קורס יסודות התכנות בשפת C

פרק 11

הקצאה זיכרון דינאמית

Dynamic Memory Allocation

ד"ר שייקה בילו
יועץ ומרצה בכיר למדעי המחשב וטכנולוגית מידע
מומחה למערכות מידע חינוכיות, אקדמיות ומנהליות

הקצאת זיכרון דינאמית

נושאי השיעור היום:

- **תזכורת:**

  מבנים ורקורסיה.
- **הקצאת זיכרון דינאמית.**

חזרה - מבנים - הגדרת מבנה חדש

- נקודה במישור מיוצגת על-ידי שתי קואורדינטות, X ו- Y.
- הגדרת מבנה שייצג נקודה תיראה למשל כך:

```
typedef struct point
{
        double x;
        double y;
 }point;
```

- ההגדרה הזו לא תופיע בתוך פונקציה, אלא בתחילת הקובץ (בד"כ מייד אחרי שורות ה- #include וה- #define).
- כעת ניתן להגדיר בתוכנית משתנים מהסוג הזה, כפי שנבחן בהמשך.

חזרה - מבנים - הגדרת מבנה חדש

- ניתן לגשת לכל אחד מהשדות של מבנה ולכתוב/לקרוא אליו/ממנו.
- למשל:

```
int main()
{
    struct point P1,P2;
    P1.x = 6;
    P1.y = 7;
    P2.x = 4;
    P2.y = 2;
    printf("%.2lf\n", P1.y);
    return 0;
}
```

	P2
x	4
y	2

	P1
x	6
y	7

(יודפס 7)

חזרה - מבנים - הגדרת מבנה חדש


```
#include <stdio.h>
struct point {
   double x;
   double y;
};
int main()
{
  struct point P1,P2;
  P1.x = 6;
  P1.y = 7;
  P2.x = 4;
  P2.y = 2;
  printf("%.2lf\n", P1.y);
  return 0;
}
```

P2

x	4
y	2

P1

x	6
y	7

חזרה - מבנים – כתיבה מקוצרת

- שמו של הטיפוס החדש שהגדרנו הוא struct point
- אפשר לחסוך את כתיבת המילה struct באמצעות הפקודה typedef, שמאפשרת לתת שם חדש לטיפוס-משתנה:

```
struct point
{
        double x;
        double y;
};
typedef  struct point   point_t;
```

טיפוס קיים שם חדש

חזרה - עוד על typedef


```
struct complex
{
   double real;
   double image;
};
int main()
{
   struct complex c={5,7};
   struct complex *pc;
}
```

```
struct complex
{
    double real;
    double image;
    };
typedef struct complex
   complex_t;
int main()
{
   complex_t c={5,7};
   complex_t *pc;
}
```

חזרה - מצביעים ומבנים

- בדוגמא זו, כדי להגיע לשדות של P דרך ptr שמצביע על P, אפשר להשתמש ב- * כרגיל:

P.x = 3; שקול ל- (*ptr).x = 3;

P.y = 7; שקול ל- (*ptr).y = 7;

- יש צורך בסוגריים כי אחרת לנקודה יש קדימות על פני ה- *.

חזרה -מצביעים ומבנים – גישה לשדות המבנה

- בדוגמא זו, כדי להגיע לשדות של P דרך ptr שמצביע על P, אפשר להשתמש ב- * כרגיל:

P.x = 5; שקול ל- (*ptr).x = 5;

P.y = 8; שקול ל- (*ptr).y = 8;

- אבל בדרך-כלל נשתמש לצורך זה בסימון מקוצר: חץ ->

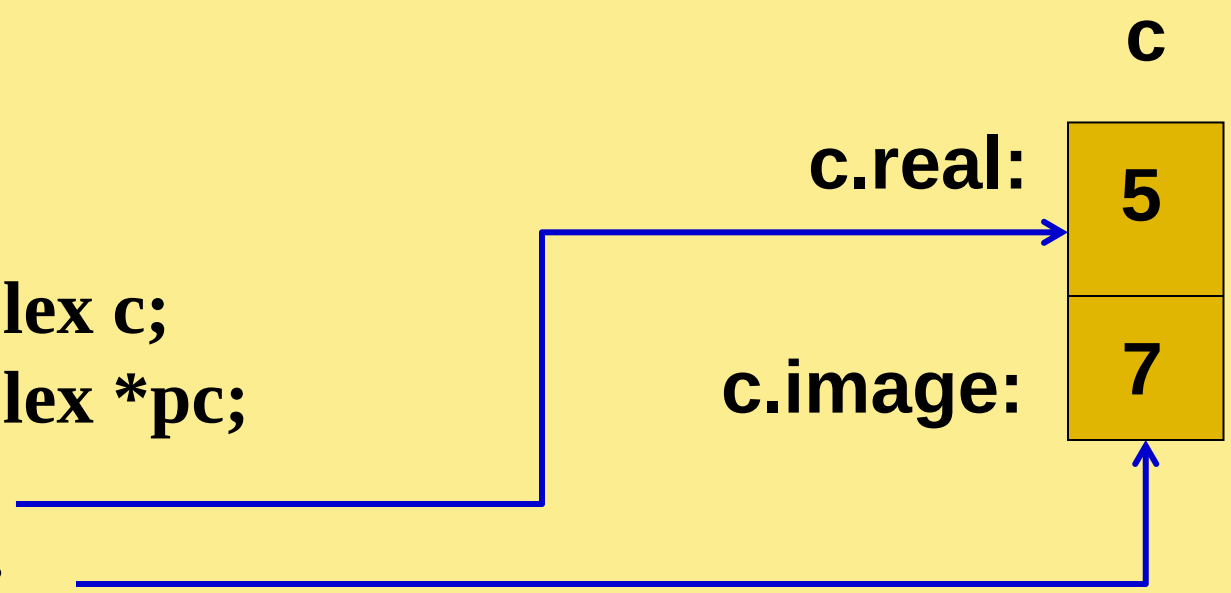
P.x = 5; שקול ל- ptr->x = 5;

P.y = 8; שקול ל- ptr->y = 8;

- כלומר משמעות החץ היא גישה לשדה במבנה שמצביעים עליו.

חזרה - מבנים Syntax - שימוש במצביעים


```
struct complex
{
    double real;
    double image;
};
int main()
{
   struct complex c;
   struct complex *pc;
   c.real = 5;
   c.image = 7;
}
```



חזרה - מבנים Syntax - כתובות ומצביעים


```
struct complex {
    double real, image;
};
int main()
{
    struct complex c;
    struct complex *pc;
    c.real = 5;
    c.image = 7;
    pc = &c;
    pc->real = 3; pc->image = 4;
}
```

pc 0x7fff9255c05c

c	0x7fff9255c05c
c.real:	5
c.image:	7
pc	0x7fff9255c05c
pc.real:	3
pc.image:	4

חזרה - מבנה בתוך מבנה

- אפשר להגדיר את זה על-ידי הקואורדינטות של שני הקודקודים:

```
struct rect
{
        double xl, xh, yl, yh;
};
```

- ברור יותר להגדיר ע"י 2 נקודות.
- נדגים כאן גם את הרישום המקוצר עם typedef:

```
struct rect {
        point_t p;
        point_t q;
};
typedef struct rect rect_t;
```

חזרה - מבנים – כתיבה מקוצרת

- שמו של הטיפוס החדש שהגדרנו היה struct point
- אפשר לחסוך את כתיבת המילה struct באמצעות הפקודה typedef, שמאפשרת לתת שם חדש לטיפוס-משתנה:

```
struct point
{
   double x;
   double y;
};
typedef struct point   point_t;
```

