Object Oriented Programming

Lecture

Review

- this Pointer
- Separation of interface and implementation
- Constant **member functions**

Problem

- Change the class ***Rectangle*** such that a *width* and *height* is given when the object is created and cannot be changed afterwards

Rectangle Class

```
class Rectangle
{
   int width, height;
public:
   Rectangle(int w, int h);
   void set_width(int w);
   void set_height(int h);
   int area();
};
```

Modified Rectangle Class

```
class Rectangle
{
   const int width, height;
public:
   Rectangle(int w, int h);

   int area();
};
```

Example

```
Rectangle::Rectangle(int w, int h);
{
   width=w;
   height=h;

   /*error: cannot modify a constant data
   member*/
};
```

Member Initializer List - Constructor

- A member initializer list is a mechanism to initialize data members
- It is given after closing parenthesis of parameter list of constructor
- In case of more then one member use comma separated list

Example

```
class Rectangle
{
    const int width, height;
public:
    Rectangle(int w, int h) :width(w), height(h)
    {

    }
    void set_width(int w);
    void set_height(int h);
    int area();

};
```

This is called member initializer list

Order of Initialization

- Data member are **initialized in order they are declared**

- Order in member initializer list is **not significant at all**

Example

```
class ABC
{
   int x;
   int y;
   int z;

public:
   ABC();
};
```

Example

```
ABC::ABC() : y(10), x(y), z(y)
{
…
}
```

Order of declaration not the member initialization list:

1) x its value will be equal to y since y is not initialized yet so garbage value
2) y its value will be equal to 10
3) z its value will be equal to y that is 10 it is initialized now

```
/*  x = Junk value
    y = 10
    z = 10 */
```

Const Objects

`const` Objects

- Objects can be declared constant with the use of ***const*** keyword
- Constant objects cannot change their state

Example

```
int main()
{
 const Rectangle rect;
 return 0;
}
```

Example

```
class Rectangle{
…
  int width;
public:
…
  int getWidth(){
      return width;
  }
};
```

Example

This code is in correct

```
int main(){
  const Rectangle rect;
  int a = rect.getWidth();
  //error
}
```

The get width is not marked as const it may change the object value
the getwidth fucntion should also be marked as constant

`const` Objects

- **`const`** objects cannot access *"non const"* member function
- Chances of unintentional modification are eliminated

Example

This code is in correct

```
class Rectangle{
…
  int width;
public:
…
  int getWidth()const{
      return width;
  }
};
```

Because width is const but it is not declared which means that we will have a garabage value everytime which should not be the case with const

Example

```
int main(){
  const Rectangle rect;
  int a = rect.getWidth();
}
```

Correct Code:

```
#include<iostream>
using namespace std;
class Rectangle
{
    const int width=10;
    public:
    int getWidth()const
    {
    return width;
    }
};
int main()
{
    const Rectangle rect;
    int a = rect.getWidth();
    cout<<"The value of a is: "<<a<<endl;
}
```

Constant data members

- Make all functions that don’t change the state of the object constant
- This will enable constant objects to access more member functions

Thanks a lot