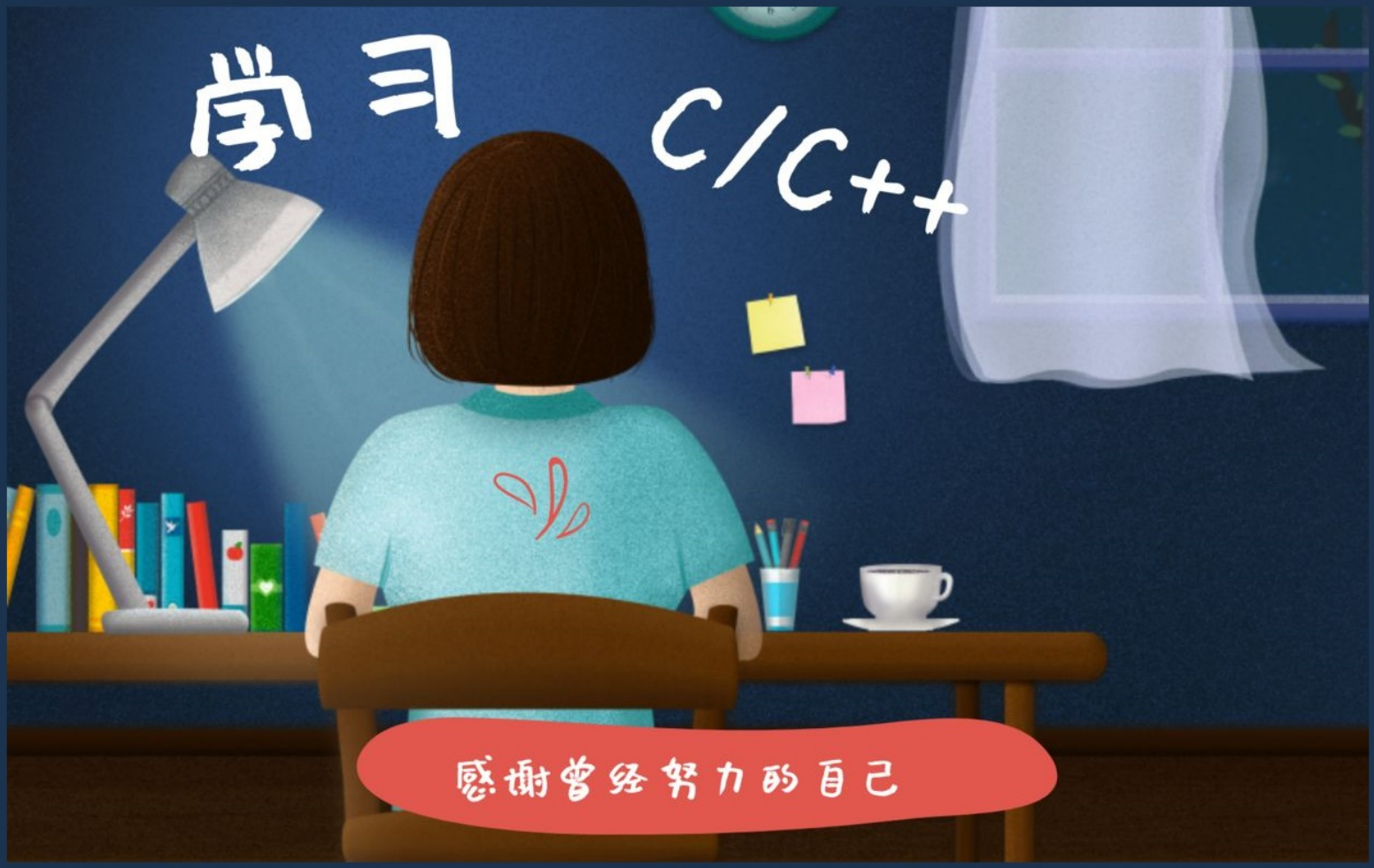

程序设计（二）实验指导书

C++ 语言习题集

作者：陈哲

组织：南京航空航天大学 计算机科学与技术学院

时间：2022 年 2 月 20 日

The best way to learn a new programming language is by writing programs in it.

—— Dennis Ritchie（C 语言发明者）

目录

第 1 章 C++ 程序设计基础

内容提要

- cout/cin 和格式化输出/输入
- 布尔类型（bool）
- 引用类型（&）
- 函数的默认参数
- 函数重载
- 内存的动态分配和释放（new/delete）

所有声称“兼容标准”的 C/C++ 编译器都遵循同样的 C/C++ 语言标准，因此任意 C/C++ 编译器都能够编译书中的 C/C++ 程序。常见的 C/C++ 编译器和集成开发环境（IDE）包括：

- Dev-C++ 5.11：Windows 下基于 GCC 编译器的开源 C/C++ 集成开发环境。（请到官网下载，原作者于 Dev-C++ 4.9.9.2 后停止更新，推荐使用 Orwell Dev-C++ 5.11或 Embarcadero Dev-C++）
- Visual Studio 2022 Community：Windows 下基于微软 cl 编译器的商业 C/C++ 集成开发环境。（请到官网下载，社区版免费）
- GCC 或 Clang：跨平台（Windows/Linux/MacOS）的开源 C/C++ 编译器，可结合 Emacs、Vim、Visual Studio Code 等编辑器组装集成开发环境。

常见的集成开发环境通常通过源文件的扩展名来判定所使用的编程语言，从而选择相应语言的编译器对源代码进行编译。因此，C 程序源文件的扩展名必须为“.c”，而 C++ 程序源文件的扩展名必须为“.cpp”，否则可能因语法不一致导致编译错误。

1.1 绪论

✍ **练习 1.1** 面向对象程序设计的三大特点是什么？

提示： 封装、继承、多态。

✍ **练习 1.2** 封装的含义是什么？有哪些具体的实现机制？

提示： 封装是指将数据和操作数据的函数放到同一个类中。封装的实现机制主要包括类和访问权限。

✍ **练习 1.3** 多态的含义是什么？有哪些具体的实现机制？

提示： 多态是指相同的行为可以产生不同形式的效果，具体而言，某个函数调用（根据类或类型）对不同的对象调用不同的函数、进行不同的计算。编译时多态的实现机制主要包括函数重载、操作符重载，运行时多态的实现机制主要是虚函数。

✍ **练习 1.4** 面向过程程序设计和面向对象程序设计的区别和优点分别是什么？

提示： 面向过程程序设计就是用“函数”把解决问题所需要的步骤逐一实现，并依次调用。

- 优点：比面向对象程序设计性能更好，因为面向对象特性的实现需要消耗额外的时间和空间资源。因此单片机/嵌入式软件、Linux/Unix 等一般采用面向过程程序设计语言。
- 缺点：比面向对象程序设计更难于维护、复用、扩展。

面向对象程序设计就是用“类”把问题相关的事物及其操作逐一实现，并通过构造类的对象来解决问题。其中，建立类和对象的目的不是为了实现一个步骤，而是为了描述某个事物在整个解决问题的步骤中的行为。

- 优点：比面向过程程序设计更易于维护、复用、扩展，因为面向对象的封装、继承、多态的特性可以设计出低耦合的系统。因此大型软件一般采用面向对象程序设计语言。
- 缺点：比面向过程程序设计性能更低。

