר זה נכתב, לא ידועה מידת התפוצה של #C, אך קיימים סיכויים טובים לכך ששפה זו תהפוך לפופולרית מאוד. אחת מסיבות המפתח היא Microsoft והעתיד של NET. החברה רוצה ש-#C תהיה פופולרית. למרות שחברה אינה יכולה לכפות את תפוצת המוצר והשימוש בו, היא יכולה לעזור לכך.

ל-#C יש סיכויים טובים להצלחה, מכיון ש- Microsoft עצמה משתמשת ב-#C. רבים ממוצריה כבר מכילים חלקים שנכתבו מחדש ב-#C. כך, באמצעות השימוש ב-#C, Microsoft מוודאת ותומכת בכך שיכולות השפה אכן יעמדו בדרישות המתכנתים.

NET. של Microsoft היא סיבה אחרת מדוע יש ל-#C סיכוי להצליח. NET. מהווה שינוי בדרך שבה נעשים תהליכי הפיתוח של יישומים. למרות שמבחינה וירטואלית ניתן להשתמש בכל שפת תכנות בסביבת העבודה NET., ברור לנו עתה ש-#C היא השפה המועדפת. הפרק הבא מסביר את נקודות הזכות של NET..

#C תהיה פופולרית גם בגלל כל התכונות שהוזכרו כבר. היא שפה פשוטה, מוכוונת עצמים, גמישה ותמציתית.

#C מול שפות תכנות אחרות

אולי כבר שמעת על Visual Basic, ++C ו-Java או תכנתת בשפות אלו, ואתה תוהה מהם ההבדלים בין #C ובינן. אולי אתה גם תוהה האם עליך ללמוד אחת משפות אלו במקום #C.

השאלות השכיחות ביותר בפורומים של קבוצות דיון באינטרנט הנוגעות ל-#C הן:

- האם #C אינה אלא רק חיקוי של Java?
- מה ההבדל בין #C ו- ++C ?
- מה אני צריך ללמוד, Visual Basic.NET או #C ?

Microsoft טוענת ש-#C מעניקה את העוצמה של ++C יחד עם הקלות של Visual Basic. #C אכן מעניקה עוצמה רבה, אך האם היא קלה כמו Visual Basic? ובכן, היא אולי לא קלה כמו Visual Basic 6, אך היא קלה כמו Visual Basic.NET (גירסה 7), אשר נכתבה מהיסוד. התוצאה הסופית היא שהתכנות ב-Visual Basic.NET אינו קל יותר מתכנות ב-#C. למעשה, באמצעות #C ניתן לכתוב תוכניות רבות בפחות קוד.

למרות ש-#C מסירה כמה מתכונות ++C שגרמו למתכנתים כאב ראש גדול, לא ממש אבדו העוצמה והיכולת של המקור. ב-#C ניתן להימנע בקלות מכמה משגיאות התוכנה שקל לייצר ב-++C. הדבר יכול לחסוך שעות ואפילו ימים בכתיבת תוכניות. תוכל להבין יותר אודות ההבדלים לעומת ++C במהלך סקירת הנושאים בספר.

שפה אחרת אשר קיבלה תשומת לב רבה היא Java. כמו ++C ו-#C, Java מבוססת על C. אם תחליט ללמוד Java מאוחר יותר, תמצא שחלק גדול ממה שלמדת על #C ניתן ליישם גם בשפה זו.

אולי גם שמעת על שפת התכנות C. אנשים רבים תוהים האם עליהם ללמוד C לפני שהם לומדים #C, ++C או Java. ובכן, לדעתי אין צורך ללמוד תחילה שפת C.

וכעת, נפסיק עם הסיבות וההסברים. קרוב לוודאי שרכשת ספר זה כדי שתוכל ללמוד איך להשתמש בשפת #C לפיתוח תוכניות משלך. הסעיפים הבאים סוקרים את השלבים הכרוכים בפיתוח תוכנית. לאחר מכן תעבור על השלבים לכתיבת תוכנית פשוטה מראשיתה ועד סופה.

הכנות לכתיבת תוכניות

כאשר אתה פותר בעיה, עליך לנקוט בצעדים מסוימים. תחילה עליך להגדיר את הבעיה. אם אינך יודע מה הבעיה, אינך יכול למצוא פתרון! לאחר שאתה יודע מהי הבעיה, אתה יכול לתכנן איך לתקן אותה. כאשר יש לך תוכנית, בדרך כלל תוכל לממש אותה. לאחר שהתוכנית מומשה, עליך לבחון את התוצאות כדי לראות אם הבעיה נפתרה. לוגיקה זו של בעיה ופתרונה ישימה בתחומים רבים אחרים, כולל תכנות.

כאשר יוצרים תוכנית ב-C# (או בכל שפה), עליך לנהוג על פי רצף דומה של שלבים:

1. קבע את היעד או את יעדי התוכנית.
2. קבע את השיטות שתרצה להשתמש בהן בכתיבת התוכנית.
3. כתוב את התוכנית לפתרון הבעיה.
4. בצע את התוכנית כדי לבחון את התוצאות.

דוגמה ליעד (ראה שלב 1) עשויה להיות כתיבת תוכנית מעבד תמלילים או מסד נתונים. יעד פשוט הרבה יותר הוא להציג את שמך על המסך, או להדפיס את רשימת החברים מתוך קובץ. אם אין לך יעד, לא תוכל לכתוב תוכנית.

השלב הבא הוא קביעת השיטות שתרצה להשתמש בהן בכתיבת התוכנית. האם אתה זקוק לתוכנית מחשב כדי לפתור את הבעיה? (אם הבעיה היא שאתה סובל מכאבי ראש, אף תוכנית מחשב לא תעזור לך... לך לרופא!) אחר איזה מידע יש לעקוב? באיזה נוסחאות צריך להשתמש? במהלך שלב זה עליך לנסות לקבוע מה יידרש ובאיזה סדר צריך לממש את הפתרון.

