EXPERIMENT NO 11

Pointers in C++

Objectives:

In this lab students will learn:

- Memory concept of variables, pointers and how to use variable identifiers and pointers to refer to the variable.
- Pointer variable declarations and initialization.
- Direct and indirect referencing a variable using the pointer operators.
- Using * and address (&) operators.

Equipment required:

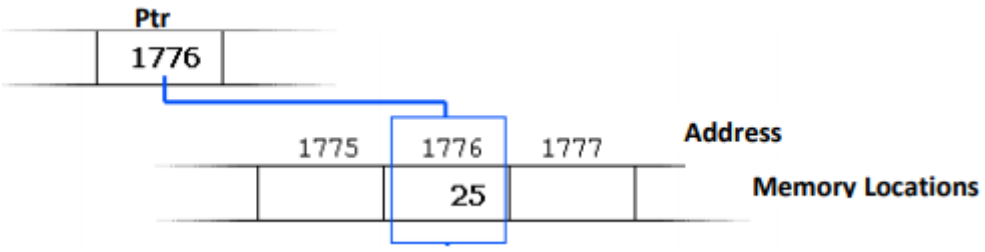
- Dev-C++/Eclipse/Visual Studio installed in PC/Windows

DISCUSSION

1. Pre-Lab

Pointers

Pointers are powerful features of C++ that differentiates it from other programming languages like Java and Python. Pointers are used in C++ to access the memory and manipulate the address. When declaring a variable, it is located at a specific location in memory, the memory address. The task of locating variables is automatically performed by the operating system during runtime. In some cases, we need to know the address where the variable is being stored during runtime. Variable which stores a reference to another variable is called a pointer. We can directly access the value stored in the variable using a pointer which points to it.



Address in C++

To know where the data is stored, C++ has an "&" operator. The & (reference) operator gives you the address occupied by a variable. If "var" is a variable then, "&var" gives the address of that variable.

Example

In the example given below, you may not get the same result on your system. The 0x in the beginning represents the address is in hexadecimal form. Notice that first address differs from second by 4-bytes and second address differs from third by 4-bytes. This is because the size of integer (variable of type int) is 4 bytes in 64-bit system.

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

int main()
{
     int var1 = 3;
     int var2 = 24;
     int var3 = 17;
     cout << &var1 << endl;
     cout << &var2 << endl;
     cout << &var3 << endl;
}
```

```
0x7fff5fbff8ac
0x7fff5fbff8a8
0x7fff5fbff8a4
```

Pointer Variables

C++ gives you the power to manipulate the data in the computer's memory directly. You can assign and de-assign any space in the memory as you wish. This is done using Pointer variables. Pointer variables are variables that points to a specific address in the memory pointed by another variable.

Syntax:

1. Pointer Declaration:

Syntax: Pointer_datatype *Pointer_name;

Example: int *Ptr1; //pointer of data type int, and name Ptr1

double *Ptr2; //pointer of data type double, and name Ptr2

2. Pointer initialization:

Syntax: Pointer_name=&variable_name;

Example: int *Ptr1, var; //variable named var and pointer named Ptr1 is initialized.

Ptr1=&var; Ptr1 holds the address of variable var.

Note: Pointer with a star "*ptr" can hold any value, while pointer without star "ptr" can hold address

Reference operator (&) and Deference operator (*)

Reference operator (&) as discussed above gives the address of a variable. To get the value stored in the memory address, we use the dereference operator (*). For example: If a "number" variable is stored in the memory address 0x123, and it contains a value 5. The reference (&number) gives value 0x123, while the dereference (*number) gives the value 5.

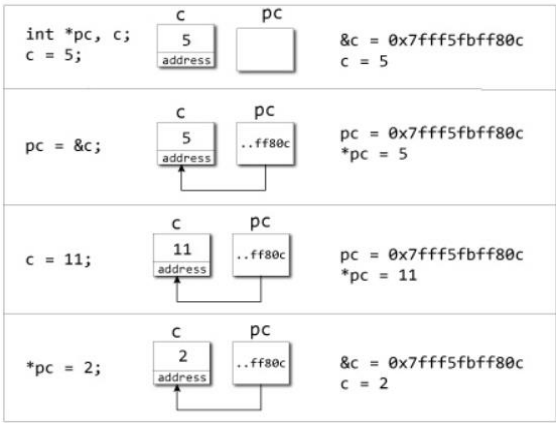
Example:

Following program demonstrate the working of pointer.

Example	OUTPUT

```
int main() {

int *pc, c;

    c = 5;

cout<<"Address of c (&c): "<<&c
<<endl;

cout<<"Value of c (c): "<< c
<<endl<<endl;


    pc = &c;    // Pointer pc holds
the memory address of variable c

cout<<"Address that pointer pc
holds(pc):"<<pc<<endl;

cout<<"Content of the address pointer
pc holds (*pc): "<< *pc <<endl<<endl;


c = 11;    // The content inside
memory address &c is changed from 5 to
11.

cout<<"Address pointer pc holds
(pc):"<<pc<<endl;

cout<<"Content of the address pointer
pc holds (*pc): "<< *pc <<endl<<endl;

    *pc = 2;

cout<<"Address of c (&c): "<<&c
<<endl;

cout<<"Value of c (c): "<< c
<<endl<<endl;

return0;}
```

```
;
Address of c (&c): 0x7fff5fbff80c
Value of c (c): 5

Address that pointer pc holds (pc): 0x7fff5fbff80c
Content of the address pointer pc holds (*pc): 5

Address pointer pc holds (pc): 0x7fff5fbff80c
Content of the address pointer pc holds (*pc): 11

Address of c (&c): 0x7fff5fbff80c
Value of c (c): 2
```



The “void” type pointer

void * p;

The pointer variable “p” can hold the memory address of variable of any data type

Pointers and Arrays

When an array is declared the name of array is the starting address of the array. In the example below the pointer will points to x[0] only.

int x[5];

int * p;

p=x;

To point to the complete array, we have to make an array of pointers.

A pointer to a pointer

A pointer to a pointer is a form of multiple indirections or a chain of pointers. Normally, a pointer contains the address of a variable. When we define a pointer to a pointer, the first pointer contains the address of the second pointer, which points to the location that contains the actual value as shown below.

A variable that is a pointer to a pointer must be declared as such. This is done by placing an additional asterisk in front of its name. For example, following is the declaration to declare a pointer to a pointer of type int as

```
int **var;  //second pointer of a first pointer
int ***var;  //third pointer of a second pointer
```

Common mistakes when working with pointers

Suppose, you want pointer "pc" to point to the address of c. Then,

1. int c, *pc; //variable and pointer declaration (correct)
2. pc=c; /* Wrong! Pc can hold the address, c is not an address. */
3. *pc=&c; /* Wrong! *pc can hold the value whereas, &c is an address. */
4. pc=&c; /* Correct! Pc can hold the address, &c is an address. */
5. *pc=c; /* Correct! *pc can hold the value whereas, c is a value too. */

2. Post-Lab (Lab Tasks)

1. Write a program to swap two numbers using pointers.
2. Write a program to find a maximum number among three numbers using pointers.
3. Declare one, two, and three-star pointers that point towards the same variable, and print all the possible addresses and values pointers are storing. Explain by writing comments. (Code discussed during lab and already shared on GCR too)

END