# פעולות על סיביות בשפת C

* בשימוש בפעולות על סיביות, אין אנו מתייחסים לערך שהמשתנה מכיל, אלא לכל אחת מהסיביות.
* פעולות AND, OR, ו-XOR מקבלות שני ערכים ופועלות על הסיביות המקבילות בכל אחד מהערכים – סיבית ראשונה מול סיבית ראשונה, סיבית שניה מול סיבית שניה וכן הלאה.
* פעולות ההזזה, מזיזות את הסיביות כמות פעמים לכיוון כלשהו. סיביות אשר אין להם מקום, לא יהיו קיימות במספר החדש.
* פעולת ה-NOT הופכת את סיביותיו של המשתנה היחיד אותו היא מקבלת.

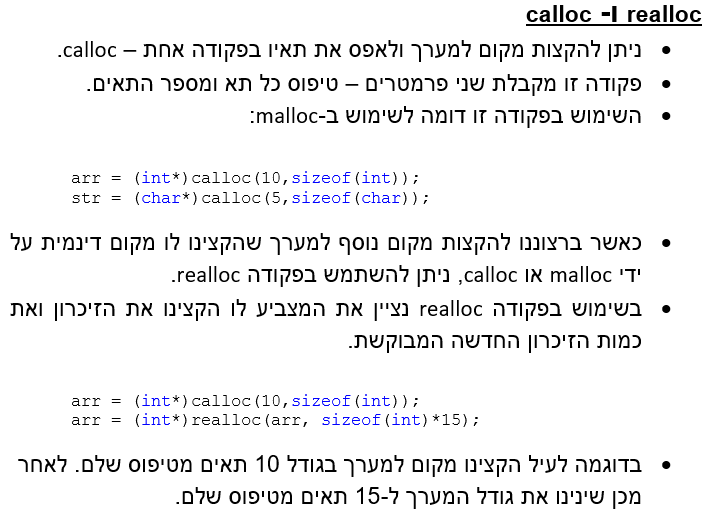


Char= בית אחד (8 ביטים)

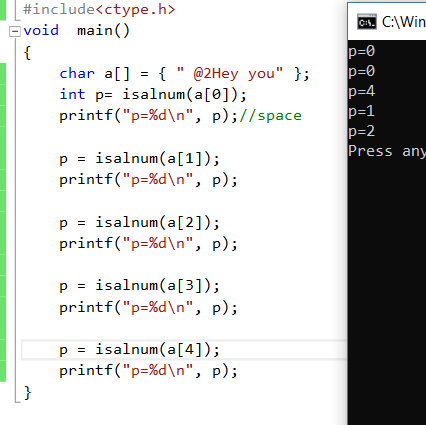
Int=ארבע בתים (32 ביטים)

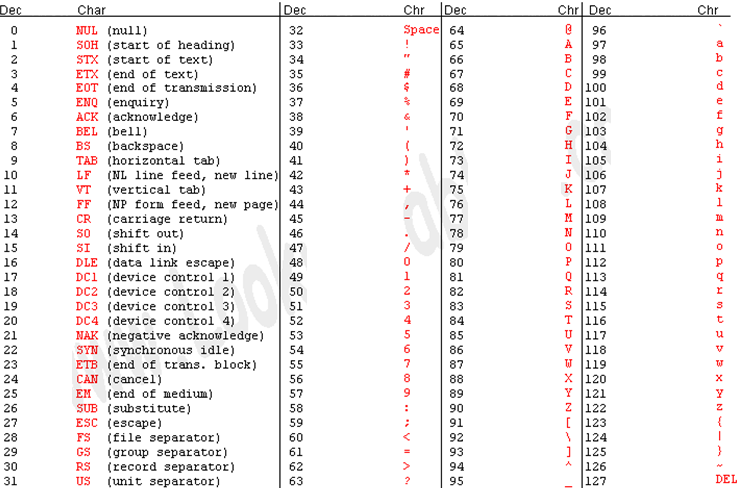
כאשר נרצה לבודד ביטים נכפיל על פי הטבלה הנתונה

|  |  |  |
| --- | --- | --- |
| 8 | D7 | 0x80 |
| 7 | D6 | 0x40 |
| 6 | D5 | 0x20 |
| 5 | D4 | 0x10 |
| 4 | D3 | 0x08 |
| 3 | D2 | 0x04 |
| 2 | D1 | 0x02 |
| 1 | D0 | 0x01 |



