סיכום תכנות בשפת C (פקודות)

פקודות **Pre-Processor**

בפקודות Pre-Processor אין נקודה פסיק ";" וכל אחת מהן תופסת שורה שלמה

#include <stdio.h>

ספריה המכילה את הפקודות הבאות :

כתיבת טקסט על המסך, **printf** ("Print Text on screen as written**n**");

ניתן לכתוב גם משתנים printf ("The sum is : %d the average is : %d " ,sum ,average);

תווים מיוחדים :

%d - הדפסת ערכים שלמים

%f - הדפסת ערכים ממשיים

%c - הדפסת תו

%s - הדפסת מחרוזת

\n - מוריד את הטקסט שורה אחת למטה

%% - הדפסת התו '%'

\\ - הדפסת התו '\'

\" - הדפסת גרשיים

\' - הדפסת גרש

קליטת משתנים ע"פ סוג :

scanf ("%**d**%**ld**%**f**%**lf**%**c**%**s**" ,&**Integer** ,&**Long**(integer) ,&**Float** ,&**Double** ,&**Char** ,&**String**);

הספרייה הזו מכילה גם את כל פקודות הקבצים

פקודות מתמטיות, כל הפונקציות מחזירות ערכי double #include <math.h>

ברדיאנים בלבד sin(x), asin(x), sinh(x), cos(x), acos(x), cosh(x), tan(x), atan(x), tanh(x)

exp(x) = e^x, log10(x) = $\log_{10}(x)$, log(x) = ln(x)

עבור משתנים מטיפוס double pow (x,y) = x^y, sqrt(x), fabs(x) = |x|

עבור משתנים שלמים power (x,y), abs (x)

פקודות של מחרוזות #include <string.h>

קליטה של מחרוזת ובסיומה שם '0\' **gets** (string name)

מדפיס מחרוזת ומעביר שורה **puts** (string name)

מכניס לתוך משתנה את אורך המחרוזת i = **strlen** (string name)

מעתיק את st2 לתוך st1. **strcpy** (st1, st2)

strcat (st1, st2) מחבר את st2 לסוף שרשרת st1

strcmp (st1, st2) בודק את הסדר המילוני של המחרוזת, אם המילים זהות הוא מחזיר 0

אם st1 בא לפני st2 הוא יחזיר ערך שלילי אם st2 בא לפני st1 הוא יחזיר ערך חיובי

#include <stdlib.h>

size of (אופרטור) מחזיר את הערך המספרי של כמות הזיכרון אותו תופס האופרטור

לדוגמא: i = size of (int) (= 4)

srand, rand, time

memory allocation - **malloc**

pointer = (* type of pointer) **malloc** (wanted size * **size of** (type of pointer));

double * arr; arr = (* double) malloc (4 * size of (double)); יוצר 4 תאי דאבל עם מצביע על התא הראשון

free (arr) משחרר את מה שיצרנו מהזיכרון

#include <path\filename> ניתן לשתף פונקציה מתוך קובץ שאנחנו כתבנו

הגדרות קבועות לאורך המסמך (define) החלפת סימון אחד בסימון אחר, בד"כ באותיות גדולות.

#define PI 3.14159 (יעבור על המסמך ויחליף את PI ב- 3.14159)

/* This is the best help ever */ or // Is this cool or what הערות לקטע או שורה

הגדרת משתנים

שם משתנה מתחיל תמיד באות (בד"כ קטנה) ואח"כ רצף אותיות וספרות (חשוב לתת שמות בעלי משמעות).

type	byte		סימן מזהה	סימון מספרים	
char	1	-128 to 127	%c	'a', 'b' etc.	
Short	2	-2^{15} to $2^{15}-1$			
int	2 או 4		%d	±123	מספרים שלמים
Long	4	-2^{31} to $2^{31}-1$		±123**L**	מספרים שלמים גדולים
Float	4	$\pm 10^{\pm 38}$	%f	.123 \ 0.123**f**	דיוק של 7 ספרות אחרי הנקודה
Double	8	$\pm 10^{\pm 300}$	%lf	1E-17 = $1x10^{-17}$	דיוק של 15 ספרות אחרי הנקודה
Long double	≥8			0.1234**L**	
String []			%s		

כל משתנה מקבל ערך התחלתי רנדומאלי, חשוב לאפס אותם !!

