עבודה עם קבצים בשפת C

קריאה עצמית

עבודה עם קבצים

- קבצים מיוצגים ב-C באמצעות הטיפוס **`FILE`**.
- טיפוס זה איננו טיפוס בסיסי, אלא **`typedef`** לטיפוס מורכב יותר. הוא מוגדר בקובץ **`stdio.h`**.
- הייצוג הפנימי של טיפוס זה לא יהיה מעניינו, כיוון שכל הטיפול בו ייעשה באמצעות פונקציות הפועלות עליו.
- הטיפוס **`FILE`** עשוי לתפוש זיכרון רב, ולכן העבודה עם קבצים נעשית תמיד באמצעות מצביעים לקבצים, מטיפוס **`FILE*`**.

ערוצי הקלט/פלט הסטנדרטיים

- שפת C מגדירה שלושה ערוצי קלט/פלט סטנדרטיים: **`stdin`**, **`stdout`**, ו-**`stderr`**. **`stderr`** הוא ערוץ השגיאה הסטנדרטי, ונכיר אותו יותר בהמשך.
- שלושת ערוצי הקלט/פלט הסטנדרטיים זמינים לשימוש לכל תוכנית C ללא צורך בכל פעולה נוספת (בניגוד לקבצים, כפי שנראה בהמשך).
- לכל אחד משלושת ערוצי הקלט/פלט הסטנדרטיים, מוגדר ב-**`stdio.h`** משתנה תואם מטיפוס **`FILE*`**, המאפשר לעבוד עם ערוץ זה כאילו היה קובץ לכל דבר.
- המשתנים הללו נקראים **`stdin`**, **`stdout`** ו-**`stderr`**, בהתאמה; כל פונקציה ב-C שפועלת על קבצים, יכולה באותה המידה לפעול גם על ההתקנים הסטנדרטיים, באמצעות משתנים אלו.

פתיחת קובץ

- על מנת לעבוד עם קובץ שנמצא על הדיסק, יש ראשית לפתוח "ערוץ תקשורת" בין התוכנית לבין קובץ זה.
- כדי לעשות זאת, נקצה משתנה מטיפוס **`FILE*`** שבאמצעותו נתייחס לקובץ, ונשתמש בפקודה **`fopen()`** כדי לקשר אותו לקובץ הרצוי:

```
FILE *my_file;
my_file = fopen(filename, mode);
```

- ***`filename`*** הינו מחרוזת המכילה את שם הקובץ. ניתן לציין את כתובתו המלאה של הקובץ על הדיסק, או את כתובתו היחסית ביחס לספריית העבודה של התוכנית (בד"כ הספרייה בה התוכנית נמצאת).
- ***`mode`*** הינו מחרוזת המציינת את אופן פתיחת הקובץ (נראה בהמשך).
- בכל מקרה של תקלה, **`fopen()`** מחזירה 0 (כלומר מצביע NULL).

אופני פתיחת קובץ

- הפרמטר **mode** יכול להיות אחת משלוש המחרוזות הבאות:

`"r"`	**read** : פתיחת הקובץ לקריאה	מצב זה משמש לקריאת נתונים מקובץ קיים. אם לא קיים קובץ בשם שציינו, הפונקציה מחזירה 0 (מצביע NULL).
`"w"`	**write** : פתיחת הקובץ לכתיבה	מצב זה משמש לכתיבת פלט לקובץ. אם כבר קיים על הדיסק קובץ בשם שציינו, תוכנו **נמחק** והכתיבה מתחילה בראש הקובץ; אם לא קיים עדיין קובץ בשם שציינו, נוצר אוטומטית קובץ חדש ריק עם שם זה.
`"a"`	**append** : פתיחת הקובץ להוספה	מצב זה מאפשר לפתוח קובץ קיים ולהוסיף לו תוכן בהמשך לתוכן הקיים. התוכן המקורי אינו נמחק, וכל המידע שאנו רושמים לקובץ מתווסף לו בסופו. במידה ולא קיים עדיין קובץ בשם שציינו, הוא נוצר אוטומטית והכתיבה מתחילה בראשו.

