**מילון מונחים בשפת C**

|  |  |  |  |
| --- | --- | --- | --- |
| מונח בשפת C | הסבר | דוגמא בתוך קוד | הערות |
| Main () | התוכנית הראשית שבתוכה רושמים את הקוד | #include <stdio.h>  #include <stdlib.h>  Int main ()  {  \_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_  Return 0;  } | מבנה הקוד הבסיסי |
| ; | סיומת של כל שורת קוד |  |  |
| Printf("\_\_\_\_") | הדפסת טקסט למסך כפלט | Printf("hello world!"); | אם רוצים להדפיס מרכאות ( " ) בתוך הפלט צריך לשים לפניו סלש ( \ ) |
| \n | הורדת שורה בפלט | Printf("hello world!\n"); |  |
| \t | הוספת רווח גדול בפלט | Printf("hello\t world!\t friend");  מדפיס למסך:  Hello world! friend |  |
| Int \_\_ = \_\_ | משתנה המכיל מספרים שלמים | Int age = 35;  Printf("my age is %**d**", age); | מיוצג בהמשך הקוד (כשרוצים להדפיס אותו) ע"י **d**% |
| Double/float \_\_ = \_\_ | משתנה המכיל מספרים עם נקודה עשרונית (לא מספרים שלמים) | Double/float money = 40.52;  Printf("I have %**f** shekels", money); | מיוצג בהמשך הקוד (כשרוצים להדפיס אותו) ע"י **f**% |
| Char \_\_ = '\_\_' | משתנה המכיל אות/מספר אחד בלבד | Char grade = 'A';  Printf("my grade is %**c**", grade); | מיוצג בהמשך הקוד (כשרוצים להדפיס אותו) ע"י **c**% |
| \*\* טבלת ascii  קוד לייצוגם של תווים (ספרות, אותיות האלפבית, סימני פיסוק ועוד) בזיכרון מחשב ובקובצי מחשב.  כל מספר בטבלה מייצג סימן בודד בהתאמה. ניתן לחלץ כל סימן ע"י השמתו במשתנה מסוג char בכל אחד משני שמותיו.  Char a = 78  או char a = 'N'  שניהם שווים!.  ניתן להדפיס אותם בשני סוגי הערכים:  Printf("%d", a); 🡪 78  Printf("%c", a); 🡪 N |  | | |
| עוד סוגים של משתנים: | * char - משתנה לאחסון תווים או סמלים, תופס בית אחד בזיכרון, ניתן לאחסן בו ערכים מספריים שלמים מ -128 עד +127 תווים וסימנים הם כל האותיות, סימני פיסוק או סימונים אחרים שניתן להפיק ל ידי המחשב. * int - משתנה לאחסון מספר שלם. תופס שני בתים בזיכרון ניתן לאחסן בו ערכים שלמים מ -32768 עד +32767. * float - משתנה של מספרים ממשיים עד 6 ספרות דיוק (מאוחסנים בשיטת נקודה צפה) המשנה תופס ארבעה בתים בזיכרון * double - משמש לאחסון ערכים ממשים בתחומים גדולים יותר, עד 10 ספרות דיוק, תופס שמונה בתים בזיכרון. * long - ניתן לאחסן בו ערכים שלמים בטווח גדול יותר מ int תופס ארבעה בתים בזיכרון. * sing/unsing- כל משתנה מספרי ניתן להגדיר כ sing (עם סימן +/- (או unsing(ללא סימן) כאשר sing מציין אחסון של מספרים שלילים וחיוביים ואילו unsing מציין אחסונם של מספריים אי שליליים ,כתוצאה מהגדרת משתנה כ unsing טווח החיוביים שלו גדל . | | |
| טבלת גדלי המשתנים לפי ביטים (מספר המקומות הפנויים שיש בכל משתנה בזיכרון המחשב) |  | | |
