What are Pointers?

In C++, a pointer refers to a variable that holds the address of another variable. Like regular variables, pointers have a data type. For example, a pointer of type integer can hold the address of a variable of type integer. A pointer of character type can hold the address of a variable of character type.

Pointer Declaration

Syntax

The declaration of C++ takes the following syntax:

datatype *variable_name;

- The datatype is the base type of the pointer which must be a valid C++ data type.

- The variable_name is should be the name of the pointer variable.

- Asterisk used above for pointer declaration is similar to asterisk used to perform multiplication operation. It is the asterisk that marks the variable as a pointer.

Here is an example of valid pointer declarations in C++:

```
int *x; // a pointer to integer
double *x; // a pointer to double
float *x; // a pointer to float
char *ch // a pointer to a character
```

Reference operator (&) and Deference operator (*)

The reference operator (&) returns the variable’s address.

The dereference operator (*) helps us get the value that has been stored in a memory address.

For example:

If we have a variable given the name num, stored in the address 0x234 and storing

the value 28.

The reference operator (&) will return 0x234.

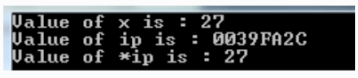
The dereference operator (*) will return 5.

Example 1:

```
#include <iostream>
using namespace std;
int main()
{    int x = 27;
     int *ip;
     ip = &x;
     cout << "Value of x is : ";
     cout << x << endl;
     cout << "Value of ip is : ";
     cout << ip<< endl;
     cout << "Value of *ip is : ";
     cout << *ip << endl;
```

```
    return 0;
}
```

Output:



1.3 Initializing Pointers via the Address-Of Operator (&)

When you declare a pointer variable, its content is not initialized. In other words, it contains an address of "somewhere", which is of course not a valid location. This is dangerous! You need to initialize a pointer by assigning it a valid address. This is normally done via the *address-of operator* (&).

The *address-of operator* (&) operates on a variable, and returns the address of the variable. For example, if number is

an int variable, &number returns the address of the variable number.

You can use the address-of operator to get the address of a variable, and assign the address to a pointer variable. For example,

```
int number = 88; // An int variable with a value
int * pNumber; // Declare a pointer variable called pNumber pointing to an int (or int pointer)
pNumber = &number; // Assign the address of the variable number to pointer pNumber
int * pAnother = &number; // Declare another int pointer and init to address of the variable number
```

Advantages of Pointers

- Pointers reduce the code and improve performance. They are used to retrieve strings, trees, arrays, structures, and functions.

- Pointers allow us to return multiple values from functions.

- In addition to this, pointers allow us to access a memory location in the computer’s memory.