הקצאות דינאמיות

קרן כליף

ביחידה זו נלמד:

- מוטיבציה להקצאות דינאמיות
- מהי הקצאה דינאמית
- יצירת מערך בגודל שאינו ידוע מראש
- החזרת מערך מפונקציה
- הקצאת מערך של מערכים
- הגדלת מערך
- הפונקציה strdup

מוטיבציה להקצאה דינאמית

- בעזרת הקצאה דינאמית נוכל:
  - להקצות מערך בזמן ריצה, בלי לדעת את גודלו בזמן קומפילציה
  - נוכל להחזיר מערך מפונקציה

מהי הקצאה דינאמית?

- הקצאה דינאמית היא הקצאת שטח זיכרון בגודל מבוקש בזמן ריצת התוכנית
  - בניגוד להקצאה סטטית שמוקצית בתחילת התוכנית וגודלה ידוע כבר בזמן קומפילציה
- הקצאה דינאמית מוקצית על שטח הזיכרון heap
  - בניגוד להקצאה סטטית שמוקצית על ה- stack של הפונקציה
  - ה- heap הוא שטח זיכרון המשותף לכל הפונקציות, בניגוד ל- stack
- בהקצאת זיכרון דינאמית המתכנת מבקש ממערכת ההפעלה זיכרון בגודל מסוים המוגדר בבתים ומקבל את כתובת הבית הראשון בקטע הזיכרון שקיבל

הגדרה נוספת ל- *<type>

- אנו יודעים שמשתנה מטיפוס *<type>מכיל כתובת של משתנה מטיפוס <type> כלשהו
- כתובת התחלה של מערך מטיפוס <type>היא גם כתובתו של האיבר הראשון במערך, שהוא משתנה מטיפוס <type>
- לכן משתנה מטיפוס *<type> יכול להכיל גם כתובת התחלה של מערך
- לכן נאמר ש- *<type> הוא פוטנציאל למערך
  - כלומר, יכול להכיל כתובת התחלה של מערך (ולא רק כתובת של משתנה יחיד מטיפוס <type>)

פונקציות לשימוש בהקצאות דינאמיות

- כדי לעבוד עם הקצאות דינאמיות יש להכליל את הספריה stdlib.h לצורך שימוש בפונקציות הבאות:
- void* malloc(size_t size)
  - הפונקציה מקבלת את גודל הזיכרון המבוקש בבתים ומחזירה כתובת לתחילת שטח הזיכרון שהוקצה, ו-NULL במקרה שלא היה מספיק זיכרון פנוי להקצאה
    - נשים לב ש- size_t הוא typedef ל- int, כלומר הפונקציה מקבלת את מספר הבתים שעליה להקצות
    - הפונקציה מחזירה void* שזה הכללה למצביע מכל טיפוס
  - שטח הזיכרון המתקבל מכיל זבל
- void* calloc(size_t n,size_t size_el)
  - כמו malloc אבל מקבלת את כמות האיברים שרוצה להקצות, ומה הגודל של כל איבר. מקצה את השטח ומאפסת אותו.
  - מקצה מערך של n איברים כל איבר בגודל size_el בתים, כל ביט מאותחל לאפס. קריאה מוצלחת תחזיר את כתובת ההתחלה של הזיכרון המוקצה , ו-NULL במקרה שלא היה מספיק זיכרון פנוי להקצאה

malloc – הקצאת מערך בגודל שאינו ידוע מראש

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

void main()
{
    int size, *arr, i;

    printf("Please enter the size of the array: ");
    scanf("%d", &size);
    arr = (int*)malloc(size*sizeof(int));

    if (!arr) // (arr == NULL) --> allocaton didn't succeed
    {
      printf("ERROR! Out of memory!\n");
      return;
    }

    printf("Values in the array: ");
    for (i=0 ; i < size ; i++)
      printf("%d ", *(arr+i)); //*(arr+i) == arr[i]

    printf("\nPlease enter %d numbers: ", size);
    for (i=0 ; i < size ; i++)
      scanf("%d", &arr[i]);

