编辑是将源程序输入到计算机中,生成后缀为.cpp 的磁盘文件。编译是将程序的源代码转换为机器语言代码。但是编译后的程序还不能由计算机执行,还需要连接。连接是将多个目标文件以及库中的某些文件连在一起,生成一个后缀为.exe 的可执行文件。最后,还要对程序进行运行和调试。

在编译和连接时,都会对程序中的错误进行检查,并将查出的错误显示在屏幕上。编译阶段查出的错误是语法错,连接时查出的错误称连接错。

1.6 小结

语言是一套具有语法、词法规则的系统。语言是思维的工具,思维是通过语言来表述的。计算机程序设计语言是计算机可以识别的语言,用于描述解决问题的方法,供计算机阅读和执行。计算机语言可以分为机器语言、汇编语言、高级语言和面向对象的语言。在本书中我们要学习的 C++ 语言就是使用最为广泛的面向对象语言。

面向对象的软件工程是面向对象方法在软件工程领域的全面应用。它包括面向对象的分析(OOA)、面向对象的设计(OOD)、面向对象的编程(OOP)、面向对象的测试(OOT)和面向对象的软件维护(OOSM)等主要内容。

计算机加工的对象是数据信息,而指挥计算机操作的是控制信息。所有的信息在计算机内部都是用二进制数表示的,具体的表示方式根据信息的类型有所不同,这些不同的表示方式也是本章的内容之一。

习 题

1-1 简述计算机程序设计语言的发展历程。

1-2 面向对象的编程语言有哪些特点?

1-3 什么是结构化程序设计方法?这种方法有哪些优点和缺点?

1-4 什么是对象?什么是面向对象方法?这种方法有哪些特点?

1-5 什么叫做封装?

1-6 面向对象的软件工程包括哪些主要内容?

1-7 计算机内部的信息可分为几类?简述之。

1-8 什么叫做二进制?使用二进制有哪些优点和缺点?

1-9 请将以下十进制数值转换为二进制和十六进制补码。

(1) 2　　(2) 9　　(3) 93

(4) −32　　(5) 65 535　　(6) −1

1-10 请将以下数值转换为十进制:

(1) $(1010)_2$　　(2) $(10001111)_2$　　(3) $(0101111111000011)_2$

(4) $(7F)_{16}$　　(5) $(2D3E)_{16}$　　(6) $(F10E)_{16}$

1-11 简要比较原码、反码、补码等几种编码方法。

能一步一步执行,每步只能执行一次运算。在运算过程中寄存器发挥了重要作用,不仅需要用寄存器存储变量值,还需要保存每一步运算的中间结果。

C++ 分支语句和循环语句的执行原理,涉及条件跳转,这在 IA-32 中涉及状态字,比较复杂,本书不再加以介绍,感兴趣的读者可以参考介绍 Intel 微处理器汇编语言的相关书籍。

2.7 小结

C++ 是从 C 语言发展演变而来的。C 语言最初是贝尔实验室的 Dennis Ritchie 于 1972 年在 B 语言基础上开发出来的,它具有很多优点:语言简洁灵活、运算符和数据结构丰富、具有结构化控制语句、程序执行效率高,而且同时具有高级语言与汇编语言的优点。

C++ 便是在 C 语言基础上为支持面向对象的程序设计而研制的一个通用目的的程序设计语言,它是在 1980 年由 AT&T 贝尔实验室的 Bjarne Stroustrup 博士创建的。C++ 语言的主要特点表现在两个方面,一是全面兼容 C,二是支持面向对象的方法。

数据是程序处理的对象,数据可以依其本身的特点进行分类。C++ 中的数据类型又分为基本类型和自定义类型,基本类型是 C++ 编译系统内置的。C++ 的基本数据类型有 bool(布尔型)、char(字符型)、int(整型)、float(浮点型,表示实数)、double(双精度浮点型,简称双精度型)等。除了 bool 型外,主要有两大类:整数和浮点数。因为 char 型从本质上说也是整数类型,它是长度为 1 字节的整数,通常用来存放字符的 ASCII 码。C++ 语言不仅有丰富的内置基本数据类型,而且允许用户自定义类型。自定义数据类型有:枚举类型、结构类型、联合类型、数组类型、类类型等。本章介绍了枚举类型,而类类型是 C++ 面向对象程序设计的核心,是本书后面章节的主要内容,其他类型也将在后面介绍。本章还介绍了数据输入输出的基本方法。

程序设计工作主要包括数据结构和算法的设计。算法要由一系列控制结构组成,顺序、选择和循环结构是程序设计中最基本的控制结构,也是构成复杂算法的基础。

习 题

2-1 C++ 语言有哪些主要特点和优点?

2-2 下列标识符哪些是合法的?

Program, -page, _lock, test2, 3in1, @mail, A_B_C_D

2-3 例 2-1 中每条语句的作用是什么?

2-4 请用 C++ 语句声明一个常量 PI,值为 3.1416;再声明一个浮点型变量 a,把 PI 的值赋给 a。

2-5 在下面的枚举类型中,BLUE 的值是多少?

```
enum Color {WHITE, BLACK=100, RED, BLUE, GREEN=300};
```

2-6　注释有什么作用？C++ 中有哪几种注释的方法？它们之间有什么区别？

2-7　什么叫做表达式？x=5+7 是一个表达式吗？它的值是多少？

2-8　下列表达式的值是多少？

(1) 201/4

(2) 201%4

(3) 201/4.0

2-9　执行完下列语句后，a,b,c 三个变量的值为多少？

```
a=30;
b=a++;
c=++a;
```

2-10　在一个 for 语句中，可以给多个变量赋初值吗？如何实现？

2-11　执行完下列语句后，n 的值为多少？

```
int n;
for (n=0; n<100; n++);
```

2-12　写一条 for 语句，计数条件为 n 从 100 到 200，步长为 2；然后用 while 和 do…while 循环完成同样的循环。

2-13　if(x=3)和 if(x==3)这两条语句的差别是什么？

2-14　已知 x,y 两个变量，写一条简单的 if 语句，把较小的值赋给原本值较大的变量。

2-15　修改下面这个程序中的错误，改正后它的运行结果是什么？

```
#include <iostream>
using namespace std;
int main()
    int i
    int j;
    i=10;                          /*给 i 赋值
    j=20;                          /*给 j 赋值*/
    cout<<"i+j=<<i+j;              /*输出结果*/
    return 0;
}
```

2-16　编写一个程序，运行时提示输入一个数字，再把这个数字显示出来。

2-17　C++ 有哪几种数据类型？简述其值域。编程显示你使用的计算机中的各种数据类型的字节数。

2-18　输出 ASCII 码为 32～127 的字符。

2-19　运行下面的程序，观察其输出，与你的设想是否相同？

```
#include <iostream>
using namespace std;
int main() {
    unsigned int x;
```

```
    unsigned int y=100;
    unsigned int z=50;
    x=y-z;
    cout<<"Difference is: "<<x<<endl;
    x=z-y;
    cout<<"\nNow difference is: "<<x <<endl;
    return 0;
}
```

2-20 运行下面的程序,观察其输出,体会 i++与++i 的差别。

```
#include <iostream>
using namespace std;
int main() {
    int myAge=39;                    //定义并初始化变量
    int yourAge=39;
    cout<<"I am: "<<myAge<<" years old. "<<endl;
    cout<<"You are: "<<yourAge<<" years old. "<<endl;
    myAge++;
    ++yourAge;
    cout<<"One year passes…"<<endl;
    cout<<"I am: "<<myAge<<" years old. "<<endl;
    cout<<"You are: "<<yourAge<<"years old."<<endl;
    cout<<"Another year passes."<<endl;
    cout<<"I am: "<<myAge++<<" years old. "<<endl;
    cout<<"You are: "<<++yourAge<<" years old. "<<endl;
    cout<<"Let's print it again."<<endl;
    cout<<"I am: "<<myAge<<"years old. "<<endl;
    cout<<"You are: "<<yourAge<<"years old."<<endl;
    return 0;
}
```

2-21 什么叫常量?什么叫变量?

2-22 变量有哪几种存储类型?

2-23 写出下列表达式的值:

(1) 2<3 && 6<9

(2) !(4<7)

(3) !(3>5) || (6<2)

2-24 若 a=1,b=2,c=3,下列各式的结果是什么?

(1) a|b-c

(2) a^b&-c

(3) a&b|c

(4) a|b&c

2-25 若 a=1,下列各式的结果是什么?

