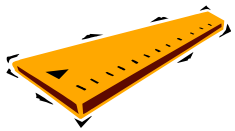
פרק 8
יצירת טיפוסים חדשים ב-C

האופרטור sizeof



```
sizeof( element )
```

- לכל טיפוס ב-C (כולל טיפוסים מורכבים, כמו מערכים) מוגדר גודל הזיכרון שיש להקצות למשתנים מטיפוס זה.
- האופרטור האונארי **sizeof** מחזיר את מספר הבתים שתופש אלמנט כלשהו בזיכרון.
- אפשר להפעיל את האופרטור **sizeof** על **כל ביטוי** שיש לו ערך (כולל משתנים, תוצאות חישובים ועוד). כמו כן (ובאופן יוצא דופן) ניתן להפעיל אותו גם על **שמות הטיפוסים** עצמם.

דוגמאות ל-sizeof

`int i;  double f;`	ערך ב-Dev-Cpp
`sizeof(f)`	8
`sizeof( (float)f )`	4
`sizeof(2*i)`	4
`sizeof(i+f)`	8
`sizeof(5)`	4
`sizeof(5.0)`	8
`sizeof(unsigned long)`	4
`sizeof( printf("%d") )`	4

פקודת typedef

- פקודת **typedef** מאפשרת לתת שם נרדף לטיפוס קיים ב-C.
- הצהרת **typedef** יכולה להופיע בכל מקום שבו יכולות לבוא הצהרות על משתנים. טווח ההכרה של השם שנגדיר באמצעות **typedef** יהיה זהה לטווח ההכרה של משתנה שהיינו מגדירים באותו מקום. במילים אחרות:
  - אם ה-**typedef** מופיע מחוץ לכל פונקציה, אז השם שאנו נצהיר עליו יהיה גלובאלי, והוא יוכר בכל התוכנית החל מהשורה שבה מופיעה ההצהרה.
  - אם ה-**typedef** מופיע בתחילת בלוק (באזור ההצהרה על משתנים), אז השם שנצהיר עליו יהיה מוכר רק בתוך בלוק זה.

שימוש ב-typedef

מבנה הצהרת **typedef**:

בתחילת השורה נכתוב את המילה השמורה **typedef**. לאחריה, נכתוב פקודה שנראית זהה לפקודת הצהרה על משתנים, אלא שכל "משתנה" שנגדיר כך לא יהיה משתנה כלל, אלא **שם חדש** לטיפוס שהוא הוגדר להיות. לדוגמה:

```
typedef int good_int, bad_int, ugly_int;

int main()
{
  good_int i, j;
  bad_int n1, n2;
  …
}
```

יהיו **i,j,n1,n2** כולם מטיפוס **int**.

דוגמאות ל-typedef

```
typedef unsigned int natural_number ;
typedef double real_number ;
```

- דוגמאות להגדרות שנמצאות בספריה הסטנדרטית (הטיפוסים עצמם תלויי קומפיילר!):

```
typedef long time_t;
typedef unsigned int size_t;
```

הטיפוס שמוחזר ע"י הפונקציה **time()** מוגדר ב- **<time.h>**

הטיפוס שמוחזר ע"י האופרטור **sizeof**. מוגדר ב- **<stdio.h>**

מדוע typedef?

- משתמשים ב-**typedef** מסיבות רבות. ביניהן:
- כאשר רוצים להשתמש לאורך כל התוכנית בטיפוס מסוים, אך זקוקים לאפשרות פשוטה להחליף טיפוס זה. למשל, סימולציה פיזיקאלית המבצעת חישובים על ערכים ממשיים, אך עוד לא הוחלט באיזו רמת דיוק (**float/double**) ייוצגו ערכים אלה.
- כאשר רוצים שהתוכנית תעבוד נכון על מגוון סביבות עבודה. למשל, הטיפוס **time_t** הוא הטיפוס שבו מיוצג הזמן במחשב. בקומפיילרים שונים, הטיפוס האמיתי שמיוצג על ידי **time_t** ישתנה בהתאם לסביבת העבודה שבה מקמפלים.
- כאשר רוצים להסתיר את המימוש הפנימי של טיפוס כלשהו. למשל, כשעובדים עם קבצים ב-C משתמשים במשתנים מטיפוס FILE. בפועל, אין צורך לדעת כלל מה הוא הטיפוס האמיתי העומד מאחורי שם זה.

sizeof עם typedef

- אפשר להפעיל את האופרטור **sizeof** על כל טיפוס – וזה כולל גם טיפוסים שהוגדרו באמצעות **typedef**.
- לדוגמה, התוכנית הבאה מדפיסה את מספר הבתים שמוקצים לייצוג הטיפוס **size_t**:

```
#include <stdio.h>

int main()
{
  printf("%d\n", sizeof(size_t) );
  return 0;
}
```

יצירת טיפוסים חדשים עם enum

```
enum color { red, green, blue };
```

- הצהרת **enum** נכתבת לפני תחילת הפונקציה **()main**, או מייד בתחילת הפונקציה (לפני ההצהרות על משתנים). היא מגדירה טיפוס חדש שניתן ליצור ממנו משתנים.
- ההצהרה שבדוגמה מגדירה טיפוס חדש בשם **enum color**.
- משתנה מטיפוס זה **enum color** יכול לקבל את אחד הערכים מקבוצת הערכים המופיעה בסוגריים המסולסלים.

```
enum color {…};
int main() {
 …
}
```

⟷

```
int main() {
 enum color {…};
 …
}
```


יצירת טיפוסים חדשים עם enum

```
enum color { red, green, blue };
```

- לכל ערך בסוגריים המסולסלים מתאים ערך מספרי מטיפוס **int**. הערכים הניתנים לקבועים אלה הם עוקבים ומתחילים מ-0:

```
red == 0 ; green == 1 ; blue == 2
```

- למעשה, הפקודה **enum** ביצעה **שתי פעולות**:
  - היא הגדירה טיפוס חדש, בשם **enum color**.
  - היא הגדירה את אוסף הערכים שהטיפוס החדש יכול לקבל, כקבועים מטיפוס **int**.