שם חדש טיפוס קיים

חזרה - מערך של מבנים

- כמו טיפוסי משתנים אחרים, אפשר להגדיר מערך של מבנים.
- מערך זה יוגדר ויכיל בתוכו שדות שהם מבנים הכוללים תתי שדות.
- למשל, אפשר להגדיר מערך של נקודות:

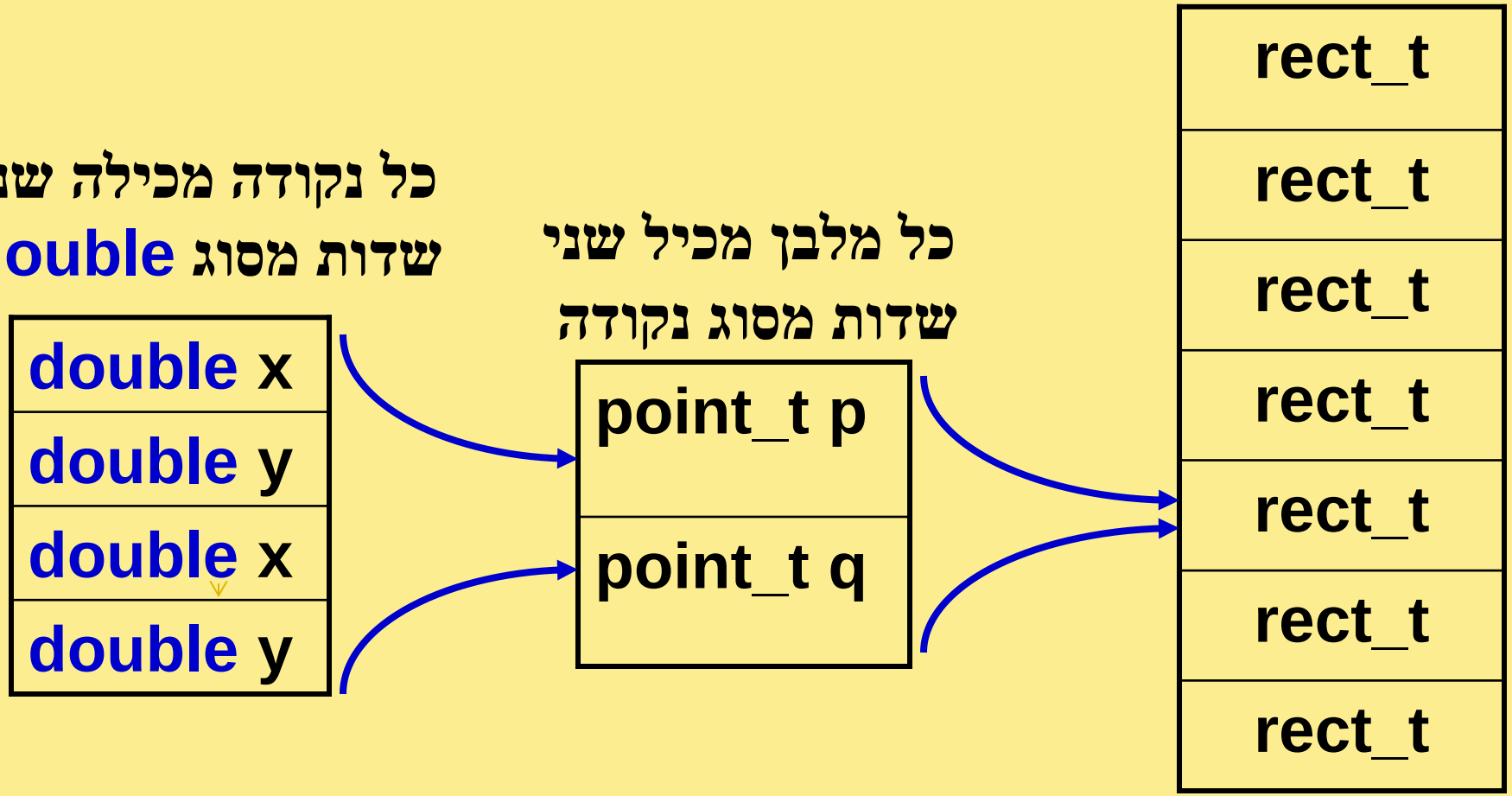
```
int main()
{
   point_t   arr[20];
   …
   …
   return 0;
}
```

חזרה - מערך של מלבנים בזיכרון

המלבנים נשמרים ברצף בזיכרון

כל נקודה מכילה שני שדות מסוג double

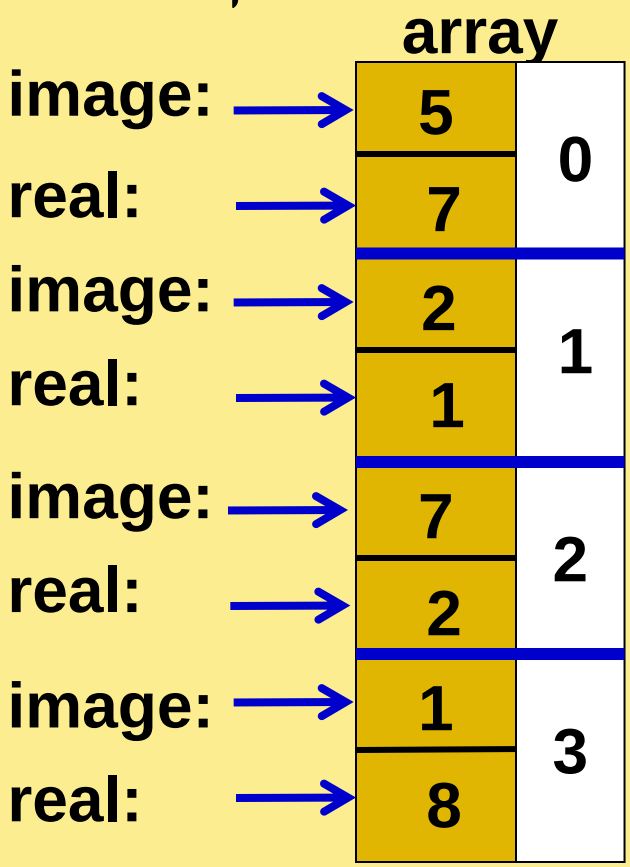
כל מלבן מכיל שני שדות מסוג נקודה



חזרה - מערכים ומבנים

- מאחר ומבנים מגדירים סוג חדש של משתנים הרי שכמו לסוגים רגילים נשאף ליצור מערכים עבור סוגי משתנים אלה.

```
complex_t arr[SIZE]=
{{5,7},{2,1},{7,2},{1,8}}
arr[1].image = 9;
arr[1].real = 2;
arr[3].image = 8;
arr[3].real = 3;
```



חזרה - מערכים ומבנים אחרי ביצוע הפקודות

complex_t arr[SIZE]= {{5,7},{2,1},{7,2},{1,8}}

arr[1].image = 9;

arr[1].real = 2;

arr[3].image = 8;

arr[3].real = 3;

שאלות?

שאלות על השעורים הקודמים?

הקצאת זיכרון דינאמית

- עד עכשיו הגדרנו איזה משתנים יהיו לנו בתוכנית כשכתבנו אותה על פי תכנון הפתרון, זה מוגדר "הקצאה סטטית" של משתנים.
- ההגדרה הייתה שכיחה במיוחד בהגדרת מערך, קבענו את טיפוס הנתונים, את גודלו, ולא יכולנו לשנות אותו בזמן הריצה (אם הגודל לא היה ידוע מראש הנחנו שיש חסם עליון).
- למשל:

```
int main()
{
        int arr[10];
}
```

- עבור המערך הזה יוקצו עשרה תאים בזיכרון, ולא נוכל להגדיל אותו תוך כדי הרצת התוכנית, כל תא יכלול ארבעה בתים (int).

