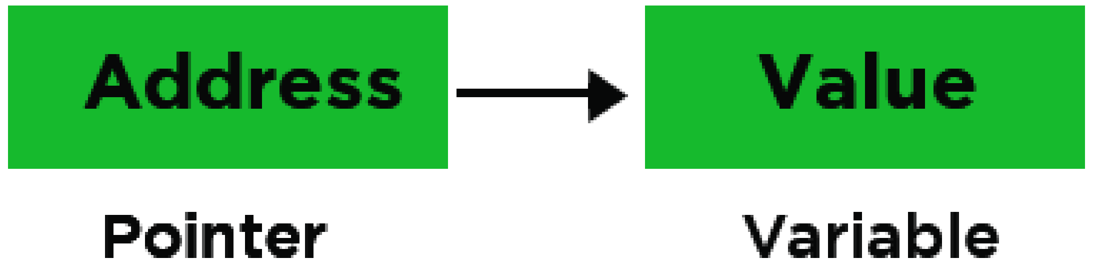
POINTER

PRESENTED BY-
NEELAM SINGH
DEPARTMENT OF COMPUTER SCIENCE

C++ Pointers

- The **pointers in C++** programming language is basically a variable that is also called as locater or installer that generally point towards the address of a provided value.



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.
- You should see a pointer as a symbolic representation of a memory address. With pointers, programs can simulate call-by-reference. They can also create and manipulate dynamic data structures. In C++, a pointer variable refers to a variable pointing to a specific address in a memory pointed by another variable.

Addresses in C++

- To understand C++ pointers, you must understand how computers store data.
- When you create a variable in your C++ program, it is assigned some space the computer memory. The value of this variable is stored in the assigned location.
- To know the location in the computer memory where the data is stored, C++ provides the **&** (reference) operator. The operator returns the address that a variable occupies.

For example, if x is a variable, &x returns the address of the variable.