(1) !a|a

(2) ~a|a

(3) a^a

(4) a>>2

2-26 编写一个完整的程序，实现功能：向用户提问“现在正在下雨吗?”，提示用户输入 Y 或 N。若输入为 Y，显示“现在正在下雨。”；若输入为 N，显示“现在没有下雨。”；否则继续提问“现在正在下雨吗?”。

2-27 编写一个完整的程序，运行时向用户提问“你考试考了多少分? (0～100)”，接收输入后判断其等级显示出来。规则如下：

$$\text{等级}=\begin{cases}\text{优} & 90\leqslant\text{分数}\leqslant 100\\ \text{良} & 80\leqslant\text{分数}<90\\ \text{中} & 60\leqslant\text{分数}<80\\ \text{差} & 0\leqslant\text{分数}<60\end{cases}$$

2-28 实现一个简单的菜单程序，运行时显示“Menu: A(dd) D(elete) S(ort) Q(uit), Select one:”提示用户输入。A 表示增加，D 表示删除，S 表示排序，Q 表示退出。输入为 A、D、S 时分别提示“数据已经增加、删除、排序。”，输入为 Q 时程序结束。

(1) 要求使用 if…else 语句进行判断，用 break，continue 控制程序流程。

(2) 要求使用 switch 语句。

2-29 用穷举法找出 1～100 间的质数并显示出来。分别使用 while，do…while，for 循环语句实现。

2-30 比较 break 语句与 continue 语句的不同用法。

2-31 声明一个表示时间的结构体，可以精确表示年、月、日、小时、分、秒；提示用户输入年、月、日、小时、分、秒的值，然后完整地显示出来。

2-32 在程序中定义一个整型变量，赋予 1～100 的值。要求用户猜这个数，比较两个数的大小，把结果提示给用户，直到猜对为止。分别使用 while，do…while 语句实现循环。

2-33 声明枚举类型 Weekday，包括 SUNDAY 到 SATURDAY 七个元素在程序中声明 Weekday 类型的变量，对其赋值，声明整型变量，看看能否对其赋 Weekday 类型的值。

2-34 口袋中有红、黄、蓝、白、黑 5 种颜色的球若干个。每次从口袋中取出 3 个不同颜色的球，问有多少种取法?

2-35 输出九九乘法算表。

2-36 有符号整数和无符号整数，在计算机内部是如何区分的?

```
    double s=add(1, 2);
    ...
    return 0;
}

double add(double a, double b) {
    return a+b;
}
```

该程序中的 double add()就是对 add 函数的不完整声明。

细节 与 C++ 不同的是，在 C 语言中，声明函数时，括号内为空，并不表示这个函数没有任何参数，而表示它所要求的参数是未知的。如果要声明一个没有参数的函数，应当在括号中写入 void。

这个程序看起来好像没什么问题，因为在调用 add 函数时，的确为它提供了两个参数，而且这两个参数都可以被转化成 double 型，可问题恰好就出在参数类型上。由于声明中没有给出参数的类型，所以编译器在编译 add(1, 2)时，不会对 1 和 2 进行类型转换，它们是被作为整型数据压入运行栈中的，函数执行的结果自然就不正确。如果对 add 函数有完整的声明，这样的错误就不会发生，因为在编译时两个参数 1 和 2 都会被自动转换成 double 型数据。

读者可能会问，这里虽然在 add(1, 2)前没有函数 add 的完整声明，但在同一源文件中能找到函数的定义，因此对编译器加以改进，还是能够避免这一问题的。但规模稍大的 C++ 程序，常常是多文件结构的，编译器会对每个单元分别处理，最后连接，如果 add 函数和 main 函数在不同的文件中，这一问题仍然无法避免。当学习到第 5 章的多文件结构时，相信读者会对这一问题有更深刻的认识。

通过对这个 C 语言反例的分析，希望读者能够对函数声明的意义有更深刻的认识。

3.7 小结

在面向对象的程序设计中，函数是功能抽象的基本单位。

一个较为复杂的系统往往需要划分为若干子系统，分别进行开发和调试。高级语言中的子程序就是用来实现这种模块划分的。C++ 语言中的子程序体现为函数，对对象的功能抽象也要借助于函数。函数编写好以后，可以被重复使用，使用时可以只关心函数的功能和使用方法而不必关心函数功能的具体实现。这样有利于代码重用，可以提高开发效率、增强程序的可靠性，也便于分工合作，便于修改维护。

一个 C++ 程序可以由一个主函数和若干子函数构成。主函数是程序执行的开始点，由主函数调用子函数，子函数还可以再调用其他子函数。

函数的重载使得具有类似功能的不同函数可以使用同一名称，这样便于使用，也能增加程序的可读性。重载函数是按照形参来区分的，同名的重载函数其形参类型或个数必须不同。

另外，C++ 的系统库中还提供了几百个函数可供程序员使用。系统函数的原型声明

已经全部由系统提供了,分类存在于不同的头文件中。程序员需要用include指令嵌入相应的头文件,然后便可以使用系统函数。

习 题

3-1 C++中的函数是什么?什么叫主调函数?什么叫被调函数?二者之间有什么关系?如何调用一个函数?

3-2 观察下面程序的运行输出,与你设想的有何不同?仔细体会引用的用法。

```
#include <iostream>
using namespace std;
int main() {
    int intOne;
    int &rSomeRef=intOne;

    intOne=5;
    cout<<"intOne:\t"<<intOne<<endl;
    cout<<"rSomeRef:\t"<<rSomeRef<<endl;

    int intTwo=8;
    rSomeRef=intTwo;
    cout<<"\nintOne:\t"<<intOne<<endl;
    cout<<"intTwo:\t"<<intTwo<<endl;
    cout<<"rSomeRef:\t"<<rSomeRef<<endl;

    return 0;
}
```

3-3 比较值传递和引用传递的相同点与不同点。

3-4 什么叫内联函数?它有哪些特点?

3-5 函数原型中的参数名与函数定义中的参数名以及函数调用中的参数名必须一致吗?

3-6 调用被重载的函数时,通过什么来区分被调用的是哪个函数?

3-7 完成函数。参数为两个unsigned short int型数,返回值为第一个参数除以第二个参数的结果,数据类型为short int;如果第二个参数为0,则返回值为-1。在主程序中实现输入输出。

3-8 编写函数把华氏温度转换为摄氏温度,公式为

$$C=\frac{5}{9}(F-32)$$

在主程序中提示用户输入一个华氏温度,转化后输出相应的摄氏温度。

3-9 编写函数判别一个数是否是质数,在主程序中实现输入输出。

3-10 编写函数求两个整数的最大公约数和最小公倍数。

3-11 什么叫做嵌套调用?什么叫做递归?

3-12　在主程序中提示输入整数 n，编写函数用递归的方法求 $1+2+\cdots+n$ 的值。

3-13　用递归的方法编写函数求 Fibonacci 级数，公式为

$$F_n = F_{n-1} + F_{n-2}(n > 2), \quad F_1 = F_2 = 1$$

观察递归调用的过程。

3-14　用递归的方法编写函数求 n 阶勒让德多项式的值，在主程序中实现输入输出。递归公式为

$$p_n(x) = \begin{cases} 1 & (n = 0) \\ x & (n = 1) \\ [(2n-1)x \cdot p_{n-1}(x) - (n-1)p_{n-2}(x)]/n & (n > 1) \end{cases}$$

3-15　编写递归函数 getPower 计算 x^y，在同一个程序中针对整型和实型实现两个重载的函数：

```
int getPower(int x, int y);                    //整型形式,当 y<0 时,返回 0
double getPower(double x, int y);              //实型形式
```

在主程序中实现输入输出，分别输入一个整数 a 和一个实数 b 作为底数，再输入一个整数 m 作为指数，输出 a^m 和 b^m。另外请读者思考，如果在调用 getPower 函数计算 a^m 时希望得到一个实型结果（实型结果表示范围更大，而且可以准确表示 $m<0$ 时的结果），该如何调用？

3-16　当函数发生递归调用时，同一个局部变量在不同递归深度上可以同时存在不同的取值，这在底层是如何做到的？

制构造函数的调用也是可以省去的，例如，如果把 fun2 改写为：

```
Point fun2() {
    return Point(1, 2);
}
```

最直接的实现方式是，先调用 Point 的构造函数 Point(int, int)生成一个 fun2 内的临时对象，再以这个临时对象为参数调用 Point 的复制构造函数，生成返回值，这两步可以简化为一步，即用构造函数 Point(int, int)直接构造出返回值，就是下面这样的形式：

```
void _fun2(Point &result) {
    result.Point(1, 2);
}
```

