CCPROG2 File Processing Programming Exercises *サルバドル・フロランテ*

A skeleton file is not given. You'll have to write your codes starting from scratch. Write one C program per challenge. Since there are 8 challenges, you should produce 8 C source programs each with its own main() function. Practice by writing functions other than just main().

Assume that a text file named "STUDENTS.TXT" exists which contains information about students stored as follows:

a. 1st column: student ID number
b. 2nd column: student's first name
c. 3rd column: student's last name
d. 4th column: student's grade (from 0.0 to 4.0)

We do not know in advance how many students information are stored in the file. It maybe for just 10 students, or it may be for hundreds of students. Your solution should work correctly regardless of the number of students. Note that it is incorrect for your program to ask the user to input the number of students.

Assume also the existence of the following declarations:

```
typedef char String20[20];

struct nameTag {
     String20 first;
     String20 last;
};

struct studentTag {
      int ID; // student ID number
      struct nameTag name;
      float grade;
};
```

It is up to you to decide if you would use these declarations in your programs or not, UNLESS it is explicitly required to use them in the challenge.

Notation: items shown in **blue** represent screen output, while items shown in **green** represent data stored in a file.

/* The EIGHT challenges are described in the following pages */

CHALLENGE #1: Read all the contents of " STUDENTS.TXT" and printf() the student information. Example screen output:

```
1010  Jose Manalo      3.5
2003  Luisa Santos     2.5
1008  Andres Espinosa  4.0
1013  Miguel Fuentes   1.0
2011  Nena Fajardo     1.5
2008  Anton Macaraig   0.0
1188  Benson Serrano   4.0
2223  Eliza Martinez   2.0
2103  Elmer Cruz       3.0
2288  Andrea Sy        3.5
1099  Melissa Garcia   0.0
2340  Alberto Trang    1.5
```

CHALLENGE #2: Read all the contents of " STUDENTS.TXT" and produce as a result two text files described below:

a. FAIL.TXT – contains data about students who failed, i.e., grade is 0.0
b. PASS.TXT – contains data about students who passed, i.e., grade is not 0.0

Note: You should NOT use printf() in solving this problem.

Using the example result in Challenge #1 above, the contents of FAIL.TXT will be:

```
2008  Anton Macaraig   0.0
1099  Melissa Garcia   0.0
```

Using the example result in Challenge #1 above, the contents of PASS.TXT will be:

```
1010  Jose Manalo      3.5
2003  Luisa Santos     2.5
1008  Andres Espinosa  4.0
1013  Miguel Fuentes   1.0
2011  Nena Fajardo     1.5
1188  Benson Serrano   4.0
2223  Eliza Martinez   2.0
2103  Elmer Cruz       3.0
2288  Andrea Sy        3.5
2340  Alberto Trang    1.5
```

CHALLENGE #3: Ask the user to input data for one more student, and append the data to " STUDENTS.TXT".

The following shows an example of user interaction and inputs.

```
Input ID: 1800
Input first name: Manuel
Input last name: Cabrera
Input grade: 2.0
```

If we run the program from Challenge #1, the corresponding output on the screen will be (take note of the last line highlighted in yellow fill color):

```
1010  Jose Manalo      3.5
2003  Luisa Santos     2.5
1008  Andres Espinosa  4.0
1013  Miguel Fuentes   1.0
2011  Nena Fajardo     1.5
2008  Anton Macaraig   0.0
1188  Benson Serrano   4.0
2223  Eliza Martinez   2.0
2103  Elmer Cruz       3.0
2288  Andrea Sy        3.5
1099  Melissa Garcia   0.0
2340  Alberto Trang    1.5
1800  Manuel Cabrera   2.0
```

CHALLENGE #4: Read all the contents of " STUDENTS.TXT" and produce as a result a new binary file named "STUDENTS.BIN" which contains the same student data from the input text file. You should fscanf() the student data into a student structure variable, and fwrite() it to the binary file.