הקצאת זיכרון דינאמית

- עבור המערך הזה, arr[10], יוקצו עשרה תאים בזיכרון, ולא נוכל להגדיל אותו תוך כדי הרצת התוכנית, כל תא יכלול ארבעה בתים בגלל טיפוס הנתונים int.
- לפעמים נבחר להגדיר משתנים נוספים בזמן הריצה, כתוצאה מצורך המתעורר במהלך תכנון הפתרון.
- התופעה נפוצה בפרט בעת הגדרת מערך בגודל שתלוי בקלט מהמשתמש, במערך כזה יתכן ולא נדע מראש את ערכו של החסם-עליון על גודל המערך.

הקצאת זיכרון דינאמית - צורך

- אפשר לממש זאת ב- C ע"י שימוש בפונקציה בשם: malloc(), פונקציה זו מקצה מקום בזיכרון תוך כדי ריצת התוכנית (הגודל יכול להיות תלוי בקלט ולכן הוא דינאמי).
  - הפונקציה malloc() נמצאת בספריה stdlib.h.
- מעבירים לפונקציה את גודל הזיכרון המבוקש (בבתים), והיא מחזירה מצביע לאזור בגודל הזה שהוקצה בזיכרון.
- הקצאת זיכרון בזמן הריצה נקראת "הקצאה דינאמית".

הקצאת זיכרון דינאמית - syntax


```
void main()
{
   int  *arr=NULL, size;
   printf("Enter array size\n");
   scanf("%d", &size);
    arr = (int  * ) malloc (size* sizeof(int) ) ;
}
```

- המשמעות היא: תקצה בזיכרון מקום רצוף בגודל 10 משתנים מסוג int, ותחזיר את כתובת ההתחלה של המקום שהוקצה, כמצביע ל-int אל המשתנה arr.
- ההקצאה מתבצעת בזמן ריצת התוכנית.
- הפונקציה sizeof מחזירה <u>גודל בבתים של משתנה או טיפוס משתנה</u> (10 intים דורשים בדרך-כלל 40 בתים שהם 320 סיביות).

הקצאת זיכרון דינאמית - הגדרה פורמלית

מצביע =(טיפוס וסוג המצביע) malloc (מס' תאים * גודל תא);

למשל:

```
arr = (int * ) malloc (size * sizeof(int) );
```

- ה- casting לסוג המצביע המתאים (במקרה הזה int *) נדרש כיוון שהפונקציה malloc( ) מחזירה כתובת של רצף בתים שטיפוסו לא מוגדר (void *).
- צריך להעביר את זה לטיפוס המתאים כדי לוודא שניגש נכון לערכים שנמצאים שם (בהתאם לגודל המשתנה).
- נוכל לגשת לתאי המערך כמו שניגשים אל מערך רגיל, וגם לבצע פעולות חשבון עם המצביע הזה (למשל גישה לכתובת ההתחלה ועוד 1 תביא אותנו לתא השני).

הקצאת זיכרון למערך בגודל משתנה


```
int main()
{
   int *arr=NULL, size,i;
   printf("Enter array size\n");
   scanf("%d", &size);
   arr = (int *) malloc (size * sizeof(int));
   for (i=0; i<size; i++)
         scanf("%d", &arr[i]);
   for (i=0; i<size; i++)
         printf("%4d", arr[i]);
   free(arr);
   return 0;
}
```

קולטים מהמשתמש את הגודל הדרוש

מקצים מקום בזיכרון

עכשיו אפשר לקלוט ערכים ולהדפיס אותם, כמו במערך רגיל

יש לזכור לשחרר את הזיכרון בסיום הטיפול

(אבל שימו לב ש- arr הוא מצביע ולא מערך, לכן ניתן למשל לשנות את ערכו כך שהוא יצביע אחר-כך למקום אחר)

הקצאת זיכרון דינאמית-בדיקת ההקצאה

- אם בקשת הקצאת הזיכרון נכשלת, כלומר אין מספיק זיכרון להקצאה שביקשנו, אז הפונקציה malloc מחזירה NULL.
- NULL הוא קבוע שערכו 0 שמוגדר בספריה stdlib.h.
- אחרי כל בקשת הקצאה יש לוודא שהזיכרון שביקשנו התקבל.

```
int *arr=NULL,size;
printf("Enter array size\n");
scanf("%d", &size);
arr = (int *) malloc (size * sizeof(int));
if (arr==NULL)
{
   printf("Out of memory\n");
  return 0;
}
```

הקצאת זיכרון דינאמית - malloc

- כל פונקציות ההקצאה הדינאמית נמצאות בספריה stdlib.h

```
void *malloc(unsigned int size);
```

- הפונקציה מקצה size בתים חדשים בזיכרון.
- ביצוע מוצלח של הקצאת הזיכרון יחזיר את כתובת תחילת הזיכרון המוקצה, אחרת יוחזר NULL.
- הקצאת הזיכרון הדינאמית באמצעות פונקציית ה- malloc מאפשרת תוספת זיכרון אך התאים המתווספים אינם ריקם, יש להניח כי קיים בהם זבל דיגיטלי כלשהו.

הקצאת זיכרון דינאמית - calloc

- פונקציה נוספת דומה מאוד:

```
void *calloc(unsigned int n, unsigned int size);
```

- הפונקציה מקצה מערך של n איברים, כל איבר בגודל size בתים, כל בית חדש המוקצה בזיכרון מאותחל אוטומטית לאפס.
- ביצוע מוצלח של הקצאת זיכרון יחזיר את כתובת תחילת הזיכרון המוקצה, אחרת יוחזר NULL.
- דוגמת שימוש:

```
ptr=(int *) calloc(10,sizeof(int));
```

הקצאת זיכרון למערך (מאופס) בגודל משתנה


```
int main()
{
   int *arr=NULL, size,i;
   printf("Enter array size\n");
   scanf("%d", &size);
   arr=(int *) calloc(size,sizeof(int));
   for (i=0; i<size; i++)
         printf("%4d", arr[i]);
   for (i=0; i<size; i++)
         scanf("%d", &arr[i]);
   free(arr);
   return 0;
}
```

קולטים מהמשתמש את הגודל הדרוש

מקצים מקום מאופס בזיכרון

מדפיסים את תוכן התאים שהוקצו

עכשיו אפשר לקלוט ערכים ולהדפיס אותם, כמו במערך רגיל

יש לזכור לשחרר את הזיכרון בסיום הטיפול

(אבל שימו לב ש- arr הוא מצביע ולא מערך,

לכן ניתן לשנות את ערכו כך שהוא יצביע אחר-כך למקום אחר)

דוגמה לשימוש בהקצאת זיכרון למערך (מאופס)


```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        int *arr=NULL, size,*ptr=NULL;
        printf("Enter array size\n");
        scanf("%d", &size);
        arr=(int *)calloc(size,sizeof(int));
        for (ptr=arr; ptr<=&arr[size-1]; ptr++)
                printf("%d\n",*ptr);
```

דוגמה לשימוש בהקצאת זיכרון למערך (מאופס)


```
        printf("\n");
        printf("Enter %d numbers to the array:\n",size);
        for (ptr=arr;ptr<=&arr[size-1];ptr++)
                scanf("%4d",&(*ptr));
        for (ptr=arr;ptr<=&arr[size-1]; ptr++)
                printf("%4d",*ptr);
free(arr);
return 0;
}
```

יש לזכור לשחרר את הזיכרון בסיום הטיפול

הקצאת זיכרון דינאמית - realloc

- פונקציה שימושית נוספת:

```
void *realloc(void *ptr, unsigned int size);
```

- הפונקציה מקבלת מצביע לשטח בזיכרון שהוקצה דינאמית (אותו מצביע שמחזירות malloc/calloc), ומספר בתים size (שהוא הגודל החדש הדרוש).
- הפונקציה משנה את גודל ההקצאה בהתאם לדרישה החדשה.
- אם הדרישה הייתה להגדיל את ההקצאה ואין אפשרות להגדיל את השטח הנוכחי, מוקצה שטח חליפי במקום אחר, והמידע מועתק לשם.
- קריאה מוצלחת לפונקציה תחזיר את כתובת תחילת הזיכרון המוקצה (שלא בהכרח השתנתה), אחרת יוחזר NULL.

