**תרגילים ופתרונות בשפת C – הקצאה דינאמית**

כתב וערך: שייקה בילו

**זיכרון דינמי**

ישנן שתי שיטות לביצוע הקצאת זיכרון: הקצאת זיכרון סטטית והקצאת זיכרון דינאמית.

**הקצאת זיכרון סטטית** - המהדר קובע את דרישות האחסון על פי הצהרת המשתנים, בזמן הקומפילציה (כך הקצאנו זיכרון עד כה!). בעיה שלעיטים צצה היא שאין אנו יכולים לנחש מראש את כמות הזיכרון שהתוכנית שלנו עלולה לצרוך. למשל, אם נרצה לכתוב תוכנית שיכולה לקלוט ולשמור כמות לא ידועה של מס' טלפון ע"מ לנהל ספר טלפונים, לא נוכל להגדיר מראש את כל המשתנים שיכילו את המספרים הללו! לכן יש צורך בשיטת ניהול זיכרון שהיא דינאמית (כלומר שיטה שנותנת מענה לצורכי הזיכרון בזמן ריצה).

**הקצאת זיכרון דינאמית** - הקצאת מקום נעשה בזמן בריצה על יד קריאה לפונקציה malloc() (קיצור ל-memory allocation). פונקציה זו מקבלת כפרמטר את מס' הבתים שברצוננו להקצות ומחזירה את הכתובת של הבית הראשון ברצף הבתים שהקצאתה. אם הפונקציה נכשלת מוחזר הערך NULL.

**שחרור זיכרון דינאמית** - מכיוון שהקצאת הזיכרון נעשתה בזמן ביצוע התוכנית, יש לדאוג לשחרר את הזיכרון לאחר שנסיים להשתמש בו. שחרור של זיכרון דינאמי נעשה ע"י קריאה לפונקציה free(). הפונקציה מקבלת מצביע לכתובת תחילת קטע הזיכרון שרוצים לשחרר!

הערות:

(1)הפונקציות free() ו-malloc() נמצאות בספרייה stdlib.h ויש להכליל אותה בתוכנית ע"מ שנוכל להשתמש

בהן.

(2)משתנה שבעבורו הייתה הקצאת זיכרון סטטית, המהדר דואג לשחרור הזיכרון .

**האופרטור sizeof**

האופרטור sizeof מחזיר את גודלו של הטיפוס בבתים.

תבנית:

sizeof (name\_of\_charecter) לדוגמא:sizeof (float) מחזיר 4. אילו הינו כותבים:

float f;

sizeof(f) היה מחזיר את אותו דבר (4).

**הקצאה דינאמית – malloc**

כעת נזכר בפונקציה להקצאת זיכרון בזמן ריצה malloc - לשם כך יש צורך להגדיר מצביע לרצף הבתים שאנו רוצים להקצות בזיכרון ולציין את גודל הרצף בבתים. את הערך שמחזירה malloc() צריך לשמור בתוך מצביע. המצביע לרצף יכיל את הכתובת של הבית הראשון ברצף. גודלו של הרצף הוא: (גודל הטיפוס בבתים)\*(כמות המשתנים מאותו טיפוס שבשביל לשמור אותם אנו מקצים את הזיכרון). למשל רצף של 5 int-ים ישמר בתוך 10 בתים (בהנחה שכל int תופס 2 בתים).

התבנית לשימוש בפונקציה malloc() היא:

variable\_pointer = (pointer\_type) malloc (size\_of\_memory);

דוגמא:

int size, \*p\_list;

printf("Enter the number of elements:");

scanf("%d", &size);

p\_list = (int\*)malloc (size \* sizeof(int));

כאשר מקצים זיכרון הערך המוחזר על ידי פונקצית הקצאת זיכרון היא כתובת. יש מקרים שבעבורם הקצאת הזיכרון נכשלת והערך המוחזר על ידי הפונקציה הוא null. מכיוון שלא קיבלנו כתובת לא ניתן להשתמש במצביע כמצביע "חוקי" ונרצה לסיים את התוכנית. לכן לאחר הקצאת זיכרון תמיד צריך לבדוק אם ההקצאה הצליחה.

void main()

{

long \*l\_list;

l\_list =(long\*) malloc (5\*sizeof(long));

if (l\_list == NULL)

{

printf ("Failed to allocate memory");

return;

}

free(l\_list);

}

תרגיל 1 - כתוב תוכנית שקולטת מהמשתמש את אורך הסדרה שהוא רוצה להכניס, קוראת את הסדרה ומדפיסה אותה. הרצת התוכנית תראה כך:

הרצה :

Enter the series length: 6

Now, enter the series

1 2 3 4 5 6

The series is:

1 2 3 4 5 6

}

תרגיל 2 - לפניך תוכנית בה מוצג מימוש הקצאת זיכרון דינאמי, בתוכנית נמחקו 10 פקודות ותווים, עליך לתקן את התוכנית ולבדוק הרצה תקינה.

