C++程序设计(II)

授课：毛国红
电话：13857144844（644844）
作业邮箱：1419470@QQ.com
浙江工业大学计算机学院

第7讲 类与对象

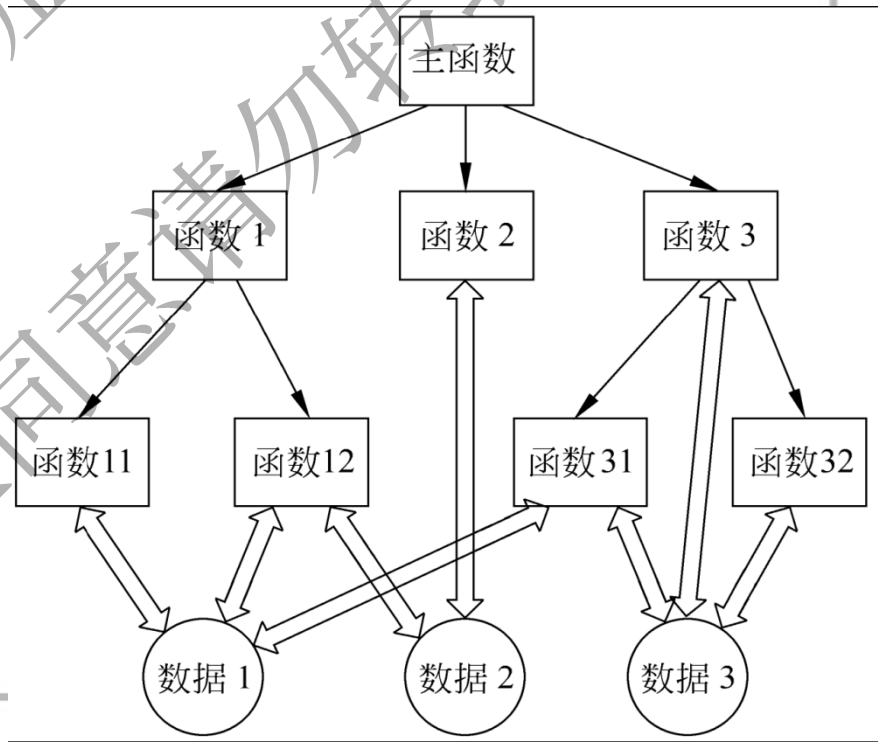
- 参考书目
  - c++程序设计 谭浩强
    第8章 类与对象的特性
  - 面向对象程序设计基础 李师贤 李文军 周晓聪
    第5章
  - c++程序设计教程（第二版） 钱能
    第8章，第9章，第11章

第7讲 类与对象

1. 简单讨论
2. 类/类的定义和使用
3. 对象/对象的声明和使用
4. 用类和对象构造程序的实例

第7讲 类与对象-简单讨论

- 面向过程的程序设计和面向对象的程序设计
  - 面向过程的程序设计
    - 传统的设计方法，围绕功能进行；
    - 用一个函数实现一个功能；
    - 所有的数据都是公用的；
    - 一个函数可以使用任何一组数据；
    - 一组数据又能被多个函数所使用。



