LAB NO. 2

Structures & Pointers Using C++

OBJECTIVE:

- To Revise the concepts of structures & pointers using C++.
- To Implement structures and pointers in C++.

STRUCTURES:

Structure is a collection of variables of different data types under a single name. It is similar to a class in that, both holds a collection of data of different data types. For example: You want to store some information about a **person**: his/her name, citizenship number and salary. You can easily create different variables name, ***citNo***, ***salary*** to store this information separately. However, in the future, you would want to store information about multiple persons. Now, you'd need to create different variables for each information per person: ***name1, citNo1, salary1, name2, citNo2, salary2***.

You can easily visualize how big and messy the code would look. Also, since no relation between the variables (information) would exist, it's going to be a daunting task.

A better approach will be to have a collection of all related information under a single name Person and use it for every person. Now, the code looks much cleaner, readable and efficient as well. This collection of all related information under a single name Person is a structure.

```
struct Person
{
   char name[50];    int age;     float salary;
};
```

Here a structure person is defined which has three members: name, age and salary.

The members of structure variable are accessed using a dot (.) operator. Suppose you want to access age of structure variable bill and assign it 50 to it. You can perform this task by using following code below:

```
bill.age = 50;
```

Example: C++ Program to assign data to members of a structure variable and display it.

```
#include <iostream>
using namespace std;

struct Person
{
   char name[50];     int age;     float salary;
};

int main()
{
   Person p1;

   cout << "Enter Full name: ";
   cin.get(p1.name, 50);
  cout << "Enter age: ";
  cin >> p1.age;
 cout << "Enter salary: ";
 cin >> p1.salary;

  cout << "\nDisplaying Information." << endl;
  cout << "Name: " << p1.name << endl;
 cout <<"Age: " << p1.age << endl;
 cout << "Salary: " << p1.salary;

   return 0;
}
```

Output:

```
Enter Full name: Muhammad Ali
Enter age: 24
Enter salary: 1024.4
```

```
Displaying Information.
Name: Muhammad Ali
Age: 24
Salary: 1024.4
```

Here a structure Person is declared which has three members name, age and salary. Inside main() function, a structure variable p1 is defined. Then, the user is asked to enter information and data entered by user is displayed.

POINTERS:

Pointers are used for:

- Accessing array elements.
- Passing arguments to a function when the function needs to modify the original argument.
- Passing arrays and strings to functions.
- Obtaining memory from the system.
- Creating data structures such as linked lists.

Pointers are much more commonly used in C++ (and C) than in many other languages (such as BASIC, Pascal, and certainly Java, which has no pointers).

The ideas behind pointers are not complicated. Here's the first key concept: ***Every byte in the computer's memory has an address.*** Addresses are numbers, just as they are for houses on a street. The numbers start at 0 and go up from there—1, 2, 3, and so on. If you have 1MB of memory, the highest address is ***1,048,575***.

Your program, when it is loaded into memory, occupies a certain range of these addresses. That means that every variable and every function in your program starts at a particular address.

The Address-of Operator &

You can find the address occupied by a variable by using the address-of operator &. Here's a short program, VARADDR, that demonstrates how to do this:

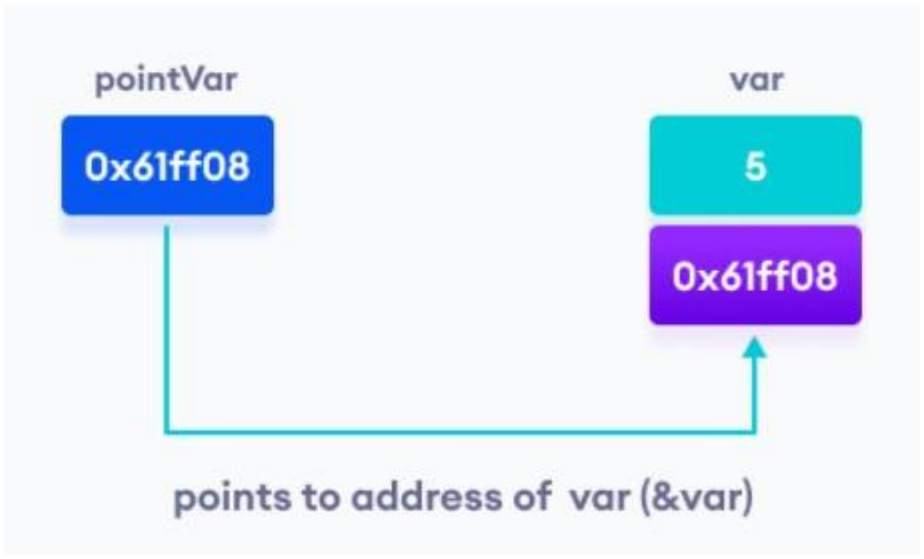
```
#include <iostream>
using namespace std;
int main()
{
int var1 = 11; //define and initialize three variables
int var2 = 22;
int var3 = 33;
cout<<&var1 << endl //print the addresses of these variables
cout<<&var2 << endl;
cout<<&var3 << endl;

return 0;
}
```