#include <stdio.h>

#include <\_\_\_\_\_\_>

void\* my\_alloc(int size\_bits)

{

void\* tmp;

tmp=malloc(size\_bits) ;

if(tmp==\_\_\_\_\_\_\_)

{

printf("Memory allocation failed\n");

exit(\_);

}

return \_\_\_\_ ;

}

int main()

{

int size ,i;

int \*ptr2memory ;

printf("Enter the series length :");

scanf("%d",&\_\_\_\_);

ptr2memory=(int\*)my\_alloc(size\_sizeof(int));

printf("Now, Enter the series\n");

for(i=0;i<size;++\_)

scanf("%\_",&ptr2memory[i]);

printf("The series is : \n");

for(i=0;i<\_\_\_;++i)

printf("%\_ ",ptr2memory[i]);

free(\_\_\_\_\_\_\_\_\_\_\_\_;

return 0;

}

תרגיל 3 – כתוב תוכנית המדגימה מערך דו-ממדי דינאמי.

הסבר: כשרוצים להגדיר מערך דו ממדי בעל גודל משתנה, עלינו ליצור מערך של מצביעים למערכים. גודל המערך לא ידוע בתחילת התכנית ולכן נגדיר את המערך באמצעות פונקצית: ( ) malloc

בתחילה נאתחל מערך של מצביעים, ולאחר מכן נאתחל כל מצביע להיות מערך של ה- type הרצוי. למשל, יצירת מערך של int-ים בגודל המוגדר ע"י המשתמש שבכל איבר בו יש את הערך של מכפלת האינדקסים שלו.

תרגיל 4 – כתוב תוכנית שתמיין סדרת מספרים שלמים הנתונה כקלט לתוכנית.

**דוגמה להרצה :**

Enter the series size followed by the elements:

5 5 4 3 2 1

the sorted series is:1 2 3 4 5

תרגיל 5 – כתוב תוכנית הקוראת סדרת שלמים באורך כלשהו. ברגע שהזיכרון נגמר הפונקציה מכפילה את גודל הזיכרון המוקצה . הפונקציה מחזירה מצביע לתחילת הזיכרון שמכיל את הנתונים שנקראו. מספר הנתונים שנקראו מעודכן ע"י המצביע pn. ניתן ואף מומלץ להשתמש בפונקציית realloc.

תרגיל 6 – לפניך תוכנית, הסבר במספר מילים מה עושה התוכנית, מהו הקלט שלה ומהו הפלט.

#include <stdio.h>

#include "stdlib.h"

#define FORMAT\_STRING "%s%s%s%s"

#define SCANF\_ARGS , tocken\_array[0], tocken\_array[1], tocken\_array[2] ,tocken\_array[3] ,tocken\_array[4]

#define NUM\_OF\_TOCKENS 5

void main()

{

int i, str\_size, num\_of\_words, max\_tocken\_size ;

char \* str, \* tocken\_array[NUM\_OF\_TOCKENS];

printf("please enter size of input string and max size of each

word:");

scanf ("%d%d",&str\_size,&max\_tocken\_size);

getchar();

printf("please enter your string:\n");

str = (char\*) malloc (str\_size+1);

for (i=0; i<NUM\_OF\_TOCKENS;i++)

tocken\_array[i] = (char\*) malloc (max\_tocken\_size+1);

gets(str);

num\_of\_words = sscanf(str, FORMAT\_STRING SCANF\_ARGS );

printf("str = %s\n",str);

for (i=0; i<num\_of\_words;i++)

printf("tocken #%d = %s\n",i,tocken\_array[i]);

for (i=0; i<NUM\_OF\_TOCKENS;i++)

free(tocken\_array[i]);

}

input/Output…

please enter size of input string and max size of each word:10 3

please enter your string:

abc d ef ghi

str = abc d ef ghi

tocken #0 = abc

tocken #1 = d

tocken #2 = ef

tocken #3 = ghi

תרגיל 7 - (I) עיין בקטע הבא וסמן את כל התשובות הנכונות:

0 1 2 3 4 5 6 7 8 9

#include <stdio.h>

#include <stdlib.h>

#define MAX 10

void main()

{

int \*ptr, \*arr[MAX];

int i, j;

for (i=MAX-1 ; i>=0; i--)

if (arr[i] = (int \*) malloc(i \* sizeof(int)))

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

\*(\*(arr+i)+j) = j\*i;

ptr = \*(arr+MAX-1);

while (\*ptr)

printf ("%d ", \*ptr--);

}

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 72 | 63 | 54 | 45 | 36 | 27 | 18 | 9 | 0 |

0 1 2 3 4 5 6 7 8

1. התכנית לא מדפיסה כלום. – הסיבה היא ש- \*ptr משתערך לאפס (ראה תרשים) וגוף לולאת ה-while לא מבוצע אף לא פעם אחת.
2. יש שגיאה בזמן ריצה .(run time error)
3. התכנית תדפיס: 9 18 27 36 45 54 63 72.
4. התכנית תדפיס אינסוף אפסים.
5. התכנית תדפיס 0.
6. התכנית תדפיס ערכים לא ידועים.
7. אף לא אחת מהתשובות לעיל.

