19CSE201 :Advanced Programming

Lecture 6
More on Objects, Methods and Classes

By
Ritwik M
Assistant Professor(SrGr)
Dept. Of Computer Science & Engg
Amrita Viswa Vidyapeetham - Coimbatore

Based on material from The Professor's C++Course by Linda W. Friedman and Stroustrup B. Programming: principles and practice using C++. Second edition, Addison Wesley; 2014.

A Quick Recap

- Classes
- Methods
- Messages
- Access Specifiers
- Class Attributes
- Static data members
- Accessors, Mutators and Auxiliary Functions

Objects

- An instance of a class
- Also known as a Class variable
- Can be uniquely identified by its name
- All objects have an identity, behaviour and state
  - The identity of the object is defined by its name
  - The behaviour of an object is represented by the functions which are defined in the objects class
    - These functions show the set of actions for every object
  - The state of the objects are reffered by the data stored within the object at any point in time.

Creating an object of a class

- Declaring a variable of a class type creates an object
  - Also known as Instantiation
- You can have multiple variables of the same class type
- Once an object of a class is instantiated, a new memory location is created for it to store its data members and code

Instantiation

```
int main(){
    test t1,t2;
    t1.setcode();
    t2.setcode();
    test t3;
    t3.setcode();
    t1.showcode();
    t2.showcode();
    t3.showcode();
    return 0;
}
```

Types of Objects

- External (global) Objects
  - Exist throughout the lifetime of the program and can be accessed anywhere within the code file (i.e. file scope)
- Automatic (local) objects
  - Persistant and visible only throughout the function/method in which they are created. (i.e. local scope)
- Static Objects
  - Persistant throughout the program, but visible only within local scope
- Dynamic Objects
  - Lifetime may be controlled within a specific/particular scope

Objects in Memory

```
class student{
    public:
        int rollNo;
        char name[20];
        int marks;
};
student s;
```

rollNo – 2 bytes
name – 20 bytes
marks – 2 bytes

Total 24 bytes

Extremely similar to the memory representation of structures, isn't it?

Array of objects

- An array of a "class" type variables is an array of objects
- Declaration
  - `className object[length]`
  - Example: `student section[63]`
  - We can use this array when calling a member function
  - `section[i].putdata()`
  - The array of such an object is stored as a multi-dimensional array

Passing Objects to Function as Arguments

- Pass by value
  - A copy of the object is passed to the function.
  - Any changes made to the object inside the function will not affect the original object used when calling the function

- Pass by Reference
  - The address of the object is passed to the function
  - Any changes made inside the function will affect the actual object.

Example – Pass by value

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

class complex{
float real,imagine;
public:
    void getData();
    void putData();
    void sum (complex A, complex B);
};

void complex :: getData(){
    cin>>real;
    cin>>imagine;
}
```

1

```
void complex :: putData(){
    if (imagine>=0){
        cout<<real<<"+"<<imagine<<"i";
    }
    else{
        cout<<real<<imagine<<"i";

    }
}
```

2

```
void complex :: sum(complex input1, complex input2){
    real = input1.real+input2.real;
    imagine = input1.imagine+input2.imagine;
}
```

3

```
int main() {
    complex X,Y,Z;
    X.getData();
    Y.getData();
    Z.sum(X,Y);
    Z.putData();
    return 0;
}
```

4

```
Input:
5
6
7
8
```

5

Returning an object

Exercise

Modify the previous program such that the SUM() function returns the value

Try to use the main() function given here in your modified program

```
int main() {
    complex X,Y,Z;
    X.getData();
    Y.getData();
    Z=X.sum(Y);
    Z.putData();
    return 0;
}
```

Solution

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

class complex{
float real,imagine;
public:
    void getData();
    void putData();
    complex sum (complex B);
};

void complex :: getData(){
    cin>>real;
    cin>>imagine;
}
```

```
void complex :: putData(){
    if (imagine>=0){
        cout<<real<<"+"<<imagine<<"i";
    }
    else{
        cout<<real<<imagine<<"i";

    }
}
```

```
complex complex :: sum(complex input2){
    complex temp;
    temp.real = real+input2.real;
    temp.imagine = imagine+input2.imagine;
    return temp;
}
```

```
int main() {
    complex X,Y,Z;
    X.getData();
    Y.getData();
    Z=X.sum(Y);
    Z.putData();
    return 0;
}
```

```
Input:
5
6
7
8
```

Up Next

Constructors and Destructors in C++