לדוגמה, נניח שהתבקשת לכתוב תוכנית לקביעת שטח עיגול. שלב 1 הושלם, מכיון שאתה יודע מהו היעד: לחשב שטח של עיגול נתון. בשלב 2 עליך לקבוע מה עליך לדעת כדי לחשב את השטח. בדוגמה זו, נניח שהמשתמש מספק לך את רדיוס העיגול. אם אתה יודע זאת, הרי שתוכל ליישם את הנוסחה πr^2 כדי לקבל את התשובה. כעת נמצאים ברשותך החלקים הנחוצים ואתה יכול להתקדם לשלבים 3 ו-4, הנקראים **מחזור פיתוח התוכנית**.

מחזור פיתוח תוכנית

מחזור פיתוח תוכנית (Program Development Cycle) הוא בעל שלבים אחדים. בשלב ראשון משתמשים בעורך טקסט ליצירת קובץ המכיל את קוד המקור של התוכנית. בשלב השני מהדרים (compile) את קוד המקור ליצירת קובץ ביניים הנקרא קובץ ביצוע (executable) או קובץ ספריה (library). בשלב השלישי מבצעים את התוכנית כדי לבדוק אם היא פועלת על פי התכנון המקורי.

יצירת קוד מקור (Source Code)

קוד מקור (source code) הוא סדרת משפטים או פקודות המורים למחשב לבצע את המשימות הדרושות. השלב הראשון במחזור פיתוח התוכנית הוא להקליד קוד מקור בעורך. לדוגמה, הנה שורת קוד מקור בשפת C#:

```
System.Console.WriteLine("Hello, Mom!");
```

משפט זה מורה למחשב להציג את ההודעה Hello, Mom! על גבי המסך (כעת אל תהיה מוטרד מאופן הפעולה של משפט זה).

השימוש בעורך (Editor)

עורך (editor) היא תוכנית המשמשת להקלדה ושמירה של קוד מקור. יש מספר עורכים מתאימים לשימוש עם C#. אחדים מהם נוצרו במיוחד עבור C# ואחרים לא.

Microsoft הוסיפה יכולות C# למוצר Visual Studio שלה הכולל את Visual C#. זהו העורך השולט הזמין כרגע. עם זאת, אם אין לך Visual Studio.NET עדיין, תוכל לתכנת ב-C# גם בלעדיו.

קיימים עורכים אחרים המתאימים לשימוש עם C#. כמו Visual Studio.NET, רבים מעורכים אלה מאפשרים לבצע את כל שלבי מחזור הפיתוח מתוך העורך (לא צריך לצאת ממנו). חשוב יותר, רובם מסוגלים לצבוע חלקי קוד שונים בצבעים שונים (קידוד צבע). תכונה זו מקלה מאוד על איתור שגיאות אפשריות. עורכים רבים כוללים גם מערכת עזרה חזקה, הכוללת תמיכה בכתיבת הקוד באמצעות מתן מידע על מה שעליך להקליד.

אם אין לך עורך C#, אל דאגה. רוב מערכות ההפעלה מכילות תוכניות שניתן להשתמש בהן כעורכים. במערכות ההפעלה Microsoft Windows ניתן להשתמש ב-Notepad או ב-WordPad. במערכת Linux או UNIX ניתן להשתמש ב-ed, ex, edit, emacs או vi.

רוב מעבדי התמלילים משתמשים בקודים מיוחדים כדי לעצב את המסמכים שלהם. תוכניות אחרות לא מסוגלות לקרוא נכון קודים אלה. מעבדי תמלילים רבים - כגון Word ו-WordPad (כתבן) מסוגלים לשמור קבצי מקור (source files) בפורמט של טקסט פשוט. כאשר רוצים לשמור קובץ של מעבד תמלילים כקובץ טקסט, בוחרים את אפשרות text בזמן השמירה.

הערה

לפניך רשימה חלקית של עורכים שניתן להשתמש בהם:

- **CodeWright** - עורך המספק תמיכה מיוחדת עבור ASP, XML, HTML, C#, Perl, Python ועוד. העורך נמצא באתר www.premia.com.
- **EditPlus** - זהו עורך טקסט מוכן לאינטרנט, עורך HTML ועורך למתכנת בסביבת Windows. למרות שעורך זה משמש תחליף טוב ל-NotePad, הוא גם מציע תכונות טובות רבות עבור המפתחים דפי אינטרנט ומתכנתים, כולל קידוד צבע של קוד. תמצא אותו בכתובת www.editplus.com
- **Jedit** - עורך עם קוד מקור פתוח (open source) עבור Java. עם זאת, ניתן להשתמש בו עבור C#. הוא כולל יכולת קידוד צבע של הקוד. נמצא באתר http://jedit.sourceforge.net.
- **Poorman IDE by Duncan Chen** - Poorman מספק עורך עם הארת תחביר עבור C# ו- Visual Basic.NET. הוא גם מאפשר להפעיל את המהדר וללכוד את פלט המסוף, כך שאין צורך לצאת מ- Poorman IDE. עורך זה נמצא בכתובת www.geocities.com/duncanchen/poormanide.htm
- **SharpDevelop by Mike Krüger** - עורך חופשי עבור פרויקטים של C# בפלטפורמת NET של Microsoft. זהו עורך קוד מקור פתוח (GPL), כך שניתן להוריד גם את קוד המקור וגם את קובץ ה-EXE מהכתובת www.icsharpcode.net
- **NoteTab** - עורך חופשי (גרסת Light) קטן עם מספר פעולות מתקדמות. עבודה עם מסמכים במקביל – www.notetab.com.

מתן שמות לקבצי מקור

כאשר אתה שומר קובץ מקור (source file), קבע לו שם המתאר מה התוכנית עושה. בנוסף לכך, בשמירת קבצי מקור של C#, קבע לקובץ את הסיומת cs. למרות שניתן לקבוע לקובץ מקור כל שם וסיומת, הצירוף cs מוכר כסיומת המתאימה לשימוש.