סגירת קובץ

- בתום העבודה עם קובץ, יש לנתק את הקשר בינו לבין התוכנית. הפקודה לסגירת קובץ היא:

```
int fclose(FILE* file);
```

- פונקציה זו מחזירה 0 במקרה של הצלחה, או **EOF** בכישלון.
- <u>הערה</u>: פעולת הסגירה אינה הכרחית, שכן עם סיום ריצת התוכנית מערכת ההפעלה סוגרת אוטומטית את כל הקבצים הפתוחים הקשורים איתה. עם זאת, רצוי לסגור מפורשות כל קובץ שאינו דרוש עוד. שתי סיבות לכך הן:

1. מספר הקבצים שהתוכנית יכולה לפתוח בו זמנית הוא מוגבל.
2. עבור קבצי פלט: לשם שיפור ביצועים, המידע שהתוכנית כותבת לקובץ למעשה אינו נכתב מיידית לדיסק, אלא נאגר, ורק כאשר מצטבר מידע רב לכתיבה הוא נכתב בפועל. סגירת קובץ הפלט גורמת למערכת ההפעלה לכתוב את כל המידע שהצטבר לדיסק, ובכך מקטינה את הסיכוי שהוא יאבד במקרה של תקלה.

קריאה וכתיבה לקובץ

- הפונקציות לקריאה וכתיבה של קבצים דומות מאוד לאלו שאנו מכירים עבור אמצעי הקלט והפלט הסטנדרטיים. אנו נכיר שלושה סוגים של פונקציות קלט/פלט בקבצים:

טיפוס	קלט	פלט
קלט/פלט של תווים	`fgetc()`	`fputc()`
קלט/פלט של מחרוזות	`fgets()`	`fputs()`
קלט/פלט חכם	`fscanf()`	`fprintf()`

- כל הפונקציות הללו מקבלות כפרמטר את המצביע לקובץ שעליו עליהן לפעול. שימו לב שפרמטר זה יכול להיות גם אחד המשתנים **stdin**, **stdout** ו-**stderr**.
- את פונקציות הקלט/פלט של מחרוזות נכיר בפרק על מחרוזות.

קלט/פלט של תווים

- על מנת לקרוא את התו הבא מקובץ מסויים שפתוח לקריאה, נשתמש בפונקציה **`fgetc()`**; הפרמטר שלה צריך להיות מצביע לקובץ ממנו אנו רוצים לקרוא (מטיפוס **`FILE*`**):

```
int fgetc(FILE* infile);
```

- הפונקציה מחזירה, בדיוק כמו **`getchar()`**, את קוד ה-ASCII של התו שנקרא, או **EOF** במידה ואין יותר תווים לקרוא מהקובץ.
- קריאה לפונקציה **`getchar()`** שקולה לקריאה ל-**`fgetc()`** עם הפרמטר **stdin**:

```
c = getchar();
```
⟺
```
c = fgetc(stdin);
```

קלט/פלט של תווים

- על מנת לכתוב תו לקובץ מסויים שפתוח לכתיבה, נשתמש בפונקציה **fputc()**; הפרמטר הראשון שלה הוא הוא התו לכתיבה, והשני הוא מצביע לקובץ אליו אנו רוצים לכתוב:

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

- הפונקציה מחזירה, בדיוק כמו **putchar()**, את קוד ה-ASCII של התו שנכתב, או **EOF** במידה וקרתה תקלה בכתיבה.
- קריאה לפונקציה **putchar()** שקולה לקריאה ל-**fputc()** עם הפרמטר **stdout**:

```
putchar(c);
```
⟺
```
fputc(c, stdout);
```


דוגמה: העתקת קובץ

- לדוגמה, נראה תוכנית שמשכפלת קובץ כלשהו. התוכנית תקרא קובץ מקור, ותעתיק את כל תוכנו לתוך קובץ יעד חדש.
- נניח לעתה כי שמות קבצי המקור והיעד נתונים כ-**#define**. בהמשך, כשנכיר מחרוזות ב-C, נראה כי ניתן גם לקרוא ערכים אלו מהמשתמש בזמן ריצת התוכנית.

```
#define SRC_FILE_NAME "source.txt"
#define TRG_FILE_NAME "target.txt"

int main() {
  …
}
```


דוגמה: העתקת קובץ

```
int main()
{
  FILE *src, *dest;
  int ch;

  if( (src = fopen(SRC_FILE_NAME, "r")) ==0 )
    printf("Error opening %s\n", SRC_FILE_NAME);
    return 1;
  }

  if( (dest = fopen(TRG_FILE_NAME, "w")) == 0 ) {
    printf("Error opening %s\n", TRG_FILE_NAME);
    return 1;
  }

  …
```