第7讲 类与对象-简单讨论

- 面向过程的程序设计和面向对象的程序设计
  - 面向对象的程序设计

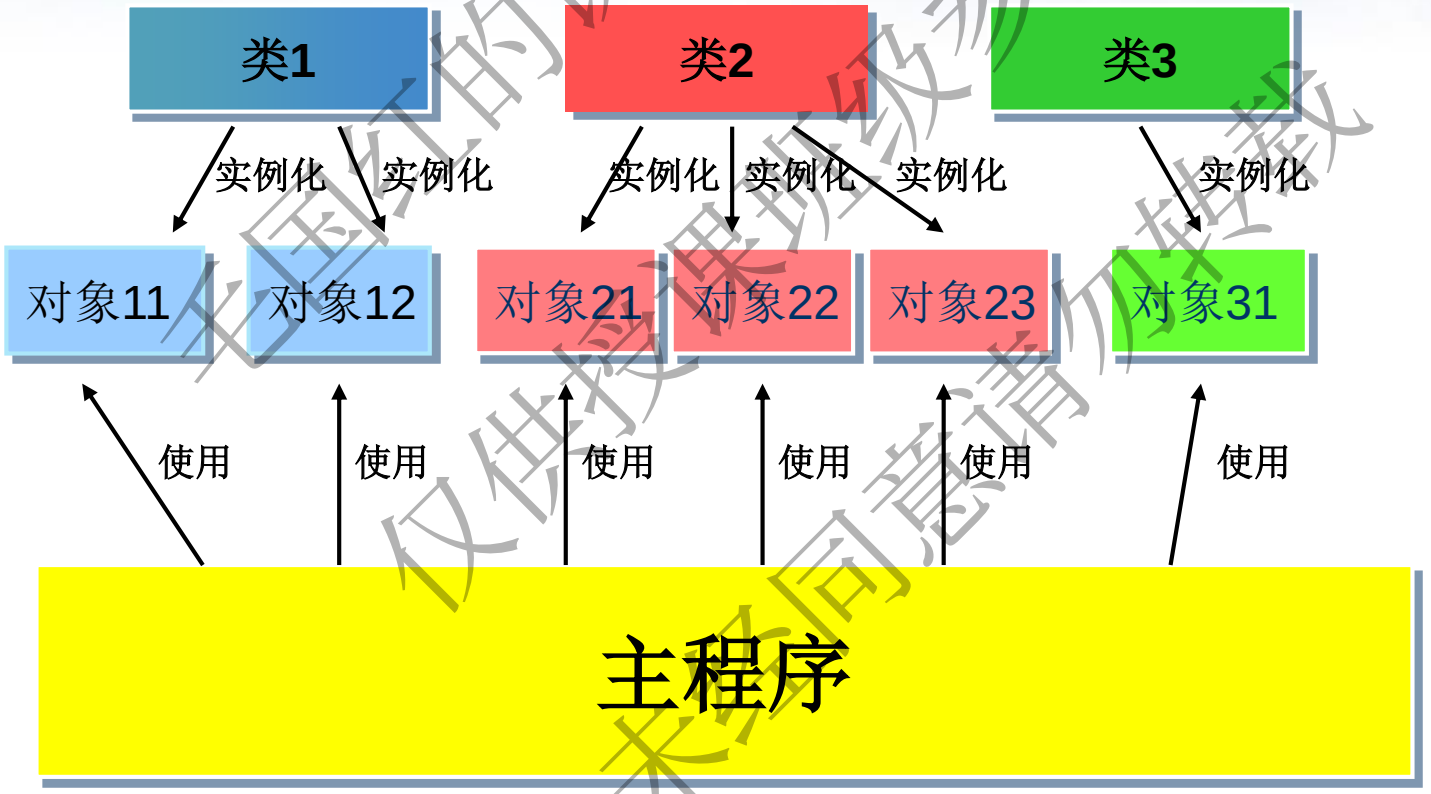
    -围绕现实世界的实体（对象）进行设计；

    -程序设计者从设计函数功能转向：设计类与对象（如何用属性和行为来描述一个实体）；如何向实体发送消息以调度实体的行为。

    - 一种以认识认识世界的方法为参考的程序设计方法，更为自然，更利于大型程序的组织和实现。

第7讲 类与对象-简单讨论

- 面向过程的程序设计和面向对象的程序设计
  - 面向对象的程序设计



第7讲 类与对象-简单讨论

- 面向过程的程序设计和面向对象的程序设计
  - 实例回顾

A工程队接了若干个建造游泳池的项目。经历数周，他们终于做完了工程。现在到了跟客户收项目款的时间，请你补充完下面的程序以便帮助这个工程队快速的完成收款工作。

备注：游泳池的造价由走道费用和围栏费用两部分构成。铺设走道的单价为167.5元/平方米，围栏的单价是36.4元/米。其中,游泳池形状如图所示，橙色部分为走道红色部分为围栏。