1.2 cout/cin 和格式化输出/输入

✍ **练习 1.5** 熟悉 C/C++ 编译器和集成开发环境，编辑、编译并运行以下程序，并分析结果。

```
#include <iostream>
using namespace std;
int main() {
  int x = 6, y;
  cout << "x = " << x << " y = " << y << endl;
  x = 6; y = ++x;
  cout << "x = " << x << " y = " << y << endl;
  cout << "input x: ";
  cin >> x; y = x++;
  cout << "x = " << x << " y = " << y << endl;
  x = 6; y = x--;
  cout << "x = " << x << " y = " << y << endl;
  cout << "input x: ";
  cin >> x; y = --x;
  cout << "x = " << x << " y = " << y << endl;
  return 0;
}
```

✍ **练习 1.6（求平均值）** 编写一个程序，提示用户输入 3 个数，然后在屏幕上输出这 3 个数以及它们的平均值。

✍ **练习 1.7（求中值）** 编写一个程序求一组整数的中值。如果这组整数的个数为奇数，那么中值就是排序后的中间那个数；如果这组整数的个数为偶数，那么中值就是排序后的中间两个数的平均值。编写一个函数接受如下两个参数：整型数组、代表该数组元素个数的一个整数。该函数应当返回数组的中值。注：为了练习指针的使用，请采用指针访问数组。

✍ **练习 1.8（口令检验）** 在许多软件中，都需要有口令。一个口令至少需要如下几个原则：（1）口令至少由 6 个字符构成；（2）口令中至少包含一个大写字母；（3）口令中至少包含一个小写字母；（4）口令中至少包含一个数字。编写一个口令检验程序，验证用户输入的口令是否

满足以上几个原则。如果不满足，显示一个信息，告诉用户为什么不满足。

✍ **练习 1.9（支票输出）** 编写一个程序模拟支票输出。程序要求用户输入日期、姓名和支票的金额，然后模拟支票的形式输出这些信息。假设输入的金额中，最多只有两位小数。程序需进行输入有效性检验：输入的金额不能大于 10000。例如：

```
日期：2017年12月26日
姓名：张三 RMB1920.68
人民币：壹仟玖佰贰拾元陆角捌分
```

1.3 布尔类型（bool）

✍ **练习 1.10** 读程序，写输出结果。思考为什么会有这样的输出结果。

```
#include <iostream>
using namespace std;
struct stb { bool a,b,c,d; };
struct stc { char c[8]; };
int main() {
  int a; bool b = 0; char c; double d;
  cout << sizeof(a) << " " << sizeof(b) << endl;
  cout << sizeof(c) << " " << sizeof(d) << endl;
  b = b + 1; cout << "b = " << b << endl;
  b = b + 1; cout << "b = " << b << endl;
  cout << sizeof(stb) << " " << sizeof(stc) << endl;
  return 0;
}
```

✍ **练习 1.11（身高预测）** 据有关生理卫生知识与数理统计分析表明，影响成人身高的因素包括遗传、饮食习惯与体育锻炼等。首先，成人的身高与其父母的身高和自身的性别密切相关，设 faHeight 为其父身高，moHeight 为其母身高，身高预测公式为：

$$\text{男性成人时身高} = (faHeight + moHeight) \times 0.54$$

$$\text{女性成人时身高} = (faHeight \times 0.923 + moHeight)/2$$

此外，如果有良好的卫生饮食习惯，那么可增加身高 1.5%；如果喜爱体育锻炼，那么可增加身高 2%。编写一个程序，提示用户从键盘输入用户的性别（用字符型变量 sex 存储，输入字符 F 表示女性，输入字符 M 表示男性）、父母身高（用实型变量存储，faHeight 为其父身高，moHeight 为其母身高）、是否有良好的饮食习惯等条件（用布尔型变量 diet 存储，输入 1 表示良好，输入 0 表示不好）、是否喜爱体育锻炼（用布尔型变量 sports 存储，输入 1 表示喜爱，输入 0 表示不喜爱），利用给定公式和身高预测方法对身高进行预测。

1.4 引用类型（&）

✍ **练习 1.12** 读程序，写输出结果。

```
#include <iostream>
using namespace std;
int main()
{
   int x = 2, y = 4;
   int &z = x;
   z++;
   cout << x << y << z;
   x = y;
   x++;
   cout << x << y << z;
   return 0;
}
```

✍ **练习 1.13（交换数值）** 编写一个函数 swap(int &x, int &y)，用于交换 x 和 y 的值。编写一个程序，提示用户输入 2 个数，然后调用该函数交换它们的数值，并在屏幕上输出结果。思考如何使用指针参数实现同样的功能。

1.5 函数的默认参数

1.6 函数重载

✍ **练习 1.14** 读程序，写输出结果。

```
#include <iostream>
#include <iomanip>
using namespace std;
int foo(int a, int n)
{ return a+n; }
int foo(int *a, int n)
{ return a[n]; }
int main()
{
   int a[9] = {  1,2,3,4,5,6,7,8,9  };
   for (int i = 0; i < 4; i++)
```

```
    {
        cout << setw(2) << foo(a[i], i);
        if (i % 3 == 2)
            cout << setw(2) << foo(a, i);
    }
    return 0;
}
```

1.7 内存的动态分配和释放（new/delete）

常见错误解析：

1. 返回指向栈空间的指针，导致释放后使用错误（在函数返回后，访问已释放的空间）。

```
char *f()
{
    char arr[200];
    ...
    return arr;
}
int main()
{
    char *str = f();
    str[0] = str[1];
    return 0;
}
```

2. 只有 new，没有 delete，导致内存泄漏。
3. 分配多余的内存空间，释放不彻底，导致内存泄漏。

```
char *merge(char *str1, char *str2)
{
    char *strx = new char[200];
    ...
    return strx;
}
int main()
{
    ....
    char *str3 = new char[100];
```

```
    str3 = merge(str1, str2);
    delete [] str3;
    return 0;
}
```

4. 释放栈空间的数组，导致内存错误；同时，申请的堆空间未使用也未释放。

```
char* merge(char *str1, char *str2)
{  return str1; }
int main()
{
    char s[100], t[100], *c;
    c = new char[length];
    c = merge(s, t);
    delete [] c;
}
```

✍ **练习 1.15（字符串合并）** 编写一个程序，提示用户输入两个已经按从小到大顺序排列好的字符串，将两个字符串合并为一个新的字符串。要求：编写一个合并两个字符串的函数 char *merge(char *str1, char *str2)，用 new 分配新的字符数组（大小为 str1 和 str2 的长度之和 +1），然后将 str1 和 str2 合并存入该字符数组，使合并后的字符串仍然从小到大排列，并返回该字符数组。在主函数中调用该函数输出合并后的字符串，并用 delete 释放分配的字符数组。必须用指针方式逐个访问字符数组元素，不得使用字符串库函数。

运行结果	输入： 1223aabcc 233abbdkm 输出： 1222333aaabbbccdkm

第 1 章作业

1. 邮件地址检验：邮件地址需要满足格式“x@y.z”，其中，x、y、z 的长度均不超过 20 个字符，且只能由字母、数字和下划线构成，但是 z 可以包含 2 个点号。编写一个邮件地址检验程序，验证用户输入的邮件地址是否满足以上格式。如果不满足，显示一个信息，告诉用户为什么不满足。
2. 学生成绩统计：编写一个程序，提示用户输入一个班每个学生的姓名和一门课的成绩（最多不超过 10 人），存入一个结构体数组中，当输入成绩为负值时，输入结束，使用函数分别实现以下功能（不使用全局变量）：
   (a). 统计不及格人数并打印不及格学生名单；
   (b). 统计成绩在全班平均分之上的学生人数，并打印这些学生的名单；
   (c). 将成绩分为六个分数段，60 分以下为第 1 段，60–69 为第 2 段，70–79 为第 3 段，80–89 为第 4 段，90–99 为第 5 段，100 分为第 6 段，统计各分数段的人数。

