1.משתנים

טיפוס	הגדרה	גודל	תחום
תו	Char	בית	127 עד 128-
שלם	Int	2 בתים	32767 עד 32768-
ממשי	Float	4 בתים	$3.4 \bullet 10^{38}$ עד $3.4 \bullet 10^{-38}$
ממשי כפול	double	8 בתים	$1.7 \bullet 10^{308}$ עד $1.7 \bullet 10^{-308}$

.א חוקים לשמות משתנים:
- שם משתנה לא יתחיל בסיפרה.
- שם משתנה לא יכיל פעולה חשבונית.
- שם משתנה לא יכיל רווח.
- משתנה לא יקרא כשם פקודה או פונקציה.
- יש להבדיל בין אותיות קטנות לגדולות.

ב. שינוי סוג משתנה:

```
Int n=4,  m=3;
Printf ("%f \n", 1.0*n/m);
```

.ג הקצאת מס' תווים להדפסה:
הקצאת 10 מקומות להדפסת המשתנה d : %10d
הקצאת 10 תאים להדפסת f מתוכם 3 מימין לנק'
העשרונית: %10.3 f

ד. משתנה סטטי:
static סוג המשתנה שם המשתנה ;
משמש ע"מ לשמור על הערך שהמשתנה קיבל במהלך
הפונקציה, עד להפעלת הפונקציה שוב.

2. לולאות

א. פקודת if מקוצרת:
פעולת ביצוע בתנאי שקר : פעולת ביצוע בתנאי אמת ? (תנאי לוגי)

ב. לולאת for:
for (אתחול משתנה בכניסה
; תנאי לוגי ; שינוי ערך המשתנה בסוף הלולאה) {

ג. לולאה אינסופית:
1. While (1) {
2. for (;;) {
* יש להשתמש ב- break אחרת הלולאה לא תסתיים אף פעם.

ד. משפט do-while:
משפט זה יבצע את ההצהרה ורק אח"כ יבדוק תנאי.
אם התנאי יתקיים אזי הפעולה תתבצע שוב.

ה. מנגנון switch:

```
switch ( משתנה ) {
  case  1 קבוע :
    :
  Break  :
  Default :
    :
}
```

3. ספריות

שם	תאור	פקודות
<stdio.h>	ספריית קלט/פלט	Printf, scanf ,puts, gets
<stdlib.h>	ספריה סטנדרטית	Atof, atoi, atol, malloc, calloc, realloc
<string.h>	מחרוזות	Srtlen, strcpy, strcat, strcmp
<ctype.h>		Isspace
<math.h>	פונק' מתמטיות	x^y pow (x,y), Sqrt(x), sin (x)

4. מערכים

* יש לזכור שכתובת התא הראשון במערך הינה 0 (אפס) ולא 1.
* הגדרת מערך **סטטי** תעשה לפני ה- main :

```
# define   שם המערך [ גודל המערך ] ;
```

* מיון תאים במערך – מהגדול לקטן: (n =מס' התאים במערך)

```
For (j=0; j<n-1; j++){
   For (i=j+1; i<n; i++){
    If  (a[j] < a[i]){ שינוי הסימן בנק' זו יגרום למיון מהקטן לגדול
        c= a[j];
        a[j] = a[i];
        a[i ] = c;
}}}
```

* תבנית להגדרת מערך חד מימדי:

```
[גודל המערך] שם המערך  סוג המשתנים
```

* אתחול המערך כולו לאפס:

```
Int maarach [100] = {0};
```

אם נאתחל תא אחד במערך, כל הבאים אחריו יאותחלו לבד לאפס.

* אתחול מערך למס' שונה מאפס: (33 בדוגמא)

```
Int maarach [50];
Int k;
For (k=0; k<50; k++){
   Maarach [k] = 33;
}
```

* החלפת סדר איברים במערך חד מימדי:

```
For (x=num_low, y=num_high; x<y; x++, y-- ){
    Z = arr [x];
    arr [x] = arr [y];
    arr [y] = z;
 }
```

Num_low - גבול תחתון num_high - גבול עליון

5. מחרוזות

* פקודות למחרוזות נמצאות בספריה : <string.h>

א. אתחול מחרוזת בשורת ההכרזה :

```
Char macrozet [6] = "barak";
```

ב. פקודות לטיפול במחרוזות – ספריית string:
* strlen – בודקת את אורך המחרוזת ומחזירה מצביע לסופה.

```
שם המצביע = strlen ( שם המחרוזת ) ;
```

