Department of Information Systems and Technologies

CTIS 152 – Data Structures and Algorithms

Summer 2020 - 2021

Lab Guide #6 – Week 2 - 4

OBJECTIVE : Structures Operations, Nested Structure and Pointer Operations

Instructor : Okyay SAY
Assistant : Ruşen ASAN

1. Today we will simulate the operation of a vending machine. The product list of the machine is stored in the file **"products.txt"** with the **unit price, quantity** and **name**. (The first line of the file consists the number of products). Create a structure for the products.

Initially the program reads all the products' information from the file into a **dynamically** created structure array and displays them on the screen. (while reading the product name you should use **%[^\n]** in the fscanf)

Then the program gets the product number from the customer and if the product does not run out of in the machine, customer inserts the money, otherwise gives an error message. After the purchase is finished, the remaining money should be returned back to the customer. If the inserted money is not enough to purchase the product, program gives an error message. When the product is run out of, the quantity of the item should be displayed OUT in the menu as in the example run. When the user enters -1 for the product number, program terminates.

Write the following functions;

- `readFromFile, display`

Project Name: LG6_Q1
File Name: Q1.cpp

Example Run:

```
     PRODUCT       UNIT PRICE QUANTITY
  1. Mr. Goodbar      1.00        10
  2. M & Ms.          0.60         1
  3. Potato Chips     4.40         2
  4. Chewing Gum      1.20         1
  5. Pretzels         0.45         6
  6. Hershey Kisses   1.25        11
  7. Kracker Jacks    0.80         4
  8. Oreos            0.55         3
  9. Fritos           0.45         1
 10. Milky Way        0.65        10
Enter product number to purchase (to stop -1):4
Insert the money: 1.50
0.30 TL returned back

     PRODUCT       UNIT PRICE QUANTITY
  1. Mr. Goodbar      1.00        10
  2. M & Ms.          0.60         1
  3. Potato Chips     4.40         2
  4. Chewing Gum      1.20       OUT
  5. Pretzels         0.45         6
  6. Hershey Kisses   1.25        11
  7. Kracker Jacks    0.80         4
  8. Oreos            0.55         3
  9. Fritos           0.45         1
 10. Milky Way        0.65        10
Enter product number to purchase (to stop -1):4
THERE IS NO MORE Chewing Gum

     PRODUCT       UNIT PRICE QUANTITY
  1. Mr. Goodbar      1.00        10
  2. M & Ms.          0.60         1
  3. Potato Chips     4.40         2
  4. Chewing Gum      1.20       OUT
  5. Pretzels         0.45         6
  6. Hershey Kisses   1.25        11
  7. Kracker Jacks    0.80         4
  8. Oreos            0.55         3
  9. Fritos           0.45         1
 10. Milky Way        0.65        10
Enter product number to purchase (to stop -1):3
Insert the money: 2
Your money is not enough, and returned back
```

products.txt

```
10
1.00 10 Mr. Goodbar
0.60 1 M & Ms.
4.40 2 Potato Chips
1.20 1 Chewing Gum
0.45 6 Pretzels
1.25 11 Hershey Kisses
0.80 4 Kracker Jacks
0.55 3 Oreos
0.45 1 Fritos
0.65 10 Milky Way
```

```
     PRODUCT       UNIT PRICE QUANTITY
  1. Mr. Goodbar      1.00        10
  2. M & Ms.          0.60         1
  3. Potato Chips     4.40         2
  4. Chewing Gum      1.20       OUT
  5. Pretzels         0.45         6
  6. Hershey Kisses   1.25        11
  7. Kracker Jacks    0.80         4
  8. Oreos            0.55         3
  9. Fritos           0.45         1
 10. Milky Way        0.65        10
Enter product number to purchase (to stop -1):3
Insert the money: 5
0.60 TL returned back

     PRODUCT       UNIT PRICE QUANTITY
  1. Mr. Goodbar      1.00        10
  2. M & Ms.          0.60         1
  3. Potato Chips     4.40         1
  4. Chewing Gum      1.20       OUT
  5. Pretzels         0.45         6
  6. Hershey Kisses   1.25        11
  7. Kracker Jacks    0.80         4
  8. Oreos            0.55         3
  9. Fritos           0.45         1
 10. Milky Way        0.65        10
Enter product number to purchase (to stop -1):-1

***********************
Today 2 products sold
Total payment is: 5.60 TL
```

2. NBA basketball team needs software to keep the players' information: **name, age, length, points** consisting **average score** and **average rebound**. USE NESTED STRUCTURE ARRAY. Maximum number of player is **20**.

