



**FEU INSTITUTE OF TECHNOLOGY**

**COLLEGE OF COMPUTER STUDIES**

**CCS007L**

**(COMPUTER PROGRAMMING 2)**

**EXERCISE**

---

**5**

**POINTERS**

<b>Student Name / Group Name:</b>		
<b>Members (if Group):</b>	<b>Name</b>	<b>Role</b>
<b>Section:</b>		
<b>Professor:</b>		

## **I. PROGRAM OUTCOME/S (PO) ADDRESSED BY THE LABORATORY EXERCISE**

- Design, implement and evaluate computer-based systems or applications to meet desired needs and requirements. [PO: C]

## **II. COURSE LEARNING OUTCOME/S (CLO)ADDRESSED BY THE LABORATORY EXERCISE**

- Apply the fundamental principles of handling pointers and memory allocation, and structures using C++in solving computing activities [CLO: 2]

## **III. INTENDED LEARNING OUTCOME/S (ILO) OF THE LABORATORY EXERCISE** At the end of this exercise, students must be able to:

- Create a program that uses pointers
- Create a program that dynamic arrays

## **IV. BACKGROUND INFORMATION**

A **pointer** is a variable whose value is the address of another variable. Like any variable or constant, you must declare a pointer before you can work with it. The general form of a pointer variable declaration is:

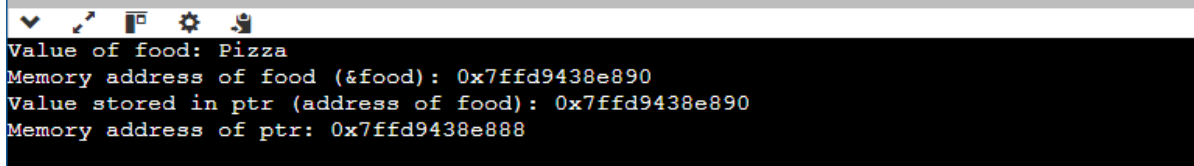
Type \*var-name;

Here, type is the pointer's base type; it must be a valid C++ type and var-name is the name of the pointer variable. The asterisk you used to declare a pointer is the same asterisk that you use for multiplication. However, in this statement the asterisk is being used to designate a variable as a pointer. The following are valid pointer declarations:

```
int  *ip;           // pointer to an integer
double *dp;        // pointer to double
float *fp;         // pointer to a float
char  *ch          // pointer to character
```

The actual data type of the value of all pointers, whether integer, float, character, or otherwise, is the same, a long hexadecimal number that represents a memory address. The only difference between pointers of different data types is the data type of the variable or constant that the pointer points to. Simply, a **pointer** is a variable that **stores the memory address as its value**. A pointer variable points to a data type of the same type, and is created with the `*` operator. The address of the variable you're working with is assigned to the pointer:

```
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 int main() {
7     // Create a string variable and assign a value
8     string food = "Pizza";
9
10    // Create a pointer to store the address of food
11    string* ptr = &food;
12
13    // Display the value of the variable food
14    cout << "Value of food: " << food << endl;
15
16    // Display the memory address of food
17    cout << "Memory address of food (&food): " << &food << endl;
18
19    // Display the value stored in the pointer (address of food)
20    cout << "Value stored in ptr (address of food): " << ptr << endl;
21
22    // Display the memory address of the pointer itself
23    cout << "Memory address of ptr: " << &ptr << endl;
24
25    return 0;
26 }
27
```



```
Value of food: Pizza
Memory address of food (&food): 0x7ffd9438e890
Value stored in ptr (address of food): 0x7ffd9438e890
Memory address of ptr: 0x7ffd9438e888
```

Create a pointer variable with the name `ptr`, that **points to** a `string` variable, by using the asterisk sign `*` (`string* ptr`). Note that the type of the pointer has to match the type of the variable you're working with. Use the `&` operator to store the memory address of the variable called `food`, and assign it to the pointer. Now, `ptr` holds the value of `food`'s memory address.

