בס"ד

תרגיל בית א – מערכות הפעלה, תשפ"ג

בתרגיל זה עליכם ללמוד בכוחות עצמכם כיצד לתכנת בשפת C.

שפה זו דומה מאד לשפות שאתם כבר מכירים, ולכן הלימוד אמור להיות מהיר ופשוט.

הוספנו כאן מעט הסברים לפקודות עיקריות. ניתן למצוא שפע של מידע נוסף ברחבי הרשת.

תכנית ראשית - main

הכותרת של התכנית הראשית התיקנית היא int main(int argc, char* argv[]).
משמעות הדבר הוא שהמתודה הראשית יכולה לקבל פרמטרים בזמן ההפעלה.

הפרמטרים הם:

1. argv – מערך של מחרוזות המכיל את כל הפרמטרים שהתוכנית הראשית מקבלת. כמובן – יש צורך בהמרה בתוך הקוד במידה ונשלחו פרמטרים מטיפוס אחר. מערכת ההפעלה אחראית לשחרר את הזיכרון ובו המחרוזת שבפרמטר זה.
2. argc – מספר הפרמטרים שהתוכנית קיבלה. פרמטר זה הינו 1 לפחות, כי שם התוכנית נחשב גם הוא כפרמטר. מערכת ההפעלה היא זו ששמה את הערך בפרמטר זה.

דוגמת קוד לשימוש בmain עם פרמטרים:
https://aaronbloomfield.github.io/pdr/readme.html

```
// hello world
#include <stdio.h>
int main(int argc, char **argv) {
    for (int i = 0; i < 3; i++) {
        printf("hello world%s!\n", argv[1]);
    }
    return 0;
}
```

פקודת הקלט scanf

פקודת הקלט התיקנית היא אמנם scanf, אך כדאי להכיר גם את הפעולה fgets שיכולה לעזור למנוע גלישה במידה והקלט ארוך מדי.

דוגמת קטע קוד להדגמה של הפעולה:
https://aaronbloomfield.github.io/pdr/tutorials/09-c/index.html

```
int age;
char grade;
char school[3];
printf("AGE: ");
scanf("%d", &age);     /* Converts input to an integer and stores it in age */
printf("GRADE: ");
scanf("%c", &grade);   /* Converts input to a letter grade (probably 'A') and
                          stores it in grade */
printf("SCHOOL: ");
scanf("%s", school);   /* Converts input to a string and stores it in school */
```

הפרמטר ... וספריית stdarg

השפה מאפשרת ליצור מתודה עם מספר פרמטרים (ארגומנטים) לא ידוע מראש.

בכותרת הפונקציה משתמשים במשתנה ... (שלוש נקודות. נקרא גם ellipsis) כדי לסמן שישנו מספר בלתי מוגבל של פרמטרים. כלומר המתודה יכולה לקבל בכל פעם מספר שונה של פרמטרים.

הספריה stdrarg מגדירה סוג משתנה va_list ושלוש פקודות מאקרו בהן ניתן להשתמש כדי לטפל בארגומנטים בפונקציה כאשר מספר הארגומנטים אינו ידוע.

- void va_start(va_list ap, last_arg) – מאקרו זה מאתחל את המשתנה ap בערך הראשון שלפני ה – ellipsis. דרך משתנה זה נוכל לגשת כעת לארגומנטים.
- type va_arg(va_list ap, type) – מאקרו זה מחזיר את הארגומנט הבא ברשימה, לפי הסוג type.
- void va_end(va_list ap) – יש לזמן מאקרו זה עם סיום המתודה שהשתמשה ב va_start. ללא ביצוע מאקרו זה – הערך החוזר מהפונקציה לא יוגדר.

דוגמת קוד להדגמה של הפעולות :
https://www.tutorialspoint.com/cprogramming/c_variable_arguments.htm

```
#include <stdio.h>
#include <stdarg.h>
double average(int num,...) {
   va_list valist;
   double sum = 0.0;
   int i;
   /* initialize valist for num number of arguments */
   va_start(valist, num);
   /* access all the arguments assigned to valist */
   for (i = 0; i < num; i++) {
      sum += va_arg(valist, int);
   }
   /* clean memory reserved for valist */
   va_end(valist);
   return sum/num;
}
int main() {
   printf("Average of 2, 3, 4, 5 = %f\n", average(4, 2,3,4,5));
   printf("Average of 5, 10, 15 = %f\n", average(3, 5,10,15));
}
```

- Include stdarg.h
- Define a function with its last parameter as ellipses and the one just before the ellipses is always an **int** which will represent the number of arguments.
- Create a **va_list** type variable in the function definition. This type is defined in stdarg.h header file.
- Use **int** parameter and **va_start** macro to initialize the **va_list** variable to an argument list. The macro va_start is defined in stdarg.h header file.
- Use **va_arg** macro and **va_list** variable to access each item in argument list.
- Use a macro **va_end** to clean up the memory assigned to **va_list** variable.