* strcpy – מעתיקה מחרוזת למחרוזת :

```
strcpy ( מחרוזת יעד , מחרוזת מקור ) ;
```

* strcat – מדביקה מחרוזת בסוף מחרוזת אחרת :

```
strcat ( מחרוזת יעד , מחרוזת מקור ) ;
```

* strcmp – משווה בין מחרוזות :

```
משתנה = strcmp( מחרוזת 1 , מחרוזת 2 ) ;
```

ג. פקודות לטיפול במחרוזות – ספריית stdio:
* puts – הדפסת המחרוזת תו אחר תו עד שתגיע ל-NULL :

```
puts ( שם המחרוזת /מערך ) ;
```

* gets – קליטת מחרוזת חדשה למערך כולל הצבת NULL בסופה.
הפונק' תחזיר 0 אם יגמר המקום במערך, עד אז היא תחזיר
1 . (גם ctrl z מפסיק את הקלט).

```
gets ( שם המחרוזת /מערך ) ;
```

ד. פקודות לטיפול במחרוזות – ספריית stdlib:
* הפונק' הבאות מבצעות המרה ממחרוזת (תווי ASCII) לסוגי מספרים שונים בהתאם :

atoi – ASCII to int : משתנה = atoi (מחרוזת) ;
atol – ASCII to long : משתנה = atol (מחרוזת) ;
atof – ASCII to double : משתנה = atof (מחרוזת) ;

ה. פקודות לטיפול במחרוזות – ספריית ctype:
* isspace – הפונק' מקבלת תו ובודקת מהו : רווח, \n, cr (enter), tab.

© ברק ספיר

במקרים הללו היא מחזירת אמת, בתווים אחרים מחזירה
שקר: isspace (תו / מצביע לתו במחרוזת) ;

ו. החלפת סדר התווים במחרוזת:

6. פונקציות

א. תבנית כללית:
{ (רשימת הפרמטרים) שם הפונקציה ערך מוחזר
* ערך מוחזר – כל סוג משתנה, או void אם הפונק' לא מחזירה כלום.
* שם הפונק' – כל שם חוקי.
* רשימת הפרמטרים – הנתונים שהפונק' צריכה לקבל ע"מ לבצע את משימתה, אם אין צורך בפרמטרים כותבים סוגריים ריקים.
* הפונקציה תסתיים בפקודת החזרת הערך המבוקש : return (__);

ב. מיקום :
*בראש התוכנית לפני הפונקציה הראשית (main).
* אם הפונק' נכתבת בסוף, חובה להכריז עליה בתחילת התוכנית.

ג. משתנים מקומיים :
* משתנים שמוגדרים בתוך הפונק' מוכרים רק לה, ומקומם בזיכרון משתחרר בסיומה וערכם מתאפס כאשר נפעיל את הפונק' שוב (אלא אם הגדרנו אותם כמשתנים סטטיים – static).

ד. משתנים גלובליים :
* משתנים שמוגדרים מחוץ לכל הפונקציות בתחילת התוכנית (אחרי הספריות). משתנים אלו מוכרים לכל הפונק' בתוכנית.

7. רקורסיה

א. תכונות :
* תמיד קיים ערך פרמטר כלשהו נתון, אשר ימנע קריאה רקורסיבית לפונקציה.
* כל קריאה של הפונקציה לעצמה תהיה עם ערך פרמטר **קטן באחד** מהערך שהתקבל.

ב. שלבים בפיתרון בעיה רקורסיבית :
1. נניח שהפונק' יודעת לטפל בבעיה מסדר n-1.
2. נרשום את תנאי העצירה שעבורו לא תהיה קריאה רקורסיבית.
3. ע"ס הנחה 1 כותבים את הפונק'. עתה צריך לבצע את שלב n.