פעולות חשבוניות על מספרים שלמים :

+ (חיבור), - (חיסור), * (כפל), / (חילוק ללא שארית), % (מחזיר רק את השארית).

פעולות חשבוניות על מספרים ממשיים :

+ (חיבור), - (חיסור), * (כפל), / (חילוק).

Casting

העברת משתמש רחב (double) **למשתנה צר** (int)

לדוגמא הכנסת הערך השלם של הביטוי לתוך i

int i; double x,y; i=(int)[x*y];

לא צריך לעשות casting מ- int ל- char, המחשב ממיר את המספר לאות ע"פ קוד האסקי.

הוראת IF

אם הביטוי שונה מ-0 בצע את ההוראה,

if (ביטוי) הוראה ;

אם הביטוי שווה ל-0 דלג הלאה או אם יש else בצע אותו.

else הוראה ;

(אם יש יותר מהוראה אחת)

if (ביטוי) {אוסף הוראות}

else {אוסף הוראות}

IF מקוצר

ביטוי3 : ביטוי2 ? ביטוי1

אם לביטוי **1** יש ערך שונה מ- 0 כל ה- IF שווה לביטוי **2**.

אם ביטוי **1** יש ערך שווה ל- 0 כל ה- IF שווה לביטוי **3**.

לדוגמא :

m = a>b ? a :b ; או m = a>b ? a : (a==b ? 0 : b);

אופרטורי יחס

< (גדול), =< (גדול שווה), > (קטן), => (קטן שווה), == (שווה), =! (שונה).

מחזירים ערך 0 אם הביטוי אינו נכון (False) או 1 אם הביטוי נכון (True).

לדוגמא :

a = 2; b = 3; | c = a>b;↔ (c=0) | m = (a>b)*a + (b>=a)*b ↔ (m=3)

טבלאות ייחוס

! (not) x = !7; ↔ (x=0) x = !!7; ↔ (x=1)

!	0	≠0
	1	1

&& (and) if (x>100 && x<200) x; בודק את קיום שני התנאים

&&	0	≠0
0	0	0
≠0	0	1

|| (or) if (x>100 || x<200) x; בודק את קיום לפחות אחד מהתנאים

||	0	≠0
0	0	1
≠0	1	1

לולאות (Loops)

לולאות **While** - כל עוד לביטוי בסוגריים יש ערך שונה מ-0 תבצע את ההוראה

While (ביטוי) הוראה; While (ביטוי) {אוסף הוראות}

While (1) a=a+1; (לולאה אינסופית)

בצע את כל ההוראות ואז Do {אוסף הוראות}

כל עוד לביטוי בסוגריים יש ערך שונה מ-0 תבצע אותן שוב While (ביטוי)

לולאות **For** - מבצע לולאה ע"פ הסדר הבא : (1) ביצוע הוראה/אוסף הוראות, (2) בדיקת תנאי/ביטוי, (3) ביצוע הוראה ראשית, (4) ביצוע הוראה/אוסף הוראות.

For ((1) הוראה, הוראה ;(2) ביטוי ;(4) הוראה, הוראה) (3) הוראה;

For ((1) הוראה, הוראה ;(2) ביטוי ;(4) הוראה, הוראה) (3) {אוסף הוראות}

נשתמש בעיקר כאשר נרצה להריץ לולאה עם מונה קבוע מראש לדוגמא :

לולאה שמריצה משתנה i מ-0 עד (לא כולל) 10 For (i = 0 , j = 100 ; i < 10 ; i++ , j--)

פקודת **Break** - גורם לתוכנית לצאת מתוך הלולאה בה הוא נמצא.