(II) איך ניתן לשנות את השורה: ptr = \*(arr+MAX-1); ע"מ שתשובה ג' תיהיה נכונה?

תרגיל 8 - כתוב פונקציה char \* strstr(char \*st1, char \*st2) שמקבלת כארגומנט שתי מחרוזות. במידה שהמחרוזת st2 מוכללת במחרוזתst1 , הפונקציה מחזירה מצביע לתו של st1 שממנו מתחילה המחרוזת הזאת, אחרת הפוקציה מחזירה NULL.

1. כתוב את הפונקציה בצורה לא רקורסיבית
2. כתוב אותה בצורה רקורסיבית.

**תיאור הזיכרון של המחשב:**

ניתן לחשוב על זיכרון המחשב כעל רצף של תאים, כל אחד בגודל בית (byte) כאשר כל בית בגודל של 8 סיביות (bits). כל תא בזיכרון מזוהה ע"י ערך מספרי המתאר את מיקומו המדויק. ערך זה הינו הכתובת של התא בזיכרון (memory address). למשל, אם נגדיר את המשתנים הבאים:

char var1; /\*Assume that on our system, required amount of storage for type char is 1 byte. /\*

double var2; /\*Assume that on our system, required amount of storage for type double is 8 bytes. /\*

int var3; /\*Assume that on our system, required amount of storage for type int is 4 bytes. /\*

…

תמונת הזיכרון עלולה להיות (אך לא בהכרח):

…300 301 302 303 304 305 306 307 308 309 310 311 312 313 …

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

var1

var2

var3

כתובת של משתנה הינה הכתובת של הבית הראשון ברצף הבתים שמשתנה זה תופס בזיכרון. למשל בדוגמה שלנו, הכתובת של var1 היא 300 , של var2 היא 301 ושל var3 היא 309.

**האופרטור & :**  אם x הוא משתנה אזי &x היא כתובת הזיכרון של x, כלומר האופרטור & מציין "כתובתו של…".

**מצביעים (pointers):** מצביע הינו משתנה שהערך שלו הוא כתובת של משתנה כלשהו. במילים אחרות, מצביע הינו משתנה שמצביע למשתנה אחר. למשתנה שנמצא בכתובת שמכיל המצביע נקרא "המוצבע".

כשנרצה להצהיר על משתנה p כמצביע למשתנה מטיפוס מסוים, נוסיף את התו '\*' מימין לשם הטיפוס שעליו p יצביע ומשמאל ל-p. למשל, אם נרצה להגדיר את p כמצביע למשתנה כלשהו מטיפוסint , שורת ההצהרה תיראה כך: int \*p;

במקרה זה, המצביע הינו משתנה מטיפוס מצביע ל-int (או בקיצור, מטיפוס int \*).

באופן כללי, תבנית הצהרה על מצביעים הינה**: variable-type> \*<variable name>;>**

ברגע שנצהיר על משתנים כמצביעים, נוכל להציב לתוכם כתובות של משתנים.לדוגמא:

int x = 200 ;

int \*p; /\*Declaration of a pointer “p” that is of type “int \*”.\*/

p=&x; /\*Assign the address of “x” to be the value of “p”.\*/

בשלב זה המשתנה p מכיל את הכתובת של המשתנה x.

**האופרטור \***

האופרטור \* הינו אופרטור שפועל על מצביעים. למשל עבור הדוגמה הקודמת \*p שקול ל-x.

ואם נכתוב \*p=500; זה אומר ל-x=500;.

