Pointers

Pointer is the memory address of a variable.

Declaration :

<data\_type> \*p; 🡺data type of variable it is pointing to.

\* operator is called dereferencing operator.

<data\_type> \*p1, \*p2, v1, v2; 🡺p1 and p2 are pointers and v1 v2 are just variables.

The following will set the variable p1 equal to a pointer that points to the variable v1

p1 = &v1;

p1 = p2 🡺copies address from one pointer variable to another.

A pointer is an address and address is an integer, but a pointer is not an integer. A pointer is not a value of type int or of any other numeric type.

We can perform a kind of addition and a kind of subtraction on pointers, but they are not the usual integer addition or subtraction.

Example 1 :

int v1 = 0;

int \*p;

p = &v1;

\*p = 42;

cout<<v1<<endl<<\*p<<endl;

output :

42

42

New operator : new can be used to create variables that have no identifiers to serve as their names. For example the following creates a new variable of type int and sets the pointer variable p1 equal to address of this new variable.

Int \*p1;

p1 = new int; //initializes pointer to a nameless variable of type int

p1 = new int(17); //initializes the value stored in nameless variable to 17

\*p1 we can do anything with this nameless variable that you can do with any other variable of type int.

Variables that are created with new operator are called dynamic variables because they are created and destroyed while the program is running.

Example 2 :

int main( )

{

int \*p1, \*p2;

p1 = new int;

\*p1 = 42;

p2 = p1;

cout << "\*p1 == " << \*p1 << endl;

cout << "\*p2 == " << \*p2 << endl;

\*p2 = 53;

cout << "\*p1 == " << \*p1 << endl;

cout << "\*p2 == " << \*p2 << endl;

p1 = new int;

\*p1 = 88;

cout << "\*p1 == " << \*p1 << endl;

cout << "\*p2 == " << \*p2 << endl;

return 0;

}

Output :

\*p1 == 42

\*p2 == 42

\*p1 == 53

\*p2 == 53

\*p1 == 88

\*p2 == 53

Dynamic arrays : Arrays whose size is determined while the program is running, rather than being fixed when the program is written.

Int \*p = NULL; //to initialize to a null ptr

Int\* p;

P = new int [10];

Program to demonstrate that an array variable is a kind of pointer variable.

#include <iostream>

using namespace std;

typedef int\* IntPtr;

int main( )

{

IntPtr p;

int a[10];

int index;

for (index = 0; index < 10; index++)

a[index] = index;

p = a; //array address will be initialized to p; ptr = ptr

for (index = 0; index < 10; index++)

cout << p[index] << " ";

cout << endl;

for (index = 0; index < 10; index++)

p[index] = p[index] + 1;

for (index = 0; index < 10; index++)

cout << a[index] << " ";

cout << endl;

return 0;

}

Output:

0 1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9 10

Passing address from main function

Function declaration : function baz(int \*parm);

Main function call :

int arg;

baz(&arg);

Assumptions

\*p 🡪 content of where it is pointing to

p[0] 🡪 content of where pointer is pointing, similar to \*p

p[0]+1 🡪 content of where pointer is pointing[ ] + 1

\*(p+i) or p++ 🡪 content of(mem location + i)

If a is an array

a[2] , p[2], \*(a+2), \*(p+2) and (p+2 and then \*p) are all same

Given that p1 is an integer pointer variable, and a1 is an integer array, which of the following statements are not legal code?



a. p1= a1;



b. cout << p1[0];

**You Answered**



c. cin >> p1[0];

   correct answer u can never initialize a pointer to an array

d. a1 = p1;