לדוגמה, קובץ תוכנית בשם NameList.cs עדיף על tgx001.cs.

ביצוע תוכנית #C

לפני שנדון בפירוט במחזור הפיתוח, חשוב להבין מעט אודות האופן שבו מתבצעת תוכנית C#. תוכניות C# שונות מתוכניות שניתן ליצור בשפות תכנות אחרות.

תוכניות C# מיועדות לביצוע בסביבת הריצה **CLR** (**C**ommon **L**anguage **R**untime). פירושו של דבר, שאם יוצרים קובץ ביצוע של C# ומנסים להפעילו במחשב שלא מותקנת בו סביבת הריצה CLR או סביבת ריצה תואמת, הקובץ לא יתבצע. **קובץ ביצוע** (executable) אומר שהתוכנית יכולה להתבצע, או "לרוץ" במחשב שלך.

היתרון בפיתוח תוכניות לסביבת זמן ריצה הוא בניידות. בשפות קודמות כגון C ו-C++, אם רצית לפתח תוכנית שתוכל להתבצע בפלטפורמות או מערכות הפעלה שונות, היית צריך להדר קבצי ביצוע שונים בהתאמה. לדוגמה, אם כתבת יישום C ורצית שיפעל במערכות הפעלה Linux ו-Windows, היית צריך ליצור שתי תוכניות ביצוע – אחת למערכת Linux ואחת למערכת Windows. ב-C# יוצרים רק תוכנית ביצוע אחת אשר תוכל להתבצע בכל אחת ממערכות הפעלה הללו.

אם רוצים שהתוכנית תתבצע מהר ככל האפשר, צריך ליצור קובץ ביצוע אמיתי. מחשב זקוק להוראות דיגיטליות, או בינאריות (binary) במתכונת הנקראת **שפת מכונה** (machine language). כלומר, חייבים לתרגם, או להמיר קוד מקור לשפת מכונה. תוכנית המבצעת עבודת המרה זו נקראת **מהדר** (compiler). המהדר מקבל קלט של קוד מקור, ומייצר קובץ דיסק המכיל את הוראות שפת המכונה המתאימות למשפטים בקוד המקור. עם תוכניות כמו C ו- C++, המהדר יוצר קובץ הניתן לביצוע ללא כל מאמץ נוסף.

ב-C# משתמשים במהדר שאינו מייצר שפת מכונה. במקום זאת המהדר יוצר קובץ **שפת ביניים** – **IL** (**I**ntermediate **L**anguage). מכיון שהמחשב אינו מבצע קובץ זה באופן ישיר, צריך שיקרה משהו נוסף להמרת הקובץ או הידור סופי של התוכנית עבור המחשב. **CLR** או סביבת זמן ריצה תואמת של C# מבצעת הידור סופי זה רק כאשר נדרש.

אחד הדברים הראשונים ש-CLR עושה עם קובץ הביניים IL הוא הידור סופי של התוכנית. בתהליך זה ה-CLR ממירה את הקוד מקובץ הקוד הנייד IL לשפה (שפת מכונה) שהמחשב יכול להבין ולבצע. CLR מהדרת למעשה רק אותם חלקים בתוכנית שבהם משתמשים באותו רגע. זהו תהליך החוסך זמן. בנוסף לכך, לאחר שחלק מקובץ IL זכה להידור אמיתי במערכת מסוימת, אין צורך להדרו שנית, כי אותו חלק מהודר של התוכנית נשמר ומשתמשים בו בפעם הבאה שחלק זה מתבצע.

הערה

בזמן הריצה צריך להדר את קובץ ה-IL, ולכן דרוש מעט יותר זמן לבצע תוכנית בפעם הראשונה, מאשר לביצוע קוד של שפה מהודרת במלואה כגון C++. לאחר הפעם הראשונה שתוכנית מבוצעת במלואה, הבדלי הזמן נעלמים, כי מעתה ואילך תנוצל הגירסה המהודרת במלואה.

הערה

ההידור "ברגע האחרון", או "כאשר נדרש", של תוכנית C# נקרא Just-In-Time compiling או בקיצור jitting.

הידור קוד מקור של C#

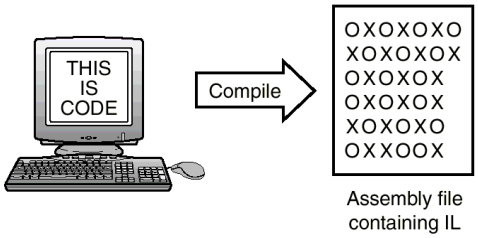
כדי ליצור קובץ ביניים IL משתמשים במהדר C# (C# compiler). בדרך כלל משתמשים בפקודה **csc** כדי להפעיל את המהדר, כשלאחריה מציינים את שם קובץ המקור. לדוגמה, כדי להדר קובץ מקור בשם radius.cs הקלד את השורה הבאה בשורת הפקודה (command line) מחלון הפקודה (command prompt):

```
csc radius.cs
```

אם אתה משתמש בסביבת פיתוח גרפית, ההידור פשוט אף יותר. ברוב הסביבות הגרפיות ניתן להדר תוכנית על ידי בחירת **סמל** (icon) המהדר או בחירת אפשרות מתאימה מהתפריט. לאחר הידור הקוד מבצעים את התוכנית על ידי בחירת סמל הביצוע או אפשרות מתאימה מהתפריט. עליך לקרוא את תיעוד המהדר שלך לקבלת מידע מוגדר אודות הידור וביצוע תוכניות.

לאחר הידור קובץ המקור מתקבל קובץ IL. אם תעיין ברשימת הקבצים בספריה שבה הידרת, אתה אמור למצוא קובץ חדש עם שם זהה לזה של קובץ המקור אך עם סיומת exe. (ולא cs.). קובץ זה, הנקרא **מכלול קוד** (assembly), הוא התוכנית "המהודרת" שלך. התוכנית מוכנה כעת לביצוע ב-CLR. קובץ מכלול הקוד מכיל את כל המידע הדרוש לסביבת הריצה כדי לבצע את התוכנית.