| Char \_\_ [] = "\_\_"  \*מערך (array)- מכיל מספר רב של משתנים בתוכו מהסוג שהגדרנו.  \* לכל איבר במערך יש מיקום הנספר מ0 ועד סוף כמות האיברים.  {איבר2, איבר1, איבר0} | משתנה מסוג מערך המכיל מספר אותיות/תווים, מכונה גם "שרשרת" (string) | Char name [] = "john";  Printf("my name is %**s**", name); | מיוצג בהמשך הקוד ע"י %**s**.  \*\*אם לא נותנים ערך בהתחלה ומשאירים את char\_[]; פתוח, צריך לכתוב מספר בתוך ה[] |
| המשך דגשים על משתנה מסוג "מחרוזת" בהמשך הטבלה... | | | |
| Int \_ []={\_, \_, \_...} | ממשתנה מסוג מערך המכיל מספר רב מספרים שלמים בלבד. | Int luckyNum [] = {4, 8, 15, 16};  Printf("%**d**", luckyNum [1]);  התשובה: 8  או  Int luckyNum [10];  luckyNum [1] = 80;  printf("%d", luckyNum [1]);  התשובה: 80 | ניתן ליצור מערך גם של double וfloat. מערך של char כבר צוין למעלה כ"שרשרת".  \*\*אם לא נותנים ערך בהתחלה ומשאירים אתint\_[]; פתוח, צריך לכתוב מספר בתוך ה[] כדי להגביל את כמות האיברים. |
| + (חיבור) - (חיסור)  / (חילוק)  \* (כפל)  Pow(\_,\_) (מס' בחזקת מס')  Sqrt (\_) (שורש מרובע)  % (מציג רק שארית חלוקה)  Ceil(\_) (מעגל מספר למעלה)  Floor(\_) (מעגל מספר למטה)  Num++ (להוסיף 1)  Num - - (להוריד 1) | פעולות חשבון בתוך הקוד | Int num1, num2, num3, sum;  Sum = num1+num2-num3;  Sum = num1/num2\*num3;  או  Printf("%f", pow(3,2)); 🡪 9.0  Printf("%d",sprt(36)); 🡪6  Printf("%d", 3%2); 🡪 1  Printf("%f",ceil(36.567)) 🡪 37.0 | אפשר לחפש עוד פעולות בגוגל |
| /\* \_ \*/ או \_ // | הערה שלא נחשבת חלק מהקוד | /\* חלק לא חשוב בלהבלה \*/  Char grade = 'A';  Printf("my grade is %c", grade); |  |
| const | הופך כל משתנה לקבוע שלא ניתן לשנות את הערך שלו בהמשך הקוד | Const int Num = 5;  Printf("%d", num);  לא ניתן להוסיף!!!  Num = 8;  לא ניתן לשנות את הערך של Num | נוהגים לתת להם שם שמתחיל באות גדולה באנגלית |
| \_\_=sizeof(\_\_) | \*\*מילת קוד שמטרתה להחזיר בסרגל הכלים locals את גודלו בביטים של המשתנה הנמצא מאחורי השווה | Int num;  Num = 100;  Int num2;  Num2 = sizeof(num);  Printf("%d", num2);  תשובה: בסרגל כלים locals יופיע שnum2 שווה ל4, כי משתנה מסוג int מכיל 4 ביטים. | **לבדוק את זה באינטרנט** |
| & | "אמפרסנט (amp)", מייצג את הכתובת של המשתנה שנכתב בצמוד אליו בזיכרון | Int num2 = 200;  Printf("variable num2: %d %d" , num2, &num2);  התוצאה: המחשב ידפיס 200 ואחריו את התא בזיכרון (מיוצג בתור מספר לא ברור)  או  Int num1, num2;  Num1=&num2;  נתנו לnum1 את המקום בזיכרון של num2 |  |