工程队一共造了5个游泳池。分别为

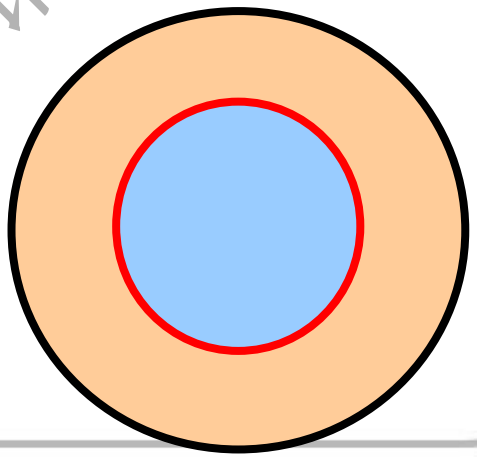
1号池半径12.2米，走道宽3米；

2号池半径5米，走道宽2.8米；

3号池半径4.8米，走道宽1米；

4号池半径宽6米；走道宽1.4米；

5号池半径8.7米，走道宽2.3米。



第7讲 类与对象-简单讨论

- 面向过程的程序设计和面向对象的程序设计
  - 实例回顾-面向过程的程序设计实现

```
#include <iostream>
#define pi  3.14
using namespace std;
double fwlc(double);
double fzds(double,double);
int main()
{  const double zddj=167.5;//走道单价
   const double wldj=36.4;//围栏单价
   double r;//池直径
   double c;//走道宽
   double wlc;//围栏长度
   double zds;//走道面积
```

函数声明

```
for(int I=1;I<=5;I++)
{  cout<<"请输入第" <<I
<<"号游泳池数据(直径，走道
宽)：";
     cin>>r>>c;
     wlc=fwlc(r);
     zds=fzds(c,r);
     cout<<"第" <<I<<"号造价为："
        <<(wlc*wldj+zds*zddj)<<"元"
        <<endl;
  }
return 0;
}
```

函数调用

第7讲 类与对象-简单讨论

- 面向过程的程序设计和面向对象的程序设计
  - 实例回顾-面向过程的程序设计实现

```
double fwlc(double r)
{//围栏周长
    double re;
     re=2*pi*r;
    return re;
}
```

```
double fzds(double c,double r)
{ //周道面积
   double re;
   re=pi* (r+c)*(r+c)-pi*r*r;
   return re;
}
```

函数定义/函数实现

第7讲 类与对象-简单讨论

- 面向过程的程序设计和面向对象的程序设计
  - 实例回顾-面向对象的程序设计实现

```
#include <iostream>
#include “pool.hpp”
using namespace std;
int main()
{  const double zddj=167.5;//走道单价
   const double wldj=36.4;//围栏单价
   double r;//池直径
   double c;//走道宽
   pool  apool;
 for(int I=1;I<=5;I++)
 {  cout<<”建造第” <<I<<”号游泳池”：
    apool.build();
    cout<<”第” <<I<<”号游泳池造价为：
<<(apool.rail_length()*wldj+apool.rail_area()*zddj)<<”元” <<endl;
  }
return 0;   }
```

类的头文件引入

声明对象/类的实例化

对象的使用

第7讲 类与对象-简单讨论

- 面向过程的程序设计和面向对象的程序设计
  - 实例回顾-面向对象的程序设计实现

```
//pool.hpp或者pool.h
class pool{
public:
   void build();
   double rail_length ();
   double rail_area();
private:
   //double cir_area(double);
   double radius;
   double c;
};
```

类的声明

第7讲 类与对象-简单讨论

- 面向过程的程序设计和面向对象的程序设计
  - 实例回顾-面向对象的程序设计实现

```
//pool.cpp
#define pi 3.14
#include "pool.hpp"
void pool::build()
{    cin>>radius>>c;
 }

double pool::rail_area()
{ return  pi*((radius+c)*(radius+c)
                    - radius*radius);
}
```

类的实现

```
double pool::rail_length()
{ return 2*pi*radius;
 }

/*
double pool::cir_area(double rv)
{ return  pi*rv*rv;  }
*/
```

pool.hpp + pool.cpp

类的定义

第7讲 类与对象-类/类的定义和使用

❖类的概念

- 类是对一群具有相同属性(特征/数据)，表现相同行为的对象的描述。→抽象
- 类→类型: c++已有的基本数据类型，int，double，char等均可归入对具体对象的抽象归纳。有相同属性的数据抽象为一种类型，具有相同的行为(运算操作)。
- pool 类型。属性：半径radius，走道宽c；行为：游泳池建造build，计算围栏长rail_length，计算走道面积rail_area等。

第7讲 类与对象-类/类的定义和使用

❖类的概念

- 类的封装：将属性（数据）和行为（操作代码）放在一起，形成一个基本的单位，对外保留访问接口，隐藏数据。比如，pool的操作对外公开可访问，数据部分隐藏。
- 类的继承：与客观的世界相仿，类可以作为父亲（父类/基类）派生出子类（派生类）。子类可以从父类那里继承东西（数据和操作）。比如,马类可派生出白马类和黑马类，pool可作为父类派生出圆游泳池类和方游泳池类。

