Object Oriented Programming
Lecture 9

Member initialization list and Multiple inheritance

Member initialization list

- Initializer list is used in initializing the data members of a class. The list of members to be initialized is indicated with constructor as a comma-separated list followed by a colon.

```
class Point {
private:
        int x;
        int y;
public:
        Point(int i, int j):x(i), y(j) {}
        /* The constructor can also be written as:
                Point(int i , int j ) {
                        x = i;
                        y = j;
                }  */
        int getX() const {return x;}
        int getY() const {return y;}
};
```

```
int main() {
Point t1(10, 15);
cout<<"x = "<<t1.getX()<<", ";
cout<<"y = "<<t1.getY();
return 0;
}
```

```
class Student {
private:
            int a;
public:
           Student(int x): a(x)
           {
          //   a=x;
           cout<<"Student(x) called"<< endl;
           }
      void print()
          {
                  cout<<"Value of a is  "<<a;
           }
};
```

```
class MS_Student : public Student {
public:
            MS_Student(int x): Student(x)
              {
           cout<<"MS student called"<< endl; }
};

int main() {
          MS_Student s1(30);
          s1.print();
}
```

Multiple Inheritance

- Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes; e.g.,

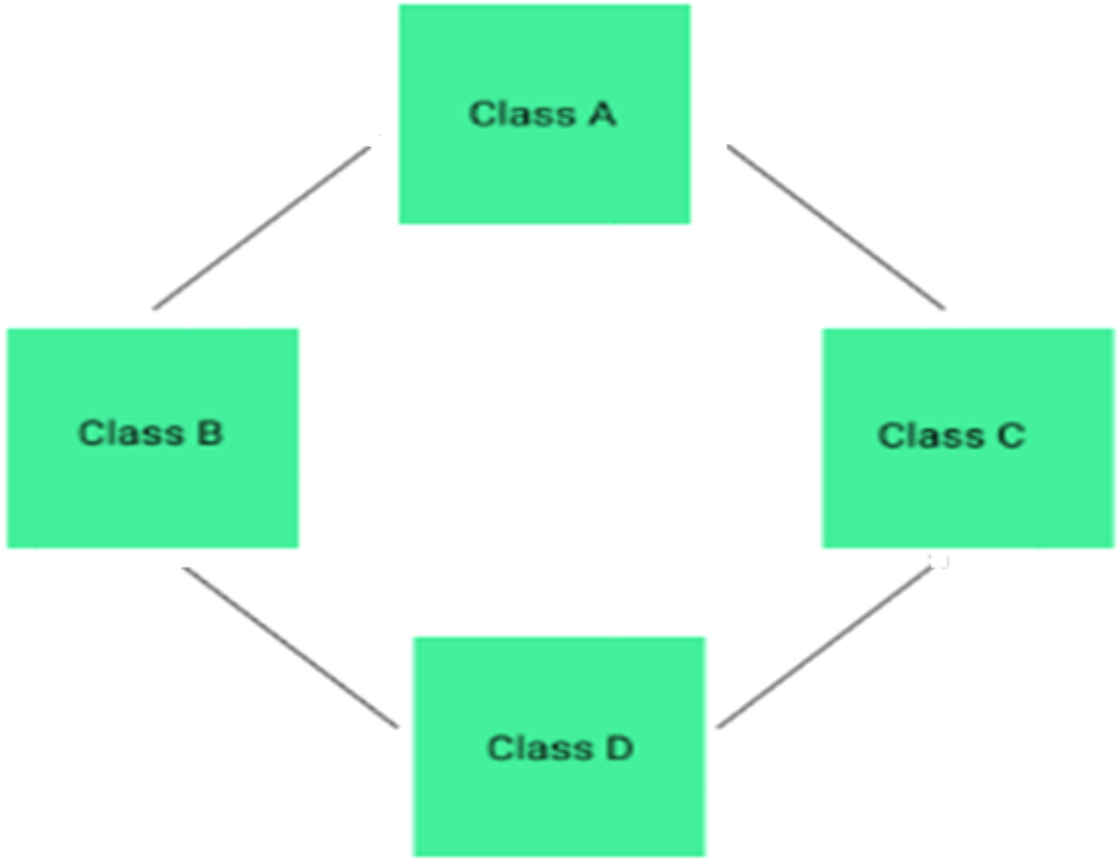
  -Vehicle, Four-wheeler, Sedan and Honda city

  -Student, MS student, BS student and TA

- The constructors of parent class is called first than the constructor of child class.
- Destructors are called in reverse order.
- The constructors of classes are called in the same order in which they are inherited.