פקודת **Continue** - גורם לתוכנית לחזור לתחילת הלולאה.

פקודות אלו יבואו בד"כ בתוך תנאי If

מעריכים (Arrays)

רצף של איברים מאותו סוג לדוגמא, 20 תווים מ-0 ל-19 (מאפס את כל המערך) int a[20]={0};

המערך מכיל ערכים שרירותיים ויש צורך לאפס אותו char letters1{'a','b','c'},letters2[3]={'a'};

לדוגמא : מכניס ערכים לשני התאים הראשונים והשאר מאפס int a[4]={1,2};

מחרוזות

מחרוזות הינם מערכים של char אשר בסופם קיים התו '**0**' char st1[]="abcd"

(מערך קבוע בזיכרון שלא ניתן לשנותו).

'a'	'b'	'c'	'd'	'\0'

מערך דו מימדי

התא השני חייב להיות מלא int num [5][3] = {{1,2}{3}{3,1,4}}

1	2	0
3	0	0
3	1	4
0	0	0
0	0	0

פונקציות Functions

שם מזהה טיפוס (שם מזהה טיפוס , שם מזהה טיפוס .....) {

הכרזות

הוראות

}

```
double factorial (int number) {
        double result=1; int I;
        for (i=2; i<=number;i++) result*=i;
        return result;
}
void main () { printf ("%f\n", factorial (3)) }
```

בכל פונקציה ניתן להשתמש בשמות משתנים ללא כל קשר לפונקציות האחרות, הטיפוס של המשתנה ב- return חייב להיות אותו טיפוס כשל הפונקציה.

פונקציות מטיפוס void כמו התוכנית הראשית או פונקציות הדפסה לא צריך להיות להם return.

פונקציה יכולה להחזיר לכל היותר ערך אחד.

Prototype

כדי שנוכל לקרוא לאיזו פונקציה שאנו רוצים במהלך התוכנית ללא חשיבות סדר הופעתם נצטרך להגדיר אותם בתחילת התוכנית.

double factorial (int number); or double factorial (int);

מערכים ופונקציות

ניתן לשלוח כתובת של מערך לפונקציה, וכל שינוי שנעשה על המערך בתוך הפונקציה נשמר ולא נעלם שמו של המערך הינו הכתובת שלו ראו דוגמא (פונקציות / מציאת שורש ריבועי).

משתנה לוקלי static

משתנה אשר נשאר קבוע גם כאשר אנו קוראים שוב לאותה פונקציה באופן רקורסיבי static int number=1;

הפונקציה מתייחסת לאתחול רק בפעם הראשונה שהפונקציה מופעלת, במידה והמשתנה אינו מאותחל ערכו ההתחלתי הינו 0, שימושי לביצוע ספירת פעולות בפונקציות רקורסיביות.

משתנה קבוע const

משתנה שלא ניתן לשנות את ערכו במהלך התוכנית const double e=2.718281828

Switch - Case

במקום לשים הרבה תנאים שונים ניתן לאגד את כל התנאים תחת סעיף אחד switch (ביטוי שלם) {

Case קבוע שלם: Case קבוע שלם: Case קבוע שלם: ...

פעולות;

break;

Case קבוע שלם:

פעולות;

break;

אם לא נמצא תנאי מתאים לביצוע תבוצע ברירת המחדל default:

}

הפונקציה מחפשת את התנאי המתאים ומבצע את כל הפקודות תחתיו עד ה- break, אין חובה לשים סעיף default. יש לשים לב כי אנו שמים ערכים שונים בכל case.

