Faculty of Computing



Data Structure and Algorithms Lab Manual

# Objective

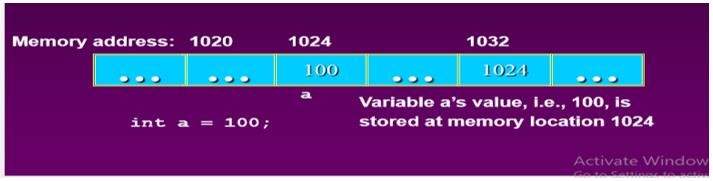
**Lab 2: Pointers**

* Memory addresses
* Pointers
* Types of pointers
* Pointers Examples
* Practice Task on pointers

# Introduction

## Computer Memory

Each variable is assigned a memory slot (the size depends on the data type) and the variable’s data is stored there



## Pointers:

Pointers are a type of variable that allow you to specify the address of a variable. They provide a convenient means of passing arguments to functions and for referring to more complex data types such as structures. You need to declare and initialize pointers just as you would other variables, but there are special operators that you need to use.

## Activity Time boxing

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Activity Name** | **Activity time** | **Total Time** |
|  | Lab Manual Lecture | 15 mins |  |
|  | Examples | 15 mins |  |
|  | Walkthrough Tasks | 15 mins |  |
|  | Practice Tasks | 100 mins |  |
|  | Tasks Evaluation/Viva | 35 mins | 180 minutes |

**Lab Manual Lecture [Expected time = 15 minutes]**

# Concept Map

## Basic Pointer Revision

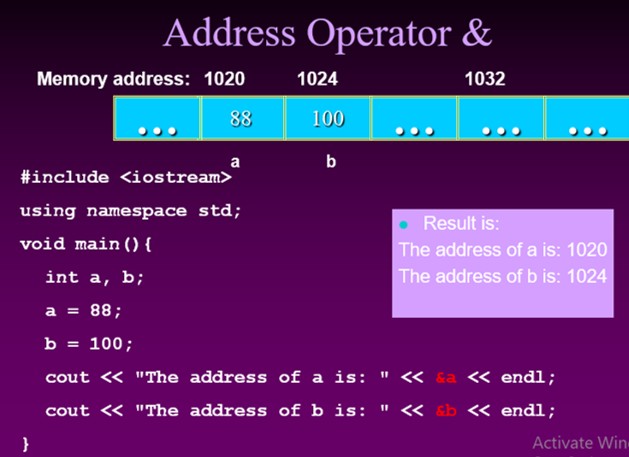
* **Declaration of Pointer variables**

type\* pointer\_name;

//or

type \*pointer\_name

**Address Operator (&):** The "address of " operator (&) gives the memory address of the variable.



**The \* Operator Dereference:** The star operator (\*) dereferences a pointer. The \* is a unary operator which goes to the left of the pointer it dereferences. The pointer must have a pointee, or it's a runtime error.

