CCF中学生计算机程序设计教材

CCF 中学生计算机程序设计

入门篇

中国计算机学会◎组编
陈 颖 邱桂香 朱全民◎主编

浅显的描述揭示深刻的内涵

信息学国际金牌教练
教你如何编程



科学出版社



CCF 中学生计算机程序设计教材

CCF 中学生计算机程序设计入门篇

中国计算机学会　组编

陈　颖　邱桂香　朱全民　主编

科学出版社

北　京

内 容 简 介

本书由 CCF 组织富有程序设计教学经验的中学老师编写。通过详实的例题，循序渐进地介绍中学生计算机程序设计的各种知识，内容包括数据的存储和读入、程序的选择执行、程序段的反复执行、数据的批量存储等，旨在普及计算机科学教育，培养中学生的计算思维能力。

本书可作为中学生计算机程序设计教材，也可供广大计算机编程爱好者参考。

图书在版编目（CIP）数据

CCF中学生计算机程序设计入门篇/中国计算机学会组编；陈颖，邱桂香，朱全民主编. — 北京：科学出版社，2016.11

（CCF中学生计算机程序设计教材）

ISBN 978-7-03-050021-2

Ⅰ. C… Ⅱ. ①中… ②陈… ③邱… ④朱… Ⅲ. ①程序设计 - 中学 - 教材 Ⅳ. ①G634.671

中国版本图书馆CIP数据核字（2016）第232487号

责任编辑：杨 凯 / 责任制作：魏 谨

责任印制：张 倩 / 封面制作：杨安安

北京东方科龙图文有限公司 制作

http://www.okbook.com.cn

科 学 出 版 社 出版

北京东黄城根北街16号

邮政编码：100717

http://www.sciencep.com

天津新科印刷有限公司 印刷

科学出版社发行 各地新华书店经销

*

2016年11月第 一 版 开本：720×1000 1/16

2016年11月第一次印刷 印张：17

印数：1 — 7 000 字数：336 000

定价：38.00元

（如有印装质量问题，我社负责调换）

CCF 中学生计算机程序设计教材
编委会名单

主　编　杜子德

主　审　吴文虎

副主编　王　宏　尹宝林　朱全民　陈　颖

编　委　（按姓氏笔画排序）

江　涛　汪星明　邱桂香　宋新波

屈运华　徐先友　廖晓刚

序

由中国计算机学会（CCF）组编的“CCF 中学生计算机程序设计教材”面世了。

早在 1984 年，邓小平就提出“计算机的普及要从娃娃抓起”。这很有先见之明，但这里的“计算机普及”是泛指，并未明确普及哪些内容。在过去的三十多年中，中小学广泛开展了计算机普及活动，2000 年教育部也曾发文，要在全国中小学开展信息技术教育。但事实上，现有的所谓“普及”大多成了对计算机工具的认识，而不是对中小学生智力的开发和思维的训练，因而效果不佳。CCF 早在 1984 年就创办了“青少年信息学奥林匹克竞赛 NOI”，这是面向那些学有余力的中学生的一项计算机科学（CS）教育活动，但具备开展这项活动的学校并不很多，每年参加 NOI 联赛的学生不过七八万，比例很小，因而普及的面并不大。

计算机科学教育的核心是算法设计和编程，这要求学生面对一个给定的现实问题要能够找到一个正确和高效的办法（算法）并将其变成计算机能理解的语言（程序设计语言），进而让计算机计算出人们需要的结果来。像快递员最佳路径算法就是一个典型的现实问题。这个过程并不容易，因为将一个问题抽象并构造一个模型，需要一定的数学基础，还得理解计算机的特点，“指挥”计算机干活。这还涉及欲求解问题的“可计算性”，因为并不是任何问题都可以由计算机求解的。计算机也并不知道什么是“问题”，是人告诉计算机，如何按照一步一步的程序求解。这个过程，就会训练一个人求解问题的能力，相应地，其具备的让计算机做事的思维能力称之为“计算思维”（Computational Thinking）。我们平常操作计算机（包括手机这些终端设备）仅仅像开关电灯那样简单，并不会使我们具备计算思维能力，而只有通过上述步骤才能训练这样的能力。随着计算机和网络的发展，未来越来越多的工作将和计算（机）有关（据美国政府的统计是 51% 以上）。我们必须知道如何让计算机做事，起码知道计算机是如何做事的，这就要求普及计算机科学教育（注意：不是计算机教育，也不是信息技术教育）。

美国政府已经把在中小学普及计算机科学当成一种国策（CS for All，每一个人学习计算机科学），并投入 40 亿美元落实这一项目。奥巴马总统说“在新经济形态中，计算机科学已不再是可选技能，而是同阅读、写作和算术一样的基础技能……因此，我制定了一项计划，以确保所有孩子都有机会学习计算机科学。”美国政府已明确把计算机教育列入（从幼儿园到 12 年级）教育体系 K12 中。英国从 2014 年起，对中小学的计算机课程进行重大改革，5 岁的儿童就开始学写程序。英国教育部启动了“计算机在学校”（Computing at School，CAS）项目。新西兰等国也把计算机编程课当作中学的必修课，并为此投入资金培训教师。未来的竞争不是资源的竞争，而是人才的竞争，如果不具有计算素养和技能，则在未来的社会中处于被动地位。

CCF 作为一个负责任的学术社会组织，应该勇于承担起 CS 普及的任务，这比 NOI 更加艰巨，更难。不过有 NOI 三十多年发展的基础，会对未来 CS 的普及提供有益的经验。