טבלת קדימויות

קודם מכניס את j לתוך i ורק אח"כ מקדם את j i = j++ ↔ i=j | j=j+1

קודם מקדם את j ורק אח"כ מכניס אותו לתוך I i = ++j ↔ j=j+1 | i=j

i = j = j+k; ↔ i = j+k, j = j+k

סדרי עדיפויות של אופרטורים

פעולות אריתמטיות הולכות משמאל לימין
לדוגמא: a=(b*c)/d הסוגריים יתווספו לבד

( ), [ ]
->, .
!, &, ++, --
*, /, %
+, -
<, <=, >, >=
==, !=
&&
||
?, : ← מימין לשמאל
=, +=, -=, *=, \=, %=

פקודת "לך אל" go to - אסור להשתמש

מדלג על קטע קוד ע"פ ההגדרה :

```
if ….. go to (שם מזהה) lable;
.
.
lable : פקודות ;
```

מצביעים - Pointers

מצביע כשמו הוא, מצביע על תא בזיכרון.

```
int a, * p;
```

p מצביע על התא a, &a הינה הכתובת של a.

```
p = &a;
```

*p הינו הערך של התא - וכעת הוא משנה אותו ל-0

```
*p = 0;
```

לא ניתן להדפיס פוינטרים רק את הערך שלהם

```
printf ("%d", *p);
```

פוינטרים יכולים להצביע על מערכים

```
int a[10], *p;
```

מצביע על התא הראשון של המערך

```
p = arr ; or p = &arr[0];
```

יקדם את המצביע לתא ה- i

```
p+=i or p = &arr[i] or p = arr+i
```

(אם נחרוג מגבולות המערך לא נקבל הודעת שגיאה)

מקדם את **הערך** בתא הראשון (arr[0]) ב-1

```
(*arr)++;
```

ביטויים דומים

*(x + number) ↔ x[number] *&i ↔ i arr[5] ↔ *(arr+5)

p = arr+3 ; *(p-2) = 0 ↔ p[-2] = 0 ↔ arr[1] = 0

הפרש בין פוינטרים שווה לערך מספרי (int) אסור לבצע סכום פוינטרים p = arr ; q = arr +10 ; i = q-p (10)

מצביע על מחרוזת קבועה

```
char * st="abcd"
```

קליטה משתנה לתוך תא של מצביע

```
scanf ("%lf", &a[i] or a+i)
```

על פוינטר (מסוג char) **אסור** לעשות *st='x' חייבים להשתמש בפונקציה **strcpy**.

מערך פוינטרים דו-מימדי

מערך פוינטרים דו-מימדי הינו משתנים מסוג ** - ראה דוגמא.

מבנים Structures

בלוק בזכרון שניתן לאחסן בו נתונים מטיפוסים שונים ומתייחסים אליו בתור משתנה אחד.

הגדרת הטיפוס.

```
typedef struct dugma {
        char name;
        int arr[10];
        struct dogma * next;
} dugma;
```

מאפשר להגדיר את המשתנה כעת בתור dugam ולא struct dugma

שימושים

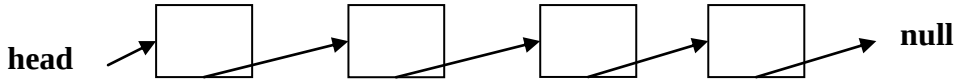
```
struct dogma stam, temp = {"davide",{1,2,3},}
```

temp.arr[6]=65

ניתן להשוות בין שני מבנים מאותו סוג stam = dogma;

רשימות מקושרות / משורשרות - Linked list

כל מבנה מכיל את הכתובת של המבנה הבא, קיים עוגן שהוא ראש הרשימה, והמבנה האחרון מצביע על null.



רשימות דו כיווניות

רשימה אשר ניתן ללכת בה בשני הכיוונים.

```
typedef struct dugma {
        char name;
        struct dogma * next;
        struct dogma * prev;
} dugma;
```

אם נרצה להדפיס רשימה חד כיוונית לא מעגלית מהסוף להתחלה נשתמש בשתי לולאות, קודם נספור את מס' המבנים ברשימה ואז נרוץ כל פעם עד הסוף (פחות 1) ונדפיס את כל האיברים כך.