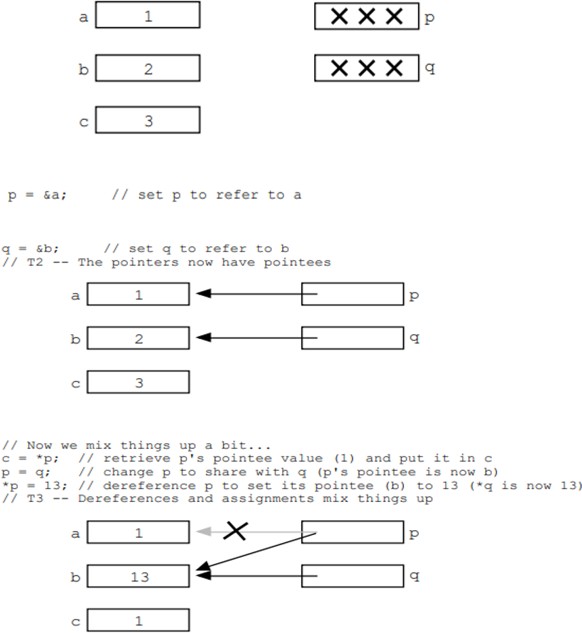
**Test Program:**

void PointerTest() {

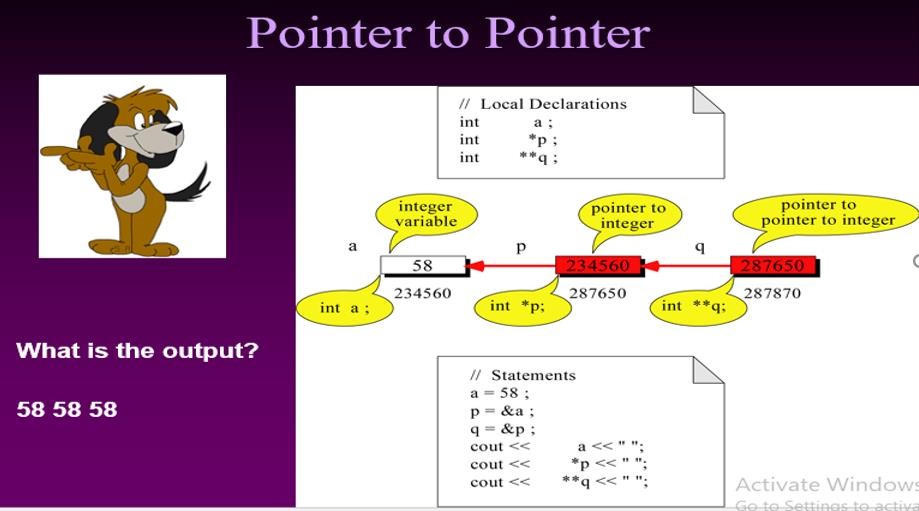
// allocate three integers and two pointers int a = 1; int b = 2; int c = 3; int\* p; int\* q;

// Here is the state of memory at this point.

// T1 -- Notice that the pointers start out bad...



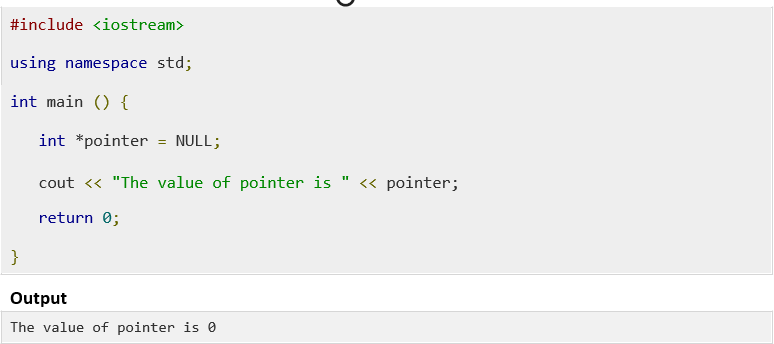
**Pointer to Pointer**



**Types of Pointers**

## Null Pointer

A pointer should be set to zero when it is not assigned to a valid address. Such a pointer is called a null pointer. Doing this will allow you to check whether the pointer can be safely dereferenced, because a valid pointer will never be zero.



## Void Pointer

A void pointer is a pointer that has no associated data type with it. A void pointer can hold address of any type and can be type casted to any type.

int a = 10; char b = 'x';

void \*p = &a; // void pointer holds address of int 'a' p = &b; // void pointer holds address of char 'b'.

## How to increment pointer address and pointer’s value?

When we are accessing the value of a variable through pointer, sometimes we just need to increment or decrement the value of variable though it or we may need to move the pointer to next int position (just like we did above while working with arrays). The ++ operator is used for this purpose. One of the examples of ++ operator we have seen above where we traversed the array using pointer by incrementing the pointer value using ++ operator.