    printf("Values in the array: ");
    for (i=0 ; i < size ; i++)
      printf("%d ", arr[i]);
    printf("\n");
}
```

לצורך שימוש בהקצאות דנאמיות

נשים לב שלמשתנה על ה- heap אין שם,
אלא רק יש אליו הצבעה מאחת
הפונקציות!

int	1	3000
int	5	3004
int	8	3008

זיכרון ה- heap

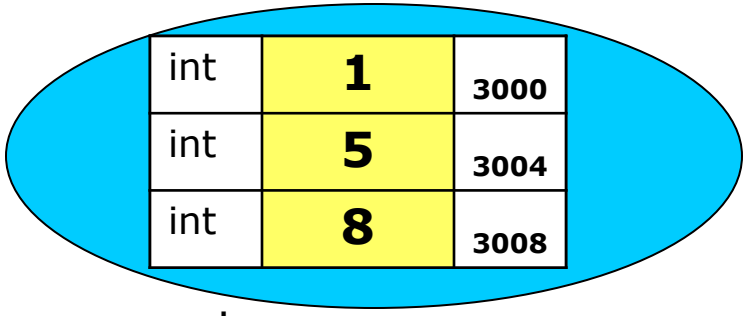
int:size	3	1000
int*: arr	3000	1004
int: i	???	1008

הזיכרון של ה- main

```
C:\WINDOWS\system32\cmd.exe
Please enter the size of the array: 3
Values in the array: -842150451 -842150451 -842150451
Please enter 3 numbers: 1 5 8
Values in the array: 1 5 8
Press any key to continue . . .
```

calloc – הקצאת מערך בגודל שאינו ידוע מראש

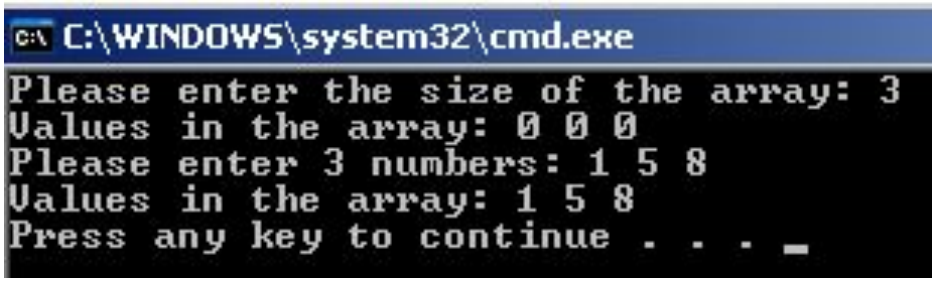
```
#include <stdio.h>
#include <stdlib.h>

void main()
{
    int size, *arr, i;

    printf("Please enter the size of the array: ");
    scanf("%d", &size);
    arr = (int*)calloc(size, sizeof(int));

    if (!arr) // (arr == NULL) --> allocaton didn't succeed
    {
      printf("ERROR! Out of memory!\n");
      return;
    }

    printf("Values in the array: ");
    for (i=0 ; i < size ; i++)
      printf("%d ", *(arr+i)); //*(arr+i) == arr[i]

    printf("\nPlease enter %d numbers: ", size);
    for (i=0 ; i < size ; i++)
      scanf("%d", &arr[i]);