הקצאת זיכרון (מאופס) למערך בגודל משתנה


```
int main()
{
   int *arr=NULL, size,new_size,i;
   printf("Enter array size\n");
   scanf("%d", &size);
   arr=(int *) calloc(size,sizeof(int));
   for (i=0; i<size; i++)
         printf("%4d", arr[i]);
   for (i=0; i<size; i++)
         scanf("%d", &arr[i]);
```

קולטים מהמשתמש את הגודל הדרוש

מקצים מקום מאופס בזיכרון

עכשיו אפשר להדפיס את תאי הזיכרון, לקלוט ערכים לתוכם ועוד..

הקצאת זיכרון למערך (מאופס) בגודל משתנה


```
printf("Enter new array size\n");
scanf("%d", &new_size);
arr= (int  *) realloc(arr, new_size * sizeof(int));
for (i=0; i<new_size; i++)
        scanf("%d", &arr[i]);
for (i=0; i<new_size; i++)
        printf("%4d", arr[i]);
free(arr);
return 0;
}
```

קולטים מהמשתמש את הגודל החדש הדרוש

מקצים מקום חדש לפי הגודל שנקלט החדש הנדרש

עכשיו אפשר לקלוט למערך בגודלו החדש ערכים חדשים.

עכשיו ניתן להדפיס את תאי המערך בגודלו החדש.

יש לזכור לשחרר את הזיכרון בסיום הטיפול

שאלות?

דוגמא: תוכנית עם realloc


```
int  *arr=NULL;
int size, new_size;
scanf("%d", &size);
arr = (int  *) malloc ( size * sizeof(int) ) ;
……….
…… ..
scanf("%d", &new_size);
arr=(int  *)realloc(arr, new_size*sizeof(int));
……
……
```

שחרור זיכרון שהוקצה דינאמית - free()

- בעבר הוסבר כי כשמשתנים מוגדרים בתוך פונקציה הם נעלמים אוטומטית עם סיומה.
- זה נכון רק לגבי משתנים אשר "מוגדרים סטטית".
- משתנים "המוגדרים ומוקצה עבורם זיכרון דינאמי" לא נעלמים עם סיום הפונקציה.
- באחריותנו להחליט מתי לשחרר את ההקצאה הזאת, כפי שנסביר בשקפים הבאים.

```
int main()
{
   .
   .
   func(ptr);
   .
}
```

```
void func()
{
  int *ptr=NULL,size=10;
  ptr = (int  *)malloc(size * sizeof(int));
  ....
}
```

שחרור זיכרון שהוקצה דינאמית - free()

- בכל מקרה ענין, אירוע או כל התרחשות אחרת יש לבדוק היטב את הצורך בהקצאה דינאמית. כל הקצאה דינאמית תופסת מקום בזיכרון ולכן יש לבדוק האם היא נדרשת ומה גודלה המקסימלי.
- במידה ואין צורך להמשיך להשתמש בזיכרון שהוקצה דינאמית בפונקציה, חובה לשחרר את הזיכרון הזה.
- שחרור זיכרון נעשה ע"י שימוש בפקודה: free(מצביע) (שתודגם בהמשך)
- במידה ויש צורך להמשיך להשתמש בזיכרון שהקצנו דינאמית בתוך פונקציה אחרי שהיא מסתיימת, אז היא צריכה להעביר את הכתובת של הזיכרון שהיא הקצתה למי שקרא לה.

שחרור זיכרון שהוקצה דינאמית - free()

- ללא העברת הכתובת שעבורה ביצענו את ההקצאה הדינאמית למשתנה (למצביע) לא נוכל לגשת לכתובת.
- בכל מקרה, כשמסיימים להשתמש בזיכרון חובה לשחרר אותו כך שעד סוף התוכנית צריך לשחרר כל מה שהקצינו דינאמית הן בפונקציות והן בתוכנית הראשית.
- למרות הגדלה ניכרת של הזיכרון במחשבים המיוצרים היום הניידים והנייחים עדיין יש צורך לשחרר את הזיכרון התפוס.

שחרור זיכרון שהוקצה דינאמית - free()

- במידה ולא נשחרר את הזיכרון שהקצנו וסיימנו להשתמש בו, אנחנו נשאיר בזיכרון המחשב אזורים שלכאורה תפוסים אבל בעצם לא נגישים, מה שנקרא "דליפת זיכרון".
- תופעה זו עשויה לגרום לתקיעת המחשב בגלל חוסר זמני בזיכרון.
- הפונקציה free(מצביע), נמצאת בספריה stdlib.h, מקבלת מצביע ומשחררת את הזיכרון שהוא מצביע אליו, ששמו (כתובתו) נמצא בתוך הסוגריים.

שחרור זיכרון שהוקצה דינאמית - free()

- הגדרת הפונקציה: void free(void ptr)
  - השימוש פשוט ע"י: free(ptr);
- המצביע אשר מועבר אל הפונקציה free( ) חייב להכיל את כתובת-התחלה של זיכרון שהוקצה דינאמית לפני כן.
- כתובת זו הוחזרה ע"י אחת משלוש הפונקציות:

1. malloc
2. calloc
3. realloc

- אחרי ביצוע הפקודה free(ptr) הקצאת הזיכרון שניתנה למצביע ptr מבוטלת.

שחרור זיכרון שהוקצה דינאמית

- אם ננסה לשחרר משהו אחר, למשל החל מהמקום השני שהוקצה, או כתובת של משתנה שלא הוקצה דינאמית, אז התוכנית "תעוף".

הערות לגבי פקודת realloc:

1. אם היא מקצה שטח חדש אז היא משחררת בעצמה את השטח הישן.
2. באחריות המתכנת לשחרר רק את השטח החדש שהוקצה, הכוונה לתוספת הזיכרון הדינאמי שהוקצה.

