C++程序设计

主讲：王老师

尚德机构

学习是一种信仰

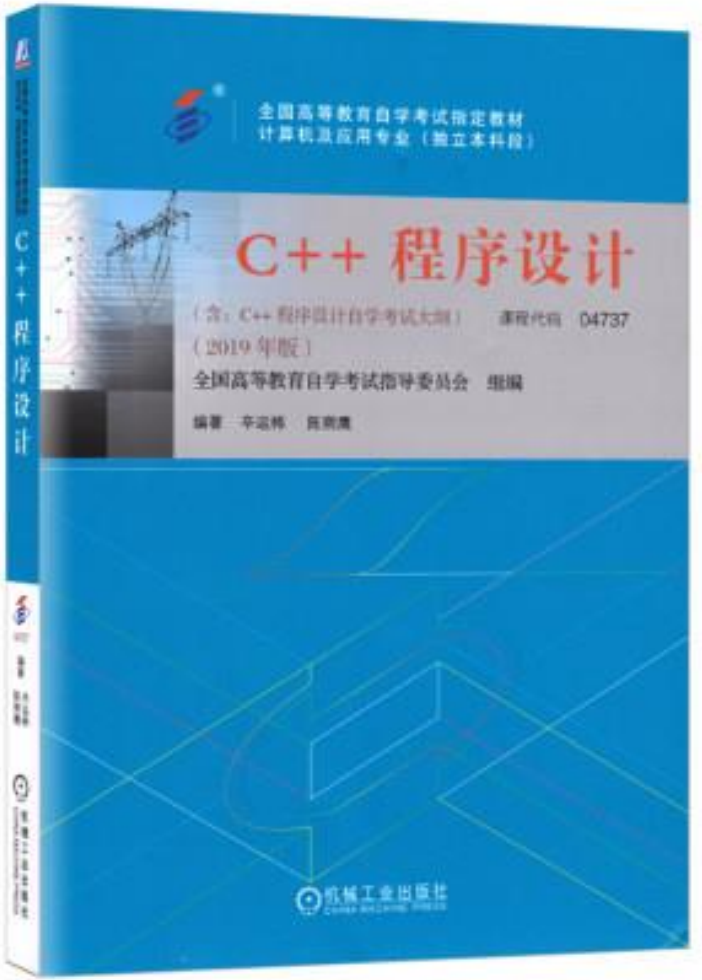
教材介绍

C++程序设计

（2019年版）

编著：辛运帷 陈朔鹰

机械工业出版社



考试题型

单选题　　1分×20题 = 20分

填空题　　1分×15题 = 15分

程序填空题　4分×5题 = 20分

程序分析题　6分×5题 = 30分

程序设计题　　2题 = 15分

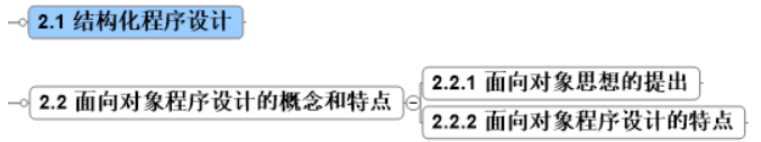
（第一题5分，第二题10分）

第二章　面向对象的基本概念

本章主要内容

第二章 面向对象的基本概念

2.1 结构化程序设计



- 在结构化程序设计中，采用**自顶向下、逐步求精及模块化**的思想，将复杂的大问题层层分解为许多简单的小问题。
- 在编写程序时，使用3种基本控制结构来构造程序。可以说，程序基本上都含有**顺序、选择、循环**3种基本控制结构，这3种结构到目前为止仍是主要的控制结构。程序以控制结构为单位，只有一个入口和一个出口，基于控制结构可以从前往后地顺序阅读程序，程序的静态描述与执行时的控制流程容易对应，所以可以独立地理解各个部分。结构化程序设计主要强调的是程序的易读性。

课堂练习

结构化程序设计所规定的三种基本控制结构是（　　）。

A:输入、处理、输出

B:树形、网形、环形

C:顺序、选择、循环

D:主程序、子程序、函数

课堂练习

结构化程序设计所规定的三种基本控制结构是（　　）。

A:输入、处理、输出

B:树形、网形、环形

C:顺序、选择、循环

D:主程序、子程序、函数

答案：C

2.2 面向对象程序设计的概念和特点

- 所谓面向对象的程序设计方法，就是使分析、设计和实现一个系统的方法尽可能地接近人们认识一个系统的方法。通常包括3个方面：面向对象的分析、面向对象的设计和面向对象的程序设计。
- 面向对象技术把问题看成是相互作用的事物的集合，也就是对象的集合。对象具有两个特性：一是状态；二是行为。状态是指对象本身的信息，也称为属性；行为是对对象的操作。通过对事物的抽象找出同一类对象的共同属性（静态特征）和行为（动态特征），从而得到类的概念。对象是类的一个具象，类是对象的一个抽象。