**זהירות בעת שימוש במצביעים:**

כשמצהירים על מצביע p אין הוא מצביע על ערך חוקי (באופן כללי, עד שלא מאתחלים משתנה לאחר שמגדירים אותו, הוא מכיל ערך "זבל"). חובה להציב ב-p ערך של כתובת חוקית (כלומר, שהגישה אליה מותרת) לפני שמשתמשים בערך המוצבע (p\*). לדוגמה, כתיבת:

int \*pi;  
\*pi = 100; /\*wrong!!!!\*/

תייצר טעות שעלולה לגרום לקריסת התוכנית (זהו ניסיון לפנות לכתובת לא חוקית ולהציב בה את הערך 100) בזמן ריצה. טעות זו ידועה בשם segmentation fault. הדרך הנכונה היא:

int \*pi;  
int x;  
pi = &x;  
\*pi = 100;

דוגמה נוספת:

char \*s = "I\_am\_a\_string";

\*s = 'a';

גם כאן תיווצר שגיאת זמן ריצה מסוג segmentation fault. תקבלו הודעת שגיאה מהסוג הבא:

Unhandled exception at 0x004136a8 in my\_program.exe: 0xC0000005: Access violation writing location 0x00415658.

הסבר: s הוגדר כמצביע למחרוזת הקבועה “I\_am\_a\_string”. כאשר תוכנית זו תעבור קומפילציה, המחרוזת “I\_am\_a\_string” תמוקם באיזור בזיכרון שלא ניתן לכתוב אליו אלא רק לקרא ממנו(read-only segment), יחד עם נתונים קבועים אחרים. כאשר נריץ את התוכנית, s יצביע לאיזור בזיכרון בו מוקמה המחרוזת “I\_am\_a\_string” וייעשה ניסיון לכתוב את התו ‘a’, באמצעות המצביע, למקום בזיכרון שאסור לכתוב אליו!

דוגמה נוספת:

#include <stdio.h>

void main()

{

int x , \*px; /\*Define a variable of type int named "x" and a

variable of type pointer to int named "px".\*/

int y, \*py ;

px=&x; /\*Assign the address of "x" to be the value of "px".\*/

py=&y;

scanf("%d%d",px,py);

/\*Read two integers values from the input and assign them to "x" and "y". Make sure that you understand why!!! \*/

printf("x=%d , y=%d\n",\*px,\*py);

/\*Print the values of the x and y which are the variables pointed to by px and py. \*/}

##### הערה: scanf אמורה לקבל כתובות של משתנים שאליהם היא מבצעת השמה ולכן העברנו לה את px ו-py ולא את \*px ו-\*py (בניגוד ל-printf שם העברנו את ערכי המשתנים שאנו רוצים להדפיס למסך).

**העברת מצביעים כפרמטרים לפונקציה**

כאשר אנו מעבירים משתנים כפרמטרים אקטואליים בקריאה לפונקציה, ערכיהם מחושבים ומועתקים לפרמטרים הפורמאליים המתאימים של הפונקציה והקשר ניתק . מכיוון שמצביעים הם משתנים לכל דבר, ניתן להעביר את ערכיהם בתור פרמטרים לפונקציות. מנגנון זה מאפשר לפונקציה נקראת לשנות את ערכיהם של משתנים בסביבה הקוראת. הסיבה לכך היא שלפונקציה שנקראה הועברו כתובות של משתנים בסביבה הקוראת (אלו הערכים שהכילו המצביעים) ולכן יש לה גישה למשתנים אלו ולערכים שהם מכילים.

לדוגמא ,נכתוב פונקציה אשר מחליפה בין ערכיהם של זוג משתנים ( מטיפוס int ) בפונקציה הקוראת :

void swap(int a, int b)

{

int tmp = \*a;

\*a = \*b;

\*b = tmp;

}

לשימוש בפונקציה swap() ראה תוכנית מס' 2 בתוכניות לדוגמה (בהמשך).

**כתובת של מערך**

שמו של מערך דומה למצביע המייצג את הכתובת של האיבר הראשון בתוך המערך. על אף הדימיון, חשוב לזכור ששם של מערך אינו מצביע שכן הוא מייצג כתובת קבועה ולא ניתן לבצע לתוכו השמה שתשנה אותה. אל כל איברי המערך אנו נגישים לפי ההיסט (offset) מהכתובת של האיבר הראשון.

לדוגמא:

int arr\_i [5];

כל התאים הם מסוג שלם , שם המערך arr\_i יכיל כתובת של ה-int (בדומה למצביע מטיפוס int\*) הראשון במערך. לפיכך, \*(arr\_i) שקול ל- arr\_i[0] שזה האיבר הראשון במערך.כמו כן, \*(arr\_i+1)שקול לאיבר השני ו- \*(arr\_i+4)שקול לאיבר האחרון . נכליל זאת ,על מנת לגשת לאיבר a[i]נכתוב \*(a+i). כמו כן, (a+i)≡&a[i].

דוגמה להדפסת מערך:

void main()

{

int i;

float arr\_f[];

for (i=0 ; i < 5; i++)

scanf(“%f”,arr\_f+i);

for (i=0 ; i < 5; i++)

printf("array[%d] = %f ",i , \*(arr\_f+i));

}

**בהצלחה!!!**