Department of Information Systems and Technologies

CTIS 152 – Algorithms and Data Structure

FALL 2024 - 2025

Lab Guide #1 – Week 2-1

OBJECTIVE : General Review of 151 Subjects

Instructor: Burcu LİMAN
Assistants : Berk ÖNDER - Engin Zafer KIRAÇBEDEL

Q1. Write a C program that will take positive numbers as input parameters, then use these numbers to call the swap function using the "call by reference method". Your program must check for the validity of the given input and then it must give an error message if the numbers are not positive numbers.

Example Run:
```
Enter first number: -999
First number must be positive.
Enter first number: 2
Enter second number: -5
Second number must be positive.
Enter second number: 99
Before swap: a = 2, b = 99
After swap: a = 99, b = 2
```

Project Name: LG01_Q1
File Name: Q1.cpp

Q2. Write a C program that will generate n (will be given by user) random numbers **(between 101 - 666)**, then the program writes each generated number and the digits of that number in reverse order into the file **reverse.txt**.

Example Run:
```
How many numbers will you generate: 11
11 numbers and their digits in reverse order were written to reverse.txt file

Reverse.txt
```

```
Generated number   Digits in reverse order
****************   ***********************
             335    5    3    3
             309    9    0    3
             473    3    7    4
             423    3    2    4
             625    5    2    6
             106    6    0    1
             397    7    9    3
             246    6    4    2
             306    6    0    3
             264    4    6    2
             516    6    1    5
```

Project Name: LG01_Q2
File Name: Q2.cpp

GENERATION OF RANDOM NUMBERS:

1. Use stdlib.h (for srand function)
2. Use time.h (for time function).
3. srand(time(0)); for getting different number every time you run the program.
4. For getting a random number between 0 – 50: num = rand() % 51;
5. Apply debug process to check the random number.

Example program:
```
#include <stdio.h>
#include <stdlib.h> //for srand funtion
#include <time.h>  //for time function

int main(void)
{
   int num;

    /* we use srand function to be able to get a random number but we cannot use the srand function on
   its own, so we also use time function in it to give a start point to the srand function; because time
   is different every time you run the program, the random number will be different also */

   srand(time(NULL));

    /* because time returns a very big number it returns the millisecond value of the hour, so we want
   to get a random number between 0 and 99, we get the modulus 100 of the rand      function */
   num = rand() % 100;

  /* to create a number between a range*/
  //num = rand() % ((Max+1)-Min) + Min
   printf("The random number is: %d", num);

   return 0;
}
```

```
Example Run #1:
The random number is: 99

Example Run #2:
The random number is: 26
```

Q3. Write a C program that will take positive numbers and stores them in an integer array until zero is entered (zero will stop inputting). Write the following functions;

- **`fillArray:`** takes an array as a parameter, reads the positive numbers from the user and stores them in the array, and returns the actual number of elements.
- **`findEvenAverage:`** takes an array as a parameter, and finds and returns the even numbers' average.

```
Example Run:
Enter a positive number:  -6
cannot calculate, please enter positive number

Example Run:

Enter a positive number:  5
Enter a positive number:  7
Enter a positive number:  -1
No even numbers in the array

Example Run:
Enter a positive number:  2
Enter a positive number:  1
Enter a positive number:  6
Enter a positive number:  23
Enter a positive number:  45
Enter a positive number:  8
Enter a positive number:  4
Enter a positive number:  -1
Average of even numbers: 5.00
```

Project Name: LG01_Q3
File Name: Q3.cpp

Q4. Write a C program that reads IDs and game scores of several bowling teams from the file **"bowling.txt";** finds and displays the average of each game and the average of each team using the functions above. See the example run.

