Lab No.0 Revision of Structured Programming

0.1 Introduction

Computer Programming course in previous semester is based on writing set of instruction in a particular sequence such that the output is based on the sequence of written statements. If this sequence of instructions is modified then it could have a large impact on the resulting program. Often a situation arises where we intend to use a particular chuck of code again and again in a single program. In this scenario using procedural programming, the programmer is forced to copy-paste the code in various locations of the source file.

Although this is a solution, it is not practical one because it causes the code to become length, unmanaged, poorly structured, and difficult to understand. In such a situation, function handling offers an attractive and easy to implement alternative that is widely accepted and promoted by programmers of all languages.

This is pre-lab for the object-oriented programming course on the use of functions, pointers, structures, and building logic. The functions that are designed will receive variables and arrays from the user. The function will perform some processing on the provided data and return the results back to the main program. The importance of this pre-lab can be highlighted through the fact that functions play critical role in establishment of classes, objects and their implementation. A large portion of object-oriented programming is based on manipulating variable using functions. Also, in this pre-lab, the use of use of pointes has been emphasized. Concepts relating to pointers, memory access, using arrays with pointer have been covered.

0.2 Objectives of the lab

Reinstate the concepts of structured programming such as

1. decision making
2. iterations
3. functions
4. structures and arrays
5. pointers

0.3 Activities

0.3.1 Activity

Write a C++ program that calculates the average of a set of numbers entered by the user. Follow the steps below:

1. Implement a function named **calculateAverage ()** that takes two arguments: an integer array and its size. This function should calculate and return the average of the numbers in the array.
2. In the **main ()** function, ask the user to input the number of values they want to average.
3. Allocate an integer array of the specified size in the **main ()** function.
4. Ask the user to input the individual values one by one into the array.
5. Call the **calculateAverage ()** function to calculate the average of the values in the array.
6. Display the calculated average to the user.

Ensure that the program does not use pointer-notation or create a temporary array for calculating the average.

0.3.2 Activity

Write a program that creates **Student** structure. It consists of three data members: NAME (character array of size 50), AGE (integer variable), and CGPA (floating-point variable). It consists of no-argument constructor **Student ()**. It consists of three-argument constructor **Student (char n [], int a, float c)** to initialize the three data members. It consists of **show ()** function to display the three data members. In main, create two Student variables: s1 and s2. To store data in s1, use three-argument constructor. To store data in s2, access the three Student data member directly. Once initialized, display the contents of s1 and s2 using **show ()** function.

0.3.3 Activity

Write a C++ program that generates random data and calculates the sum of the generated data. Follow the steps below:

1. In the **main ()** function, ask the user to input the length of the array.
2. Declare an integer array of the specified length.

3. Use the **rand ()** function to randomly generate integer data within the range of 0 to 100 and store it in the array using pointer-notation. You can use a *for* loop for this purpose. *Hint*: use *#include <cstdlib>* for **rand ()** function.
4. Display the generated data using pointer-notation.
5. Calculate and display the sum of the generated data using pointer-notation.

0.3.4 Activity

Write a C++ program to find transpose of a NxN matrix. Declare integer matrix and its transpose using **calloc ()** *(use #include <cstdlib>)***.** Use 2D pointers in this activity for storage, retrieval, and finding transpose. Free the pointer variables in the end.

0.4 Testing

Test Cases for Activity 1

Sample Inputs	Sample Outputs
Enter the number of values you want to average: 2 Enter 2 integer values, one by one: Value 1: 4 Value 2: 5 Enter the number of values you want to average: 3 Enter 3 integer values, one by one: Value 1: 5 Value 2: 9 Value 3: 4	The average of the entered values is: 4.5 The average of the entered values is: 6

Test Cases for Activity 2

Sample Inputs	Sample Outputs
Student 1 Name: Ali Khan Age: 22 CGPA: 3.12 Student 2 Name: Ramiz Shah Age: 21 CGPA: 3.5	Student 1 details Student 2 details

Test Cases for Activity 3

Sample Inputs	Sample Outputs
Enter the length of the array: 5 Generated data using pointer-notation: 35 34 57 62 66	Sum of the generated data: 254

Enter the length of the array: 3 Generated data using pointer-notation: 46 11 28	Sum of the generated data: 85

Test Cases for Activity 4

Sample Inputs	Sample Outputs
Enter the size of the square matrix (NxN): 2 Matrix Contents: 5 9 4 3 Enter the size of the square matrix (NxN): 3 Matrix Contents: 1 2 3 4 5 6 7 8 9	 Transpose: 5 4 9 3 Transpose: 1 4 7 2 5 8 3 6 9

0.5 Tools

Code: Blocks or Dev-C++ or Eclipse

0.6 References

1 **Object-Oriented Programming in C++ by *Robert Lafore***

2 **How to Program C++ by *Deitel & Deitel***