גירסה 1.00 - 26.8.2002

Starting Programming in Visual C++

מסמך זה הורד מהאתר http://underwar.livedns.co.il.
אין להפיץ מסמך זה במדיה כלשהי, ללא אישור מפורש מאת המחבר.
מחבר המסמך איננו אחראי לכל נזק, ישיר או עקיף, שיגרם עקב השימוש במידע המופיע במסמך, וכן לנכונות התוכן של הנושאים המופיעים במסמך. עם זאת, המחבר עשה את מירב המאמצים כדי לספק את המידע המדויק והמלא ביותר.

כל הזכויות שמורות ל**ניר אדר**

Nir Adar
Email: underwar@hotmail.com
Home Page: http://underwar.livedns.co.il

אנא שלחו תיקונים והערות אל המחבר.

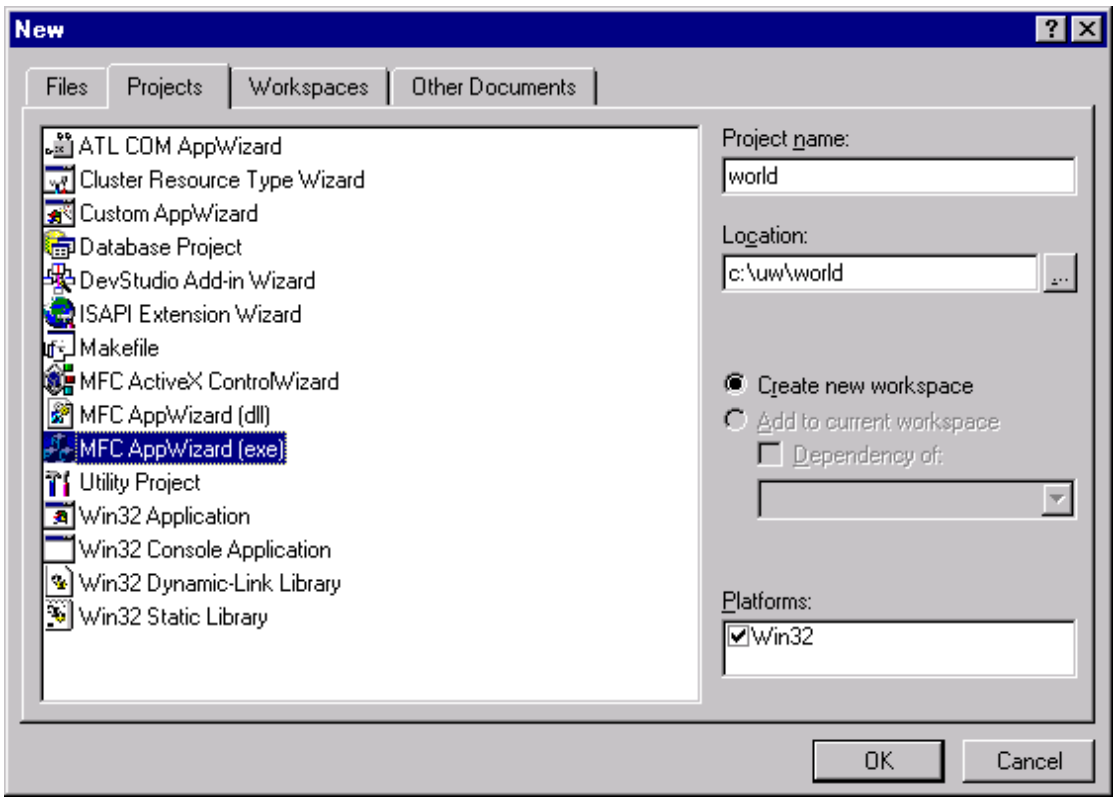
במסמך זה נביט ב-Visual C++, נכיר את Application Wizard ונציג את ארכיטקטורת "מסמך-תצוגה".
ידע קודם הנדרש להבנת מסמך זה הוא שליטה ב C++. כמו כן הכרות של שפת תכנות כלשהי ב-Windows יכולה לזרז הבנת חלק מהנושאים במסמך.

הכרות עם Application Wizard, דוגמא וארכיטקטורת "מסמך-תצוגה".