***פונקציות :ctype.h***





|  |  |  |
| --- | --- | --- |
| טווח ערכים  Value Range | גודל בבתים  Size in Bytes | טיפוס  Data Type |
| מספרים שלמים מסומנים | | |
| -128 … 127 | 1 | **(signed) short int** |
| -32,768 … 32,767 | 2 | **(signed) int** |
| -2,147,483,648 … 2,147,483,647 | 4 | **(signed) long int** |
| מספרים שלמים לא מסומנים | | |
| 0 … 255 | 1 | **unsigned short int** |
| 0 … 65,535 | 2 | **unsigned int** |
| 0 … 4,294,967,295 | 4 | **unsigned long int** |
| מספרים ממשיים | | |
| דיוק של עד 7 ספרות | 4 | **(signed/unsigned) float** |
| דיוק של עד 15 ספרות | 4 | **(signed/unsigned) double** |
| תווים | | |
| -128 ... 127  התו מזוהה לפי ערך ASCII | 1 | **(signed/unsigned) char** |

|  |  |  |
| --- | --- | --- |
| טיפוס המשתנה | התו | דוגמא לכיצד יופיע הערך על המסך |
| מספר שלם (int, char) | %d | 97 |
| מספר שלם ארוך ((long int | %ld | 97 |
| מספר ממשי  (float) | %f | 97.384 |
| מספר ממשי  (double) | %lf | 97.3846696 |
| תו (char) | %c | a |
| \* מחרוזת | %s | Hello world |
| \* מספר שלם בייצוג בבסיס 8 (Octal) | %o | 141 |
| \* מספר שלם בייצוג בבסיס 16 (Hexadecimal) | %x  או  %X | 61 |
| \* כתובת זיכרון | %p | B800:3443 |

***מודולו***

5%3=2

2%3=2

-2%2=0

-1%2=0

-3%2=-1

**פונקציות למחרוזות בספריה string.h**

1.    הופך את הסדר של כל האותיות עד '0\' במחרוזת:

      strrev(str);

1. השוואת מחרוזות:

א) משווה אלפבתית  בין שתי מחרוזות:

num=strcmp(str1,str2);

            ב) משווה אלפבתית  n תווים ראשונים בין שתי מחרוזות:

num=strncmp(str1,str2,4);

      ג) משווה אלפבתית  n תווים ראשונים בין שתי מחרוזות, ומפסיק את   
          ההשוואה ברגע שנמצא שוני כלשהו:

num=strncmpi(str1,str2,n);

      בכל הפונקציות של סעיף זה:

אם 0 num>      אזי : str1>str2

      אם 0 num<      אזי : str1<str2

      אם 0 num=      אזי : str1=str2 .

1. ממיר במחרוזת  את האותיות הגדולות לאותיות קטנות:

strlwr(str);

1. ממיר במחרוזת  את האותיות הקטנות לאותיות גדולות:

strupr(str);

1. מחזיר אורך מחרוזת לא כולל '0\'

num=strlen(str);

1. העתקת מחרוזות:

א) מעתיק מחרוזת למחרוזת עד לתו '0\':

strcpy(str1,str2);

         (גם stpcpy עושה את אותה הפעולה).

     ב) מעתיק n איברים ממחרוזת למחרוזת:

strncpy (str1,str2,n);

      ג) יוצר העתק של str ,מחזיר מצביע להעתק שנוצר:

ptr=strdup(str);

1. חיבור מחרוזות:

א) מוסיף את str2 ל- str1 מייד בסוף המחרוזת של str1.