另一方面，在主调函数中，也未必一定要为返回值生成临时对象。例如，如果主调函数是这样调用 fun2 的：

```
Point p=fun2();
```

这时不必为返回值生成临时对象，而可以直接用对象 p 的空间存储返回值，就是这样：

```
Point p;          //这里只分配空间，不调用构造函数
_fun2(p);         //p的构造函数会在_fun2函数中被调用
```

这能够省去一次复制构造函数的调用。前面提到过，如果把表示返回值的临时对象放在运行栈的传参区域上，不利于有些优化，指的就是这一项优化。按照那种实现方式，返回值的存放位置不由主调函数决定，也就无法直接将返回值存在对象 p 的空间中。

通过本节的分析，应当能够看出，复制构造函数的调用次数，会因编译器的优化程度而有所差异，因此复制构造函数中一定要只完成对象复制的任务，而不要有其他能产生副作用的操作，否则程序运行结果会因编译器的优化程度而有所差异。

4.9 小结

面向对象程序设计通过抽象、封装、继承和多态使程序代码达到最大限度的可重用和可扩展，提高软件的生产能力，控制软件开发和维护的费用。类是面向对象程序设计的核心，利用它可以实现数据的封装、隐蔽，通过它的继承与派生，能够实现对问题深入地抽象描述。

类是逻辑上相关的函数与数据的封装，它是对所要处理的问题的抽象描述。类实际上也就相当于用户自定义的类型，和基本数据类型的不同之处在于，类这个特殊类型中同时包含了对数据进行操作的函数。

访问控制属性控制着对类成员的访问权限，实现了数据隐蔽。对象是类的实例，一个对象的特殊性就在于它具有不同于其他对象的自身属性，即数据成员。对象在定义的时候进行的数据成员设置，称为对象的初始化。在对象使用结束时，还要进行一些清理工

作。C++ 中初始化和清理的工作,分别由两个特殊的成员函数来完成,它们就是构造函数和析构函数。复制构造函数是一种特殊的构造函数,可以用已有对象来初始化新对象。

习 题

4-1 解释 public 和 private 的作用,公有类型成员与私有类型成员有哪些区别?

4-2 protected 关键字有何作用?

4-3 构造函数和析构函数有什么作用?

4-4 数据成员可以为公有的吗?成员函数可以为私有的吗?

4-5 已知 class A 中有数据成员 int a,如果定义了 A 的两个对象 a1,a2,它们各自的数据成员 a 的值可以不同吗?

4-6 什么叫做复制构造函数?复制构造函数何时被调用?

4-7 复制构造函数与赋值运算符(=)有何不同?

4-8 定义一个 Dog 类,包含了 age,weight 等属性,以及对这些属性操作的方法。实现并测试这个类。

4-9 设计并测试一个名为 Rectangle 的矩形类,其属性为矩形的左下角与右上角两个点的坐标,根据坐标能计算矩形的面积。

4-10 设计一个用于人事管理的"人员"类。由于考虑到通用性,这里只抽象出所有类型人员都具有的属性:编号、性别、出生日期、身份证号等。其中"出生日期"声明为一个"日期"类内嵌子对象。用成员函数实现对人员信息的录入和显示。要求包括:构造函数和析构函数、复制构造函数、内联成员函数、带默认形参值的成员函数、类的组合。

4-11 定义并实现一个矩形类,有长、宽两个属性,由成员函数计算矩形的面积。

4-12 定义一个 DataType(数据类型)类,能处理包含字符型、整型、浮点型 3 种类型的数据,给出其构造函数。

4-13 定义一个 Circle 类,有数据成员 radius(半径),成员函数 getArea(),计算圆的面积,构造一个 Circle 的对象进行测试。

4-14 定义一个 Tree(树)类,有成员 ages(树龄),成员函数 grow(int years)对 ages 加上 years,age()显示 tree 对象的 ages 的值。

4-15 根据例 4-3 中关于 Circle 类定义的源代码绘出该类的 UML 图形表示。

4-16 根据下面 C++ 代码绘出相应的 UML 图形,表示出类 ZRF、类 SSH 和类 Person 之间的继承关系。

```
class Person {
public:
    Person(const Person& right);
    ~Person();
private:
```

```
    char Name;
    int Age;
};
class ZRF : protected Person {};
class SSH : private Person {};
```

4-17 在一个大学的选课系统中，包括两个类：CourseSchedule 类和 Course 类。其关系为：CourseSchedule 类中的成员函数 add 和 remove 的参数是 Course 类的对象，请通过 UML 方法显式表示出这种依赖关系。

4-18 在一个学校院系人员信息系统中，需要对院系(Department)和教师(Teacher)之间的关系进行部分建模，其关系描述为：每个 Teacher 可以属于零个或多个 Department 的成员，而每个 Department 至少包含一个 Teacher 作为成员。根据以上关系绘制出相应的 UML 类图。

4-19 编写一个名为 CPU 的类，描述一个 CPU 的以下信息：时钟频率，最大不会超过 3000MHz；字长，可以是 32 位或 64 位；核数，可以是单核、双核或四核；是否支持超线程。各项信息要求使用位域来表示。通过输出 sizeof(CPU)来观察该类所占的字节数。

4-20 定义一个负数类 Complex，使得下面的代码能够工作。

```
Complex c1(3, 5);          //用复数 3+5i 初始化 c1
Complex c2=4.5;            //用实数 4.5 初始化 c2
c1.add(c2);                //将 c1 与 c2 相加，结果保存在 c1 中
c1.show();                 //将 c1 输出(这时的结果应该是 7.5+5i)
```

程，执行结束后进程就会消失。

程序是存储在磁盘上的，在执行前，操作系统需要首先将它载入到内存中，并为它分配足够大的内存空间来容纳代码段和数据段，然后把文件中存放的代码段和初始化的数据段的内容载入其中——一部分静态生存期对象的初始化就是通过这种方式完成的，这与动态生存期对象的初始化不同。例如 b.cpp 中的：

```
int x=3;
```

x 的初始化在操作系统载入初始化的数据段时就已经完成了。而 a.cpp 中的下列代码：

```
int z=1;
```

z 在局部作用域中，z 的初始化，需要等到执行到这条语句时，由编译器生成的代码来完成。

细节　那些需要用构造函数来初始化的静态生存期对象又有所不同，它们的初始化，需要由编译器生成专门的代码来调用构造函数，这些代码被调用的时机也由编译器控制。命名空间作用域中的此类对象的初始化代码，一般在执行 main 函数之前，由引导代码调用；局部作用域中的此类对象，其初始化代码一般会内嵌在函数体中，并用一些静态的标志变量来标识这样的对象是否已初始化，从而保证它们的初始化代码只被执行一次。

此外，操作系统还要做一些进程的初始化工作，这些工作完成后，就会跳转到程序的引导代码，开始执行程序。当程序执行结束后，引导代码会通知操作系统，操作系统会完成一些善后工作，程序的一个执行周期就这样结束了。

5.9　小结

在 C++ 中，数据的共享与保护机制是一个很重要的特性。其包含的内容主要为标识符的作用域、可见性和生存期，通过类的静态成员实现同一个类的不同对象之间数据和操作的共享，通过常成员来设置成员的保护属性。

程序的多文件结构有助于编写多个源代码文件来组织大型程序。另外通过编译预处理命令可以为源程序做必要的预处理工作，从而可以避免很多不必要的麻烦和错误。

习　题

5-1　什么叫做作用域？有哪几种类型的作用域？

5-2　什么叫做可见性？可见性的一般规则是什么？

5-3　下面程序的运行结果是什么？实际运行一下，看看与你的设想有何不同。

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

```
int x=5, y=7;
void myFunction() {
    int y=10;
    cout<<"x from myFunction: "<<x<<"\n";
    cout<<"y from myFunction: "<<y<<"\n\n";
}
int main() {
    cout<<"x from main: "<<x<<"\n";
    cout<<"y from main: "<<y<<"\n\n";
    myFunction();
    cout<<"Back from myFunction!\n\n";
    cout<<"x from main: "<<x<<"\n";
    cout<<"y from main: "<<y<<"\n";
    return 0;
}
```

5-4 假设有两个无关系的类 Engine 和 Fuel,使用时,如何使 Fuel 成员访问 Engine 中的私有和保护的成员?

5-5 什么叫做静态数据成员?它有何特点?

5-6 什么叫做静态函数成员?它有何特点?

5-7 定义一个 Cat 类,拥有静态数据成员 numOfCats,记录 Cat 的个体数目;静态成员函数 getNumOfCats(),读取 numOfCats。设计程序测试这个类,体会静态数据成员和静态成员函数的用法。

5-8 什么叫做友元函数?什么叫做友元类?

