

第3讲

类和对象的概念及定义

主讲人：赵文彬

本次课主要内容

- 类的声明和对象的定义
- 类的成员函数
- 对象成员的引用
- 类的封装性和信息隐蔽

难点：对类和对象的理解

类的声明和对象的定义

➤ 类和对象的关系

- 类是对象的抽象，对象是类的具体实例(instance)。
- 类是抽象的，不占用内存，而对象是具体的，占用存储空间。

在C++中，需要先声明一个类类型，然后再用它去定义若干个同类型的对象。对象就是类类型的一个变量。可以说类是对象的模板，是用来定义对象的一种抽象类型。

类的声明和对象的定义

➤ 声明类类型

class 类名

{private:

私有的数据和成员函数;

public:

公用的数据和成员函数;

protected:

受保护的数据和成员函数;

};

private、protected、public
称为成员访问限定符
(member access specifier)。

注意:

如果在类的定义中
不指定限定符, 则
系统默认为私有的
(private)。

类的声明和对象的定义

➤ 说明：

- 用 **private** 声明的成员称为**私有成员**，它只能被类中的成员函数访问，不能被类外访问，但可以被友元类的成员函数访问（后面讲到）。
- 用 **protected** 声明的成员称为**受保护成员**，它不能被类外访问(这点与私有成员类似)，但可以被派生类的成员函数访问（后面讲到）。
- 用 **public** 声明的成员称为**公有的成员**，它可以被类中的成员函数访问，也可以被类外访问。

类的

➤ 对比结构体类

```
struct student
{ int num;
  char name[20];
  char sex;
};
student stud1;
```

```
class student
{ int num;
  char name[20];
  char sex;
public:
  void display()
  { cout<<"num:"<<num<<endl;
    cout<<"name:"<<name<<endl;
    cout<<"sex:"<<sex<<endl;
  }
};
student stud1;
```

类的声明和对象的定义

➤ C++允许用struct定义类类型

```
struct student  
{...//类成员声明与class一致  
};  
student stud1
```

用struct声明的
结构体类型实际
上也就是类。

➤ 区别

- 用**struct**声明的类，如果对其成员不作private或public的声明，系统将其**默认为public**。
- 用**class**定义的类，如果不作private或public声明，系统将其成员**默认为private**。

类的声明和对象的定义

```
class student
{ private:
    ...
    public:
    ...
    protected:
    ...
};
student stud1
```

型,

型
类名

student stud1;

```
class student
{ private:
    ...
    public:
    ...
    protected:
    ...
};
stud1;
```

```
class
{ private:
    ...
    public:
    ...
    protected:
    ...
};
stud1;
```

- 定义类类型的同时定义对象
- 不出现类名直接定义对象

类的成员函数

- 类的成员函数(简称类函数)是函数的一种, 它的用法和作用与**普通函数基本上是一样的**。

- **类函数的定义**

```
函数返回值类型  函数名 (参数表)  
{...//函数体  
}
```

可以被指定为
private (私有的)、public
(共有的) 或
protected (受保护的)

```
class student
{ public:
    void display();
private:
    int num;
    char name[20];
    char sex;
};
void student :: display()
{ cout<<"num:"<<num<<endl;
  cout<<"name:"<<name<<endl;
  cout<<"sex:"<<sex<<endl;
}
student stud1;
```

类外定义类函数，必须在函数名面加上**类名和作用域限定符**“::”。

```
class student
{ public:
    inline void display();
private:
    int num;
    char name[20];
    char sex;
};
inline void student :: display()
{ cout<<"num:"<<num<<endl;
  cout<<"name:"<<name<<endl;
  cout<<"sex:"<<sex<<endl;
}
student stud1;
```

如果在类外定义inline函数，则必须将类定义和成员函数定义放在同一个头文件中或同一个源文件中。

类的成员函数

➤ 注意：

- 类内定义的成员函数，默认为内联函数(inline)
- 在类外定义时，在类体中要对函数作声明。
- 如果在作用域运算符“::”前没有类名，或者函数名前面既无类名又无作用域运算符“::”，则表示该函数不属于任何类，为普通函数。

`::display()` 或 `display()`

`display()`函数不属于任何类，为普通函数。

类的成员函数

```
class time
{public:
    int hour,minute,sec;
    void set()
    { cin>>a>>b>>c;
    }
};
cout<<sizeof(time)<<endl;
```

12

对象 n 数据 n

函数

每个对象所占的空间大小只取决于对象中数据成员所占的空间，而与函数无关。

数据1

数据2

公用函数代码

类的成员函数

➤ 提问: 对结构体的成员访问方法?