     כלומר במקום הסימן '0\' שנמצא ב- str1 תושם האות הראשונה של str2

     וכ"ו.

strcat(str1,str2);

ב) מוסיף n תווים ראשונים מ- str2   לסוף- str1. כלומר :האיבר הראשון של   
    str2 יתווסף במקום ה-'0\' שב- str1 ,האיבר השני יבוא לאחר מכן וכ"ו עד   
    האיבר ה-n –י .לאחר האיבר  ה-n –י שהתווסף יושם '0\' .

Strncat(str1,str2,n);

לדוגמא:

;“AAA”int str1[7]=  
”BBB”int str2[7]=

strncat(str1,str2,2);

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ? | '\0' | B | B | A | A | A |

      ב-str1 יהיה:

1. הצבת תווים במחרוזת:

א) ממלא מחרוזת באות מסויימת :

Strset(str,ch);

ב) מכניס n תווים ch ל-str ,יעצור אם יפגוש '0\' .

strnset(str1,ch,13);

     או

strnset(str1,ch,strlen(str)-1);

      א) כנ"ל עם מצביעים memset.

1. חיפוש תו:

א) מוצא תו ch במחרוזת str ומחזיר מצביע לאיבר הראשון שנמצא:

ptr=strchr(str,ch);

ב)מוצא את ההופעה האחרונה של ch ב- str ומחזיר מצביע לאיבר שנמצא ,

      אם לא נמצא איבר כזה הפונקציה תחזיר NULL.

Ptr=strchr(str,ch);

      ג) מחפש תו ch ב-n איברים הראשונים ומחזיר מצביע לאיבר אם נמצא,  
          או NULL אם לא נמצא:

ptr=memchr(str,ch,strlen(str));

1. מחזירה את האינדקס של של האיבר הראשון שבו str1  שונה מ- str2.

num=strspn(str1,str2);

1. מחזיר מספר שמייצג את האינדקס של האיבר הראשון ב- str1  שנמצא גם ב-str2 . אם לא נמצא איבר כזה יוחזר האינדקס של '0\'  שבעצם נמצא בשניהם (אורך המחרוזת).

num=strcspn(str1,str2);

1. מוצא את המופע הראשון של מחרוזת str2 ב- str1, ומחזירה מצביע לאיבר הראשון ב- str1 שבו מתחילה str2. אם לא נמצא מופע כזה מוחזר NULL.

ptr=strstr(str1,str2);

1. מחזירה מצביע לאיבר הראשון ב- str1  שקיים גם ב - str2.

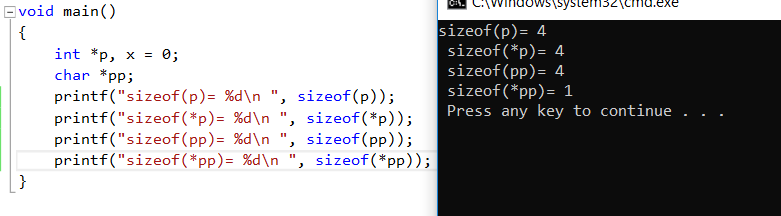
ptr=strpbrk(str1,str2);

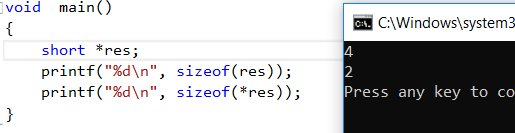
1. מוצא ב- str1 את האיבר הראשון שלא קיים ב- str2 ,ומציבה '0\' לפני איבר זה ב- str1, מחזירה מצביע , אם לא נמצאו איברים שונים מחזירה NULL.

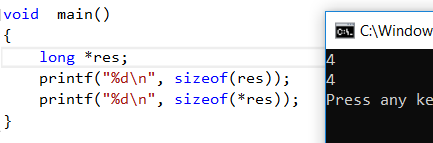
Str2 בעצם מגדיר סימנים מפרידים ב- str1.