Application Wizard הוא כלי עזר שבא להפוך את תהליך התחלת כתיבת התוכנית לאוטומטי. על מנת ליצור חלון סטנדרטי ב-Windows, עוד לפני מימוש כל תכונה של התוכנית, דרוש ב-Windows קוד רב, שדי חוזר על עצמו. Application Wizard בא על מנת לכתוב קוד זה במקום המתכנת.
בעזרת Application Wizard, אנו יוצרים יישומים מבוססי חלונות, שרצים בסביבת Windows.

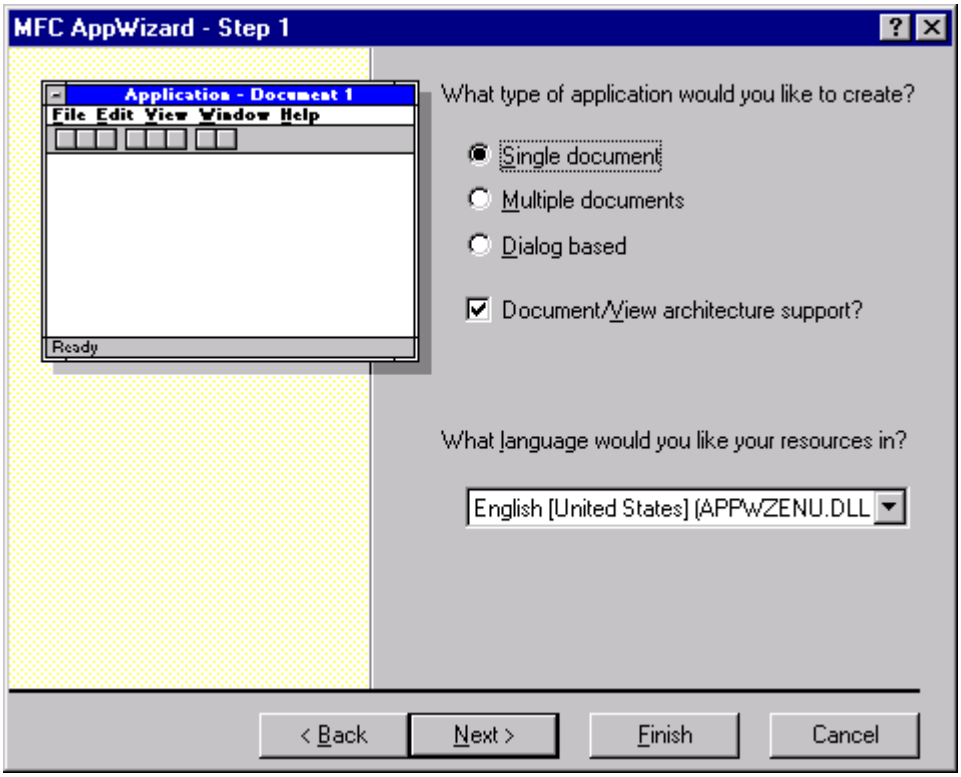
נתחיל בדוגמא פשוטה - יצירת התוכנית הקלאסית - המדפיסה "Hello World" על המסך.
בנוסף, יהיו לתוכנית שלנו את התפריטים הרגילים של תוכניות Windows - File, Edit, View, Help, אם כי לא נעשה בהם שימוש בתוכנית זו.

נתחיל על ידי פתיחת Visual C++, והתחלת פרויקט חדש. כאשר נבקש להתחיל את הפרויקט, יפתח החלון הבא:



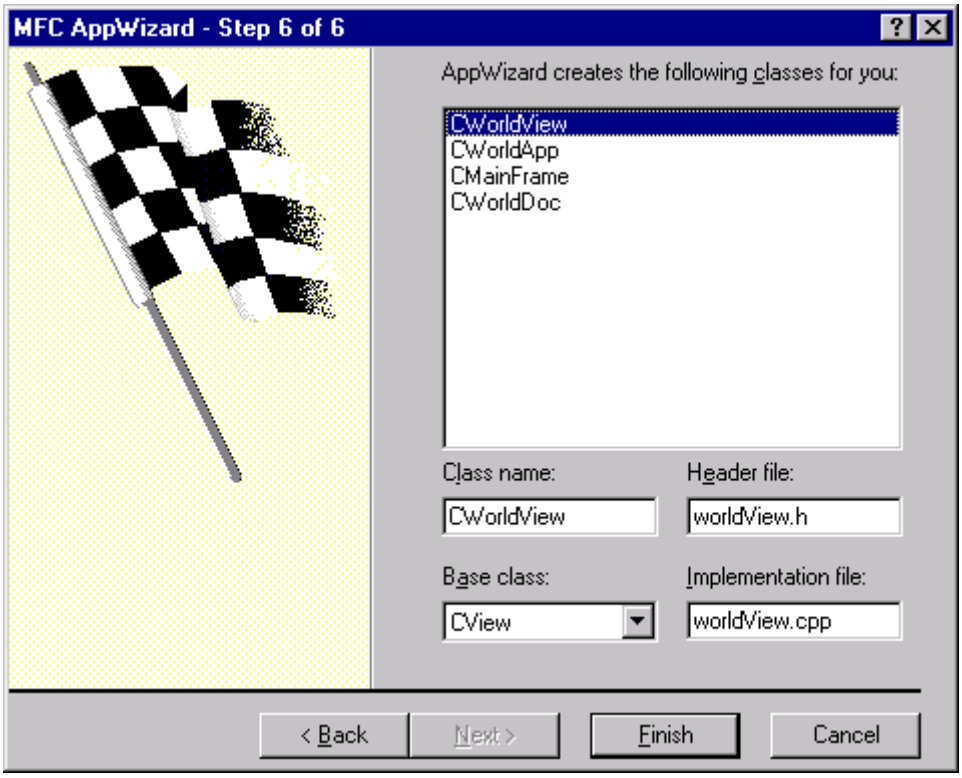
נבחר כעת ב- MFC AppWizard (exe), ובצד ימין, נכתוב את שם הפרויקט אותו אנו רוצים ליצור. במקרה זה שם הפרויקט הוא world.
נלחץ על Ok, וכעת Application Wizard (אשף היישומים) יכנס לפעולה, וישאל אותנו פרטים אודות הפרויקט אותו אנו רוצים ליצור.

ראשית נשאל האם ברצוננו ליצור תוכנית מרובת חלונות, בעלת חלון בודד או מבוססת דיאלוגים.
תוכנית מרובת חלונות היא תוכנית דוגמת Microsoft Word 97. בתוכנית זו אנו מסוגלים לעבוד על מספר מסמכים בו זמנית באותו חלון.
תוכנית בעלת חלון בודד היא תוכנית דוגמת פנקס רשימות (Notepad). בכל רגע נתון אנו מסוגלים לעבוד על מסמך אחד בלבד.
תוכנית מבוססת דיאלוגים היא תוכנית המכילה בעיקר חלונות עם כפתורים, תיבות טקסט ובקרים אחרים, ואין לה בהכרח חלון ראשי אחד.
לצורך הדוגמה, אנו נבחר באפשרות הראשונה - Single Document.



את שאר האפשרויות שמציע לנו אשף היישומים נשאיר ללא שינוי. נלחץ על Next שוב ושוב, עד שנגיע לשלב 6.

כעת אשף היישומים יציג בפנינו את המחלקות השונות שהולכות להיווצר בתוכנית החדשה: CWorldView, CWorldApp, CMainFrame, CWorldDoc.
אנו יכולים לבחור לשנות את שמות המחלקות, או את הקבצים בהם הן ימומשו.
אנו נשאיר בדוגמא זו את ההגדרות כפי שאשף היישומים מציע אותן.
נלחץ על Finish כדי לסיים את יצירת התוכנית.
את התפקיד של המחלקות השונות שאשף התצוגה ייצר, נראה בהמשך.



תוכניות שאנו יוצרים עם אשף היישומים, הן תוכניות התומכות בארכיטקטורת "מסמך-תצוגה". בארכיטקטורה זו, אנו דואגים להפרדה בין המחלקה בה נמצאים הנתונים עצמם של התוכנית, לבין המחלקה האחראית להציג נתונים אלו על המסך. בצורה זו, נוכל להציג את אותם נתונים במספר אופנים שונים, בעזרת מספר מחלקות שונות שיהיו אחראיות לתצוגה.