在程序中经常需要访问对象中的成员，与访问结构体成员类似。

➤ 三种方法:

- 通过**对象名和成员运算符**访问对象中的成员;
- 通过**指向对象的指针**访问对象中的成员;
- 通过**对象的引用变量**访问对象中的成员。

```
class student
{ public:
    void display();
    int num;
    char name[20];
    char sex;
};

void student::display()
{ cout<<"num:"<<num<<endl;
  cout<<"name:"<<name<<endl;
  cout<<"sex:"<<sex<<endl;
}

student stud1,*p;
```

田

(1) stud1.num=1001;
(2) p=&stud1;
 p->num=1001;
 或(*p).num=1001;
(3) student &stud2=stud1;
 stud2.num=1001;

类的封装性和信息隐蔽

- **公用接口与私有实现的分离**
 - 如果不必公开的信息，使用：private
 - 要公开的，才使用：public
- **类声明和成员函数定义的分离**
 - 为便于多个程序使用某类，应当把类的声明和成员函数的声明放在**头文件**中（即.h文件），把成员函数的定义放在**另一个文件**中（即.cpp文件）。

类的封装性和信息隐蔽

➤ 面向对象程序设计中的几个名词

- “**方法**”(method): “方法”是指对数据的操作（类的成员函数）。一个“**方法**”对应一种操作。
- 外界是通过发“**消息**”来**激活**有关方法的。
- “**消息**”：其实就是一个命令，由程序语句来实现。
- **发送“消息”**：一个对象**通知**另一个对象执行它的某条成员函数。
- **接收“消息”**：一个对象**接收到**另一个对象要求它执行它的某条成员函数的信息。

信息隐蔽

```
class student
{ public:
    void display();
private:
    int num;
    char name[20];
    char sex;
};
void student :: display()
{ cout<<"num:"<<num<<endl;
  cout<<"name:"<<name<<endl;
  cout<<"sex:"<<sex<<endl;
}
student stud1,*p;
```

对象:
stud1
方法:
display()
消息
stud1.display()

类和对象的简单应用举例

```
#include <iostream>
using namespace std;
class Time
{public:
    int hour;
    int minute;
    int sec;
};
```

```
int main( )
{ Time t1;
  cin>>t1.hour;
  cin>>t1.minute;
  cin>>t1.sec;
  cout<<t1.hour<<":";
  cout<< t1.minute<<":";
  cout<<t1.sec <<endl;
  return 0;
}
```

运行情况如下：

11 11 11✓

11:11:11

类和对象的基本应用实例

找出一个整型数组中:

//Array_max.h文件

```
class Array_max
{public:
    void set_value( );
    void max_value( );
    void show_value( );
private:
    int array[10];
    int max;
};
```

//Array_max.cpp文件

```
#include <iostream>
#include "Array_max.h"
using namespace std;
void Array_max::set_value( )
{ int i;
  for (i=0;i<10;i++)
    cin>>array[i];}
void Array_max::max_value( )
{ int i;
  max=array[0];
  for (i=1;i<10;i++)
    if(array[i]>max) max=array[i];
}
void Array_max::show_value( )
{cout<<"max="<<max;}
```

类和对象的简单应用举例

```
//main.cpp文件
#include <iostream>
#include "Array_max.h"
using namespace std;
int main( )
{ Array_max arrmax;
  arrmax.set_value( );
  arrmax.max_value( );
  arrmax.show_value( );
  return 0;
}
```

运行结果如下：

12 12 39 -34 17 134 045 -91 76✓

max=134

类和对象的简单应用举例

//CRectangle.h

```
class CRectangle
{ int x, y;
public:
    void set_values (int,int);
    int area ()
    {      return (x*y);  }
};
```

//CRectangle.cpp

```
#include "CRectangle.h"
void CRectangle::set_values (int a, int b)
{  x = a;
  y = b;}
```

//main.cpp

```
#include <iostream>
#include "CRectangle.h"
using namespace std;
int main ()
{ CRectangle rect;
  rect.set_values (3,4);
  cout << "area: " << rect.area();
  return 0;}
```



ca. C:\Windows\system32\cmd.exe

area: 12Press any key to continue . . .

类和对象的简

//Complex.cpp

#include "Complex.h"

void Complex::setValue(double a,
double b){ real = a;
imag = b;}

void Complex::display()

{ cout << real;
if(imag > 0)
cout << "+";
cout << imag;
cout << "i";}

Complex Complex::conjugate()

{ Complex con;
con.real = real;
con.imag = -1.0 * imag;
return con;}

//Complex.h

#include <iostream>

using namespace std;

class Complex

{ private:

double real,imag;

public:

void setValue(double, double);

double getReal(){return real;}

double getImag(){return imag;}

void display();

Complex conjugate();

};

类和对象的简单应用举例

```
//main.cpp
```

```
#include <iostream>
```

```
#include "Complex.h"
```

```
using namespace std;
```

```
void main()
```

```
{ Complex c,conj;
```

```
  c.setValue(1.1,2.1);
```

```
  cout << "实部为: " << c.getReal() << endl;
```

```
  cout << "虚部为: " << c.getImag() << endl;
```

```
  conj = c.conjugate();
```

```
  cout << "复数的";
```

```
  c.display();
```

```
  cout << "共轭为: ";
```

```
  conj.display();
```

```
  cout << endl;}
```

```
C:\Windows\system32\cmd.exe
```

```
实部为: 1.1
```

```
虚部为: 2.1
```

```
复数的1.1+2.1i共轭为: 1.1-2.1i
```

```
Press any key to continue . . .
```


小结

- 类的声明和对象的定义；
- 类的成员函数的定义；
- 对象成员的引用；