שחרור זיכרון שהוקצה דינאמית - הדגמה


```
void func()
{
  int  *arr=NULL, size=10;
  arr = (int *) malloc(size * sizeof(int));
  ....

  free(arr);
}
```

החזרת זיכרון שהוקצה דינאמית - הדגמה


```
int *func()
{
  int *arr=NULL, size=10;
  arr = (int *) malloc(size * sizeof(int));
  ....
  ....
  return arr;
}
```

דוגמה לשימוש בהקצאה דינאמית ושחרור


```
#include <stdio.h>
#include <stdlib.h>
void main()
 {
float *fptr=NULL; // הגדרת מצביע למספר ממשי
int i, size;
printf("Enter the size:\n"); //קליטת גודל המערך העתידי בזמן ריצה
scanf("%d", &size);
fptr = (float*)malloc(size*sizeof(float)); //הקצאת זיכרון למצביע
if ( fptr == NULL)  //בדיקה עם הקצאת הזיכרון הדינאמי הצליחה
      printf("Not enough memory to allocate buffer\n");
```

דוגמה לשימוש בהקצאה דינאמית ושחרור


```
for (i=0; i < size; i++)  //הכנסת ערכים לאיברי במערך
        fptr[i] = i;
                                //הצגת הערכים באיברי המערך
for(i=0; i < size; i++)
        printf("fp[%d] = %f\n",i,*(fptr+i));
                                //שחרור הזיכרון בסיום השימוש בו
free(fptr);
printf("Free fp was O.K. !!!\n");
return 0;
}
```

החזרת זיכרון שהוקצה דינאמית - הדגמה


```
int *allocate_int_array(int size)
{
   int *ptr=NULL;
   ptr = (int *) malloc(size * sizeof(int));
   if (ptr==NULL)
   {
      printf("Out of memory");
      return 0;
   }
   return ptr;
}
```

הפונקציה מקצה דינאמית מערך של מס' שלמים בגודל המבוקש, בודקת שההקצאה הצליחה, ומחזירה את כתובת המערך שהוקצה.

אפשר להחזיר את כתובת המערך הזה, שלא משוחרר בסוף הפונקציה כי הוא הוקצה דינאמית, אבל מי שקרא לפונקציה צריך לדאוג לשחררו בסיום השימוש בו.

שאלות?

חזרה על מצביעים, והקצאת זיכרון דינאמי

מצביעים לא מאותחלים

- חשוב לזכור, שכמו כל משתנה גם מצביעים אינם מאותחלים בהגדרה שלהם, וצריך לתת להם ערך התחלתי.
- במידה וננסה לגשת לכתובת שנמצאת במצביע בלי שנתנו לו ערך התחלתי, התוכנית "תעוף".
- למשל:

```
int *ptr;
*ptr=10;
```

הכתובת של - ptr לא אותחלה !!!

ולכן לא ניתן לבצע לה השמה

מצביעים לא מאותחלים

- ניתן לגשת לכתובת שנמצאת במצביע רק אם שמנו בו קודם כתובת של משתנה או של שטח שהוקצה דינאמית וזאת ע"י:

```
int i, size=10;
int *ptr=NULL;
ptr=&i;
```

וע"י הגדרה כזו:

```
ptr=(int *) malloc (size*sizeof(int));
```

מצביעים לא מאותחלים

- כזכור, אפשרות נוספת שראינו למתן ערך למצביע היא ע"י הכנסת כתובת של מערך או של מצביע אחר אליו.

```
int arr[10];
int *ptr=NULL;
ptr=arr;      //ptr=&arr[0];
```

- הפעולה תעבוד באופן תקין בהנחה שהמצביע ptr הוגדר לפני שתי השורות הנ"ל ע"י ביצוע הפקודה:

```
int *ptr=NULL;
```

- ההגדרה הנ"ל מאפשרת השמת כתובת בתוך ptr שהוא החלק במצביע המסוגל לקבל כתובת כאשר *ptr מוגדר כמצביע.

מחרוזת קבועה

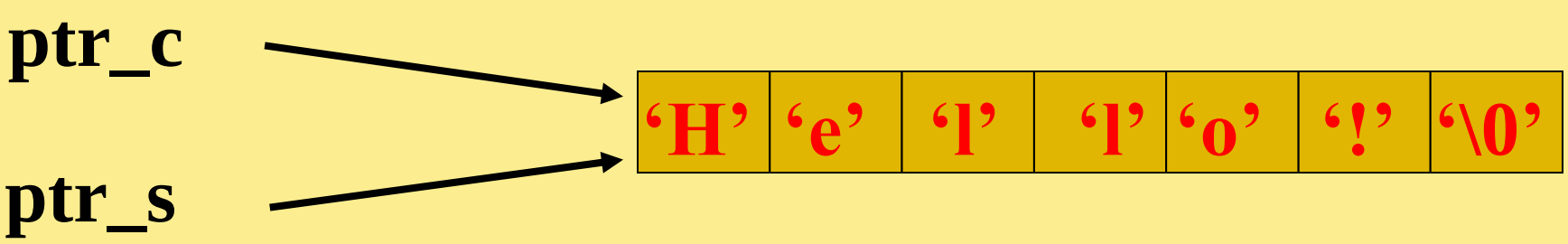
- במידה ונציב מחרוזת קבועה למצביע על char בלי לאתחל אותו (לא מחרוזת מהקלט) אז לא תתרחש "תעופה" של התוכנית, אפילו שלא בוצעה הקצאה כלשהי ואפילו שלא ניתן להעתיק מחרוזות על-ידי השמה.
- זה מתרחש כיוון שזו בעצם ביצוע השמה של כתובות:

כיוון שהקומפיילר שומר בזיכרון כל מחרוזת קבועה שהוא מוצא בתוכנית, למחרוזת הזאת כבר יש כתובת כלשהי בזיכרון, כתובת שבה היא מאוחסנת ולכן המצביע רק מצביע לשם.

מחרוזת קבועה

- למשל אפשר לכתוב:

```
char *ptr_c=NULL, *ptr_s=NULL;
ptr_c="Hello!";
ptr_s="Hello!";
```



- המשמעות היא ששני המצביעים מצביעים על אותה מחרוזת.
- קומפיילרים רבים לא יאפשרו לשנות מחרוזת כזו.
- כל שינוי כזה עשוי להשפיע על כל הופעות המחרוזת בתוכנית.

שאלות?

הקצאת זיכרון דינאמית למערך של מבנים

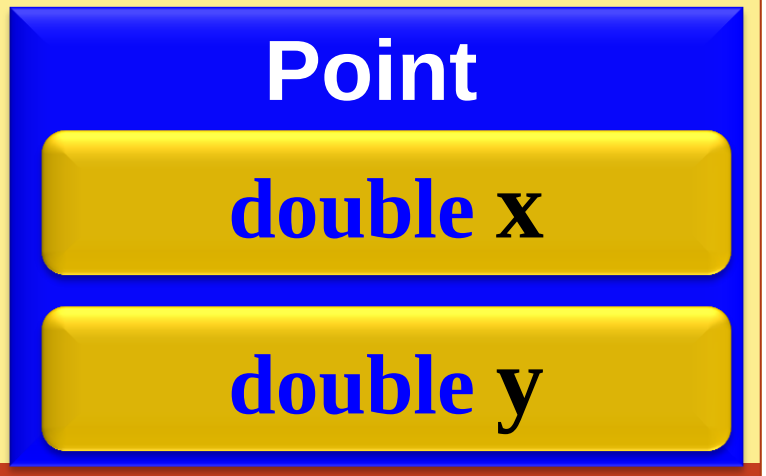
ניתן לבצע הקצאת זיכרון דינאמית למערך של מבנים וזאת על ידי הקצאת זיכרון למצביע שהוא מטיפוס מבנה:

- שלב ראשון – יצרת מבנה חדש
- שלב שני – הגדרת מצביע מטיפוס המבנה החדש
- שלב שלישי – הקצאת זיכרון דינאמי למצביע
- שלב רביעי – קליטת נתונים למערך החדש שנוצר ע"י הקצאה דינאמית ולא ע"י גודל סטטי והדפסתם.