5-9 如果类 A 是类 B 的友元,类 B 是类 C 的友元,类 D 是类 A 的派生类,那么类 B 是类 A 的友元吗?类 C 是类 A 的友元吗?类 D 是类 B 的友元吗?

5-10 静态成员变量可以为私有的吗?声明一个私有的静态整型成员变量。

5-11 在一个文件中定义一个全局变量 n,主函数 main(),在另一个文件中定义函数 fn1(),在 main()中对 n 赋值,再调用 fn1(),在 fn1()中也对 n 赋值,显示 n 最后的值。

5-12 在函数 fn1()中定义一个静态变量 n,fn1()中对 n 的值加 1,在主函数中,调用 fn1()十次,显示 n 的值。

5-13 定义类 X,Y,Z,函数 h(X*),满足:类 X 有私有成员 i,Y 的成员函数 g(X*)是 X 的友元函数,实现对 X 的成员 i 加 1;类 Z 是类 X 的友元类,其成员函数 f(X*)实现对 X 的成员 i 加 5;函数 h(X*)是 X 的友元函数,实现对 X 的成员 i 加 10。在一个文件中定义和实现类,在另一个文件中实现 main()函数。

5-14 定义 Boat 与 Car 两个类,二者都有 weight 属性,定义二者的一个友元函数 getTotalWeight(),计算二者的重量和。

5-15 在函数内部定义的普通局部变量和静态局部变量在功能上有何不同?计算机底层对这两类变量做了怎样的不同处理,导致了这种差异?

5-16　编译和连接这两个步骤的输入输出分别是什么类型的文件？两个步骤的任务有什么不同？在以下几种情况下，在对程序进行编译、连接时是否会报错？会在哪个步骤报错？

(1) 定义了一个函数 void f(int x, int y)，以 f(1)的形式调用。

(2) 在源文件起始处声明了一个函数 void f(int x)，但未给出其定义，以 f(1)的形式调用。

(3) 在源文件起始处声明了一个函数 void f(int x)，但未给出其定义，也未对其进行调用。

(4) 在源文件 a.cpp 中定义了一个函数 void f(int x)，在源文件 b.cpp 中也定义了一个函数 void f(int x)，试图将两源文件编译后连接在一起。

数组元素,而既然 element 并不是常成员函数,它当然可以把这种权利给主调函数。

这样一改,就可以使得以后这一 element 重载函数不用再加以维护,一切对常成员函数 element 的修改都能反映到其中来,同时又丝毫没有破坏 const 的作用。这就是 const_cast 的安全用法。

请读者思考一下,为什么不能够反过来用,也就是说,为什么不能够把函数的内容写在普通成员函数中,而由常成员函数去调用普通成员函数?

6.9 小结

本章主要介绍了 C++ 中利用数组和指针来组织数据的方法。数组是最为常见的数据组织形式,是具有一定顺序关系的若干变量的集合体。组成数组的变量称为该数组的元素,同一数组的各元素具有相同的数据类型。这里的数据类型可以是一般的简单数据类型,也可以是用户自定义的数据类型,如结构体、类等。如果数组的数据类型是类,则数组的每一个元素都是该类的一个对象,我们称这样的数组为对象数组。对象数组的初始化就是每一个元素对象调用构造函数的过程。可以给每一个数组元素指定显式的初始值,这时会调用相应有形参的构造函数,如果没有指定初始值,就会调用默认构造函数。同样,当一个数组中的元素对象被删除时,系统会调用析构函数来完成。

指针也是一种数据类型,具有指针类型的变量称为指针变量。指针变量是用来存放地址的变量。因此,指针提供了一种直接操作地址的手段。

指针可以指向简单变量,也可以指向对象,同时还可以指向函数(包括普通函数和对象的函数成员)。使用指针一般要包括 3 个步骤——声明、赋初值和引用。指针的初值很重要,使用指针之前必须对指针赋值,让它指向一个已经存在的数据或函数的地址,然后才可以使用;否则可能会造成系统瘫痪等较为严重的问题。指针是 C++ 中的一个难点,正确地使用指针,可以方便、灵活而有效地组织和表示复杂的数据结构。

本章还介绍了内存动态分配可以动态地进行内存管理。

最后,以字符串为例,讲解了数组的特例——字符数组的使用方法,同时介绍了对字符数组的各种操作进行了良好封装的 string 类。

习 题

6-1 数组 a[10][5][15]一共有多少个元素?

6-2 在数组 a[20]中第一个元素和最后一个元素是哪一个?

6-3 用一条语句声明一个有 5 个元素的 int 型数组,并依次赋予 1～5 的初值。

6-4 已知有一个数组名叫 oneArray,用一条语句求出其元素的个数。

6-5 用一条语句声明一个有 5×3 个元素的二维 int 型数组,并依次赋予 1～15 的初值。

6-6 运算符"*"和"&"的作用是什么?

6-7 什么叫做指针?指针中储存的地址和这个地址中的值有何区别?

6-8 声明一个 int 型指针,用 new 语句为其分配包含 10 个元素的地址空间。

6-9　在字符串"Hello,world!"中的结束符是什么?

6-10　声明一个有 5 个元素的 int 型数组,在程序中提示用户输入元素值,最后再在屏幕上显示出来。

6-11　引用和指针有何区别? 何时只能使用指针而不能使用引用?

6-12　声明下列指针: float 类型的指针 pfloat,char 类型的指针 pstr,struct Customer 型的指针 pcus。

6-13　给定 float 类型的指针 fp,写出显示 fp 所指向的值的输出流语句。

6-14　在程序中声明一个 double 类型变量的指针,分别显示指针占了多少字节和指针所指的变量占了多少字节。

6-15　const int * p1 和 int * const p2 的区别是什么?

6-16　声明一个 int 型变量 a,一个 int 型指针 p,一个引用 r,通过 p 把 a 的值改为 10,通过 r 把 a 的值改为 5。

6-17　下列程序有何问题,请仔细体会使用指针时应如何避免出现这个问题。

```
#include <iostream>
using namespace std;
int main(){
    int * p;
    * p=9;
    cout<< "The value at p: "<< * p;
    return 0;
}
```

6-18　下列程序有何问题,请改正;仔细体会使用指针时应如何避免出现这样的问题。

```
#include <iostream>
using namespace std;
int fn1(){
   int * p=new int (5);
   return * p;
}
int main(){
   int a=fn1();
   cout<< "the value of a is: "<<a;
   return 0;
}
```

6-19　声明一个参数为 int 型、返回值为 long 型的函数指针;声明类 A 的一个成员函数指针,其参数为 int 型,返回值为 long 型。

6-20　实现一个名为 SimpleCircle 的简单圆类。其数据成员 int * itsRadius 为一个指向其半径值的指针,存放其半径值。设计对数据成员的各种操作,给出这个类的完整实现并测试这个类。

6-21　编写一个函数,统计一条英文句子中字母的个数,在主程序中实现输入输出。

6-22 编写函数 void reverse(string &s),用递归算法使字符串 s 倒序。

6-23 设学生人数 $N=8$,提示用户输入 N 个人的考试成绩,然后计算出他们的平均成绩并显示出来。

6-24 基于 char * 设计一个字符串类 MyString,并且具有构造函数、析构函数、复制构造函数,重载运算符"+","=","+=","[]",尽可能完善它,使之能满足各种需要(运算符重载功能为选做,参见第 8 章)。

6-25 编写一个 3×3 矩阵转置的函数,在 main()函数中输入数据。

6-26 编写一个矩阵转置的函数,矩阵的行数和列数在程序中由用户输入。

6-27 定义一个 Employee 类,其中包括表示姓名、地址、城市和邮编等属性,包括 setName()和 display()等函数。display()使用 cout 语句显示姓名、地址、城市和邮编等属性,函数 setName()改变对象的姓名属性,实现并测试这个类。

6-28 分别将例 6-10 程序和例 6-16 程序中对指针的所有使用都改写为与之等价的引用形式,比较修改前后的程序,体会在哪些情况下使用指针更好,哪些情况下使用引用更好。

6-29 运行下面的程序,观察执行结果,指出该程序是如何通过指针造成安全性隐患的,思考如何避免这种情况的发生。

```
#include <iostream>
using namespace std;
int main(){
    int arr[]={1, 2, 3};
    double * p=reinterpret_cast<double * > (&arr[0]);
    * p=5;
    cout<<arr[0]<<" "<<arr[1]<<" "<<arr[2]<<endl;
    return 0;
}
```

6-30 static_cast,const_cast 和 reinterpret_cast 各自应在哪些情况下使用?

