סדנה מתקדמת בתכנות

פרק 7: קבצים

למה קבצים?

- רוב תכניות משתמשות בנתונים מסוגים שונים שנשמרים בזיכרון פנימי של מחשב, ואז זיכרון זמני!!
- אנו צריכים כלי או שיטה לשמור נתונים לטווח ארוך!
- לשם כך, קיימים רכיבי זיכרון קבועים, כגון דיסק קשיח או disk on key שמסוגלים לשמור מידע לאורך זמן, ולהביאו לפי דרישה.
- המידע נשמר על התקנים אלו בקבצים, שמזוהים על ידי שמם ועל ידי התיקייה בה הם נמצאים.

למה קבצים?

לשמירת נתונים

- לטווח ארוך
- ללא הגבלה וללא הגדרת גודל מראש

קבצים ב-C

- שפתC מאפשרת לקרוא ולכתוב קבצים. כדי לקרוא או לכתוב קובץ, אנו יוצרים **מצביע** לטיפוס FILE ואז משתמשים במגוון פונקציות כדי לבצע את הפעולה הרצויה.

- FILE הוא למעשהstruct המוגדר ב- stdio.h שמיועד לעבודה עם קבצים.

- יש שני סוגי קבצים לאחסון נתונים:
קובץ טקסט או קובץ סדרתי (text file or sequential-access file)
וקובץ בינארי (binary file or random-access file).

אלגוריתם לשימוש:

- להצביע לקובץ רצוי
  - לציין סוג גישה (כתיבה, קריא, עדכון)
- לקלוט/לפלוט נתונים לקובץ
- לסגור קשר עם הקובץ

ממשק: פתיחה וסגירה

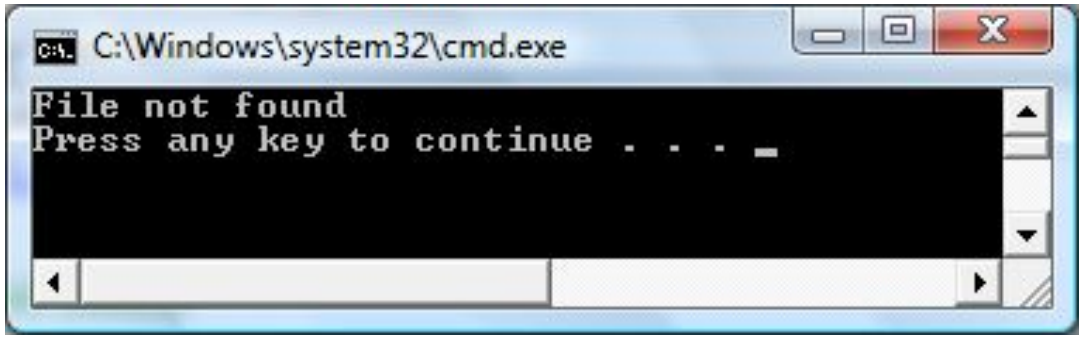
- תחביר
- מצביע לקובץ = fopen (שם קובץ, סוג פעולה)
  - שם לוגי של הקובץ
  - מחרוזת: שם פיסי של הקובץ
  - מחרוזת
- סוגי פעולה
  - **"w" - write: כתיבה**
  - **"r" - read: קריאה**
  - **"a" - append: הוספה לסוף קובץ קיים**
- fopen מחזירה ערך מסוג FILE * (מצביע ל- FILE) במידה והפתיחה הצליחה. אחרת מחזירה NULL.
- אחרי שסיימנו לעבוד עם הקובץ
- fclose(מצביע לקובץ)

חשוב לזכור

1) במידה ונפתח **לכתיבה או הוספה** קובץ שלא קיים, ייווצר קובץ בשם זה.
2) במידה ונפתח **לכתיבה** קובץ שקיים, הכתיבה לקובץ זה תמחק את הקיים (overwrite).
3) במידה ונפתח **לקריאה** קובץ שלא קיים או ללא הרשאה מתאימה, fopen תחזיר NULL.

דוגמה לפתיחת קובץ לקריאה

- ניסוי לפתיחת קובץ שלא קיים:

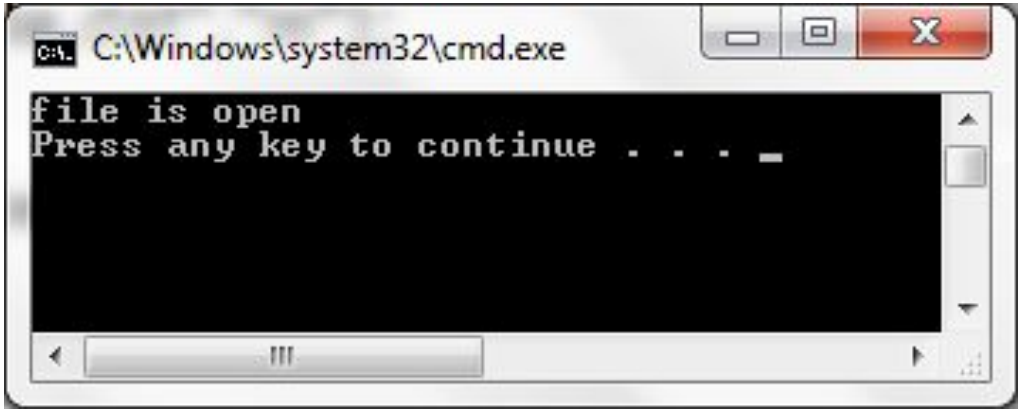



```
#include <stdio.h>


void main()
{
    FILE *fin;
    fin=fopen("testFile.dat","r");
    if( fin==NULL)
        { printf("File not found\n"); return; }
    fclose(fin);
}
```

קבצים לכתיבה

- פתיחת קבצים לכתיבה בד"כ מצליחה:

```
#include <stdio.h>

void main()
{
    FILE *fout;
    fout=fopen("textFile.dat","w");
    if(fout != NULL)
        { printf("file is open\n"); fclose(fout); }
}
```

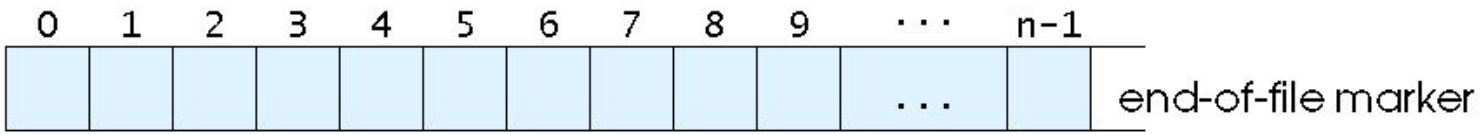


פתיחת קובץ טקסט

Mode	Description
r	Open a file for reading.
w	Create a file for writing. If the file already exists, discard the current contents.
a	Append: open or create a file for writing at end of file.
r +	Open a file for update (reading and writing).
w+	Create a file for update. If the file already exists, discard the current contents.
a +	Append: open or create a file for update; writing is done at the end of the file.

קבצים וזרמים

- שפת C רואה כל קובץ כזרם (stream) סדרתי של בתים:



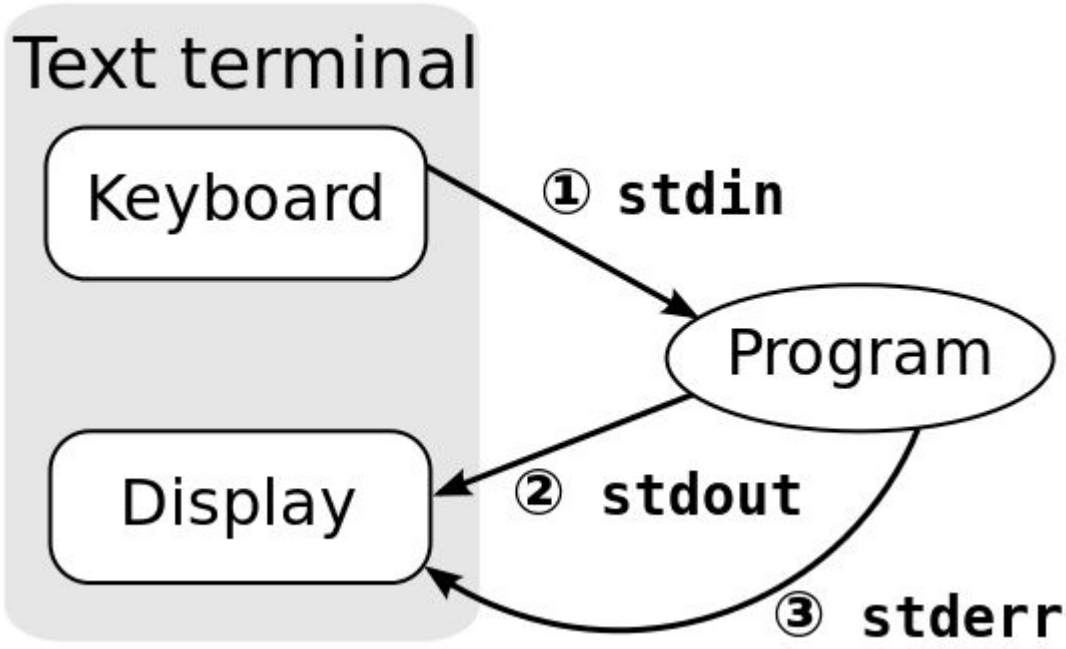
- כאשר קובץ נפתח, הזרם מזוהה עם הקובץ ומספק ערוץ בין הקובץ לתכנית

- 3 קבצים והזרמים הקשורים בהם נפתחים אוטומטית בעת תחילת ביצוע התכנית:

  standard input, standard output, standard error

קבצים וזרמים סטנדרטיים

- stdin – מצביע למקלדת
- stdout, stderr – מצביעים למסך



פונקציות קלט/פלט לקבצי טקסט (stdio.h)

fgetc (מצביע לקובץ)

- קוראת ומחזירה תו אחד מהקובץ
- fgetc(stdin) זהה ל- getchar()

fputc(תו, מצביע לקובץ)

- כותבת תו אחד לקובץ ומחזירה את התו
- fputc(ch, stdout) זהה ל- putchar(ch)

פונקציות קלט/פלט לקבצי טקסט

fprintf(מצביע לקובץ, תבנית, פרמטרים)

- זהה ל- printf פרט לעובדה שכותבים לקובץ שאינו המסך.
- fprintf(stdout, "%d", x) זהה ל- printf("%d", x)
- מחזירה מספר בתים כתובים.

fscanf(מצביע לקובץ, תבנית, פרמטרים)

- זהה ל- scanf פרט לעובדה שקוראים מקובץ שאינו המקלדת.
- fscanf(stdin, "%d", &x) זהה ל- scanf ("%d", &x)
- מחזירה מספר פרמטרים נקלטים.

פונקציות קלט/פלט לקבצי טקסט

fputs(שם **מחרוזת**, מציע ל**קובץ**)

- לכתיבת **מחרוזות** ל**קובץ**
- לעומת puts, לא מוסיפה '\n' בסוף ה**מחרוזת**
- מחזירה מספר אי-שלילי

fgets(שם **מחרוזת**, **מספר** שלם, מציע ל**קובץ**)

- לקריאת **מחרוזות** מ**קובץ**
- קולטת תווים מה**קובץ**
  - עד (וכולל) '\n' הראשון
  - או עד שיוקלטו **מספר** תווים (כולל '\0')
  - או עד סוף ה**קובץ**

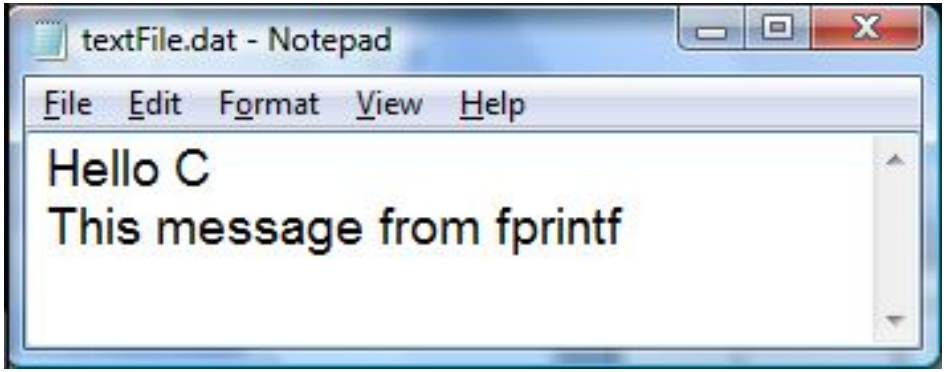
פונקציות קלט/פלט לקבצי טקסט

fgets(שם **מחרוזת**, **מספר** שלם, מצביע ל**קובץ**)

- מוסיפה '\0' בסוף ה**מחרוזת**
- לעומת gets, לא מחליפה '\n' ע"י '\0' אלא מוסיפה '\0' אחרי '\n'
- מחזירה **מחרוזת** אם הקלט מוצלח או NULL אם הקלט לא מוצלח

קלט/פלט

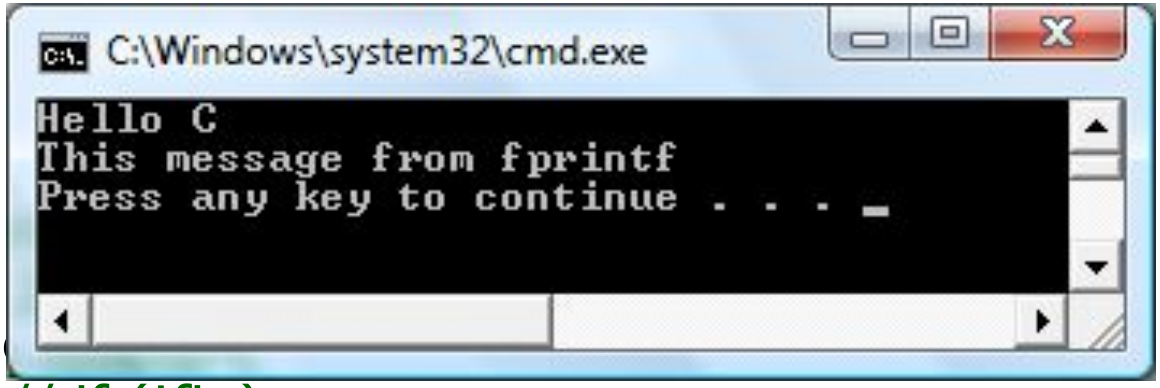
```
#include <stdio.h>
#include <stdlib.h>
void main()
{
    FILE *fout;
    fout=fopen("textFile.dat","w");
    if (fout==NULL)
        exit(1); //unsuccessful  termination
    fputs("Hello C\n",fout);
    fprintf (fout, "This message from fprintf\n");
    fclose(fout);
}
```



exit(0) - successful termination

קלט/פלט

```
void main()
{
    FILE *fin;
    char st[60];
    fin=fopen("textFil
    if( fin==NULL)     // if (!fin)
    {
        printf("File not found\n");
        return;
    }
    while( fgets(st, 60, fin) )
        printf("%s",st);
    fclose(fin);
}
```



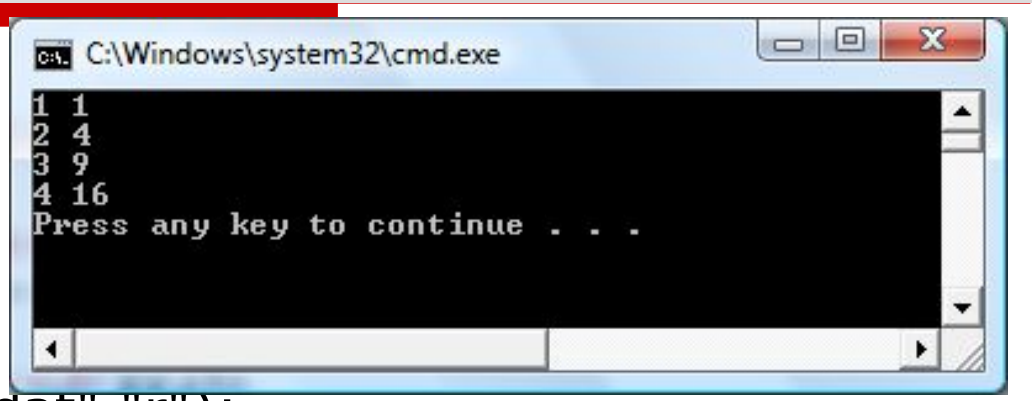
המיקום הנוכחי בקובץ (**file pointer position**)
מתקדם אוטומטית אחרי כל פעולת קלט/פלט

קלט/פלט מספרי

- נשמור בקובץ ריבועים של מספרים טבעיים:

```
void main()
{
    FILE *fout;
    int i;
    fout=fopen("textFile.dat","w");
    if( !fout )
        { printf("file is not open\n"); return; }
    for(i=1; i<5; i++)
        fprintf(fout,"%d %d\n",i,i*i);
    fclose(fout);
}
```

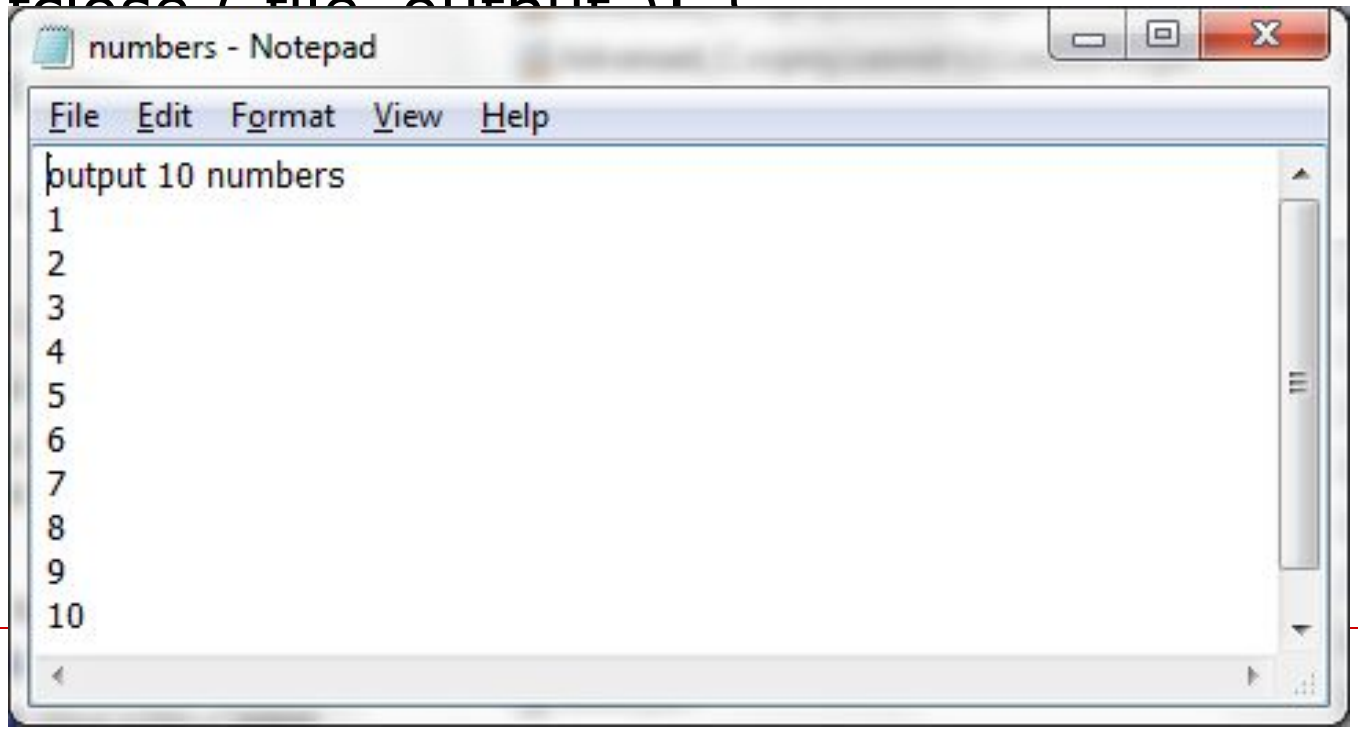
קלט/פלט מספרי




```
void main()
{
    FILE *fin;
    int i,a,b;
    fin=fopen("textFile.dat","r");
    if( !fin )
        { printf("file is not open\n"); return; }
    for(i=1; i<5; i++)
    {
        fscanf(fin,"%d%d",&a,&b);
        printf("%d %d\n",a,b);
    }
    fclose(fin);
}
```

פלט טקסט ומספרים

```
FILE *file_output = fopen("numbers.dat","w");
if (file_output) {
    fputs("output 10 numbers \n", file_output);
    for( i = 1; i <= 10; i++)
        fprintf ( file_output,  "%d\n",  i );
fclose ( file_output ); }
```



תו סיום הקובץ EOF

- תו מיוחד, EOF, מסמן לפי מוסכמה את סיום תוכן הקובץ.
- ערך **EOF** שווה בדרך כלל ל- **1-**

```
while ((c = fgetc(f)) != EOF)
{
   ...
}
```

תו סיום הקובץ EOF

- איך לקלוט סיום הקובץ stdin?

```
void main()
{
    int c;
    while ((c=getchar())!=EOF)
        putchar (c);
}
```

Computer system	Key combination
UNIX systems	*<return> <ctrl> d*
IBM PC and compatibles	**<ctrl> z**
Macintosh	*<ctrl> d*
End-of-file key combinations for various popular computer systems.	

סיום קובץ

feof(מצביע ל**קובץ**)

- פונקציה בוליאנית אשר מחזירה אמת בסיום ה**קובץ**

```
fscanf(f,"%d%d",&a,&b);
while( ! feof(f) )
{
    printf("%d %d\n",a,b);
    fscanf(f,"%d%d",&a,&b);
}
```

צריך לבצע קלט לפני בדיקת סוף הקובץ!!!

פלט לקובץ סדרתי

```
/* Create a sequential file */
#include <stdio.h>
void main()
{
   int account;     /* account number */
   char name[ 30 ]; /* account name */
   double balance;  /* account balance */
   FILE *cfPtr;     /* cfPtr = clients.dat file pointer */
   if ( ( cfPtr = fopen( "clients.dat", "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%lf", &account, name, &balance );
```

פלט לקובץ סדרתי

```
      /* write account, name and balance into file with fprintf */
      while ( !feof( stdin ) ) {
         fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
         printf( "? " );
         scanf( "%d%s%lf", &account, name, &balance );
      } /* end while */
       fclose( cfPtr ); /* fclose closes file */
   } /* end else */
} /* end main */
```

```
Enter the account, name, and balance.
Enter EOF to end input.
? 100 Jones 24.98
? 200 Doe 345.67
? 300 White 0.00
? 400 Stone -42.16
? 500 Rich 224.62
? ^Z
```

קלט מקובץ סדרתי

```
/* Reading and printing a sequential file */
#include <stdio.h>
void main()
{
   int account;     /* account number */
   char name[ 30 ]; /* account name */
   double balance;  /* account balance */
   FILE *cfPtr;     /* cfPtr = clients.dat file pointer */
   if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL )
      printf( "File could not be opened\n" );
```

קלט מקובץ סדרתי

```
else { /* read account, name and balance from file */
      printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
      fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
      /* while not end of file */
      while ( !feof( cfPtr ) ) {
         printf( "%-10d%-13s%7.2f\n", account, name, balance );
         fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
      } /* end while */
      fclose( cfPtr ); /* fclose closes the file */
   } /* end else */
} /* end main */
```

```
Account    Name          Balance
100        Jones           24.98
200        Doe            345.67
300        White            0.00
400        Stone          -42.16
500        Rich           224.62
```

העברת קובץ לפונקציה

```
#include <stdio.h>

void func (char *filename)
{
    FILE *f = fopen(filename,"r");
    …
    fclose(f);
}

void main() {
    …
    func("input.txt");
    …
}
```

העברת קובץ לפונקציית עזר לביצוע פעולות לוקליות

```
void func_auxiliary (FILE *f)
{
        …
}

void func (char *filename)
{
    FILE *f = fopen(filename,"r");
    …
    func_auxiliary (f);
    …
    fclose(f);
}
```

פונקציות שינוי ומציאת המיקום בקובץ

- rewind(מצביע ל**קובץ**)

  מזיזה את המיקום הנוכחי ב**קובץ** (file position pointer) לתחילתו.

- ftell(מצביע ל**קובץ**)

  מחזירה את המיקום הנוכחי ב**קובץ**.

- fseek(מצביע ל**קובץ**, offset, origin)

  מזיזה את המיקום הנוכחי ב**קובץ**:

  - offset הוא מספר הבתים שיש לזוז (יכול להיות גם שלילי).
  - origin הוא מוצא התזוזה (אחד משלושה):
    - SEEK_CUR - המיקום הנוכחי
    - SEEK_SET - תחילת ה**קובץ**
    - SEEK_END - סוף ה**קובץ**

מחזירה 0 אם הצליחה, אחרת מחזירה מספר שונה מאפס.

מחיקת קובץ

- int remove(const char *filename)

מחיקה את הקובץ.

מחזירה 0 אם הצליחה, אחרת מחזירה -1.

```
#include <stdio.h>
void main()  {
   int status;
   char file_name[25];
   printf("Enter the name of file you wish to delete\n");
   gets(file_name);
   status = remove(file_name);
   if( status == 0 )
        printf("%s file deleted successfully.\n",file_name);
   else  printf("Unable to delete the file\n");
}
```

שינוי שם הקובץ

- int rename(const char *oldname,
  const char *newname)

משנה שם הקובץ.

מחזירה 0 אם הצליחה, אחרת מחזירה מספר שונה מאפס.

```
#include<stdio.h>
void main()  {
    char  buffer_old[101], buffer_new[101];
    printf("Current filename: "); gets(buffer_old);
    printf("New filename: "); gets(buffer_new);
    if(rename(buffer_old, buffer_new) == 0)
        printf("%s has been renamed to %s.\n",
            buffer_old, buffer_new);
    else  fprintf(stderr, "Error renaming %s.\n", buffer_old);
}
```

איך למחוק חלק של תוכן הקובץ?

- לסרוק את הקובץ ולהעתיק לקובץ חדש את חלק התוכן שלא אמור להימחק.
- למחוק את הקובץ המקורי.
- לשנות שם של הקובץ החדש לשם של הקובץ המקורי.

המלצות

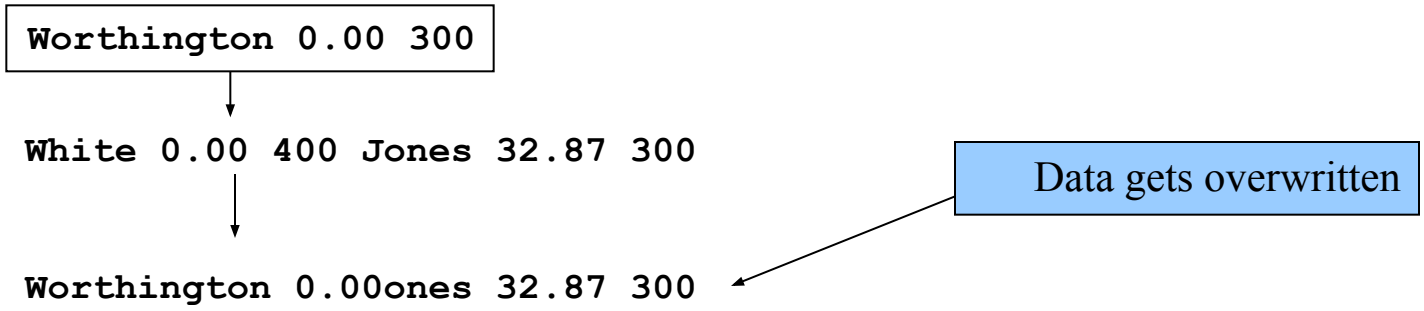
- לא לפתוח אותו קובץ עם שני מצביעים במקביל.
- לא כדאי להכניס EOF לאמצע קובץ באמצעות פונקציית פלט לקובץ.
- אם הקובץ נפתח לעדכון (כתיבה וקריאה) להשתמש ב-fseek בין פעולת כתיבה לפעולת קריאה.

חסרונות קובץ טקסט (סדרתי)

- לא ניתן לשנות נתונים ללא הסיכון של הרס נתונים אחרים.
- שדות יכולים להיות שונים בגודלם
  - ייצוג בקבצים ובמסך שונה מייצוג פנימי
  - 1, 35, -849 הם מספרים בעלי אותו טיפוס (int) אבל גדלים שלהם בדיסק הם שונים

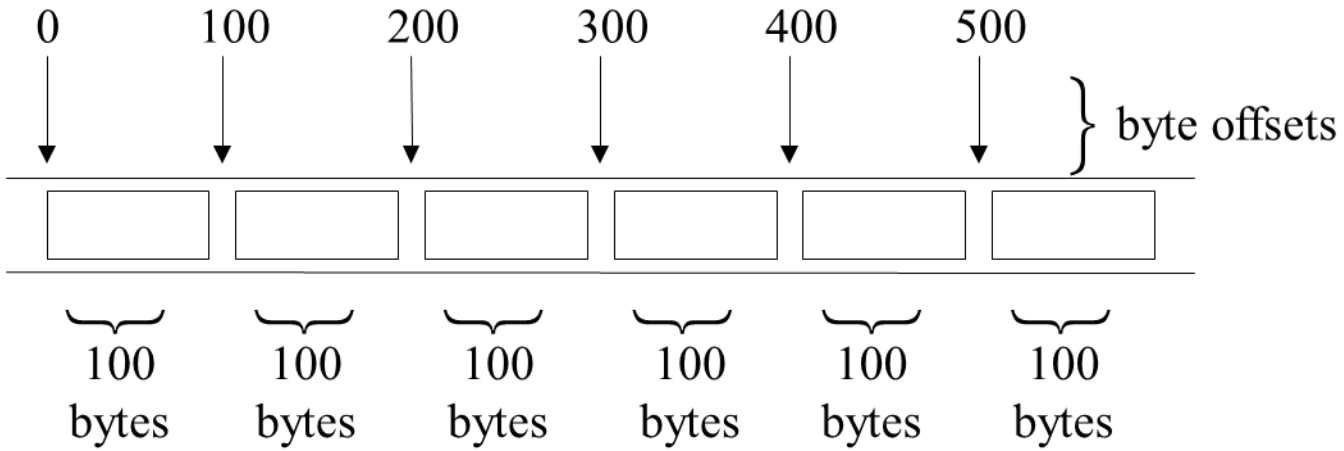
White 0.00 400 Jones 32.87 300 (נתונים ישנים בקובץ)

רוצים לשנות White ל- Worthington



קבצים בינאריים (Random-Access)

- רשומות (records) הן בעלות גודל קבוע.
- נתונים יכולים להיות מוכנסים מבלי להרוס נתונים אחרים.
- יש אפשרות לגישה ישירה לרשומות ללא חיפוש דרך רשומות אחרות.
- ניתן לעדכן ולמחוק נתונים בלי יצירת כל הקובץ מחדש.



קבצים בינאריים (Random-Access)

- רשומות של אותו טיפוס משתמשות באותה כמות הזיכרון.
- הנתונים הם לא מפורמטים (מאוחסנים כשורות הבתים).
- הנתונים לא נתונים לקריאה ע"י בן-אדם.

פתיחת קובץ בינארי

Mode	Description
r b	Open a file for reading in binary mode.
w b	Create a file for writing in binary mode. If the file already exists, discard the current contents.
a b	Append: open or create a file for writing at end of file in binary mode.
r b+	Open a file for update (reading and writing) in binary mode.
w b+	Create a file for update in binary mode. If the file already exists, discard the current contents.
a b+	Append: open or create a file for update in binary mode; writing is done at the end of the file.

פונקציות קלט/פלט לקבצים בינאריים (stdio.h)

size_t **fwrite**(const void ***buffer**, size_t **size**,
size_t **count**, FILE ***stream**)

- כותבת נתונים לקובץ

- **buffer** – מצביע למערך הנתונים להיכתב
- **size** – גודל בבתים של כל איבר להיכתב
- **count** – מספר האיברים להיכתב, כל אחד בגודל **size**
- **stream** – מצביע לקובץ

- מחזירה את מספר האיברים שנכתבו בהצלחה

פונקציות קלט/פלט לקבצים בינאריים (stdio.h)

size_t **fread**(void ***buffer**, size_t **size**,
size_t **count**, FILE ***stream**)

- קוראת נתונים מהקובץ

- **buffer** – מצביע לבלוק הזיכרון
- **size** – גודל בבתים של כל איבר לקריאה
- **count** – מספר האיברים לקריאה, כל אחד בגודל **size**
- **stream** – מצביע לקובץ

- מחזירה את מספר האיברים שנקראו בהצלחה

קלט/פלט לקבצים בינאריים

```
#define SIZE 5
void main()
{
	FILE *stream;
	int a[SIZE]={1, 2, 3, 4, 5}, b[SIZE], i;
	if ( ( stream = fopen("bin_file.dat","wb") ) != NULL )
	{
		fwrite(a, sizeof(int), SIZE, stream);
		fclose (stream);
	}
	else
		printf ("File could not be opened\n");
	if ( ( stream = fopen("bin_file.dat","rb") ) != NULL )
	{
		fread(b, sizeof(int), SIZE, stream);
		for (i=0; i<SIZE; i++)
			printf ("%d ", b[i]);
		printf ("\n");
		fclose (stream);
	}
	else
		printf ("File could not be opened\n");
}
```

פלט:

1 2 3 4 5

יצירת קובץ בינארי

```
/* Creating a random-access file sequentially */
#include <stdio.h>

/* clientData structure definition */
struct clientData {
   int acctNum;          /* account number */
   char lastName[ 15 ];  /* account last name */
   char firstName[ 10 ]; /* account first name */
   double balance;       /* account balance */
}; /* end structure clientData */
```

יצירת קובץ בינארי

```
void main()
{
   int i; /* counter used to count from 1-100 */
   /* create clientData with default information */
   struct clientData blankClient = { 0, "", "", 0.0 };
   FILE *cfPtr; /* credit.dat file pointer */
   if ( ( cfPtr = fopen( "credit.dat", "wb" ) ) == NULL )
      printf( "File could not be opened.\n" );
   else {
      /* output 100 blank records to file */
      for ( i = 1; i <= 100; i++ )
         fwrite( &blankClient, sizeof( struct clientData ), 1, cfPtr );
      fclose ( cfPtr ); /* fclose closes the file */
   } /* end else */
} /* end main */
```

פלט לקובץ בינארי

```
/* Writing to a random-access file */
#include <stdio.h>

/* clientData structure definition */
struct clientData {
   int acctNum;          /* account number */
   char lastName[ 15 ];  /* account last name */
   char firstName[ 10 ]; /* account first name */
   double balance;       /* account balance */
}; /* end structure clientData */
```

פלט לקובץ בינארי

```
void main()
{
   FILE *cfPtr; /* credit.dat file pointer */
   /* create clientData with default information */
   struct clientData client = { 0, "", "", 0.0 };
   if ( ( cfPtr = fopen( "credit.dat", "rb+" ) ) == NULL )
      printf( "File could not be opened.\n" );
   else {
      /* require user to specify account number */
      printf( "Enter account number ( 1 to 100, other to end input )\n? " );
      scanf( "%d", &client.acctNum );
```

פלט לקובץ בינארי

```
   /* user enters information, which is copied into file */
   while ( client.acctNum > 0 && client.acctNum <= 100 ) {
      /* user enters last name, first name and balance */
      printf( "Enter lastname, firstname, balance\n? " );
      scanf( "%s%s%lf", client.lastName, client.firstName, &client.balance );
      /* seek position in file to user-specified record */
      fseek( cfPtr, ( client.acctNum - 1 ) * sizeof( struct clientData ), SEEK_SET );
      /* write user-specified information in file */
      fwrite( &client, sizeof( struct clientData ), 1, cfPtr );
      /* enable user to input another account number */
      printf( "Enter account number\n? " );
      scanf( "%d", &client.acctNum );
   } /* end while */
   fclose( cfPtr ); /* fclose closes the file */
} /* end else */
} /* end main */
```

פלט לקובץ בינארי

```
Enter account number ( 1 to 100, 0 to end input )
? 37
Enter lastname, firstname, balance
? Barker Doug 0.00
Enter account number
? 29
Enter lastname, firstname, balance
? Brown Nancy -24.54
Enter account number
? 96
Enter lastname, firstname, balance
? Stone Sam 34.98
Enter account number
? 88
Enter lastname, firstname, balance
? Smith Dave 258.34
Enter account number
? 33
Enter lastname, firstname, balance
? Dunn Stacey 314.33
Enter account number
? 0
```

קלט מקובץ בינארי

```
/* Reading a random-access file sequentially */
#include <stdio.h>

/* clientData structure definition */
struct clientData {
   int acctNum;          /* account number */
   char lastName[ 15 ];  /* account last name */
   char firstName[ 10 ]; /* account first name */
   double balance;       /* account balance */
}; /* end structure clientData */
```

קלט מקובץ בינארי

```
void main()
{
   FILE *cfPtr; /* credit.dat file pointer */
   /* create clientData with default information */
   struct clientData client = { 0, "", "", 0.0 };
   if ( ( cfPtr = fopen( "credit.dat", "rb" ) ) == NULL )
      printf( "File could not be opened.\n" );
   else {
      printf( "%-6s%-16s%-11s%10s\n", "Acct", "Last Name",
         "First Name", "Balance" );
```

קלט מקובץ בינארי

```
      /* read all records from file (until eof) */
      fread( &client, sizeof( struct clientData ), 1, cfPtr );
      while ( !feof( cfPtr ) ) {
         /* display record */
         if ( client.acctNum != 0 )
            printf( "%-6d%-16s%-11s%10.2f\n",
               client.acctNum, client.lastName,
               client.firstName, client.balance );
         fread( &client, sizeof( struct clientData ), 1, cfPtr );
      } /* end while */
      fclose( cfPtr ); /* fclose closes the file */
   } /* end else */
} /* end main */
```

```
Acct  Last Name       First Name    Balance
29    Brown           Nancy          -24.54
33    Dunn            Stacey         314.33
37    Barker          Doug             0.00
88    Smith           Dave           258.34
96    Stone           Sam             34.98
```