Department of Information Systems and Technologies

CTIS 152 –Algorithms and Data Structure

Spring 2024 - 2025

Lab Guide #5 – Week 3 – 2

OBJECTIVE : Structures as Function Parameters, Nested Structures

Instructor : Serpil TIN
Assistants : Berk ÖNDER, Hatice Zehra YILMAZ

Q1. A real Estate Agency keeps the property information in the text file **properties.txt**. Each property information consists of **Property ID**, **Property type** (**R**-Rental or **S**-Sale), property **price,** the number of **bedrooms,** and the number of **bathrooms**.

The first line of the file contains the number of properties (***n)*** and each following line provides the property details.

Write a C program that reads the information for ***n*** properties from the text file into a **dynamically** created property structure array. Then, it displays the property details, calculates and displays the total price for rental and sale properties.

Write the following functions;

- **readFromFile:** takes the file pointer, dynamically created properties array, and the number of properties as parameters, then reads the properties' information from the file into the properties array.
- **displayAndCalculate:** takes the dynamically created properties array and the number of properties as parameters and displays the information of all properties as in the example run. The function calculates and returns the total price for **rental** and **sale** properties.

Project Name: LG5_Q1
File Name: Q1.cpp

Example Run:

```
Real Estate Agency
ID      Type      Price (TL)       Bedroom#  Bathroom#
******************************************************
  111   Rented      12000.50          2          1
  222   Sold      2500000.00          3          2
  333   Rented      38000.75          4          2
  444   Sold      3200000.00          3          3
  555   Rented      19500.00          1          1
  666   Sold      8450000.00          5          4
  777   Rented      21000.00          3          2
  888   Sold      6500000.00          6          5

Total price of the Sold Properties   -> 20650000.00 TL
Total price of the Rented Properties -> 90501.25 TL
```

Properties.txt

```
8
111 R 12000.50 2 1
222 S 2500000.00 3 2
333 R 38000.75 4 2
444 S 3200000.00 3 3
555 R 19500.00 1 1
666 S 8450000.00 5 4
777 R 21000.00 3 2
888 S 6500000.00 6 5
```

Q2. Write a C program that reads several appliance records consisting of **appliance ID**, **power consumption** (in watts), and **hours** used per day from the text file “**appliances.txt**” into a **dynamically** allocated structure array. The first line of the file indicates the number of appliances in the file.

The program should display the contents of the array, calculating the daily energy consumption (in kWh) for each appliance. Then, the user can enter an appliance ID to check if it exists in the structure array. If the ID exists, the program should display the corresponding appliance details. Otherwise, it should display an error message. The program stops when a negative id is given.

Write the following functions:

- **`readFromFile`**: takes the file pointer, dynamically allocated appliance structure array, and the number of appliances as parameters. The function reads the appliance information from the file into the array and returns the actual number of appliances.

  `display`: takes the appliance array and the number of appliances as parameters and displays the contents of the array by calculating the daily energy consumption (kWh) for each appliance. *1 kilowatt (kWh) = 1000 watts (Wh).*

$$Daily\ Energy\ (kWh)\ = \frac{Power\ (Wh)\ \times\ Hours\ Used\ Per\ Day}{1000}$$

- **`searchAppliance`**: takes the appliance array, the number of appliances, and the appliance ID to be searched in the array as parameters. It searches for the appliance in an array and, if the ID exists, returns the index of the appliance. Otherwise, it returns -1.

Project Name: LG5_Q2
File Name: Q2.cpp

Example Run#1:

```
ID   Cons (W)    Hours    Daily Energy (kWh)
--------------------------------------------
101     1500     3.50     5.25
102      800     5.00     4.00
103     1200     2.50     3.00
104      600     6.00     3.60
105     2000     1.50     3.00
106     1000     4.00     4.00
107      750     5.50     4.13

Enter the id of an appliance : 103

Id        :    103
Power Cons:   1200
Hours     :   2.50

Enter the id of an appliance : 587
Appliance NOT found!!

Enter the id of an appliance : 107

Id        :    107
Power Cons:    750
Hours     :   5.50

Enter the id of an appliance : 102

Id        :    102
Power Cons:    800
Hours     :   5.00

Enter the id of an appliance : -8
```

Appliances.txt

```
7
101 1500 3.5
102 800 5
103 1200 2.5
104 600 6
105 2000 1.5
106 1000 4
107 750 5.5
```

Q3. Write a C program that manages product warranties. Each product has an **ID**, a **warranty period** (in months), and **warranty details,** including the warranty start month, warranty start year, warranty end month, and warranty end year.

The text file **“warranty.txt”** keeps the number of products on the first line. The following lines contain the product info; product ID, warranty period in months, warranty start month, and warranty start year.

Write a C program that reads the product warranty information from the text file into a **dynamically** created NESTED structure array, displays the products’ information including their warranty status (active / expired), and finds and displays the product with the longest remaining warranty.

Write the following functions:

- **`readInfo`**: takes a file pointer, a product array, and the number of products as parameters, reads the warranty details for several products from the file into the array. For each product, the function should find and store the warranty end month and warranty end year. (*e.g. warranty period:* ***18****, warranty start month:* ***5****, warranty start year:* ***2023, then the*** *warranty start month:* ***11****, warranty start year:* ***2025***)
- **`display:`** `takes` a product array and the number of products as parameters and displays the information of all products including their warranty status (active / expired), by checking whether each product is still under warranty based on the current month and year.
- **`findLongestWarranty:`** takes a product array and the number of products as parameters, finds and displays the product with the longest remaining warranty.

