

קבצים



קרון כליף

ביחידה זו נלמד:

מוטיבציה לעבודה עם קבצים □

פעולות על קבצים: □

- פתיחת קובץ

- קריאה מקובץ

- כתיבה לקובץ

- סגירת קובץ

סוגי קבצים: □

- בינאריים

- טקסט

פעולות נוספות: `fseek`, `ftell`, `rewind`, `ferror`, `feof` □

קבצים - מוטיבציה

- כאשר אנחנו כותבים תוכנה, המידע אשר אנחנו מכניסים בכל הרצה הולך לאיבוד עם סיומה, מאחר והתוכנה רצה בזיכרון ה-RAM של המחשב
- היינו רוצים לשמור את המידע בין ההרצות בזיכרון הקבוע (Hard Disk) כדי:
 - ליצור תוכנה בעלת משמעות ורצף
 - להקל עלינו בעת בדיקות התוכנה
 - קריאת נתונים מקובץ מוכן מראש ולא מהמשתמש
 - שמירת נתונים לשימוש ע"י תוכניות אחרות
- ניתן לשמור את המידע בכמה אופנים שונים:
 - DB (לא יילמד בקורס זה)
 - קבצים

עבודה כללית עם קבצים - מוטיבציה

□ יתכן ותהייה לנו תוכנית שנרצה את הפלט שלה לשמור לקובץ

□ יתכן ותהייה לנו תוכנית שנרצה שהקלט שלה יהיה מקובץ, ולא מהמקלדת

□ דוגמאות:

■ קריאת אוסף מספרים מקובץ בפורמט ידוע מראש והצגת סכום המספרים

■ כתיבת תוכנית המכינה דו"ח כלשהו ושומרת את הדוח בקובץ, בנוסף להצגתו על המסך

דוגמא לעבודה עם קובץ

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

FILE * הוא מבנה למשתנה מטיפוס קובץ

```
void main()  
{
```

fopen היא פקודה הפותחת את הקובץ בזיכרון. מקבלת כפרמטר ראשון את שם הקובץ וכפרמטר שני את אופן הפתיחה

```
FILE* f = fopen("myFile.txt", "w"),
```

"w" משמע פתיחת הקובץ לכתיבה, ודריסתו אם קיים

```
if (f == NULL)  
{
```

```
printf("Failed opening the file. Exiting!\n");
```

```
return;
```

```
}
```

בדיקה האם פתיחת הקובץ הצליחה

```
fprintf(f, "%s %d %lf\n", "KerenK", 28, 99.8);
```

```
fputs("Hello World!", f);
```

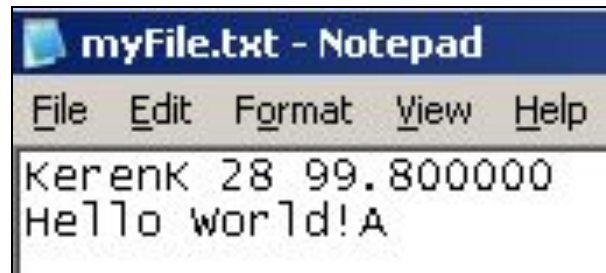
```
fputc('A', f);
```

פקודות כתיבה לקובץ

```
fclose(f);
```

סגירת הקובץ

```
}
```



מבנה עבודה כללי עם קבצים

- קובץ הוא משתנה מטיפוס FILE* ומוגדר ב- `stdio.h`
- בתחילת העבודה יש לפתוח קובץ לעבודה
- יש לבדוק האם פתיחה הצליחה
- לבסוף יש לסגור את הקובץ

```
void main()
{
    FILE* f = fopen(<file name>, <open parameters>);
    if (f == NULL)
    {
        printf("Failed opening the file. Exiting!\n");
        return;
    }
    // here some operation with the file..
    fclose(f);
}
```

פתיחת קובץ – פרמטרים לסוג הפתיחה

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

פתיחת קובץ טקסט לקריאה: "r" □

פתיחת קובץ טקסט לכתיבה: □

- "w" - במידה ויש נתונים בקובץ הם ידרסו
- "a" – כותב לסוף קובץ (כלומר לא דורס את המידע, אם קיים)

פתיחת קובץ טקסט לקריאה ולכתיבה: □

- "+r" – הקובץ חייב להיות קיים, יתקבל NULL אם לא
- "+w" – אם הקובץ קיים, דורס אותו
- "+a" – אם הקובץ קיים כותב לסופו, אחרת יוצר קובץ חדש

פתיחת קובץ – סיבות לכשלון

□ קובץ המוגדר לקריאה בלבד שמנסים לפתוח אותו
לכתיבה

□ פתיחה ע"י "r" קובץ שאינו קיים

כתיבה לקובץ טקסט

□ פקודות הכתיבה לקובץ טקסט הן כמו הפקודות לכתיבה למסך, פרט לכך ש:

■ שמן מתחיל עם f

■ הן מקבלות פרמטר נוסף שהוא מצביע לקובץ אליו רוצים לכתוב

□ *int fprintf(FILE* , char*, ...);*

□ *int fputs(char*, FILE*);*

□ *int fputc(char, FILE*);*

סגירת קובץ

int fclose(FILE file);*

□ מקבלת מצביע לקובץ ומחזירה 0 אם הצליחה לסגור אותו

■ תכשל למשל כאשר מנסים לסגור קובץ שאינו פתוח

דוגמא 1: כתיבה לקובץ טקסט

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

```
void main()  
{
```

פתיחת הקובץ לכתיבה. אם קיים, ידרס

```
    FILE* f = fopen("myFile.txt", "w");
```

```
    if (f == NULL)
```

```
    {
```

```
        printf("Failed opening the file. Exiting!\n");
```

```
        return;
```

```
    }
```

```
    fputs("Hello World!\n", f);
```