ג. רקורסיה לסדרת פיבונצ'י :
* תוכנית זו תדפיס את כל האיברים בסדרה עד ל- limit המבוקש.

```
int fibo (int n);                    הכרזה מראש על הפונקציה -
Void main (){
    Int I, limit;.                   limit – מס' האיברים הרצויים בסדרה
   For (i=o; i<limit; i++){
      Printf ("%d \n",fibo (i));     קריאה לפונק' בתוך פק' ההדפסה
}}
int fibo (int n){
 if (n<=1) return 1;                 הגדרת שני האיברים הראשונים בסדרה
 else  return (fibo (n-1) + fibo (n-  רקורסיה
2));
}
```

ד. רקורסיה לעצרת :
* הפונק' תחזיר את סכום האיברים (העצרת של המס' המבוקש.)
* יוצאים מנקודת הנחה שה- n חיובי (במקרה זה).

```
Int azeret ( int n ){
   Int a;
   If (n!=0){
      a = n+azeret (n-1);
      return a; }
   else return 1; }
```

ה. פונק' רקורסיבית להפיכת מחרוזת :

```
char *invers (char *strung){
char temp;
if (strlen (string)<=1) return string;
temp = *string;
strcpy (string, invers (string+1));
*string + strlen (string) = temp;
For (i=0, j=strlen (s1) -1; i<j; i++, j-- ){      מחרוזת – (s1)
    c = s1 [i];
    s1 [i] = s1 [j];
    s1 [j] = c;
return string;
```

8. מצביעים

* Ptr הוא מצביע למשתנה מסוג int (מכיל את כתובת המשתנה) :
int *ptr ;

א. אתחול מצביע : int *ptr = &a ;
*מצביע ptr מאותחל לכתובת (&) של המשתנה a .
* המשתנה שכתובתו נמצאת ב- ptr מקבל את הערך 6: *ptr = 6;
* מצביע ptr מקבל את הכתובת של משתנה a: ptr = & a;
* השימוש באופרטור '*' שלא בשורת ההגדרה מציין שאנו מתייחסים **לתוכן התא** שכתובתו נמצאת ב- a .

ב. מצביעים ומערכים :
1. **שם המערך** הוא בעצם **כתובת התא הראשון במערך.**
2. במערך: int a[10] , הביטוי a+n הוא כתובתו של a כלומר &a[n] הביטוי (a+n)* הוא ערך התא a[n].
3. ההבדל בין שם מערך למצביע הינה העובדה ששם המערך הוא דבר קבוע, בעוד שאת הכתובת שמכיל המצביע אפשר לשנות.
4. כשבעבירים **מערך לפונקציה**, מעבירים לה בעצם את **כתובת התא הראשון** במערך.
5. **אתחול מערך דו-מימדי :**
סוג המשתנה שם המערך [שורות] [עמודות] =
א. {{נתוני שורה 1 }, =
{{נתוני שורה 2 }
ב. אתחול מערך char : {"מחרוזת 1 " , "מחרוזת 2"}=
מחרוזת 1 תכנס לשורה הראשונה במערך כשכל תו יקבל תא משלו וכך הלאה. בסוף כל מחרוזת יוצב NULL .

ג. מערך של מצביעים :
1. מערך של מצביעים הוא סדרה של איברים שכל אחד מהם הוא מצביע : int *ptr [3] ;
2. אפשר לאתחל מערך מצביעים בשורת ההגדרה במידה ונשתמש במחרוזות קבועות בתוכנית.
3. **אתחול מערך של מצביעים במחרוזות:**
Char *ptr[3] = { " מחרוזת 1 " , "מחרוזת 2 " , "מחרוזת 3"} ;
Ptr [0] = & מחרוזת 1 ;
Ptr [1] = & מחרוזת 2 ;
Ptr [2] = & מחרוזת 3 ;

4. כשרוצים לבצע פעולות על איברים במערך נבצע קודם אתחול של איברי המערך ונצמיד לכל איבר ונצמיד לכל איבר מצביע (ממערך של מצביעים), ואת הפעולות נבצע בעזרת המצביעים.
הגדרת המערך ואתחולו: char names [5][8]={"…","…","…"}
הגדרת מערך של מצביעים : char *ptr [5];
int I;
for (i=o; i<5; i++){
הצמדת מצביע לכל מחרוזת במערך: ptr [i] = names [i]; }
* בשלב זה אפשר להתחיל לבצע פעולות על האיברים/מחרוזות.
* ראה דוגמא : ברק7.cpp

9. הקצאת זיכרון דינאמית

* בפקודות נמצאות בספריה : stdlib.h.

א. malloc - הקצאת מקום בזיכרון :
1. תבנית כללית לשימוש בפונקציה:
משתנה מסוג מצביע = (סוג המצביע) malloc (מס' איברים * גודל כל איבר);
* דוגמא: ptr = (int *) malloc (number * sizeof (int)) ;
2. malloc מקצה מקום בזיכרון (לפי הגודל שדרשנו) ומחזירה את הכתובת של המערך שיצרה (אותו מקום בזיכרון) למצביע.