Project Name: LG5_Q3
File Name: Q3.cpp

Example Run:

```
Enter current month (1-12): 2
Enter current year: 2025

Warranty Status:
ID      Period   Start    End      Status
---------------------------------------
 1001      18    05/2023  11/2024  Expired
 1002      12    03/2023  03/2024  Expired
 1003      36    07/2024  07/2027  Active
 1004      18    08/2024  02/2026  Active
 1005      48    01/2023  01/2027  Active
 1006      24    11/2021  11/2023  Expired
 1007      12    09/2024  09/2025  Active
 1008      36    03/2022  03/2025  Active
 1009      18    06/2024  12/2025  Active
 1010      24    04/2023  04/2025  Active

Product with Longest Warranty Remaining: ID 1003 with 29 months left.
```

warranty.txt

```
10
1001 18 5 2023
1002 12 3 2023
1003 36 7 2024
1004 18 8 2024
1005 48 1 2023
1006 24 11 2021
1007 12 9 2024
1008 36 3 2022
1009 18 6 2024
1010 24 4 2023
```

Additional Questions

AQ1.

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 :** takes 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 :** calculates the overall applicants' grades' average and the overall grade of each applicant with loads of English proficiency being 30%, a jury being 50%, and the graduate exam being 20%)
- **display :** displays the content of the structure array of **applicantsOfII_t** type.
- **findPassFail :** 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 the 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: LG05_AQ1
File Name: AQ1.cpp

applicants.txt

```
1111 49 65 94
2222 79 58 45
3333 76 88 95
```

Example Run :

```
Applicant ID: 1111

Scores:
Applicant English Proficiency: 49
Applicant Jury: 65
Applicant Graduate Examination: 94
Applicant Overall: 66.0

Applicant ID: 2222

Scores:
Applicant English Proficiency: 79
Applicant Jury: 58
Applicant Graduate Examination: 45
Applicant Overall: 61.7

Applicant ID: 3333

Scores:
Applicant English Proficiency: 76
Applicant Jury: 88
Applicant Graduate Examination: 95
Applicant Overall: 85.8

Average is 71.2
Number of the applicants who pass is 1
Number of the applicants who fail is 2
```

AQ2.

The weekly temperature for **15** cities is kept in the **weather.txt** file.

Write a C program that reads the **number of cities(*n*)** from the user, and the information for ***n cities*** from the text file into the **dynamically** created **two-dimensional array**. The program displays the temperatures on the screen, then finds and displays the coldest city with its temperature and the day of the week such as MONDAY, TUESDAY, ... (See the example run).

Write the following functions;

- **`readFromFile:`** takes the file pointer, a two-dim array (dynamically created), and the number of cities (***n***) as parameters, and reads weekly temperatures for **n** cities from the file into the array.
- **`display:`** takes a two-dim array and the number of cities(***n***) as parameters, and displays the weekly temperatures on the screen.
- **`findColdestDay:`** takes a two-dimensional array and the number of cities(***n***) as parameters, finds and returns indexes of the coldest city and the day of the week.

Project Name: LG5_QAQ2
File Name: AQ2.cpp

Example Run#1:

```
How many city will be displayed in the report? 25
How many city will be displayed in the report? 8

City   M   T   W   R   F   S   S
----  --- --- --- --- --- --- ---
   1    3  13   6  20   4  18   5
   2    1   4   1   8   4   3   5
   3   13  14   6   5   3   7   8
   4   19   8   9  14   8   3  -3
   5    8   2   4   6  -7  -1   0
   6    9  10   5  -2   0   2   4
   7    7   6  10   0  -1   4  -2
   8    1  -5  -4  -3  -2   1  -4

City 5 is the coldest one with the temperature -7 on FRIDAY
```

weather.txt

```
3 13 6 20 4 18 5
1 4 1 8 4 3 5
13 14 6 5 3 7 8
19 8 9 14 8 3 -3
8 2 4 6 -7 -1 0
9 10 5 -2 0 2 4
7 6 10 0 -1 4 -2
1 -5 -4 -3 -2 1 -4
19 15 10 9 4 2 0
20 19 16 12 11 14 16
13 10 14 13 4 -3 -4
6 -8 3 5 6 7 6
19 9 8 10 1 2 8
8 9 10 12 15 17 16
17 18 16 8 10 11 12
```

Example Run#2:

```
How many city will be displayed in the report? -1

How many city will be displayed in the report? 18
How many city will be displayed in the report? 13

City   M   T   W   R   F   S   S
----  --- --- --- --- --- --- ---
   1    3  13   6  20   4  18   5
   2    1   4   1   8   4   3   5
   3   13  14   6   5   3   7   8
   4   19   8   9  14   8   3  -3
   5    8   2   4   6  -7  -1   0
   6    9  10   5  -2   0   2   4
   7    7   6  10   0  -1   4  -2
   8    1  -5  -4  -3  -2   1  -4
   9   19  15  10   9   4   2   0
  10   20  19  16  12  11  14  16
  11   13  10  14  13   4  -3  -4
  12    6  -8   3   5   6   7   6
  13   19   9   8  10   1   2   8

City 12 is the coldest one with the temperature -8 on TUESDAY
```