2.2.2 面向对象程序设计的特点

- 面向对象的程序设计有**“抽象”“封装”“继承”和“多态”**4个基本特点。
- 抽象：对象是系统中用来描述客观事物的一个实体，如各位员工是员工类的一个个对象。对象的特点包括两个方面：**属性和操作**。属性指的是描述对象静态特征的数据项，如员工的姓名、职位、薪水等，可以用变量来表示；操作指的是描述对象动态特征（即行为）的函数序列，也称为方法或服务，如员工可以请假、加班，员工还可以获得提拔、加薪等。
- C++中使用**对象名、属性和操作**三要素来描述对象。
- 封装：在C++中，通过用户定义的**类**来支持数据封装和信息隐藏。
- 继承：在C++现有类的基础上可以声明新的类，将一个已有类中的数据和函数保留，并加上自己特殊的数据和函数，从而构成一个新类，这就是继承和复用的思想。**原来的类是基类，也称为父类或超类。新类是派生类，也称为子类**。
- 多态：多态是指不同种类的对象都具有**名称相同**的行为，而具体行为的实现方式却有所不同。在一个类或多个类中，可以让多个方法使用同一个名字，从而具有**多态性**。这是通过**函数重载及运算符重载实现**的多态。

2.2.2 面向对象程序设计的特点

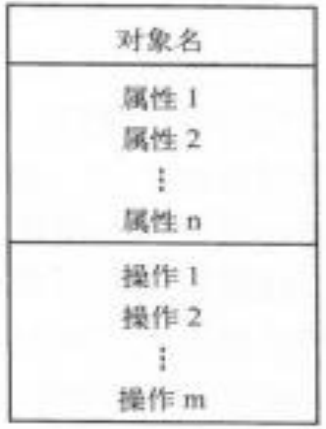


图 2-1　对象结构图

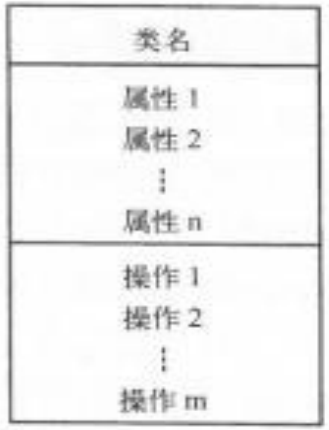


图 2-2　类模型结构图

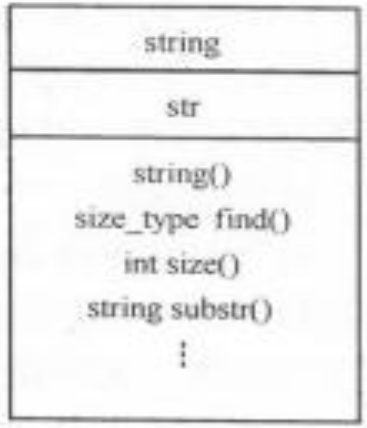
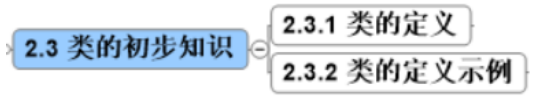


图 2-3　string 类模型结构示意图

2.3 类的初步知识



C++的基本数据类型	名称
bool	布尔型
char	字符型
int	整型
float	浮点型
double	双精度浮点型

课堂练习

关于对象概念的描述中，说法错误的是（　）。

A:对象就是C语言中的结构体

B:对象代表着正在创建的系统中的一个实体

C:对象是类的一个变量

D:对象之间的信息传递是通过消息进行的

课堂练习

关于对象概念的描述中，说法错误的是（　）。

A:对象就是C语言中的结构体

B:对象代表着正在创建的系统中的一个实体

C:对象是类的一个变量

D:对象之间的信息传递是通过消息进行的

答案：A

课堂练习

以下有关类与对象的叙述中，错误的是（　）。

A:对象是类的一个实例

B:一个类可以有多个对象

C:任何一个对象都归属于一个具体的类

D:只要是某个类的对象，那么该对象就可以访问这个类的所有成员

课堂练习

以下有关类与对象的叙述中，错误的是（　）。

A:对象是类的一个实例

B:一个类可以有多个对象

C:任何一个对象都归属于一个具体的类

D:只要是某个类的对象，那么该对象就可以访问这个类的所有成员

答案：D

课堂练习

下面关于对象概念的描述中错误的是（　）。

A:任何对象都必须有继承性

B:对象是属性和方法的封装体

C:对象间的通信靠消息传递

D:操作是对象的动态属性

课堂练习

下面关于对象概念的描述中错误的是（　）。

A:任何对象都必须有继承性

B:对象是属性和方法的封装体

C:对象间的通信靠消息传递

D:操作是对象的动态属性

答案：A

课堂练习

下列选项中，不属于类模型结构图中的是（　）。

A:类名

B:属性

C:操作

D:对象名

课堂练习

下列选项中，不属于类模型结构图中的是（　）。

A:类名

B:属性

C:操作

D:对象名

答案：D

课堂练习

关于封装，下列说法中不正确的是（　）

A:通过封装，对象的全部属性和操作结合在一起，形成一个整体

B:通过封装，一个对象的实现细节被尽可能地隐藏起来

C:通过封装，每个对象都成为相对独立的实体

D:通过封装，对象的属性都是不可见的

课堂练习

关于封装，下列说法中不正确的是（　）

A:通过封装，对象的全部属性和操作结合在一起，形成一个整体

B:通过封装，一个对象的实现细节被尽可能地隐藏起来

C:通过封装，每个对象都成为相对独立的实体

D:通过封装，对象的属性都是不可见的

答案：D

2.3.1 类的定义

- 类中的成员按功能划分，包括成员变量和成员函数；按访问权限划分，包括公有成员、私有成员和保护成员。
- 在C++中还可以定义不是任何类的成员的函数，这样的函数可称为“全局函数”。
- 成员函数既可以在类体内定义，也可以在类体外定义。如果成员函数定义在类体内部，则默认是内联函数。也可以在类体内部声明函数，并加上inline关键字，然后在类体外给出函数定义，这样的成员函数也是内联函数。