   编写主函数测试以上函数的正确性。

3. 加法函数重载：利用函数重载实现十进制（int）、字符串（char *）、无符号二进制（struct BinNum）的加法。加法函数格式为：TYPE add(TYPE n1, TYPE n2)，其中 TYPE 为各种不同的类型。要求：（1）字符串的加法是将 char *n2 追加到 char *n1 的尾部，该功能不得使用 C++ 的 string 类；（2）二进制的类型为结构体 BinNum（其中包含一个字符数组），加法用 BinNum add(BinNum n1, BinNum n2) 实现。编写主函数，调用这些加法并在屏幕上输出结果。

   例如输入：1122 2211 abc ab 00110 00010

   输出：3333, abcab, 01000
4. 字符串删除：编写一个程序，提示用户输入两个字符串，将第二个字符串中包含的字符从第一个字符串中删除。要求：编写一个字符删除函数 char *remove(char *str, char *chars)，用 new 分配新的字符数组（大小为 str 的长度 +1），将 chars 中包含的字符从 str 中删除后存入该字符数组，并返回该字符数组。在主函数中调用该函数输出结果，并用 delete 释放分配的字符数组。必须用指针方式逐个访问字符数组元素，不得使用字符串库函数。

   例如输入：fjejfpifhjg933pakpj ajf3

   输出：epihg9pkp

第 2 章 文件操作

内容提要

- 文件流类型（fstream）
- 打开文件和关闭文件（open、close）
- 读写文本文件（流操作符、成员函数）
- 读写二进制文件（read、write）
- 随机读写文件（seep、seekg）
- 流对象作为函数参数

2.1 文件流类型（fstream）

2.2 打开文件和关闭文件（open、close）

2.3 读写文本文件（流操作符、成员函数）

✍ **练习 2.1（文本输出）** 编写一个程序，要求用户输入文件名，在屏幕上显示文件的内容。在显示时，每行前面都要带上一个行号和一个冒号。行号从 1 开始，例如：

```
1: This a test
2: for you.
3: Date is 2022 5 31.
```

如果一屏显示不完文件的内容，那么显示 24 行后，暂停一下，等待用户按任意键以后继续显示后面的 24 行。采用编辑器（如记事本）创建一个文本文件，以测试这个程序。

✍ **练习 2.2（词频统计）** 编写一个程序，提示用户输入一个文本文件名，统计该文件中各个单词出现的次数。采用编辑器（如记事本）创建一个文本文件，以测试这个程序。

✍ **练习 2.3（文本复制）** 编写一个程序，定义 fstream 对象 in，与文本文件 file1.txt 建立关联，然后定义 fstream 对象 out，与文本文件 file2.txt 建立关联。打开文本文件 file1.txt 成功后，将其内容逐个字符读出，将小写字母转换成大写字母，并输出到 file2.txt 中。采用编辑器（如记事本）创建一个文本文件，以测试这个程序。

✍ **练习 2.4（文本追加）** 编写一个程序，定义 fstream 对象 in，与文本文件 file1.txt 建立关联，然后定义 fstream 对象 out，与文本文件 file2.txt 建立关联。打开文本文件 file1.txt 成功后，将其内容追加到 file2.txt 的尾部。采用编辑器（如记事本）创建一个文本文件，以测试这个程序。

✍ **练习 2.5（文本加解密）** 编写一个文件加密解密程序。首先，将一个文件中的内容，按照一定的方法，对每个字符加密后存储到第二个文件中。尽管加密技术很多，你可以采用一种简单的加密方法，例如将每个字母的 ASCII 码加 2。然后，将这个加密文件解密，然后写到第三个文件中。采用编辑器（如记事本）创建一个文本文件，以测试这个程序。

2.4 读写二进制文件（read、write）

✍ **练习 2.6（学生成绩读写）** 编写一个程序，实现以下功能：

1. 输入 10 个学生信息（包括姓名、学号和三门课程成绩），将学生信息存入一个学生结构体数组，然后将数组存入二进制文件 student.dat 中；
2. 从 student.dat 文件中读出这些信息并显示到屏幕。

✍ **练习 2.7（学生通讯录管理）** 编写一个程序，将下面的学生信息存储到二进制文件中：

1. name：具有 21 个元素的字符数组；
2. age：一个整型变量。输入的年龄不能为负，也不能大于 200；
3. address：具有 51 个元素的字符数组；
4. phone：具有 14 个元素的字符数组；
5. E-mail：具有 51 个元素的字符数组。

该程序具有一个菜单，便于用户完成如下操作：

1. 向文件增加记录；
2. 显示文件中的所有记录；
3. 按照姓名查找某个学生的记录；
4. 按照姓名查找并修改某个学生的记录；
5. 按照姓名查找并删除某个学生的记录；
6. 统计文件中学生的个数、计算学生的平均年龄。

2.5 随机读写文件（seep、seekg）

✍ **练习 2.8** 读程序，写输出结果。

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
  fstream file ("nums.txt", ios :: in | ios :: out | ios :: trunc);
  char ch;
  if (! file )    exit (0);
  file << "123456789";
  cout << file . tellp ();
  file .seekg(3L, ios :: beg);
  file . get(ch);          cout << ch;
  file .seekg(-3L, ios :: end);
  file . get(ch);          cout << ch;
```

```
    file .seekg(-3L, ios :: cur);
    file . get(ch);          cout << ch;
   cout << file . tellg ();
    file . close ();
    return 0;
}
```

2.6 流对象作为函数参数

第 2 章作业

1. 文本输出：编写一个程序，要求用户输入文件名，在屏幕上显示文件的内容。在显示时，每行后面都要带上一个冒号和该行的单词数目。例如：

   ```
   This a test : 3
   for you. : 2
   Date is 2022 5 31. : 5
   ```

   如果一屏显示不完文件的内容，那么显示 24 行后，暂停一下，等待用户按任意键以后继续显示后面的 24 行。采用编辑器（如记事本）创建一个文本文件，以测试这个程序。
2. 文本转存：编写一个文件转存程序。首先，将输入文件中的每一行的单词逆序排列后存储到第二个文件中。然后，将输入文件中的每一行的单词按字典序排列后存储到第三个文件中。采用编辑器（如记事本）创建一个文本文件，以测试这个程序。
3. 酒店客房管理：编写一个程序，将下面的酒店客房信息存储到二进制文件中：
   (a). id：客房号，具有 10 个元素的字符数组；
   (b). price：价格，整型变量。价格不能为负，也不能大于 10000；
   (c). booked：是否已经被预订（包括已入住，但可能尚未入住），布尔变量；
   (d). cname：住客姓名，具有 20 个元素的字符数组；
   (e). cid：住客身份证号码，具有 20 个元素的字符数组。

   该程序具有一个菜单，便于用户完成如下操作：
   (a). 增加客房信息：向文件增加记录；
   (b). 删除客房信息：从文件删除记录；
   (c). 显示客房信息：显示文件中的所有记录；
   (d). 预订客房功能：修改某个客房的预订记录；
   (e). 办理入住功能：修改某个客房的住客记录；
   (f). 按照姓名查找某个住客的客房号；
   (g). 办理退房功能，删除某个客房的住客记录；
   (h). 统计文件中客房的个数、被预订但未入住的客房个数、入住的客房个数。

第 3 章 类的基础部分

内容提要

- 类（成员变量、成员函数、私有公有）
- 对象（类的实例/变量）
- 内联函数
- 构造函数
- 析构函数
- 对象数组和对象指针

3.1 类（成员变量、成员函数、私有公有）

3.2 对象（类的实例/变量）

练习 3.1 读程序，写输出结果。练习程序的多文件组织。E.h 的内容如下：