    printf("Values in the array: ");
    for (i=0 ; i < size ; i++)
      printf("%d ", arr[i]);
    printf("\n");
}
```



זיכרון ה- heap

int:size	3	1000
int*: arr	3000	1004
int: i	???	1008

הזיכרון של ה- main



שחרור הזיכרון שהוקצה

- אחריות המתכנת לשחרר את כל זיכרון שהוקצה דינאמית
- במילים אחרות, המתכנת אחראי להחזיר למערכת ההפעלה כל שטח זיכרון שביקש ממנה בזמן ריצה
- הסיבה:
  - שטח ה- heap עליו מוקצות ההקצאות הדינאמיות מוגבל בשטחו ומשותף לכל התוכניות, ואז התוכנית הבאה שתבקש זיכרון עלולה לקבל NULL כי אנחנו לא החזרנו את הזיכרון שביקשנו בסיום העבודה...
  - <u>צריך לזכור:</u> כאשר יש סביבת עבודה משותפת, צריך להתחשב גם באחרים (וזה שיעור חשוב בכלל לחיים ;-))
- הקומפיילר לא מתריע על אי-שחרור הזיכרון ואין שום אינדיקציה לדעת זאת, לכן חייבים לשים לב!!

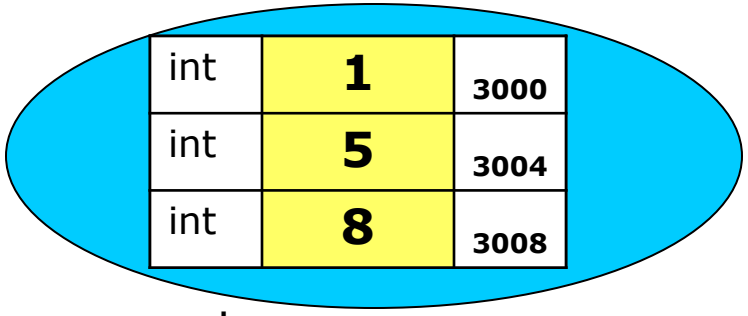
פונקציה לשחרור הקצאות דינאמיות

void free(void* ptr)

- הפונקציה מקבלת את כתובת ההתחלה של הזיכרון שברצוננו לשחרר
  - הפונקציה מקבלת void* שזה הכללה למצביע מכל טיפוס
- גם פונקציה זו נמצאת ב- stdlib.h

free – שחרור זיכרון

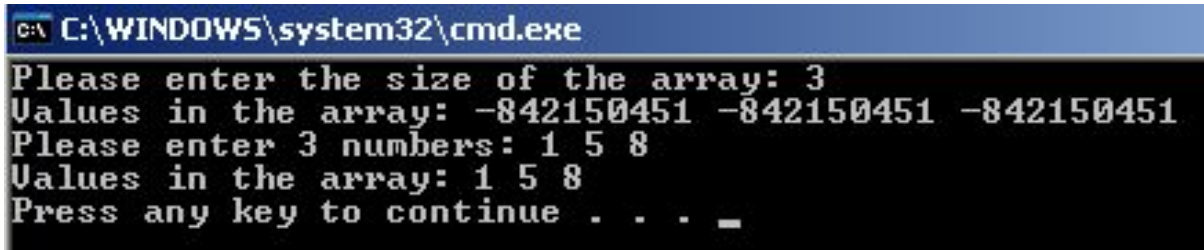
```
#include <stdio.h>
#include <stdlib.h>

void main()
{
    int size, *arr, i;

    printf("Please enter the size of the array: ");
    scanf("%d", &size);
    arr = (int*)malloc(size*sizeof(int));

    if (!arr) // (arr == NULL) --> allocaton didn't succeed
    {
      printf("ERROR! Out of memory!\n");
      return;
    }

    printf("Values in the array: ");
    for (i=0 ; i < size ; i++)
      printf("%d ", *(arr+i)); //*(arr+i) == arr[i]

    printf("\nPlease enter %d numbers: ", size);
    for (i=0 ; i < size ; i++)
      scanf("%d", &arr[i]);

    printf("Values in the array: ");
    for (i=0 ; i < size ; i++)
      printf("%d ", arr[i]);
    printf("\n");
    free(arr);
}
```



זיכרון ה- heap

int:size	3	1000
int*: arr	3000	1004
int: i	???	1008

הזיכרון של ה- main



תזכורת: החזרת מערך מפונקציה

- ראינו כי כאשר מעבירים מערך לפונקציה, מעבירים רק את כתובת ההתחלה שלו, ולא עותק של כל המערך
- ובאופן דומה, כאשר מחזירים מערך שהוגדר בפונקציה, חוזרת כתובת ההתחלה שלו, ולא עותק של כל המערך
- <u>הבעייתיות:</u> כאשר יוצאים מהפונקציה שטח הזיכרון שלה משתחרר ויש לנו מצביע לזיכרון שנמחק...
- <u>הפתרון:</u> הקצאה דינאמית

תזכורת: הבעייתיות בהחזרת מערך מפונקציה - דוגמא

```
#define SIZE 3

int* readArray()
{
    int arr[SIZE], i;
    printf("Please enter %d numbers: ", SIZE);
    for (i=0 ; i < SIZE ; i++)
      scanf("%d", &arr[i]);
    return arr;
}

void main()
{
    int* arr, i;
    arr =  readArray();

    printf("The array is: \n");
    for (i=0 ; i < SIZE ; i++)
      printf("%d ", arr[i]);
    printf("\n");
}
```