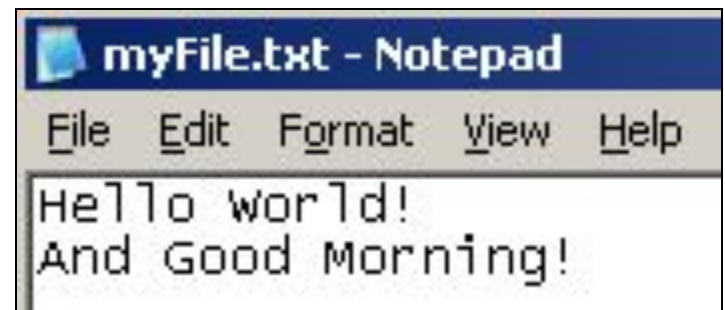
```
    fclose(f);
```

```
    f = fopen("myFile.txt", "a");
```

```
    fputs("And Good Morning!\n", f);
```

```
    fclose(f);
```

```
}
```



פתיחת הקובץ וכתיבה לסופו

דוגמא 2: כתיבה לקובץ טקסט

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

```
void main()
```

```
{
```

```
    FILE* f = fopen("myFile.txt", "w");
```

```
    if (f == NULL)
```

```
    {
```

```
        printf("Failed opening the file. Exiting!\n");
```

```
        return;
```

```
    }
```

```
    fputs("Hello World!\n", f);
```

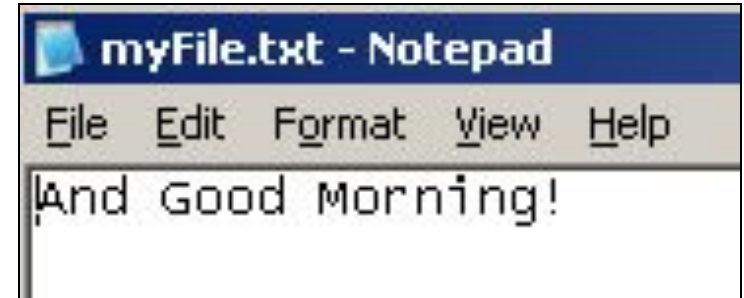
```
    fclose(f);
```

```
    f = fopen("myFile.txt", "w");
```

```
    fputs("And Good Morning!\n", f);
```

```
    fclose(f);
```

```
}
```



דריסת הקובץ הקודם...

שמירה וקריאה מקובץ - כללי

□ כאשר נשמור מידע לקובץ, אנו בוחרים לשמור את המידע בסדר מסוים

□ צריכה להיות התאמה בין סדר כתיבת הנתונים לקובץ לסדר קריאת הנתונים מהקובץ

■ למשל: אם כתבתי מידע על סטודנט בסדר הבא: שם, ת.ז. וגיל, גם סדר קריאת הנתונים יהיה זהה

□ כלומר, צריכה להיות הסכמה וידיעה בין מי שכותב לקובץ לבין מי שקורא ממנו לגבי סדר הנתונים בקובץ

פקודות קריאה מקובץ טקסט

□ פקודות הקריאה מקובץ טקסט הן כמו פקודות הקריאה למסך, פרט לכך ש:

- שמן מתחיל עם `f`
- הן מקבלות פרמטר נוסף שהוא מצביע לקובץ שממנו רוצים לקרוא

□ `int fscanf(FILE* , char*, ...);`

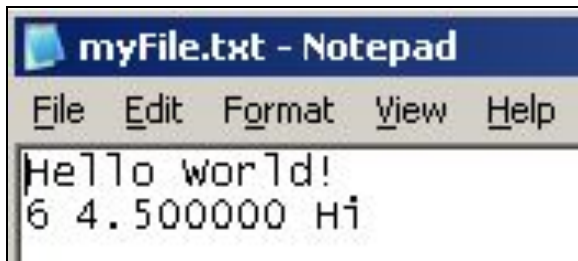
מקבלת גם את כמות התווים לקריאה. קוראת עד '\n' או עד n-1 תווים

□ `char* fgets(char*, int n, FILE*);`

מחזירה את המחרוזת שקראה או NULL אם לא הצליחה (למשל הקובץ נגמר)

□ `int fgetc(FILE*);`

מחזירה את ערך האסקי של התו או את הקבוע EOF (שערכו -1) אם לא הצליחה (למשל הקובץ נגמר)



