Pointers to Class Members in C++

Just like pointers to normal variables and functions, we can have pointers to class member functions and member variables.

Let's see how this works.

Defining a Pointer of Class type

We can define pointer of class type, which can be used to point to class objects.

```
class Simple
{
    public:
    int a;
};

int main()
{
    Simple obj;
    Simple* ptr;   // Pointer of class type
    ptr = &obj;

    cout << obj.a;
    cout << ptr->a;  // Accessing member with pointer
}
```

Here you can see that we have declared a pointer of class type which points to class's object. We can access data members and member functions using pointer name with arrow `->` symbol.

Pointer to Data Members of Class

We can use pointer to point to class's data members (Member variables).

Syntax for Declaration :

```
datatype class_name :: *pointer_name;
```

Syntax for Assignment:

```
pointer_name = &class_name :: datamember_name;
```

Both declaration and assignment can be done in a single statement too.

```
datatype class_name::*pointer_name = &class_name::datamember_name ;
```

Using Pointers with Objects

For accessing normal data members we use the dot `.` operator with object and `->` qith pointer to object. But when we have a pointer to data member, we have to dereference that pointer to get what its pointing to, hence it becomes,

```
Object.*pointerToMember
```

and with pointer to object, it can be accessed by writing,

```
ObjectPointer->*pointerToMember
```

Lets take an example, to understand the complete concept.

```
class Data
{
    public:
    int a;
    void print()
    {
        cout << "a is "<< a;
    }
};

int main()
{
    Data d, *dp;
    dp = &d;     // pointer to object

    int Data::*ptr=&Data::a;   // pointer to data member 'a'

    d.*ptr=10;
    d.print();

    dp->*ptr=20;
    dp->print();
}
```

a is 10 a is 20

The syntax is very tough, hence they are only used under special circumstances.

Pointer to Member Functions of Class

Pointers can be used to point to class's Member functions.

Syntax:

```
return_type (class_name::*ptr_name) (argument_type) = &class_name::function_name;
```

Below is an example to show how we use ppointer to member functions.

```
class Data
{
    public:
    int f(float)
    {
        return 1;
    }
};

int (Data::*fp1) (float) = &Data::f;   // Declaration and assignment
int (Data::*fp2) (float);        // Only Declaration

int main(0
{
    fp2 = &Data::f;   // Assignment inside main()
}
```

Some Points to remember

1. You can change the value and behaviour of these pointers on runtime. That means, you can point it to other member function or member variable.
2. To have pointer to data member and member functions you need to make them public.

- ← Prev
- Next →