הקצאת זיכרון דינמי

הקצאת זיכרון דינמי מתבצעת באמצעות הפקודה malloc(). הפקודה מחזירה מצביע לזיכרון שהוקצה בערימה, או NULL במקרה שההקצאה לא הצליחה. (חשוב לבדוק את הערך שהוחזר לפני הגישה לזיכרון).

חשוב! אזור הזיכרון שמוקצה אינו מאותחל בשום צורה! יש צורך לאתחל אותו בפירוש באמצעות מתודות מתאימות, לפי טיפוס הנתונים שאתה רוצה להקצות. (ניתן גם להשתמש בפקודה calloc [בפורמט אחר], שמאתחלת ל-0.)

הצורה הבסיסית של הפקודה היא: void* malloc(size_t size) , כאשר יש לשלוח כפרמטר מפורש את הגודל של הזיכרון שרוצים להקצות. לצורך כך – האופרטור sizeof שימושימאד.

דוגמת קוד להדגמה של הפעולה:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    // dynamically allocate an array of ints
    int* p = (int*) malloc(sizeof(int) * 5);
    if (p == NULL) {
        // memory allocation failed; handle the error somehow
        return 1;
    }
    // Initialize p[1] to 10. Everything else is still uninitialized.
    // Trying to access any other index without first initializing it is
                                                        undefined behavior!
    p[1] = 10;
    printf("%d\n", p[1]);
    // free up that array
    free(p);
    return 0;
}
```

שחרור הזיכרון הדינמי נעשה ע"י הפעולה free(). (פעולה זו מקבילה לפעולה delete שאתם מכירים מ ++C). הצורה הבסיסית של הפקודה הוא: void free(void* ptr)

שגיאות נפוצות בשחרור זיכרון:

1. שחרור זיכרון שלא הוקצה ע"י הפעולה malloc()
2. הפעלה של הפעולה free() על מצביע לNULL

3. שחרור פעמיים של אותו זיכרון (ע"י שני מצביעים לאותה כתובת)

כל אחת מהשגיאות הללו תגרום נזק לערמה (heap) בצורה שתביא לתוצאות מוזרות שיהיה קשה מאד לדבג .לכן – הדרך הטובה ביותר לדבג שגיאות כאלו הוא פשוט לא לעשות אותן מלכתחילה. בשונה משפת ++ C, שפת C מאפשרת הרבה יותר הזדמנויות לעשות באגים.

<u>מבני נתונים שנוצרים ע"י המשתמש</u>

בשפת C, לא ניתן ליצור מחלקות – אלא מבנים (struct) בלבד.

סוג נוסף של מבנה הוא union (איגוד) – שמאפשר לגשת בזמן ריצה רק לאחד מהשדות שלו. (כלומר – בפועל, למרות שהגדרנו מספר שדות, רק אחד מהם בא לידי שימוש)

דוגמת קוד להדגמה של שימוש במבנה:

```
struct example_struct {
    int type;
    union {
        int i;
        float f;
        double d;
    };
};
struct example_struct s;
s.type = 1;
switch (s.type) {
case 0:
    printf("%d\n", s.i);
    break;
case 1:
    printf("%+2.3f\n", s.f);
    break;
case 2:
    scanf("%lf", &s.d);
    break;
}
```

<u>ועכשיו לעבודה...</u>

את התרגיל הבא עלייך לכתוב בשפת C ולקמפל באמצעות הפקודה cl, בCMD.

כתוב תוכנית אשר תטפל ברשימה מקושרת (linked list) פשוטה.

התוכנית תקבל מהמשתמש את גודל הרשימה כפרמטר של התוכנית הראשית.

התוכנית תקלוט נתונים לפי המספר שנכנס כפרמטר, ותכניס אותם לרשימה (הסדר אינו משנה). חוליות הרשימה יוקצו באופן דינמי.

לאחר מכן התוכנית תדפיס את הרשימה המקושרת למסך וגם לתוך קובץ.

דוגמת הרצה (דוגמת מסך. במקביל, ישנה הדפסה לקובץ):

```
>Project1.exe
Please run with parameter of input size, i.e. project.exe 4

>Project1.exe 0
Please run with parameter of input size greater than 0, i.e.
project.exe 4

>Project1.exe 4
Enter number 1
12
Enter number 2
15
Enter number 3
17
Enter number 4
14
List of numbers
14 17 15 12
```

הערות:

יש להשתמש בכל הפקודות והפעולות שתוארו בקובץ זה.

אין צורך בכתיבת תוכנית מסובכת. ניתן לכתוב תוכנית ראשית אחת מסודרת, עם הערות ברורות.

הגישו את הקבצים הבאים:

1. קובץ הקוד של התוכנית
2. קובץ הפלט של ההרצה

צרפו בקובץ נוסף צילומי מסך:

3. של ההרצה ב CMD

להגשה

בהצלחה רבה!