```
#ifndef IS_E_H_INCLUDED
#define IS_E_H_INCLUDED
#include <iostream>
using namespace std;
class C {
  int a;
public:
  void set(int i);
  void print();
};
#endif
```

E.cpp 的内容如下：

```
#include "E.h"
void C::set(int i) {
  a = i;
}
void C::print() {
  cout << a << endl;
}
```

example.cpp 的内容如下：

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

```
#include "E.h" // try multiple inclusion
int main() {
  C obj;
  obj.set(100);
  obj.print();
  return 0;
}
```

✍ **练习 3.2** 读程序，写输出结果。思考成员函数 print() 是否占用类 C 的内存空间。注意，print(C*) 实际上相当于成员函数 print() 的编译结果。

```
#include <iostream>
using namespace std;
class C {
public:
  int a;
  float b;
public:
  void print();
};
void C::print() { cout << "P1: (" << a << "," << b << ")" << endl; }
void print(C c) { cout << "P2: (" << c.a << "," << c.b << ")" << endl; }
void print(C *c) { cout << "P3: (" << c->a << "," << c->b << ")" << endl; }
int main() {
  C c;
  c.a = 1; c.b = 2;
  cout << sizeof(C) << " " << sizeof(c) << endl;
  c.print();
  print(c);
  print(&c);
  return 0;
}
```

✍ **练习 3.3** 读程序，写输出结果。

```
#include <iostream>
using namespace std;
class C {
public:
  int a;
```

```
    float b;
private:
  void data() { cout << "(" << a << "," << b << ")" << endl; }
public:
  void show() {
    cout << "This is the data: ";
    data();
  }
};
int main() {
  C obj;
  obj.a = 1;
  obj.b = 2;
  obj.show(); // try data() here
  return 0;
}
```

3.3 内联函数

3.4 构造函数

练习 3.4 读程序，写输出结果。

```
#include <iostream>
using namespace std;
class C // Can here be class or struct?
{
public: // Can here be public or private?
  int a;
  float b;
  // How about add another member: char c;
public:
  C() { cout << "constructor" << endl; }
  void print(){ cout << "(" << a << "," << b << ")" << endl; }
};
int main() {
  C c;
```

```
    c.a = 1;
    c.b = 2;
    c.print();
    cout << "size = " << sizeof(C) << endl;  // Can here be C or c?
    return 0;
}
```

3.5 析构函数

✍ 练习 3.5 读程序，写输出结果。

```
class Sac {
    int n;
public:
    Sac() : n(1) { cout << n; }
    Sac(int k) : n(k) { cout << n + 1; }
    ~Sac() { cout << n + n; }
};
int main() {
    Sac s1, *s2;
    s2 = new Sac(2);
    s2 = new Sac(3);
    delete s2;
    return 0;
}
```

✍ 练习 3.6 读程序，写输出结果。

```
#include <iostream>
using namespace std;
class C {
public:
    char* str;
public:
    C() {
        str = new char[100];
        cout << "constructor ()" << endl;
    }
    C(const char *s) {
```

```
        str  = new char[100];
        strcpy ( str , s);
        cout << " constructor (s)" << endl;
    }
    ~C() {
        if ( str ) delete [] str ;
        cout << " destructor " << endl;
    }
};
int main() {
    C c;
    // try c.C(); here
    // try c.~C(); here
    C* p = new C("abcd");
    delete p;
    cout << "termnating" << endl;
    return 0;
}
```

✍ **练习 3.7（学生类）** 编写一个程序，定义学生类 student，要求：

1. 包含姓名 name（字符指针），年龄 age（整数），地址 address（字符指针）等成员变量；
2. 包含成员函数，用于在屏幕输出学生的信息；
3. 包含构造函数，用于对成员变量初始化，尤其是对姓名、地址进行动态内存分配；
4. 重载构造函数，用多种方式对成员变量初始化；
5. 包含析构函数，用于释放已分配的内存。

编写主函数，创建学生对象，测试构造函数和析构函数的调用顺序、成员函数的功能。

✍ **练习 3.8（动态数组类）** 设计一个类，它具有一个浮点数指针类型的成员变量（用于存储动态数组的地址）和两个整型成员变量（用于存储动态数组的大小和有效元素的个数）。构造函数具有一个整型参数 count，它为指针成员变量分配 count 个元素的空间。析构函数释放指针成员变量指向的空间。另外，设计 5 个成员函数完成如下功能：

1. 向指针指向的空间末尾追加存储数据；
2. 删除指针指向的空间末尾的数据（变为非有效元素）；
3. 返回所有元素的总和；
4. 返回所有元素的平均值；
5. 在屏幕上显示所有元素。

编写一个完整的程序检验该类的正确性。

3.6 对象数组和对象指针

✍ 练习 3.9 读程序，写输出结果。

```
#include <iostream>
using namespace std;
class C {
public:
  int a;
  float b;
public:
  void print() { cout << "(" << a << "," << b << ")" << endl; }
};
int main() {
  C obj;
  obj.a = 1;
  obj.b = 2;
  obj.print();
  C* p = &obj;
  p->print();
  return 0;
}
```

✍ 练习 3.10 读程序，写输出结果。

```
class Item {
  char name[20];
  int count;
public:
  Item(char *n = "1", int c = 1) {
    strcpy(name, n);
    count = c;
    cout << "C" << name << count << " ";
  }
  ~Item() {
    cout << "D" << name << " ";
  }
};
```

```
int main() {
  Item Obj[3] = { Item("2",3) ,Item("4") };
  return 0;
}
```

✍ **练习 3.11（学生管理）** 编写一个程序，定义学生类 student，要求包含姓名 name（字符指针），年龄 age（整数），地址 address（字符指针）等成员变量。编写主函数，建立一个对象数组，用于存储 5 个学生的信息，然后用指针指向数组首元素，输出第 1、3、5 个学生的信息。

✍ **练习 3.12（薪水计算）** 设计一个计算薪水的类 payroll，它的成员变量包括：单位小时的工资、本周已经工作的小时数、本周应付工资数。编写一个程序，定义一个具有 10 个元素的对象数组（代表 10 个雇员），询问每个雇员本周已经工作的小时数，然后显示本周应付工资数。程序需进行输入有效性检验：每个雇员每周工作的小时数不能大于 60，同时也不能为负数。

✍ **练习 3.13（商品销售）** 设计一个商品条目类 invoiceItem，要求：

1. 包含商品名称 desc（字符指针）、商品数量 units（整型）、成本价 cost（浮点型）等成员变量；
2. 包含价格函数，在成本价的基础上加上 30% 的利润，得到每个商品的单价；
3. 包含计税函数，将商品价格乘 6%，得到商品的销售税。

设计一个商品销售类 sale，完成如下功能：

1. 询问客户购买的商品名称和数量，购买的商品数量不能为负数；
2. 询问每个商品的成本价，存入 invoiceItem 对象；
3. 将商品单价与购买商品的数量相乘，得到商品小计；
4. 将商品小计与商品销售税相加得到该商品的销售额；
5. 显示客户本次交易购买商品的小计、销售税和销售额。

编写一个程序测试以上两个类的正确性。

第 3 章作业

1. 编写一个程序，定义课程类 course，要求：
   (a). 包含课程名 name（字符指针）和该课程的学分 credit（浮点数）等成员变量；
   (b). 包含成员函数，用于在屏幕输出课程的信息；
   (c). 包含构造函数，用于对成员变量初始化，尤其是对课程名进行动态内存分配；
   (d). 重载构造函数，用多种方式对成员变量初始化；
   (e). 包含析构函数，用于释放已分配的内存。
   编写主函数，创建课程对象，测试构造函数和析构函数的调用顺序、成员函数的功能。
2. 开销计算：设计一个计算个人开销的类 cost，它的成员变量包括：每天的餐饮开销 meal、每月的交通开销 transport、一年的学习开销 study。编写一个程序，定义一个具有 10 个元素的对象数组（代表 10 个人），询问每个人的每项开销，然后显示一年（365 天）的总开销。程序需进行输入有效性检验：每个人每天的餐饮开销不能小于 20。

3. 酒店客房管理：编写一个程序，用于管理 180 间客房（9 层楼，每层楼 20 间客房），定义客房类 room，要求包含房号 id（整数，范围从 101 到 920）、价格 price（整数）、入住的顾客姓名 cname（字符指针）和身份证号 cid（字符指针）等成员变量，还包含成员函数，实现订房功能（增添住客的信息）和退房功能（打印消费金额、删除住客信息）。编写主函数，建立一个 9*20 的二维对象数组，用于存储客房的信息，然后演示订房和退房功能（要求用指针指向数组元素进行操作）。

第 4 章 类的高级部分

内容提要

- 静态成员（static）
- 友元（friend）
- 拷贝构造函数
- 运算符重载
- 对象组合

4.1 静态成员（static）

练习 4.1 读程序，写输出结果。