תרשים 1.1 מציג את ההתקדמות מקוד מקור לקובץ בר-ביצוע.



תרשים 1.1 קוד המקור של C# מתורגם לקוד שפת ביניים (IL) באמצעות המהדר

הערה

באופן כללי, מופקים שני דברים כתוכניות C#: קבצי ביצוע (executables) וספריות (libraries). בשני חלקי הספר הראשונים נתמקד בקבצי ביצוע שהם קבצי exe.. ניתן להשתמש ב-C# גם לסוגים אחרים של תכנות, כולל כתיבת תסריטי ASP בדפי ASP.NET. על ספריות תלמד בחלק השלישי.

סיום מחזור הפיתוח

לאחר שיש בידך התוכנית שהיא קובץ ביניים IL מהודר, תוכל לבצע אותה על ידי הקלדת שמה בשורת הפקודה, או כפי שאתה מבצע כל תוכנית אחרת.

אם בהרצת התוכנית אתה מקבל תוצאות שונות ממה שציפית לקבל, עליך לחזור לשלב הראשון של תהליך הפיתוח. אתה חייב לזהות מה גרם לבעיה ולתקן זאת בקוד המקור. אם שינית את קוד המקור, עליך להדר שוב את התוכנית כדי ליצור גירסה נכונה (עדכנית) של קובץ הביצוע. כך עליך להמשיך ולעקוב לפי מחזור זה עד שתקבל תוכנית המתבצעת כפי שרצית.

מחזור הפיתוח של #C

שלב 1 – השתמש בעורך לכתיבת קוד המקור. לקבצי מקור של #C קובעים בדרך כלל את הסיומת cs. (לדוגמה: a_program.cs, database.cs, וכן הלאה).

שלב 2 – הדר את התוכנית באמצעות מהדר #C. אם המהדר אינו מוצא שגיאות בתוכנית, הוא מייצר קובץ מכלול (assembly) עם הסיומת exe. או dll.. לדוגמה, myprog.cs מהודר ל-myprog.exe כברירת מחדל. אם המהדר מוצא שגיאות, הוא מדווח עליהן. עליך לחזור לשלב 1 כדי לבצע תיקונים בקוד המקור שלך.

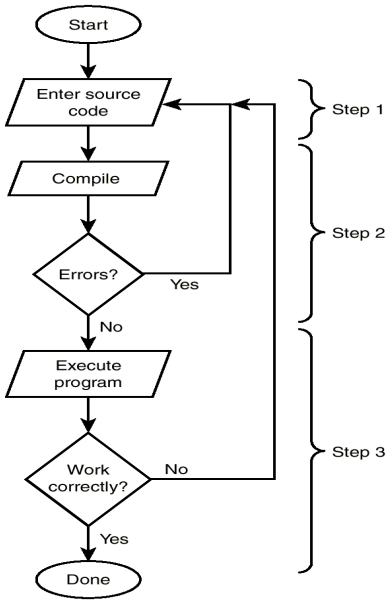
שלב 3 – בצע את התוכנית במערכת עם סביבת ריצה תואמת #C, כמו ה-CLR. עליך לבחון את ביצוע התוכנית כדי לקבוע אם היא מתפקדת בצורה תקינה. אם לא, החל שוב משלב 1 ובצע שינויים ותוספות בקוד המקור שלך.

תרשים 1.2 מציג את שלבי פיתוח התוכנית. עבור כל התוכניות חוץ מתוכניות פשוטות מאוד, אתה עשוי לעבור סדרת שלבים זו שוב ושוב לפני שתסיים את תוכניתך. אפילו המתכנתים המנוסים ביותר אינם יכולים לכתוב תוכנית שלמה נטולת שגיאות במהלך אחד בלבד! מכיון שתצטרך לעקוב אחר הלולאה של עריכה-הידור-בחינה פעמים רבות, חשוב שתהיה בקיא בכלים שאתה מפעיל: העורך, המהדר וסביבת זמן הריצה.

תוכנית #C הראשונה שלך

אולי אתה כבר משתוקק לנסות את תוכניתך הראשונה ב-#C. כדי להכיר את המהדר, היעזר בתוכנית הקצרה שבקוד 1.1. אולי לא תבין הכל בשלב זה, אך עליך לקבל את התחושה של תהליך כתיבה, הידור וביצוע תוכנית #C ממשית.

הדוגמה משתמשת בתוכנית הנקראת hello.cs, אשר אינה עושה דבר פרט להצגת המילים Hello, World! על המסך. זו תוכנית מקובלת מאוד להיכרות עם עולם התכנות. היא גם טובה עבורך למטרות לימוד. קוד המקור עבור hello.cs מופיע בקוד 1.1. כאשר אתה מקליד את הרשימה, אל תכלול את מספרי השורות והנקודתיים המופיעים בצד שמאל של הקוד.



תרשים 1.2 השלבים הכרוכים בפיתוח תוכנית C#

קוד 1.1 hello.cs

```
1:  class Hello
2:  {
3:    static void Main()
4:    {
5:      System.Console.WriteLine("Hello, World!");
6:    }
7:  }
```

עליך לוודא שהתקנת את המהדר כמצוין בהוראות ההתקנה המסופקות עם התוכנה. כאשר המהדר והעורך מוכנים לשימוש, פעל על פי השלבים בסעיף הבא.