**V. GRADING SYSTEM/ RUBRIC**

<b>Trait</b>	<b>(Excellent)</b>	<b>(Good)</b>	<b>(Fair)</b>	<b>(Poor)</b>
<b>Requirement Specification (30pts)</b>	Able to identify correctly all input and output and provide alternative. <b>(28-20pts)</b>	Able to identify correctly all input and output <b>(25-17pts)</b>	Able to identify only one input or output <b>(22-14pts)</b>	Unable to identify any input and output <b>(20-11pts)</b>
<b>Data type(20pts)</b>	Able to apply required data type or data structure and produce correct results <b>(18-20pts)</b>	Able to apply required data type or data structure and produce partially correct results <b>(15-17pts)</b>	Able to identify required data type or data structure but does not apply correctly <b>(12-14pts)</b>	Unable to identify required data type <b>(9-11pts)</b>
<b>Input Validation(20pts)</b>	The program works and meets all specifications. Does exception al checking for errors and out-of-range data <b>(18-20pts)</b>	The program works and meets all specifications. Does some check for errors and out of range data <b>(15-17pts)</b>	The program produces correct results but does not display correctly Does not check for errors and out of range data <b>(12-14pts)</b>	The program produces s incorrect results <b>(9-11pts)</b>
<b>Free from syntax, logic, and runtime errors (10pts)</b>	Unable to run program <b>(10pts)</b>	Able to run program but have logic error <b>(8-9pts)</b>	Able to run program correctly without any logic error and display inappropriate output <b>(6-7pts)</b>	Able to run program correctly without any logic error and display appropriate output <b>(5pts)</b>
<b>Delivery (10pts)</b>	The program was delivered on time <b>(10pts)</b>	The program was delivered after 5 minutes from the time required. <b>(8-9pts)</b>	The program was delivered after 10 minutes from the time required. <b>(6-7pts)</b>	The program was delivered after 15 (or more) minutes from the time required. <b>(5pts)</b>
<b>Use of Comments (10pts)</b>	Specific purpose is noted for each function, control structure, input requirements, and output results. <b>(10pts)</b>	Specific purpose is noted for each function and control structure. <b>(8-9pts)</b>	Purpose is noted for each function. <b>(6-7pts)</b>	No comments included. <b>(5pts)</b>

**VI. LABORATORY ACTIVITY INSTRUCTIONS:**

Copy your source codes to be pasted in this document as well as a screen shot of your running output.

### ACTIVITY 5.1: Understanding pointers

Create a C++ program that:

1. Declares an integer variable num and assigns it a value (e.g., 10).
2. Creates an integer pointer ptr that stores the address of num.
3. Modifies the value of num indirectly via the pointer ptr.
4. Prints:
  - The value of num before and after modification.
  - The memory address of num using &num.
  - The value stored in ptr (address of num).
  - The value of num accessed through the pointer \*ptr.

Sample output:

Initial value of num: 10

Memory address of num (&num): 0x7ffee6b76c78

Value stored in ptr (address of num): 0x7ffee6b76c78

Value accessed via ptr (\*ptr): 10

After modifying value through ptr:

Updated value of num: 20

Value accessed via ptr (\*ptr): 20

### ACTIVITY 5.2: Dynamic Arrays

Write a C++ program that:

1. Dynamically allocates an array of integers using pointers.
2. Asks the user for the size of the array.
3. Fills the array with values entered by the user.
4. Displays the array's elements and their memory addresses.
5. Frees the allocated memory using delete [].

Sample Input:

Enter the size of the array: 5

Enter 5 integers:

10

20

30

40

50

Sample output:

```
Array elements and their memory addresses:
arr[0] = 10 (Address: 0x625dffa63ad0)
arr[1] = 20 (Address: 0x625dffa63ad4)
arr[2] = 30 (Address: 0x625dffa63ad8)
arr[3] = 40 (Address: 0x625dffa63adc)
arr[4] = 50 (Address: 0x625dffa63ae0)
```

## VII. QUESTION AND ANSWER

How does a pointer work in C++ to reference the memory location of a variable? Explain with an example that includes declaring a pointer, assigning it the address of a variable, and dereferencing it to access or modify the variable's value.

How can you traverse an array using pointer arithmetic instead of regular indexing? Write a code snippet to demonstrate iterating over an integer array and printing its elements using pointer arithmetic?

How is memory allocated dynamically for an array in C++ using the new keyword? Describe the process with an example program that dynamically creates an array of user-defined size, fills it with values, and prints the elements.

How do you ensure proper memory management when using dynamic arrays in C++? Provide an example of a program that dynamically allocates memory for an array and then deallocates it using delete[]. Explain what happens if the delete[] step is skipped.

How can pointers be used to modify the values of variables or arrays inside a function? Write a program where a function accepts a pointer to an integer and doubles its value. Similarly, demonstrate passing a dynamically allocated array to a function and printing its elements

## VIII. REFERENCES

- Zak, Dianne (2016). An Introduction to Programming with C++
- Deitel, Paul & Deitel, Harvey (2012). C++ How to Program, Eighth Edition
- <https://www.cprogramming.com/tutorial/lesson6.html>