קבצים ופונקציות קלט פלט

מצביע על קובץ (ההגדרה חייבת להיות באותיות גדולות) **FILE * fp** (File Pointer)

כל פקודות הקבצים נמצאים בספריה <Stdio.h> וכל הפונקציות הרלוונטיות נמצאים עם תחילית f.

פתיחת קובץ fopen

הפקודה חייבת לבוא בתוך תנאי if (fp=fopen ("Path \ FileName","r"(read)/"w"(write)/"a"(append))

אם אין Path המערכת תחפש/תיצור את הקובץ בספריה בה היא נמצאת.

"r" (read)

כאשר יש קובץ קיים ואנו רוצים לפתוח אותו לקריאה, במידה והקובץ לא קיים יוחזר לנו NULL

"w" (write)

כשהקובץ לא קיים המערכת פותחת קובץ חדש לכתיבה

(אם הקובץ קיים המערכת תמחק אותו ותיצור קובץ עם שם חדש)

"a" (append)

פתיחת קובץ קיים עם הרשאות כתיבה (אם הקובץ לא קיים הפקודה מתפקדת כ- "w" ויוצרת קובץ חדש)

דוגמא:

לשים לב כי צריך לשים עוד \ כשכותבים את הכתובת של הקובץ

```
if ((fp=fopen ("c:\\Natun.dat","r"))==NULL) {
        printf ("Error in opening file\n");
        exit (1);
}
```

יוצא לגמרי מהתוכנית

סגירת קובץ fclose

טיפוס מסוג int אשר מחזיר EOF (-1) אם יש תקלה, אם הפקודה מצליחה לסגור היא מחזירה 0.

```
if (fclose (fp)==EOF) {
        printf ("Error in closing file\n");
        exit (1);
}
```

פוינטרים מסוג (FILE *) שאין צורך להכריז עליהם

קלט - מקלדת **Stdin**

פלט - מסך **Stdout**

<u>קריאת תו מתוך קובץ</u>

int fgetc (FILE *)

קורא מתוך קובץ תו ומחזיר את הערך המספרי שלו מכניס לתוך המשתנה 8 ביטים

ומ‌מיר אותו חזרה לתו (התווים של המקלדת הינם מתורגמים אוטומטית כ- 8 ביטים)

הפקודה מעבירה תו לתוך משתנה מטיפוס char ועוברת לתו הבא בקובץ....

```
char ch;
ch=fgetc(fp)
```

לדוגמא:

התוכנית פותחת קובץ לקריאה ובודקת כמה פעמים מופיע בו התו $ עד שהיא מגיע לסוף הקובץ.

```
void main () {
FILE *fp; int count=0; char ch;
if ((fp=fopen ("c:\\Bill.dat","r"))==NULL) { printf ("Error in opening file\n"); exit (1); }
```

או

```
fp=fopen ("c:\\Bill.dat","r");
if (!fp) { printf ("Error in opening file\n"); exit (1); }

while ((ch=fgetc(fp))!=EOF)  if (ch==$) count++;

if (fclose (fp)==EOF) { printf ("Error in closing file\n"); exit (1); }
}
```

אם אנחנו רוצים לקלוט תו לתוך משתנה מטיפוס char מתוך המקלדת נשתמש באחד מהפקודות הבאות:

```
ch=getchar();
ch=fgetc(stdin);
```

ניתן להזין ידנית סוף קובץ ע"י Ctrl+Z.

אם הגענו לסוף הקובץ ניתן לחזור לתחילתו ע"י (int) rewind (FILE *)

<u>הערה</u>

אין צורך לעשות Casting מ- Int ל- Char התוכנה לוקח את 8 הביטים הראשונים וממירה אותם לתו:

```
char ch; int a,b=1;
a = ch = b+1000;        ↔        a=1001, ch=תו
```

<u>כתיבת תו לתוך קובץ</u> int fputc (int , FILE *)

קליטת טקסט לתוך משתנה מטיפוס "תו" int putchar (int);

אם נכניס מספר תווים ולאחריו נלחץ Enter זה יקלט בקובץ בתור תו '\n'.

