MIT World Peace University

Computer Networks
Second Year B.Tech Semister 3
Academic Year 2022-23

Module 1 - Class Notes

Notes

Prepared By

P34. Krishnaraj Thadesar

Batch A2

September 6, 2022

Contents

1 Things to do

1. Types of Inheritance
2. Virtual base Classs
3. Polymorphism
4. Vitual functions

2 Inheritance

- It is the mechanism by which one class acquires the properties of another class
- Provides a way to create a new class from an existing class
- The new class is a specialized fersion of the existing class
- Inheritance establishes an "is a" relationship or a parent child relationship between classes.
- Allows sharing off the behavior of the parent class into its child classes
- child class can add new behavior or override existing behaviour from parent
- It allows a hierarchy of classes to be built moving from the most general to the most specific class.

2.1 Differnece between overloading and overiding

- Overloading is when you write the same function many times within the same class
- Overriding is when you do that same thing, but in sub classes.

2.2 Benefits of using Inheritance

- Reusablity : Reuse the methods and data of the existing class
- Extendability: Extend the existing class by adding new data and new methods.
- Modifyability: Modify the existang class by overloading its methods with newer implementations, saves memory space, increases reliability, saves the developing process.

3 Class Derivation in C++

syntax: class DerivedClassName : specification BaseClassName
like class child : public parent() // private by default ;

4 Types of Inheritance

1. Single level Inheritance : You have 1 base class –> 1 Child class.
2. Multiple Inheritance : 2 or more Base Classes –> 1 Child Class
3. Multi-Level inheritance : 1 Base Class –> 1 Child Class –> Another Child Class and so on
4. Heirarchical Inheritance : 1 Base Class –> 2 or more Child Classes.
5. Hybrid Inhertiance : Any legal combination of any of these things.

4.1 What Access modifiers mean when inheriting

1. If you do class child : private parent; then every private data member becomes inaccessible, coz anyway thats what should happen, then the protected data members become private, and public data members also become private.
2. If you do class child : protected parent; then its the same thing, except you still cant access private variables, but protected and public data members become protected
3. Same with class child : public parent; everything remains unchanged. The objects will behave in accordance with the usual laws of objects.

4.2 Constructors and Destructors in Base and Derived classes

1. Derived classes can have their own constructors and destructors
2. When an object of a derived class is created, the base class's constructor is executed frist followed by the derived class's constructor is executed first, followed bt the derived class's constructor
3. In case of multiple inheritances, the base classes are constructed in the order in which they appear in the declaration of the derived class.
4. For destructors, the order is reversed.

5 Overriding Member Functions

- If a base and derived class have member functions with same name, and arguements then method is said to be overridden and it is caled as "function overriding" or "method overriding".
- The Child class provides alternative implementation for parent class method specific to a particular subclass type.
- You might need to do this if your child class has something to add to the previous definiton. You could still call it from that function.
- If you have multiple functions tho, you could have some ambiguity in your code, and to fix that you could the scope resolution operator.

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

class A
{

    public:
    void show()
    {
        cout << "Hello from A";
    }
};

class B
{
    public:
    void show()
    {
        cout << "Hello from B";
    }
};

class C : public A, public B
{

};

int main()
{
    C c;
    // c.show(); "C: show is ambigious"
    c.A::show(); // would be the syntax to call it.
    c.B::show(); // to call B
    // c.show() // if C had a show method.
    return 0;
}
```

6 Virtual Base Class

- In hybrid inheritance child class has two direct parents which themselves have a common base class.
- So you can prevent mumtiple copies of the base class coming into the child class by declaring the base class as virtual when its being inherited.
- So like imagine you have 2 base classes each inheriting the same class. Now imagine a third class that inherits from both of them. So the base, or the grandparent classes methods are copied twice. You can prevent this by declaring them as virtual base classes.

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


class Base
{
```

```
    public:
    int i;
};

class D1 : virtual public Base
{
    public:
    int j;
};


class D2 : virtual public Base
{
    public:
    int k;
};


class C : public D1, public

 D2
{

    public:
    int l;
    void product()
    {
        return i*j*k;
    }
};

int main()
{
    // usual driver code.
    return 0;
}
```

7 Inheritance in Java

- It is pretty Much similar to cpp

```
Syntax:
class derived_class extends base_class Name
{
// methods and stuff.
}
```

8 Operator Overloading

- Operator overloading is a feature in C++ Programming that allows programmer to redefine the meaning of an existing operator when they operator on class obects.
- It is the ability to tell the compiler thow to perform a cerain operation when its corresponding opeator is used on one or more ariables.

- Closely Related to function overloading.
- Allows existing operators to be redefined or overloaded to have new meaning for a specific class objects.
- Already used the + and - operator when you are adding ints and floats and stuff. They have been overloaded to implicitly convert the operands if they are compatible, but not same.
- Overloading of operators are achieved by creating operaotr function.
- An operator function defined the operations that the overloaded operator can perform relative to the class.
- An operator function is created using the keyword Operator.
- Operator functions can be either members or non members of a class.
- Non member operator functions are always friend functions of the class, coz you need to access all the data members. So you gotta use the friend keyword.

Syntax:

```
Returntype classname::Operator OperatorSymbol (Arguement list)
{
// function body.
}
```

Restrictions on Operator Overloading

- Precedence or Associativity of an operator cannot be changed by overloading. So use parenthesis to force order of overloaded operators in an expression.
- C++ doesnt allow new operators to be created.
- Number of operands an operator takes cannot be changed, unary remain unary and stuff.
- Cannot overload the meaning of operators if all arguments are primitie data types.
- So that means no overloading for operators for built in types
- You cant change how 2 integers are added.
- You cant change ::, ., .*
- .