Pointers are the most powerful feature of C & C++. These are used to create and manipulate data structures such as linked lists, queues, stacks, trees etc. These are used in advance programming techniques. They provide means by which functions can modify their calling arguments.

Pointers are variables that hold address values. The use of pointers can improve the efficiency of the certain routines.

Example:

```
 #include <iostream>
using namespace std;
int main()
{

        int var = 5;
// declare pointer variable
        int* pointVar;

// store address of var
        pointVar = &var;

// print value of var
        cout << "var = " << var << endl;

// print address of var
        cout << "Address of var (&var) = " << &var<< endl;

// print pointer pointVar
        cout << "pointVar = " << pointVar << endl;

// print the content of the address pointVar points to
        cout << "Content of the address pointed to by pointVar (*pointVar) = " << *pointVar <<
endl;
   return 0;
}
```



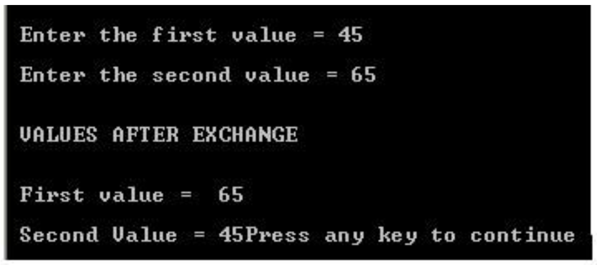
Example 3: Changing Value Pointed by Pointers

```
#include <iostream>
using namespace std;
int main() {
int var = 5;
int* pointVar;

pointVar = &var;                        // store address of var
cout << "var = " << var << endl;        // print var

cout << "*pointVar = " << *pointVar << endl     // print *pointVar
cout << "Changing value of var to 7:" << endl;
var = 7;                                // change value of var to 7
cout << "var = " << var << endl;        // print var
cout << "*pointVar = " << *pointVar << endl             // print *pointVar
cout << "Changing value of *pointVar to 16:" << endl;
*pointVar = 16;                         // change value of var to 16
cout << "var = " << var << endl;        // print var
cout << "*pointVar = " << *pointVar << endl;    // print *pointVar
return 0;
}
```

LAB TASKS:

Task 1:

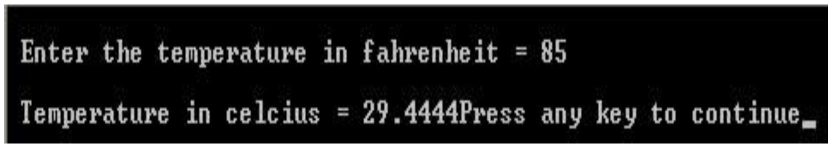
Write a program to swap two values by passing pointers as argument to the function. The output of the program should be like:



```
Enter the first value = 45
Enter the second value = 65

VALUES AFTER EXCHANGE

First value =  65
Second Value = 45Press any key to continue
```

Task 2:

Write a program to convert Fahrenheit temperature to Celsius degrees by passing pointers as arguments to the function. The formula for the conversion is: c = (f-32) *5 .0/9.0. The output of the program should be like:



```
Enter the temperature in fahrenheit = 85

Temperature in celcius = 29.4444Press any key to continue_
```

Task 3:

Create a structure(student) is which should contain name, roll and marks as its data member. Then, create a structure variable(s). Then take data (name, roll and marks) from user and store it in data members of structure variables. Display the data Entered by the user.