## Pointer Arithmetic

pointer is an address which is a numeric value; therefore, you can perform arithmetic operations on a pointer just as you can a numeric value. There are four arithmetic operators that can be used on pointers: ++, --, +, and -

To understand pointer arithmetic, let us consider that ptr is an integer pointer which points to the address 1000. Assuming 32-bit integers, let us perform the following arithmetic operation on the pointer −

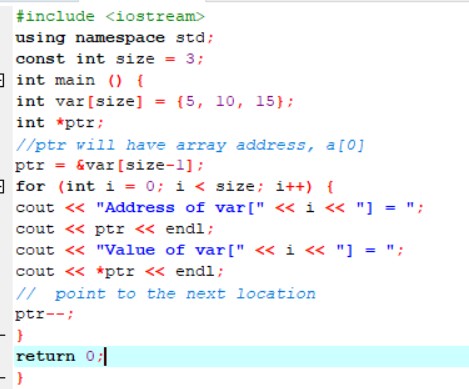
ptr++

the ptr will point to the location 1004 because each time ptr is incremented, it will point to the next integer. This operation will move the pointer to next memory location without impacting actual value at the memory location. If ptr points to a character whose address is 1000, then above operation will point to the location 1001 because next character will be available at 1001.

# Sample Program

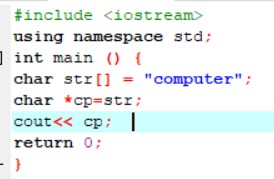
## Incrementing Pointer

**Decrementing Pointer**



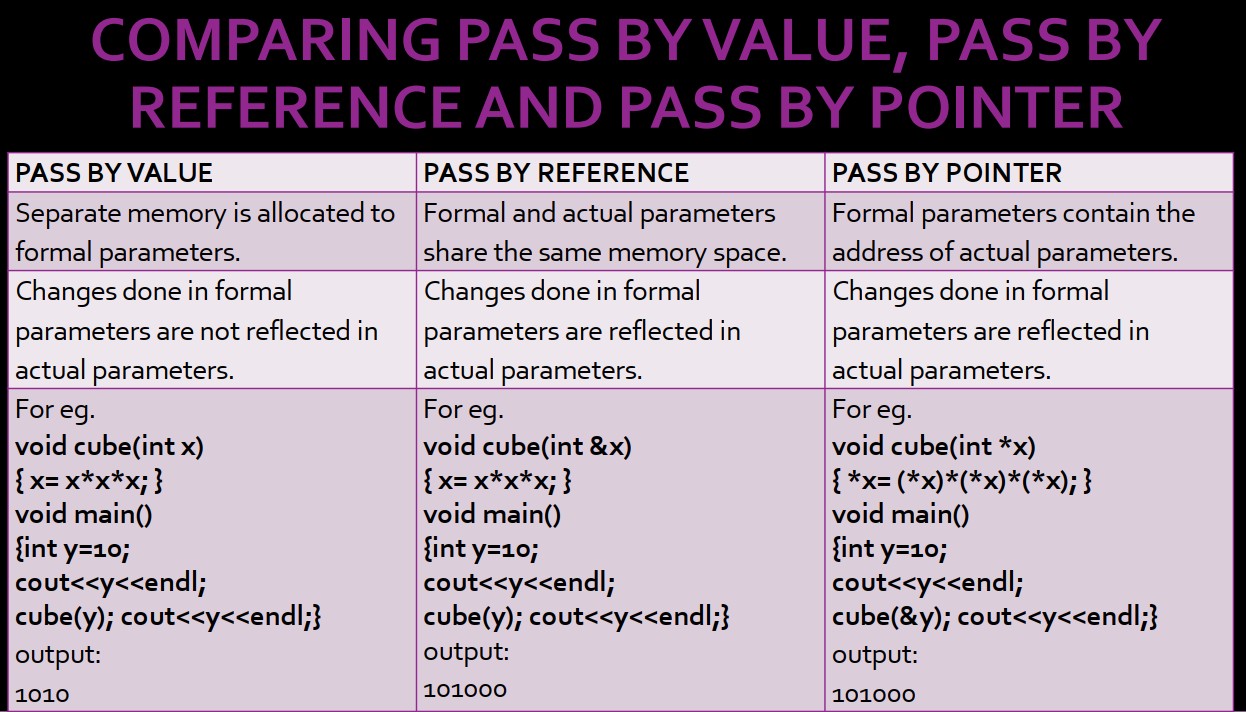
## Pointers as Strings

A string is a sequence of characters. A string type variable is declared in the same manner as an array type variable is declared. This is because the string is an array of character type variables.