```
#include <iostream>
using namespace std;
class C {
public:
    static int count;
public:
  C() {
    count++;
    cout << " constructor " << endl;
  }
  ~C() {
    count--;
    cout << " destructor " << endl;
  }
  void PrintCount() {
    cout << count << endl;
  }
   static void StaticPrintCount () {
    cout << count << endl;
  }
};
int C::count; // = 0
int main() {
  // Print static members
  cout << C::count << endl;
```

```
    C c1;
    cout << C::count << endl;
    cout << c1.count << endl;
    C c2;
    cout << C::count << endl;
    cout << c1.count << endl;
    cout << c2.count << endl;
    // Call static functions
    C::StaticPrintCount();
    c1.StaticPrintCount();
    c2.StaticPrintCount();
    // Call non-static functions
    c1.PrintCount();
    // try C::PrintCount();
    return 0;
}
```

✍ **练习 4.2（学生类）** 编写一个程序，定义学生类 student，要求：

1. 包含姓名 name（字符指针）、成绩 score（整数）等成员变量；
2. 包含静态成员变量：学生人数 count 和所有学生成绩之和 totalScore；
3. 包含构造函数用于为姓名分配存储空间、将学生人数加 1；
4. 包含析构函数用于释放姓名的存储空间、将学生人数减 1；
5. 包含成员函数 void account() 用于设置分数，同时累计成绩之和；
6. 包含成员函数 int compScore(student &st2)，实现学生的成绩比较，如果 this 对象的成绩大于 st2 则返回 1，相等则返回 0，小于则返回-1。
7. 静态成员函数 int compScore(student &st1，student &st2)，实现学生的成绩比较；
8. 静态成员函数 int sum() 用于返回学生成绩之和；
9. 静态成员函数 double average() 用于求全班成绩的平均值。

编写主函数，输入 5 个学生的成绩，存入对象数组，将学生按成绩从高到低排序后输出，然后调用上述函数求出全班学生的成绩之和和平均分并输出。

4.2 友元（friend）

✍ **练习 4.3** 读程序，写输出结果。注意友元的作用、类的声明顺序。

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

```
class A {
public:
  void func(C c); // It depends on the declaration of C
};
class C
{
private:
  int i;
  friend void func(C);
  friend void A::func(C); // It depends on the definition of A::func
};
void func(C c) {
  cout << c.i << endl;
}
void A::func(C c) {
  cout << c.i << endl; // // It depends on the definition of C::i
}
int main() {
  return 0;
}
```

✍ **练习 4.4（投资类）** 编写股票投资类 StockInvest 和艺术品投资类 ArtInvest。每个类都包含一个私有数据 int invest，用于存放投资者在该投资项目的投资额。每个类都有一个构造函数、一个 setInvest() 函数用于输入投资额、一个 disp() 函数用于显示投资额、一个友元函数 void total(StockInvest A, ArtInvest B)，该函数是一个外部函数，用于计算并显示投资者在这两个项目中的总投资额。编写主函数演示如下功能：定义两个类的对象，然后分别显示投资者在两个项目的投资额和总投资额。

4.3 拷贝构造函数

✍ **练习 4.5** 读程序，写输出结果。思考为什么缺少拷贝构造函数会导致内存错误。

```
#include <iostream>
using namespace std;
class C {
private:
  char *str;
public:
```

```
  C() {
    str = new char[100];
  }
  ~C() { delete [] str; }
  void setString(const char* s) {
    strcpy(str, s);
  }
  void print() {
    cout << str << endl;
  } /*
  C(const C &c) // copy constructor
  {
    str = new char[100];
    strcpy(str, c.str);
  }*/
};
int main() {
  C c1;
  c1.setString("abc");
  c1.print();

  C c2 = c1;
  c2.setString("xyz");
  c2.print();

  c1.print(); // Note: here should print "abc".
  // Memory error: double delete.
  return 0;
}
```

✍ **练习 4.6（公司类）** 设计一个公司类 Company，有两个私有成员变量：char name[15] 表示公司名称，int year 表示公司成立年份，还有一个静态成员变量 int total 描述公司对象的总数。

1. 设计构造函数，实现名称和年份的初始化，公司总数加 1；
2. 设计拷贝构造函数，实现名称和年份的拷贝，公司总数加 1；
3. 设计析构函数，实现公司总数减 1；
4. 定义一个外部友元函数 void ShowInfo(Company c) 显示公司 c 的名称和年份；
5. 设计一个静态成员函数 void ShowTotal()，显示公司对象总数；

编写主函数演示上述功能。

4.4 运算符重载

✍ 练习 4.7 读程序，写输出结果。

```
#include <iostream>
using namespace std;
class C {
public:
  C() {
    cout << "A" << endl;
  }
  C(const C &c) { // try C(C c) here
    cout << "B" << endl;
  }
  void operator=(const C& c) {
    cout << "C" << endl;
  }
  ~C() {
    cout << "D" << endl;
  }
};
C f(C c) { // try "C &f(C &c)" here
  cout << "in f()" << endl;
  return c;
}
int main() {
  C c1;
  C c2(c1);
  C c3 = c1;
  c3 = f(c1);
  cout << "terminating" << endl;
  return 0;
}
```

✍ 练习 4.8 读程序，写输出结果。

```
#include <iostream>
using namespace std;
class counter {
```

```
private :
  unsigned value;
public :
  counter(unsigned int a) { value = a; }
  counter &operator++() {
    value++;
    return *this;
  }
  counter &operator++(int) {
    value++;
    return *this;
  }
  void display () {
    cout << value << " ";
  }
};
int main() {
  counter i(0), n(10);
  i = ++n;
  i.display(); n.display();
  i = n++;
  i.display(); n.display();
  return 0;
}
```

练习 4.9 读程序，写输出结果。

```
class test {
  long x;
  static int num;
public :
  test (long m = 0) { x = m; }
  test operator++();
  test operator++(int);
  void Show() {
    cout << x << num;
  }
};
```

