Practical Lecture : Pointers

Quick Recap

Let's take a quick recap of previous lecture –

A) Friend Function

B) Friend Class

C) Call by value

D) Call by reference

E) Call by Address

F)Recurssion

Today's

Today we are going to cover -

● Pointers

● Difference b/w pointers and reference variables

● Void pointer

● Pointer to Pointer

Let's Get Started-

Pointers

Pointer is a variable in C++ that holds the address of another variable. They have data type just like variables, for example an integer type pointer can hold the address of an integer variable and an character type pointer can hold the address of char variable.

Syntax:-
data_type *pointer_name;

int *p, var

As I mentioned above, an integer type pointer can hold the address of another int variable. Here we have an integer variable var and pointer p holds the address of var. To assign the address of variable to pointer we use ampersand symbol (&).

p = &var;

Pointers

```cpp
#include <iostream>
using namespace std;
int main(){
    //Pointer declaration
    int *p, var=101;
    //Assignment
    p = &var;

    cout<<"Address of var: "<<&var<<endl;
    cout<<"Address of var: "<<p<<endl;
    cout<<"Address of p: "<<&p<<endl;
    cout<<"Value of var: "<<*p;
    return 0;
}
```

Output

Address of var: 0x7fff5dfffc0c
Address of var: 0x7fff5dfffc0c
Address of p: 0x7fff5dfffc10
Value of var: 101

Difference b/w pointers and reference variable

Sr. No.	Reference	Pointer
1	Reference is a temporary variable.	The pointer is a variable which stores the address of a variable.
2	A reference variable can be used directly to access the value.	Pointer variable requires an indirection operator to access the value of a variable.
3	It cannot be reassigned with different address values once it is assigned.	It can be reassigned to point to different objects.
4	A null value cannot be assigned to the reference variable.	A null value can be assigned to the reference variable.
5	It is necessary to initialize the variable at the time of declaration.	It is not necessary to initialize the variable at the time of declaration.

Void Pointers

A void pointer is a general-purpose pointer that can hold the address of any data type, but it is not associated with any data type.
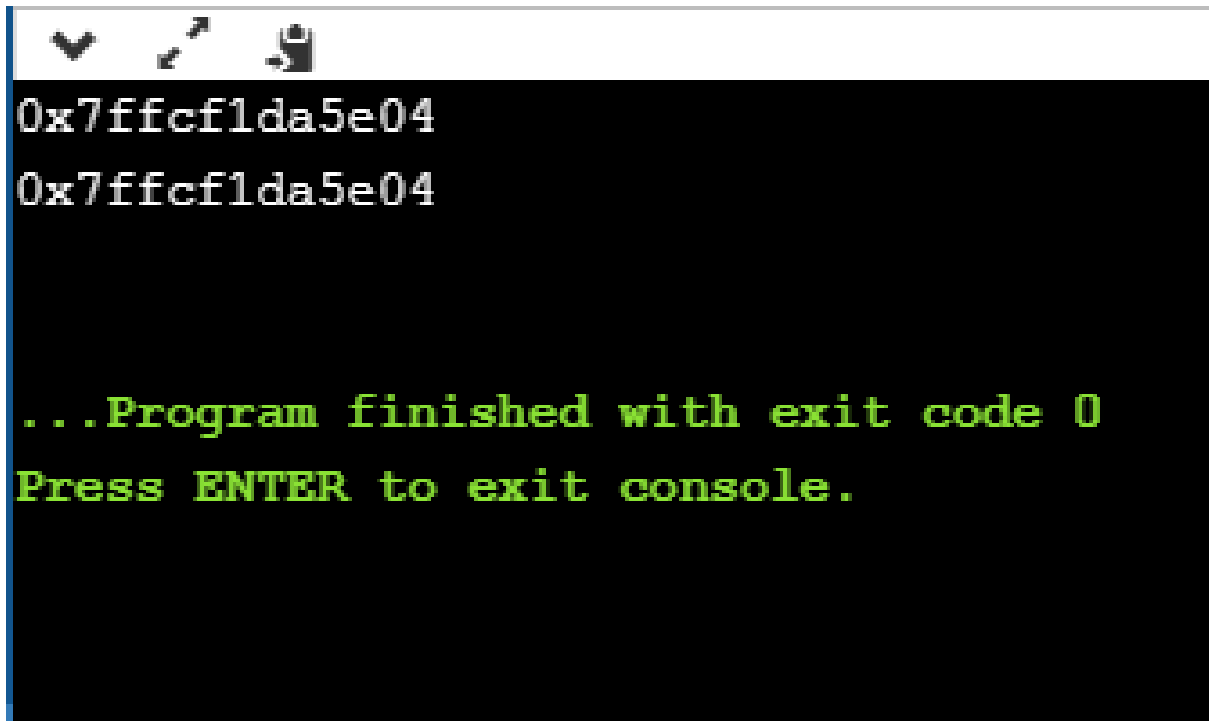
void *ptr;

Void Pointers

```cpp
#include <iostream>
using namespace std;
int main()

{
 void *ptr;   // void pointer declaration
 int a=9;   // integer variable initialization
 ptr=&a;   // storing the address of 'a' variable in a void pointer variable.
 std::cout << &a << std::endl;
 std::cout << ptr << std::endl;
 return 0;
}
```

Output

```
0x7ffcf1da5e04
0x7ffcf1da5e04



...Program finished with exit code 0
Press ENTER to exit console.
```

Arithmetic Pointer

A 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.

Incrementing a Pointer

```cpp
#include <iostream>
using namespace std;
const int MAX = 3;

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

Output

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

```cpp
#include <iostream>
using namespace std;
const int MAX = 3;

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

Decrementing a Pointer

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.

```cpp
#include <iostream>

using namespace std;
const int MAX = 3;
```

Pointer Comparisons

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

Output

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

Pointer to Pointer

We already know that a pointer points to a location in memory and thus used to store the address of variables. So, when we define a pointer to pointer. The first pointer is used to store the address of the variable. And the second pointer is used to store the address of the first pointer. That is why they are also known as double pointers.

Declaration:-

int **ptr;

Pointer to Pointer

```cpp
#include<iostream>
using namespace std;
int main()
{
    int var = 789;
    int *ptr2;
    int **ptr1;
    ptr2 = &var;
    ptr1 = &ptr2;
    cout<< var<<endl;
    cout<<*ptr2<<endl ;
    cout<<**ptr1;

}
```

Output

789
789
789

Any
Questions ??

Thank You!

See you guys in next class.