SDF II(15B11CI211)

EVEN Semester 2021

2nd Semester , First Year

Jaypee Institute Of Information Technology (JIIT), Noida

Lecture 4 – Pointers arithmetic in C++

Pointers

- Pointers store address of variables or a memory location.
- // General syntax

datatype *var_name;

// An example pointer "ptr" that holds address of an integer variable or holds address of a memory whose value(s) can be accessed as integer values through "ptr"

int *ptr;

POINTER ARITHMETIC:

- pointer is an address which is a numeric value; therefore, you can perform arithmetic operations on a pointer just as you can a numeric value.
- There are four arithmetic operators that can be used on pointers:
  1. Increment Operator ++,
  2. Decrement Operator --,
  3. Addition +,
  4. Subtraction -

POINTER ARITHMETIC:

- To understand pointer arithmetic, let us consider that **ptr** is an integer pointer which points to the address 1000.
- Assuming 32-bit integers, let us perform the following arithmetic operation on the pointer—

ptr++

- The **ptr** will point to the location 1004 because each time ptr is incremented, it will point to the next integer.
- This operation will move the pointer to next memory location without impacting actual value at the memory location.
- If ptr points to a character whose address is 1000, then above operation will point to the location 1001 because next character will be available at 1001.

Incrementing a Pointer

- We prefer using a pointer in our program instead of an array because the variable pointer can be incremented, unlike the array name which cannot be incremented because it is a constant pointer.
- The following program increments the variable pointer to access each succeeding element of the array −

Example


```
#include <iostream>

using namespace std;
const int MAX = 3;

int main () {
   int  var[MAX] = {10, 100, 200};
   int  *ptr;

   // let us have array address in pointer.

   ptr = var;

for (int i = 0; i < MAX; i++) {
      cout << "Address of var[" << i << "] = ";
      cout << ptr << endl;

      cout << "Value of var[" << i << "] = ";
      cout << *ptr << endl;

      // point to the next location
      ptr++;
   }

   return 0;
}
```

Output:

- When the above code is compiled and executed, it produces result something as follows—

```
Address of var[0] = 0xbfa088b0
Value of var[0] = 10
Address of var[1] = 0xbfa088b4
Value of var[1] = 100
Address of var[2] = 0xbfa088b8
Value of var[2] = 200
```

Decrementing a Pointer

The same considerations apply to decrementing a pointer, which decreases its value by the number of bytes of its data type as shown below −

```
#include <iostream>

using namespace std;
const int MAX = 3;

int main () {
   int  var[MAX] = {10, 100, 200};
   int  *ptr;

   // let us have address of the last element in pointer.

   ptr = &var[MAX-1];

   for (int i = MAX; i > 0; i--) {
      cout << "Address of var[" << i << "] = ";
      cout << ptr << endl;

      cout << "Value of var[" << i << "] = ";
      cout << *ptr << endl;

      // point to the previous location

      ptr--;
   }

   return 0;
}
```

Output

- When the above code is compiled and executed, it produces result something as follows −

  Address of var[3] = 0xbfdb70f8

  Value of var[3] = 200

  Address of var[2] = 0xbfdb70f4

  Value of var[2] = 100

  Address of var[1] = 0xbfdb70f0

  Value of var[1] = 10

Pointer Comparisons

- Pointers may be compared by using relational operators, such as ==, <, and >.
- If p1 and p2 point to variables that are related to each other, such as elements of the same array, then p1 and p2 can be meaningfully compared.
- The following program modifies the previous example one by incrementing the variable pointer so long as the address to which it points is either less than or equal to the address of the last element of the array, which is &var[MAX - 1] −

Example


```
#include <iostream>

using namespace std;
const int MAX = 3;

int main () {
   int  var[MAX] = {10, 100, 200};
   int  *ptr;

   // let us have address of the first element in pointer.

   ptr = var;
   int i = 0;

   while ( ptr <= &var[MAX - 1] ) {
      cout << "Address of var[" << i << "] = ";
      cout << ptr << endl;

      cout << "Value of var[" << i << "] = ";
      cout << *ptr << endl;

      // point to the previous location

      ptr++;
      i++;
   }

   return 0;
}
```

Output—

- When the above code is compiled and executed, it produces result something as follows −

  Address of var[0] = 0xbfce42d0

  Value of var[0] = 10

  Address of var[1] = 0xbfce42d4

  Value of var[1] = 100

  Address of var[2] = 0xbfce42d8

  Value of var[2] = 200

References

- https://www.tutorialspoint.com/cplusplus/cpp_pointer_arithmatic.htm
- http://www.infobrother.com/Tutorial/C++/C++-Pointer-Arithmetic
- https://www.learncpp.com/cpp-tutorial/introduction-to-pointers/