| Scanf("%\_", &\_) | מכניס ערכים שהמשתמש הקליד לתוך הקוד, הערך חייב להתאים לסוג המשתנה שכתבנו אחרי ה%, ומציב אותו במשתנה שהגדרנו אחרי ה&. | Int age;  Printf("enter your age:");  Scanf("%d", &age);  Printf("you are %d", age);  או  char name [20];  Printf("enter your name:");  Scanf("%s", name);  Printf("you are %s", name);  או (אם רוצים לבקש שני ערכים)  Int num1;  Int num2;  Scanf("%d,%d",&num1,&num2);  \*המשתמש יצטרך להקליד גם ","  או  Scanf("%d %d",&num1,&num2);  \*המשתמש יצטרך להקליד גם את הרווח | \*\*משתנה מסוג **double** מיוצג בתוך scanf ע"י %**lf**.  \*\*משתנה מסוג **char[]** (שרשרת) מיוצג בתןך scanf ע"י %**s** ושמו **ללא** הסימן &.  **המחשב עוצר לקבל מידע מהמשתמש ברגע שהוא מזהה רווח/אנטר.** |
| Fgets(\_\_, \_\_,stdin) | מכניס ערכים שהמשתמש הקליד לתוך הקוד **גם אם מדובר בערכים עם רווח בניהם**  **(כמו: ג'ון סמית)** | Char name [20];  Printf("enter your name:");  Fgets(name, 20, stdin);  Printf("your name is %s", name);  🡪ניתן להכניס ערכים כמו "שלומי כהן" עד 20 אותיות. | המחשב מדפיס את הפלט של המשתמש עם שורה ריקה אחריה (סתם חוק של הקוד הספציפי הזה) |
| If(\_\_)  {  \_\_\_;  }  Else  {  \_\_\_;  } | משפט תנאי, שאומר למחשב לקיים פעולה מסוימת רק אם התנאי שבראשה מתקיים. ואם לא, לבצע פעולה אחרת | Int num;  Scanf("%d", &num);  If (num>100)  {  Printf("big");  }  Else  }  Printf("small");  } | סימנים מוסכמים לתנאים:  \*< (גדול)  \*> (קטן)  \*== (שווה ל) \*=! (לא שווה ל)  \*=< (גדול שווה)  \*=> (קטן שווה). |
| תנאי מקוצר  (condition)?(T):(F) | דרך מקוצרת לכתוב תנאי | Scanf("%d", &num);  If (num > 100)  { printf("100"); }  Else { printf("0"); }  דרך מקוצרת: **num = (num > 100) ? printf("100") : printf("0");** |  |
| הרחבה לתנאי else if | הוספת תנאי נוסף למשפט תנאי רגיל | Int num;  Scanf("%d", &num);  If (num==100)  { Printf("bingo");  }  Else if (num>100)  { Printf("winner");  }  Else  { printf("looser");  } | אין הגבלה לכמות הelse if שאפשר להוסיף. |
| תנאי מקוצר מורחב  (condition)?( (condition)?(T):(F)):(F) | דרך מקוצרת לכתוב תנאי מורחב | לא הדגים.  (cond')?((cond')?(T):(F)):(F) |  |
| הרחבה לתנאי && או || | הוספת "וגם" או "או" לתוך התנאי | Int a,b,c;  Printf("enter 3 numbers");  Scanf("%d %d %d", &a,&b,&c);  If (a==b && a==c) {  Printf("all nums are equal");  }  Else {  Printf("all nums are not equal");  } | \***&&** - וגם - כלומר הif מתקיים רק אם שני תנאים שהגדרנו מראש מתקיימים שניהם  \***||** - או - כלומר הif מתקיים רק אם לפחות אחד מהתנאים שהגדרנו מראש מתקיים  \***חוקים נוספים**:  !True=False\*  !False=True\*  \*קדימות לסוגרים  \*קדימות ל AND |
| While(condition)  {  \_\_\_;  } | לולאה שמטרתה לחזור על קטע קוד מסוים שוב ושוב עד שהתנאי מפסיק להתקיים או להיות נכון | Int a;  A = 0;  While (a<10)  {  Printf("\*\*");  A = a+1;  } | \*counter loop לולאה מונה – מתקדמת במרווחים קבועים |
