# Pass-by-Reference with Pointers

* There are three ways in C++ to pass arguments to a function—pass-by-value, pass-by-reference with reference arguments and pass-by-reference with pointer arguments.
* Here, we explain pass-by-reference with pointer arguments.
* Pointers, like references, can be used to modify one or more variables in the caller or to pass pointers to large data objects to avoid the overhead of passing the objects by value.
* You can use pointers and the indirection operator (\*) to accomplish pass-by-reference.
* When calling a function with an argument that should be modified, the *address* of the argument is passed

## An Example of Pass-By-Value

* The following programs present two versions of a function that cubes an integer.

#include <iostream>

using namespace std;

int cubeByValue( int ); // Prototype

int main()

{

int number 5;

cout << “The original value of number is “ << number \n”;

number = cubeByValue( number );//Pass number by value to the function

cout << “The new value of number is “ << number \n”;

}

int cubeByValue( int n)

{

return n \* n \* n;

}

## Pass-by-Reference with Pointers

* The figure below passes the variable number to function cubeByReference using pass-by-reference with a pointer argument—the address of number is passed to the function.
* The function uses the dereferenced pointer to cube the value to which nPtr points.
* This *directly* changes the value of number in main

#include <iostream>

using namespace std;

int cubeByValue( int \* ); // Prototype

int main()

{

int number 5;

cout << “The original value of number is “ << number \n”;

number = cubeByValue( &number );//Pass address to the function

cout << “The new value of number is “ << number \n”;

}

int cubeByValue( int \*nPtr)

{

return nPtr \* nPtr \* nPtr;

}