```
C:\WINDOWS\system32\cmd.exe
Please enter 3 numbers: 3 5 7
The array is:
270655952 -2 1245032
Press any key to continue . . .
```

int[]: arr	3	2000
	5	2004
	6	2008
int: i	3	2012

הזיכרון של readArray

הקומפיילר נותן warning:
returning address of local variable or temporary
שפירושה שאנחנו מחזירים כתובת למשתנה בזיכרון שישתחרר

לעולם לא נחזיר
מפונקציה
כתובת של משתנה
שהוגדר בה מקומית!

int*: arr	2000	1000
int: i	???	1004

הזיכרון של ה- main

הפתרון: הקצאת המערך דינאמית

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 3

int* buildArray()
{
    int i;
    int* arr = (int*)malloc(SIZE*sizeof(int));

    if (!arr)
    {
      printf("ERROR! Not enough memory!\n");
      exit(1); // to exit the program immedietly.
       // use only when memory allocation fails
    }

    for (i=0 ; i < SIZE; i++)
      arr[i] = i+1;

    return arr;
}

void main()
{
    int  *arr, i;

    arr =  buildArray();

    printf("Values in the array: ");
    for (i=0 ; i < SIZE ; i++)
      printf("%d ", arr[i]);
    printf("\n");

    free(arr);
}
```

אם אנחנו יוצאים מפונקציה שהקצתה דינאמית ולא שחררה, חובה להחזיר את כתובת ההתחלה של ההקצאה, כדי שנוכל לשחרר אותה בהמשך!
אחרת כאשר נצא מהפונקציה כבר לא יהיה משתנה שיכיל את מיקום ההקצאה!

int:i	???	2000
int*: arr	3000	2004

הזיכרון של buildArray

int	1	3000
int	2	3004
int	3	3008

זיכרון ה- heap

int*: arr	3000	1000
int: i	???	1004

הזיכרון של ה- main

הקצאה בתוך פונקציה

- אחריות שלנו כמתכנתים לשחרר את כל הזיכרון שהקצינו
- יש לשים לב בייחוד במקרים בהם ההקצאה מתבצעת בפונקציה אחת, והשחרור צריך להיות בפונקציה אחרת!

החזרת מערך מפונקציה by pointer

- למדנו שפונקציה יכולה להחזיר ערכים ע"י קבלת כתובת לעדכון התשובה
- למשל:

```
int getSum(int arr[], int size);
```

- לעומת:

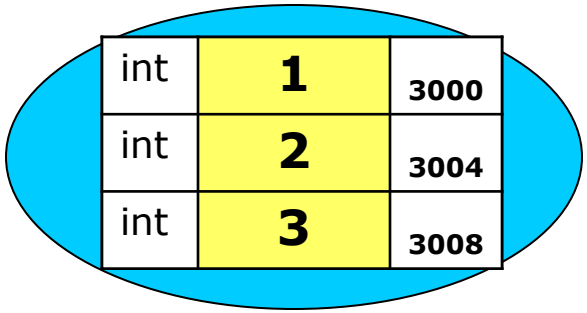
```
void getSum(int arr[], int size, int* sum);
```

- באותו אופן ניתן גם להחזיר מערך שייוצר בתוך הפונקציה
  - לא לשכוח להעביר את המערך כ- **

החזרת מערך מפונקציה דוגמא - by pointer

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

void buildArray(int* arr, int size)
{
      int i;
      arr = (int*)malloc(size*sizeof(int));

      if (!arr)
      }
        printf("ERROR! Not enough memory!\n");
        exit(1); // to exit the program immedietly.
         // use only when memory allocation fails
      {

      for (i=0 ; i < size ; i++)
        arr[i] = i+1;
{

void main()
{
      int size, *arr=NULL, i;

      printf("Please enter the size of the array: ");
      scanf("%d", &size);
      buildArray(arr, size);

      printf("Values in the array: ");
      for (i=0 ; i < size ; i++)
        printf("%d ", arr[i]);
      printf("\n");

      free(arr);
}
```

פה התוכנית תעוף...



זיכרון ה- heap

int:size	3	2000
int: i	???	2004
int*: arr	3000	2008

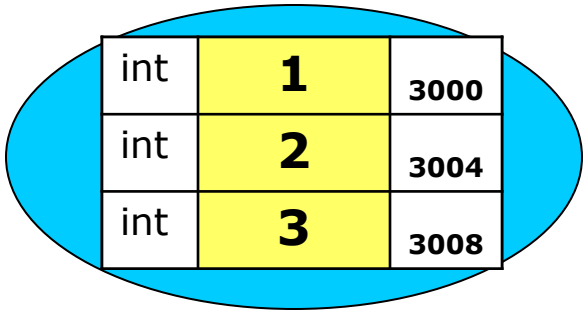
הזיכרון של buildArray

int:size	3	1000
int*: arr	NULL	1004
int: i	???	1008

הזיכרון של ה- main

החזרת מערך מפונקציה התיקון - by pointer

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

void buildArray(int** arr, int size)
}
      int i;
      *arr = (int*)malloc(size*sizeof(int));

      if (!*arr)
      }
        printf("ERROR! Not enough memory!\n");
        exit(1); // to exit the program immedietly.
         // use only when memory allocation fails
      {

      for (i=0 ; i < size ; i++)
        (*arr)[i] = i+1;
{

void main()
{
      int size, *arr=NULL, i;

      printf("Please enter the size of the array: ");
      scanf("%d", &size);
      buildArray(&arr, size);

      printf("Values in the array: ");
      for (i=0 ; i < size ; i++)
        printf("%d ", arr[i]);
      printf("\n");

      free(arr);
}
```