足继承关系上的兼容性，是不可靠的。当然，最好不用 void 指针。

通过上面几点可以看出，用 static_cast 执行涉及类类型指针或引用的转换时，有很多不安全因素，而且还存在一些限制。第 8 章将介绍基类向派生类转换的更安全、更灵活的方式——dynamic_cast，不过使用它要付出一定的效率代价。

7.9　小结

本章首先介绍了类的继承。类的继承允许程序员在保持原有类特性的基础上，进行更具体、更详细的类的定义。新的类由原有的类所产生，可以说新类继承了原有类的特征，或者原有类派生出新类。派生新类的过程一般包括吸收已有类的成员、改造已有类成员和添加新的成员 3 个步骤。围绕派生过程，着重讨论不同继承方式下的基类成员的访问控制问题、添加构造函数和析构函数。接着讨论了在较为复杂的继承关系中，派生类成员的唯一标识和访问问题。最后讨论派生类对象的使用场合和范围问题。用全选主元高斯消去法求解线性方程组和人员信息管理的例子，是对本章内容的回顾与总结。

继承就是从先辈处得到特性和特征。类的继承，是新的类从已有类那里得到已有的特性，而从已有类产生新类的过程就是类的派生。派生类同样也可以作为基类派生新的类，这样就形成了类的层次结构。类的派生实际是一种演化、发展过程，即通过扩展、更改和特殊化，从一个已知类出发建立一个新类。类的派生通过建立具有共同关键特征的对象家族，从而实现代码的重用。这种继承和派生的机制，对于已有程序的发展和改进，是极为有利的。

在类的层次结构中，最高层是抽象程度最高的，是最具有普遍和一般意义的概念，下层具有了上层的特性，同时加入了自己的新特征，而最下层是最为具体的。在这个层次结构中，由上到下，是一个具体化、特殊化的过程，由下到上，是一个抽象化的过程。

派生新类的过程包括 3 个步骤：吸收基类成员、改造基类成员和添加新的成员。C++ 类继承中，派生类包含了它所有基类的除构造、析构函数之外的所有成员。对基类成员的改造包括两个方面，第一个是基类成员的访问控制问题，依靠派生类定义时的继承方式来控制。第二个是对基类数据或函数成员的隐藏，对基类的功能进行改造。派生类新成员的加入是继承与派生机制的核心，是保证派生类在功能上有所发展的关键。可以根据实际情况的需要给派生类添加适当的数据和函数成员，来实现必要的新增功能。同时，在派生过程中，基类的构造函数和析构函数是不能被继承下来的，一些特别的初始化和扫尾清理工作，需要加入新的构造和析构函数。

派生过程完成之后，我们讨论了派生类及其对象的成员（包括了基类中继承来的部分和新增的部分）标识和访问问题。访问过程中，实际上有两个问题需要解决，第一是唯一标识问题，第二个问题是成员本身的属性问题，也就是可见性问题。为了解决成员的唯一标识问题，我们介绍了同名隐藏原则、作用域分辨符和虚基类等方法。

类型兼容规则，实际就是派生类对象的使用场合问题。基类经过公有派生产生的派生类，具有了基类的全部功能，凡是能够使用基类的地方，都可以由公有派生类来代替。

这一特性,为第 8 章要学习的类的多态性(动态联编)奠定了基础。

习 题

7-1 比较类的 3 种继承方式 public(公有继承)、protected(保护继承)、private(私有继承)之间的差别。

7-2 派生类构造函数执行的次序是怎样的?

7-3 如果派生类 B 已经重载了基类 A 的一个成员函数 fn1(),没有重载基类的成员函数 fn2(),如何在派生类的函数中调用基类的成员函数 fn1(),fn2()?

7-4 什么叫做虚基类?它有何作用?

7-5 定义一个基类 Shape,在此基础上派生出 Rectangle 和 Circle,二者都有 getArea()函数计算对象的面积。使用 Rectangle 类创建一个派生类 Square。

7-6 定义一个哺乳动物类 Mammal,再由此派生出狗类 Dog,定义一个 Dog 类的对象,观察基类与派生类的构造函数和析构函数的调用顺序。

7-7 定义一个基类及其派生类,在构造函数中输出提示信息,构造派生类的对象,观察构造函数的执行情况。

7-8 定义一个 Document 类,有数据成员 name,从 Document 派生出 Book 类,增加数据成员 pageCount。

7-9 定义一个基类 Base,有两个公有成员函数 fn1(),fn2(),私有派生出 Derived 类,如何通过 Derived 类的对象调用基类的函数 fn1()?

7-10 定义一个 Object 类,有数据成员 weight 及相应的操作函数,由此派生出 Box 类,增加数据成员 height 和 width 及相应的操作函数,声明一个 Box 对象,观察构造函数与析构函数的调用顺序。

7-11 定义一个基类 BaseClass,从它派生出类 DerivedClass。BaseClass 有成员函数 fn1(),fn2(),DerivedClass 也有成员函数 fn1(),fn2()。在主函数中声明一个 DerivedClass 的对象,分别用 DerivedClass 的对象以及 BaseClass 和 DerivedClass 的指针来调用 fn1(),fn2(),观察运行结果。

7-12 组合与继承有什么共同点和差异?通过组合生成的类与被组合的类之间的逻辑关系是什么?继承呢?

7-13 思考例 7-6 和例 7-8 中 Derived 类的各个数据成员在 Derived 对象中存放的位置,编写程序输出它们各自的地址来验证自己的推断。

7-14 基类与派生类的对象、指针或引用之间,哪些情况下可以隐含转换,哪些情况下可以显示转换?在涉及多重继承或虚继承的情况下,在转换时会面临哪些新问题?

7-15 下面的程序能得到预期的结果吗?如何避免类似问题的发生?

```
#include <iostream>
using namespace std;
struct Base1 {int x;};
struct Base2 {float y;};
```

```
struct Derived : Base1, Base2 {};
int main(){
    Derived * pd=new Derived;
    pd->x=1; pd->y=2.0f;
    void * pv=pd;
    Base2 * pb=static_cast<Base2 * >(pv);
    cout<<pd->y<<" "<<pb->y<<endl;
    delete pb;
    return 0;
}
```

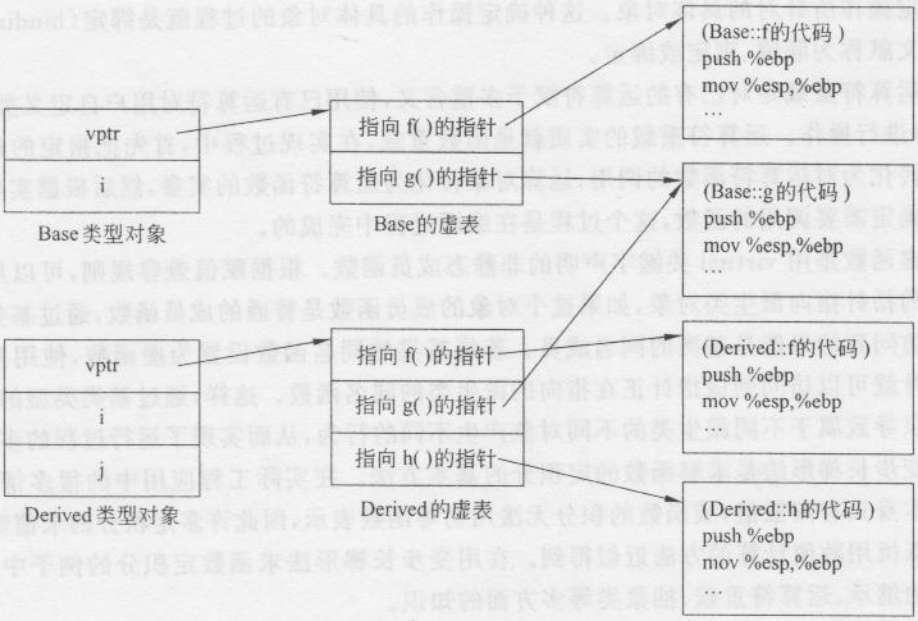

图 8-9　动态绑定通常的实现方式

的虚表指针指向其中一段的首地址)。

事实上,一个类的虚表中存放的不只是虚函数的指针,用于支持运行时类型识别的对象的运行时类型信息也需要通过虚表来访问。只有多态类型有虚表,因此只有多态类型支持运行时类型识别。