דוגמא: קריאה מקובץ טקסט

```
#define SIZE 20
```

```
void main()
```

```
{
```

```
    char str[]="Hi", sentence[]="Hello World!", str2[SIZE], sentence2[SIZE];
```

```
    int num1=6, num2=0;
```

```
    float f1=4.5, f2=0;
```

```
    FILE* f = fopen("myFile.txt", "w");
```

```
    // check if open file succeeded..
```

```
    fputs(sentence, f);
```

```
    fprintf(f, "\n%d %f %s\n", num1, f1, str);
```

```
    fclose(f);
```

```
    printf("Before reading from file:\nnum2=%d f2=%f str=|%s| sentence2=|%s|\n",  
           num2, f2, str2, sentence2);
```

```
    f = fopen("myFile.txt", "r");
```

```
    // check if open file succeeded..
```

```
    fgets(sentence2, SIZE, f);
```

```
    fscanf(f, "%d %f %s", &num2, &f2, str2);
```

```
    printf("\nAfter reading from file:\nnum2=%d f2=%f str=|%s| sentence2=|%s|\n",  
           num2, f2, str2, sentence2);
```

```
    fclose(f);
```

```
}
```

```
C:\WINDOWS\system32\cmd.exe
Before reading from file:
num2=0 f2=0.000000 str=|Hello World!! sentence2=|Hello World!!
```

הפונקציה feof ודוגמא להעתקת קובץ

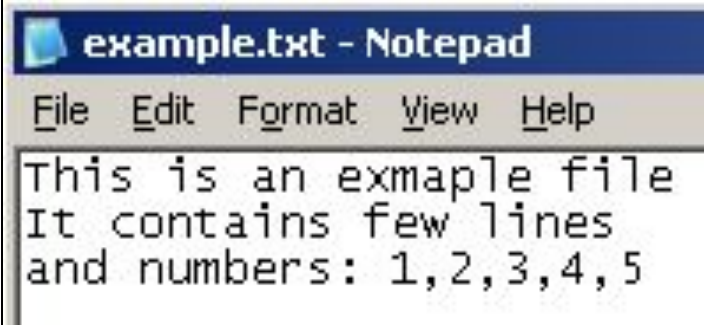
```
int feof (FILE * f);
```

פונקציה המקבלת קובץ פתוח ומחזירה 1 אם הגענו לסופו

```
void main()
{
    char ch;
    FILE* fSource, *fDest;
    fSource = fopen("example.txt", "r");
    // check if open file succeeded..

    fDest = fopen("exampleCopy.txt", "w");
    // check if open file succeeded..

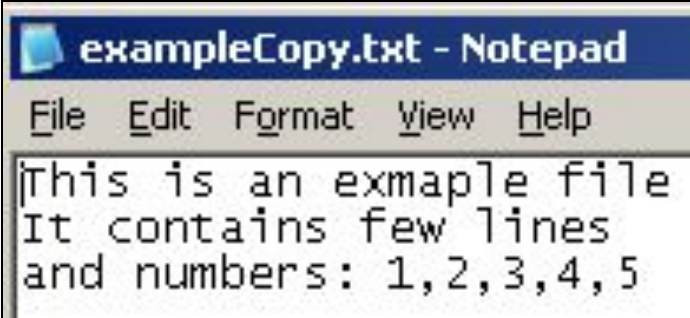
    ch = fgetc(fSource);
    while (!feof(fSource)) // OR: (ch != EOF)
    {
        fputc(ch, fDest);
        ch = fgetc(fSource);
    }
    fclose(fSource);
    fclose(fDest);
}
```



example.txt - Notepad

File Edit Format View Help

This is an exmaple file
It contains few lines
and numbers: 1,2,3,4,5



exampleCopy.txt - Notepad

File Edit Format View Help

This is an exmaple file
It contains few lines
and numbers: 1,2,3,4,5

דוגמא: קריאת מספרים מקובץ וחישוב הממוצע

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

```
void main()
```

```
{
```

```
    FILE* f;
```

```
    int num, sum=0, counter=0, rc;
```

```
    f = fopen("numbers.txt", "r");
```

```
    // check if open succeed...
```

```
    rc = fscanf(f, "%d", &num);
```

```
    while (rc != EOF) //OR: (!feof(f))
```

```
    {
```

```
        printf("The read numbers is %d\n", num);
```

```
        sum += num;
```

```
        counter++;
```

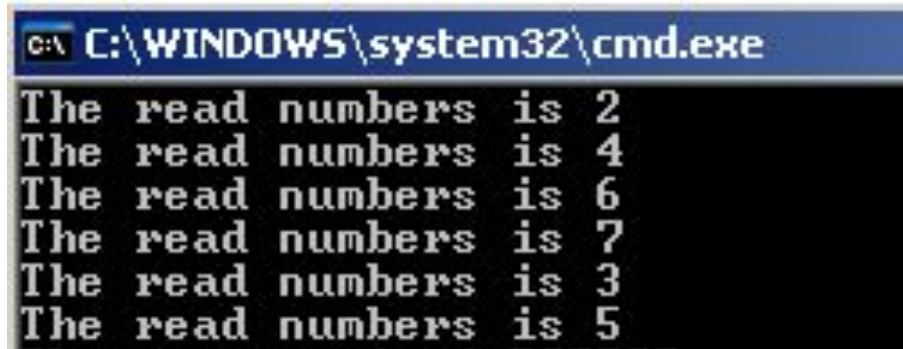
```
        rc = fscanf(f, "%d", &num);
```

```
    }
```

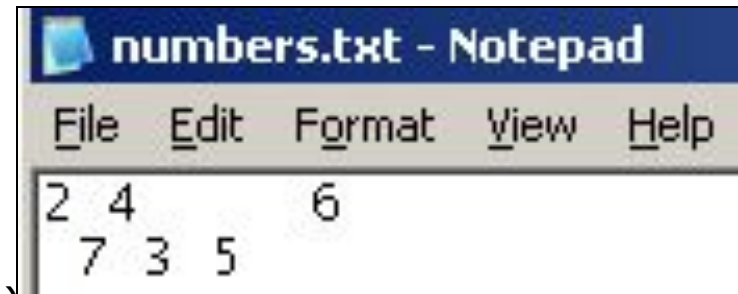
```
    fclose(f);
```

```
    printf("The average is %f\n", (float)sum/counter),
```

```
}
```



```
C:\WINDOWS\system32\cmd.exe
The read numbers is 2
The read numbers is 4
The read numbers is 6
The read numbers is 7
The read numbers is 3
The read numbers is 5
```



```
numbers.txt - Notepad
File Edit Format View Help
2 4 6
7 3 5
```

שמירת וטעינת מבנים לקובץ טקסט

□ כאשר כותבים מבנה לתוך קובץ טקסט, יש לכתוב
שדה-שדה

□ כאשר קוראים מבנה מקובץ טקסט, יש לקרוא שדה-
שדה

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

```
#define SIZE 20
```

```
struct Person
```

```
{
```

```
    char name[SIZE];
```

```
    long id;
```

```
    float age;
```

```
} typedef person_t;
```

```
void main()
```

```
{
```

```
    person_t p1={"momo", 1111, 23.5}, p2 = {"gogo", 2222, 24.8}, p3, p4;
```

```
    FILE* f = fopen("persons.txt", "w");
```

```
    fprintf(f, "%s %ld %.2f\n", p1.name, p1.id, p1.age);
```

```
    fprintf(f, "%s %ld %.2f\n", p2.name, p2.id, p2.age);
```

```
    fclose(f);
```

```
    f = fopen("persons.txt", "r");
```

```
    fscanf(f, "%s %ld %f\n", p3.name, &p3.id, &p3.age);
```

```
    fscanf(f, "%s %ld %f\n", p4.name, &p4.id, &p4.age);
```

```
    fclose(f);
```

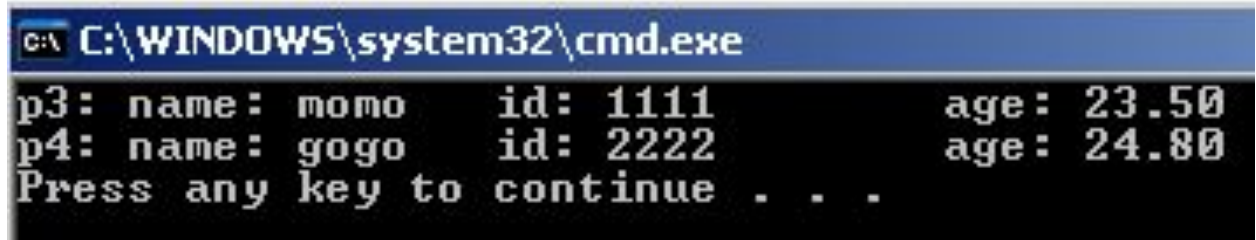
```
    printf("p3: name: %s\t id: %ld\t age: %.2f\n", p3.name, p3.id, p3.age);
```

```
    printf("p4: name: %s\t id: %ld\t age: %.2f\n", p4.name, p4.id, p4.age);
```

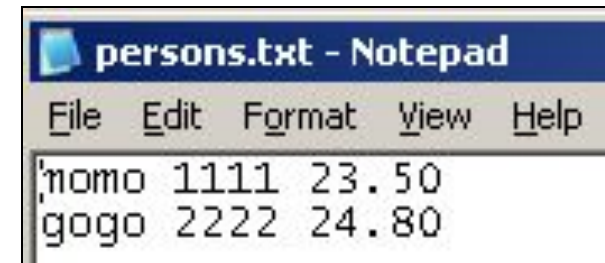
```
}
```

דוגמא: שמירת וטעינת מבנים

מקובץ טקסט



```
C:\WINDOWS\system32\cmd.exe
p3: name: momo    id: 1111    age: 23.50
p4: name: gogo    id: 2222    age: 24.80
Press any key to continue . . .
```



```
persons.txt - Notepad
File Edit Format View Help
momo 1111 23.50
gogo 2222 24.80
```

שמירה וטעינת מערך של מבנים מקובץ טקסט

□ כאשר כותבים מערך לקובץ, השדה הראשון צריך להיות מספר המבנים שאנו כותבים, כדי שקורא הקובץ ידע כמה מבנים יש בתוכו

דוגמא: שמירה וטעינה מערך מבנים (1)

```
struct Person
{
    char  name[SIZE];
    long  id;
    float age;
} typedef person_t;
```

```
void main()
```

```
{
    FILE* f;
    person_t* personsSource, *personsDest;
    int sizeSource, sizeDest, i;
    printf("How many persons? ");
    scanf("%d", &sizeSource);
```

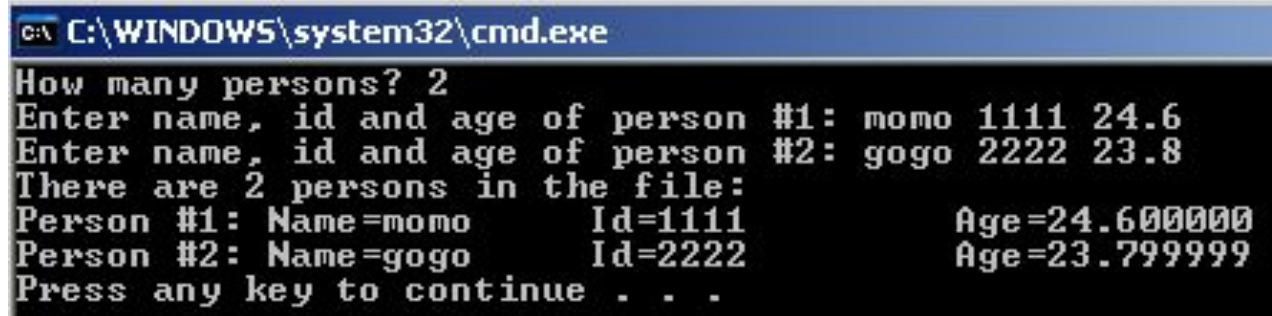
```
// allocating the persons array
```

```
personsSource = (person_t*)malloc(sizeSource*sizeof(person_t));
```

```
// reading persons..
```

```
for (i=0 ; i < sizeSource ; i++)
```

```
{
    printf("Enter name, id and age of person #%d: ", i+1);
    scanf("%s %ld %f", personsSource[i].name,
        &personsSource[i].id, &personsSource[i].age);
}
```



```
C:\WINDOWS\system32\cmd.exe
How many persons? 2
Enter name, id and age of person #1: momo 1111 24.6
Enter name, id and age of person #2: gogo 2222 23.8
There are 2 persons in the file:
Person #1: Name=momo      Id=1111      Age=24.600000
Person #2: Name=gogo      Id=2222      Age=23.799999
Press any key to continue . . .
```

דוגמא: שמירה וטעינה

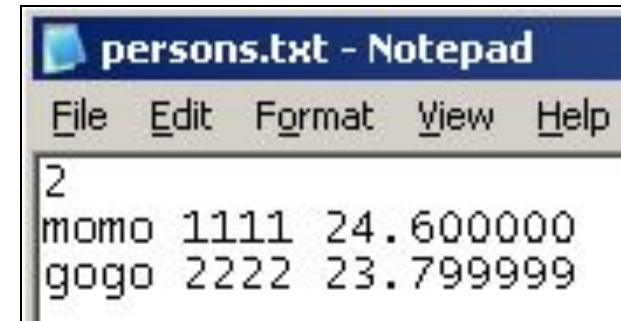
מערך מבנים (2)

```
f = fopen("persons.txt", "w");
// writing the size to the file
fprintf(f, "%d\n", sizeSource);
// writing each person to the file
for (i=0 ; i < sizeSource ; i++)
    fprintf(f, "%s %ld %f\n", personsSource[i].name,
            personsSource[i].id, personsSource[i].age);
fclose(f);
// don't forget to free the array!!
free(personsSource);

f = fopen("persons.txt", "r");
// reading the size from the file
fscanf(f, "%d\n", &sizeDest);
// allocating the new array
personsDest = (person_t*)malloc(sizeDest*sizeof(person_t));
// reading each person from the file
for (i=0 ; i < sizeDest ; i++)
    fscanf(f, "%s %ld %f\n", personsDest[i].name,
            &personsDest[i].id, &personsDest[i].age);
fclose(f);

printf("There are %d persons in the file:\n", sizeDest);
for (i=0 ; i < sizeDest ; i++)
    printf("Person # %d: Name=%s\t Id=%ld\t Age=%f\n", i+1,
           personsDest[i].name, personsDest[i].id, personsDest[i].age);

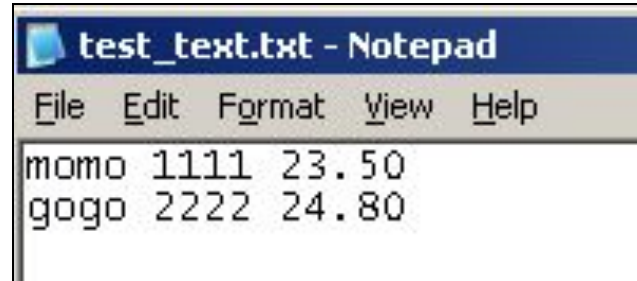
// don't forget to free the array!!
free(personsDest);
```



סוגי קבצים

□ קובץ טקסט:

- כתוב בשפה אותה אנו יכולים לקרוא ולהבין



```
test_text.txt - Notepad
File Edit Format View Help
momo 1111 23.50
gogo 2222 24.80
```

□ קובץ בינארי:

- לא ניתן להבין את התוכן



```
test_bin.txt - Notepad
File Edit Format View Help
momo w €7@ gogo 8-@
```

מצבי הפתיחה לקובץ בינארי

□ בדיוק כמ מצבי הפתיחה לקובץ טקסט, פרט לתוספת
התו **b** במחרוזת

□ למשל, פתיחת קובץ לקריאה בינארית:
`fopen("test.bin", "rb")`

כתיבה לקובץ בינארי

□ פקודת הכתיבה לקובץ בינארי:

□ `int fwrite(const void * ptr, // address of variable to write
int size, // size of type
int count, // number of elements
FILE * stream (; // pointer to the file`

■ מחזירה את כמות האיברים שכתבה

□ כאשר כותבים לקובץ בינארי ניתן לכתוב בלוק של מידע

■ למשל: מערך או מבנה בפעולת כתיבה בודדת

□ יש להיזהר מכתיבת כתובות לקובץ!

קריאה מקובץ בינארי

□ פקודת הקריאה מקובץ בינארי:

□ `int fread(void* ptr, // address of variable to read into
int size, // size of type
int count, // number of elements
FILE* stream (; // pointer to the file`

■ מחזירה את כמות האיברים שקראה

□ כאשר קוראים מקובץ בינארי ניתן לקרוא בלוק של מידע

■ למשל: מערך או מבנה בפעולת קריאה בודדת

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

```
#define SIZE 20
```

```
struct Person  
{  
    char name[SIZE];  
    long id;  
    float age;  
} typedef person_t;
```

```
void main()  
{
```

```
    person_t p1={"momo", 1111, 23.5}, p2 = {"gogo", 2222, 24.8}, p3, p4;  
    FILE* f = fopen("persons.bin", "wb");  
    fwrite(&p1, sizeof(person_t), 1, f);  
    fwrite(&p2, sizeof(person_t), 1, f);  
    fclose(f);
```

```
    f = fopen("persons.bin", "rb");  
    fread(&p3, sizeof(person_t), 1, f);  
    fread(&p4, sizeof(person_t), 1, f);  
    fclose(f);
```

```
    printf("p3: name: %s\t id: %ld\t age: %.2f\n", p3.name, p3.id, p3.age);  
    printf("p4: name: %s\t id: %ld\t age: %.2f\n", p4.name, p4.id, p4.age);
```

```
}
```

שמירת וטעינת מבנים

מקובץ בינארי - דוגמא



ניתן לקרוא ולכתוב רשומה בפעולה אחת

```
C:\WINDOWS\system32\cmd.exe  
p3: name: momo      id: 1111      age: 23.50  
p4: name: gogo      id: 2222      age: 24.80  
Press any key to continue . . .
```

שמירה וטעינה מערך מבנים

מקובץ בינארי (1)

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

```
#define SIZE 20
```

```
Struct Person
```

```
{  
    char name[SIZE];  
    long id;  
    float age;  
} typedef person_t;
```

```
void main()  
{
```

```
    FILE* f;  
    person_t* personsSource, *personsDest;  
    int sizeSource, sizeDest, i;  
    printf("How many persons? ");  
    scanf("%d", &sizeSource);
```

```
// allocating the persons array
```

```
personsSource = (person_t*)malloc(sizeSource*sizeof(person_t));
```

```
// reading persons..
```

```
for (i=0 ; i < sizeSource ; i++)
```

```
{  
    printf("Enter name, id and age of person #%d: ", i+1);  
    scanf("%s %ld %f", personsSource[i].name,  
        &personsSource[i].id, &personsSource[i].age);  
}
```

```
C:\WINDOWS\system32\cmd.exe  
How many persons? 2  
Enter name, id and age of person #1: momo 1111 24.6  
Enter name, id and age of person #2: gogo 2222 23.8  
There are 2 persons in the file:  
Person #1: Name=momo      Id=1111      Age=24.600000  
Person #2: Name=gogo      Id=2222      Age=23.799999  
Press any key to continue . . .
```

שמירה וטעינה מערך מבנים

מקובץ בינארי (2)

```
f = fopen("persons.bin", "wb");
// writing the size to the file
fwrite(&sizeSource, sizeof(int), 1, f);
// writing all persons to the file
fwrite(personsSource, sizeof(person_t), sizeSource, f);
fclose(f);
// don't forget to free the array!!
free(personsSource);
```



```
f = fopen("persons.bin", "rb");
// reading the size from the file
fread(&sizeDest, sizeof(int), 1, f);
// allocating the new array
personsDest = (person_t*)malloc(sizeDest*sizeof(person_t));
// reading all persons from the file
fread(personsDest, sizeof(person_t), sizeDest, f);
fclose(f);

printf("There are %d persons in the file:\n", sizeDest);
for (i=0 ; i < sizeDest ; i++)
    printf("Person #%d: Name=%s\t Id=%ld\t Age=%f\n", i+1,
           personsDest[i].name, personsDest[i].id, personsDest[i].age);

// don't forget to free the array!!
free(personsDest);
```

}

כתיבת מבנים המכילים כתובות

יש לשים לב לא לכתוב כתובות לתוך קובץ, שכן אין הבטחה שבפעם הבאה שהתוכנית תרוץ יהיו ערכים בכתובות הנוכחיות שאנו שומרים

לכן, כאשר יש משתנה שהוא כתובת יש להקפיד לרשום את ערכו לקובץ

■ **נשים לב:** לא נוכל להשתמש בשיטה של לרשום מבנה ע"י `fwrite`, אלא נהיה חייבים לכתוב את המבנה לקובץ שדה-שדה

■ בד"כ אם המצביע הוא למערך, נרשום כשדה נוסף את כמות האלמנטים במערך, ורק אח"כ את איבריו (כנ"ל עבור מחרוזת)

דוגמא – שמירה וטעינה של מבנה עם כתובות

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

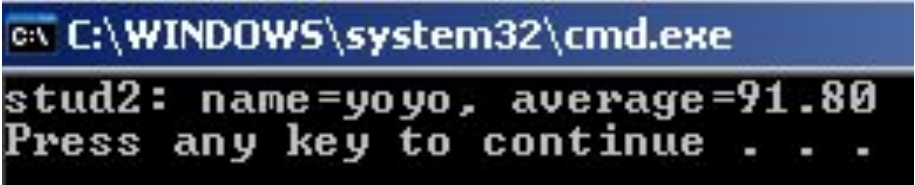
```
struct Student
{
    char* name;
    float average;
} typedef student_t;
```

```
void main()
{
    student_t stud1 = {"yoyo", 91.8}, stud2;
    int len1, len2;
    FILE* f = fopen("students.bin", "wb");
    // check open succeed...

    len1 = strlen(stud1.name)+1;
    fwrite(&len1, sizeof(int), 1, f);
    fwrite(stud1.name, sizeof(char), len1, f);
    fwrite(&stud1.average, sizeof(float), 1, f);
    fclose(f);

    f = fopen("students.bin", "rb");
    // check open succeed...

    fread(&len2, sizeof(int), 1, f);
    stud2.name = (char*)malloc(len2*sizeof(char));
    fread(stud2.name, sizeof(char), len2, f);
    fread(&stud2.average, sizeof(float), 1, f);
    fclose(f);
    printf("stud2: name=%s, average=%.2f\n", stud2.name, stud2.average);
    free(stud2.name);
}
```



```
C:\WINDOWS\system32\cmd.exe
stud2: name=yoyo, average=91.80
Press any key to continue . . .
```

פקודה fseek

□ מאפשרת לטייל בקובץ בלי לקרוא חלק משדותיו

```
int fseek(FILE* f,           // the file
          long  offset,      // how much
          int   origin (;    // from where
```

□ origin מקבל אחד מהערכים הבאים:

■ SEEK_SET – לזוז מתחילת הקובץ

■ SEEK_END – לזוז לסוף הקובץ

■ SEEK_CUR – לזוז מהמיקום הנוכחי

□ הפונקציה תחזיר 0 אם הצליחה לזוז כמתבקש

□ לידע כללי: יש מערכות הפעלה בהן fseek לא עובדת

טוב על קבצי טקסט

דוגמה - fseek

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

```
int main ()
```

```
{
```

```
    FILE * f;
```

```
    f = fopen("myfile.txt" , "w");
```

```
    // check if open succeed
```

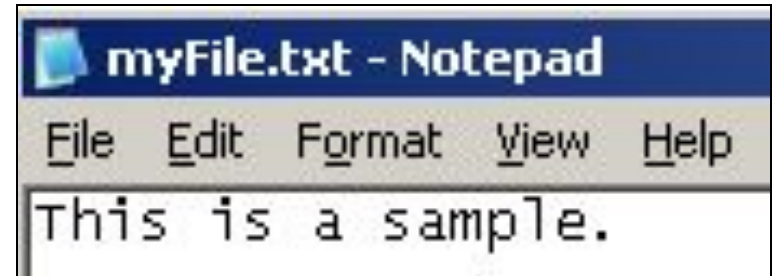
```
    fputs("This is an apple" , f);
```

```
    fseek (f , 9 , SEEK_SET);
```

```
    fputs(" sam" , f);
```

```
    fclose(f);
```

```
}
```



הפקודה ftell ודוגמאת מציאת גודל הקובץ

מחזירה את מרחק הסמן מתחילת הקובץ

```
long ftell (FILE * f);
```

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

```
int main ()  
{
```

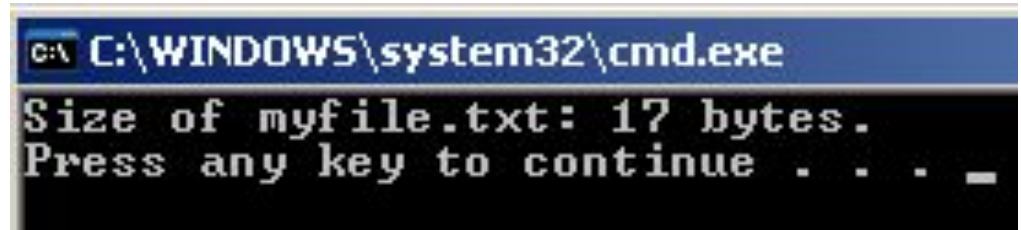
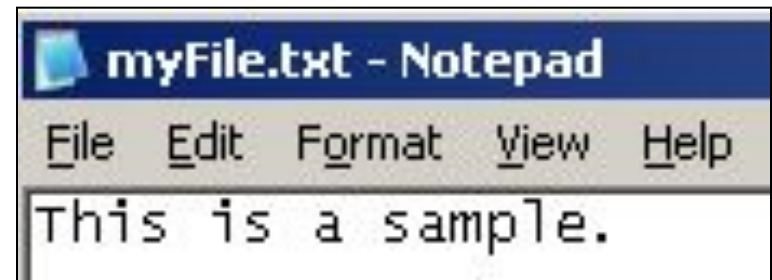
```
    FILE* f;  
    long size;
```

```
    f = fopen ("myFile.txt", "r");  
    // check if open succeed
```

```
    fseek (f, 0, SEEK_END);  
    size = ftell(f);  
    fclose (f);
```

```
    printf ("Size of myfile.txt: %ld bytes.\n",size);
```

```
}
```