אם פתיחת הקובץ נכשלת, **fopen()** מחזירה 0 ואנו עוצרים את התוכנית

פתיחת קובץ המקור לקריאה

פתיחת קובץ היעד לכתיבה

דוגמה: העתקת קובץ

```
  …

  while ( (ch=fgetc(src)) != EOF )
  {
    if (fputc(ch, dest) == EOF) {
      return 1;
    }
  }

  fclose(src);
  fclose(dest);

  return 0;
}
```

הלולאה נמשכת כל עוד **fgetc()** לא נתקלת בסוף הקובץ.

אם הכתיבה נכשלת, אנו עוצרים את ריצת התוכנית.

העתקת תוכן קובץ המקור לקובץ היעד תו אחר תו.

סגירת הקבצים.

קלט ופלט חכמים

- קיימות גם פונקציות קלט/פלט חכמות לעבודה עם קבצים, הדומות מאוד לאלו שראינו עבור אמצעי הקלט והפלט הסטנדרטיים:
- הפונקציה **`fprintf()`** היא המקבילה של **`printf()`**.
- הפונקציה **`fscanf()`** היא המקבילה של **`scanf()`**.
- פונקציות אלה מקבלות כפרמטר ראשון מצביע לקובץ איתו עובדים; מעבר לכך, יתר הפרמטרים שלהן זהים לחלוטין לאלו שראינו בפונקציות הקלט/פלט הסטנדרטיות, וגם ערך ההחזרה שלהן זהה. לדוגמה:
- כתיבה לקובץ:

```
fprintf(outfile, "i = %d", i);
```

- קריאה מקובץ:

```
fscanf(infile, "%lf", &d);
```


קלט ופלט חכמים

- קריאה לפונקציה **`printf()`** שקולה לקריאה ל-**`fprintf()`** עם הפרמטר **`stdout`**:

```
printf("boo!");   ⟺   fprintf(stdout,"boo!");
```

- קריאה לפונקציה **`scanf()`** שקולה לקריאה ל-**`fscanf()`**, עם הפרמטר **`stdin`**:

```
scanf("%d", &x);   ⟺   fscanf(stdin,"%d", &x);
```

- כמו כן ניתן לקרוא ל-**`fprintf()`** עם התקן הפלט **`stderr`**.

ערוץ השגיאה הסטנדרטי

- ערוץ השגיאה הסטנדרטי **`stderr`** הוא ערוץ פלט מיוחד, שמיועד להצגת הודעות שגיאה ע"י התוכנית. בדרך כלל, ערוץ זה מקושר למסך.
- ערוץ השגיאה **`stderr`** מופרד מערוץ הפלט הסטנדרטי **`stdout`**, וזאת כדי שהודעות שגיאה יוצגו על המסך **גם כאשר מפעילים redirection על הפלט הסטנדרטי**. כלומר, גם אם יתר הפלט נכתב לקובץ, למשל, הודעות השגיאה מוצגות תמיד למסך.
- על מנת לכתוב ל-**`stderr`**, נשתמש בפונקציות הרגילות לכתיבה לקבצים, כשאנו מציינים את **`stderr`** בתור המצביע לקובץ. למשל:

```
if ((infile = fopen("input.dat","r") == 0) {
  fprintf(stderr, "unable to open file\n");
  exit(1);
}
```


תוכנית לדוגמה: קליטת וניתוח ציונים

- נכתוב תוכנית פשוטה לניהול ציונים בקורס. הפעולות שבהן יש לתמוך הן:
  - קליטת הציונים של תרגיל כלשהו מהמשתמש, וכתיבתם לקובץ חדש.
  - קריאת הציונים של תרגיל מקובץ קיים, וחישוב סטטיסטיקות על הציונים.
  - קריאת הציונים של תרגיל מקובץ קיים, הפעלת פקטור עליהם, והוספת הציונים המעודכנים לקובץ ראשי שמרכז את ציוני כל התרגילים בקורס.

- הערה: לשם הפשטות, נניח בתוכנית זו כי מספר זהות הוא בעל 8 ספרות, ולכן ניתן לייצוג ע"י **`long`**.