```
test  test :: operator++() {
  cout << x++ << num++;
  return *this;
}
test  test :: operator++(int) {
  test temp(*this);
  cout << ++x << ++num;
  return temp;
}
int  test :: num = 0;
int main() {
  test a(4), b = a;
  ++a;
  a++;
  b.Show();
  return 0;
}
```

练习 4.10 读程序，写输出结果。

```
class point {
  double X, Y;
public:
  point(double xx = 0, double yy = 0) {
    X = xx;   Y = yy;
    cout << "cons(" << X << "," << Y << ") ";
  }
  point(point &p) {
    X = p.X;   Y = p.Y;
    cout << "copy(" << X << "," << Y << ") ";
  }
  point &operator=(point &p) {
    X = p.X;   Y = p.Y;
    cout << "op=(" << X << "," << Y << ") ";
    return *this;
  }
};
point f() {
```

```
    point d(3, 4);
    return d;
}
int main() {
    point d1(1, 2), b = d1;
    b = f();
    return 0;
}
```

✍ **练习 4.11（学生类赋值运算符）** 已知学生类 student 包括两个成员变量：int id 和 char *name 分别表示学号和姓名。请编写构造函数、析构函数，重载学生类的 = 赋值运算符。编写主函数演示如何使用重载操作符。

✍ **练习 4.12（日期类运算符）** 已知日期类 date 包括三个成员变量：int y，int m 和 int d 分别表示年、月、日。请编写构造函数、析构函数，重载日期类的 < 和 > 运算符实现日期大小比较，重载日期类的前缀和后缀 ++ 和--运算符实现对天数的加 1 和减 1。注意，月末加 1 需要修改日和月，年末加 1 需要修改日、月和年。编写主函数演示如何使用重载操作符。

✍ **练习 4.13（复数类运算符）** 已知复数类 complex 包括两个成员变量：float real 和 float img 分别表示实部和虚部。请编写构造函数、析构函数，重载复数类的 + 和-运算符（成员函数形式）实现两个 complex 对象的加法和减法（real 和 img 分别相加和相减），重载! 运算符（友元函数形式）实现复数实部和虚部的交换，重载 << 运算符实现复数的格式化输出，格式为 real+img i，例如：2+3i，-2-3i 等。编写主函数演示如何使用重载操作符。

✍ **练习 4.14（工作时间类）** 设计一个工作时间类 NumDays，它的功能是将以小时（hours）为单位的工作时间，转换为天数（days）。例如，8 个小时转换为 1 天，12 小时转换为 1.5 天。该类的构造函数具有一个代表工作小时的参数，此外还有一些成员函数，实现小时和天的存储和检索。同时，该类还要重载下列操作符：

1. +：加操作符。当两个 NumDays 对象相加时，重载后的 + 操作符函数应当返回这两个对象的 hours 成员之和。
2. -：减操作符。当两个 NumDays 对象相减时，重载后的-操作符函数应当返回这两个对象的 hours 成员之差。
3. ++：前置增 1 操作符和后置增 1 操作符。这两个函数的功能是对 NumDays 对象的 hours 成员变量增 1。hours 增加以后，应当自动重新计算对应的天数。
4. --：前置减 1 操作符和后置减 1 操作符。这两个函数的功能是对 NumDays 对象的 hours 成员变量减 1。hours 减 1 以后，应当自动重新计算对应的天数。

编写主函数演示如何使用重载操作符。

4.5 对象组合

✍ **练习 4.15** 读程序，写输出结果。

```
#include <iostream>
using namespace std;
class A {
  const char* str;
public:
  A(const char* s) {
    str = s;
    cout << str << endl;
  }
  ~A() { cout << "~" << str << endl; }
};
class C {
    A a; // Note the order of constructions
    A b;
    A c;
public:
  C(const char *sa, const char *sb, const char *sc) : c(sc), a(sa), b(sb) {
    cout << "D" << endl;
  }
  ~C() { cout << "~D" << endl; }
};
int main() {
  C c("A", "B", "C"); // Note the order of destructions
  return 0;
}
```

✍ **练习 4.16** 读程序，写输出结果。

```
class part {
private:
  int value;
public:
  part(int i = 0) { cout << "P1-"; value = i; }
  ~part() { cout << "P2-"; }
  void print() { cout << value; }
```

```
};
class whole {
private :
  part one;
  part two;
  int n;
public :
  whole(int i, int j = 1, int k = 0) : two(i), one(j), n(k) { cout << "W1-"; }
  ~whole() { cout << "W2-"; }
  void print ();
};
void whole:: print () {
  one. print ();
  two. print ();
  cout << n << "-";
}
int main() {
  whole obj (1, 0);
  obj . print ();
  return 0;
}
```

✍ **练习 4.17（休假类）** 设计一个休假类 TimeOff，用于计算雇员生病和休假的时间。该类包含 6 个 NumDays 类型的成员对象（参见练习4.14）：

1. maxSickDays：用于记录雇员因生病可以不工作的最多天数。
2. sickTaken：用于记录雇员因生病已经不工作的天数。
3. maxVacation：用于记录雇员可以带薪休假的最多天数。雇员带薪休假累计不能超过 24 个小时，因此该对象的 hours 成员不能大于这个数。
4. vacTaken：用来记录雇员已经带薪休假的天数。
5. maxUnpaid：用来记录在不支付薪水的情况下，雇员可以休假的最多天数。
6. unpaidTaken：用来记录在不支付薪水的情况下，雇员已经休假的天数。

此外，该类还应当有一些用于存储雇员姓名和工号的成员变量，并提供适当的构造函数和成员函数，用于存储和检索上面成员对象的信息。编写主函数，定义该类的一个对象，要求用户输入某雇员已经工作的月数（months），然后采用 TimeOff 类对象计算并显示雇员因病休假和正常休假的最多天数。注：雇员每月可以有 12 小时的带薪休假和 8 小时的生病休假。

第 4 章作业

1. 坐标点类：设计一个描述直角坐标系中坐标点的类 point，包含三个私有成员变量：char

*name 指向存储坐标点名字的字符数组，float x 和 float y 表示 x 坐标和 y 坐标的值。请为该类编写构造函数、拷贝构造函数，然后分别编写成员函数和友元函数计算给定两个坐标点之间的距离。编写主函数创建两个对象 A 和 B，要求 A 的坐标点在第 2 象限，B 的坐标点在第 3 象限，并按如下格式输出结果：

```
n1(x1,y1) n2(x2,y2)
Distance 1 = d1
Distance 2 = d2
```

其中：n1、n2 为指定的坐标点名字，x1、y1、x2、y2 为指定的坐标值，d1 和 d2 分别为采用成员函数和友元函数计算得到的两个坐标点之间的距离。

2. 坐标点类运算符：请为上一题中的 point 类重载 ==、!=、>、<< 运算符，其中，给定两个 point 对象 p1 和 p2，
   (a). p1 == p2，即 p1.x == p2.x 并且 p1.y == p2.y；
   (b). p1 != p2，即 p1.x != p2.x 或者 p1.y != p2.y；
   (c). p1 > p2，即 p1.x > p2.x 或者 p1.x == p2.x 并且 p1.y > p2.y；
   (d). cout « p1 可以按 name(x,y) 的格式输出一个 point 对象。
   编写主函数演示如何使用重载操作符。
3. 日期类和人员类：设计日期类 date，包含 3 个成员变量：int year、int month 和 int day 分别表示年、月、日；然后设计人员类 person，包含 4 个成员变量：姓名 name（字符指针），性别 gender（字符），生日 birthday（date 类对象）、现金 cashFlow（整数），还包含一个静态成员函数 int getCount() 返回总人数。为这两个类设计成员函数，保证满足以下测试要求：