לתוכנית שאנו יוצרים בעזרת אשף היישומים ארבעה חלקים עיקריים: האובייקט יישום (Application), האובייקט חלון ראשי (Frame), האובייקט מסמך (Document) והאובייקט תצוגה (View).
התוכניות שאנו כותבים מתחילות מאובייקט היישום. כאשר התוכנית מתחילה, נוצר עותק בודד של אובייקט היישום. כאשר אובייקט זה נוצר, הוא מציג את החלון הראשי על המסך. אובייקט החלון הראשי מציג את התוכנית עצמה. אובייקט זה כולל את שורת התפריטים, את כותרת החלון, ואת סרגל הכלים. האובייקט חלון ראשי מגדיר למעלה את איזור הפעולה של התוכנית שלנו. כל הפעולות - ציור, טקסט וכו' מתרחשים בתוך החלון הראשי.
האובייקט תצוגה הוא השטח שבתוך החלון הראשי - השטח בו מוצגים נתוני המשתמש. ב-Word למשל, אובייקט התצוגה הוא החלון בו מוצג המסמך של המשתמש.
הנתונים המוצגים באובייקט התצוגה, נשמרים באובייקט המסמך.

נחזור לדוגמא שלנו. אנו רוצים להציג את הטקסט "Hello, World" בחלון התוכנית. על מנת לעשות זאת נוסיף קוד לפונקציה OnDraw השייכת למחלקת CWorldView. פונקציה זו נקראת כל פעם שהתוכנית רוצה להציג את איזור הלקוח של התוכנית. (למשל, כשהתוכנית מופעלת לראשונה, או כאשר חלון שהסתיר את התוכנית זז וכו').

ננווט בתצוגת המחלקות של Visual C++, עד שנמצא את הפונקציה.
הנה הקוד שאשף היישומים כתב עבורנו:

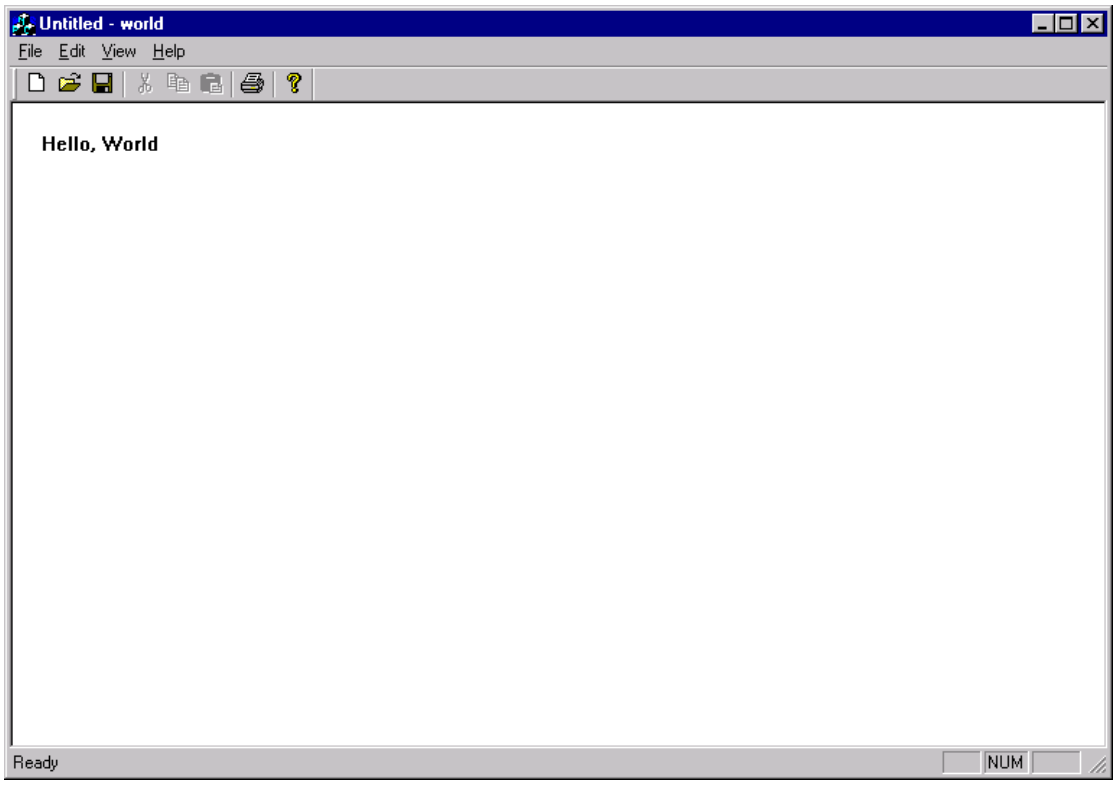
```
void CWorldView::OnDraw(CDC* pDC)
{
	CWorldDoc* pDoc = GetDocument();
	ASSERT_VALID(pDoc);
	// TODO: add draw code for native data here
}
```