8.8　小结

本章介绍了类的多态特性。多态是指同样的消息被不同类型的对象接收时导致完全不同的行为,是对类的特定成员函数的再抽象。这里的消息是指对类的成员函数的调用,不同的行为是指不同的实现,也就是调用了不同的函数。

C++ 支持的多态又可以分为 4 类:重载多态、强制多态、包含多态和参数多态。前面两种统称为专用多态,而后面两种称为通用多态。我们学习过的普通函数及类的成员函数的重载都属于重载多态。重载指的是同一个函数、过程可以操作于不同类型的对象。强制多态是通过语义操作把一个变元的类型加以变化,以符合一个函数或者操作的要求。包含多态是研究类族中定义于不同类中的同名成员函数的多态行为,主要是通过虚函数来实现。参数多态与类属相关联,类属是一个可以参数化的模板,其中包含的操作所涉及的类型必须用类型参数实例化。这样,由类属实例化的各类都具有相同的操作,而操作对象的类型却各不相同。本章主要介绍重载和包含两种多态类型,其中运算符重载、虚函数是学习的重点。

多态从实现的角度来讲可以划分为两类,即编译时的多态和运行时的多态。前者是在编译的过程中确定了同名操作的具体操作对象,而后者则是在程序运行过程中才动态

地确定操作所针对的具体对象。这种确定操作的具体对象的过程就是绑定(binding),也有的文献称为联编、束定或绑定。

运算符重载是对已有的运算符赋予多重含义,使用已有运算符对用户自定义类型(比如类)进行操作。运算符重载的实质就是函数重载,在实现过程中,首先把指定的运算表达式转化为对运算符函数的调用,运算对象转化为运算符函数的实参,然后根据实参的类型来确定需要调用的函数,这个过程是在编译过程中完成的。

虚函数是用 virtual 关键字声明的非静态成员函数。根据赋值兼容规则,可以用基类类型的指针指向派生类对象,如果这个对象的成员函数是普通的成员函数,通过基类类型指针访问到的只能是基类的同名成员。若将基类的同名函数设置为虚函数,使用基类类型指针就可以访问到该指针正在指向的派生类的同名函数。这样,通过基类类型的指针,就可以导致属于不同派生类的不同对象产生不同的行为,从而实现了运行过程的多态。

变步长梯形法是求解函数的定积分的基本方法。在实际工程应用中的很多情况下,函数本身只有离散值,或函数的积分无法用初等函数表示,因此许多定积分的求值都是通过计算机用数值计算的方法近似得到。在用变步长梯形法求函数定积分的例子中,利用了类的继承、运算符重载、抽象类等多方面的知识。

本章最后,仍以一个个人银行账户管理程序为例,对例 7-10 进行了改进,以此说明了虚函数的作用和使用方法。

习　题

8-1 什么叫做多态性?在 C++ 中是如何实现多态的?

8-2 什么叫做抽象类?抽象类有何作用?抽象类的派生类是否一定要给出纯虚函数的实现?

8-3 在 C++ 中,能否声明虚函数?为什么?能否声明虚析构函数?有何用途?

8-4 请编写一个计数器 Counter 类,对其重载运算符“+”。

8-5 编写一个哺乳动物类 Mammal,再由此派生出狗类 Dog,二者都声明 speak()成员函数,该函数在基类中被声明为虚函数。声明一个 Dog 类的对象,通过此对象调用 speak 函数,观察运行结果。

8-6 请编写一个抽象类 Shape,在此基础上派生出类 Rectangle 和 Circle,二者都有计算对象面积的函数 getArea()、计算对象周长的函数 getPerim()。

8-7 对类 Point 重载“++”(自增)、“--”(自减)运算符,要求同时重载前缀和后缀的形式。

8-8 定义一个基类 BaseClass,从它派生出类 DerivedClass。BaseClass 有成员函数 fn1(),fn2(),fn1()是虚函数;DerivedClass 也有成员函数 fn1(),fn2()。在主函数中声明一个 DerivedClass 的对象,分别用 BaseClass 和 DerivedClass 的指针指向 DerivedClass 的对象,并通过指针调用 fn1(),fn2(),观察运行结果。

8-9 请编写程序定义一个基类 BaseClass,从它派生出类 DerivedClass。在 BaseClass 中声明虚析构函数,在主函数中将一个动态分配的 DerivedClass 的对象地址赋给一个

BaseClass 的指针，然后通过指针释放对象空间。观察程序运行过程。

8-10 编写程序定义类 Point，有数据成员 x，y，为其定义友元函数实现重载“+”。

8-11 在例 8-6 的基础上，通过继承 Rectangle 得到一个新的类 Square，然后在 Shape 中增加一个函数 int getVertexCount() const 用来获得当前图形的顶点个数。用以下几种方法分别实现，并体会各自的优劣。

(1) 使用 dynamic_cast 实现 Shape::getVertexCount 函数。

(2) 使用 typeid 实现 Shape::getVertexCount 函数。

(3) 将 Shape::getVertexCount 声明为纯虚函数，在派生类中给出具体实现。

常整数为次数的乘方问题就这样得到了比较理想的解决。

模板元编程的方法、技巧十分丰富，本节仅通过两个小例子使读者对它有一个初步的认识，以达到开拓思路的目的。

9.6　小结

本章首先介绍了模板的概念，然后在模板的基础上详细介绍了几种常用的线性群体类模板：数组类、链表类、栈和队列，最后介绍了群体数据的组织，并简要介绍了 3 种排序方法和两种查找方法的原理。

模板是 C++ 支持参数化多态性的工具，函数模板实现了类型参数化，将函数处理的数据类型作为参数，提高了代码的可重用性。类模板使用户可以为类定义一种模式，使得类中的某些数据成员、某些成员函数的参数、某些成员函数的返回值能取任意类型（包括系统预定义的和用户自定义的）。

线性群体中的元素次序与其位置关系是对应的。在线性群体中，又可按照访问元素的不同方法分为直接访问、顺序访问和索引访问。在本章介绍了直接访问和顺序访问的线性群体。

插入排序的基本思想是：每一步将一个待排序元素按其关键字值的大小插入到已排序序列的适当位置上，直到待排序元素插入完为止。

选择排序的基本思想是：每次从待排序序列中选择一个关键字最小的元素（当需要按关键字升序排列时），顺序排在已排序序列的最后，直至全部排完。

交换排序的基本思想是：两两比较待排序序列中的元素，并交换不满足顺序要求的各对元素，直到全部满足顺序要求为止。

顺序查找方法的基本思想是：从数组的首元素开始，逐个将元素与待查找的关键字进行比较，直到找到相等的。若整个数组中没有与待查找关键字相等的元素，则查找不成功。

折半查找方法的基本思想是：对于已按关键字排序的序列，经过一次比较，可分割成两部分，然后只在有可能包含待查元素的一部分中继续查找，并根据试探结果继续分割，逐步缩小查找范围，直至找到或找不到为止。

对群体数据的排序和查找方法还有很多，本章介绍的只是一些最简单、常用的方法，读者如需了解更多有关这方面的问题，可参考有关数据结构的书籍。

习　题

9-1　编写程序提示用户输入一个班级中的学生人数 n，再依次提示用户输入 n 个人在课程 A 中的考试成绩，然后计算出平均成绩，显示出来。请使用本书第 9 章中的数组类模板 Array 定义浮点型数组存储考试成绩。

9-2　链表的结点类至少应包含哪些数据成员？单链表和双向链表的区别是什么？

9-3　链表中元素的最大数目为多少？

9-4 在双向链表中使用的结点类与单链表中使用的结点类相比,应有何不同?试声明并实现双向链表中使用的结点类 DNode。

9-5 使用本章中的链表类模板,声明两个 int 类型的链表 a 和 b,分别插入 5 个元素,然后把 b 中的元素加入 a 的尾部。

9-6 通过对从本章的链表类模板 LinkedList 进行组合,编写有序链表类模板 OrderList,添加成员函数 insert 实现链表元素的有序(递增)插入。声明两个 int 类型有序链表 a 和 b,分别插入 5 个元素,然后把 b 中的元素插入 a 中。

9-7 什么叫做栈?对栈中元素的操作有何特性?

9-8 什么叫做队列?对队列中元素的操作有何特性?