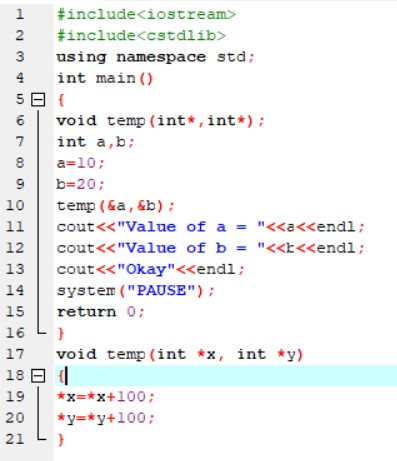
## Passing Pointers as Arguments to Functions:

The pointer variables can also be passed to functions as arguments. When a pointer variable is passed to a function the address of the variable is passed to the function. Thus a variable is passed to a function not by its value but by its reference.

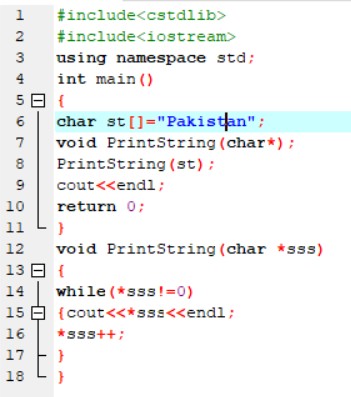


## Sample Program

### Write a program to pass two parameters to the function to add a constant value of 100 to the passed values using pointers.



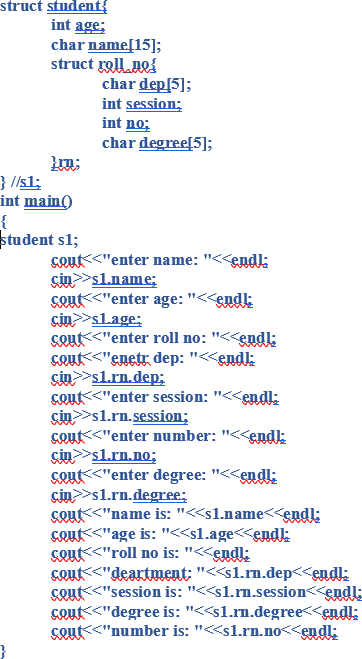
In the above program the function “temp” has two parameters which are pointers and are of int type. When the function “temp” is called the addresses of the variables “a” and “b” are passed to the function. In the function a value 100 is added to both variables “a” and “b” through their pointers. That is the previous values of variables “a” and “b” are increased by 100. When the control returns to the program the value of variable a is 110 and that of variable b is 120.



## Pointers with Structure:

An Arrow operator in C/C++ allows to access elements in Structures. It is used with a pointer variable pointing to a structure. The arrow operator is formed by using a minus sign, followed by the greater than symbol (->)

* The Dot(.) operator is used to normally access members of a structure.
* The Arrow(->) operator exists to access the members of the structure using pointers.



**(s1)->name, (s1)->rn.dep**

# Further Reading

## Books Slides

The slides and reading material can be accessed from the folder of the class instructor available at Moellim.

**Task:**

Practice all these tasks using C++ and submit on moellum.

# Evaluation Criteria

This lab is not evaluated.

**Outcomes:**

The outcomes of this lab were:

1. Students learn Pointers in C++
2. Learn and implement Pointers operations on array, strings etc.
3. Understand pointers types in C++