יצירת טיפוסים חדשים עם enum

```
enum color c1, c2 ;
```

- פקודה זו יוצרת שני משתנים **c1** ו-**c2** מטיפוס **enum color**.
- ניתן לאחסן בהם כל ערך שמופיע בהגדרת הטיפוס **enum color**.
- למעשה, **c1** ו-**c2** הם פשוט משתנים מסוג מספר שלם. השמה של אחד מערכי ה-**enum** לתוכם זהה להשמה מפורשת של הערך המספרי של אותו קבוע לתוכם (למרות שחלק מהקומפיילרים יידרשו casting מפורש).
- זוהי אחריות המתכנת לכך ש-**c1** ו-**c2** יכילו ערכים חוקיים בלבד.

```
c1 = green;
```

⟷

```
c1 = (enum color)1;
```


שליטה על ערכי הקבועים

```
enum numbers { one=1, two, three } ;
```

- ניתן לתת באופן מפורש ערכים לקבועים של ה-**enum**.
- ערכים אלה יכולים להיות מספרים שלמים בלבד.
- קבועים שלא צוין עבורם ערך מפורש יקבלו אוטומטית ערכים עוקבים, החל מערך המפורש האחרון שצוין.
- למשל, בדוגמה שלמעלה אנו מקבלים את הקבועים:

```
one == 1 ; two == 2 ; three == 3
```

שליטה על ערכי הקבועים

```
enum nums { zero, three = 3, four, five,
            ten = 10, eleven } ;
```

- בדוגמה זו נתנו ערכים מפורשים לחלק מן הקבועים, וליתר נתנו לקומפיילר לתת ערכים עוקבים אוטומטית. הערכים שנקבל הם:

```
zero = 0 ;  three = 3 ;  four = 4 ;
five = 5 ;  ten = 10  ;  eleven = 11
```

- מותר אפילו לתת לשני קבועים שונים אותו הערך.. במקרה זה פשוט יוגדרו שני קבועים שלמים עם אותו הערך. למשל:

```
enum zeros { null, nil = 0, minus_one = -1,
             zero, zit=0, nada=0, nothing=0 };
```


וריאציות על enum

```
enum answer { yes , no }  q1, q2, q3 ;
```

- בשורה זו הגדרנו את **enum answer**, את הקבועים **yes** ו-**no**, ובו זמנית גם יצרנו שלושה משתנים מהטיפוס החדש.

```
enum { yes , no }  q1, q2, q3 ;
```

- בשורה זו הגדרנו את שני הקבועים **yes** ו-**no**, ויצרנו שלושה משתנים מטיפוס **enum** חסר שם.
- כאן הצירוף **enum {yes, no}** עצמו מציין את סוג הטיפוס.
- כיוון שלא נתנו שם מפורש לטיפוס **enum {yes, no}**, לא נוכל ליצור משתנים נוספים מטיפוס זה.

דוגמאות ל-enum (1)

```
int main()
{
  enum day {sun, mon, tue, wed, thu, fri, sat};
  enum day today, tomorrow ;
  …
  switch (today) {
    case sun: tomorrow = mon; break;
    case mon: tomorrow = tue; break;
    …
    case sat: tomorrow = sun; break;
  }
  …
}
```


דוגמאות ל-enum (2)

```
tomorrow =
  (enum day)(( (int)today + 1 ) % 7) ;
```

- הפכנו את **today** לטיפוס **int** על מנת לקבל את ערכו המספרי.
- הוספנו 1, זה מחזיר את <u>הערך המספרי</u> של היום הבא (פרט למקרה של **today==sat** שבו הוספת 1 חורגת מהטווח).
- הפעלת מודולו 7 מבטיחה שתוצאת החיבור תהיה בתחום החוקי (0 עד 6). במרבית המקרים היא לא משנה את תוצאת החיבור, פרט למקרה של 7 (כלומר כאשר **today==sat**), שהופך ל-0 (שמייצג את **sun**).
- ה-casting ממיר חזרה את המספר לטיפוס **enum day**.

דוגמאות ל-enum (3)

בדוגמה זו אנו משתמשים ב-**enum** חסר שם. למעשה, הפקודה משמשת אך ורק לצורך הגדרת אוסף הקבועים, והיא אינה יוצרת אף טיפוס חדש.

```
enum {new='n', open='o', save='s', quit='q'};
int cmd;
…
printf("enter your command: ");
cmd = getchar();
switch (tolower(cmd)) {
  case new:  { … ; break; }
  case open: { … ; break; }
  …
  default: printf("invalid command!"); break;
}
```


typedef-ו enum

לעיתים מתכנתים מעדיפים לקצר את השם שניתן אוטומטית לטיפוסי **enum**, וזאת כדי להיפטר מהצורך לכתוב את המילה **enum** בשם הטיפוס כל פעם. ניתן לעשות זאת בעזרת **typedef**:

```
enum piece { pawn, knight, bishop,
             rook, king, queen };
typedef enum piece piece ;
…
piece p1, p2;
```

```
typedef enum { ready, set, go } mode;
…
mode m1, m2;
```