קליטת טקסט מהמשתמש והכנסה לקובץ - דוגמא:

כל עוד הקלט אינו נקודה

```
while ((ch=getchar())!='.')
if (fputc (ch,fp)==EOF) { printf ("Error in input\n"); exit (1); }

if (fclose(fp)==EOF) { printf ("Error in closing file\n"); exit (1); }
```

קליטת מחרוזת מתוך קובץ char * fgets (char * , int , FILE *)

int - גודל המחרוזת שאנו רוצים להעתיק

while (fgets(str,40,fp)!=NULL) puts (str);

מעתיק מחרוזת בגודל 40 לתוך str ומדפיס אותו,

אם בקובץ יש מחרוזת יותר גדולה מ-40 אז הוא יכניס לתוך str את 39 התווים הראשונים והתו הארבעים יהיה '\0', אם יש מחרוזת ובאמצע יש '\n' אז הוא יקרא עד ה- '\n' (כולל) ויוסיף ל- str '\0'.

השלמות

הדפסה על המסך printf (רשימת ביטויים ,"מחרוזת המרה")

הדפסה לתוך מחרוזת sprintf (char *, רשימת ביטויים ,"מחרוזת המרה")

הדפסה לתוך קובץ fprintf (FILE * (יעד), רשימת ביטויים ,,"מחרוזת המרה")

סוגי מחרוזות המרה:

%5d - 123__ (מספרים שלמים) סה"כ ידפיס 5 תווים ואם יש פחות מ- 5 תווים ישים לפניו רווחים להשלים ל- 5 תווים, אם נרצה שהרווחים יהיו אחרי המספרים ולא לפני נשים מספר שלילי (%-5d).

%6f - 12.345 , (מספרים עשרוניים) המחשב מחשיב את הנקודה כתו.

%.2f - 12.34, מדפיס 2 ספרות אחרי הנקודה.

%11.4f 12.3400____, מדפיס סה"כ 11 תווים (כולל הנקודה), עם 4 מספרים אחרי הנקודה והשאר רווחים, אם אין מספיק איברים אחרי הנקודה הוא מוסיף אפסים.

%.0f - מדפיס ללא הנקודה וללא ספרות אחריו.

%c - מאפשר להדפיס תו אחד.

%s - הדפסת מחרוזת.

%2s - הדפסת שני התווים הראשונים של המחרוזת, במידה ואין מספיק תווים יוסיף רווחים.

<u>דוגמאות</u>

char st[]="abc" ; printf ("%c",*st) {ידפיס a} ; printf ("%2c",*st) {ידפיס a ושני רווחים לפניו} ;

printf ("%8s",*st) {ידפיס abc וחמישה רווחים לפניו};

char st[100]; int i=123; sprintf (st, "%d", i) {st='1','2','3','\n'}

קליטה מתוך המקלדת רצף תווים וממיר אותם לפי מחרוזת ההמרה scanf (רשימת ביטויים ,"מחרוזת המרה")

מוציא char מתוך מחרוזת ועושה המרה ע"פ מחרוזת ההמרה sscanf (char *, רשימת ביטויים ,"מחרוזת המרה")

מוציא char מתוך קובץ ועושה המרה ע"פ מחרוזת ההמרה fscanf (FILE * (יעד), רשימת ביטויים ,,"מחרוזת המרה")

<u>הפונקציות מחזירות ערך המייצג את כמות הערכים שהוא הצליח לקלוט</u>

%d - קליטת int, %ld - קליטת long int, %h - קליטת short (לא בשימוש)

%3d - קליטת 3 תווים או עד Enter, Tab או תו.