第7讲 类与对象-类/类的定义和使用

❖类的声明

- class 类名{
  public:
  公有数据和函数
  private:
  私有数据和函数
  };

```
//pool.hpp或者pool.h
class pool{
public:
   void build();
   double rail_length ();
   double rail_area();
private:
    //double cir_area(double);
   double radius;
   double c;
};
```

- 类名：有效的C++标识符

第7讲 类与对象-类/类的定义和使用

❖类的声明

- class 类名{
  public:
    公有数据和函数
  private:
    私有数据和函数
  };

```
//pool.hpp或者pool.h
class pool{
public:
   ……
private:
   double radius=0.0;//X
   static double c;//V
};
```

- 数据成员的声明:
  - 像声明普通变量的方式来声明，但不允许声明时初始化；
  - 允许是任何数据类型, 包括用户自定义的类型(不允许是当前正在定义的类型);
  - 声明时可以用static修饰，可以用const修饰；不允许使用auto,register和extern修饰。

第7讲 类与对象-类/类的定义和使用

❖类的声明

- class 类名{
  public:
  公有数据和函数
  private:
  私有数据和函数
  };

类内会有一些特殊的成员函数声明方式跟普通函数不同：

- 构造函数-负责对象的初始化
- 拷贝构造函数-负责对象的拷贝初始化
- 析构函数-负责对象的撤销

- 成员函数的声明:
  - 普通函数：非成员函数，即类外定义的函数。
  - 大部分成员函数可像声明普通函数的方式在类内声明，可以使用const修饰。→常量成员函数。
  - 常量成员函数可以改变局部变量，全局变量或其他类对象的值，但不允许修改本类中的数据成员的值。
  - 普通函数不可以作为常量函数（不能用const修饰）

第7讲 类与对象-类/类的定义和使用

- 类的声明
  - class 类名{
    public:
      公有数据和函数
    private:
      私有数据和函数
    };

```
//pool.hpp或者pool.h
class pool{
public:
    double rail_length() const;
    ……
private:
   //double cir_area(double);
   double radius;
   double c;
};
```

```
double pool::rail_length() const
{    radius=radius+3; //X
     return 2*pi*radius;
 }
```

第7讲 类与对象-类/类的定义和使用

❖类的实现

- 类的声明文件中只包含类的所有数据成员及成员函数原型，没有成员函数的定义。→类的实现文件

类X

1

```
//类声明x.hpp
class x {
public:
  int func1();
  int func2();
   ……
};
```

2

```
//类实现x.cpp
#include “x.hpp”
int x::func1()
{……}
int x::func2()
{……}
```

3

```
//程序A.cpp
#include “x.hpp”
int main()
{  x obj;
    int I;
    ……
    i=obj.func1();
    ……          }
```

第7讲 类与对象-类/类的定义和使用

❖类的实现

- 类的实现文件中对成员函数的定义:
- 返回类型 类名::成员函数名(参数说明)

```
{
    函数体
}
```

- :: 类作用域运算符，类名:: 表示其后的成员函数名是在这个类中声明的，在函数体中可以直接访问该类中声明的成员（数据和函数）。

```
//pool.cpp
#define pi 3.14
#include "pool.hpp"
void pool::build( )
{   cin>>radius>>c;
}
```

第7讲 类与对象-类/类的定义和使用

❖类的实现

- **inline 内联函数/内置函数：** 如果在类中定义的成员函数中不包括循环等控制结构，C++系统会自动将它们作为内联函数来处理。→成员函数的实现有时候可以写在类内

```
//pool.cpp
#define pi 3.14
//pool.hpp或者pool.h
class pool{
public:
   double rail_length()
   {   return 2*pi*radius;   }
   ……
private:
   //double   cir_area(double);
   double radius;
   double c;
};
```