Write the following functions;

- **readFromFile** that reads the players' information from the file "**players.txt**" into an array of structures and returns the number of players.
- **displayAll** that displays the information of all players as in the example run.
- **`findBestRebound`** that finds and returns the index of the player who has the maximum average rebound score.
- **`findAvgHeight`** that calculates and returns the average length of the players in the team.

Write a main program that reads and displays the players' information. The program will also do the followings;

- Displays the information of the best rebound player,
- Displays the average length of the players in the team,

Please examine the given example runs, and test your programs.

Project Name: LG6_Q2
File Name: Q2.cpp

Example Run :

```
NAME       Age  Length  Avg_Score  Avg_Rebound
-----------------------------------------------
Brandon     28   2.03      18.40        10.40
Avery       23   1.88      27.30        14.40
Jordan      25   1.93      16.10         9.70
Kevin       37   2.11      28.60        15.80
Jeff        27   2.06      15.80        11.30
Courtney    28   1.96      28.40        14.60
Jason       36   1.88      16.80         6.90
Chris       31   2.08      17.40        12.70
Nikola      24   2.01      16.70        13.20

The Best rebound player:
NAME       Age  Length  Avg_Score  Avg_Rebound
-----------------------------------------------
Kevin       37   2.11      28.60        15.80

The Avg Length of all players: 1.99
```

players.txt

```
Brandon  28 2.03 18.4 10.4
Avery    23 1.88 27.3 14.4
Jordan   25 1.93 16.1 9.7
Kevin    37 2.11 28.6 15.8
Jeff     27 2.06 15.8 11.3
Courtney 28 1.96 28.4 14.6
Jason    36 1.88 16.8 6.9
Chris    31 2.08 17.4 12.7
Nikola   24 2.01 16.7 13.2
```

Additional Question

Create a nested structure **applicantsOfII** and **grades** as follows:

```
typedef struct{
        int englishProficiency, jury, graduateExam;
 } grades_t;

 typedef struct{
       int id;
       grades_t gr;
       double overall;
}applicantsOfII_t;
```

Write the following functions:

- **readFile,** which gets a set of application information from a text file named **applicants.txt** until the end of the file is reached, also returns the size of the structure array (Do not forget to initialize the overall grade to 0 for each student).
- **calculate,** which calculates the overall applicants' grades' average and the overall grade of each applicant with the loads of English proficiency being 30%, jury being 50%, and the graduate exam being 20%)
- **display,** which displays the content of the structure array of **applicantsOfII_t** type.
- **findPassFail,** a function that finds and displays the number of the applicants who fail and pass the elimination as well as displaying the average of all applicants' grades'. (An applicant passes if overall >= average, otherwise student fails).

Write a C program that reads the entirety of applicants' information from applicants.txt file into an array of structures, and displays all the information on the screen as necessary, as shown in the example run below.

Project Name: LG6_AQ
File Name: AQ.cpp

applicants.txt

```
1222 45 67 98
1333 89 45 33
1444 67 76 99
```

Example Run:

```
Applicant ID: 1222
Scores:
Applicant English Proficiency: 45
Applicant Jury: 67
Applicant Graduate Examination: 98
Applicant Overall: 66.6

Applicant ID: 1333
Scores:
Applicant English Proficiency: 89
Applicant Jury: 45
Applicant Graduate Examination: 33
Applicant Overall: 55.8

Applicant ID: 1444
Scores:
Applicant English Proficiency: 67
Applicant Jury: 76
Applicant Graduate Examination: 99
Applicant Overall: 77.9

Average is 66.8
Number of the applicants who pass is 1
Number of the applicants who fail is 2
```