זיכרון ה- heap

int:size	3	2000
int: i	???	2004
int**: arr	1004	2008

הזיכרון של buildArray

int:size	3	1000
int*: arr	3000	1004
int: i	???	1008

הזיכרון של ה- main

הקצאת מערך של מערכים (1)

- כעת אנחנו יכולים לייצר מטריצה שבכל שורה יש מספר שונה של איברים

```
C:\WINDOWS\system32\cmd.exe
Enter number of rows in the matrix: 3
Enter size of row #1: 2
Enter size of row #2: 3
Enter size of row #3: 4
The matrix is:
11 12
21 22 23
31 32 33 34
Press any key to continue . . .
```

הקצאת מערך של מערכים (2)

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

void main()
{
    int rows, **matrix, i, j, *sizes;
    printf("Enter number of rows in the matrix: ");
    scanf("%d", &rows);
    matrix = (int**)malloc(rows*sizeof(int*));
    sizes = (int*)malloc(rows*sizeof(int));

    for ( i=0 ; i < rows ; i++ )
    {
      printf("Enter size of row #%d: ", i+1);
      scanf("%d", &sizes[i]);
      matrix[i] = (int*)calloc(sizes[i], sizeof(int));
      for (j=0 ; j < sizes[i] ; j++)
          matrix[i][j] = (i+1)*10+j+1;
    }

    printf("The matrix is:\n");
    for (i=0 ; i < rows ; i++)
    {
      for (j=0 ; j < sizes[i] ; j++)
          printf("%d ", matrix[i][j]);
      printf("\n");
    }
    free(sizes);
    for ( i=0 ; i < rows ; i++ )
      free(matrix[i]);
    free(matrix);
}
```

int:rows	3	1000
int**: matrix	3000	1004
int: i	3	1008
int: j	???	1012
int*: sizes	4000	1016

הזיכרון של ה- main

int*	4050	3000
int*	5000	3004
int*	2200	3008

int	11	4050
int	12	4054

int	21	5000
int	22	5004
int	23	5008

int	31	2200
int	32	2224
int	33	2228
int	34	2232

int	2	4000
int	3	4004
int	4	4008

זיכרון ה- heap

הגדלת מערך

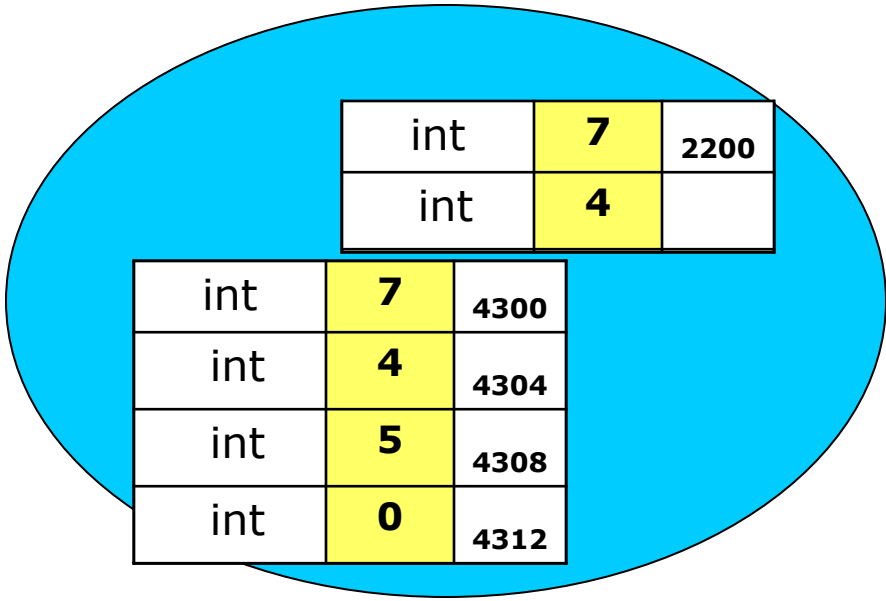
- בדוגמא הבאה אנו קולטים מהמשתמש מספרים לתוך מערך, לא ידוע כמה איברים המשתמש יכניס
- כל פעם כאשר כבר אין מקום במערך צריך להגדיל אותו פי 2
- האלגוריתם:
  - קרא את האיבר החדש, אם 1- צא
  - אם אין מקום במערך, הקצה מערך גדול פי 2
    - העתק למערך החדש את האיברים מהמערך הישן
    - שחרר את המערך המקורי
    - שנה את מצביע המערך להצביע למערך החדש

הגדלת מערך - הפלט