שלב ראשון – הגדרת מבנה

- נגדיר מבנה של מספר מורכב:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct {
        double x, y;
}Point;
```



שלב רביעי - הפעלת פונקציות לקלט ופלט


```
void Fillarray(Point *arr, int size)
{
    int i;
    printf("\nEnter %d peers of doubles:\n",size);
    for(i=0;i<size;i++)
        scanf("%lf %lf",&arr[i].x,&arr[i].y);
}
void Printarray(Point *arr, int size)
{
    int i;
    for(i=0;i<size;i++)
        printf("X=%lf \tY=%lf",arr[i].x,arr[i].y);
}
```

שלב שני ושלישי הגדרת מצביע והקצאת זיכרון


```
void main()
{
   Point *arr=NULL;
   int size,i;
   printf("Enter size for array\n");
   scanf("%d", &size);
   arr =(Point *) malloc (size*sizeof(Point));
   Fillarray(arr,size);
   Printarray(arr,size);
   free(arr);
}
```

שאלות?

הקצאת זיכרון דינאמית - סיכום

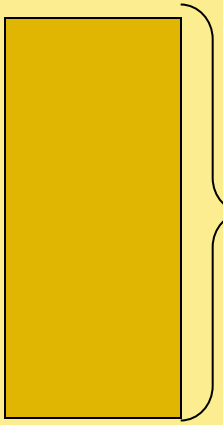
- הזיכרון המקסימאלי ששימש את התוכנית שלנו עד היום היה קבוע מראש על פי הגדרת טיפוסי וסוגי המשתנים בתוכנית.
- לפני הרצת התוכנית, לאחר שהתוכנית עברה את שלב התרגום, כל המערכים שבתוכנית גודלם נקבע ואינו ניתן לשינוי במהלך ריצת התוכנית!
- <u>בעיות אפשריות:</u>

1. אם הגדרנו מערך בגודל 100 לשמור את ציוני כל הסטודנטים וכעת מגיע סטודנט נוסף מה נעשה?
2. התוכנית כבר כתובה ולא ניתן לשנות את גודל המערך!
3. נצטרך פתרון יצירתי המסוגל לייצר עוד תאים על פי דרישה.

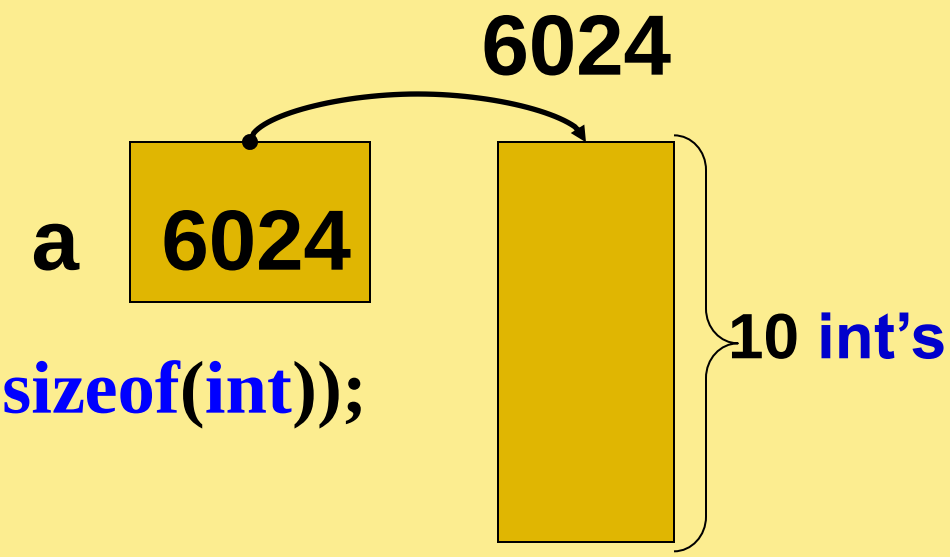
הקצאת זיכרון דינאמית - סיכום

שפת C מאפשרת לנו להקצות זיכרון תוך כדי ריצת התוכנית וכך להגדיר מערכים ומבנים אחרים בצורה דינאמית:

```
int main()
{
    int arr[10];
    arr[6] = 7;
}
```



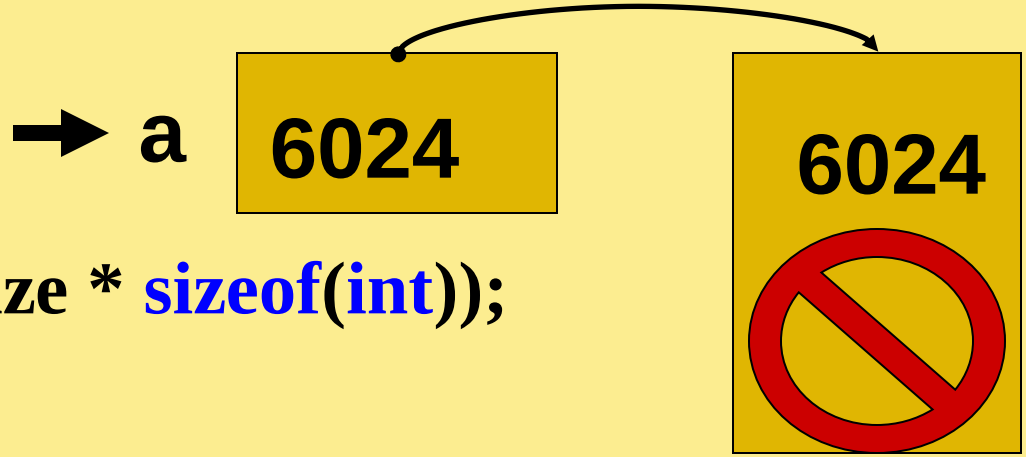

```
int main()
{
    int *arr,size=10;
    arr = (int*) malloc(size * sizeof(int));
    arr[6] = 7;
}
```



הקצאת זיכרון דינאמית - סיכום

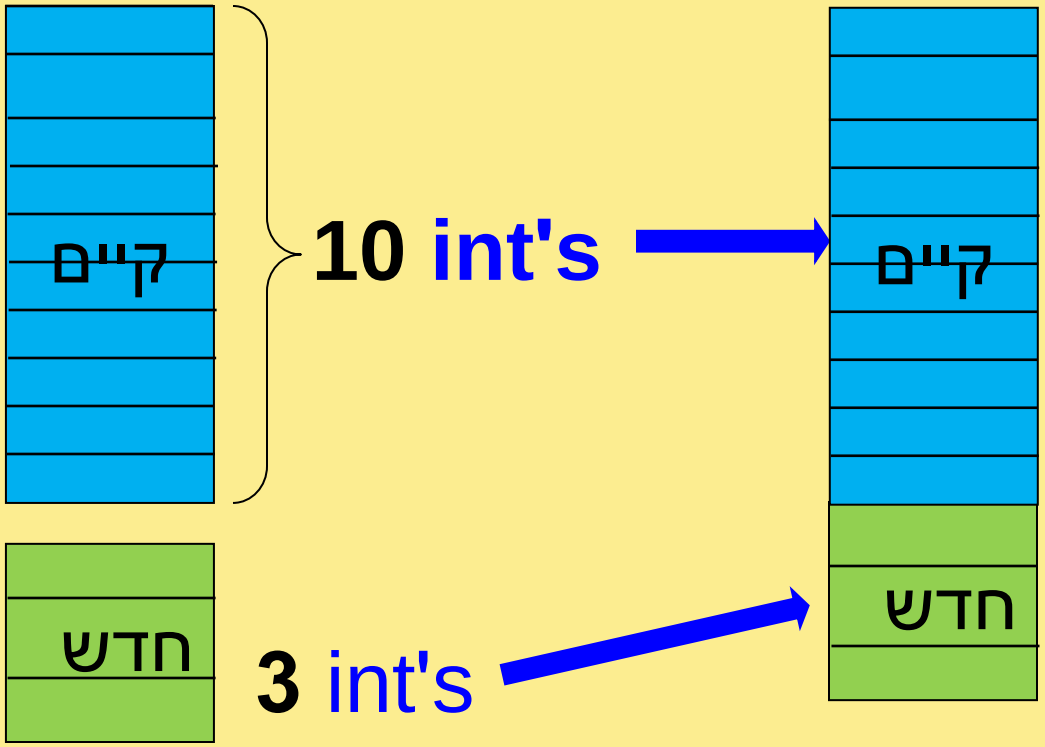
- כל שטח זיכרון שהוקצה ע"י המתכנת צריך להיות משוחרר בסיום השימוש בו.
- לכן חייבים תמיד להחזיק את כתובת האזור שהוקצה על מנת שנוכל לשחרר אותו בסיום השימוש.