9-9 简单说明插入排序的算法思想。

9-10 初始化 int 类型数组 data1[]={1,3,5,7,9,11,13,15,17,19,2,4,6,8,10,12,14,16,18,20},应用本章的直接插入排序函数模板进行排序。对此函数模板稍做修改,加入输出语句,在每插入一个待排序元素后显示整个数组,观察排序过程中数据的变化,加深对插入排序算法的理解。

9-11 简单说明选择排序的算法思想。

9-12 初始化 int 类型数组 data1[]={1,3,5,7,9,11,13,15,17,19,2,4,6,8,10,12,14,16,18,20},应用本章中的直接选择排序函数模板进行排序。对此函数模板稍做修改,加入输出语句,每次从待排序序列中选择一个元素添加到已排序序列后,显示整个数组,观察排序过程中数据的变化,加深对直接选择排序算法的理解。

9-13 简单说明交换排序的算法思想。

9-14 初始化 int 类型数组 data1[]={1,3,5,7,9,11,13,15,17,19,2,4,6,8,10,12,14,16,18,20},应用本章中的起泡排序函数模板进行排序。对此函数模板稍做修改,加入输出语句,每完成一趟起泡排序后显示整个数组,观察排序过程中数据的变化,加深对起泡排序算法的理解。

9-15 本章例题的排序算法都是升序排序,稍做修改后即可完成降序排序。请编写降序的起泡排序函数模板,然后在程序中初始化 int 类型的数组 data1[]={1,3,5,7,9,11,13, 15,17,19,2,4,6,8,10,12,14,16,18,20},应用降序的起泡排序函数模板进行排序,加入输出语句,每完成一趟起泡排序后显示整个数组,观察排序过程中数据的变化。

9-16 简单说明顺序查找的算法思想。

9-17 初始化 int 类型数组 data1[]={1,3,5,7,9,11,13,15,17,19,2,4,6,8,10,12,14,16,18,20},提示用户输入一个数字,应用本章的顺序查找函数模板找出它的位置。

9-18 简单说明折半查找的算法思想。

9-19 初始化 int 类型数组 data1[]={1,3,5,7,9,11,13,15,17,19,2,4,6,8,10,12,14,16,18,20},先使用任一种算法对其进行排序,然后提示用户输入一个数字,应用本章的折半查找函数模板找出它的位置。

9-20 模板的实例化在什么情况下会发生?请指出例 9-1 和例 9-2 的程序中有哪些模板实例。

9-21　C++ 提供了哪些为模板定义特殊实现的语言机制？模板的特化和偏特化分别是指什么？

9-22　例 9-13 的 bubbleSort 函数模板可以为数组排序，但如果将一个指针数组传递给它，在排序时将比较指针所存储地址的大小。

(1) 请利用函数模板重载技术，使得对指针数组进行排序时，以指针所指向内容为比较依据。

(2) 在解决问题(1)时，如果采取了对 bubbleSort 函数模板重载的办法，将使得起泡排序的主要代码被书写两遍，造成代码的冗余，请思考如何解决这一问题。

9-23　尝试通过模板元编程解决实现以下功能。

(1) 设计一个类模板 template<unsigned M, unsigned N>Permutation，内含一个枚举值 VALUE，Permutation<M, N>::VALUE 的值为排列数 P_M^N。

(2) 设计一个类模板 template<unsigned M, unsigned N>Gcd，内含一个枚举值 VALUE，Gcd<M, N>::VALUE 的值为 M 和 N 的最大公约数。提示：求最大公约数可以用以下的辗转相除法：

$$\gcd(m,n)=\begin{cases} m & (m \text{ 能整除 } n) \\ \gcd(n\%m,m) & (m \text{ 不能整除 } n) \end{cases}$$

或函数类,Lambda 程序库允许用户以一种更简便的方式创建函数对象,提供了更大的方便。例如:

```
transform(s.begin(), s.end(), cout<<_1*_1<<" ");
                                        //将 s 容器中每个元素的平方输出
```

- String Algo:该程序库提供了一些实用的字符串算法。例如字符串的大小写转换(to_upper 和 to_lower),去掉一个字符串首部和尾部的空白字符(trim),判断一个字符串是否以指定字串开头或结尾(starts_with 和 ends_with),判断一个字符串是否包含指定字串(contains),将一个字符串容器中的所有元素连接成为一个字符串(join),将一个字符串按指定分隔符拆分为若干个子串并放到一个字符串容器中(split)等。
- uBLAS:uBLAS 程序库提供了对矩阵和向量进行计算的功能。
- Unit:Unit 程序库允许对各种物理单位进行表示和计算。

以上列出的程序库仅是 Boost 的一小部分,读者如有兴趣想进一步了解 Boost,可以登录 Boost 的官方网站 http://www.boost.org/。

10.8 小结

泛型程序设计是一种重要的程序设计方法,合理利用泛型程序设计,可以大大提高软件的复用性。C++ 的模板是进行泛型程序设计的主要工具。标准模板库 STL 是 C++ 标准库的一部分,它定义了一套概念体系,容器、迭代器、函数对象和算法都是这个体系中最基本的概念,STL 的主要组件都是其中某种概念的模型。容器、迭代器和函数对象都有各自的子概念,每个 STL 组件所属的概念决定了它的功能以及它能和其他哪些组件协同工作。

容器是数据的载体,算法用于对数据序列执行操作,函数对象用来描述算法对单个数据执行的具体运算,迭代器充当算法和容器的桥梁。算法通过迭代器而非直接通过容器来操作数据,算法对每个元素所执行的具体运算用独立于算法的函数对象来描述,正是这两点造成了 STL 具有良好的通用性、灵活性和可扩展性。此外,STL 还提供了各种适配器,STL 的适配器包括迭代器适配器、容器适配器和函数适配器。适配器的存在进一步增强了 STL 的灵活性。

习 题

10-1 STL 的容器、迭代器和算法具有哪些子概念? vector, deque, list, set, multiset, map, multimap 各容器的迭代器分别属于哪种迭代器?

10-2 若 s 是一个大小为 5 的静态数组 int s[5],[s+1, s+4)这个区间包括数组的哪几个元素? [s+4, s+5),[s+4, s+4)和[s+4, s+3)是合法的区间吗?

10-3 建立一个向量容器的实例 s,不断对 s 调用 push_back 向其中增加新的元素,观察

在此过程中 s.capacity()的变化。

10-4 如果需要使用一个顺序容器来存储数据，在以下几种情况下，分别应当选择哪种顺序容器？

(1) 新元素全部从尾部插入，需要对容器进行随机访问。

(2) 新元素可能从头部或尾部插入，需要对容器进行随机访问。

(3) 新元素可能从任意位置插入，不需要对容器进行随机访问。

10-5 约瑟夫问题：n 个骑士编号 1,2,…,n，围坐在圆桌旁。编号为 1 的骑士从 1 开始报数，报到 m 的骑士出列，然后下一个位置再从 1 开始报数，找出最后留在圆桌旁的骑士编号。

(1) 编写一个函数模板。以一种顺序容器的类型作为模板参数，在模板中使用指定类型的顺序容器求解约瑟夫问题，m,n 是该函数模板的形参。

(2) 分别以 vector<int>，deque<int>，list<int>作为类型参数调用该函数模板，调用时将 n 设为较大的数，将 m 设为较小的数(例如令 $m=100\ 000$，$n=5$)。观察 3 种情况下调用该函数模板所需花费的时间。

10-6 编写一个具有以下原型的函数模板。

```
template<class T>
void exchange(list<T>& l1, list<T>::iterator p1, list<T>& l2, list<T>::
iterator p2);
```

该模板用于将 l1 链表的[p1, l1.end())区间和 l2 链表的[p2, l2.end())区间的内容交换。在主函数中调用该模板，以测试该模板的正确性。

10-7 分别对 stack<int>，queue<int>，priority_queue<int>的实例执行下面的操作：调用 push 函数分别将 5,1,4,6 压入；调用两次 pop 函数；调用 push 函数分别将 2,3 压入；调用两次 pop 函数。请问对于三类容器适配器，每次调用 pop 函数时弹出的元素分别是什么？请编写程序验证自己的推断。

10-8 编写一个程序，从键盘输入一个个单词，每接收到一个单词后，输出该单词是否曾经出现过以及出现次数。可以尝试分别用多重集合(multiset)和映射(map)两种途径实现，将二者进行比较。

10-9 编写程序对比 STL 中的 3 个元素交换函数 swap，iter_swap 和 swap_ranges 对数组中的元素进行的交换操作。

10-10 编写一个程序，从键盘输入两组整数(可以看作两个集合)，分别输出同属于两组的整数(即两个集合的交集)、属于至少一组的整数(即两个集合的并集)、属于第一组但不属于第二组的整数(即两个集合的差集)。程序中需要用到 sort，set_intersection，set_union，set_difference 等算法。