```
C:\WINDOWS\system32\cmd.exe
Please enter numbers, -1 to stop:
7 4 5 2 4 9 7 -1
Read number is 7
Read number is 4
Doubled the array size to 4
Read number is 5
Read number is 2
Doubled the array size to 8
Read number is 4
Read number is 9
Read number is 7
The array has 7 elements (physSize=8):
7 4 5 2 4 9 7
Press any key to continue . . .
```

הגדלת מערך – הקוד

```
void main()
{
    int num, i;
    int physSize = 2, logicSize = 0;
    int* arr = (int*)calloc(physSize, sizeof(int));
    int* tmp;
    printf("Please enter numbers, -1 to stop:\n");
    while (1)
    {
        scanf("%d", &num);
        if (num == -1)
      break;
        if (physSize == logicSize)
        {
      physSize *= 2;
      tmp = (int*)calloc(physSize, sizeof(int));
      for (i=0 ; i < logicSize ; i++)
          tmp[i] = arr[i];
      free(arr);
      arr = tmp;
      printf("Doubled the array size to %d\n", physSize);
        }
        printf("Read number is %d\n", num);
        arr[logicSize] = num;
        logicSize++;
    }
    printf("The array has %d elements (physSize=%d):\n",
                logicSize, physSize);
    for (i=0 ; i < logicSize ; i++)
      printf("%d ", arr[i]);
    printf("\n");
    free(arr);
}
```



זיכרון ה- heap

Int: num	1-	1000
Int: I	???	1004
Int: physSize	4	1008
Int: logicSize	3	1012
Int*: arr	4300	1016
Int*: tmp	4300	1020

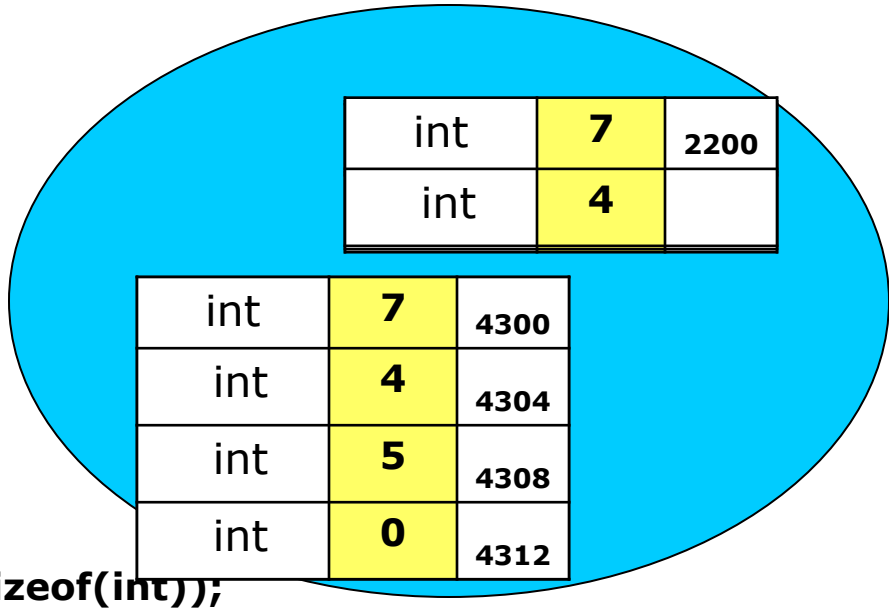
הזיכרון של ה- main

הפונקציה realloc

- שפת C נותנת לנו פונקציה שיודעת להגדיל מערך:
  - הפונקציה יודעת להקצות מקום חדש בגודל המבוקש ולהעתיק אליו את האיברים מהמערך הישן, ובסוף גם לשחרר את הזיכרון של המערך המקורי
- void* realloc(void* ptr, size_t size; (
  - הפונקציה מקבלת כפרמטר את כתובת ההתחלה של המערך שאותו ברצונה להגדיל וכן את גודל שטח הזיכרון החדש שרוצים
  - הפונקציה מחזירה את כתובת ההתחלה של השטח החדש שהוקצה
    - יתכן וזו תהיה הכתובת המקורית, במידה והיה מספיק מקום להגדלה במקום המקורי בו המערך הוקצה

הגדלת מערך – הקוד בשימוש realloc