2.3.1 类的定义

名称	描述	代表
成员变量	是类中的一类成员，个数不限，也称为数据成员。成员变量的声明方式与普通变量的声明相同。	代表对象的“属性”。
成员函数	是类中的另一类成员，个数不限，其声明方式与普通函数的声明相同。	代表对该类对象所含数据进行操作的方法。

2.3.1 类的定义

- **标识符命名规则：字母、数字和下划线**的组合，**大小写敏感**，但不能以数字开头，也不能和系统中使用的关键字完全相同。
- **类**是具有唯一标识符的实体，就是说类名不能重复。类定义以“;”结束，大括号中的部分称为类体。
- **定义类时系统并不为类分配存储空间**，而只是把类看作是一种模板或样板。或者说，类可以看作是用户自定义的一种数据类型。在C++98标准下，类中声明的任何成员不能使用auto、extern和register关键字进行修饰。

课堂练习

按照标识符的要求，不能组成标识符的符号是（　　）。

A:连接符

B:下划线

C:大小写字母

D:数字字符

课堂练习

按照标识符的要求，不能组成标识符的符号是（　　）。

A:连接符

B:下划线

C:大小写字母

D:数字字符

答案：A

2.3.1 类的定义

- 如果成员函数定义在类体外，则类体内必须要有函数原型，**类体外函数定义**的前面必须用“类名::”来限定，格式如下：

```
返回值类型 类名::成员函数名(参数列表)
{
    成员函数的函数体
}
```

- 类名是成员函数所属类的名字，符号::是类作用域运算符，表明它后面的成员函数是属于类名标识的这个类的。返回值类型就是这个成员函数返回值的类型。
- **类C中不能定义类C的成员变量，但可以定义类C的指针和引用。**

2.3.2 类的定义示例


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

class myDate
{
public:
    myDate();                           //构造函数
    myDate(int, int, int);              //构造函数
    void setDate(int, int, int);        //设置日期
    void setDate(myDate);               //设置日期
    myDate getDate();                   //获取日期
    void setYear(int);                  //设置年
    int getMonth();                     //获取月
    void printDate() const;             //打印日期
private:
    int year, month, day;               //成员变量，表示年、月、日
};
```

```
//在类体外定义成员函数
myDate::myDate()
{
        year=1970,month=1,day=1;
}
myDate::myDate(int y, int m, int d)
{
       year=y;month=m;day=d;
}
void myDate::setDate(int y, int m, int d)
{
       year=y;month=m;day=d;
       return;
}
void myDate::setDate(myDate oneD)
{     year = oneD.year; month = oneD.month; day = oneD.day;
      return;
}
myDate myDate::getDate()
{     return *this;
}
void myDate::setYear(int y)
{
       year = y;
       return;
}
int myDate::getMonth()
{
       return month;
}
void myDate::printDate() const
{
       cout<<year <<"/"<<month<<"/"<<day;
       return;
}
```

```
class Student
{
public:
    void setStudent(string, myDate);  //设置学生信息
    void setName(string);             //设置姓名
    string getName();                 //获取姓名
    void setBirthday(myDate);         //设置生日
    myDate getBirthday();             //获取生日
    void printStudent() const;        //打印信息
private:
    string name;                      //姓名
    myDate birthday;                  //生日
};
//在类体外定义成员函数
void Student::setStudent(string s, myDate d)
{   name = s;
    birthday.setDate(d);
    return;}
void Student::setName(string n)
{   name = n;
    return;}
string Student::getName()
{   return name;}
void Student::setBirthday(myDate d)
{   birthday.setDate(d);
    return;}
myDate Student::getBirthday()
{   return birthday;}
void Student::printStudent() const
{   cout<<"姓名：  "<<name<<"\t生日：  ";
    birthday.printDate(); //调用类myDate的成员函数
    cout<<endl;
}