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

class A
{
public:
A() { cout << "A's constructor called" << endl; }
~A() { cout << "A's destructor called" << endl; }
};

class B
{
public:
B() { cout << "B's constructor called" << endl; }
~B() { cout << "B's destructor called" << endl; }
};

class C: public B, public A // Note the order
{
public:
C() { cout << "C's constructor called" << endl; }
~C() { cout << "C's destructor called" << endl; }
};

int main()
{
        C c;
        return 0;
}
```

The diamond problem



- The diamond problem occurs when two super classes of a class have a common base class. For example, the TA class gets two copies of all attributes of Student class; it causes ambiguities.

- BS-Student and MS-Student are inherited from Student and TA is inherited from BS-Student and MS-Student classes.

```
//Example diamond problem
class A
{
int x;
public:
void setX(int i) {x = i;}
void print() { cout << x; }
};

class B: public A
{
public:
B() { setX(10); }
};
```

```
class C: public A
{
public:
C() { setX(20); }
};

class D: public B, public C {
};

int main()
{
        D d;
        d.print(); //error
        return 0;
}
```

Solution of diamond problem

1) Virtual inheritance is a C++ technique that ensures only one copy of a base class's member variables are inherited by grandchild derived classes.

2) Explicitly writing class name with the data member /member function causing diamond problem

```
class Student {
private:
          int a;
public:
          Student(int x)
          {
          a=x;
          cout<<"Student(x) called"<< endl;
          }
          Student()
          {
          cout<<"Student() called"<< endl;
          }
};

class MS_Student :  public Student {
public:
            MS_Student(int x): Student(x)
            {
            cout<<"MS student called"<< endl; }
};

class BS_Student :  public Student {
public:
          BS_Student(int x): Student(x)
           {
           cout<<"BS student called"<< endl; }
           };

class TA : public MS_Student, public BS_Student {
public:
          TA(int x):MS_Student(x), BS_Student(x)
           {
            cout<<"TA called"<< endl;
            }
};

int main() {
            TA ta1(30);
}
```

- In the above program, constructor of ‘Student’ is called two times. Destructor of ‘Student’ will also be called two times when object ‘ta1’ is destructed. So object ‘ta1’ has two copies of all members of ‘Student’, this causes ambiguities. *The solution to this problem is ‘virtual’ keyword*. We make the classes ‘MS-Student’ and ‘BS-Student’ as virtual base classes to avoid two copies of ‘Student’ in ‘TA’ class.

- In the above program, constructor of ‘Student’ is called once. One important thing to note in the output is, *the default constructor of ‘Student’ is called*. When we use ‘virtual’ keyword, the default constructor of grandparent class is called by default even if the parent classes explicitly call parameterized constructor.

Calling parameterized constructor

- **How to call the parameterized constructor of the ‘Student’ class?**
- The constructor has to be called in ‘TA’ class. For example, see the following program.

```
#include<iostream>
using namespace std;
class Student {
public:
Student(int x) { cout << “Student(int ) called" << endl; }
Student(){ cout << “Student() called" << endl; }
};

class MS_Student :virtual public Student {
public:
        MS_Student(int x):Student(x) {
        cout<<"MS-Student called"<< endl;
        }
};

class BS_Student : virtual public Student {

public:
        BS_Student(int x):Student(x) {
        cout<<"BS_Student called"<< endl;
        }
```

```
class TA : public MS_Student, public BS_Student {
public:
        TA(int x):BS_Student(x), MS_Student(x),
Student(x) {
                cout<<"TA(int ) called"<< endl;
        }
};

int main() {
        TA ta1(30);
}
```

Inheritance – class activity

- Write a program that implements seven classes named point, line, shape, triangle, rectangle, square, and circle. The shape class is inherited from point and line classes and has further sub-classes called triangle, rectangle, square, and circle. The point class has two members named as x, and y, and the line has member called slope. The shape class has a data member for the name of the shape, and method called area. The area method is specifically implemented for each sub-class of shape according to formula of the area according to the specific shape. Create object of each sub-type of shape to display the areas of each shape.

Class activity – Design of the term project

- Discuss the design of your term project in which you need to elaborate:
  - Number of classes to implement the given problem statement
  - Role each class is performing in the given system
  - Relationship between different classes

References

- C++ How to Program
  By Deitel & Deitel
- The C++ Programming Language
  By Bjarne Stroustrup
- Object oriented programming using C++ by Tasleem Mustafa, Imran Saeed, Tariq Mehmood, Ahsan Raza
- https://www.tutorialspoint.com/cplusplus
- http://ecomputernotes.com/cpp/introduction-to-oop
- https://www.tutorialspoint.com/cplusplus/cpp_inheritance.htm
- https://www.guru99.com/c-loop-statement.html
- www.w3schools.com
- https://www.geeksforgeeks.org/multiple-inheritance-in-c/