דוגמה לריצת התוכנית

נראה ראשית דוגמה לריצת התוכנית שברצוננו לבנות. נתחיל בתהליך קליטת הנתונים מהמשתמש; בתהליך זה, הציונים של תרגיל מסויים נקראים מהמשתמש ונכתבים לקובץ בשם **hwN.dat** (כש-**N** הוא מספר התרגיל).

```
Grades XP 1.003.564
Copyright Jan 2005, Technion Systems, Inc

Enter command:

0 - quit
1 - read new exercise grades
2 - get exercise statistics
3 - finalize exercise grades and add to main database

> 1
…
```


דוגמה לריצת התוכנית

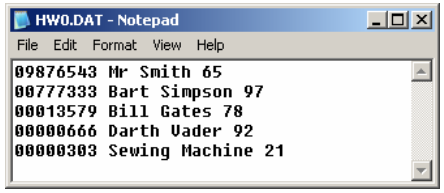
```
Enter exercise number (0-9): 0
Enter number of grades: 5
ID: 9876543
Name (first and last): Mr Smith
Grade: 65
ID: 777333
Name (first and last): Bart Simpson
Grade: 97
ID: 13579
Name (first and last): Bill Gates
Grade: 78
ID: 666
Name (first and last): Darth Vader
Grade: 92
ID: 303
Name (first and last): Sewing Machine
Grade: 21
```


דוגמה לריצת התוכנית

hw0.dat

```
09876543 Mr Smith 65
00777333 Bart Simpson 97
00013579 Bill Gates 78
00000666 Darth Vader 92
00000303 Sewing Machine 21
```

- פעולה זו יוצרת קובץ חדש בשם **hw0.dat**, שנראה כך: (שימו לב שמספרי הזהות נכתבים בשמונה ספרות)



- קובץ זה הוא קובץ טקסט לכל דבר, וניתן לעיין בתוכנו, למשל בעזרת תוכנת Notepad של Windows:

דוגמה לריצת התוכנית

נעבור כעת לפעולת חישוב הסטטיסטיקות. פעולה זו קוראת את הציונים של תרגיל כלשהו מהקובץ **hwN.dat** (שנוצר בתהליך דומה לקודם) ומציגה את מספר הציונים בקובץ, הממוצע שלהם, וסטיית התקן:

```
Enter command:

0 - quit
1 - read new exercise grades
2 - get exercise statistics
3 - finalize exercise grades and add to main database

> 2
Enter exercise number (0-9): 0

Number of grades: 5
Average grade: 70.600000
Grade standard deviation: 27.207352
```

דוגמה לריצת התוכנית

- בפעם הראשונה שאנו מבצעים פעולה זו, הקובץ **master.dat** עדיין איננו קיים, ולכן התוכנית מייצרת אותו אוטומטית. בתום התהליך הקובץ מכיל את הציונים המעודכנים של תרגיל 0 (לאחר הכפלה ב-0.8), ונראה כך:

master.dat

```
*** Grades for exercise 0 ***

09876543 Mr Smith 52
00777333 Bart Simpson 77
00013579 Bill Gates 62
00000666 Darth Vader 73
00000303 Sewing Machine 16
```


דוגמה לריצת התוכנית

לסיום, נתבונן בתהליך כתיבת קובץ הציונים הסופי. בתהליך זה נקרא קובץ הציונים של תרגיל מסויים (**hwN.dat**), המשתמש בוחר בפקטור הרצוי, והציונים המעודכנים משורשרים לסוף הקובץ **master.dat** שמרכז את הציונים הסופיים של כל התרגילים.

```
Enter command:
…

> 3
Enter exercise number (0-9): 0
Enter factor type:
0 - multiply by 0.8
1 - square root factor
2 - failure factor
> 0
```


דוגמה לריצת התוכנית

- אנו יכולים כעת להפעיל את תהליך מס' 3 גם על קבצי הציונים של תרגילים 1 ו-2, כאשר בכל אחד מהם נשתמש בפקטור אחר כרצוננו. התוכנית תחשב לכל תרגיל את הציונים המעודכנים, ותוסיף אותם בסוף הקובץ **master.dat**.

```
Enter command:
…
> 3
Enter exericse number (0-9): 1
Enter factor type:
0 - multiply by 0.8
1 - square root factor
2 - failure factor
> 1

Enter command:
…
> 3
Enter exericse number (0-9): 2
Enter factor type:
0 - multiply by 0.8
1 - square root factor
2 - failure factor
> 2
```