int main( )
{
    Student ss;
    int y,m,d;
    string name_;
    cout<<"请输入学生的姓名和生日，生日以\"年月日\“
            的次序输入:";
    cin>>name_>>y>>m>>d;
    ss.setStudent(name_, myDate(y, m, d));
    ss.printStudent();
    return 0;
}
```

课堂练习

下列关于C++标识符的命名不合法的是（　）

A:Pad

B:name_1

C:A#bc

D:_a12

课堂练习

下列关于C++标识符的命名不合法的是（　）

A:Pad

B:name_1

C:A#bc

D:_a12

答案：C

课堂练习

作用域运算符“::”的功能是（　）。

A:标识作用域的级别

B:指出作用域的范围

C:给定作用域的大小

D:标识成员是属于哪个类

课堂练习

作用域运算符“::”的功能是（　）。

A:标识作用域的级别

B:指出作用域的范围

C:给定作用域的大小

D:标识成员是属于哪个类

答案：B

课堂练习

已知类A中一个成员函数说明"void Set(A &a);"，其中A &a的含义是（　）。

A:指向类A的指针为a

B:将a的地址值赋给变量Set

C:a是类A的对象引用，用来作函数Set()的形参

D:变量A与a按位相与作为函数Set()的参数

课堂练习

已知类A中一个成员函数说明"void Set(A &a);"，其中A &a的含义是（　）。

A:指向类A的指针为a

B:将a的地址值赋给变量Set

C:a是类A的对象引用，用来作函数Set()的形参

D:变量A与a按位相与作为函数Set()的参数

答案：C

课堂练习

下列关于类的权限的描述错误的是（　）。

A:类本身的成员函数只能访问自身的私有成员

B:类的对象只能访问该类的公有成员

C:普通函数不能直接访问类的公有成员，必须通过对象访问

D:一个类可以将另一个类的对象作为成员

课堂练习

下列关于类的权限的描述错误的是（　）。

A:类本身的成员函数只能访问自身的私有成员

B:类的对象只能访问该类的公有成员

C:普通函数不能直接访问类的公有成员，必须通过对象访问

D:一个类可以将另一个类的对象作为成员

答案：A

课堂练习

下列关于类定义的叙述中，正确的是（　）。

A:类中可以定义成员变量，并赋初值

B:类中可以声明成员函数，并可以给出参数的默认值

C:类体不能为空，即必须定义成员变量及成员函数

D:类中仅能包含成员变量

课堂练习

下列关于类定义的叙述中，正确的是（　）。

A:类中可以定义成员变量，并赋初值

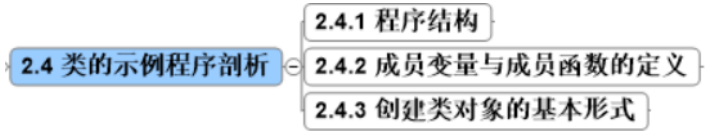
B:类中可以声明成员函数，并可以给出参数的默认值

C:类体不能为空，即必须定义成员变量及成员函数

D:类中仅能包含成员变量

答案：B

2.4 类的示例程序剖析



- 一个完整的C++程序包括以下几部分：
  - ✓ 一个主函数，可以调用其他函数，但不能被调用，也称为主程序。
  - ✓ 用户定义的任意多个的类及全局函数。
  - ✓ 全局说明。在所有函数和类定义之外的变量说明及函数原型。
  - ✓ 注释。
  - ✓ 头文件。
- 对于比较大的程序，根据主函数和各用户定义的类及全局函数的功能及相互关系，可以把类及全局函数划分为几个程序文件，包括.cpp文件和.h文件。.cpp文件是源程序文件，.h文件是头文件。
- 从逻辑关系上看，典型的C++程序的结构包括类的定义、类中成员函数的实现及主函数main。

2.4.2 成员变量与成员函数的定义

实现成员函数时要指明类的名称，在类体外定义的一般格式如下：

返回值类型 **类名::**函数成员名(参数表)

{

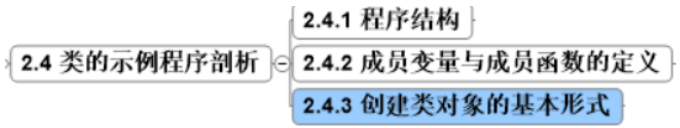
    函数体

}

成员函数并非每个对象各自存有一份。成员函数和普通函数一样，在内存中只有一份，它可以作用于不同的对象，为类中各对象共享。

通常，因为函数体代码较长，所以在类体内仅给出成员函数的原型，然后在类体外给出对应的函数体。如果函数体定义在类体内，则系统将其视为内联函数。类中定义的成员函数允许重载。

2.4.3 创建类对象的基本形式



方法一的基本格式如下：

类名 对象名;

或是

类名 对象名(参数);

或是

类名 对象名=类名(参数);

可以扩展为多个对象，如下所示:

类名 对象名1, 对象名2, ...;

或是

类名 对象名1(参数1), 对象名2(参数2), ...;

2.4.3 创建类对象的基本形式

方法二的基本格式如下：

类名 *对象指针名 = new 类名;

或是

类名 *对象指针名 = new 类名();

或是

类名 *对象指针名 = new 类名(参数);

用new创建对象时返回的是一个对象指针，这个指针指向本类刚创建的这个对象。C++分配给指针的仅仅是存储指针值的空间，而对象所占用的空间分配在堆上。使用new创建的对象，必须用delete来撤销。

2.4.3 创建类对象的基本形式

与基本数据类型一样，还可以声明对象的引用、对象的指针及对象的数组。

声明对象引用，即变量别名的基本格式如下:

类名 &对象引用名 = 对象;

声明对象指针，即指向对象的指针的基本格式如下:

类名 *对象指针名 = 对象的地址;

声明对象数组的格式如下:

类名 对象数组名[数组大小];

同类型的对象之间可以相互赋值。对象和对象指针都可以用作函数参数。函数的返回值可以是对象或指向对象的指针。

例如，定义了类C后，可以有如下的声明:

```
C a1,b1;          //定义了C类的对象a1和b1
C *p = &a1;       //定义了指向对象a1的C类类型的指针p
C &R = b1;        //定义了C类类型对象b1的引用R
C A[3];           //定义了C类类型对象的数组A，含3个元素
```

真题演练

下列关于对象数组的描述中，错误的是（　）

A:对象数组的下标是从0开始的

B:对象数组的数组名是个常量指针

C:对象数组的每个元素是同一个类的对象

D:对象数组只能赋初值，不能被赋值

真题演练

下列关于对象数组的描述中，错误的是（　）

A:对象数组的下标是从0开始的

B:对象数组的数组名是个常量指针

C:对象数组的每个元素是同一个类的对象

D:对象数组只能赋初值，不能被赋值

答案：D

真题演练

下列关于类和对象的说法中，正确的是（　）

A:编译器为每个类和类的对象分配内存

B:类的对象具有成员函数的副本

C:类的成员函数由类来调用

D:编译器为每个对象的成员变量分配内存

真题演练

下列关于类和对象的说法中，正确的是（　）

A:编译器为每个类和类的对象分配内存

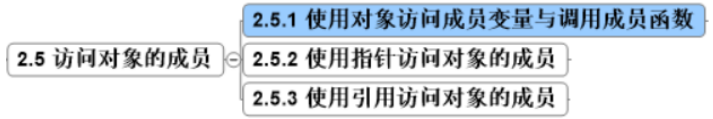
B:类的对象具有成员函数的副本

C:类的成员函数由类来调用

D:编译器为每个对象的成员变量分配内存

答案：D

2.5 访问对象的成员



定义了类和对象后，就可以访问对象的成员。通过对象访问成员变量的一般格式如下：

对象名.成员变量名

调用成员函数的一般格式如下：

对象名.成员函数名(参数表)

2.5.1使用对象访问成员变量与调用成员函数

```
int main( )
{
    Student ss;
    int y,m,d;
    string name_;
    cout<<"请输入学生的姓名和生日，生日以\"年月日\"的次序输入:";
    cin>>name_>>y>>m>>d;
    ss.setStudent(name_, myDate(y, m, d));
    ss.printStudent();
    return 0;
}
```

```
请输入学生的姓名和生日，生日以"年月日"的次序输入:张三 1999 1 1
姓名：张三        生日：1999/1/1
```

2.5.2 使用指针访问对象的成员

除了“对象名.成员名”的格式外，还可以使用指针或引用的方式来访问类成员。如果是通过指针访问成员变量，则点运算符.换为箭头运算符->，即使用“指针->成员名”的方式来访问对象的成员。例如，将程序2-3中的主函数修改如下：

```
int main( )
{
    Student ss;
    int y, m, d;
    string name_;
    Student *sp = &ss;                          //指向 ss 的指针sp
    cout<<"请输入学生的姓名和生日，生日以\"年月日\"的次序输入： ";
    cin >>name_>>y>>m>>d;
    sp->setStudent(name_,myDate(y,m,d));
    sp->printStudent( );
    return 0;
}
```

```
请输入学生的姓名和生日，生日以"年月日"的次序输入：张三  1997 1 1
姓名：张三      生日：1997/1/1
```

2.5.3 使用引用访问对象的成员


