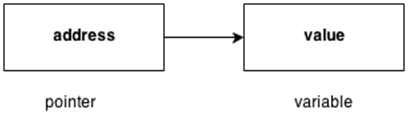
# C++ Pointers

The pointer in C++ language is a variable, it is also known as locator or indicator that points to an address of a value.



**Advantage of pointer**

1) Pointer reduces the code and improves the performance, it is used to retrieving strings, trees etc. and used with arrays, structures and functions.

2) We can return multiple values from function using pointer.

3) It makes you able to access any memory location in the computer's memory.

**Usage of pointer**

There are many usage of pointers in C++ language.

**1) Dynamic memory allocation**

In c language, we can dynamically allocate memory using malloc() and calloc() functions where pointer is used.

**2) Arrays, Functions and Structures**

Pointers in c language are widely used in arrays, functions and structures. It reduces the code and improves the performance.

## Symbols used in pointer

| **Symbol** | **Name** | **Description** |
| --- | --- | --- |
| & (ampersand sign) | Address operator | Determine the address of a variable. |
| ∗ (asterisk sign) | Indirection operator | Access the value of an address. |

## Declaring a pointer

The pointer in C++ language can be declared using ∗ (asterisk symbol).

**int** ∗ a; //pointer to int

**char** ∗ c; //pointer to char

Example

#include <iostream>

**using** **namespace** std;

**int** main()

{

**int** number=30;

**int** ∗ p;

p=&number;//stores the address of number variable

cout<<"Address of number variable is:"<<&number<<endl;

cout<<"Address of p variable is:"<<p<<endl;

cout<<"Value of p variable is:"<<\*p<<endl;

**return** 0;

}

## Pointer Program to swap 2 numbers without using 3rd variable

#include <iostream>

**using** **namespace** std;

**int** main()

{

**int** a=20,b=10,∗p1=&a,∗p2=&b;

cout<<"Before swap: ∗p1="<<∗p1<<" ∗p2="<<∗p2<<endl;

∗p1=∗p1+∗p2;

∗p2=∗p1-∗p2;

∗p1=∗p1-∗p2;

cout<<"After swap: ∗p1="<<∗p1<<" ∗p2="<<∗p2<<endl;

**return** 0;

}

There are 3 ways to pass C++ arguments to a function:

* call-by-value
* call-by-reference with pointer argument
* call-by-reference with reference argument

| // C++ program to illustrate call-by-methods in C++    #include <bits/stdc++.h>  using namespace std;  //Pass-by-Value  int square1(int n)  {  //Address of n in square1() is not the same as n1 in main()  cout << "address of n1 in square1(): " << &n << "\n";    // clone modified inside the function  n \*= n;  return n;  }  //Pass-by-Reference with Pointer Arguments  void square2(int \*n)  {  //Address of n in square2() is the same as n2 in main()  cout << "address of n2 in square2(): " << n << "\n";    // Explicit de-referencing to get the value pointed-to  \*n \*= \*n;  }  //Pass-by-Reference with Reference Arguments  void square3(int &n)  {  //Address of n in square3() is the same as n3 in main()  cout << "address of n3 in square3(): " << &n << "\n";    // Implicit de-referencing (without '\*')  n \*= n;  }  void geeks()  {  //Call-by-Value  int n1=8;  cout << "address of n1 in main(): " << &n1 << "\n";  cout << "Square of n1: " << square1(n1) << "\n";  cout << "No change in n1: " << n1 << "\n";    //Call-by-Reference with Pointer Arguments  int n2=8;  cout << "address of n2 in main(): " << &n2 << "\n";  square2(&n2);  cout << "Square of n2: " << n2 << "\n";  cout << "Change reflected in n2: " << n2 << "\n";    //Call-by-Reference with Reference Arguments  int n3=8;  cout << "address of n3 in main(): " << &n3 << "\n";  square3(n3);  cout << "Square of n3: " << n3 << "\n";  cout << "Change reflected in n3: " << n3 << "\n";      }  //Driver program  int main()  {  geeks();  } |
| --- |

// C++ program to illustrate Array Name as Pointers in C++

#include <bits/stdc++.h>

using namespace std;

void geeks()

{

//Declare an array

int val[3] = { 5, 10, 20 };

//declare pointer variable

int \*ptr;

//Assign the address of val[0] to ptr

// We can use ptr=&val[0];(both are same)

ptr = val ;

cout << "Elements of the array are: ";

cout << ptr[0] << " " << ptr[1] << " " << ptr[2];

}

//Driver program

int main()

{

geeks();

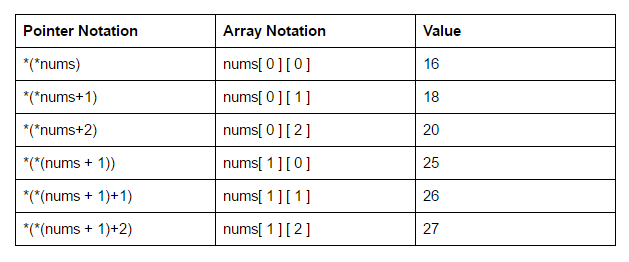
}

**Advanced Pointer Notation**

Consider pointer notation for the two-dimensional numeric arrays. consider the following declaration

int nums[2][3] = { { 16, 18, 20 }, { 25, 26, 27 } };

**In general, nums[ i ][ j ] is equivalent to \*(\*(nums+i)+j)**

****