```
int main()
{
    int *arr,size=10;
    arr = (int*) malloc(size * sizeof(int));
    arr[6] = 7;
    free(a);
}
```



הקצאת זיכרון דינאמית - סיכום

- במערך אם נרצה להוסיף איבר חדש נוסף נצטרך להעתיק את כל האיברים הישנים למערך גדול יותר אשר הקצנו.
- נקצה מערך חדש ונעתיק אליו את המערך הישן.



הקצאת זיכרון דינאמית - סיכום

הפונקציות נמצאות ב- stdlib.h:

```
void *malloc(size * sizeof( ));
```

- הקצאת בתים בגודל נדרש בזיכרון באופן דינאמי.
- קריאה להקצאה מוצלחת מחזירה כתובת בסיס חדשה, אחרת מוחזרת כתובת NULL.
- מאפשרת שימוש בכתובת שהוחזרה ככתובת להתחלת מערך חד ממדי או דו ממדי.

הקצאת זיכרון דינאמית - סיכום

הפונקציות נמצאות ב- stdlib.h:

void *calloc(size, sizeof( ))

- הקצאת מערך של אלמנטים כל חד בגודל של size_el
- כל תא שהוקצה בזיכרון מאותחל לאפס.
- קריאה להקצאה מוצלחת מחזירה כתובת בסיס חדשה, אחרת יוחזר NULL.
- מאפשרת שימוש בכתובת שהוחזרה ככתובת להתחלת מערך חד ממדי או דו ממדי שערכך מאופסים.

הקצאת זיכרון דינאמית - סיכום

הפונקציות נמצאות ב- stdlib.h:

```
void free(void *ptr)
```

- מבטל את הקצאת הזיכרון.
- מקיים מצביע על המיקום בו הוקצה הזיכרון הקודם.
- משחרר/מוחק את הכתובת הקודמת הנמצאת במצביע.
- מונע אפשרות גישה לכתובת המערך שהוקצתה למצביע המשוחרר.
- מאפשרת שימוש חוזר בכתובת זו לצורכי הקצאת זיכרון עתידית.

הקצאת זיכרון דינאמית - סיכום

הפונקציות נמצאות ב- stdlib.h:

```
void *realloc(void ptr, newsize * sizeof())
```

- מקצה מחדש מקום בזיכרון לצורך שינוי גודל קודם.
- המקום החדש בזיכרון הוא על פי הגודל של newsize והחל מהמצביע ptr.
- מאפשרת הגדלה או הקטנה של כמות התאים בזיכרון אשר מוקצים דינאמית החל מכתובת מצביע התחלתי.
- למצביע ההתחלתי נעשתה הקצאת זיכרון קודמת ע"י malloc או calloc.

הקצאת זיכרון דינאמית למערך חד ממדי

ספריות והצהרות על פונקציות

```
#include <stdio.h>
#include <stdlib.h>

void FillArray(int arr[], int size);

void PrintArray(int arr[], int size);
```

דוגמא – הקצאת זיכרון דינאמית


```
int main()
{
    int *arr, size;
    printf("Enter the size:\n");
    scanf("%d", &size);
    arr = (int *)calloc(size, sizeof(int));
    FillArray(arr, size);
    PrintArray(arr, size);
    free(arr);
    return 0;
}
```

תוכנית
ראשית

דוגמא – הקצאת זיכרון דינאמית


```
void FillArray(int arr[], int size)
{
    int i;
    for(i = 0; i < size; i++)
    {
        arr[i] = rand()%100 + 1;
    }
}
```

מילוי המערך החד ממדי
במספרים אקראיים

דוגמא – הקצאת זיכרון דינאמית


```
void PrintArray(int arr[], int size)
{
    int i;
    for(i = 0; i < size; i++)
    {
        printf("%3d ", arr[i]);
        if (i % 10 == 0)
            printf("\n");
    }
}
```

הדפסת המערך

החד ממדי

שאלות?

קליטת מספר לא ידוע של תווים

- כתוב תוכנית המקבלת קלט מהמשתמש רצף של תווים שאורכו אינו ידוע מראש המסתיים בלחיצה על Enter.
- התוכנית תדפיס את אוסף התווים שנקלט מההתחלה לסוף.
- התוכנית תדפיס את אוסף התווים שנקלט מהסוף להתחלה.
- רמז: יש להקצות דינאמית בכל סיבוב מקום לתו הנקלט ויש לוודא סגירת המחרוזת ע"י השמת '\0' בסוף הקלט.
- אין, בשום מצב, להשתמש ברקורסיה!!!.

קליטת מספר לא ידוע של תווים-פתרון


```
#include <stdio.h>
#include <stdlib.h>
void main()
{
        char *input=(char *)malloc(1),ch;
        int size=1,current=0;
        scanf("%c",&ch);
```

קליטת מספר לא ידוע של תווים-פתרון


```
while (ch != '\n')
{
        if (current  + 1 == size)
            input = (char *)realloc(input, ++size);
//הגדלת הקצאת הזיכרון באחד כל סיבוב והצבת התו במקום המוקצה
        input[current++] = ch;
        scanf("%c",&ch);
} //while
```

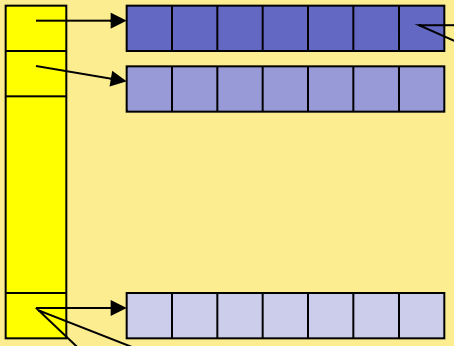
קליטת מספר לא ידוע של תווים-פתרון


```
    input[current]='\0';
    puts("Original String:");
    puts(input);
    puts("Reverse String:");
    puts(strrev(input));
    free(input);
}
```

הקצאת זיכרון דינאמית למערך דו ממדי

ייצוג מערך דו-ממדי כמערך של מערכים:

Matrix :



מערכים חד ממדים המייצגים אוסף של עמודות המהווה שורה

מערך של מצביעים – כאשר כל מצביע בעצם מצביע לשורה המתאימה שהחל ממנה מתחילות העמודות המרכיבות את השורה

הגדרת מערך דו ממדי בהקצאת זיכרון

- כתוב תוכנית הקולטת נתונים מספריים למטריצה דו ממדית בגודל דינאמי שמקבל הקצאת זיכרון, קלט נתונים, פלט נתונים.