| שני סוגים של לולאות while: | counter loop - לולאה מונה | Int x = 0;  While(x<10)  }  Printf("%d", x);  X++;  } | לולאה שסופרת במרווחים קבועים |
| Condition loop לולאת תנאי - | Int sum = 0;  Int num = 0;  While(sum<100)  }  Scanf("%d", &num);  Sum=sum+num;  } | לולאה שרצה עד שהתנאי מפסיק להתקיים |
| For (\_\_; \_\_; \_\_\_);  for (1.declare variable ; 2.continue condition ; 3.increment)  1.הגדרת משתנה  2.תנאי לקיום  3.פעולה לביצוע | לולאת for, לולאה הממשיכה לבצע את קטע הקוד שבתוכה עד שהתנאי הנכתב בראשה מפסיק להתקיים. | For (int x=1; x<=20; x++)  {  Int num1 = x;  Int num2 = x\*3;  Int result = num1+num2;  Printf("%d+%d=%d", num1, num2, result);  } | דרך מקוצרת לכתיבת לולאת while |
| לולאה פנימית (nested loop) – דוגמאות ללולאה בתוך לולאה |  | | |
| Int \_1\_ (\_2\_)  }  קוד כלשהו  Return \_3\_;  }  1.שם הפונקציה  2.שמות המשתנים שהפונקציה אמורה לקלוט  3. הפלט שהפונקציה אמורה להוציא בסוף הקוד | **פונקציה חיצונית** – מסוג int.  נכתבת מחוץ לפונקציית הmain, מבצעת קטע קוד שהוגדר לה ומחזירה פלט מסוים לפונקציה המרכזית ברגע ש"קוראים" לה בשמה. | Int main()  {  Int a;  A = calsum(20,30,40);  Printf("%d", a);  Return 0;  }  Int calsum(int num1, int num2)  {  Int ret;  Ret = num1+num2;  Return ret;  } | \*כדי "לקרוא" לפונקציה, פשוט צריך לציין את שמה בmain.  \*אם הפונקציה לא מחזירה ערך מספרי, יש לסיימה בruturn0. |
| float \_1\_ (\_2\_)  }  קוד כלשהו  Return \_3\_;  }  1.שם הפונקציה  2.שמות המשתנים שהפונקציה אמורה לקלוט  3. הפלט שהפונקציה אמורה להוציא בסוף הקוד | **פונקציה חיצונית** – מסוג float. | int main()  {  Float b = getdevide(234234);  Printf("%d", b);  Return 0;  }  Float getdevide (float a)  {  Float d = 23;  Float Ret = a / d;  Return ret;  } | \*ניתן לכתוב פונקציות מכל סוגי המשתנים.  \*ניתן לרשום את הפונקציה מתחת לmain, עדיף אבל **תמיד למעלה** (לפחות אזכור), למחשב יהיה יותר קל לקרוא ככה |
| void \_ ()  {  קוד כלשהו  } | **פונקציה חיצונית –** מסוג void  לא מבצעת פעולה על משתנה כלשהו, אלא מחזירה פלט קבוע. ניתן להכניס בה משתנים. | Int main ()  {  Sayhi();  Return 0; }  void sayhi ()  { printf("hellow user"); }  ----------------------------------------  Int main ()  { Sayhi ("mike", 40);  Return 0; }  void sayhi (char a [], int b)  { printf("hellow %s, your age is %d", a, b); } | \*לא רושמים בסוף הפונקציה החיצונית מסוג void את המילה return |
| gets(\_\_); | משמש לקליטת ערך מהמשתמש והשמתו בתוך משתנה מסוג "מחרוזת" (string = char[]) | Int main()  {  Char str [100];  gets(str);  printf("%c", str[2]);  printf("%s", str);  return 0;  } | דומה לscanf ולfgets, רק יותר מתאים לשימוש עם מחרוזות. |