דוגמה לריצת התוכנית

- כעת נניח שיצרנו באמצעות התוכנה גם את קבצי הציונים של תרגילים 1 ו-2, והם נראים כך:

hw1.dat

```
09876543 Mr Smith 85
00777333 Bart Simpson 76
00013579 Bill Gates 30
00000666 Darth Vader 97
00000303 Sewing Machine 53
```

hw2.dat

```
09876543 Mr Smith 85
00777333 Bart Simpson 76
00013579 Bill Gates 30
00000666 Darth Vader 97
00000303 Sewing Machine 53
```

דוגמה לריצת התוכנית

master.dat

```
*** Grades for exercise 0 ***

09876543 Mr Smith 52
00777333 Bart Simpson 77
00013579 Bill Gates 62
00000666 Darth Vader 73
00000303 Sewing Machine 16

*** Grades for exercise 1 ***

09876543 Mr Smith 92
00777333 Bart Simpson 87
00013579 Bill Gates 54
00000666 Darth Vader 98
00000303 Sewing Machine 72

*** Grades for exercise 2 ***

09876543 Mr Smith 65
00777333 Bart Simpson 90
00013579 Bill Gates 54
00000666 Darth Vader 100
00000303 Sewing Machine 54
```

- בסוף התהליך, הקובץ **master.dat** מכיל את הציונים של כל התרגילים, ונראה כך:

בחזרה לתוכנית הציונים

- בשקפים הבאים נראה בקצרה את המימוש של הפונקציות השונות שמרכיבות את התוכנית.
- קוד התוכנית המלא נמצא באתר הקורס, ומצורף לשקפים אלה. מתוך כלל הקוד, רק חלקים נבחרים מופיעים באוסף השקפים, על מנת להדגים את השימוש בפונקציות הקלט/פלט השונות.
- לצורך פישוט וקיצור הקוד המופיע בשקפים, רוב הקוד לטיפול בשגיאות הוסר, וניתן למצוא אותו בקוד המלא. לפיכך, ערכי ההחזרה של פונקציות הקלט/פלט אינם תמיד נבדקים ואנו מניחים שהן מצליחות. בפועל, יש לוודא תמיד את ערכי ההחזרה, ולטפל בכל מקרה של שגיאה לגופו.

בחזרה לתוכנית הציונים

- שימו לב, למשל, שעל אף שערך ההחזרה של הפונקציה **printf()** לא נבדק על פי רוב, הרי שהפונקציה **fprintf()** יכולה להיכשל כאשר היא מנסה לכתוב לקובץ, ולכן יש לוודא את ערך ההחזרה שלה.
- בקורס זה לא ננסה להתמודד ולפתור בעיות של קלט/פלט; לכן, בכל מקרה של תקלת קלט/פלט, אנו פשוט נעצור את התוכנית. עם זאת, מומלץ לכתוב הודעת שגיאה כלשהי ל-**stderr** לפני עצירת התוכנית.
- מומלץ לעבור על הקוד המלא של התוכנית, על מנת לראות את אופן הטיפול בשגיאות. כפי שאולי תבחינו, הקוד עצמו מעט מסורבל; אך אל דאגה, בשפת C קיימים כלים לפשט מאוד קוד זה, וביניהם מחרוזות, מבנים, Macros ועוד. חלק מנושאים אלו יילמדו בהמשך הקורס, ואלו יאפשרו לכם לכתוב בעתיד קוד קצר וקריא יותר.