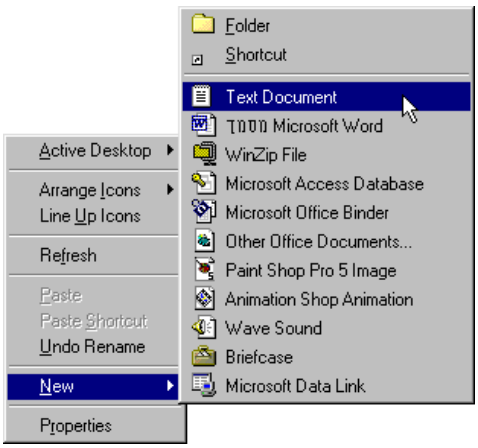
כתיבה והידור של hello.cs

כדי להקליד ולהדר את התוכנית hello.cs, עקוב אחר השלבים הבאים:

1. פתח את Windows Explorer (סייר Windows).
2. עבור לדיסק C.
3. צור תיקיה בשם TYCSharp בדיסק C (תוכל לבחור שם אחר).
4. עבור לתיקיה TYCSharp.
5. צור תיקיה בשם C1 (C עבור C# ו- 1 עבור פרק 1. תוכל לבחור שם אחר) בתוך התיקיה TYCSharp.
6. ב- Windows Explorer, בחר בתפריט Tools ⮜ Folder Options ⮜ View.
7. ודא שביטלת את האפשרות Hide File Extensions for known file types (הסתר סיומת קובץ עבור סוגי קבצים ידועים) – תיבת סימון ריקה.
8. לחץ OK (אישור).

אם כאשר אתה מנסה לשמור את הקוד שלך, בעזרת Notepad, כ- hello.cs, תקבל קובץ ששמו hello.cs.txt, יהיה עליך לחזור ולוודא שביטלת את אפשרות הסתרת סיומת הקובץ (סעיף 6, 7 ו-8).

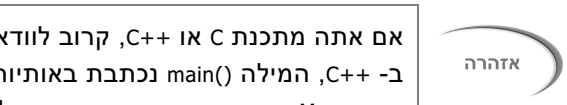
כדי ליצור את קובץ הטקסט החדש, לחץ לחיצה ימנית בתיקיה המתאימה (במקרה זה TYCSharp\C1), ובחר New ⮜ Text Document (ראה תרשים 1.3). פעולה זו יוצרת קובץ טקסט בשם **New Text Document.txt**. כדי לציין את תוכנו הסופי, שנה את שם הקובץ מ- **New Text Document.txt** ל- **hello.cs**.



תרשים 1.3 לחיצה ימנית בחלונית הקבצים שבסייר Windows תציג את התפריט שיספק לך דרך ישירה ליצירת קובץ טקסט ומסמכים רבים אחרים.

עם הקובץ בעל השם החדש והתיקיה הנכונה תוכל להתקדם ולכתוב את הקוד הממשי שלך. פתח את הקובץ ב-Notepad על ידי לחיצה ימנית ובחירה ב- Open With. בתיבת הדו-שיח המוצגת בחר ב-Notepad, לחץ OK והקובץ ייפתח. בנקודה זו הקובץ לא יכיל דבר:

1. הקלד את קוד המקור של hello.cs המוצג בקוד 1.1. אל תקליד את מספרי השורות ואת הנקודתיים. אלה משמשים רק להתייחסות בספר. הקש Enter בסוף כל שורה. ודא שאתה מקליד את הקוד באותיות רישיות או רגילות **בדיוק** כפי שהוא מופיע. #C היא שפה **הרגישה** לסוג האותיות (case sensitive), כך שאם תשנה את האותיות תקבל שגיאות. לדוגמה, A זה לא a.



אם אתה מתכנת C או ++C, קרוב לוודאי שתעשה שגיאה נפוצה. ב-C וגם ב- ++C, המילה ()main נכתבת באותיות קטנות. ב-#C, כותבים ()Main עם אות M רישית. אם תכתוב אות רגילה m – תקבל שגיאה.

2. שמור את קוד המקור בקובץ בשם hello.cs.
3. ודא שהקובץ hello.cs נשמר על ידי עיון ברשימת הקבצים בתיקיה.

אם אתה משתמש בסביבת פיתוח משולבת – IDE (**I**ntegrated **D**evelopment **E**nvironment), בחר בסמל המתאים, או באפשרות המתאימה בתפריט. בסיום תקין תקבל הודעה האומרת שלא היו שגיאות או אזהרות כלשהן. זהו סוף התהליך. אם אתה משתמש בשורת הפקודה, עבור לסעיף הבא.

4. בחר בתפריט Start ➢ Programs ➢ Microsoft Visual Studio.NET ➢ Visual Studio.NET Tools ➢ Visual Studio.NET Command Prompt.
5. בשורת הפקודה הקלד:

```
cd \tycsharp\c1
```

והקש Enter.

6. הדר את hello.cs. הקלד את הפקודה הבאה:

```
C:\TYCSharp\c1>csc hello.cs
```

תשובת המהדר:

```
Microsoft (R) Visual C# .NET Compiler version 7.00.9466
for Microsoft (R) .NET Framework version 1.0.3705
Copyright (C) Microsoft Corporation 2001. All rights reserved.
```

7. בדוק את הודעות המהדר. אם היתה שגיאה בהקלדת התוכנית, המהדר ילכוד אותה ויציג הודעת שגיאה. לדוגמה, אם הקלדת Consol במקום המילה Console, תקבל הודעת שגיאה כמו זו:

```
hello.cs(5,7): error CS0117: 'System' does not contain a
definition for 'Consol'
```

8. חזור לשלב 1 אם מוצגת שגיאה זו או כל שגיאה אחרת. פתח את הקובץ hello.cs בעורך. השווה את תוכן הקובץ לקוד 1.1, בצע בתשומת לב רבה כל תיקון נדרש, והמשך לשלב 2.