© ברק ספיר

ב. calloc – הקצאת מקום בזיכרון ואיפוסו :
1. תבנית כללית לשימוש בפונקציה:
משתנה מסוג מצביע = (סוג המצביע) calloc (מס' איברים , גודל כל איבר);
* דוגמא: ptr = (int *) calloc (number , sizeof (int)) ;
2. calloc פועלת כמו malloc בהבדל אחד- יהא גם מאפסת את כל התאים שהוקצו.

ג. בדיקת כישלון והצלחה של הקצאת הזיכרון :
1. **חובה** לבצע בדיקה האם ההקצאה הצליחה או נכשלה.
2. יש להוסיף את האלגוריתם הבא מיד לאחר שלב ההקצאה:

```
if ( המשתנה אליו הוחזרה הכתובת == NULL ) {
    Printf ("faild to allocate memory !!!");
    Return ;  }
```

ד. realloc – שינוי גודל מערך דינאמי :
* שינוי גודל המערך תוך כדי עבודה.
1. תבנית כללית לשימוש בפונקציה: משתנה מסוג מצביע = (סוג המצביע) realloc (כתובת המערך , מס' איברים * גודל כל איבר);
* דוגמא: ptr = (int *) realloc (ptr , number * sizeof (int)) ;

* אפשר להשתמש בפונק' גם בכדי להקצות זיכרון חדש (בדומה ל- malloc, calloc) ואז מציבים בתור כתובת המערך - NULL.

10. מבנים - structures

א. תבנית להגדרת מבנה: struct שם המבנה רשימת משתנים ;
דוגמא: struct car formula 1;
ב. גישה למשתנה בתוך מבנה : שם המשתנה הפנימי . שם המשתנה
הגדרת המבנה: struct car {
הגדרת השדות במבנה : long license;
char maker [20];
} ;
* אם נרצה לגשת לשדה maker במשתנה formula 1 אז נכתוב :
formula 1 . maker
ג. אתחול משתנים מסוג מבנה בשורת ההגדרה :
struct שם המבנה שם המשתנה = { ערך 1 , ערך 2 ,...} ;
ד. פונקציה ומבנה :
* פונק' יכולה **לקבל** מבנה או מצביע למבנה כאחד מהפרמטרים שלה:
void printcar (struct car details)
* פונק' יכולה **להחזיר** מבנה. אחד השימושים במקרה כזה הינה פונק' לקליטת נתונים למבנה.
ראה דוגמא – ברק8.cpp
ה. מערכים של מבנים :
הגדרת מערך של 50 מבנים : struct person group [50];
אתחול משתנה : group [10].age = 45;
* תבנית גישה לאיברי מבנה בתוך מערך:
סוג מבנה שם משתנה [אינדקס] . משתנה פנימי
ו. מבנה בתוך מבנה :
מבנה 1 :

```
Struct date{
  Char year;
}
```

מבנה 2 :

```
struct student {
struct date birthday;
}
```

קריאה למבנה 1 בתוך מבנה 2 :
* כאשר יש נתיב גישה ארוך לשדה (מבנה בתוך מבנה בתוך מבנה) :
מבנה . מבנה . ... מבנה . מבנה . מבנה . מבנה . משתנה
ז. מצביע למבנה :
הגדרת מבנה :

```
Struct bike
{
    Long
plate;
}
```

הגדרת שדה :
הגדרת משתנה bk מסוג struct bike : struct bike bk;

הגדרת מצביע מסוג struct bike ואתחולו לכתובת של bk :
struct bike *ptr = & bk;
אתחול השדה הפנימי plate של המבנה ש-ptr מצביע אליו :
ptr -> plate = 1020109;
ח. הקצאה דינאמית של מבנים :
* דוגמא- **הקצאה דינאמית** של 100 איברים מסוג struct student :
Struct student *stdarr
= (struct student *) malloc (sizeof (struct student)*100);
*דוגמא- **שינוי גודל הקצאה דינאמית** :
Stdarr =
(struct student *) realloc (stdarr , sizeof (struct student)*300);