תוכנית לדוגמה: קליטת וניתוח ציונים

- נתכנת את התוכנית בגישת Top-Down. התוכנית תכלול שלוש פונקציות מרכזיות, אחת לכל משימה שהתוכנית צריכה לבצע. לפיכך, פונקצית ה-**main()** מקבלת את הצורה הזו:

```
printf("\nGrades XP 1.003.564\n"
   "Copyright Jan 2005, Technion Systems, Inc\n");

do {
  cmd = get_command();
  switch(cmd) {
    case CMD_GET_GRADES: get_new_grades();      break;
    case CMD_DO_STATS:   do_grade_statistics(); break;
    case CMD_FINALIZE:   finalize_grades();     break;
  }
} while (cmd != CMD_QUIT);
```

תוכנית לדוגמה: קליטת וניתוח ציונים

- נתחיל לתכנת את הפונקציות השונות. שימו לב שלא נפרט כאן את המימוש של כל הפונקציות כולן, אלא רק חלקים נבחרים. את הקוד המלא של התוכנית ניתן למצוא באתר הקורס, מצורף לשקפים אלו. מומלץ בחום להציץ בו.
- הפונקציה הראשונה שהשתמשנו בה היא בעלת החתימה הבאה:

```
int get_command();
```

- פונקציה זו מציגה למשתמש תפריט, ומבקשת ממנו לבחור במשימה הרצויה. הפונקציה מחזירה אחד משלושת הקבועים השלמים **CMD_GET_GRADES**, **CMD_DO_STATS** ו-**CMD_FINALIZE**, המוגדרים כ-**#define**. פונקציה זו לא תמומש בשקפים.

הפונקציה לקליטת הציונים

```
int readgrade()
{
  FILE *gradefile; int hw_num;

  printf("Enter exercise number (0-9): ");
  hw_num = read_int_range(0,9);

  gradefile = open_hw_file(hw_num, MODE_WRITE);
  if (gradefile==0) return (-1);

  …

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

פונקציה הקוראת מהמשתמש **int** בטווח המציין

פונקציה שפותחת את הקובץ המתאים בהינתן מספר התרגיל, ומחזירה מצביע לקובץ זה

ערך החזרה זה מציין שגיאה

הפונקציה לקליטת הציונים

- הפונקציה **read_int_range()** היא פונקציית עזר שמבקשת מהמשתמש להכניס מספר שלם בטווח הנקוב, ומבצעת את כל הווידוא הדרוש. החתימה שלה היא:

```
int read_int_range(int min, int max);
```

- הפונקציה **open_hw_file()** מקבלת את מספר התרגיל הרצוי (0-9) ואת אופן הפתיחה של הקובץ (קריאה/כתיבה) ופותחת את קובץ הנתונים המתאים. הפרמטר השני שלה מציין את אופן הפתיחה הרצוי של הקובץ, ויכול להיות אחד משני הקבועים השלמים **MODE_READ** ו-**MODE_WRITE**, המוגדרים כ-**#define**. היא מחזירה מצביע לקובץ שפתחה. החתימה שלה היא:

```
FILE* open_hw_file(int hw_num, int mode);
```


הפונקציה לקליטת הציונים

```
FILE* open_hw_file(int hw_num, int mode)
{
  FILE *gradefile;

  switch(hw_num) {
    case 0: gradefile =
              fopen("hw0.dat",(mode==MODE_READ) ? "r":"w");
            break;
    case 1: gradefile =
              fopen("hw1.dat",(mode==MODE_READ) ? "r":"w");
            break;
    …
  }

  return gradefile;
}
```

פתיחת הקובץ לקריאה/כתיבה בהתאם לפרמטר **mode**

שימו לב שבמקרה של כישלון הפונקציה **fopen()**, מוחזר כאן 0 (null)

המשך – הפונקציה לקליטת הציונים

- נמשיך עם מימוש הפונקציה לקליטת הציונים. לב הפונקציה הוא :

```
printf("enter number of grades: ");
grade_num = read_int_min(1);

for (i=0; i<grade_num; ++i) {
  get_student_info(gradefile)
}
```

- הפונקציה **`read_int_min()`** דומה ל-**`read_int_range()`**; היא קוראת מהמשתמש מספר שלם הגדול או שווה לפרמטר שלה.
- הפונקציה **`get_student_info()`** קוראת מהמשתמש את הנתונים של סטודנט אחד, וכותבת אותם לקובץ שהיא מקבלת כפרמטר. נראה את המימוש שלה כעת.

המשך – הפונקציה לקליטת הציונים

