Lecture 11

C++ *this* Keyword

In C++, **this** keyword can be used to –

- Access the ***currently executing object*** of a class.
- Access the **data members** of the *currently executing object*.
- Calling the **member functions** associated with the *currently executing object*.
- To *resolve* the **shadowing issue**, when a *local variable has a same name as an instance variable*.

Friend functions do not have a **this** pointer, because friends are not members of a class. Only member functions have a **this** pointer.

- ### *Accessing the currently executing object using this keyword.*

We can access the currently executing object using **this** keyword, which in this case also known as *this reference*.

```
#include<iostream>
using namespace std;

class A
{
public:
void message()
{
cout<< "Hello from A" << "\n";
cout<< this;
}
};

int main()
{
A ob;
ob.message();
}
```

Output

```
Hello from A
0x28ff3f
```

Program Analysis

We have created an object of class A and have called function message() on this object. Because **this** refers to the *currently executing object*, hence by printing **this**, we have got the *hexadecimal representation of the object's address* in the memory.

- ## Accessing the data members through this keyword.

If we can access an object using **this** keyword, then we can also easily access data members associated with an object, using **this** keyword . Let's see how -

```
#include<iostream>

using namespace std;

class A
{
private:
int a=10;

public:

void message()
{
cout<< "Hello from A" << "\n";
cout<< this->a;
}
};


int main()
{
A ob;
ob.message();
}
```

Output -

```
Hello from A
10
```

Because **this** refers to the currently executing object, hence, using **this** keyword, we have accessed the value of a *data member* named **a**, associated with the *current executing object*.

- ## *Calling a member function using this keyword.*

As we can access an object of the class using **this** keyword, hence, using this object, we can even access the functions associated with it.

```
#include<iostream>

using namespace std;

class A
{
private:
int a=10;

public:
void message();
void hello();
};

//Definition of message() function of A class
void A :: message()
{
cout<< "Hello from A" << "\n";
this->hello();
}

//Definition of hello() function of A class
void A :: hello()
{
cout<< "Bonjour" << "\n";
cout<< "Hello" << "\n";
cout<< "Namaste" << "\n";
}


int main()
{
A ob;
ob.message();
}
```

Output is

Hello from A
Bonjour
Hello
Namaste

In the preceding code, we have called the function hello() of an object using **this** keyword.

- ## *Shadowing issue without using this keyword.*

Shadowing issue is caused when a local variable overshadows an instance variable with the same name.

```
#include<iostream>

using namespace std;

class A
{
public:
int a;                    //instance variable, a

void putValue(int);       //local variable, a
};


//Defintion of putValue() function
void A :: putValue(int a)
{
a=a;
}


int main()
{
A ob;

//passing 10 to putValue() function, for 10 to be stored in data member a
ob.putValue(10);

//accessing the value of data member, a
Cout<< "Value in a : " << ob.a;
}
```

Output is

Value in a : 4201003

Program Analysis

- We have created a class with a function *putValue()*, used to initialize data member, a.
- This function takes an int value in its local variable named **a**, similar to the name of data member **a**.
- By calling putValue(10) function, the value 10 is passed to local variable **a**.
- As the local variable of putValue() function and data member has the same name, a. Hence, a=a, puts 10 in the local variable **a** and data member **a** is not accessed at all.
- Hence, when you access *data member* **a**, a random int garbage value is displayed in the output, because **a** is not initialized.

  Note : This issue is also known as shadowing as local variable in a function has overshadowed data member, having the same name.

- ## Using this keyword to resolve the shadowing issue.

Using **this keyword**, we can access the currently executing object and its data members.

```
#include<iostream>

using namespace std;

class A
{
public:
int a;

void putValue(int);
};

//Defintion of putValue() function of A class
void A :: putValue(int a)
{

//accessing an data member with this keyword, to differ it from a local variable with same name
this->a = a;
```

```
}

int main()
{
A ob;
ob.putValue(10);
cout<< "Value in a : " << ob.a;
}
```

Output-

```
Value in a : 10
```

Here in the putValue() function, we have accessed an data member **a** with **this** keyword, in order to distinguish its name from a local variable **a**, which resolves the shadowing issue.

C++ this Pointer Example

Let's see the example of this keyword in C++ that refers to the fields of current class.

```
#include <iostream>
using namespace std;
class Employee {
  public:
     int id; //data member (also instance variable)
     string name; //data member(also instance variable)
     float salary;
     Employee(int id, string name, float salary)
     {
         this->id = id;
        this->name = name;
        this->salary = salary;
     }
     void display()
```

```
    {
        cout<<id<<"  "<<name<<"  "<<salary<<endl;
    }
};
int main(void) {
    Employee e1 =Employee(101, "Sonoo", 890000); //creating an object of Employee
    Employee e2=Employee(102, "Nakul", 59000); //creating an object of Employee
    e1.display();
    e2.display();
    return 0;
}
```

Output-

```
101  Sonoo  890000
102  Nakul  59000
```