לדוגמא: עבור הקלט 1234567 והביטוי "%3d%3d",&i,&j {i=123} {j=456}

%f - קליטת float, %lf - קליטת double, %c קליטת char.

לדוגמא : עבור הקלט 12.34567 והביטוי "%4f%2d",&i,&j {i=12.3} {j=45}

%s - קליטת מחרוזת על לרווח, Enter או Tab.

[]% - קליטה של מחרוזת במקום %s, ואם כתוב משהו בסוגריים, הוא יקלוט כל עוד יש לו את אחד התווים מהסוגריים

לדוגמא : עבור הקלט 11*db2c והביטוי "%[ab1*d]",&i {i='11*db'}

קליטת רק אותיות או עד 20 תווים %20[A-Za-z]

תקלוט תווים עד שאתה מגיע למה שנמצא אחרי הכובע %[^0-9] (קליטה עד שהוא מגיע למספר)

תקלוט עד שאתה מגיע ל-$ או עד 9 תווים %9[^$].

<u>תתעלם</u> מקלט **%*c**, <u>תדלג</u> על 10 התווים הבאים **%*10c**, תקלוט מספר אבל אל <u>תעתיק</u> אותו **%*d**

דוגמא : עבור הביטוי "%d,%d" - תקלוט 2 מספרים ופסיק ותתעלם מהפסיק בינהם

<u>תוכנית לדוגמא</u>

החלפת סדר של משתנים בתוך מערך של 100 תווים :

```
int a[100], i, temp;
for (i=0 ; i<50 ; i++) {
        temp = a[i];
        a[i] = a[99-i];
        a[99-i] = temp;
}
```

<u>Selection Sort</u> <u>ממין את הרשימה מהקטן לגדול</u>

```
#define max 100
int i, j, temp, arr[max], imax;
for (i = max-1 ; i>0 ; i--) {
        imax = 0;
        for (j = 1; j<=i, j++) if (arr[j]>arr[imax]) imax = j;    מוצא את המיקום של האיבר הגדול ביותר
        temp = arr[i];
        arr[i] = arr[imax];
        arr[imax] = temp;
}
```

<u>Quick sort</u>

```
void swap (int * arr, int i , int j){
```

```
        int temp;
        temp = arr[i]; arr[i]=arr[j]; arr[j]=temp; }

void quick_sort (int * arr, int l, int r){
        int i, temp;
        if (l>=r) return;
        temp=l;
        for (i=l+1 ; i<=r ; i++) if (arr[i]<arr[l]) swap (arr, i, temp);
        swap (arr , l, temp);
        quick_sort (arr, l, temp-1);
        quick_sort (arr, temp+1, r);
}
```

Insertion Sort

```
#define max 100
int i, j, temp, arr[max];
for (i=1; i<max; i++){
        temp=arr[i];
        for (j=I; j>0 && arr[j-1]>temp; j--){
                arr[j] = arr[j-1];
                arr[j-1] = temp;
        }
}
```

פונקציות / מציאת שורש ריבועי

```
int roots (double a, double b, double c, double x[]); prototype
void main () {
int sol, Q[2];
        sol = roots (4.2, 7.3, -8.5, Q);
        if (!sol) printf ("no solution\n");
                else    if (sol==1) printf ("one solution: %f\n",Q[0]);
                        else printf ("two solutions: %f, %f\n",Q[0], Q[1]);
}
```

```
int roots (double a, double b, double c, double x[]) {
        int delta;
        delta = b*b-4*a*c;
        if (delta<0) return 0;
        if (!delta) {
                x[0]= -b/(2*a);
                return 1;
        }..... המשך הפונקציה
}
```

פונקציות של מחרוזות

```
int strlen (char st[]){
        int i;
        for (i=0; st[i]!='\0' ; i++);
        return i;
}
```

```
void strlen (char * st){
int i;
        for (i=0 ; *st ; st++);
        return i;
}
```

```
void strcpy (char st1[], char st2[]){
        int i;
        for (i=0; st2[i]; i++) st[i]=st2[i];
        st1[i]='\0';
}
```

```
void strcpy (char * st1, char *  st2){
        while (*(st1++)=*(st2++));
}
```