```
void main()
{
    int num, i;
    int physSize = 2, logicSize = 0;
    int* arr = (int*)calloc(physSize, sizeof(int));
    printf("Please enter numbers, -1 to stop:\n");
    while (1)
    {
        scanf("%d", &num);
        if (num == -1)
            break;
        if (physSize == logicSize)
        {
            physSize *= 2;
            arr = (int*)realloc(arr, physSize*sizeof(int));
            printf("Doubled the array size to %d\n", physSize);
        }
        printf("Read number is %d\n", num);
        arr[logicSize] = num;
        logicSize++;
    }
    printf("The array has %d elements (physSize=%d):\n",
                    logicSize, physSize);
    for (i=0 ; i < logicSize ; i++)
        printf("%d ", arr[i]);
    printf("\n");
    free(arr);
}
```



זיכרון ה- heap

int: num	-1	1000
int: i	???	1004
int: physSize	4	1008
int: logicSize	3	1012
int*: arr	4300	1016

הזיכרון של ה- main

מה יהיה פלט התוכנית הבאה?

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

void main()
{
    char* c = (char*)malloc (sizeof(char)*5);
    char** s = (char**)malloc (sizeof(char*)*3);
    int i;

    for( i=0  ;  i < 3  ; i ++  )
    {
     printf("enter word : ");
     gets(c);
     s[i] = c;
    {

    for(i=0; i<3; i++)
     printf("strings : %s \n",s[i]);

    free(c);
    free(s);
}
```

```
enter word : hi
enter word : bye
enter word : good
strings : good
strings : good
strings : good
```

char	'g'	4300
char	'o'	4301
char	'o'	4302
char	'd'	4303
char	0	4304

char*	4300	5300
char*	4300	5304
char*	4300	5308

char*: c	4300	1000
char**: s	5300	1004
int: i	3	1008

הזיכרון של ה- main

התיקון לתוכנית הקודמת

```
void main()
{
    char* c;
    char** s = (char**)malloc (sizeof(char*)*2);
    int i;

    for( i=0  ;  i < 2  ; i ++  )
    {
     c = (char*)malloc (sizeof(char)*5);
     printf("enter word : ");
     gets(c);
     s[i] = c;
    {

    for(i=0; i<2; i++)
     printf("strings : %s \n",s[i]);

    for (i=0 ; i < 2 ; i++)
        free(s[i]);
    free(s);
}
```

char	'b'	6300
char	'y'	6301
char	'e'	6302
char	0	6303
char	???	6304

char	'h'	4300
char	'i'	4301
char	0	4302
char	???	4303
char	???	4304

char*	4300	5300
char*	6300	5304

זיכרון ה- heap

char*: c	6300	1000
char**: s	5300	1004
int: i	2	1008

הזיכרון של ה- main

```
enter word : hi
enter word : bye
strings : hi
strings : bye
```

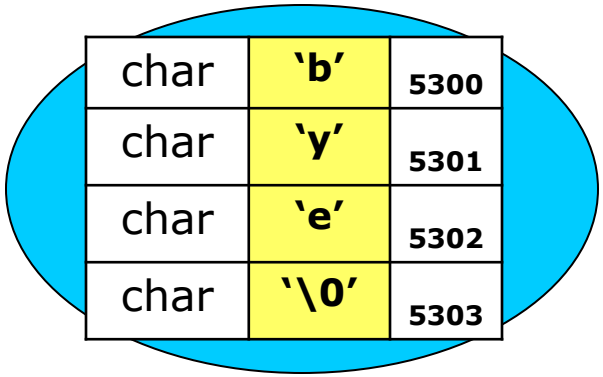
הפונקציה strdup

```
char* strdup(const char *str)
```