```
int main( )
{
    Student ss;
    int y,m,d;
    string name_;
    Student &sy=ss;                          //指向 ss 的指针sp
    cout<<"请输入学生的姓名和生日，生日以\"年月日\"的次序输入： ";
    cin>>name_>>y>>m>>d;
    sy.setStudent(name_,myDate(y,m,d));
    sy.printStudent( );
    return 0;
}
```

程序中定义了引用sy并进行初始化后，sy与ss成为同一个对象的两个不同的名字。访问成员时仍使用点操作符，即“引用名.成员名”。

真题演练

已知：p是一个指向类A数据成员m的指针，A1是类A的一个对象。如果要给m赋值为5，正确的是（　）。

A:A1.p=5;

B:A1->p=5;

C:A1.*p=5;

D:*A1.p=5;

真题演练

已知：p是一个指向类A数据成员m的指针，A1是类A的一个对象。如果要给m赋值为5，正确的是（　）。

A:A1.p=5;

B:A1->p=5;

C:A1.*p=5;

D:*A1.p=5;

答案：D

真题演练

下列访问成员的方式哪个是正确的（　）。

A:对象名.对象成员名

B:对象指针名.对象成员名

C:对象名->对象成员名

D:类名->对象成员名

真题演练

下列访问成员的方式哪个是正确的（　）。

A:对象名.对象成员名

B:对象指针名.对象成员名

C:对象名->对象成员名

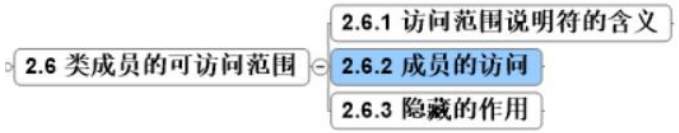
D:类名->对象成员名

答案：A

2.6 类成员的可访问范围

访问范围说明符	含义	作用
public	公有的	使用它修饰的类的成员可以在程序的任何地方被访问。
private	私有的	使用它修饰的类的成员仅能在本类内被访问。
protected	保护的	它的作用介于public与private之间，使用它修饰的类的成员能在本类内及子类中被访问。

私有类型的成员在类外是不能访问的，通过公有函数访问的效率比直接访问的效率要低。为了权衡这两方面的因素，C++提供了友元访问方式。只有在类内和在友元函数内才可以访问私有成员。



课堂练习

以下叙述中正确的是（　）。

A:类成员的定义必须放在类体内部

B:在类中，不作特别说明的数据成员均为私有类型

C:在类中，不作特别说明的数据成员均为公有类型

D:类成员的定义必须是成员变量之前，成员函数在后

课堂练习

以下叙述中正确的是（　）。

A:类成员的定义必须放在类体内部

B:在类中，不作特别说明的数据成员均为私有类型

C:在类中，不作特别说明的数据成员均为公有类型

D:类成员的定义必须是成员变量之前，成员函数在后

答案：B

例如：有如下类声明“class A{int x；…};”，则A类的成员x是私有数据成员。

2.6.2 成员的访问

下列关于类中定义的成员作用域的叙述中，正确的是（　）。

A.使用public修饰的成员，仅可以在本类内及主函数内访问

B.使用private修饰的成员，可以在本类内及主函数内访问

C.使用protected修饰的成员，可以在本类内及其子类内访问

D.没有使用访问修饰符的成员，可以在本类内及其子类内访问

2.6.2 成员的访问

下列关于类中定义的成员作用域的叙述中，正确的是（　）。

A.使用public修饰的成员，仅可以在本类内及主函数内访问

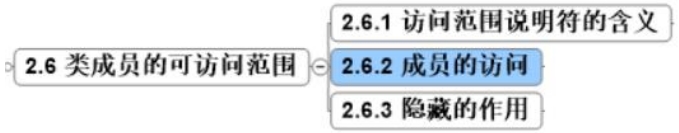
B.使用private修饰的成员，可以在本类内及主函数内访问

C.使用protected修饰的成员，可以在本类内及其子类内访问

D.没有使用访问修饰符的成员，可以在本类内及其子类内访问

答案：C

2.6.2 成员的访问



下列关于类的权限描述错误的是（　）。

A:类本身的成员函数可以访问自己定义的任何成员

B:类的对象只能访问公有成员

C:普通函数只能通过对象访问类的公有成员

D:一个类不能包含另一个类的对象作为成员

2.6.2 成员的访问

下列关于类的权限描述错误的是（　）。

A:类本身的成员函数可以访问自己定义的任何成员

B:类的对象只能访问公有成员

C:普通函数只能通过对象访问类的公有成员

D:一个类不能包含另一个类的对象作为成员

答案：D

2.6.2 成员的访问

类的私有成员可在何处访问（　）。

A:通过子类的对象访问

B:本类及子类的成员函数中

C:通过该类对象访问

D:本类的成员函数中

2.6.2 成员的访问

类的私有成员可在何处访问（ ）。

A:通过子类的对象访问

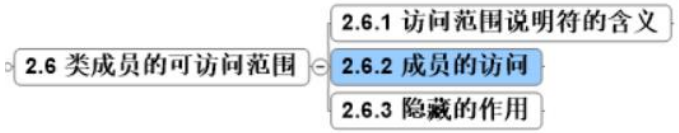
B:本类及子类的成员函数中

C:通过该类对象访问

D:本类的成员函数中

答案：D

2.6.2 成员的访问



只能在自身类和子类成员函数中被访问，无法通过对象在类外访问的成员属于（　）。

A:private

B:protected

C:public

D:publish

2.6.2 成员的访问

只能在自身类和子类成员函数中被访问，无法通过对象在类外访问的成员属于（　）。

A:private

B:protected

C:public

D:publish

答案：B

2.6.2 成员的访问

以下叙述中正确的是（　）。

A:类成员的定义必须放在类体内部

B:在类中，不作特别说明的数据成员均为私有类型

C:在类中，不作特别说明的数据成员均为公有类型

D:类成员的定义必须是成员变量之前，成员函数在后

2.6.2 成员的访问

以下叙述中正确的是（　）。

A:类成员的定义必须放在类体内部

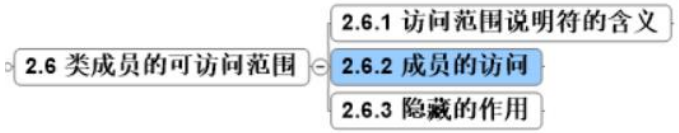
B:在类中，不作特别说明的数据成员均为私有类型

C:在类中，不作特别说明的数据成员均为公有类型

D:类成员的定义必须是成员变量之前，成员函数在后

答案：B

2.6.2 成员的访问



在类定义的外部，可以被访问的成员有（　）。

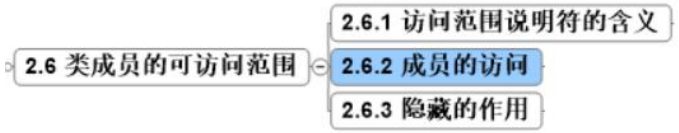
A:所有类成员

B:private或protected的类成员

C:public的类成员

D:public或private的类成员

2.6.2 成员的访问



在类定义的外部，可以被访问的成员有（　）。

A:所有类成员

B:private或protected的类成员

C:public的类成员

D:public或private的类成员

答案：C

```
#include <iostream>
using namespace std;
class Box
{
public:                          //公有的
    double length;
    void   setWidth(double wid);
    double getWidth();
private:                         //私有的
    double width;
};
//类体外定义成员函数
double Box::getWidth()
{           return width ;
}
void Box::setWidth(double wid)
{           width=wid;
}
```