בשתי השורות הראשונות של הפונקציה, אנו מקבלים מצביע אל אובייקט המסמך, ובודקים את תקינותו. תוכנית זו שנכתוב לא תעשה שימוש במצביע זה.
כעת נגרום לתוכנית להדפיס את המחרוזת הרצויה.

```
void CWorldView::OnDraw(CDC* pDC)
{
	CString szString1 = "Hello, World";
	CWorldDoc* pDoc = GetDocument();
	ASSERT_VALID(pDoc);
	pDC->TextOut(20, 20, szString1);
}
```

השורות המודגשות הן אלו שהוספו.

כעת התוכנית הושלמה, ונוכל להפעיל אותה.
בתפריט Build, נבחר ראשית Build world.exe (ניתן לקצר על ידי לחיצה על F7) ולאחר מכן נלחץ באותו תפריט על Execute world.exe (ניתן לקצר על ידי לחיצה על Ctrl+F5).



כעת נבין מה בעצם הקוד שיצר עבורנו אשף היישומים מבצע.
התוכנית מכילה ארבעה אובייקטים, המתקשרים ביניהם. התוכנית מתחילה לרוץ באובייקט יישום, בפונקציה CWorldApp::InitInstance(). נביט בקטע מהקוד שלה:

```
CSingleDocTemplate* pDocTemplate;
pDocTemplate = new CSingleDocTemplate(
     IDR_MAINFRAME,
     RUNTIME_CLASS(CWorldDoc),
     RUNTIME_CLASS(CMainFrame), // main SDI frame window
     RUNTIME_CLASS(CWorldView));
AddDocTemplate(pDocTemplate);
```

בשורות אלו אנו מציבים את המחלקות השונות שלנו ב-"תבנית מסמך", שהיא תבנית בעזרתה האובייקט יישום יציג את שאר התוכנית.
בסוף הפונקציה CWorldApp::InitInstance(), מופיעה השורה הבאה:

```
m_pMainWnd->ShowWindow(SW_SHOW);
```

שורה זו היא האחראית להצגת החלות הראשי של התוכנית, תוך שימוש בפונקציה ShowWindow ששייכת למחלקה "חלון ראשי".

הערכים האפשריים לפונקציה זו, ופירושם, להלן:

ערך	פירוש
SW_HIDE	החבא את החלון והעבר את הפוקוס לחלון אחר.
SW_MINIMIZE	מזער את החלון והעבר את הפוקוס לחלון הבא ברשימת החלונות של מערכת ההפעלה.
SW_SHOW	הפוך את החלון לפעיל והצג אותו בגודלו ומיקומו הנוכחיים.
SW_SHOWMINNOACTIVE	הצג את החלון כסמל. החלון שכרגע פעיל יישאר החלון הפעיל.
SW_SHOWMINIMIZED	הצג את החלון כסמל, והפוך אותו לפעיל.
SW_SHOWMAXIMIZED	הפוך את החלון לפעיל והצג אותו בגודל מקסימלי.
SW_SHOWNA	הצג את החלון במצבו הנוכחי. הנוכחיים. החלון שהיה קודם החלון הפעיל, יישאר החלון הפעיל.
SW_SHOWNOACTIVATE	הצג את החלון בגודלו ומיקומו הנוכחיים. החלון שהיה קודם החלון הפעיל, יישאר החלון הפעיל.
SW_SHOWNORMAL	הפוך את החלון לפעיל והצג אותו. אם החלון היה ממוזער או פרוש מקסימלית על כל המסך, הוא יוחזר לגודלו המקורי.

האובייקט חלון ראשי אחראי לחלון של התוכנית, מלבד לחלק של איזור הלקוח.
אובייקט זה אחראי, בין היתר, על התפריטים של התוכנית, על פס המצב (Status Bar) ועל סרגלי הכלים של התוכנית.
פקודות היצירה של כל רכיבי החלון האלו נעשים בפונקציה CMainFrame::OnCreate. להלן הקוד של פונקציה זו.