הפקודה rewind ודוגמאת כתיבת וקריאת כל ה-ABC

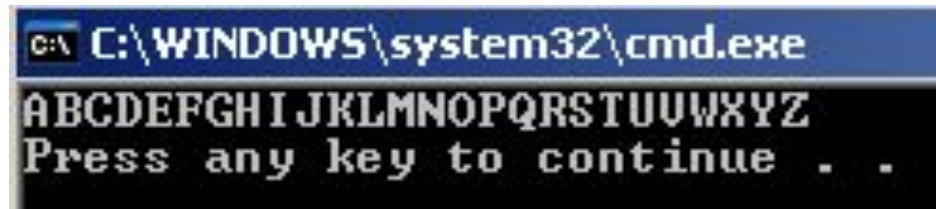
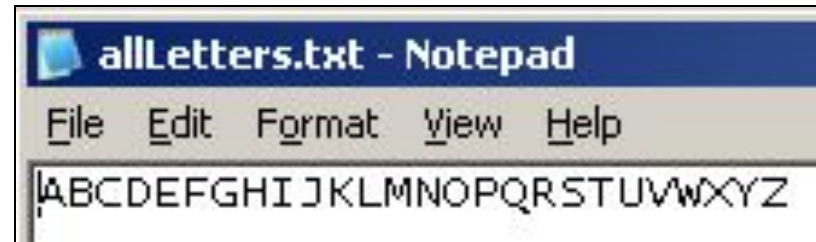
□ מחזירה את הסמן לתחילת הקובץ

```
void rewind(FILE* f);
```

```
int main ()
{
    int n;
    FILE *f;
    char buffer [27];

    f = fopen ("allLetters.txt","w+");
    // check if open succeed

    for ( n='A' ; n<='Z' ; n++)
        fputc ( n, f);
    rewind (f);
    fscanf(f, "%s", buffer);
    fclose (f);
    buffer[26]='\0';
    puts (buffer);
}
```



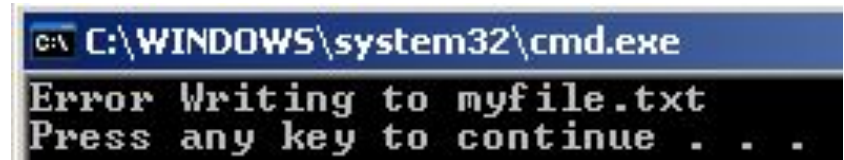
ferror

□ פקודה זו מחזירה 0 במידה והפעולות שבוצעו על הקובץ תקינות

■ דוגמא: כאשר ננסה לכתוב לקובץ שנפתח לקריאה בלבד

```
void main()
{
    FILE* f = fopen("myfile.txt","r");
    if (f == NULL)
    {
        printf("Error opening file\n");
        return;
    }

    fputc('x', f);
    if (ferror(f))
        printf ("Error Writing to myfile.txt\n");
    fclose(f);
}
```



ביחידה זו למדנו:

מוטיבציה לעבודה עם קבצים □

פעולות על קבצים: □

- פתיחת קובץ

- קריאה מקובץ

- כתיבה לקובץ

- סגירת קובץ

סוגי קבצים: □

- בינאריים

- טקסט

פעולות נוספות: fseek, ftell, rewind, ferror, □
feof

תרגיל 1:

- הגדר את המבנה `student_t` אשר נתוניו הם שם הסטודנט (מחרוזת בגודל מקסימלי 20), ת.ז. וממוצע ציונים
- כתוב פונקציה המקבלת מערך של סטודנטים, גודלו ושם קובץ, וכותבת את גודל המערך ואת נתוני הסטודנטים לקובץ טקסט
- יצר את הקובץ באמצעות הפונקציה שכתבת
- כתוב פונקציה המקבלת את שם הקובץ, ומחזירה את שם הסטודנט שהממוצע שלו הגבוה ביותר

תרגיל 2:

- הגדר את המבנה `student_t` אשר נתוניו הם שם הסטודנט (מחרוזת בגודל מקסימלי 20), ת.ז. וממוצע ציונים
- כתוב פונקציה המקבלת מערך של סטודנטים, גודלו ושם קובץ, וכותבת את גודל המערך ואת נתוני הסטודנטים לקובץ בינארי
- יצר את הקובץ באמצעות הפונקציה שכתבת
- כתוב פונקציה המקבלת את שם הקובץ, ומחזירה את שם הסטודנט שהממוצע שלו הגבוה ביותר

תרגיל 3:

□ הגדר את המבנה student_t אשר נתוניו הם שם הסטודנט (אורך המחרוזת אינו ידוע), ת.ז. וממוצע ציונים

□ כתוב פונקציה המקבלת מערך של סטודנטים, גודלו ושם קובץ, וכותבת את גודל המערך ואת נתוני הסטודנטים לקובץ טקסט

□ יצר את הקובץ באמצעות הפונקציה שכתבת

□ כתוב פונקציה המקבלת את שם הקובץ, ומחזירה את שם הסטודנט שהממוצע שלו הגבוה ביותר

תרגיל 4:

□ הגדר את המבנה student_t אשר נתוניו הם שם הסטודנט (אורך המחרוזת אינו ידוע), ת.ז. וממוצע ציונים

□ כתוב פונקציה המקבלת מערך של סטודנטים, גודלו ושם קובץ, וכותבת את גודל המערך ואת נתוני הסטודנטים לקובץ בינארי

□ יצר את הקובץ באמצעות הפונקציה שכתבת

□ כתוב פונקציה המקבלת את שם הקובץ, ומחזירה את שם הסטודנט שהממוצע שלו הגבוה ביותר