9. כדי לבצע (execute), או "להריץ" את hello.exe הקלד hello בשורת הפקודה.

```
C:\TYCSharp\c1>hello
```

ההודעה:

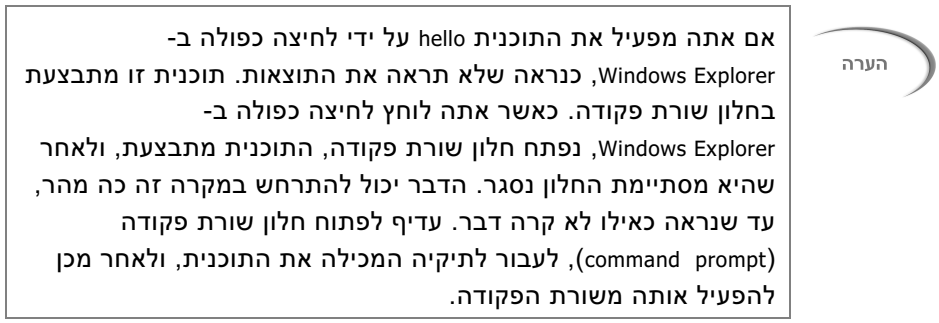
```
Hello, World!
```

תוצג על המסך.

תוכנית #C הראשונה שלך אמורה להיות מהודרת ומוכנה לביצוע. אם תציג את רשימת כל הקבצים בתיקיה (בעזרת הפקודה dir) המכילים את השם hello (עם כל סיומת), אתה אמור לראות את הרשימה הבאה:

hello.cs – קובץ קוד המקור שיצרת עם העורך.

hello.exe – קובץ הביצוע שנוצר לאחר הידור hello.cs.



אם אתה מפעיל את התוכנית hello על ידי לחיצה כפולה ב-Windows Explorer, כנראה שלא תראה את התוצאות. תוכנית זו מתבצעת בחלון שורת פקודה. כאשר אתה לוחץ לחיצה כפולה ב-Windows Explorer, נפתח חלון שורת פקודה, התוכנית מתבצעת, ולאחר שהיא מסתיימת החלון נסגר. הדבר יכול להתרחש במקרה זה כה מהר, עד שנראה כאילו לא קרה דבר. עדיף לפתוח חלון שורת פקודה (command prompt), לעבור לתיקיה המכילה את התוכנית, ולאחר מכן להפעיל אותה משורת הפקודה.

איחולי! זה עתה הקלדת, הידרת וביצעת את תוכנית #C הראשונה שלך. יש להודות ש- hello.cs היא תוכנית פשוטה שאינה עושה שום דבר מועיל, אך זו התחלה. למעשה, רוב המתכנתים המנוסים של ימינו החלו ללמוד בדיוק בדרך זו – על ידי הידור תוכנית המציגה "Hello, World!".

הבנת שגיאות הידור

שגיאת הידור מתרחשת כאשר המהדר מוצא בקוד המקור צירוף מילים או סימנים שלא ניתן להידור. שגיאת איות, שגיאת הקלדה, או כל דבר אחר שיכול לגרום למהדר לעצור. למרבה המזל, מהדר #C אינו עוצר סתם כך; הוא מודיע מדוע זה קרה והיכן גילה בעיה. כך קל יותר לאתר ולתקן שגיאות בקוד המקור.

ניתן להמחיש נקודה זו על ידי הקלדת שגיאה מכוונת בתוכנית hello.cs שהקלדת קודם. אם עבדת על דוגמה זו (וכך היית צריך לעשות), יש לך עותק של hello.cs בדיסק. באמצעות העורך העבר את הסמן לסוף שורה 5 ומחק את התו נקודה-פסיק (;) המסיים את השורה. התוכנית hello.cs נראית כעת כמוצג בקוד 1.2.

קוד 1.2 hello.cs עם שגיאה

```
1:  class Hello
2:  {
3:    static void Main()
4:    {
5:      System.Console.WriteLine("Hello, World!")
6:    }
7:  }
```

כעת שמור את הקובץ והדר אותו. עשה זאת על ידי הקלדת הפקודה עבור המהדר שלך. זכור, הפקודה לשורת הפקודה היא:

```
csc hello.cs
```

מכיון שהקלדת שגיאה בקוד המקור, ההידור אינו מסתיים כראוי והמהדר מציג לך הודעה בדומה לזאת:

```
hello.cs(5,48):  error CS1002:  ; expected
```

תוכל להבחין ששורה זו מורכבת משלושה קטעים:

hello.cs – שם הקובץ שבו נמצאה השגיאה.

(5,48): – מספר השורה והמיקום (העמודה) שבה המהדר זיהה שגיאה: שורה 5, תו 48 (החל מצד שמאל של השורה).

אתה עשוי לקבל מספר שונה מ-48 כי הדבר תלוי במיקום בשורה שבו התחלת להקליד את הפקודה הזו.

error CS1002: ; expected – תיאור השגיאה.

הודעה זו נותנת מידע מספק ואומרת שכאשר המהדר הגיע לתו ה-48 בשורה 5 של hello.cs הוא ציפה למצוא נקודה-פסיק, אך לא מצא תו זה. למרות שהמהדר "פיקח" מאוד בכל הנוגע לאיתור ומיפוי שגיאות, הוא איננו איינשטיין. אתה חייב לפרש את הודעות המהדר תוך ניצול הידע שלך בשפת C#, ולקבוע את המיקום בפועל של כל שגיאה מדווחת. בדרך כלל הן נמצאות בשורה שהמהדר מדווח עליה, אך אם לא כך, השגיאות נמצאות כמעט תמיד בשורה הבאה אחריה. בהתחלה אולי יקשה עליך לאתר שגיאות, אך תוך זמן קצר יכולתך זו תשתפר.

בטרם נסיים עם נושא זה, התבונן בדוגמה נוספת של שגיאת מהדר. טען שוב את hello.cs אל העורך שלך ובצע את השינויים הבאים:

1. החזר את הנקודה-פסיק בסוף שורה 5.
2. מחק את תו הגרשיים לפני המילה Hello.