```
int get_student_info(FILE* gradefile)
{
  long id; int grade;

  printf("ID: "); id = read_long_min(0);
  fprintf(gradefile, "%08ld ", id);

  printf("Name (first and last): ");
  copy_word(stdin, gradefile); fputc(' ', gradefile);
  copy_word(stdin, gradefile); fputc(' ', gradefile);

  printf("Grade: "); grade = read_int_min(0);
  fprintf(gradefile, "%d\n", grade);

  return 1;
}
```


המשך – הפונקציה לקליטת הציונים

הסברים לפונקציה הקודמת:

- אנו מתחילים מלקרוא את מספר הזהות מהמשתמש, כטיפוס **`long`**. הפונקציה **`read_long_min()`** מבצעת אותה הפעולה כמו **`read_int_min()`**, רק עבור משתנים מטיפוס **`long`**.
- שימו לב שמספר הזהות נכתב לקובץ הפלט ברוחב שדה של 8 ספרות, עם אפסים מובילים, בדיוק כפי שראינו ב-**`printf()`**.
- הפונקציה **`copy_word()`** קוראת את המילה הבאה (= רצף של תווים ללא רווחים) מקובץ הקלט הנתון, וכותבת אותה לקובץ פלט. החתימה שלה היא:

```
int copy_word(FILE *src, FILE *dest);
```


המשך – הפונקציה לקליטת הציונים

- שימו לב שבמקרה שלנו "קובץ הקלט" שאנו מעבירים לפונקציה **`copy_word()`** הוא **הקלט הסטנדרטי**: המשתמש מקליד את השם הפרטי ושם המשפחה של הסטודנט (נניח כי כל אחד מאלו הוא מילה אחת), והתוכנית מעתיקה שתי מילים אלו לקובץ הציונים בזו אחר זו ע"י קריאה פעמיים לפונקציה **`copy_word()`**. אנו נממש פונקציה זו.
- הפונקציה **`copy_word()`** נדרשת לקרוא ולהשמיט רווחים (אם יש כאלו) מקובץ הקלט עד שהיא מגיעה לתחילת המילה הקרובה (כלומר לתו שאינו רווח); כעת היא מעתיקה לקובץ הפלט את כל התווים במילה, עד שהיא נתקלת שוב בתו רווח, בו היא מפסיקה את ההעתקה.
- בכתיבה לקובץ הציונים, אנו מוסיפים ריווח בין המילים, על ידי שימוש בפונקציה **`fputc()`**.

הפונקציה להעתקת מילה

```
int copy_word(FILE* src, FILE* dest)
{
  int ch;

  do {
    if (ch=fgetc(src)==EOF)
      return EOF;
  } while (isspace(ch));

  do {
    if (fputc(ch, dest)==EOF)
      return 0;
    ch = fgetc(src);
  } while (ch!=EOF && !isspace(ch));

  return 1;
}
```

דילוג על רווחים

העתקת המילה

אם קובץ הקלט מסתיים בשלב הזה, הפונקציה מחזירה **EOF** כדי לציין שלא נקראה כל מילה מקובץ הקלט.

ההעתקה נמשכת כל עוד לא נתקלנו בתו רווח או בסוף קובץ הקלט.

0 מציין בעייה בכתיבה לקובץ הפלט

1 מציין הצלחה

הפונקציה לביצוע הסטטיסטיקות

- נעבור לפונקציה שמחשבת סטטיסטיקה עבור תרגיל מסויים. פונקציה זו קוראת את קובץ הציונים של תרגיל מסויים, ומדפיסה את מספר הציונים בקובץ, את הממוצע שלהם ואת סטיית התקן שלהם.
- נזכיר כי הממוצע של $\{x_1, x_2, \ldots, x_n\}$ הינו $\mu_x = \frac{1}{n}\sum_i x_i$
- וסטיית התקן הינה $\sigma_x = \sqrt{\frac{1}{n}\sum_i x_i^2 - \left(\frac{1}{n}\sum_i x_i\right)^2}$

הפונקציה לחישוב הסטטיסטיקות

```
int getinfo()
{
  FILE *gradefile; int hw_num;

  printf("Enter exercise number (0-9): ");
  hw_num = read_int_range(0,9);

  gradefile = open_hw_file(hw_num, MODE_READ);
  if (gradefile==0) return (-1);