Write the following functions;

- **`readFromFile`:** takes a file pointer, a one-dim array to keep the team Ids and a two-dimensional array to keep the game scores as parameter. The function reads the team IDs into the one-dim array and 4 game scores of several bowling teams into the two-dim array from the specified file. The function also returns the number of teams.
- **`findTeamAvg`:** takes the two-dim scores array and the number of team as input parameters, finds the average of each team and stores the averages into a one-dim array.
- **`findGameAvg`** takes the two-dim scores array and the number of team as input parameters, finds the average of each game and stores the averages into a one-dim array.
- **`displayGameAvg`:** takes the one-dim array which keeps the game averages as input parameter and displays the averages of all games on the screen.

Project Name: LG01_Q4
File Name: Q4.cpp

Example Run:

```
Team Number  Average
***********  *******
 12          483.50
 24          436.25
 33          505.25
 45          470.00
 57          517.50
 68          449.00
 79          444.25
 89          500.00
 96          484.00
 98          455.50

Game Number  Average
***********  *******
1            475.7
2            482.1
3            496.0
4            444.3
```

bowling.txt

```
12 482 570 500 382
24 350 395 575 425
33 475 482 552 512
45 552 545 418 365
57 660 385 475 550
68 446 520 345 485
79 273 582 498 424
89 445 510 570 475
96 624 347 465 500
98 450 485 562 325
```

ADDITIONAL QUESTION

Write a program that reads numbers from a text file into a two dimensional integer array. There are 5 columns in each row, while the number of rows is unknown. If the matrix is square, the program will calculate the sum of the elements on the minor diagonal, otherwise it will calculate the product of elements in the given row.

Write the following functions;

- **`display`**: Display all elements in the matrix.
- **`sumOfMinor`**: Find the sum of the elements on the minor diagonal of the matrix.
- **`productOfRow`**: Find the product of the elements on a specified row.

Project Name: LG01_AQ
File Name: AQ.cpp

input1.txt

```
2 52 4 7 8
3 36 95 47 48
26 12 25 41 85
15 36 45 73 5
48 7 11 98 12
79 35 64 72 19
36 44 21 36 15
45 77 5 25 4
14 4  6 78 23
78 36 12 3 98
25 89 7 12 54
95 26 36 85 74
42 78 69 42 1
```

Example Run with input1.txt:

```
 2  52   4   7   8
 3  36  95  47  48
26  12  25  41  85
15  36  45  73   5
48   7  11  98  12
79  35  64  72  19
36  44  21  36  15
45  77   5  25   4
14   4   6  78  23
78  36  12   3  98
25  89   7  12  54
95  26  36  85  74
42  78  69  42   1

The matrix is not a SQUARE matrix
Enter the row number: 1
The product of the elements on the given row: 23296
```

input2.txt:

```
2 52 4 7 8
3 36 95 47 48
26 12 25 41 85
15 36 45 73 5
48 7 11 98 12

```

Example Run with input2.txt:

```
 2  52   4   7   8
 3  36  95  47  48
26  12  25  41  85
15  36  45  73   5
48   7  11  98  12

The matrix is a SQUARE matrix
The sum of the elements on the Minor Diagonal is: 164
```

Debugger Short Keys

Display documentation for the active window.	F1
Display a system menu for the application window.	ALT+SPACEBAR
Add and remove breakpoints on the current lines.	**F9**
Clear all breakpoints.	CTRL+SHIFT+F9
Disable breakpoint.	CTRL+F9
Display Breakpoints dialog box.	CTRL+B
Adds a watch on the currently selected word.	SHIFT+F9
End debugging session.	**SHIFT+F5**
Execute code one statement at a time, following execution into function calls (Step Into).	**F11**
Execute the next line of code but without following execution through any function calls (Step Over).	**F10**
Execute the remaining lines of a function in which the current execution point lies (Step Out).	SHIFT+F11
Restart a debugging session.	CTRL+SHIFT+F5
Resume execution of your code from the current statement to the selected statement (Run to Cursor).	CTRL+ F10
Run the application to the next break point.	**F5**
Set the next statement.	CTRL+SHIFT+ F10
Stop execution (Break).	CTRL+BREAK
Open watch window #x.	**CTRL+ALT+W, #{1, 2, 3}**
Open locals' window.	**CTRL+ALT+V, L**