11. רשימה מקושרת

* רשימה מקושרת היא מערך של מבנים שאין צורך להגדירו ברצף זיכרון ואין צורך לציין מראש את מס' המבנים בו.
* רשימה מקושרת מורכבת ממבנים אשר כוללים את שדות המידע וכן שדה שהוא מצביע למבנה נוסף מאותו טיפוס (המבנה הבא ברשימה). וכך מתקבלת שרשרת מבנים המקושרים ע"י מצביעים.
א. כללים:
* המבנה האחרון ברשימה מקושרת יכיל 0 בשדה " המצביע למבנה הבא ". האפס מסמן את סוף השרשרת.
* משתנה מסוג " מצביע למבנה ", המצביע למבנה הראשון יקרא **עוגן**.
דוגמא:
הגדרת המבנה: struct phonenum {
הגדרת השדות : char name [10], tel [11]
הגדרת השדה שיצביע למבנה הבא struct phonenum *nxt_phone;
} ;
הגדרת העוגן לרשימה המקושרת: struct phonenum *first_phone;
הקצאת זיכרון (לא לשכוח לבדוק הצלחת ההקצאה) : first_phone=
(struct phonenum*)malloc(sizeof (struct phonenum));
הכנסת נתונים: first_phone->name=yos, first_phone->tel=1234
הקצאת מבנה חדש לרשימה (לא לשכוח לבדוק הצלחת ההקצאה) :
First_phone->nxt_phone= malloc sizeof (struct phonenum));

ב. רשימה מקושרת דו כיוונית :
ע"מ להפוך את הרשימה המקושרת לדו כיוונית צריך רק להגדיר מבנה נוסף שיצביע על האיבר הקודם ברשימה:
הגדרת המבנה: struct phonenum {
הגדרת השדות : char name [10], tel [11]
הגדרת השדה שיצביע למבנה הבא struct phonenum *nxt_phone;
} ;
הגדרת השדה שיצביע למבנה הקודם struct phonenum *prev_phone;
} ;

12. קבצים

* הפקודות לטיפול בקבצים נמצאות בספריה stdio.h .
א. fopen ( ) - פתיחת קובץ :
* תבנית: משתנה = fopen (שם הקובץ , סוג הגישה) ;
* **משתנה** הוא משתנה מסוג מצביע למבנה FILE .
* **שם קובץ** הינו שם קובץ חוקי, מורכב משם וסיומת (כמו - data.txt) ואם צריך אז גם הנתיב לקובץ (כמו –"//dos//readme.txt").
* **סוג הגישה** – מחרוזת המציינת את סוג הפעולה שנבצע בקובץ, מורכבת משני תווים - סוג הקובץ וסוג הגישה (נכתבים בתוך גרשיים):

סוג	תו	משמעות	תנאים
קובץ	t	קובץ טקסט - txt	
	b	קובץ בינארי - bin	
גישה	r	קריאה בלבד	יש צורך בקובץ קיים אחרת התוכנית תחזיר NULL
	w	כתיבה / שכתוב	אם הקובץ לא קיים אז התוכנית תיצור אותו
	a	כתיבה בסוף המידע הקיים	

דוגמא – כדי לפתוח את הקובץ myfile.txt לקריאה כקובץ טקסט :
FILE *fp;
fp = fopen ("my file.txt" , "rt");

© ברק ספיר

* **בדיקת הצלחה** – **חובה** לבדוק שפתיחת הקובץ הצליחה :

```
fp = fopen ("my file.txt" , "rt");
if ( fp == NULL) {
   printf ("error opening file \n");
   exit (1);   }
```

* נחזיר 1 במקרה של שגיאה, ו- 0 במקרה של הצלחה.

ב. fclose () – סגירת קובץ :

* תבנית : fclose (מצביע למבנה FILE) ;

* **מצביע למבנה FILE** הינו משתנה מסוג FILE שמצביע למבנה הקובץ.

דוגמא : הגדרת מצביע למבנה מסוג קובץ : FILE *fp;

פתיחת הקובץ לקריאה : fp = fopen ("my file.txt" , "rt");

...

סגירת הקובץ : fclose (fp);

* **חובה לסגור קובץ בסוף השימוש בו !!!**

ג. fgetc () - **קריאה** של **תו** מקובץ :

* הפונק' קוראת מהקובץ תו אחר תו עד שתגיע לסוף הקובץ (EOF), בכל פעם יקודם המצביע לתו הבא.

* תבנית : משתנה = fgetc (מצביע למבנה FILE) ;

* **משתנה** הינו משתנה מסוג int, שיכיל את התו הנקרא או EOF .

* **מצביע למבנה FILE** הינו משתנה מסוג FILE שמצביע למבנה הקובץ.