```
int main()
{   Box box;                              //声明一个对象
    //不使用成员函数设置长度
    box.length=10.0;                      //正确，因为 length 是公有的
    cout<<"Length of box："<<box.length<<endl; //输出Length of box：10
    //不使用成员函数设置宽度
    //box.width=10.0;                     //错误，因为width是私有的
    box.setWidth(10.0);                   //必须使用成员函数设置宽度
    cout<<"Width of box："<<box.getWidth()<<endl; //输出Width of box：10
    return 0;
}
```

```
Length of box: 10
Width of box: 10
```

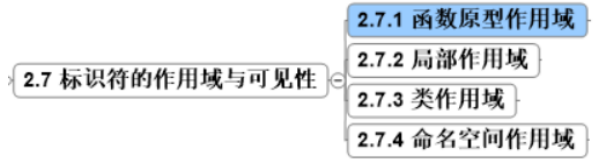
```
#include <iostream>
#include <string>
using namespace std;
class CEmployee
{
private:
    string szName;              //姓名
    int salary;                 //工资
public:
    void setName(string);       //设置姓名
    string getName();           //获取姓名
    void setSalary(int);        //设置工资
    int getSalary();            //获取工资
    int averageSalary(CEmployee);   //计算两人的平均工资
};
void CEmployee::setName(string name)
{           szName = name;}
string CEmployee::getName()
{           return szName;}
void CEmployee::setSalary(int mon)
{           salary = mon;}
int CEmployee::getSalary()
{           return salary;}
int CEmployee::averageSalary(CEmployee e1)
{           return ( salary + e1 .getSalary() )/2;
}
int main()
{  CEmployee eT, eY;
    //eT.szName ="Tom1234567";
    eT.setName("Tom1234567");
    //eT.salary=5000; //编译错误，不能直接访问私有成员
    eT.setSalary(5000); //需要通过公有成员函数访问
    cout<<eT.getName()<<endl; //输出 Tom1234567
    eY.setName("Yong7654321");
    eY.setSalary(3500);
    cout<<eY.getName()<<endl; //输出 Yong7654321
    cout<<"aver="<<eT.averageSalary(eY)<<endl; return 0;
}
```

```
Tom1234567
Yong7654321
aver=4250
```

2.6.3 隐藏的作用

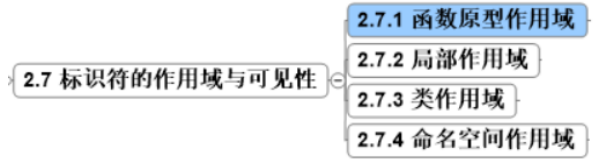
设置私有成员的机制叫作“隐藏”。“隐藏”的一个目的就是强制对私有成员变量的访问一定要通过公有成员函数进行。这样做的好处是：如果以后修改了成员变量的类型等属性，只需要更改成员函数即可；否则，所有直接访问成员变量的语句都需要修改。

2.7 标识符的作用域与可见性



- 标识符是组成程序的最小成分之一。类名、函数名、变量名、常量名和枚举类型的取值等都是标识符。这些标识符有各自的作用域和可见性。标识符的作用域是指标识符的有效范围，即它在程序中的存在区域。标识符的可见性是指在程序的哪个区域里可以使用。对于同一个标识符来说，这两个区域可能是不完全重合的。
- C++中标识符的作用域有：函数原型作用域、局部作用域（块作用域）、类作用域和命名空间作用域。

2.7.1 函数原型作用域



- 在声明函数原型时形参的作用范围就是函数原型作用域，这是C++程序中最小的作用域。例如，有如下的函数声明：

  double area(double radius);

- 标识符radius的作用范围就在函数area形参列表的左右括号之间，在程序的其他地方不能引用这个标识符。因为函数声明中形参仅在形参列表中有效，所以，函数声明中往往不写形参名，而仅写形参的类型。作用域。

2.7.2 局部作用域

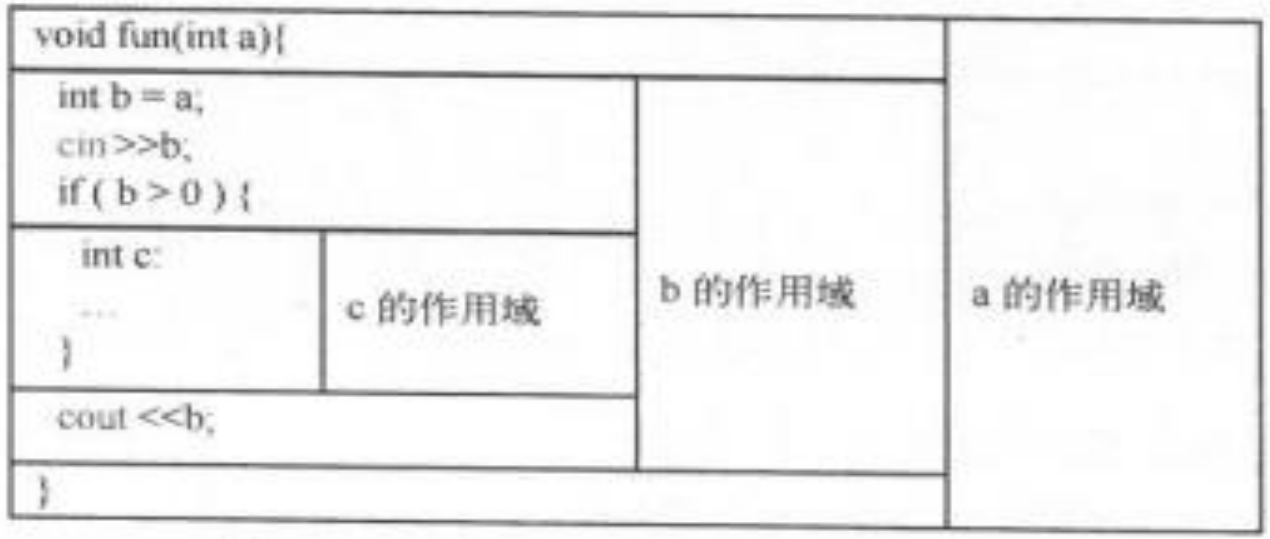


图 2-4　函数 fun()中 3 个变量的作用域

2.7.3 类作用域

类可以被看成是一组有名字的成员的集合，类X的成员m具有类作用域，对m的访问方式有如下3种：

1)如果在类X的成员函数中没有声明同名的局部作用域标识符，那么在该函数内可以直接访问成员m。也就是说，m在这样的函数中起作用。

2)在类外，可以通过表达式x.m或者X::m来访问，其中x是类X的对象。这正是程序中访问对象成员的最基本方法。当然，这样的访问不能违反m的访问修饰符的限定。

3)在类外，可以通过ptr->m这样的表达式来访问，其中ptr为指向类X的一个对象的指针。当然，这样的访问不能违反m的访问修饰符的限定。

2.7.4 命名空间作用域

定义命名空间的一般形式如下：

