PH5720: Numerical Methods and Programming

Week - 06

How to program

A few words on C++ and STL

C++ Class and objects

Prabhat R. Pujahari

Indian Institute of Technology Madras

A few words on C++ and STL : different types of variables

- Global Variable: defined outside the function

- Program #1: variable1.cpp

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

int irow = 5, icol = 64;

Void Print()
{
   cout << “irow = “ << irow << “ icol = “ << icol << endl;
}

int main()
{
    cout << “irow = << irow << endl;
    // prints 5
    irow = 20;
    Print();
    // prints 20   64
    return 0;
}
```

A few words on C++ and STL : different types of variables

❖ Program #2: variable2.cpp

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

Void Print()
{
   cout << “irow = “ << irow << “ icol = “ << icol << endl;
}

int main()
{
    int irow = 5, icol = 64;

    cout << “irow = << irow << endl;    // prints 5

    Print();

    return 0;
}
```

2. Local Variable: defined inside the function

- Inside Print(), variables irow and icol are not defined. Will give compilation error.

❖ Program #3: variable3.cpp

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

float Sum(float data)
{
   static float total = 0.;
   total += data;
   return total;
}

int main()
{
    float result;
    for (int ii = 0; ii < 10; ii++)
        {
          data = (float) ii;
          result = Sum(data);
          cout <<  data << " " << result << endl;
        }
    return 0;
}
```

2. Static Local Variable:

Defined inside the function.

This variable is initialized once in the program. When it is updated, then it keeps the updated value in the memory.

❖ Program #3: variable3.cpp

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


int main()
{
    const float result = 5.;

    result = 10.; // This is not allowed


    return 0;
}
```

3. Constants – Syntax

```
const int total = 5;
```

Here you can not change the value once it is assigned.

The compiler will give error.

C++ Classes & Objects

- Important feature of Object-Oriented programming
- It is a user defined data type which has data members and member functions
- Data members are the data variables and member functions are the functions used to manipulate data members
- An object is an instance of a class

Note: No memory is allocated for a class when it is defined. However, memory is allocated when it is instantiated (i.e., an object is created)

❖ Class

A class is a collection of variables with related functions.
A class enables us to bundle various parts and various Functions into one collection which is called an Object.

- Let us design a class for a Cube.
- A Cube has the following things to calculate: Area, Volume
- It has only one parameter, i.e. length

```
#include <iostream>
using namespace std;
class Cube
{
    public:
     float FindArea()
       {
          float area = 6.* length * length;
          return area;
       }
     float FindVolume()
       {
         float vol = length * length * length;
         return vol;
       }

     float Length;
};
```

❖ Analysis:

- No data encapsulation
- Need improvement

```
int main()
{
   Cube aa;
   aa.length = 5.;
   cout << " area = " << aa.FindArea() << endl;

   return 0;
}
```

```
#include <iostream>
using namespace std;
class Cube
{
    public:
     float FindArea()
       {
          float area = 6 * Length * Length;
          return area;
       }
     float FindVolume()
       {
         float vol = Length * Length * Length;
         return area;
       }
     void SetLength(float len)
      {
        Length = len;
      }
    private:
     float Length;
};
```

❖ Analysis:

- aa.Length = 5. will not work
- Need to have a public function to access the data member

```
int main()
{
   Cube aa;
   aa.Length = 5.;
   cout << " area = " << aa.FindArea() << endl;

   return 0;
}
```

```
#include <iostream>
using namespace std;
class Cube
{
    public:
     float FindArea()
       {
          float area = 6.* Length * Length;
          return area;
       }
     float FindVolume()
       {
         float vol = Length * Length * Length;
         return vol;
       }
     void SetLength(float len)
       {
          Length = len;
       }
    private:
     float Length;
};
```

❖ Analysis:

- ➢ You can not check what value is set to the data member
- ➢ Need to have a public function to access the data member

```
int main()
{
   Cube aa;
   float length = 5.;
   aa.SetLength(length);
   cout << " area = " << aa.FindArea() << endl;

   return 0;
}
```

```
#include <iostream>
using namespace std;
class Cube
{
    public:
     float FindArea()
       {
          float area = 6.* Length * Length;
          return area;
       }
     float FindVolume()
       {
         float vol = Length * Length * Length;
         return vol;
       }
     void SetLength(float itsLength)
       {
          Length = itsLength;
       }
     float GetLength() {return Length;}
    private:
     float Length;
};
```

❖ Analysis:

- We are not changing the value
- Hence, it is better to define it constant.

```
int main()
{
   Cube aa;
   float length = 5.;
   aa.SetLength(length);
   cout << aa.GetLength() << endl;
   cout << " area = " << aa.FindArea() << endl;

   return 0;
}
```

```
#include <iostream>
using namespace std;
class Cube
{
    public:
     float FindArea()
       {
          float area = 6.* Length * Length;
          return area;
       }
     float FindVolume()
       {
         float vol = Length * Length * Length;
         return vol;
       }
     void SetLength(float itsLength)
       {
          Length = itsLength;
       }
     void GetLength() const {return Length;}
    private:
     float Length;
};
```