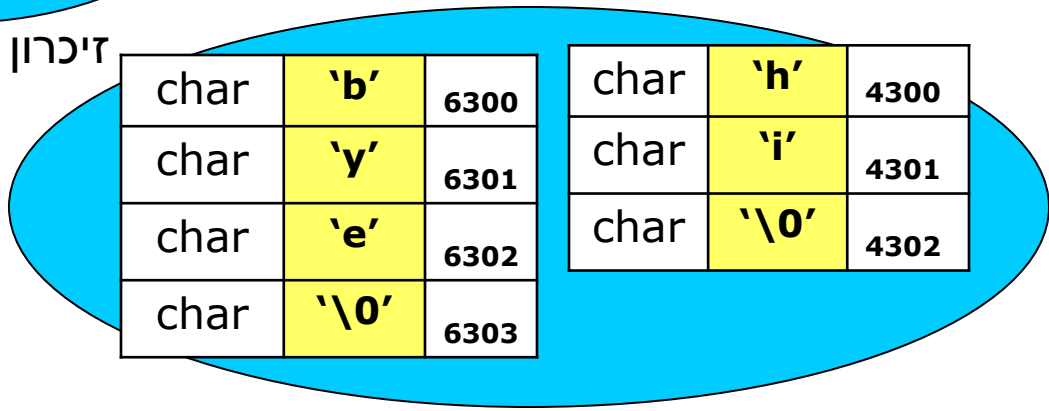
- מקבלת מחרוזת ומחזירה העתק שלה:
  - מקצה דינאמית על ה- heap מערך של תווים בגודל המחרוזת המקורית, מעתיקה אליו את התוכן ומחזירה את כתובת ההתחלה שלו
  - תחזיר NULL במידה וההקצאה נכשלה
- אחריות המתכנת לשחרר את המחרוזת שחזרה!!

strdup – דוגמא

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

void main()
}
    char str1[] = "hi";
    char* str2 = "bye";
    char* newStr1 = strdup(str1);
    char* newStr2 = strdup(str2);

    printf("The first duplicated string: |%s|\n", newStr1);
    printf("The second duplicated string: |%s|\n", newStr2);

    free(newStr1);
    free(newStr2);
{
```



static storage -זיכרון ה



heap -זיכרון ה

char: str1[]	'h'	1000
	'i'	1001
	'\0'	1002
char*: str2	5300	1003
char*: newStr1	4300	1007
char*: newStr2	6300	1011

הזיכרון של ה- main

```
The first duplicated string: |hi|
The second duplicated string: |bye|
```

ביחידה זו למדנו:

- מוטיבציה להקצאות דינאמיות
- מהי הקצאה דינאמית
- יצירת מערך בגודל שאינו ידוע מראש
- החזרת מערך מפונקציה
- הקצאת מערך של מערכים
- הגדלת מערך
- הפונקציה strdup

תרגיל 1

- כתוב תוכנית המבקשת מהמשתמש להכניס את כמות המספרים שהוא רוצה שיהיו במערך, והקצה מערך בהתאם
- הגרל ערכים למערך והגרל מספר נוסף (בטווח 0-9)
- ייצר מערך חדש המכיל את האינדקסים במערך המקורי שערך האיבר שבתוכם שווה למספר הנוסף שהתקבל
- הצג את המערך שחדש שייצרת

- דוגמא:

עבור המערך 1,2,5,2,2,9 גודלו 6 והמספר 2, יוחזר מערך בגודל 3 שערכיו 1,3,4

תרגיל 2

- כתוב פונקציה המקבלת מחרוזת ותו. הפונקציה תייצר ותחזיר מחרוזת חדשה שאורכה ככמות הפעמים שהתו מופיעה במחרוזת המקורית והיא תכיל תו זה כמספר פעמים זה

- דוגמאות:
  - עבור המחרוזת "hello" והתו 'l' תוחזר המחרוזת "ll"
  - עבור המחרוזת "keren" והתו 'e' תוחזר המחרוזת "ee"

תרגיל 3

- כתוב פונקציה המקבלת 2 מערכים וגודלם. הפונקציה תחזיר מערך חדש המכיל את הערכים שבשני המערכים
- הפוקנציה גם תחזיר את גודל המערך המוחזר

- דוגמא:

עבור המערך 1,8,2 וגודלו 3 והמערך 9,2,6,7 וגודלו 4 יוחזר המערך החדש 1,8,2,9,2,6,7 וגודלו 7