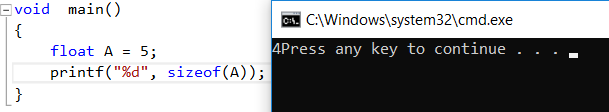
***קימפולים :***

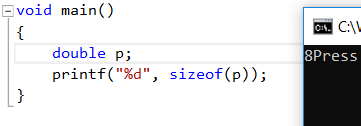
**הדפסת גודל של פויינטר:**

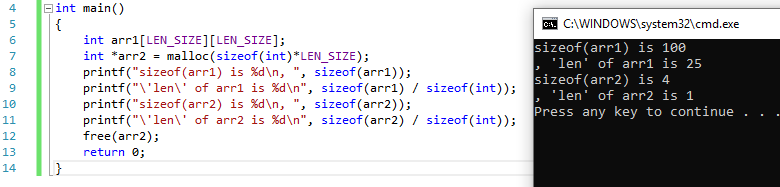












1. If(!x)זה כמו שנרשום כךif(x==0)
2. If(x) זה כמו שנרשום if(x!=0)
3. אם התנאי נמצא בתוך הreturn(!str1[i])

אז הוא יחזר 1 עם התנאי נכון הוא יחזיר 0 אם הוא לא נכון

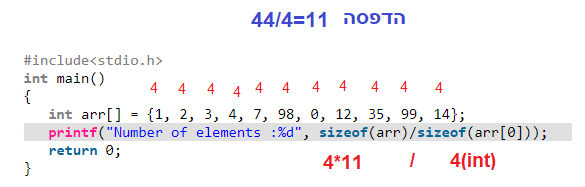
4.אם המספר היחיד קטן מהמספר של המודלו זה שווה למספר

היחיד 3%10=3

5. num = (mat[n][i] >= mat[i][n]) ? num + mat[n][i] : num - mat[i][n];

אם התנאי שלפני הסימן שאלה מתקיים, אז תבצע את מה שאחרי הסימן שאלה  
אם התנאי שלפני הסימן שאלה לא מתקיים, אז תבצע את מה שאחרי הנקודותיים

6.



7.אם הפונקציה void והערכים שהיא מקבלת לא פוינטר אין שינוי בתוכנית

דוגמא:

void swap1(int a, int b)

{

int temp;

temp = a;

a = b;

b = temp;

}

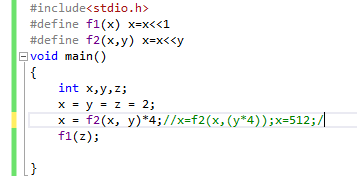
9. static int tmp=3 אם מוגדר משתנה סטטי זה אומר שנגידר אותו 3 פעם אחת וכל פעם המשתנה

10.אם יש m=**++i ? j++ : ++k**

נעלה את i אחר כך נבדוק שi!=0 אם כן נכניס את j ל m

ואז נעלה את j

11.



***פוינטרים***

\*(++str+1) = '\0'; \*

--str

שלבים לביצוע:

1.מצביעה לתחילת המערך

2.מעלים + 1 וזה תחילת המערך החדש

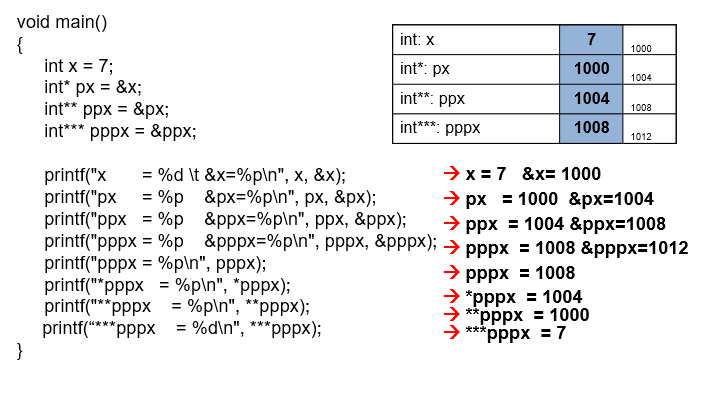
3.אתה מתקדם עוד צעד אחד מתחילת המערך החדש ואחרי שהתקדמת מציבים 0 וחוזרים לאיפה שזה היה