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
     if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
          return -1;

     if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT,
WS_CHILD | WS_VISIBLE | CBRS_TOP
          | CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY |
CBRS_SIZE_DYNAMIC) ||
          !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
     {
          TRACE0("Failed to create toolbar\n");
          return -1;      // fail to create
     }

     if (!m_wndStatusBar.Create(this) ||
          !m_wndStatusBar.SetIndicators(indicators,
            sizeof(indicators)/sizeof(UINT)))
     {
          TRACE0("Failed to create status bar\n");
          return -1;      // fail to create
     }

     // TODO: Delete these three lines if you don't want
the toolbar to
     //  be dockable
     m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
     EnableDocking(CBRS_ALIGN_ANY);
     DockControlBar(&m_wndToolBar);

     return 0;
}
```

האובייקט תצוגה הוא האובייקט שאחראי להצגת הנתונים באיזור הלקוח. רוב הפעילות של התוכניות שנכתוב, תתרחש באובייקט זה.
ניזכר בקוד שהוספנו לאובייקט:

```
void CWorldView::OnDraw(CDC* pDC)
{
      CString szString1 = "Hello, World";
      CWorldDoc* pDoc = GetDocument();
      ASSERT_VALID(pDoc);
      pDC->TextOut(20, 20, szString1);
}
```

השורה הראשונה שהוספנו מגדירה אובייקט מסוג CString, ומאתחלת אותו בעזרת מחרוזת.
CString הוא אובייקט שנועד לטיפול במחרוזות. הוא בא להחליף את השיטה של C לטיפול במחרוזות, שהיא התייחסות למחרוזת כאל מערך של תווים.

נביט במבחר מהפונקציות של CString:

בנאים	
CString( );	בנאי ברירת מחדל היוצר מחרוזת ריקה.
CString(const CString& *stringSrc* **);**	Copy Constructor
CString(TCHAR *ch*, **int** *nRepeat* **= 1);**	בנאי המאפשר לאתחל מחרוזת עם אות אחת החוזרת nRepeat פעמים.
CString(const unsigned char* *psz* **);**	בנאי המאפשר לאתחל את CString בעזרת מחרוזת בסגנון C.
מחרוזת כמערך של תווים	
int GetLength( ) const;	פונקציה המחזירה את מספר התווים במחרוזת.
BOOL IsEmpty( ) const;	פונקציה המחזירה אמת אם המחרוזת ריקה.
TCHAR GetAt(int *nIndex* **) const;** **TCHAR operator [](** **int** *nIndex* **) const;**	שתי פונקציות, המחזירות את התו הנמצא במיקום nIndex (כאשר האינדקס של האות הראשונה הוא 0).
void SetAt(int *nIndex*, **TCHAR** *ch* **);**	החלפת האות הנמצאת במיקום nIndex.
השוואת מחרוזות	
int Compare(LPCTSTR *lpsz* **) const;**	פונקציה המשווה את המחרוזת עם מחרוזת שניה.
שליפת תתי מחרוזות	
CString Mid(int *nFirst* **) const;** **CString Mid(int** *nFirst*, **int** *nCount* **) const;**	החזרת המחרוזת החל מהאינדקס nFirst, עד סופה או עד שהוצאו nCount תווים.
CString Left(int *nCount* **) const;**	החזרת nCount התווים הראשונים במחרוזת.
CString Right(int *nCount* **) const;**	החזרת nCount התווים האחרונים במחרוזת.

פונקציות של CString (המשך):

המרות שונות	
void MakeUpper( );	הפיכת כל האותיות במחרוזת לגדולות.
void MakeLower( );	הפיכת כל האותיות במחרוזת לקטנות.
void MakeReverse( );	הפיכת סדר האותיות במחרוזת.
void Format(LPCTSTR *lpszFormat*, **...);** **void Format(UINT** *nFormatID*, **...);**	השמת תוכן חדש במחרוזת, בפורמט זהה לזה של sprintf().
void TrimLeft( );	מחיקת רווחים, טאבים וסימני שורה חדשה מהתחלת המחרוזת.
void TrimRight( );	מחיקת רווחים, טאבים וסימני שורה חדשה מסוף המחרוזת.
חיפוש	
int Find(TCHAR *ch* **) const;** **int Find(LPCTSTR** *lpszSub* **) const;**	חיפוש תו או תת מחרוזת בתוך המחרוזת. מחזירה את מיקום המופע הראשון של התו או תת המחרוזת.
int ReverseFind(TCHAR *ch* **) const;**	חיפוש תו או תת מחרוזת בתוך המחרוזת. מחזירה את מיקום המופע האחרון של התו או תת המחרוזת.
אופרטורים	
TCHAR operator [](int *nIndex* **) const;**	מחזירה את התו הנמצא במיקום nIndex (כאשר האינדקס של האות הראשונה הוא 0).
const CString& operator =(const CString& *stringSrc* **);** **const CString& operator =(TCHAR** *ch* **);** **const CString& operator =(const unsigned char*** *psz* **);**	החלפת תוכן המחרוזת בתוכן חדש.
friend CString operator +(const CString& *string1*, **const CString&** *string2* **);** **friend CString operator +(const CString&** *string*, **TCHAR** *ch* **);** **friend CString operator +(TCHAR** *ch*, **const CString&** *string* **);** **friend CString operator +(const CString&** *string*, **LPCTSTR** *lpsz* **);** **friend CString operator +(LPCTSTR** *lpsz*, **const CString&** *string* **);**	חיבור שתי מחרוזות. כתוצאה מהחיבור מוחזרת מחרוזת חדשה.
const CString& operator +=(const CString& *string* **);** **const CString& operator +=(TCHAR** *ch* **);** **const CString& operator +=(LPCTSTR** *lpsz* **);**	שרשור מחרוזות.

פונקציות של CString (המשך):

אופרטורים	
BOOL operator ==(const CString& *s1*, **const CString&** *s2* **);** **BOOL operator ==(const CString&** *s1*, **LPCTSTR** *s2* **);** **BOOL operator ==(LPCTSTR** *s1*, **const CString&** *s2* **);** **BOOL operator !=(const CString&** *s1*, **const CString&** *s2* **);** **BOOL operator !=(const CString&** *s1*, **LPCTSTR** *s2* **);** **BOOL operator !=(LPCTSTR** *s1*, **const CString&** *s2* **);**	פונקציות השוואה בין מחרוזות.
friend CArchive& operator <<(CArchive& *ar*, **const CString&** *string* **);** **throw(CArchiveException);** **friend CArchive& operator >>(CArchive&** *ar*, **CString&** *string* **);** **throw(CArchiveException);**	פונקציות לשליפה וכתיבה של מחרוזות ל-Archives.

ישנם פונקציות נוספות השייכות למחלקה זו, אולם נסתפק בהצגת פונקציות אלו.

נחזור לפונקציה OnDraw:

```
void CWorldView::OnDraw(CDC* pDC)
{
     CString szString1 = "Hello, World";
     CWorldDoc* pDoc = GetDocument();
     ASSERT_VALID(pDoc);
     pDC->TextOut(20, 20, szString1);
}
```

כעת נביט במחלקה השניה המשחקת תפקיד בפונקציה זו - CDC.
CDC היא מחלקה התומכת בהקשרי התקנים - Device Contents.
כל פעולות הציור או הצגת הנתונים ב-Visual C++ מתבצעים בעזרת הקשרי התקן.
הקשר התקן הוא אובייקט המכיל פונקציות הקשורות לגרפיקה - ציור צורות שונות, הדפסת טקסט על המסך, ועוד.
כדי לצייר בתצוגה של התוכנית, למשל, אנו משתמשים באובייקט הקשר התקן, שתואם לתצוגה. ניתן להשתמש בהקשרי התקנים כדי למשל לצייר על המסך כולו (לאו דווקא בתצוגה), או למשל כדי לשלוח נתונים למדפסת.

למחלקה CDC פונקציות רבות, והצגתן חורגת מהיקף מסמך זה. הקורא המתעניין יוכל להכירן בעזרת מערכת העזרה של Visual C++.

בדוגמא שלנו השתמשנו בפונקציה TextOut() כדי להדפיס בתצוגה.
הפונקציה OnDraw() קיבלה מצביע להקשר התקן המתאים לתצוגה, וכל שעליה לעשות כדי להדפיס את המחרוזת, הוא לקרוא לפונקציה TextOut(), המקבלת מחרוזת ומיקום ומדפיסה אותה על המסך.

BOOL TextOut(int *x*, **int** *y*, **const CString&** *str*);

בזאת הסתיימה הדוגמא, וגם המסמך.
הדרך להכרת Visual C++ עוד ארוכה.
במסמך זה ראינו פינה קטנה בשפה הענקית הזו.

EOF