דוגמא : הלולאה תקלוט ותדפיס תווים עד שתגיע לסוף הקובץ :

```
…
do {
   ch = fgetc (fp) ;          קליטת התו מהקובץ והשמתו במשתנה ch :
   putchar (ch) ;             הדפסת התו :
} while ( ch != EOF ) ;       תנאי להפסקת הלולאה – סוף הקובץ :
…
```

ד. fputc () – **כתיבה** של **תו** לקובץ :

* הפונק' מאפשרת כתיבת תו לקובץ, אחרי כל תו מצביע הקובץ יקודם באחד. כאשר נכתוב תו נוסף הוא ייכתב בסוף התו הקודם.

* תבנית : משתנה = fputc (תו , מצביע למבנה FILE) ;

* משתנה - מסוג int .

* **מצביע למבנה FILE** הינו משתנה מסוג FILE שמצביע לקובץ שאליו נכתוב.

ה. fgets () - **קריאה** של **שורה** מקובץ :

* הפונק' קוראת מהמיקום הנוכחי בקובץ ועד סוף השורה (התו \n).

* תבנית : fgets (מחרוזת , אורך מרבי , מצביע למבנה FILE) ;

* **מחרוזת** – המקום אליו קוראים את השורה מהקובץ.

* **אורך מרבי** – אורך המחרוזת המרבי שניתן לאחסן ב**מחרוזת**.

* **מצביע למבנה FILE** - יצביע על הקובץ ממנו נקרא את השורה.

* במקרה של כישלון הפונק' תחזיר NULL .

דוגמא – קריאת השורה מקובץ תוך כדי בדיקה שהקריאה הצליחה :

```
while ( fgets (machrozet1 , 35 , fp)!=NULL)
      puts (machrozet1);          הדפסת המחרוזת :
```

ו. fputs () – כתיבת שורה לקובץ :

* תבנית : fputs (מחרוזת , מצביע למבנה FILE) ;

* **מחרוזת** – הינה מחרוזת המקור שרוצים לכתוב לקובץ.

* התו \n לא מתווסף אוטומטית ולכן אם רוצים לסיים שורה יש להוסיפו לבד, אחרת השורה הבאה שנכתוב לקובץ תהיה צמודה לשורה הקודמת שכתבנו אליו.

ז. fprintf () – הדפסה לקובץ :

תבנית: fprintf (מצביע למבנה FILE , מחרוזת בקרה , ערכים);

* התבנית דומה לזאת של printf מלבד התוספת של המצביע לקובץ.

דוגמא: fprintf (fp, "the number is %d \n", num1) ;

ח. fscanf () - קריאה מקובץ :

תבנית: fscanf (מצביע למבנה FILE , מחרוזת בקרה , כתובות);

* התבנית דומה לזאת של scanf מלבד התוספת של המצביע לקובץ.

* כאשר פעולת הקריאה תיכשל, הפונק' תחזיר EOF .

13. עצים בינאריים

* עץ בינארי מורכב משורש בודד (root), צמתים (nodes) , ועלים (leaves).
* בעץ בינארי ממוין (**B.S.T**) ימצא האיבר קטן ביותר בקצה השמאלי ביותר.
* לכל צומת שני מצביעים: ימין ושמאל, שמצביעים על צמתים אחרות.
* אם מצביע לא מצביע על צומת אז הוא מצביע ל-NULL.
* עומק או גובה של עץ בינארי מציין כמה רמות יש לו.

דוגמא- הגדרת מבנה אחד בעץ בינארי :

```
struct node {                    הגדרת המבנה :
   int data;                     איבר המידע במבנה:
   struct none *left;            מצביע לצד שמאל :
   struct node *right;           מצביע לצד ימין :
}
```

14. תוכן דוגמאות

#	עמוד	נושא	תוכן
1.	1	מערכים	מיון תאים במערך
2.	1	מערכים	אתחול מערך
3.	1	מערכים	החלפת סדר איברים במערך חד מימדי
4.	1	מחרוזת	החלפת סדר התווים במחרוזת
5.	2	רקורסיה	רקורסיה לסדרת פיבונצ'י
6.	2	רקורסיה	רקורסיה לעצרת
7.	2	רקורסיה	פונק' רקורסיבית להפיכת מחרוזת
8.	2	הקצאה	בדיקת כישלון והצלחה של הקצאת זיכרון
9.	3	קבצים	בדיקת הצלחה לפתיחת קובץ
0.			
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			
9.			
0.			
1.			
2.			
3.			

© ברק ספיר