| ***Computer Engineering Department*** |
| --- |
| ***CE100L: Computing Fundamentals & Programming*** |

| ***Course Instructor: Usama Bin Shakeel*** | ***Dated: 22/11/2021*** |
| --- | --- |
| ***Teaching Assistant: Aqsa Khalid*** | ***Semester: Fall 2021*** |
| ***Lab Engineer: Nadir Abbas*** | ***Batch: BSCE2021*** |

# **Lab 9A. Pointers**

| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| --- | --- | --- | --- | --- |
| Nimra Maqbool | BSCE21012 |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this session is to learn the working and advantages of pointers and recursion in C++.

## **Equipment and Component**

| **Component Description** | **Value** | **Quantity** |
| --- | --- | --- |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

## **Theory and Background**

The variable that is used to hold the memory address of another variable is called a pointer variable or simply a pointer. The data type of the variable (whose address a pointer is to hold) and the pointer variable must be the same.

A pointer variable is declared by placing a asterisk (\*) after data type or before variable name in data type statement. E.g. if pointer variable “p” is to hold memory address of an integer variable it is declared as:

**int \*p;**

or to hold address of a float type variable we can declare as:

**float \*rep;**

**Lab Task**

1. Write a C++ code to print a string character by character using pointers.

**Input**: Enter string: Pakistan

**Output:** Pakistan

| void printCharacter() {  char ch[100]; //initialize a char array  char \*ptr; //initialize an pointer  cout << "please enter the string:" << endl;  cin.get(ch, 100); //enter an array of string having size 100  cout << "enter string is:" << endl;  ptr = ch; //store the ch in ptr  while (\*ptr != '\0') //condition of \*ptr is not equal to \0  {  cout << \*ptr; //display ptr value  ptr++; //iterating  }  cout << endl;  } |
| --- |

1. Write a C++ code to get size and array elements from the user and print the array and sum of array elements using pointers.

**Input:**

How many numbers do you want to add? : 3

Enter 1st element: 5

Enter 2nd element: 6

Enter 2nd element: 10

**Output:**

Elements of array are: [5 6 10 ]

Sum of all elements of array using Pointers as Parameters: 21

| void sumOfElements() {  int num;  cout << "please enter how many numbers u want to add" << endl; //enter the size as num  cin >> num;  int \*arr = new int[num]; //given a new memory  int sum = 0; //initialize sum  cout << "please enter " << num << " number" << endl;  for (int i = 0; i < num; i++) //loop for input of array  {  cout << "enter " << i + 1 << " number";  cin >> arr[i];  }  cout << "elements of array are :{"; //loop for displaying array  for (int i = 0; i < num; i++) {  cout << arr[i];  }  cout << "}" << endl;  int \*ptr;  ptr = &sum;  for (int j = 0; j < num; j++) { //loop for finding sum  sum = \*ptr + arr[j];  }  cout << "sum of elements is " << sum; //displaying sum  delete[]arr; //deleting the memory taken  } |
| --- |

1. Write a C++ code to swap two values using pointers through a function named “SwapValues”.

**void swapValues ( float \*, float \* );**

Before swapping x = 10.8 and y = 20

After swapping x = 20 and y = 10.8

| void swapValue (float \* numa ,float \*numb)  {  float \*ptr = new float; //giving a new memory  \*ptr = \*numa; //storing the value of ptr in numa  \*numa = \*numb; //storing the value of numa in numb  \*numb = \*ptr; //storing numb in ptr  cout<<"x="<<\*numa<<endl; //displaying numa  cout<<"Y="<<\*numb<<endl; //displaying numb  delete ptr; //deleting the memory taken  } |
| --- |

#### **Assessment Rubric for Lab**

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

| Performance metric | Mapping (task no. and description) | | Max marks | Exceeds expectation | Meets expectation | Does not meet expectation | Obtained marks |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1. Realization of experiment (a) | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 2 | Viva | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | Code commenting | 5 | Observes lab safety rules; handles the equipment and parts with care and adheres to the lab disciplinary guidelines aptly (4-5) | Generally observes safety rules and disciplinary guidelines with minor lapses (2-3) | Disregards lab safety and disciplinary rules (0-1) |  |
| 5. Data collection (c) | 1 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | Documentation | 5 | Timely documented (4-5) | Late documented (2-3) | Not documented (0-1) |  |
|  | Max Marks (total): | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_