10-11 下面的程序段首先构造了一个元素按升序排列的向量容器 s，然后试图调用 unique 算法去掉其中的重复元素，并将结果输出。

```
int arr[]={ 1, 1, 4, 4, 5 };
vector<int>s(arr, arr+5);
unique(s.begin(), s.end());
```

```
copy(s.begin(), s.end(), ostream_iterator<int>(cout, "\n"));
```

(1) 以上的输出结果是什么？是否真正达到了去除重复元素的目的？如未达到目的，应如何对程序进行修改？

(2) 如果 s 是列表，是否有更方便高效的方法？

10-12 编写一个产生器，用来产生 0～9 范围内的随机数。建立一个顺序容器，使用该产生器和 generate 算法为该容器的元素赋值。

10-13 编写一个二元函数对象，用来计算 x 的 y 次方，其中 x 和 y 都是整数。利用该函数对象和 transform 算法，并结合适当的函数适配器，对于 10-12 题所生成的整数序列中的每个元素 n，分别输出 5^n，n^7 和 n^n。

10-14 为例 9-3 的 Array 类增加一个成员函数 swap。

10-15 对例 10-25 中的 IncrementIterator 进行扩充，使它成为一个随机访问迭代器。

10-16 对例 10-26 中的 mySort 算法进一步改进，使得当传入的第三个参数为随机访问迭代器时，直接在输出的区间中进行排序，避免使用 s 作为中转，从而节省时间和空间。

10-17 登录 Boost 的官方网站，下载最新的 Boost 程序库，阅读文档，完成以下任务。

(1) 修改例 10-3，将主函数中的静态数组 a 变为 array<double, 5>类型的对象，对与 a 相关的代码进行相应修改，比较修改前后的程序。

(2) 修改例 10-17，使用 Bind 程序库的 bind 代替 bind2nd。

(3) 用另一种方式修改例 10-17，使用 Lambda 程序库的相关功能来生成判断一个数是否大于 40 的函数对象。

(4) 改用 unordered_map 实现例 10-10 的功能。

11.7 小结

本章首先介绍流的概念,然后介绍了流类库的结构和使用。C++ 语言中没有输入输出语句。但 C++ 编译系统带有一个面向对象的 I/O 软件包,它就是 I/O 流类库。流是 I/O 流类的中心概念。流是一种抽象,它负责在数据的生产者和数据的消费者之间建立联系,并管理数据的流动。程序将流对象看作是文件对象的化身。

一个输出流对象是信息流动的目标,最重要的 3 个输出流是 ostream,ofstream 和 ostringstream。

一个输入流对象是数据流出的源头,3 个最重要的输入流类是 istream,ifstream 和 istringstream。

一个 iostream 对象可以是数据的源或目的。两个重要的 I/O 流类都是从 iostream 派生的,它们是 fstream 和 stringstream。

本章只介绍了流类库的概念和简单应用,关于流类库的详细说明,还需要参阅类库参考手册或联机帮助。

习 题

11-1 什么叫做流? 流的提取和插入是指什么? I/O 流在 C++ 中起着怎样的作用?

11-2 cout,cerr 和 clog 有何区别?

11-3 使用 I/O 流以文本方式建立一个文件 test1.txt,写入字符"已成功写入文件!",用其他字处理程序(例如 Windows 的记事本程序 Notepad)打开,看看是否正确写入。

11-4 使用 I/O 流以文本方式打开 11-3 题建立的文件 test1.txt,读出其内容并显示出来,看看是否正确。

11-5 使用 I/O 流以文本方式打开 11-3 题建立的文件 test1.txt,在文件后面添加字符"已成功添加字符!",然后读出整个文件的内容显示出来,看看是否正确。

11-6 定义一个 Dog 类,包含体重和年龄两个成员变量及相应的成员函数。声明一个实例 dog1,体重为 5,年龄为 10,使用 I/O 流把 dog1 的状态写入磁盘文件。再声明另一个实例 dog2,通过读文件把 dog1 的状态赋给 dog2。分别使用文本方式和二进制方式操作文件,看看结果有何不同;再看看磁盘文件的 ASCII 码有何不同。

11-7 观察下面的程序,说明每条语句的作用,写出程序执行的结果。

```
#include<iostream>
using namespace std;
int main() {
    ios_base::fmtflags original_flags=cout.flags();                    //1
    cout<<812<<'|';
    cout.setf(ios_base::left,ios_base::adjustfield);                   //2
```

```
    cout.width(10);                                                    //3
    cout<<813<<815<<'\n';
    cout.unsetf(ios_base::adjustfield);                                //4
    cout.precision(2);
    cout.setf(ios_base::uppercase|ios_base::scientific);               //5
    cout<<831.0;
    cout.flags(original_flags);                                        //6
    return 0;
}
```

11-8　编写程序提示用户输入一个十进制整数，分别用十进制、八进制和十六进制形式输出。

11-9　编写程序实现如下功能：打开指定的一个文本文件，在每一行前加行号后将其输出到另一个文本文件中。

11-10　使用宽输入流从一个有中文字符的文本文件中读入所有字符，统计每个字符出现的次数，将统计结果用宽输出流输出到另一个文本文件中。

11-11　修改本章的综合实例，不再在文件中保存用户输入的命令，而是在每次程序结束前使用 boost 的 Serialization 程序库将当前状态保存到文件中，程序启动后再从文件中将状态恢复。

```
auto_ptr<SomeClass>ap2(ap1);
```

等价于

```
auto_ptr<SomeClass>ap2(ap1.release());
```

而执行赋值

```
ap2=ap1;
```

等价于

```
ap2.reset(ap1.release());
```

这是因为同时至多只能有一个智能指针对象负责同一个指针的删除。

提示 除 auto_ptr 外,还有多种被广泛应用的智能指针对象,如 shared_ptr,它们即将进入下一版的 C++ 标准。Boost 的 Smart Ptr 程序库提供了它们的实现,10.7.3 节曾对该程序库有所介绍。

将 auto_ptr 对象定义在栈上,可以很方便地删除堆对象,在发生异常时无须做特别处理。例如:

```
void someFunction() {
    auto_ptr<SomeClass>ap(new someClass(…));
                                        //创建一个堆对象,将它关联到 ap
    …
    if (…) {
        throw someException();          //可以直接抛出异常,无须手工释放资源
    }
    …
    someOtherFunction();        //如果该函数抛出异常, ap 维护的堆对象也会被自动删除
    …
}
```

12.7 小结

程序运行中的有些错误是可以预料但不可避免的,当出现错误时,要力争做到允许用户排除环境错误,继续运行程序,这就是异常处理程序的任务。C++ 语言提供对处理异常情况的内部支持。try,throw 和 catch 语句就是 C++ 语言中用于实现异常处理的机制。

为了加强程序的可读性,使用户能够方便地知道所使用的函数会抛掷哪些异常,可以在函数的声明中列出这个函数可能抛掷的所有异常类型,这就是异常接口声明。

C++ 异常处理的真正功能,不仅在于它能够处理各种不同类型的异常,还在于它具有在栈解旋期间为异常抛掷前构造的所有局部对象自动调用析构函数的能力。

最后,对 C++ 标准程序库中标准异常类及功能进行了介绍,同时介绍了 C++ 标准程序库对异常处理做的保证,即 C++ 标准程序库在面对异常时,应当保证不会发生资源泄漏,也不能破坏容器的不变特性。

习　题

12-1　什么叫做异常？什么叫做异常处理？

12-2　C++ 的异常处理机制有何优点？

12-3　举例说明 throw，try，catch 语句的用法。

12-4　设计一个异常抽象类 Exception，在此基础上派生一个 OutOfMemory 类响应内存不足，一个 RangeError 类响应输入的数不在指定范围内，实现并测试这几个类。

12-5　练习使用 try，catch 语句，在程序中用 new 分配内存时，如果操作未成功，则用 try 语句触发一个 char 类型异常，用 catch 语句捕获此异常。

12-6　修改例 9-3 的 Array 类模板，在执行“[]”运算符时，若输入的索引 i 在有效范围之外，抛出 out_of_range 异常。

12-7　例 9-10 的 Queue 模板中有哪些函数不是异常安全的？请尝试对这些函数进行修改，使之成为异常安全的。

12-8　智能指针 auto_ptr 有什么用处？设计一个类 SomeClass，在它的默认构造函数、复制构造函数、赋值运算符和析构函数中输出提示信息，在主程序中创建多个 auto_ptr<SomeClass>类型的实例，彼此之间进行复制构造和赋值。观察输出的提示信息，体会 auto_ptr 的工作方式。