```
int main() {
  person p1, p2("zhang",'f',date(1991,1,1)), p3(p2);
  p3.print(); //输出: 姓名:zhang; 性别:女; 生日:1991-1-1;
  cout << "收支为：" << p3.getCashFlow() << "元" << endl; //输出：0元
  person *p = new person("li",'m',date(1990,2,2),800);
  p->print(); //输出: 姓名:li; 性别:男; 生日:1990-2-2;
  cout << "收支为：" << p->getCashFlow() << "元" << endl; //输出：800元
  cout << "总人数：" << p1.getCount() << endl; //输出：4
  delete p;
  cout << "总人数：" << p1.getCount() << endl; //输出：3
  return 0;
}
```

第 5 章 继承、多态和虚函数

内容提要

- 继承、多重继承
- 继承中的私有/保护/公有成员
- 继承中的构造函数和析构函数
- 覆盖基类的成员函数（静态绑定）
- 虚函数（动态绑定）
- 多继承

5.1 继承、多重继承

练习 5.1（坐标点类和球类） 设计一个描述三维坐标系中坐标点的类 point3D，包含 3 个成员变量分别表示 x 轴、y 轴、z 轴坐标。从 point3D 类公有派生出球类 ball，继承基类成员表示中心坐标，增加一个表示半径的成员变量，并通过成员函数实现体积和表面积的计算、输出中心坐标等。编写一个完整的程序，要求用户从键盘输入球的信息，然后在屏幕上显示这些计算结果，以验证程序的正确性。

练习 5.2（雇员类和工资雇员类） 设计一个雇员类 Employee，包含如下成员变量：姓名（字符指针）、编号（字符数组）、受雇日期，其中编号的格式为 XXX-L，此处的 X 是 0-9 之间的数字，L 是 A-M 之间的一个字母。向该类增加构造函数、析构函数和其他相关的成员函数，其中，构造函数能动态分配内存以存储姓名，析构函数能释放分配的空间。

然后设计一个 Employee 类的子类 EmployeePay，包含如下成员变量：月工资（浮点数）、部门号（整型）。向该类增加相关的成员函数，用于在屏幕上输出雇员的信息。编写一个完整的程序，要求用户从键盘输入雇员的信息，然后在屏幕上显示这些信息，以验证程序的正确性。注意输入数据的有效性检验。

练习 5.3（钟点工类） 设计一个 EmployeePay 类的子类 HourlyPay，包含如下成员变量：正常工作每小时的工资（不能为负数，也不能大于 50 元）、超时工作每小时的工资（不能为负数，也不能大于 100 元）、每月正常工作的小时数（不能为负数，也不能大于 100 小时）和已经工作的小时数（不能为负数，也不能大于 176 小时）。编写一个完整的程序，要求用户从键盘输入雇员的信息，然后在屏幕上显示这些信息以及月工资，以验证程序的正确性。注意输入数据的有效性检验。

5.2 继承中的私有/保护/公有成员

5.3 继承中的构造函数和析构函数

练习 5.4 读程序，写输出结果。

```
class  A {
```

```
public:
  A()  { cout << "1 "; }
  ~A() { cout << "2 "; }
};
class B :public A {
public:
  B()  { A a; cout << "3 "; }
  ~B() { cout << "4 "; }
};
int main() {
  B b;
  return 0;
}
```

练习 5.5 读程序，写输出结果。

```
#include <iostream>
using namespace std;
class A {
public:
  int i;
  A()  { cout << "A1"; }
  A(int ii) { i = ii;  cout << "A2"; }
  ~A() { cout << "B"; }
};
class C : public A {
  int j;
public:
  C()  { cout << "C1"; }
  C(int ii, int jj) : A(ii) { j = jj;  cout << "C2"; }
  ~C() { cout << "D"; }
};
int main() {
  C c1; // Note the order of constructions and destructions
  C c2(0, 0);
  cout << "E";
  return 0;
}
```

5.4 覆盖基类的成员函数（静态绑定）

5.5 虚函数（动态绑定）

练习 5.6 读程序，写输出结果。

```
#include <iostream>
using namespace std;
class A {
public:
  void f() { cout << "Af "; }
  virtual void g() { cout << "Ag "; }
};
class C : public A {
public:
  int i;
  void f() { cout << "Cf "; }
  void g() { cout << "Cg "; }
  void h() { cout << "Ch" << i << " "; }
};
int main() {
  A a, *pa;
  C c, *pc; c.i = 0;

  a.f(); // Af
  a.g(); // Ag

  c.f(); // Cf
  c.g(); // Cg

  pa = &c; // (基类指针指向派生类对象)
  pa->f();
  pa->g();
  //pa->h(); is not allowed

  pc = &c;
  pc->f(); // Cf
  pc->g(); // Cg
```

```
  pc->h(); //Ch0

  pc = (C*)&a; // this conversion is risky (派生类指针指向基类对象)
  pc->f();
  pc->g();
  pc->h(); // this call is risky
  return 0;
}
```

思考以下程序是否与上面的程序类似。