  …

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

כעת אנו פותחים את הקובץ לקריאה בלבד

הפונקציה לביצוע הסטטיסטיקות

- והנה לב הפונקציה :

```
status = 1; grade_num = grade_avg = grade_std = 0;
do {
  status = get_next_grade(gradefile, &grade);
  if (status != 1) break;

  grade_num ++ ;
  grade_avg += grade;
  grade_std += grade*grade;
} while(1);

grade_avg /= grade_num;
grade_std = sqrt(grade_std/grade_num-grade_avg*grade_avg);

printf("Average grade: %lf\n", grade_avg);
printf("Grade standard deviation: %lf\n", grade_std);
```

status מקבל את הערך 1 אם הפעולה הצליחה, ו-**EOF** כאשר אין אין עוד ציונים לקרוא

הפונקציה לביצוע הסטטיסטיקות

- הפונקציה **get_next_grade()** קוראת מקובץ הציונים את הפרמטרים של הסטודנט הבא, וכותבת את הציון שקראה לתוך המצביע **grade**. היא מחזירה **EOF** אם לא היה סטודנט נוסף לקרוא מהקובץ.

טיפול ב-ID של הסטודנט

דילוג על שם הסטודנט (פרטי ומשפחה)

קריאת הציון לתוך **grade**

```
int get_next_grade(FILE* gradefile,int *grade)
{
  long id;
  if (fscanf(gradefile, "%ld", &id)) == EOF)
    return EOF;

  skip_word(gradefile);
  skip_word(gradefile);

  fscanf(gradefile, "%d", grade);

  return 1;
}
```


הפונקציה לביצוע הסטטיסטיקות

הערות לפונקציה:

- אנו זקוקים אך ורק לציון של הסטודנט ולא לכל יתר הפרמטרים שלו. לכן, תעודת הזהות נקראת מהקובץ אך אנו מתעלמים ממנה, וכך גם שם הסטודנט, שאותו אנו קוראים על ידי שימוש כפול בפונקציה **skip_word()**.
- הפונקציה **skip_word()** קוראת מקובץ הקלט הנתון מילה אחת, ולא עושה איתה דבר. במילים אחרות, היא מקדמת אותנו "מילה אחת" בקובץ. אופן הפעולה שלה דומה מאוד לפונקציה **copy_word()**, אלא שהיא אינה כותבת את המילה שקראה לקובץ אחר. חתימתה היא:

```
int skip_word(FILE *src);
```

וניתן למצוא את הקוד שלה בקובץ הקוד של התוכנית.

הפונקציה לכתיבת הציונים הסופיים

```
int writegrade()
{
  FILE *gradefile, *masterfile; int hw_num;

  printf("Enter exercise number (0-9): ");
  hw_num = read_int_range(0,9);

  gradefile = open_hw_file(hw_num, MODE_READ);
  if (gradefile==0) return (-1);

  if ((masterfile = fopen(MASTER_FILE,"a")) == 0)
    return (-1);

  …

  fclose(gradefile);
  fclose(masterfile);
  return 0;
}
```

פתיחת הקובץ הראשי להוספה (אנו נכתוב את הציונים הסופיים של התרגיל לסוף קובץ זה)

הפונקציה לכתיבת הציונים הסופיים

- והנה לב הפונקציה :

קבלת סוג הפקטור הרצוי מהמשתמש

כתיבת שורת כותרת עם מספר התרגיל לקובץ הראשי

כתיבת הציון הסופי של סטודנט אחד לקובץ הראשי

```
factor_type = get_factor();

fprintf(masterfile,
     "\n*** Grades for exercise %d ***\n\n",
     hw_num );

status = 1;
do {
  status = finalize_student_grade(
             gradefile, masterfile, factor_type);
} while(status == 1);
```

הפונקציה לכתיבת הציונים הסופיים

הערות לפונקציה:

- פונקצית העזר **`get_factor()`** מציגה למשתמש תפריט עם הפקטורים האפשריים, קוראת ממנו את הבחירה שלו, ומחזירה אותו כאחד מן הקבועים האפשריים לפקטור, במקרה זה **`FACTOR_MULT`**, **`FACTOR_SQRT`**, ו-**`FACTOR_FAIL`**. הפונקציה מבצעת את כל הווידוא הנדרש.
- הפונקציה **`finalize_student_grade()`** קוראת את הנתונים של סטודנט בודד מקובץ הציונים, וכותבת אותם (אחרי הפעלת הפקטור) לקובץ הראשי. היא דומה מאוד לפונקציות שכבר ראינו, וניתן למצוא את הקוד המלא שלה בקובץ הקוד של התוכנית.