1.写在类内的函数自动默认为内联，inline可省略。

2. 若函数实现写在类外，则需显示加上inline 。

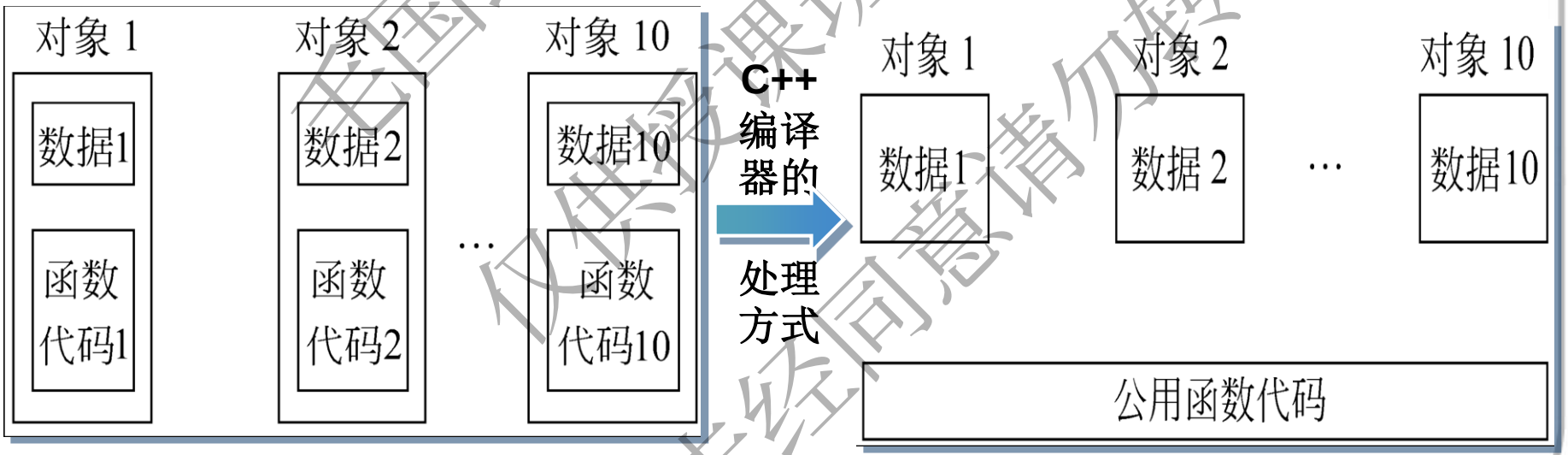
```
inline double rail_length();

inline double pool::rail_length()
{return 2*pi*radius;}
```

第7讲 类与对象-类/类的定义和使用

❖类的使用

- 类类型是一种用户自定义类型，程序员可以使用这个新类型在程序中声明变量，具有类类型的变量称为对象。
- **pool apool;**
- **X x1,x2,x3;**



- 可以使用sizeof()验证对象所占用的空间大小。

第7讲 类与对象-类/类的定义和使用

❖类的访问控制

- 规定类的成员(数据成员和成员函数)对外部的开放程度。

- **public**：可以让本类内成员和其他程序代码使用的部分放在公有访问控制内。--类的对外接口

  在声明了一个类以后，用户主要是通过调用公用的成员函数来实现类提供的功能(例如对数据成员设置值，显示数据成员的值，对数据进行加工等)。
- **private**：只允许本类内成员使用的部分放在私有访问控制内。--类的对外隐藏
- **protected**：允许继承体系中的类成员访问的部分放在受保护访问控制内。

第7讲 类与对象-类/类的定义和使用

❖类的访问控制

```
//pool.hpp或者pool.h
class pool{
public:
   void build();
   double rail_length ();
   double rail_area();
private:
   double cir_area(double);
   double radius;
   double c;
};
```

```
#include <iostream>
#include “pool.hpp”
int main()
{  …….
   pool  apool;
 for(int l=1;l<=5;l++)
 {  ….
    apool.build();
    apool.radius=r; //X
    apool.c=c; //X
    cout<<apool.cir_area(r+c)-
apool.cir_area(r);//X
    cout<<apool.rail_length()*wldj
              +apool.rail_area()*zddj;
 }
……
```

第7讲 类与对象-类/类的定义和使用

❖类的访问控制

```
//pool.cpp
……
double pool::rail_area()
{
    return pi*((radius+c)*(radius+c) - radius*radius);
}
```

```
double pool::rail_area()
{
    return cir_area(r+c) - cir_area(r);
}
```

第7讲 类与对象-类/类的定义和使用

❖类的访问控制

- 原则：

将需要被外界调用的成员函数指定为public(类的对外接口)。

有的函数并不是准备为外界调用的，而是为本类中的成员函数所调用的，就应该将它们指定为private。工具函数(辅助函数utility function)。

C++程序设计（II）

浙江工业大学计算机学院