```
#include <iostream>
using namespace std;
class A {
public:
  void f() { cout << "Af "; }
  virtual void g() { cout << "Ag "; }
};
class C : public A {
public:
  int i;
  void f() { cout << "Cf "; }
  void g() { cout << "Cg "; }
  void h() { cout << "Ch" << i << " "; }
};
int main() {
  A a;
  C c; c.i = 0;
  A& ra = c; // (基类引用指向派生类对象)
  ra.f(); //Af
  ra.g(); //Cg
  //ra.h(); is not allowed
  //C& rc = a; is not allowed (派生类引用指向基类对象)
  return 0;
}
```

✍ **练习 5.7（坐标点类和圆形类）** 设计一个描述二维坐标系中坐标点的类 point，包含 2 个私有成员变量分别表示 x 轴、y 轴坐标，包括如下公有成员函数：

1. 构造函数：point(double i, double j)，其中 i 和 j 用于构造 x 轴和 y 轴坐标；

2. 成员函数：virtual double area() 返回点的面积：0。

从 point 类公有派生出圆形类 circle，新增一个私有成员变量：半径 double r，设计如下公有成员函数：

1. 构造函数：circle(double i, double j, double radius)，其中 i 和 j 用于构造圆心点（即基类 point 对象），radius 为半径；
2. 成员函数 area()，覆盖 point 类中的同名虚函数，用于计算圆形的面积；
3. 成员函数 point getTopPoint()，用于计算圆形的最高点坐标，即（x, y+r）。

编写主函数，定义以上两个类的对象，并对 area 函数等进行调用。要求使用赋值兼容规则，使基类 point 的指针指向基类对象和派生类对象进行 area 函数调用。

✍ **练习 5.8（形状类、圆形类和矩形类）** 设计一个描述基本形状的抽象类 BasicShape，包含一个 double 类型的私有成员变量 area，用于存储面积，包括如下公有成员函数：

1. getArea：返回 area 变量中的值。
2. calcArea：纯虚函数。

从 BasicShape 类公有派生出圆形类 circle，新增 3 个私有成员变量：圆中心的 x 坐标 int centerX，圆中心的 y 坐标 int centerY，半径 double r，设计如下公有成员函数：

1. 构造函数：接受初始化 centerX、centerY、r 的 3 个参数，并调用下面的 calcArea 函数计算圆的面积；
2. getCenterX：返回 centerX 的值；
3. getCenterY：返回 centerY 的值；
4. calcArea：计算圆的面积，并将结果存储在继承所得的 area 变量中。

从 BasicShape 类公有派生出矩形类 rectangle，新增 2 个私有成员变量：矩形的宽 long width，矩形的长 long length，设计如下公有成员函数：

1. 构造函数：接受初始化 width、length 的两个参数，并调用下面的 calcArea 函数计算矩形的面积；
2. getWidth：返回 width 的值；
3. getLength：返回 length 的值；
4. calcArea：计算矩形的面积，并将结果存储在继承所得的 area 变量中。

编写主函数，定义 circle 对象和 rectangle 对象。检验程序能否正确的计算各形状的面积。

✍ **练习 5.9（长方体类）** 从上一题的矩形类 rectangle 公有派生出长方体类 cube，新增一个私有成员变量：高度 long height，并设计 calcArea 成员函数，覆盖基类中的同名虚函数，用于计算长方体的表面积。编写主函数，定义 cube 类的对象，并对 calcArea 函数等进行调用。要求使用赋值兼容规则，使基类 BasicShape 的指针指向 cube 类对象进行 calcArea 函数调用。

5.6 多继承

多继承是指一个类同时继承于两个以上的类，从而同时具有多个父类的成员，基本原理与单继承相同，因此不再提供额外练习。

第 5 章作业

1. 语句类：设计一个描述程序中语句的类 Stmt，包含 2 个私有成员变量：unsigned line 存储语句在源文件中的行号，char *str 指向一个存储语句的内容的动态数组，请编写构造函数动态分配内存、析构函数释放分配的内存、如下公有成员函数：
   (a). getDetails：用 new 分配字符数组存储格式为“行号：line”的字符串并返回；
   (b). getAsString：虚函数，用 new 分配字符数组存储 str 指向的字符串并返回；
   (c). print：调用 getAsString 函数，将语句的内容输出到屏幕，然后 delete 字符数组。
   编写主函数，定义 Stmt 类的对象，调用 getDetails 和 print 函数输出各种语句的内容。
2. 二元表达式类：从上一题的 Stmt 类公有派生出描述程序中二元表达式的类 BinaryOperator，从父类中继承 str 存储操作符左边的字符串，新增 3 个私有成员变量：char op[10] 存储二元表达式中的操作符（如 >、<、==），char *str2 存储操作符右边的字符串，char type[10] 存储表达式的类型（如 bool、int、float），请编写构造函数动态分配内存、析构函数释放分配的内存、如下公有成员函数：
   (a). getDetails：覆盖基类同名函数，用 new 分配字符数组存储格式为“行号：line，类型：type”的字符串并返回；
   (b). getAsString：覆盖基类同名虚函数，用 new 分配字符数组存储格式为“str op str2”的字符串并返回。
   编写主函数，定义 BinaryOperator 类的对象，调用 getDetails 和 print 函数输出各种语句的内容。要求使用赋值兼容规则，使基类 Stmt 的指针指向 BinaryOperator 类对象进行函数调用。
3. 数组下标类：从上一题的 BinaryOperator 类公有派生出描述程序中数组下标表达式的类 ArraySubscriptExpr，请编写构造函数将 op 设置为空、如下公有成员函数：
   (a). getDetails：覆盖基类同名函数，用 new 分配字符数组存储格式为“数组下标表达式的行号：line，类型：type”的字符串并返回；
   (b). getAsString：覆盖基类同名虚函数，用 new 分配字符数组存储格式为“str[str2]”的字符串并返回。
   编写主函数，定义 ArraySubscriptExpr 类的对象，调用 getDetails 和 print 函数输出各种语句的内容。要求使用赋值兼容规则，使基类 Stmt 的指针指向 ArraySubscriptExpr 类对象进行函数调用。

第 6 章 异常处理

内容提要

- 抛出异常（throw）
- 处理异常（try-catch）

6.1 抛出异常（throw）

6.2 处理异常（try-catch）

✍ **练习 6.1（学生类异常）** 已知学生类 student 包含两个私有成员变量：int id、char *name 分别表示学号、姓名。请编写构造函数初始化成员变量、析构函数释放内存空间。在构造函数中用 new 动态分配姓名所需的内存空间，由于 new 分配内存失败后直接抛出 bad_alloc 类型的异常，因此需要用 try 语句尝试 new 操作，用 catch 语句捕获该异常，然后再抛出 BadNew 类型的异常；当传递给学号的值无效时 (小于 0 或大于 9999)，抛出 BadId 类型的异常。编写完整的程序演示如何抛出并处理上述异常。

第 6 章作业

1. 时间格式类异常：已知时间格式类 time 包含三个私有成员变量：int hours、int minutes、int seconds 分别表示小时、分钟、秒。请编写成员函数用于设置这三个成员变量的值，其中，当传递给小时的值无效时 (小于 0 或大于 23)，抛出 BadHours 类型的异常；当传递给分钟的值无效时 (小于 0 或大于 59)，抛出 BadMinutes 类型的异常；当传递给秒的值无效时 (小于 0 或大于 59)，抛出 BadSeconds 类型的异常。编写完整的程序演示如何抛出并处理上述异常。

第 7 章 模板

内容提要

- 函数模板
- 类模板

7.1 函数模板

练习 7.1 读程序，写输出结果。

```
template<class T>
T f(T *a, T *b, int n)
{
  T s = (T)0;
  for (int i = 0; i<n; i++)
    s += a[i] * b[i];
  return s;
}
int main() {
  double c[5] = { 1.1, 2.2, 3.3, 4.4, 5.5 }, d[5] = { 1.0, 0, 10.0, 100.0 };
  cout << f(c, d, 4) << endl;
  return 0;
}
```

7.2 类模板

练习 7.2（动态数组类模板） 设计一个参数为类型 T 的类模板，它具有一个 T* 类型的成员变量（用于存储动态数组的地址）和两个整型成员变量（用于存储动态数组的大小和有效元素的个数）。构造函数具有一个整型参数 count，它为指针成员变量分配 count 个元素的空间。析构函数释放指针成员变量指向的空间。另外，设计 5 个成员函数完成如下功能：

1. 向指针指向的空间末尾追加存储数据；
2. 删除指针指向的空间末尾的数据（变为非有效元素）；
3. 返回所有元素的总和；
4. 返回所有元素的平均值；
5. 在屏幕上显示所有元素。

编写一个完整的程序检验该模板对于浮点数类型和整数类型参数的正确性。

第 7 章作业

1. 数组求和：编写一个参数为类型 T、计算并返回数组元素总和的函数模板 sum，其中函数的参数是一个 T 类型的数组和它的长度。编写完整的程序，检验该模板能否对 int、float 等各种数据类型的数组进行求和。
2. 开销计算：编写一个参数为类型 T、计算个人开销的类模板 cost，它的成员变量包括：每天的餐饮开销 meal、每月的交通开销 transport、一年的学习开销 study，均为 T 类型。编写完整的程序，定义一个 T 为整数类型的对象和一个 T 为浮点数类型的对象，询问每个人的每项开销，然后显示一年（365 天）的总开销。程序需进行输入有效性检验：每个人每天的餐饮开销不能小于 20。
3. 具有排序功能的数组：从本章的数组类模板 FreewillArray 派生出具有排序功能的子类模板 SortableArray，该类模板包含一个成员函数 sort() 实现对数组元素的升序排列（任选排序算法）。编写一个完整的程序测试该模板。