| דגשים במשתנה מסוג מחרוזת (string) – char [] | האיבר האחרון במחרוזת חייב להכיל 0 = null (טבלת ascii) | Char str [4];  Str[0] = 'A';  Str[1] = 'B';  Str[2] = 'A';  Str[3] = 0; //end of string = NULL |  |
| כתובת של מחרוזת/מערך הוא השם שלה, אין צורך לכתוב & כדי לאזכר את הכתובת של המחרוזת | Char str [100];  Int num;  //&num 🡪 address of num location  //str 🡪 address of array |  |
| פעולות מיוחדות במחרוזת | #include <string.h>  Char str [100];  Char str2 [100];  Int a;  **מחזיר את האורך של המחרוזת:**  a = strlen(str);  **מעתיק מחרוזת אחת למחרוזת אחרת (יכול להעתיק תווים בודדים):**  Strcpy(str2, str);  Strcpy(str2, "abada");  **מדביק מחרוזת אחת לסוף של השנייה (יכול להדביק תווים בודדים):**  Strcat(str2, " ");  Strcat(str2, str);  **משווה שתי מחרוזות (לא יכול להשוות תווים בודדים):**  A = strcmp("ABC","ABC") 🡪 a=0  A = strcmp("ABC","ABC") 🡪 a=1  **מוצא את המופע הראשון של מחרוזת אחת בתוך השנייה, ומוביל אליה:**  Char str3[] = "abada1234";  Char str4[] = "da";  Strstr(str3, str4);  **מעתיק למחרוזת אחת מחרוזת אחרת, לפי מספר "צעדים" שהוגדרו:**  Char word [] = "abcd";  Char newword [100];  Strncpy(newword, word, 3); 🡪  יעתיק לתוך newword רק את "abc". | כדי להשתמש בכל הפעולות המיוחדות, צריך לאזכר מילון מיוחד לפני פונקציית הmain שמכילה את כל הפרמטרים המיוחדים:  **#include <string.h>** |
| \_\_\*  משתנה מסוג פוינטר: | מצביע על הכתובת של המשתנה שכתבנו לו, מהסוג שהגדרנו מראש. יש פוינטרים לכל סוגי המשתנים. | https://lh4.googleusercontent.com/6mEryD6IzFgh8-Wf9G1N1nvO9Ta-U09pNDe8bhPjiiHkBqcZgK7ALIVJ41THESHnNPeE9qF-SWJA-0o4d35FmhQkMnBK_kCL6YXsV1f08GLHo9iUxAa1xW12y7a7cSn2x2hO3Nml_jn5wDaJPQ  char str [1000];  strcpy(str, "Shalom");  **// kuku is variabel of pointer type**  char \*kuku;  **// Now str and kuku are pointed in same memory** **address**  kuku = str;  **// Now str and kuku are pointed to "Lhit" value**  strcpy(kuku, "Lhit");  //pointer of an int  int a = 199;  int\* b;  b = &a; 🡪 **b and a are pointed in same memory** **address**  **\*b** = 198; **🡪 by putting \* before the pointer name (b) we went inside to the pointer value. Now a and b equal to 198 and not to 199.** | |
| \_1\_ = atoi (\_2\_);  1.משתנה מסוג int  2.משתנה מסוג char/string | פונקציה המימרה סימנים/אותיות/מחרוזות למספרים שהקוד יודע לחשב ולעשות עליהם פעולות חשבון | Char str1[] = "100";  Char str2[] = "20";  Int num1, num2, sum;  Num1 = atoi (str1);  Num2 = atoi (str2);  Sum = num1+num2; 🡪 120 | כדי שהפונקציה תעבוד צריך להכליל את הספרייה הבאה בתחילת הקוד: #include <stdlib.h> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |