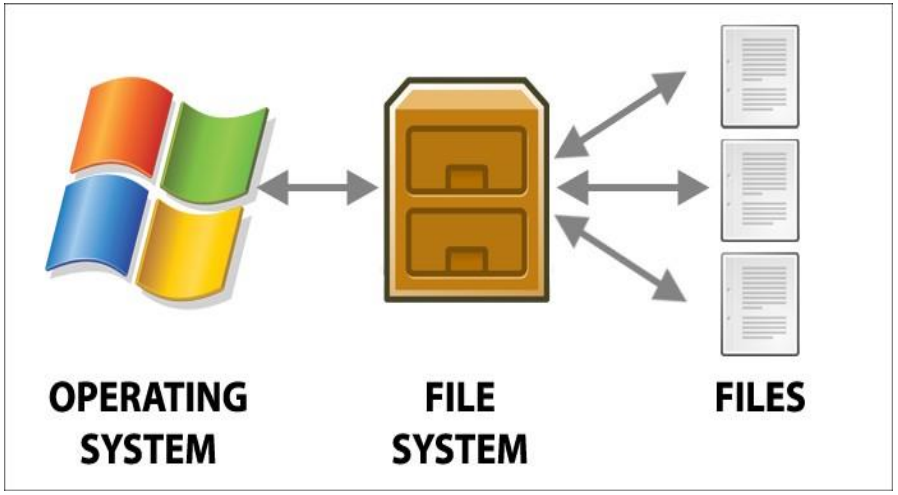
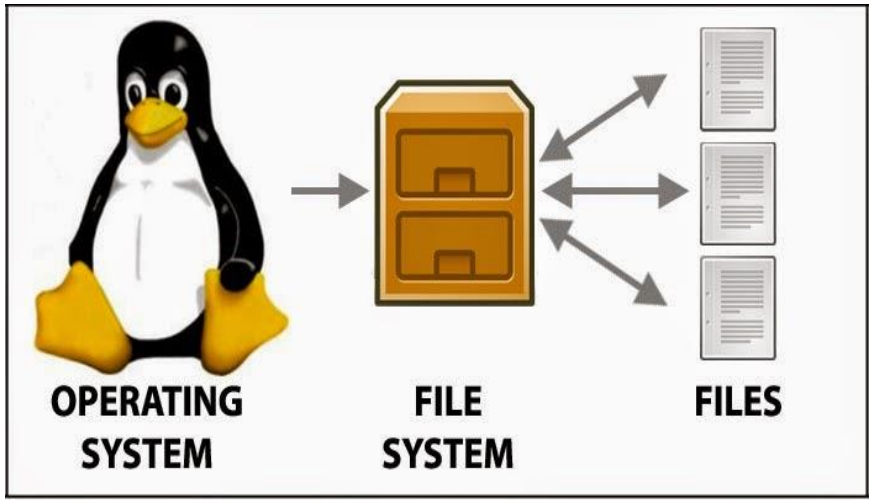
קורס תכנות בשפת C

פרק 17

קבצים - Files



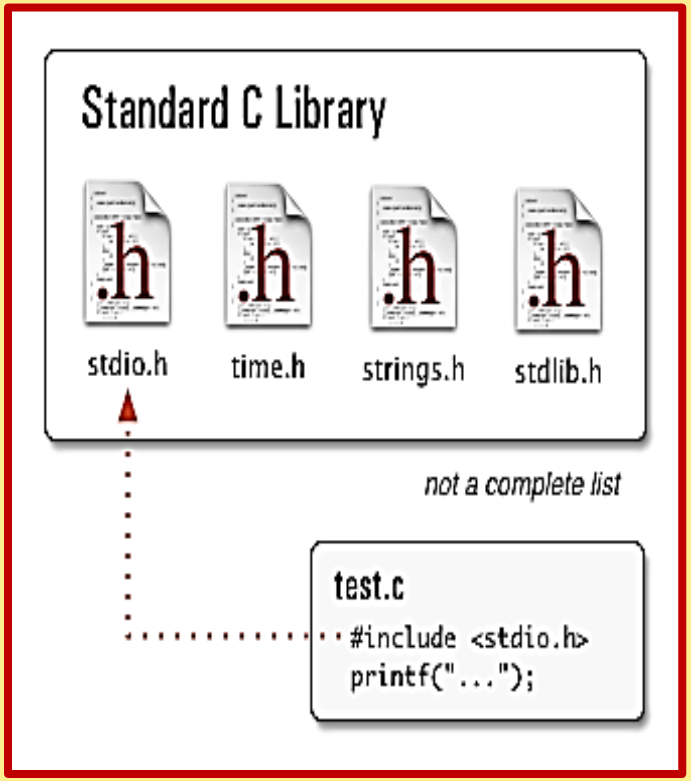


ד"ר שייקה בילו

יועץ ומרצה בכיר למדעי המחשב וטכנולוגית מידע

מומחה למערכות מידע חינוכיות, אקדמיות ומנהליות

קבצים – נושאים



- עבודה עם קבצים ב- C
- הגדרת פתיחת קובץ.
- פתיחת קובץ לקריאה.
- פתיחת קובץ לכתיבה.
- סוגי קבצים שונים והשימוש בהם.
- סגירת קובץ.
- דוגמאות לשימוש בקבצי טקסט ובינאריים

שימוש בקבצים

- שימוש בקבצים מאפשר לנו לשמור נתונים ע"ג התקני אחסון חיצוניים ובעיקר ע"ג הדיסק הקשיח או זיכרון flash.
- שימוש בקבצים מאפשר לשמור כמות גדולה של נתונים, מעבר לקיבולת הזיכרון, ולשמור את הנתונים גם כאשר המחשב כבוי.
- על מנת שהנתונים יהיו נגישים עלינו לשמור אותם בסדר הגיוני, כך כל נתון ישמר בשדה נפרד, לדוגמא שם פרטי.

שימוש בקבצים

- כל השדות המתייחסים לאותו פריט ישמרו באותה רשומה, לדוגמא רשומת סטודנט המכילה את מס' תעודת הזהות שלו, שמו הפרטי ושם המשפחה.
- רשומות המתייחסות לאותו נושא ישמרו באותו קובץ. לדוגמא הסטודנטים להנדסת תוכנה (נתוני כל סטודנט ישמרו ברשומה נפרדת)
- קבצים בעלי קשר ישמרו באותה תיקיה. לדוגמא כל קבצי התלמידים בחוג להנדסת תוכנה.

שימוש בקבצים

- בתוכניות מחשב רבות יש צורך לשמור נתונים
  - תוכנות כמו מעבדי תמלילים, יומני פגישות, הנהלת חשבונות וכדומה שומרות נתונים לשימוש חוזר
- בכל התוכניות שהצגנו שמרנו את הנתונים בזיכרון של המחשב, אך כאשר כיבינו את המחשב הנתונים הללו נמחקו מהזיכרון.
- במקרים רבים רוצים לשמור מידע ולטעון את המידע בחזרה לזיכרון גם לאחר זמן רב.

שימוש בקבצים

- קיימים התקני אחסון חיצוניים לזיכרון המחשב: דיסק קשיח, תקליטון או תקליטור אופטי ועוד.
- באמצעים אלה ניתן לשמור את הנתונים ללא תלות בכיבוי המחשב או בסיום בלתי צפוי של התוכנית.
- לשם כך נרכז את הנתונים שנרצה לשמור בקבצים שהם אוסף של בתים ויש לו שם שנקבע על ידי המשתמש.
- מערכת ההפעלה היא האחראית לנהל את השמירה של הנתונים בקבצים ואת הטעינה שלהם (היא עושה זאת על ידי ניהול טבלה בה רשומים שמות הקבצים ומיקומם).

שימוש בקבצים

- עד כה כל התכניות שכתבנו קראו מקובץ הקלט הסטנדרטי (בעזרת הפונקציות scanf() ו- getchar()) וכתבו לקובץ הפלט הסנדרטי (בעזרת הפונקציות printf() ו-putchar()).
- בשפת C ניתן לקרוא מקבצים ולכתוב לקבצים ישירות מהתכנית עצמה.
- ברב הקבצים קיים פורמט מסוים שמוכר לתוכניות שכותבות או קוראות ממנו.
- פורמט זה מציין כיצד המידע מאוחסן בקובץ.

קבצים בשפת C

- קובץ טקסט ערוך בשורות אשר מופרדות על ידי התווים '\n' ו- '\r' ובסיום כל קובץ נמצא גם התו eof המציין סוף קובץ (end of file).
- במערכת חלונות EOF שקול ל- Ctrl+Z
- לקובץ בינארי אין פורמט מסוים, הפורמט נקבע ע"י המתכנת.
- אבל, לעיתים נרצה להשתמש גם בקבצים נוספים שלא מחוברים אוטומטית לתוכנית.
- קיימים שני סוגי קבצים: קבצי טקסט וקבצים בינאריים.

קבצים – מקורות מידע במחשב

קיימים שני סוגי קבצי עבודה:

1. **<u>קובץ טסקט (text file):</u>**

- **כל מספר/תו מיוצג ע"י ערך ה- ascii שלו.**
- **ניתן לקריאה ע"י עורך (editor) כמו: vi, notepad, nano, pico, emacs ובקובץ זה לא מופיעים תווים מיוחדים כ-null.**
- **הוא ערוך בשורות אשר מופרדות על ידי התווים ' '\n ו- ' '\r ובסיום כל קובץ נמצא גם התו eof המציין סוף קובץ (end of file).**
- **כדי לגשת לקובץ נגדיר משתנה *FILE f שהוא מצביע לקובץ (מצביע *). תו הכוכבית מציין כי f אינו קובץ בעצמו אלא מצביע לראשיתו.**

קבצים – מקורות מידע במחשב

2. **קובץ בינארי (binary file):**

- מיועד להחזקת נתונים, ולא ניתן לקריאה ע"י עורך.
- כל מספר/תו תופסים בקובץ בדיוק אותה כמות של בתים כפי שהם תופסים בזיכרון.
- למשל, המספר 1234.56789 יתפוס בקובץ טקסט 10 בתים ואילו בקובץ בינארי 4 בתים (float).
- מיועד לשמירה וגישה מהירה לכמות גדולה של נתונים.
- רצף תווים בקובת בינארי אינו קריא על ידי עורכי הטקסט למיניהן, אלא רק על ידי התוכנית שיצרה אותו ומכירה את מבנהו.
- קטן יותר מאשר קובץ טקסט.

עבודה עם קבצים

- כדי שנוכל לעבוד עם קבצים (בדומה לקלט/פלט רגיל) יצרו עבורנו בספריה stdio.h מבנה קבוע בשם FILE, ופונקציות קלט/פלט העובדות איתו באופן ישיר.
- בהמשך נתאר את אופן העבודה הזה, אשר מאפשר לנו לכתוב קבצי-טקסט ולקרוא קבצי-טקסט.
- בנוסף נתאר את אופן העבודה עם קבצים בינאריים, שתאפשר לנו לכתוב קבצים בינאריים ולקרוא נתונים מקבצים בינאריים.

שדות המבנה FILE

- המבנה מוגדר בקובץ stdio.h ומכיל מידע אודות קובץ. המבנה הוא:

```
typedef  struct  FILE
{
   short              level;      /* fill/empty level of buffer */
   unsigned           flags;      /* File status flags  */
   char               fd;         /* File descriptor  */
   unsigned char      hold;       /* Ungetc char if no buffer   */
   short              bsize;      /* Buffer size                */
   unsigned char      *buffer;    /* Data transfer buffer */
   unsigned char      *curp;      /* Current active pointer  */
   unsigned           istemp;     /* Temporary file indicator  */
   short              token;      /* Used for validity checking  */
}FILE;                            /* This is the FILE object  */
```

פתיחת קובץ fopen

- לפני התחלת השימוש בקובץ, חייבים לפתוח את הקובץ.
- הפתיחה הופכת את הקובץ לזמין. הפונקציה כלולה בספריה stdio.h האבטיפוס של הפונקציה לפתיחת קובץ:

```
FILE *fopen(const char *filename, const char *mode);
```

- כלומר הפונקציה מחזירה מצביע לטיפוס FILE, שהוא מבנה אשר מוצהר עליו ב stdio.h - בשדות השונים של הטיפוס נעשה שימוש ע"י פונקציות שונות המתייחסות לקבצים. קריאה לפונקציה יוצרת מופע של המבנה ומחזירה מצביע למבנה זה.
- תבנית הפונקציה

```
fopen(file_path, mode);
```

פתיחת קובץ - fopen - המשך

- הסבר התבנית:
  - הפונקציה מחזירה מצביע לנתון מסוג FILE אם הפונקציה נכשלת בפתיחת הקובץ, מוחזר הערך.NULL
  - הארגומנט filepath הוא שם הקובץ שרוצים לפתוח, כולל המסלול. אפשר לכתוב מחרוזת כלואה בין גרשיים או מצביע למחרוזת המאוחסנת בזיכרון. שם הקובץ נקבע בהתאם לכללי מערכת ההפעלה
  - הארגומנט Mode מציין את סוג הגישה לקובץ (טקסט או בינארי) ואת האופן שבו רוצים לפתחו את הקובץ (קריאה, כתיבה או קריאה וגם כתיבה).
  - על מנת לפנות לקובץ עלינו להגדיר משתנה מסוג FILE לדוגמא:

```
FILE *fp;
```

פתיחת קובץ - fopen - המשך

- דוגמא לשימוש בפונקציה לפתיחת קובץ:

```
fopen ("C:\MyBooks\Autoexec.bat", "r");
```

- הכונן הוא C התיקייה היא MyBooks ושם הקובץ הוא Autoexec.bat
- הקובץ נפתח למטרת קריאה (r) אפשר לפתוח קובץ כקובץ קלט (Input) או כקובץ פלט (Output) או כקובץ קלט ופלט (Append) .
- מומלץ לבצע את הפעולה בדרך הבאה:

```
char file_name [] = "C:\MyBooks\Autoexec.bat";
char mode[] = "r";
fopen(file_name, mode);
```

הערכים שהארגומנט mode יכול לקבל

- **r** - לפתיחת קובץ טקסט <u>לקריאה בלבד</u> !!! (read)

אם הקובץ לא קיים מוחזר הערך NULL.

- **w** - לפתיחת קובץ טקסט <u>לכתיבה בלבד</u> !!! (write)

אם הקובץ לא קיים, הוא נוצר, אם הקובץ קיים הוא נדרס.

- **a** - לפתיחת קובץ טקסט <u>להוספה בלבד</u> !!! (append)

אם הקובץ לא קיים, הוא נוצר.

אם הקובץ קיים יתווספו הנתונים בסוף הקובץ.

הערכים שהארגומנט mode יכול לקבל

- **rb** - לפתיחת קובץ בינארי <u>לקריאה בלבד !!!</u>

אם הקובץ לא קיים מוחזר הערך NULL , כמו "r" .

- **wb** - לפתיחת קובץ בינארי <u>לכתיבה בלבד !!!</u>

אם הקובץ לא קיים, הוא נוצר, אם הקובץ קיים הוא נדרס, כמו "w" .

- **ab** פתיחת קובץ בינארי <u>להוספה בלבד !!!</u>

אם הקובץ לא קיים, הוא נוצר.

אם הקובץ קיים יתווספו הנתונים בסוף הקובץ.

כמו "a".

הערכים שהארגומנט mode יכול לקבל

- r+ פתיחת קובץ טקסט לקריאה ולכתיבה, השאר כמו r
- w+ פתיחת קובץ טקסט לקריאה ולכתיבה, השאר כמו w
- a+ פתיחת קובץ טקסט לקריאה ולהוספה, השאר כמו a
- rb+ פתיחת קובץ בינארי לקריאה ולכתיבה, השאר כמו rb
- wb+ פתיחת קובץ בינארי לקריאה ולכתיבה, השאר כמו wb
- ab+ פתיחת קובץ בינארי לקריאה ולהוספה, השאר כמו ab

קובץ טקסט Text file

- קובץ זה הנוצר ע"י שפת C יכול להיקרא ע"י כל תוכנה המסוגלת לקרוא קובץ טקסט.
- הקובץ הוא למעשה רצף של שורות המופרדות ע"י התווים \r ו- (\n-סוף שורה ושורה חדשה). האורך המרבי של שורה הוא 255 תווים.
- השורה מסתיימת בתו אחד או יותר המאותתים על "סוף שורה" בקובץ על דיסק וזאת בהתאם למערכת ההפעלה. במערכות הפעלה Windows- תו "סוף שורה" הוא שילוב של CR-LF, כלומר: \n + \r ביחד.

קובץ טקסט Text file

- בכתיבת מחרוזת לקובץ טקסט, מוחלף התו '\0' מסוף המחרוזת בתו "סוף שורה".
- בקריאה מקובץ טקסט מוסף התו '\0' בסוף המחרוזת.
- בכתיבה לקובץ מוחלף התו שורה חדשה '\n' בתו "סוף שורה."
- בקריאה מקובץ מוחלף התו "סוף שורה" לתו '\n'

תחילת העבודה עם קובץ: פתיחת הקובץ

- כדי לעבוד עם קובץ, ראשית נפתח אותו לעבודה בעזרת הפונקציה fopen.
- למשל:

שם הקובץ

מצב עבודה

```
fopen("info.txt","r");
```

- fopen מקבלת את שם הקובץ, ואת מה שרוצים לעשות איתו:

קריאה – "r", הוספה – "a", כתיבה – "w".

- פתיחת קובץ קיים לכתיבה "w" מוחקת את תוכנו הקודם.
- אם רוצים לשמור את התוכן הקיים נשתמש בהוספה "a" .

פתיחת הקובץ - המשך

- הפונקציה fopen מחזירה מצביע למשתנה מסוג FILE.
- אז השימוש הוא לדוגמא ע"י:

```
FILE  *fp;
fp = fopen("info.txt", "r");
```

- במידה ונעשה ניסיון לפתוח את הקובץ המצוין לקריאה אבל הקובץ אינו קיים – יוחזר NULL.
- במידה ונעשה ניסיון לפתוח את הקובץ לכתיבה או להוספה אבל הקובץ אינו קיים – המחשב ינסה ליצור קובץ חדש, אם הוא לא יצליח יוחזר NULL.

פתיחת קובץ – וידוא פתיחה נכונה

- אחרי פתיחת קובץ נוודא שהפתיחה הצליחה.
- למשל:

```
FILE  *fp;
fp = fopen("in.txt", "r");
if (fp == NULL)
    printf("The file could not be opened\n");
```

דוגמאות לשגיאות בעת פתיחת קובץ

- התייחסות לשם קובץ שאינו קיים בהוראת קריאה מהקובץ
- ניסיון לפתוח קובץ כאשר הכונן אינו מוכן לעבודה
- ניסיון לפתוח קובץ כאשר התיקייה או הכונן אינם קיימים.
- ניסיון לפתוח קובץ מסוג שגוי, לדוגמא ניסיון לפתוח קובץ בינארי בתור קובץ טקסט.
- ניסיון לפתוח קובץ לכתיבה כאשר הכונן חסום או מוגן בסיסמה.
- אין למשתמש הרשאה לפתיחת הקובץ ברשת.

פונקציה לקריאה קלט מובנה fscanf

- הפונקציה fscanf(...) משמשת לקריאת קלט מובנה מקובץ לתוך משתנים.
- תבנית הפונקציה:

int fscnaf(FILE *fp, "מחרוזת בקרה", &mis,&num ...)

- הפונקציה תחזיר את מספר הנתונים שנקלטו, הומרו ואוחסנו בהצלחה. אם הפונקציה נכשלת היא תחזיר EOF
- "מחרוזת בקרה" – היא רשימת ממשקי המשתנים אותם יש לקרוא אשר לפני כל אחד מהם חובה להוסיף %.
- לדוגמא: %d %c %s %f %lf

קריאה מקובץ

- אחרי שפתחנו קובץ לקריאה נוכל לקרוא ממנו קלט ע"י שימוש בפונקציה fscanf, שפועלת כמו scanf, אבל מקבלת כפרמטר ראשון מצביע לקובץ לדוגמא:

```
FILE  *ifp;
ifp = fopen("in.txt", "r");
fscanf(ifp, "%d %c", &num, &tav);
printf("Num=%d,  Tav=%c", num, tav);
```

- המבנה FILE שומר את המקום שהגענו אליו בקובץ, כך שבקריאה הבאה נמשיך משם.
- הפונקציה fscanf() מחזירה את מספר המשתנים שהיא קראה.
- הפונקציה printf() מדפיסה את ערכם על המסך.

קריאה מקובץ – סוף הקובץ

- אם ניסינו לקרוא מקובץ והקובץ הסתיים, אז getc ו- fscanf יחזירו את הערך 1-, שמשמעותו סוף קובץ.

- בספריה stdio.h מוגדר הקבוע EOF (End Of File) שערכו 1- .

- לצורך קריאות-התוכנית, נשתמש בקבוע הזה כשנבדוק אם הגענו לסוף (במקום לרשום 1-):

```
if (fscanf(ptr,"%d", &i) != EOF)
```

(כמו שרושמים NULL במקום 0 כדי לבדוק אם זו כתובת 0)

שאלות?

פתיחת קובץ לכתיבה


```
fopen("out.txt", "w");
```

הפונקציה מחזירה מצביע ל-FILE, לכן נכתוב:

```
FILE  *ofp;
ofp = fopen("out.txt", "w");
```

כאמור פתיחת קובץ להוספה:

```
ofp = fopen("out.txt", "a");
```

פונקציה לכתיבת פלט מובנה fprintf

- הפונקציה כותבת לקובץ את כל מה שהוגדר במחרוזת הבקרה, כאשר תווי הבקרה המוגדרים בתו % מאפשרים לשלוט באופן כתיבת הנתונים. תבנית הפונקציה:

```
int fprintf(FILE *ofp, "מחרוזת בקרה", var1…)
```

- דוגמא :

```
fprintf (ofp, "%10d %8.2f %25s\r\n" ,a, b, c);
```

- ישנם 3 משתנים ולכן גם 3 תווי המרה.

פונקציה לכתיבת פלט מובנה fprintf

- הכתיבה מתבצעת משמאל לימין.
- למשתנה a מוקצים 10 תווים + רווח. למשתנה b מוקצים 8 תווים מתוכם 2 לאחר הנקודה העשרונית + רווח.
- למשתנה c מוקצים 25 תווים והוא ייכתב כמחרוזת.
- השורה מסתיימת בתו שורה חדשה.(\n)
- אם הפונקציה מצליחה - יוחזר מספר התווים שנכתבו. אם הפונקציה נכשלה - יוחזר EOF .

```
fprintf (ofp, "%10d %8.2f %25s\r\n" ,mis, num, str);
```

כתיבה לקובץ

- אחרי שפתחנו קובץ לכתיבה או הוספה אפשר לכתוב בו.
- הכתיבה תתבצע למשל ע"י:

```
fprintf (ofp, "%d %c\r\n", mis, tav);
```

- השימוש זהה לשימוש ב-printf, מלבד זה שהפרמטר הראשון הוא מצביע לקובץ שאליו נכתוב.

דוגמה לכתיבה לקובץ


```
#include <stdio.h>
int main()
{
        FILE *ptr_file;
        int i;
        ptr_file = fopen("output.txt", "w");
        if (!ptr_file)
                return 1;
        for (i=1; i<=10; i++)
                fprintf(ptr_file,"%d\n", i);
        fclose(ptr_file);
        return 0;
}
```

דוגמה לקריאה מקובץ


```
#include <stdio.h>
int main()
{
        FILE *ptr_file; int i,number;
        ptr_file = fopen("output.txt","r");
        if (!ptr_file) return 1;
        for (i=1; i<=10; i++)
        {
                fscanf(ptr_file,"%d\n",&number);
                printf("Number = %d\n",number);
        }
        fclose(ptr_file);
return 0;
}
```

שאלות?

שימוש בפקודה rewind

- אחרי שפתחנו קובץ לכתיבה או הוספה אפשר לכתוב בו אבל אז אנחנו נעים קדימה עד לסופו.
- כדי לחזור לתחילת הקובץ, כלומר למצביע על התא הראשון אנחנו משתמשים בפקודה: rewind(ofp);
- משמעות הפקודה היא הבאת נקודת הקריאה לתחילת הקובץ כך שניתן כעת להציג את התוכן מההתחלה ללא צורך לפתוח את הקובץ לקריאה שוב.

עבודה עם קבצים - דוגמא:

- נכתוב תוכנית שמקבלת שם של קובץ וסופרת כמה שורות יש בו.
- נניח שאורך כל שורה ואורך שם הקובץ הוא 80 לכל היותר.

```
#define SIZE 81
int main()
{
   FILE *ifp;
   int count = 0;
   char line[SIZE], filename[SIZE];
   gets(filename);
   ifp = fopen(filename, "r");
....
```

משתנה לספירת השורות
קולטים את שם הקובץ

פתיחת הקובץ לקריאה

תזוזות בקובץ

איך אנו נעים על פני הקובץ?

- כל קריאה ממשיכה מאיפה שסיימנו לקרוא.
- כל כתיבה ממשיכה מאיפה שכתבנו בעבר.
- אפשר לקרוא ולכתוב לאותו קובץ במקומות שונים!
- append מול trunk!

אינדקסים בקובץ!

- יש לנו אנדקס למיקום קריאה ואינדקס למיקום כתיבה.
- כל קריאה / כתיבה מזיזה את האינדקס הרלוונטי.
- יש לנו אפשרות להזיז את האינדקס בעצמנו.

עבודה עם קבצים - דוגמא:

בודקים אם הפתיחה הצליחה

```
if (ifp == NULL)
{
    printf("File could not be opened\n");
    return -1;
}
```

קוראים שורות כל עוד יש, וסופרים אותן

```
while(fgets(line, SIZE, ifp) != NULL)
        count++;
printf("There are %d lines\n", count);
```

סגירת הקובץ בסיום

```
fclose(ifp);
return 0;
}
```

פונקציה לכתיבת תו fputc()

- הפונקציה כותבת תו אחד לקובץ ש- fp מצביע עליו. לאחר הכתיבה יקודם סמן הקובץ למיקום הבא. הפונקציה כלולה בספריה.stdio.h
- האבטיפוס של הפונקציה לכתיבת תו לקובץ:

```
FILE *fputc(int ch, FILE *fp);
```

- כלומר הפונקציה מחזירה מצביע לטיפוס FILE, שהוא מבנה אשר מוצהר עליו ב: stdio.h
- תבנית הפונקציה:

```
fputc(variable_name,  FILE *fp);
```

- דוגמא לשימוש בפונקציה לכתיבת תו לקובץ:

```
fputc(msg[i], fp);
```

- אם הכתיבה נכשלת או שמגיעים לסוף הקובץ, תחזיר הפונקציה EOF

פונקציה לקריאת תו fgetc()

- הפונקציה ניגשת לקובץ ש fp-מצביע עליו וקוראת תו אחד ומחזירה אותו. לאחר הקריאה יקודם סמן הקובץ לתו הבא.
- האבטיפוס של הפונקציה לקריאת תו מקובץ:

```
FILE *fgetc(FILE *);
```

- כלומר הפונקציה מחזירה מצביע לטיפוס FILE, שהוא מבנה אשר מוצהר עליו ב: stdio.h
- תבנית הפונקציה:

```
fgetc(FILE *fp);
```

- דוגמא לשימוש בפונקציה לקריאת תו מקובץ:

```
msg[i] = fgetc(fp);
```

- אם הקריאה נכשלת או שמגיעים לסוף הקובץ, תחזיר הפונקציה.EOF

קבצים - קריאה וכתיבה של תווים בודדים

- על מנת לקרוא תו מקובץ נשתמש בפונקציה

```
int fgetc (FILE *fp)
```

- פונקציה זו פועלת בדיוק כמו הפונקציה getchar(), רק שהקלט הוא מהקובץ שמצביעו fp במקום מקובץ הקלט הסטנדרטי.
- הפונקציה מחזירה EOF בסוף הקובץ או במקרה של טעות.

- על מנת לכתוב תו מקובץ נשתמש בפונקציה

```
int fputc (int c, FILE *fp)
```

- פונקציה זו פועלת בדיוק כמו הפונקציה putchar(), רק שהפלט הוא לקובץ שמצביעו fp במקום לקובץ הפלט הסטנדרטי.

קבצים - קריאה וכתיבה מפורמטת

- הפונקציה מחזירה את התו שנכתב או EOF במקרה של טעות.

int fscanf (FILE *fp, …)

- פונקציה זו פועלת בדיוק כמו הפונקציה scanf(), רק שהקלט הוא מהקובץ שמצביעו fp במקום מקובץ הקלט הסטנדרטי.

int fprintf (FILE *fp, …)

- פונקציה זו פועלת בדיוק כמו הפונקציה printf(), רק שהפלט הוא לקובץ שמצביעו fp במקום לקובץ הפלט הסטנדרטי.

קבצים – דוגמא - כתיבה

```
/* Create a sequential file*/
#include <stdio.h>

main()
{
     int account;
     char name[30];
     float balance;
     FILE *cfPtr;

     if ((cfPtr = fopen("clients.txt", "w"))==NULL)
          printf("File could not be opened\n");
     else{
          printf("Enter the account, name, and balance.\n");
          printf("Enter EOF to end input.\n");
          printf("? ");
          scanf("%d%s%f", &account, name, &balance);
          while (!feof(stdin)){
               fprintf(cfPtr, "%d %s %.2f\n", account, name, balance);
               printf("? ");
               scanf("%d%s%f", &account, name, &balance);
          }
          fclose(cfPtr);
     }
     return 0;
}
```

קבצים – פלט הקובץ

clients.txt
1 Bob 200.00
2 Lee 400.00
3 Rich -5.00

קבצים – דוגמא - קריאה

```
/*Reading and printing a sequential file */
#include <stdio.h>

main()
{
    int account;
    char name[30];
    float balance;
    FILE *cfPtr;

    if ((cfPtr = fopen("clients.txt", "r"))==NULL)
        printf("File couldn't be opened.\n");
    else{
        printf("%-10s%-13s%s\n", "Account", "Name", "Balance");
        fscanf(cfPtr, "%d%s%f", &account, name, &balance);

        while (!feof(cfPtr)){
            printf("%-10d%-13s%7.2f\n", account, name, balance);
            fscanf(cfPtr, "%d%s%f", &account, name, &balance);
        }
        fclose(cfPtr);
    }
    return 0;
}
```

Account	Name	Balance
1	Bob	200
2	Lee	300
3	Rich	-5

נניח שרוצים לשנות Lee ב- Dominic.

צריכים לפתוח את הקובץ לקריאה וכתיבה (r+)

להגיע לרשימה 2 ולהחליף אותה ב- "2 Dominic 3000"

המחרוזת החדשה גדולה מ-"Lee"

זה יהרוס את הרשימות הבאות בקובץ זה!

פונקציות קלט/פלט של רצף תווים

- **קריאת רצף תווים:**

```
char* fgets(char* line, int max, FILE* stream);
```

קורא לכל היותר את max-1 התווים (שומר מקום ל-'\0') מהקובץ המוצבע ע"י stream לתוך line. יעצור אם נתקל ב-'\n' (לאחר שהעתיקו לתוך line). יחזיר את line, או EOF.

- **כתיבת רצף תווים:**

```
char* fputs(const char* line, FILE* stream);
```

כותב את רצף התווים שב-line, לתוך קובץ המוצבע ע"י stream. לא יכתוב את '\0' שבסוף line לתוך הקובץ.

פונקציה לקליטת מחרוזת (שורה) fgets

- הפונקציה דומה לפונקציה gets() בכך שהיא קוראת שורת טקסט מ-stream של קלט.
- המתכנת יכול להגדיר את שם ה- stream הרצוי ואת מספר התווים המרבי שיקראו.
- הפונקציה ניגשת לקובץ במקום ש- fp מצביע עליו וקוראת עד ל-(Max פחות 1) תווים או עד לסוף שורה (\n, CR, LF) הנתונים מאוחסנים במשתנה (line) כולל NULL מסיים.
- הקריאה לתוך מערך מסוג char
- האב טיפוס:

```
char *fgets(char *str, int n, FILE *fp)
```

פונקציה לקליטת מחרוזת (שורה) fgets

- תבנית הפונקציה:

```
char *fgets (char *line, int Max, FILE *fp);
```

- דוגמא לשימוש בפונקציה:

```
FILE *fp=fopen("data.txt", "rt")
char buffer[100];
while(fgets(buffer, 100,fp));
```

- הקריאה מתבצעת ל- buffer עד סוף הקובץ.
- במקרה של שגיאה או סוף הקובץ, תחזיר הפונקציה NULL

פונקציה לכתיבת שורה fputs()

- הפונקציה כותבת לקובץ את המחרוזת שבמשתנה (line) ללא NULL מסיים.
- הפונקציה איננה מוסיפה \n, CR או LF בסוף השורה.
- הפונקציה מחזירה EOF אם נכשלה או את התו האחרון אם הצליחה
- האבטיפוס של הפונקציה לכתיבת שורה לקובץ :

```
int  fputs(const char *, FILE *);
```

- תבנית הפונקציה:

```
int      FPUTS(char *line, FILE *fp);
```

- דוגמא לשימוש בפונקציה לכתיבת שורה לקובץ:

```
FILE *fp=fopen("data.txt", "wt");
fputs (f_name, fp);
```

סגירת קובץ fclose

- לאחר סיום השימוש בקובץ, חייבים לסגור את הקובץ. הסגירה עושה את הקובץ ללא זמין. הפונקציה כלולה בספריה stdio.h האבטיפוס של הפונקציה לסגירת קובץ:

```
FILE *fclose(const char *filename);
```

- כלומר הפונקציה מחזירה מצביע לטיפוס FILE שהוא מבנה אשר מוצהר עליו ב: stdio.h-תבנית הפונקציה לסגירת קובץ:

```
fclose (FILE *fp);
```

- דוגמא לשימוש בפונקציה לסגירת קובץ:

```
fclose (fp);
```

- הפונקציה מחזירה EOF אם סגירת הקובץ נכשלה.

איתור סוף קובץ EOF

- סוף קובץ אפשר לאתר בדרכים הבאות:
  - לבדוק אם התו התורן שנקרא הוא התו .EOF עלולה להיות בעיה בקבצים בינאריים כיוון שהתו EOF ערכו הוא 1- והוא יכול להופיע באמצע הקובץ. דוגמא לשימוש:

```
while ((c=fgetc(fp)!=EOF)
{
        אפשר להמשיך לקרוא ולעבד את הקובץ
}
```

  - באמצעות הפונקציה ,() feof אשר מחזירה 0 אם לא הגענו לסוף הקובץ או ערך שונה מ-0 אם הגענו לסוף הקובץ. תבנית הפונקציה:

```
int feof(FILE *fp);
```

סגירת קובץ

- כשנסיים להשתמש בקובץ שפתחנו, נסגור אותו:

```
FILE  *fp;
fp = fopen("out.txt", "r");
....
fclose(fp);
```

- הסגירה תתבצע ע"י קריאה לפונקציה fclose() עם מצביע לקובץ (למבנה FILE) שפתחנו.

שאלות?

קבצים - קריאה וכתיבה מפורמטת

- כאשר משתמשים בפונקציות של קלט/פלט מפורמט, הנתונים מאותו טיפוס יכולים להכיל מספר תווים שונה, למשל 14 ו-2074 שניהם int ולכן אתו מספר בתים בזיכרון, אבל מודפסים במסך ובקובץ בגדלים שונים.
- לכן גישה סדרתית עם fprintf() ו- fscanf() לא נוחה לעדכון של קובץ קיים.
- אם רוצים לגשת ולעדכן את הקובץ במקומות שונים, יותר נוח ליצור רשימות של אורך זהה ומוגדר מראש.

קבצים - קריאה וכתיבה לא סדרתית

- אם כל רשימה בקובץ נשמרת באותו גודל אז ניתן לומר:
  - ניתן לחשב את המיקום של כל רשימה בקובץ
  - ניתן להחליף את התוכן של רשימה כלשהי בקובץ בלי לפגוע ברשומות אחרות.
- נשתמש בפונקציות קריאה וכתיבה שעובדות עם בתים ולא תווים.

קריאה לפי כמות הבתים


```
sizet fread(void *ptr, sizet size, sizet nitems, FILE *stream)
```

מספר יחידות שנקראו בפועל

מצביע לתחילת הבלוק בזיכרון

מספר הביטים שיחידה תופסת

מספר יחידות שרוצים לקראו

מצביע לקובץ

הפונקציה הזאת קוראת בלוק של ביטים ממקום הנוכחי בקובץ לבלוק בזיכרון ש- ptr מצביע להתחלתו.

כתיבה לפי כמות הבתים


```
size_t fwrite(void *ptr, size_t size, size_t nitems, FILE* stream)
```

מספר יחידות שנכתבו בפועל

מצביע לתחילת הבלוק בזיכרון

מספר הביטים שיחידה תופסת

מספר יחידות שרוצים לכתוב

מצביע לקובץ

הפונקציה הזאת כותבת בלוק של ביטים בזיכרון ש- ptr מצביע להתחלתו למקום הנוכחי בקובץ.

תזוזה בקובץ

- FILE מכיל שדה כתובת מורחב offset שקובע את המקום שאליו תתייחס fread() או fwrite() הבא.
- שדה ה- offset מתעדכן אחרי כל קריאה וכתיבה.
- ניתן לשנות את תוכן שדה ה- offset בלי לבצע קריאה או כתיבה ע"י שימוש בפונקציות rewind(), fseek()
- פונקציה ftell() מקבלת מצביע ל- FILE ומחזירה את ה- offset שלו.

תזוזה בקובץ

- פונקציה fseek() מאפשרת לנוע למקום נתון חדש (תנועה על לפי ביטים) בתוך המבנה של קובץ.

```
int fseek(FILE *stream, long offset, int whence);
```

מצביע לקובץ

ההתייחסות למיקום חדש, תנוע ע"י דיטים

כמות התווים לקריאה החל מהכתובת החדשה

- SEEK_SET מתחילת הקובץ קדימה
- SEEK_CUR מקום נוכחי קדימה
- SEEK_END מסוף הקובץ אחורה

```c
#include <stdio.h>
typedef struct clientData {
    int acctNum;
    char lastName[15];
    char firstName[10];
    float balance;
}Client;
/* Create random access file*/
void create(char* name)
{
    FILE *cfPtr;
    Client blankClient={0, "","",0.0};
    int i;

    if ((cfPtr=fopen(name, "w"))==NULL)
        printf("File can not be openned.\n");
    else{
        for (i=1; i<=100; i++)
            fwrite(&blankClient,sizeof(Client),1,cfPtr);
        fclose(cfPtr);
    }
}
```

```c
/*Write to File*/
void writeTofile(char* name)
{
    FILE *cfPtr;
    Client client;

    if ((cfPtr = fopen(name, "r+")) == NULL)
        printf("File can not be openned.\n");
    else{
        printf("Enter account number"
            " (1 to 100, 0 to end input)\n? ");
        scanf("%d", &client.acctNum);

        while(client.acctNum !=0){
            printf("Enter lastname, firstname, balance\n");
            scanf("%s%s%f",&client.lastName,
                      &client.firstName,&client.balance);
            fseek(cfPtr,(client.acctNum -1)*sizeof(Client),
            SEEK_SET);
            fwrite(&client,sizeof(Client),1,cfPtr);
            printf("Enter account number\n ");
            scanf("%d", &client.acctNum);
        }
    }
    fclose(cfPtr);
}
```

```c
/*Read the whole File*/
void readFile(char* name)
{
    FILE* cfPtr;
    Client client;

    if ((cfPtr = fopen(name, "r")) == NULL)
        printf("File can not be openned.\n");
    else{
        printf("%-6s%-16s%-11s%10s\n","Acct", "Last Name",
            "First Name", "Balance");
        fread(&client, sizeof(Client),1,cfPtr);

        while (!feof(cfPtr)){
            if (client.acctNum!=0){
                printf("%-6d%-16s%-11s%10.2f\n",
                client.acctNum, client.lastName,
                client.firstName, client.balance);
            }
            fread(&client, sizeof(Client),1,cfPtr);
        }
    fclose(cfPtr);
    }
}
```

```c
void updateRecord(char* name, int acctNum)
{
    Client client;
    float transaction;
    FILE* cfPtr;

    if ((cfPtr = fopen(name, "r+")) == NULL)
        printf("File can not be openned.\n");
    else{
        fseek(cfPtr,(acctNum -1)*sizeof(Client), SEEK_SET);
        fread(&client, sizeof(Client),1,cfPtr);
        if(client.acctNum==0)
            printf("Account #%d has no inforamtion. \n", acctNum);
        else{
            printf("%-6d%-16s%-11s%10.2f\n\n",
                      client.acctNum,client.lastName,
                   client.firstName,client.balance);
            printf("Enter charge (+) or payment (-):");
            scanf("%f",&transaction);
            client.balance+=transaction;
            printf("%-6d%-16s%-11s%10.2f\n\n",
                      client.acctNum,client.lastName,
                   client.firstName,client.balance);
            fseek(cfPtr,(acctNum -1)*sizeof(Client), SEEK_SET);
            fwrite(&client,sizeof(Client),1,cfPtr);
        }
        fclose (cfPtr);
    }
}
```

```
main()
{
  int account;
  create("client.dat");
  writeTofile("client.dat");
  readFile("client.dat");
  printf("Enter account to update (1-100):");
  scanf("%d",&account);
  updateRecord("client.dat",account);
  readFile("client.dat");
}
```

```
Enter account number (1 to 100, 0 to end input)
? 1
Enter lastname, firstname, balance
AAA A 100
Enter account number
 30
Enter lastname, firstname, balance
BBB B 300
Enter account number
 40
Enter lastname, firstname, balance
VVV V -100
Enter account number
 0
Acct  Last Name      First Name    Balance
1     AAA            A             100.00
30    BBB            B             300.00
40    VVV            V             -100.00
Enter account to update (1-100): 30
30    BBB            B             300.00

Enter charge (+) or payment (-):-500
30    BBB            B            -200.00

Acct  Last Name      First Name    Balance
1     AAA            A             100.00
30    BBB            B            -200.00
40    VVV            V             -100.00
Press any key to continue . . .
```

קבצים מיוחדים

- שפת C מתייחסת גם אל הקלט (מהמקלדת) והפלט (אל המסך) כאל קריאה/כתיבה של קבצים.
- בתחילת התוכנית נפתחים באופן אוטומטי הקבצים עבור זה (והם נסגרים באופן אוטומטי בסיומה).
  - stdin – הקלט הסטנדרטי (מהמקלדת).
  - stdout – הפלט הסטנדרטי (למסך).
  - stderr – פלט של הודעות שגיאה (מנותב גם הוא למסך).
  - כך למשל הפקודה printf(“%d”, 10); היא למעשה:

fprintf (stdout, “%d”, 10);

קבצים מיוחדים - שימוש

- למשל, אם רוצים לקרוא שורת-קלט מהמקלדת למחרוזת, אבל להגביל את מספר התווים ל-20 (כולל יריידת השורה וה- '\0'), אז אפשר להשתמש ב-

```
fgets(str, 20, stdin);
```

- אם התכוונו שהתו האחרון יהיה יריידת-שורה וזה לא כך, אז נוכל פשוט לבדוק זאת ולתת הודעת שגיאה למשתמש.

דוג' – קובץ בינארי (1)


```
typedef struct  Nums
{
   int     x;  double  y;
} Nums;
int main( int argc, char* argv[] )
{
   FILE *src=NULL;
   Nums data[] = {{1, 1.1}, {2, 2.2}, {3, 3.3}}, *trg;
   int i, num, SIZE = sizeof(data)/sizeof(Nums);
   if (argc != 2)
   {
         printf("Bad usage \n");   return 1;
   }
```

דוג' – קובץ בינארי (2)


```
// write the binary data
src = fopen(argv[1], "wb");
if(NULL == src)
{
        printf("Failed opening file %s for writing", argv[1]);
        return 1;
}
num = fwrite(data , sizeof(Nums) , SIZE , src);
printf("Wrote %d items to %s.\n", num, argv[1]);
fclose(src);
```

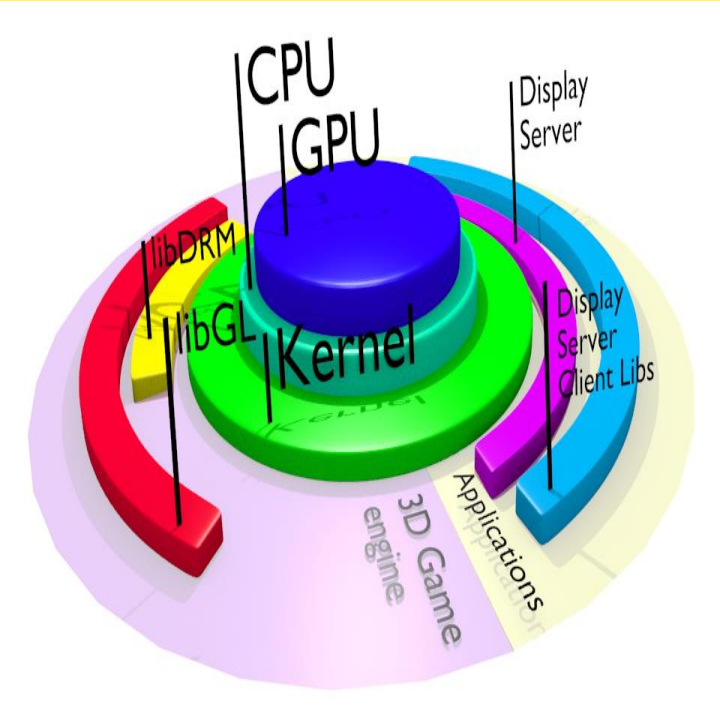
דוג' – קובץ בינארי (3)


```
// read the binary data
src = fopen(argv[1], "rb");
if (NULL == src)
{
      printf("Failed opening file %s for reading", argv[1]);
      return 1;
}
trg = (Nums *) malloc (SIZE * sizeof(Nums));
if (NULL==trg)
{
    printf("Failed mallocing \n");
    return 1;
}
```

דוג' – קובץ בינארי (4)


```
  num = fread(trg, sizeof(Nums) , SIZE , src);
  printf("Read %d items\n" , num);
  // print the read data
  for(i=0; i<SIZE; i++)
    printf("trg[%i]= <%d, %lf> \n", i, trg[i].x, trg[i].y);
  fclose(src);
}
```

קבצי - סיכום



- עבודה עם קבצים ב- C.
- הגדרת פתיחת קובץ/סגירת קובץ
- פתיחת קובץ לקריאה.
- פתיחת קובץ לכתיבה.
- פתיחת קובץ להוספה.
- סוגי קבצים שונים והשימוש בהם.
- אפשר לבצע ב- C קלט/פלט עם קבצים בדומה לקלט/פלט עם המקלדת והמסך.

שאלות?

תרגיל כיתה

1. כתוב תכנית בשפת C אשר תקרא שם וציון של num סטודנטים מהמשתמש ותאחסן אותם בקובץ. התוכנית תדפיס את הנתונים שנקלטו בסיום הקלט.

2. כתוב תכנית בשפת C אשר תקרא שם וציון מקובץ של num סטודנטים שאוחסנו בקובץ. אם הקובץ קיים בעבר, יש להוסיף את המידע של הסטודנטים לקובץ הקיים.

3. כתוב תכנית בשפת C אשר תכתוב את נתוני המערך של מבנים בקובץ ע"י שימוש בפונקציה fwrite(). לאחר מכן יש לקרוא את המערך מהקובץ ולהציג את הנתונים על המסך.

תרגיל כיתה

4. כתוב תכנית המבצעת את המשימות הבאות ע"י שימוש בקבצי טקסט ופונקציות:

- תבנה ותפעיל פונקציה שתייצר קובץ מספרים בשם numbers.txt שכל רשומה בו תכלול מספר שלם בן 3 ספרות.
- הפונקציה תבנה לפחות 10 רשומות בקובץ שתכלולנה את כל המספרים שיוזנו ע"י המשתמש.
- תבנה ותפעיל פונקציה שתדפיס את תוכן הקובץ למסך לאחר שיקלטו לתוכו עשרת המספרים שהוזנו ע"י המשתמש.
- תבנה ותפעיל פונקציה שתחשב ותדפיס מהו המספר הגדול ביותר בקובץ, הקטן ביותר בקובץ ומהו ממוצע המספרים בקובץ.