```
namespace 命名空间名
{
    命名空间内的各种声明（函数声明、类声明、...）
}
```

在命名空间内部可以直接引用当前命名空间中声明的标识符，如果需要引用其他命名空间的标识符，需要使用下面的方式：

命名空间名::标识符名

2.7.4 命名空间作用域

例如，定义一个命名空间如下:

```
namespace SomeNs
{
    class SomeClass{……};
    someFunc(int a){ }
};
```

如果需要引用类名SomeClass或者函数名someFunc()，需要使用下面的方式:

```
SomeNs::SomeClass obj1;        //声明一个SomeNS::SomeClass型的对象obj1
SomeNs::someFunc(5);           //调用someFunc()函数
```

在标识符前面总要加上这样的命名空间限定会显得过于冗长，为了解决这一问题，C++又提供了using语句，using语句有两种形式:

```
using命名空间名::标识符名;
using namespace命名空间名;
```

真题演练

局部变量可以隐藏全局变量，那么在有同名全局变量和局部变量的情形时，可以用下列哪一项提供对全局变量的访问（　）

A:作用域运算符

B:指针运算符

C:提取运算符

D:插入运算符

真题演练

局部变量可以隐藏全局变量，那么在有同名全局变量和局部变量的情形时，可以用下列哪一项提供对全局变量的访问（　）

A:作用域运算符

B:指针运算符

C:提取运算符

D:插入运算符

答案：A

作用域隐藏规则

具有命名空间作用域的变量也称为全局变量。

对于在不同的作用域声明的标识符，可见性的一般原则如下：

- 标识符要声明在前，引用在后。
- 在同一个作用域中，不能声明同名的标识符。在没有互相包含关系的不同作用域中声明的同名标识符，互不影响。
- 如果存在两个或多个具有包含关系的作用域，外层声明了一个标识符，而内层没有再次声明同名标识符，那么外层标识符在内层仍然可见。如果在内层声明了同名标识符，则外层标识符在内层不可见，这时称内层标识符隐藏了外层同名标识符，这种机制称为隐藏规则。

作用域隐藏规则


```
#include <iostream>
using namespace std;
int main()
{
    int a=1;
    cout<<a<<“\n”;                  //输出 1
    for(int i=1; i<2; i++)
    {
        int a=2;
        cout<<a<<"\n";              //输出 2
    }
    cout<<a<<“\n”;                  //输出 1
    return 0;
}
```

【程序说明】程序2-8中，主函数中有嵌套的两个块。外层块中定义了变量a，赋初值1。内层块（for循环）中也定义了变量a，赋初值2。这两个变量互相独立。在for循环中，外层定义的变量a将不可见。

本章总结


第二章 面向对象的基本概念
2.1 结构化程序设计
2.2 面向对象程序设计的概念和特点
2.2.1 面向对象思想的提出
2.2.2 面向对象程序设计的特点
2.3 类的初步知识
2.3.1 类的定义
2.3.2 类的定义示例
2.4 类的示例程序剖析
2.4.1 程序结构
2.4.2 成员变量与成员函数的定义
2.4.3 创建类对象的基本形式
2.5 访问对象的成员
2.5.1 使用对象访问成员变量与调用成员函数
2.5.2 使用指针访问对象的成员
2.5.3 使用引用访问对象的成员
2.6 类成员的可访问范围
2.6.1 访问范围说明符的含义
2.6.2 成员的访问
2.6.3 隐藏的作用
2.7 标识符的作用域与可见性
2.7.1 函数原型作用域
2.7.2 局部作用域
2.7.3 类作用域
2.7.4 命名空间作用域

祝大家顺利通过考试！