שמור את הקובץ בדיסק והדר שוב את התוכנית. כעת המהדר יציג את הודעת השגיאה הבאה:

```
hello.cs(5,32) :  error CS1010:  Newline in constant
```

הודעת השגיאה מדווחת על המיקום הנכון של השגיאה – שורה 5. הודעות השגיאה מוצאות את השגיאה במיקום 32 בשורה 5. זהו מיקום הגרשיים הראשונים עבור הטקסט שיוצג על המסך. הודעת שגיאה זו "פיספסה" שהגרשיים חסרים בקוד. במקרה זה, המהדר השתמש בהערכה הטובה ביותר שלו אודות השגיאה. למרות קרבתה לתחום הבעיה, הערכה זו לא היתה מושלמת.

אם המהדר מדווח על מספר שגיאות, ואתה יכול למצוא רק אחת מהן, תקן שגיאה זו והדר שוב את התוכנית. אתה עשוי לגלות שתיקון בודד זה הוא כל שהיה נחוץ, והידור התוכנית יושלם בהצלחה.

הבנת שגיאות לוגיות

קיים סוג אחר של שגיאות: שגיאות לוגית. אלו אינן מסוג השגיאות הנובעות מהקוד או מהבנת המהדר; הגורם לשגיאות אלו הוא אתה בלבד. אפשר לכתוב תוכנית עם קוד #C מושלם מבחינת התחביר, אך הוא לא יבצע את משימותיו, כי יש בו שגיאות הקרויות "לוגיות". לדוגמה, נניח שאתה רוצה לחשב שטח עיגול על ידי הנוסחה 2 כפול הערך של PI כפול הרדיוס:

שטח = $2\pi r$

ניתן לרשום נוסחה זו בתוכנית, להדר ולבצע אותה. תוצאת החישוב תתקבל, אך לא תהיה נכונה, כי תוכנית #C יכולה להיות כתובה נכון מבחינה תחבירית, אך הלוגיקה שלה עלולה להיות שגויה. הנוסחה הזו לעולם לא תיתן לך את שטח העיגול, אלא את היקפו. לחישוב השטח היית צריך להשתמש בנוסחה πr^2 !

אין כל חשיבות לטיבו של המהדר, הוא לעולם לא יהיה מסוגל למצוא שגיאות לוגיות. איתור שגיאות כאלו הוא באחריותך, ועליך למצוא אותן בכוחות עצמך על ידי סקירת הקוד והרצת התוכניות שלך.

סוגי תוכניות #C

לקראת סוף הפרק, מן הראוי לדעת איזה סוגי יישומים ניתן ליצור ב-#C. תוכל לפתח מספר סוגי יישומים:

- **יישומי מסוף** (Console applications). יישומים המתבצעים משורת הפקודה (command prompt). במהלך הלימוד בספר תפתח יישומים כאלה המבוססים מעיקרם על טקסט או תווים, ולכן נשארים קלים להבנה באופן יחסי.
- **יישומי Windows** (Windows Application). ניתן לפתח יישומי Windows המנצלים את ממשק המשתמש הגרפי (GUI) שמספקת מערכת Microsoft Windows (החל מפרק 17).
- **שירותי אינטרנט** (Web Services). שירותי אינטרנט הם שגרות שניתן לקרוא להן דרך רשת האינטרנט (החל מפרק 18).

❖ **יישומי Web Forms/ASP.NET**. יישומי ASP.NET מתבצעים בשרת אינטרנט, ומייצרים דפי אינטרנט דינמיים (פרק 18).

בנוסף לסוגי יישומים אלה, C# משמשת לפיתוח תוכנות ויישומים שונים, כגון יצירת ספריות, פקדים ועוד.

סיכום

בתחילת הפרק למדת מה יש ל-C# להציע והכרת את עוצמתה, יכולותיה, גמישותה והיותה מכוונת עצמים. גם למדת ש-C# נחשבת לשפה פשוטה ומודרנית.

במהלך הפרק סקרת את השלבים השונים הכרוכים בכתיבת תוכנית C#, המוכרים כתהליך פיתוח תוכנית. לפני שתמשיך חייבת להיות לך הבנה ברורה של המחזור עריכה-הידור-בחינה.

שגיאות הן חלק בלתי נמנע מפיתוח תוכנית. מהדר C# מאתר שגיאות תחביריות בקוד המקור ומציג הודעות שגיאה, המספקות הן את טבעה והן את מיקומה של השגיאה. בעזרת מידע זה ניתן לערוך את קוד המקור כדי לתקן את השגיאה. עם זאת, זכור שהמהדר אינו מסוגל תמיד לדווח באופן מדויק על טבעה ומיקומה של שגיאה. לעיתים עליך להשתמש בידע שלך בשפת C# כדי לאתר במדויק מה גורם להודעת שגיאה נתונה. זכור שהמהדר אינו יכול לאתר שגיאות לוגיות.

שאלות ותשובות

ש: האם תוכנית C# תתבצע בכל מערכת מחשב?

ת: לא. תוכנית C# תתבצע רק במערכות שבהן מותקנת סביבת זמן הריצה CLR (**C**ommon **L**anguage **R**untime). אם תעתיק את התוכנית למערכת שאינה מכילה את CLR, תקבל שגיאה. בגרסאות של Windows שאינן כוללות את CLR, בדרך כלל תקבל הודעה שחסר קובץ DLL.

ש: אם אני רוצה למסור לאנשים תוכנית שכתבתי, איזה קבצים עלי לתת להם?

ת: אחד הדברים הנחמדים אודות C# הוא שזו שפה מהודרת. כלומר, לאחר הידור קוד המקור מקבלים תוכנית ביצוע. תוכל, אם תרצה, לתת לכל חבריך בעלי המחשבים את התוכנית hello – עליך לתת להם את קובץ הביצוע של התוכנית, hello.exe. הם אינם זקוקים לקוד המקור hello.cs, וגם לא נחוץ להם מהדר C#. הם רק צריכים להשתמש במערכת מחשב המכילה זמן ריצה של C#, כמו Common Language Runtime (CLR) של Microsoft.

ש: לאחר שיצרתי קובץ ביצוע, האם עלי לשמור את קובץ המקור (.cs)?