❖ Analysis:

- Data members are not initialized
- Can be done through constructor
- The moment constructor is defined we need to define destructor

```
int main()
{
   Cube aa;
   float length = 5.;
   aa.SetLength(length);
   cout << aa.GetLength() << endl;
   cout << " area = " << aa.FindArea() << endl;

   return 0;
}
```

cube.h

```
#include <iostream>
using namespace std;
class Cube
{
    public:
      Cube();
      Cube(float itsLength);
      ~Cube();
      void FindArea();
      void FindVolume();
      void SetLength(float itsLength);
      float GetLength() const;
      float GetArea() const;
      float GetVolume() const;

    private:
      float Length;
      float Area;
      float Volume;
};
```

cube.cxx

```
#include <cube.h>
Cube::Cube():
  Length(0.),
  Area(0.),
  Volume(0.)
{
}
Cube::Cube(float itsLength):
  Length(itsLength),
  Area(0.),
  Volume(0.)
{
}
Cube::~Cube()
{
}
void Cube::FindArea()
  {
    Area = 6. * Length * Length;
  }
void Cube::SetLength(float itsLength)
{
  Length = itsLength;
}
float Cube::GetLength() const
{
  return Length;
}
```

➢ How to use it?

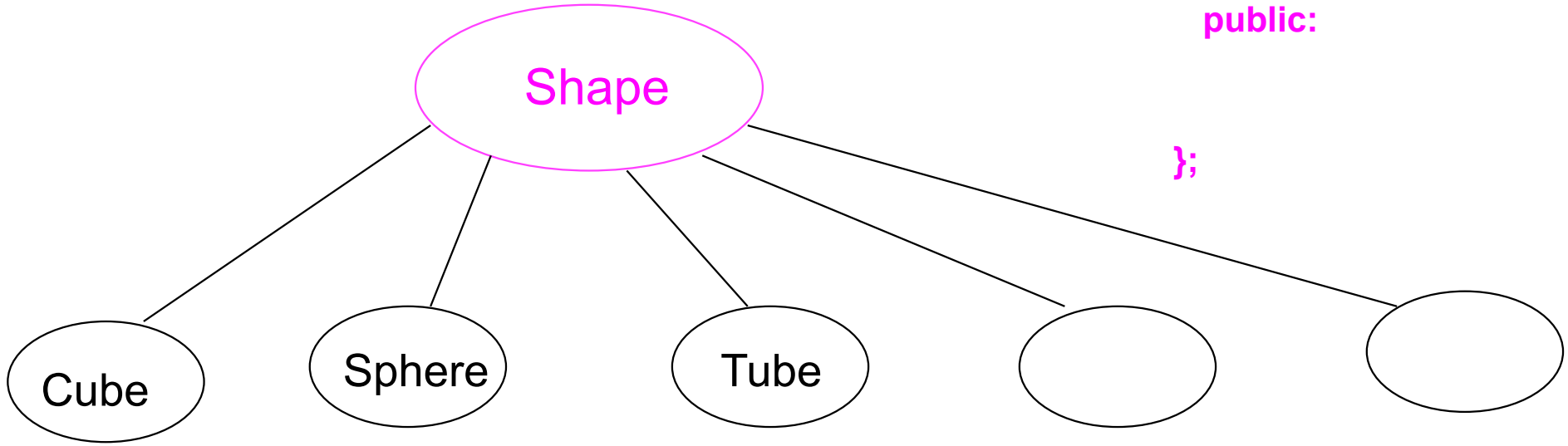
```
#include <iostream>
int main()
{
      using std::cout;
      using std::endl;
      float x = 5.;


       Cube aa(x);
       aa.FindVolume();
       cout << “Volume = “ << aa.GetVolume()
              << endl;



      return 0;
}
```

➢ How to use it?

```
#include <iostream>
int main()
{
      using std::cout;
      using std::endl;
      float x = 5.;


       Cube *aa = new Cube(x);
       aa->FindVolume();
       cout << “Volume = “ << aa->GetVolume()
              << endl;



      return 0;
}
```

Abstract Class

```
class Shape
{
  public:


};
```




```
class Cube:public Shape
{
  public:
    ……..

  protected:
    ……..
};
```

```
class Sphere:public Shape
{
  public:
    ……..

  protected:
    ……..
};
```

How to use it?

```
#include <iostream>
int main()
{
  Shape *aa;
  aa = new Cube;
  ……..
  ……..
  ……..
  aa = new Sphere;
  ……..
  ……..
  return 0;
}
```