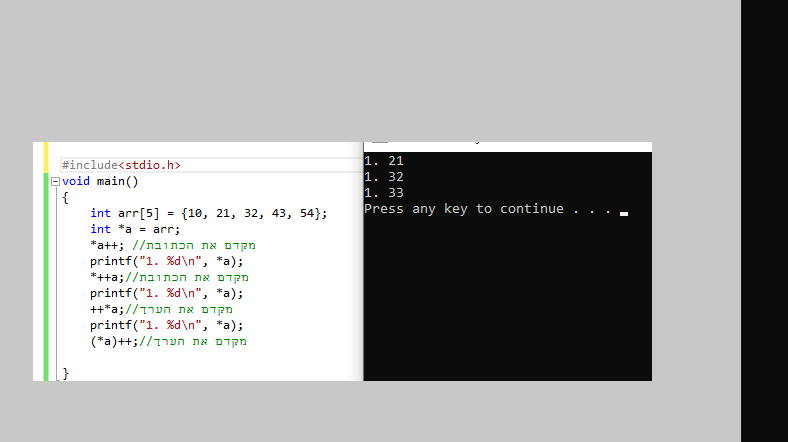
4.ומציב 0 במקום שתקדמנו .

scope\_sum\_matrix(int mat[N][N], int \*per)

כשולחים לפנוקציה פוינטר חייב בהחזר של הפונקציה לעשות & לפוינטר זה יראה ככה

scope\_sum\_matrix(mat, &per)

****



int main()

{

int \*\*mat = NULL;//עמודה ראשונה

int i = 0;//

mat = (int \*\*)malloc(3 \* sizeof(int \*));//מקצה עמודה ראשונה בגודל 3 כל תא יכיל int

mat[0] = (int \*)malloc(12 \* sizeof(int));//לשורה הראשונה נעשה 12 תאים

for (i = 0; i<12; i++)//כל פעם נוסיף 5 לכל אתחול תא

mat[0][i] = (i + 1) \* 5;

mat[1] = mat[0] + 3;//מאט[1] מחזיק את המיקום של מאט[0] ולהתקדם 3 תאים

mat[2] = mat[1] + 5;// מאט[2] מחזיק את המיקום של מאט[1] ולהתקדם 5 תאים

printf(" A: %d\n", mat[1][2]);//מה שמאט[1] מצביע ונתקדם 2 צעדים

printf(" B: %d\n", mat[1] - mat[0]);//חיסור בין כתובות

printf(" E: %d\n", ((mat + 1) + 6));//מאט[1] ועוד 6 צעדים

}***יצירה מערך דו מימדי עם פוינטרים***

void main()

{

int i, j,m = 4, n = 4;

int\*\* array = (int\*\*)malloc(m \* sizeof(int\*));//נקצה עמודה בגודל m

for (i = 0; i<m; i++)

{

array[i] = (int\*)malloc(n \* sizeof(int));//לולאה שמקצה 4 תאים לכל שורה

if (array[i] == NULL)

return -1;

}

}

***ביטים***

# כפל סיביות

|  |  |  |
| --- | --- | --- |
| & | כפל סיביות  AND |  |

/\* 13 = 1101

14 = 1110

----------

1100 \*/

# חיבור סיביות – OR

|  |  |  |
| --- | --- | --- |
| | | חיבור סיביות  OR |  |

// 13 = 1101

// 2 = 0010

//----------

// 15= 1111

# eXclusive OR – XOR

|  |  |  |
| --- | --- | --- |
| ^ | XOR  (Exclusive OR) |  |

// 13 = 1101

// 7 = 0111

//----------

// 10 = 1010

אופרטור המשלים

האופרטור ~ הינו המשלים של המספר

אופרטור זה הוא אופרטור אונרי (פועל על משתנה יחיד) בהינתן מספר בעל 8 בתים

~0x00000008 == 0xFFFFFFF7

~0xFFFFFFF8 == 0x00000007