אותו דבר כמו :

```
*st1=*st2; st1++; st2++;
```

לפני שהוא מעתיק את st2 הוא בודק האם ערכו 0 או לא

```
void strcat (char st1[], char st2[]){
        int i,j;
        for (i=0; st1[i] ; i++);
        for (j=0; st1[i]=st2[j]; i++, j++);
}
```

```
void strcat (char * st1, char * st2){
        while (*st1) st1++;
        while (*(st1++)=*(st2++));
}
```

```
int strcmp (char st1[], char st2[]){
        int i, j ;
        for (i=0 ; st1[i]==st2[i] ; i++);
        if (!st1[i]) return 0;
        return st1[i]-st2[i];
}
```

```
int strcmp (char * st1, char * st2){
        for ( ; *st1==*st2 ; st1+, st2++);
        if (!*st1) return 0;
        return *st1-*st2;
}
```

פונקציה שמשכפלת מערך חד מימדי

```
char * str_duplicate (char * st) {
char * dup;
dup = (char *) malloc (strlen(st) + 1);
```

יצירה של מערך חד מימדי של תווים (גודל תו שווה 1)

```
strcpy (dup, st);
return dup;
}
```

יצירת מערך פוינטרים דו-מימדי

```
void main (){
int ** a, i, num;
scanf ("%d", num);
a=(int **) malloc (size of (int *) * num);
for (i=0 ; i<num ; i++) a[i]=(int *) malloc (size of (int) * i);
```

יצירת מערך דו מימדי מדורג

```
for (i=0 ; i<num ; i++) free (a[i]);
free (a);
}
```

שחרור כל התאים

חיפוש ליניארי במערך ממוין (הפונקציה מחזירה את מיקום המספר)

```
int linear_search (int arr[], int size, int num){
	int i;
	for (i=0; i<size; i++) if (arr[i]==num) return i;
	return -1;
}
```

חיפוש בינארי

```
int binary_search (int arr[], int size, int num){
	int left=0, right, middle;
	right=size;
	while (left<right) {
		middle=(left+right)/2;
		if (number==arr[middle] return middle;
		if (arr[middle]>num) right=middle;
		else left=middle;
		}
	return -1;
}
```

חיפוש בינארי בפונקציה רקורסיבית

```
int binary_search_recurse (int arr[], int left, int right, int number){
	int middle;
	if (left>right) return -1;
	middle=(left+right)/2;
```

```
        if (arr[middle]==number) return middle;
        if (number<arr[middle]) return binary_search_recurse (arr, left, middle-1, number);
        return binary_search_recurse (arr, middle+1, right, number);
}
```

חיפוש בינארי רקורסיבי עם פוינטרים

```
int  * binary_search2 (int * left, int * right, int num){
        int * middle;
        if (left>right) return null;
        Int *   Int *   int
        middle=left +(right-left)/2;
        if (*middle==num) return middle;
        if (number<*middle) return binary_search2 (left, middle-1, num);
        return binary_search2 (middle+1, right, num);
}
```

מיון מערך בשיטת הבועות

```
void bubble_sort (int arr[], int size){
	int i, j, temp, sorted=0;
	for (i=size-1 ; i>0 && !sorted; i--){
		sorted = 1;
		for (j=0; j<i; j++) if (arr[j]>arr[j+1]){
						temp=arr[j];
						arr[j]=arr[j+1];
						arr[j+1]=temp;
						sorted=0;
	}}
```

פונקציה המדפיסה את כל הוקטורים הבינאריים באורך מסוים

```
void comb (int arr[], int cur, int len){
if (cur==len){print_arr (arr, len); return;}
arr [cur]=0;
comb (arr,cur+1, len);
arr[cur]=1;
comb (arr, cur+1, len);
}
```