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
```

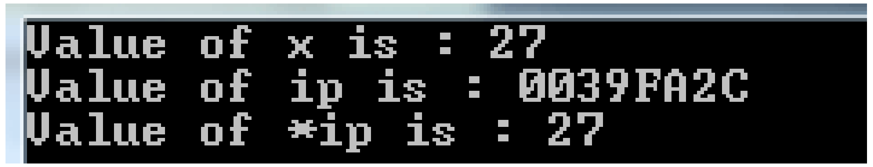
Reference operator (&) and Deference operator (*)

- 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

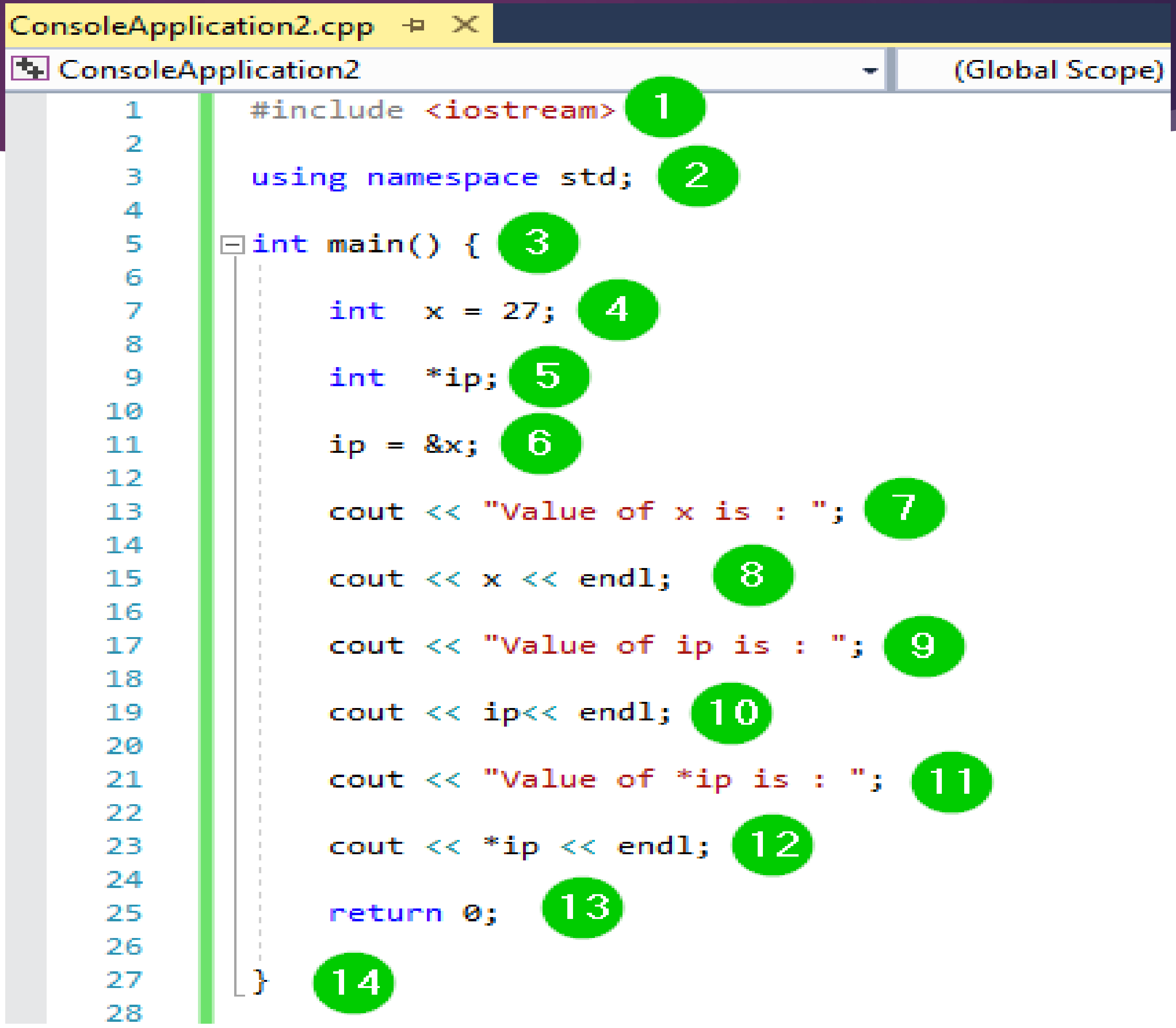
```
#include <iostream>
using namespace std;
int main()
 {
 int x = 27;
int *ip;
 ip = &x;
 cout << "Value of x is : ";
 cout << x << endl;
cout << "Value of ip is : ";
cout << ip<< endl;
cout << "Value of *ip is : ";
 cout << *ip << endl;
return 0; }
```

OUTPUT



```
Value of x is : 27
Value of ip is : 0039FA2C
Value of *ip is : 27
```

Here is a screenshot of the code:



```
#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;

}
```

Code Explanation:

- Import the iostream header file. This will allow us to use the functions defined in the header file without getting errors.
- Include the std namespace to use its classes without calling it.
- Call the main() function. The program logic should be added within the body of this function. The { marks the beginning of the function's body.
- Declare an integer variable x and assigning it a value of 27.
- Declare a pointer variable *ip.
- Store the address of variable x in the pointer variable.
- Print some text on the console.
- Print the value of variable x on the screen.
- Print some text on the console.
- Print the address of variable x. The value of the address was stored in the variable ip.
- Print some text on the console.
- Print value of stored at the address of the pointer.
- The program should return value upon successful execution.
- End of the body of the main() function.

Advantages of Pointer

- Less time in program execution.
- Working on the original variable.
- With the help of pointers, we can create data structures (linked-list, stack, queue).
- Returning more than one values from functions.
- Searching and sorting large data very easily.
- Dynamically memory allocation.

Uses of Pointers

- To pass arguments by reference.
- For accessing array elements.
- To return multiple values.
- Dynamic memory allocation.
- To implement data structures.
- To do system level programming where memory addresses are useful.

THANK YOU