תרגיל כיתה

5. כתוב תכנית המבצעת את המשימות הבאות ע"י שימוש בקבצי טקסט ופונקציות:

- תבנה ותפעיל פונקציה שתייצר קובץ סטודנטים בשם students.txt שכל רשומה בו תכלול שלושה שדות לפי הסדר הבא: ציון ממוצע, שם פרטי ושם משפחה. הפונקציה תבנה לפחות 10 רשומות בקובץ.
- תבנה ותפעיל פונקציה שתדפיס את הקובץ לאחר שיקלטו לתוכו לפחות פרטי 10 סטודנטים.
- תבנה ותפעיל פונקציה שתעבור על הקובץ students.txt ותמצא מיהו הסטודנט בעל ממוצע הציונים הגבוה ביותר ותדפיס את הפרטים שלו/שלהם.

תרגיל כיתה

6. כתוב תכנית המבצעת את המשימות הבאות ע"י שימוש בקבצי טקסט ופונקציות:

- תייצר פונקציה שתבנה קובץ לשימור קוד מורס של הספרות 0 עד 9 בשם morsecode.txt שכל רשומה בו תכלול שדה מחרוזתי אחד בו רשום קוד מורס של מספר. הקובץ יהיה ממוין מ- 0 עד 9.
- תייצר פונקציה שתדפיס את הקובץ morsecode.txt לאחר שיקלטו לתוכו את קידודי המורס של הספרות 0 עד 9 כולל.
- תייצר פונקציה שתקלוט מהמשתמש מספר שלם עד 9 ספרות, תעבור על הקובץ ותמצא מה הקידוד של כל ספרה ותדפיס את הקידוד על המסך.

תרגיל כיתה

7. כתוב תכנית המבצעת את המשימות הבאות ע"י שימוש בקבצי טקסט ופונקציות:

- תבנה ותפעיל פונקציה שתייצר קובץ ציוני סטודנטים בשם marks.txt שכל רשומה בו תכלול שלושה שדות לפי הסדר הבא: מספר ת.ז., שם וציון. קליטת הציונים לקובץ תסתיים כאשר ייקלט לשדה ת.ז. הערך 0.
- תבנה ותפעיל פונקציה שתדפיס את הקובץ לאחר שיקלטו לתוכו לפחות פרטי 10 סטודנטים.
- תבנה ותפעיל פונקציה שתעבור על הקובץ marks.txt ותאפשר מציאת סטודנט לפי ת.ז., או לפי ציון ותדפיס את פרטיו על המסך.

תרגיל כיתה

8. כתוב תכנית המבצעת את המשימות הבאות ע"י שימוש בקבצי טקסט ופונקציות:

- תבנה ותפעיל פונקציה שתייצר קובץ לניהול חשבון בנק בשם clients.txt שכל רשומה בו תכלול שלושה שדות לפי הסדר הבא: מספר חשבון בנק, שם הלקוח ויתרה בחשבון.
- קליטת הנתונים לקובץ תסתיים כאשר ייקלט לשדה מספר חשבון הערך 0.
- תבנה ותפעיל פונקציה שתדפיס את הקובץ לאחר שיקלטו לתוכו לפחות פרטי 10 לקוחות.
- תבנה ותפעיל פונקציה שתעבור על הקובץ clients.txt ותאפשר עדכון יתרה לפי מספר חשבון, עדכון יתרה לפי שם לקוח.

תרגיל כיתה

9. כתוב תכנית המבצעת את המשימות הבאות ע"י שימוש בקבצי טקסט ופונקציות:

- תבנה ותפעיל פונקציה שתייצר קובץ לניהול צפייה של צפר בציפורים בשם birdsfile.txt שכל רשומה בו תכלול מספר ציפורים בהן צפה הצפר, החודש בו נערכה הצפייה והיום בחודש.
- קליטת הנתונים לקובץ תתחיל בקביעת כמה חודשי צפייה היו לצפר, אחר כך תמשיך בהזנת מספר ימי הצפייה בכל חודש ותסתיים במספר הציפורים שנצפו בכל יום.
- תבנה ותפעיל פונקציה שתדפיס את הקובץ לאחר שיקלטו לתוכו פרטי חודשי הצפייה, מספר הימים ומספר הציפורים.

תרגיל כיתה

10. כתוב תכנית המבצעת את המשימות הבאות ע"י שימוש בקבצי טקסט ופונקציות:

- תבנה ותפעיל פונקציה שתייצר קובץ לניהול ספרייה בשם booksfile.txt שכל רשומה כוללת את השדות הבאים:
- שם ספר, שם מחבר, שם הוצאה לאור, שנת ההוצאה, מחיר, מספר סידורי.
- התוכנית תאפשר קליטת פרטי הספר ויצירת מאגר ספרים, מחיקת ספר מהמאגר, הוספת ספר חדש למאגר, מציאת ספר לפי שם הספר, לפי שם המחבר ולפי מספר סידורי, הדפסת דוח מלאי הספרים בספריה.

תרגיל כיתה

11. נתון קובץ טקסט בשם CdList.txt המיועד לניהול אוסף הדיסקים הפרטים שלך, בקובץ רשומים בכל רשומה שלושה פרטים לגבי כל דיסק: שם הדיסק, שם הלהקה/האומן ושנת הפרסום של הדיסק.

- לא קיים מידע על מספר הדיסקים באוסף הפרטי אשר פרטיהם רשומים בקובץ.
- עליך לכתוב פונקציה העוברת על הקובץ ומאתרת בו כמה דיסקים יש לכל להקה/אומן, מהי שנת הפרסום של הדיסק הישן ביותר ושל הדיסק החדש ביותר.
- את שלושת הנתונים הפונקציה תדפיס בסיום ריצתה.
- הנתונים מופיעים בקובץ לפי הסדר הבא: שנת הפרסום, רווח, שם הלהקה/האומן, רווח ושם הדיסק.

תרגיל כיתה - המשך

2010 Loud RIHANNA
2012 Unapologetic RIHANNA
1973 Ring Ring ABBA
1979 Voulez-Vous ABBA
1980 Super Trouper ABBA
1981 The Visitors ABBA
1986 A Kind of Magic QUEEN
1974 Waterloo ABBA
1976 Arrival ABBA
1977 The Album ABBA
1989 The Miracle QUEEN
1991 Innuendo QUEEN
1995 Made in Heaven QUEEN
2007 Good Girl Gone Bad RIHANNA

תרגיל כיתה - המשך

דוגמה לפלט ההרצה לקובץ הנתון:

Number of RIHANNA Discs 2

Number of ABBA Discs 6

Number of QUEEN Discs 2

New Disc details(year, Disc_Name, Band_Name)

2012 Unapologetic RIHANNA

Old Disc details(year, Disc_Name, Band_Name)

1973 RingRing ABBA

שאלות?