```
#include <stdio.h>
#include <stdlib.h>
```

ספריות והצהרות על פונקציות

```
void FillArray(int **arr, int size);

void PrintArray(int **arr, int size);
```

הגדרת מערך דו ממדי בהקצאת זיכרון

תוכנית ראשית

```
int main()
{
        int **arr, size,i;
        printf("Enter the size:\n");
        scanf("%d", &size);
        arr = (int **)calloc(size, sizeof(int));
        for (i=0;i<size;i++)
                *(arr+i)=(int *)calloc(size,sizeof(int));
        FillArray(arr, size);
        PrintArray(arr, size);
        free(arr);
        return 0;
}
```

פונקציית קלט הנתונים למטריצה


```
void FillArray(int **arr, int size)
{
   int i,j;
   for(i = 0; i < size; i++)
   {
         for(j = 0; j < size; j++)
         {
               *(*(arr+i)+j)= rand()%100 + 1;
         }
   }
}
```

קלט המערך
הדו ממדי

פונקציית הדפסת הנתונים של המטריצה


```
void PrintArray(int  **arr, int size)
{
    int i,j;
    for(i = 0; i < size; i++)
    {
        for(j = 0; j < size; j++)
            printf("%3d ", *(*(arr+i)+j));
        printf("\n");
    }
}
```

פלט המערך הדו ממדי

שאלות?

הקצאת זיכרון דינאמית - סיכום

- אפשר להקצות מקום בזיכרון בזמן הריצה.
- אפשר ליצור מערך בגודל שתלוי בקלט (גודל שלא ידוע מראש).
- בספריה stdlib.h נמצאות הפקודות:

malloc, calloc, realloc, free().

- כשמקצים מקום בזיכרון מוחזרת הכתובת שלו, הנשמרת במצביע (בהתאם לסוג המשתנה שיבחר לשמירה במקום הזה).

הקצאת זיכרון דינאמית - סיכום

- אי-אפשר לתת שם למשתנה/מערך החדש, אבל אפשר לגשת אליהם דרך המצביע.
- צריך לזכור לשחרר זיכרון שהוקצה כשמסיימים את השימוש בו .
- כתיבה לכתובת שנמצאת במצביע לא מאותחל תגרום תמיד ל- "תעופה" – צריך לזכור לאתחל אותו.
- מצביע אשר מיועד להצביע על שדה מסכם ניתן לאפס ע"י השמת הערך: NULL.

שאלות?

תרגילי כיתה

1. כתוב תוכנית המחשבת את סכום של n אלמנטים שהוזנו על ידי משתמש. לביצוע תכנית זו, עליך להקצות זיכרון דינמי באמצעות פונקציית ה- malloc().
2. כתוב תוכנית המחשבת את סכום של n אלמנטים שהוזנו על ידי משתמש. לביצוע תכנית זו, עליך להקצות זיכרון דינמי באמצעות פונקציית ה- calloc().
3. כתוב תוכנית הנותנת מענה למצב בו אין למתכנת מושג לגבי האורך של הטקסט הנקלט שצריך לאחסן במחרוזת. במקרה זה עליך להגדיר מצביע מבלי להגדיר כמה זיכרון נדרש ולאחר מכן על בסיס דרישה יוקצה זיכרון שאליו תיכנס המחרוזת הנקלטת. בסיום הקליטה תודפס המחרוזת שנקלטה.

תרגילי כיתה

4. כתוב עדכון לתוכנית הקודמת המטפל במצב בו יש צורך לשנות את הגודל לאחר הקליטה כלומר לבצע הגדלה או הקטנה של הזיכרון ע"י שימוש בפונקציה realloc().

5. כתוב תוכנית המייצרת מערך דו ממדי של מספרים שלמים המקבל הקצאת זיכרון ע"י שימוש במצביע למצביע. אחר כך התוכנית תמלא את המערך שנוצר במספרים אקראיים ותדפיס אותו, והכל בשימוש במצביעים בלבד!!!.

6. כתוב תוכנית הקולטת אינסוף מספרים שלמים למערך חד ממדי ומחשבת את סכום כל המספרים, את הממוצע את הגדול והקטן ביותר מהמספרים שנקלטו. לאחר קליטה של 4 מספרים התוכנית מגדילה את הזיכרון המוקצה פי שתיים. לסיום יש להקליד 1-.

תרגילי כיתה

7. כתוב תוכנית המבצעת את המשימות הבאות:

a) מבצעת הקצאת זיכרון דינאמי למערך דו ממדי/חד ממדי.

b) מבצעת הכנסה של מספרים אקראיים בין 1 ל- 1000 למערך הדו ממדי ע"י שימוש בפונקציה.

c) מבצעת בדיקה האם במערך שנוצר קיימים מספרים משוכללים ומדפיסה אותם ע"י שימוש בפונקציה.

d) מבצעת הדפסה של המערך הדו ממדי ע"י שימוש בפונקציה.

מספר משוכלל הוא מספר טבעי השווה לסכום כל המחלקים הטבעיים שלו מלבד המספר עצמו.

המספר המשוכלל הראשון הוא 1+2+3=6, ואחריו באים 1+2+4+7+14=28, 496 ו-8128.

תרגילי כיתה

8. כתוב תוכנית המדגימה שימוש במערך דו-ממדי דינאמי מחרוזתי. תזכורת: כשרוצים להגדיר מערך דו ממדי בעל גודל משתנה, עלינו ליצור מערך של מצביעים למערכים. גודל המערך לא ידוע בתחילת התכנית ולכן נגדיר את גודל המערך באמצעות פונקציית: malloc( ) או calloc( )

בתחילה נאתחל מערך של מצביעים, ולאחר מכן נאתחל כל מצביע להיות מערך של ה- type הרצוי.

למשל, בתרגיל זה עלינו לייצר מערך דו ממדי של char-ים בגודל המוגדר ע"י המשתמש כך שבכל תא תופיע אות אקראית מ- A עד Z או a עד z.

רמז, יש להשתמש בערכי כתובות התווים מטבלת ה- ascii.

תרגילי כיתה

9. כתוב תוכנית המבצעת הקצאת זיכרון דינאמית למערך דו ממדי. פונקציה נוספת תמלא את המערך במספרים אקראיים בין 1 ל-100. כתובת המערך וגודלו יועברו לפונקציה בדיקה נוספת וזאת יחד עם 2 מספרים. הפונקציה, פונקציית הבדיקה, תחשב ותדפיס את כמות האיברים במערך המתחלקים במספר הראשון ללא שארית ואת כמות האיברים המתחלקים במספר השני ללא שארית.

דוגמא:

עבור המערך {{7,1,2},{3, 4, 5},{6, 7, 8,},{9, 10, 11}} והמספרים 2 ו-3, הפונקציה תחזיר שעבור המספר 2, 5 ערכים מתחלקים בו (2, 4, 6, 8, 10) ועבור המספר 3 תחזיר הפונקציה ש- 3 מספרים מתחלקים בו (3,6,9).

תרגילי כיתה

10. כתוב תוכנית הכוללת פונקציה המקצה מערך חד ממדי לקליטת ציונים, קולטת אליו ציונים מחשבת ומדפיסה את הממוצע. אחר-כך משנה את גודל המערך, לפי בקשת המשתמש. לאחר כל שינוי בגודל וקריאת ציונים חדשים, הממוצע יחושב מחדש וכן יחושב ויודפס כמה ציונים היו מעל הממוצע.

<u>הערות לתוכנית</u>:

מומלץ להשתמש בלולאת **do..while** המאפשרת לשנות את גודל המערך כמה פעמים שרוצים.

מומלץ להגדיר פונקציית קלט נפרדת, פונקציית פלט נפרדת ופונקציית חישוב נפרדת.

שאלות?