ת: אם תיפטר מקובץ המקור, לא תהיה לך דרך פשוטה לבצע שינויים עתידיים בתוכנית. לכן, מומלץ לשמור את קבצי המקור.

רוב סביבות הפיתוח המשולבות יוצרות קבצים נוספים בנוסף לקובץ המקור (.cs) וקובץ הביצוע (.exe). כל זמן שאתה שומר את קובץ המקור (.cs), כמעט תמיד תוכל ליצור מחדש קבצים אחרים. אם תוכניתך משתמשת במשאבים חיצוניים, כגון תמונות וטפסים, עליך לשמור תמיד גם על קבצים אלה למקרה שתצטרך לבצע שינויים וליצור מחדש את קובץ הביצוע.

ש: אם למהדר שלי מצורף עורך, האם אני חייב להשתמש בו?

ת: בהחלט לא. אתה יכול להשתמש בכל עורך, כל זמן שאותו עורך שומר את קוד המקור במבנה טקסט. אם למהדר נלווה עורך, עליך לנסות להשתמש בו. אם אתה מעדיף עורך אחר, השתמש בו. אני משתמש בעורך שרכשתי בנפרד, למרות שכל המהדרים שלי הם בעלי עורכים משלהם. העורכים המגיעים עם מהדרים משתפרים עם הזמן. חלקם מעצבים את הקוד באופן אוטומטי; אחרים מעניקים קידוד צבע לחלקים שונים של קוד המקור, ובכך מקלים על איתור שגיאות.

ש: האם אני יכול להתעלם מהודעות אזהרה (warning)?

לחלק מהודעות האזהרה אין השפעה על אופן ביצוע התוכנית ולחלק מהן יש. אם המהדר מציג הודעת אזהרה, יש בכך איתות שמשהו אינו כשורה. ברוב המהדרים ניתן לקבוע את רמת האזהרה. על ידי הגדרת רמת האזהרה ניתן לקבל רק את האזהרות החמורות ביותר, או לקבל את כל האזהרות, כולל הקלות ביותר. מספר מהדרים מציעים אפילו מספר רמות ביניים באמצע. בתוכניותיך עליך לבחון כל אזהרה ולהחליט מה לעשות. הטוב ביותר הוא תמיד לנסות לכתוב את כל התוכניות, ללא כל שגיאות או אזהרות (במקרה של שגיאה, המהדר לא ייצור את קובץ הביצוע).

סדנה

הסדנה מספקת שאלות המסייעות לבסס את הבנתך בחומר הנלמד, וגם תרגילים המעניקים לך התנסות בידע שרכשת. נסה לענות על השאלות והתרגילים לפני שתבדוק את התשובות בנספח א'. ודא שהבנת את כל התשובות לפני שתמשיך לפרק הבא.

בוחן

1. מנה שלוש סיבות מדוע C# היא בחירה מצוינת של שפת תכנות.
2. מהם הקיצורים IL ו-CLR?
3. מהם השלבים במחזור פיתוח תוכנית?
4. איזו פקודה עליך להקליד כדי להדר תוכנית הנקראת my_prog.cs באמצעות המהדר שלך?
5. באיזו סיומת קובץ עליך להשתמש עבור קבצי המקור?
6. האם filename.txt הוא שם חוקי עבור קבצי מקור של C#?
7. אם אתה מבצע תוכנית שהידרת והיא אינה פועלת כפי שציפית, מה עליך לעשות?

8. מהי שפת מכונה?

9. באיזו שורה סביר ביותר שהשגיאה הבאה התרחשה?

```
My_prog.cs(35,6) :  error CS1010:  Newline in constant
```

10. קרוב לאיזו עמודה סביר ביותר שהשגיאה הבאה התרחשה?

```
My_prog.cs(35,6) :  error CS1010:  Newline in constant
```

תרגילים

1. השתמש בעורך הטקסט שלך כדי לעיין בתוכן קובץ ה-EXE שנוצר מקוד 1.1. האם קובץ EXE נראה כמו קוד מקור? (אל תשמור קובץ זה כאשר אתה יוצא מהעורך).

2. הקלד את התוכנית הבאה והדר אותה (אל תכלול את מספרי השורות או את הנקודתיים). מה תוכנית זו עושה?

```
1:   // circle.cs - Using variables and literals
2:   // This program calculates some circle stuff.
3:   // -----------------------------------------------
4:
5:   using System;
6:
7:   class variables
8:   {
9:    public static void Main()
10:   {
11:    // Declare variables
12:
13:    int radius = 4;
14:    const double PI = 3.14159;
15:    double circum,area;
16:
17:    // Do calculations
18:
19:    area = PI * radius * radius;
20:    circum = 2 * PI * radius;
21:
22:    // Print the results
23:
24:    Console.WriteLine("Radius= {0}, PI= {1}", radius, PI);
25:    Console.WriteLine("The area is {0}", area);
26:    Console.WriteLine("The circumference is {0}", circum);
27:   }
28:  }
```

3. הקלד והדר את התוכנית הבאה. מה התוכנית עושה?

```
1:   class AClass
2:   {
3:    static void Main()
4:      {
5:        int x,y;
6:        for (x = 0; x < 10;x++, System.Console.Write("\n "))
7:           for (y = 0; y < 10; y++)
8:              System.Console.Write("X ") ;
9:       }
10:  }
```

4. **תיקון שגיאה:** בתוכנית הבאה יש בעיה. הקלד את התוכנית בעורך והדר אותה. אלו שורות מייצרות הודעות שגיאה?

```
1:   class Hello
2:   {
3:     static void Main()
4:     {
5:       System.Console.WriteLine(Keep Looking!);
6:       System.Console.WriteLine(You'll find it!);
7:     }
8:   }
```

5. ערוך את השינוי הבא בתוכנית של תרגיל 3. הדר והרץ מחדש את התוכנית. מה עושה התוכנית כעת?

```
8:      System.Console.Write( "{0}", (char) 1) ;
```