The following shows some idea on how it is done:

```
struct studentTag student;

// read student data from text file, file pointer is fp_txt
fscanf(fp_txt, "%d", &student.ID);
fscanf(fp_txt, "%s %s", student.name.first, student.name.last);
fscanf(fp_txt, "%f", &student.grade);

// write student data to binary file, file pointer is fp_bin
fwrite(&student, sizeof(struct studentTag), 1, fp_bin);
```

CHALLENGE #5: Read all the contents of the binary file " STUDENTS.BIN" and printf() the student information. Please take note that you do not know in advance how many student data values are stored in the binary file.

You should fread() the student data into a student structure variable.

The following shows some idea on how it is done:

```
struct studentTag student;

fread(&student, sizeof(struct studentTag), 1, fp_bin);
```

Example screen output using example data from Challenge #3:

```
1010  Jose Manalo      3.5
2003  Luisa Santos     2.5
1008  Andres Espinosa  4.0
1013  Miguel Fuentes   1.0
2011  Nena Fajardo     1.5
2008  Anton Macaraig   0.0
1188  Benson Serrano   4.0
2223  Eliza Martinez   2.0
2103  Elmer Cruz       3.0
2288  Andrea Sy        3.5
1099  Melissa Garcia   0.0
2340  Alberto Trang    1.5
1800  Manuel Cabrera   2.0
```

CHALLENGE #6: Read contents of the binary file " STUDENTS.BIN" in **reversed order** and copy them onto a new binary file named "STUDENTS-REV.BIN".

You are NOT allowed to declare/use an array of student structure for this challenge.

Hint: to solve this problem, you will need to use fseek() to move the file position. Also, recall that an fread() moves the file position.

Example: using the data from Challenge #3, the contents of "STUDENTS-REV.BIN" will be

```
1800  Manuel Cabrera   2.0
2340  Alberto Trang    1.5
1099  Melissa Garcia   0.0
2288  Andrea Sy        3.5
2103  Elmer Cruz       3.0
2223  Eliza Martinez   2.0
1188  Benson Serrano   4.0
2008  Anton Macaraig   0.0
2011  Nena Fajardo     1.5
1013  Miguel Fuentes   1.0
1008  Andres Espinosa  4.0
2003  Luisa Santos     2.5
1010  Jose Manalo      3.5
```

CHALLENGE #7: Perform linear search on the "STUDENTS.BIN" binary file. The user will input the ID number. If the ID number is found, the program should print the information about the student. Otherwise, it should output "NOT FOUND!" The search will be repeated until the user inputs a sentinel value of -999, in which case the program outputs "END SEARCH".

You are NOT allowed to declare/use an array of student structure for this challenge. You should implement the linear search on the binary file itself. You should open the binary file only ONCE. Hint: use fseek() to reset the file position.

Example:

Input ID number: 2008
```
2008  Anton Macaraig   0.0
```

Input ID number: 2999
```
NOT FOUND
```

Input ID number: 1010
```
1010  Jose Manalo      3.5
```

Input ID number: -999
```
END SEARCH
```

CHALLENGE #8: Search and change the grade of a student stored in "STUDENTS.BIN" file. Use the ID number as a search key. If the ID number exists, the program should printf() the current student information, and then ask the user to input the new grade. If the ID number does not exist, the program should output "NOT FOUND!" The search will be repeated until the user inputs a sentinel value of -999, in which case the program outputs "END SEARCH".

You are NOT allowed to declare/use an array of student structure for this challenge. You should implement the linear search on the binary file itself.

Hint: You need to open the file using "rb+" mode to allow modification/changes in the binary file. You will also need to use fseek() to set the file position.

Example:

Input ID number: 2008
```
2008  Anton Macaraig   0.0
Input new grade: 4.0
```

Input ID number: 2999
```
NOT FOUND
```

Input ID number: 1010
```
1010  Jose Manalo      3.5
```
Input new grade: 1.5

Input ID number: -999
```
END SEARCH
```

** Note: Verify that your solution works correctly by running the program in Challenge #5. The screen output should show that the grade of Anton Macaraig changed to 4.0 and that of Jose Manalo changed to 1.5.

終