普及计算机科学教育的难点在于师资，而培训师资需要合适的教材。CCF 组织富有程序设计教学经验的中学老师编写了“CCF 中学生计算机程序设计教材”，分为入门篇、基础篇、提高篇和专业篇，只要有一定数学基础的老师，均可从入门篇修起。学习编程并不像人们想象的那么困难，只要从现实中遇到的（简单）问题出发，循序渐进，通过和计算机的互动，一旦入门就好办了，以后就可以逐步深入下去。

感谢朱全民、陈颖、徐先友、江涛、邱桂香、宋新波、汪星明、屈运华、廖晓刚等老师的贡献，他们花了两年时间写成了这套教材。感谢吴文虎教授、王宏博士审阅本书，在此向他们表示感谢。

杜子德

中国计算机学会秘书长

2016 年 8 月 29 日

目　录

第 1 章　编程如此简单

1.1　程序设计概念 ……1
1.2　整数算术运算 ……4
1.3　实数算术运算 ……8
附录 A　Dev-C++ 集成开发环境 ……11

第 2 章　数据的存储和读入

2.1　变量和变量的类型 ……17
2.2　赋值语句和数学表达式……21
*2.3　数据类型转换 ……32
2.4　变量的读入 ……38
*2.5　C 语言中的 scanf 语句和 printf 语句 ……43
2.6　顺序结构程序设计实例……51
附录 B　基本数据类型 ……56
附录 C　常用数学函数 ……57

第 3 章　程序的选择执行

3.1　if 语句和关系表达式 ……59
3.2　逻辑表达式和条件表达式……68
3.3　嵌套 if 语句 ……75
3.4　switch 语句 ……83
*3.5　分支结构程序设计实例……92

第 4 章　程序段的反复执行

4.1　for 语句 ……105
4.2　while 语句 ……119
4.3　do-while 语句 ……131
4.4　多重循环 ……142
*4.5　在循环结构中应用位运算……156
*4.6　循环结构程序设计实例……167

附录 D　break 语句和 continue 语句 …… 178

第 5 章　数据的批量存储

5.1　一维数组 …… 183
5.2　活用数组下标 …… 192
5.3　数值排序和查找 …… 204
5.4　字符数组 …… 220
5.5　二维数组 …… 233
*5.6　数组的综合应用实例 …… 245

参考文献 …… 261

索引 …… 263

第1章　编程如此简单

为什么计算机能管理庞大的系统？为什么人能指挥计算机按自己的想法做事？计算机能超越人的思维吗？面对计算机，很多人都会带着或多或少的疑问。其实，计算机是通过执行一个个程序来实现各种神奇的功能。人们要指挥计算机做事，需要编写程序，把自己的思想融入程序中，再利用这些程序按照自己的思路去指挥计算机工作。

1.1　程序设计概念

【例1.1】用记事本编写一个程序，让计算机告诉大家我爱编程，即“I love programming.”

程序如下：

```
1  //exam1.1
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      cout<<"I love programming."<<endl;
7      return 0;
8  }
```

编写完这个程序后，大家还会有些疑问，程序为什么要这样写？编程到底要遵循什么样的规则？这个程序编写后怎样让计算机去执行呢？为了回答这些问题，我们必须先弄清一些概念。

1.1.1　编程概念

1. 程序和计算机语言

程序就是为实现特定目标或解决特定问题而用计算机语言编写的一系列指令序列。

计算机语言是人与计算机之间通信的语言。计算机语言有很多种，比

如BASIC语言、PASCAL语言、C语言、C++语言、Java语言等，每种程序语言都有自身所定义的规则，即使是同一个语言也有不同的版本，内部规则也会有少许不同，比如C++语言就有Visual C++、Dev-C++等。

编程可以简单理解为程序员为解决特定问题，按照自己的思路，在遵循特定的计算机语言规则下编写程序的过程。因此，要学好编程，首先选择一种计算机语言，然后按照该程序语言规则编写程序。

本书以C++语言规则为例来学习编程。

2．编辑和编译

在某种计算机语言环境下编写程序的过程，叫程序的编辑。

程序编辑好后，计算机是不是就可以立即运行呢？其实不然。这个程序在没有被计算机识别之前，仅仅只是一些符号所组成的文本。就像我们读的书籍一样，如果不经过大脑去领会书籍内容所包含的含义，文字永远只能是文字，课本也只能是课本，但经过我们的大脑学习和领会了书籍文字中所包含的内涵和思想，那么看起来死板的文字就成为大脑所学到的知识。程序也一样，也需要计算机去学习和领会程序所包含的含义，这个学习和领悟的过程当然不完全类似人脑对知识的学习，而仅仅是按照计算机语言所对应的规则，对所编写的程序进行解析的过程，这个过程叫**翻译**。翻译分为两种方式，一种叫**解释**，是计算机对程序的指令翻译一句执行一句的行为，BASIC语言和Java语言都属于解释型语言；另一种叫**编译**，是计算机对程序的全部指令一次性全部翻译后，再让计算机执行的行为，PASCAL语言和C++语言都属于编译型语言。

高级语言程序要通过编译器才能运行得到结果，不同的计算机语言使用的编译器不同，如不能用C++编译器编辑运行BASIC语言编写的程序，同时，编译后的程序，在不同操作系统环境下进行的解析也会略有区别。

3．集成开发环境

知道了编辑和编译后，自然就需要一个用来编辑程序的软件和一个对程序进行编译的软件。编辑的软件有很多，所有的文本编辑器都可以，比如记事本。编译的软件只能是特指，比如C语言用的是GCC编译器，C++语言用的是G++编译器等。

一个程序编辑后，需要经过编译，方可执行。可万一编写程序时出错了，或者程序思路有问题，那么就要反复对程序进行编辑和编译，显得非常麻烦。为了方便程序的编辑和编译，软件公司开发了程序语言的集成环境，也就是将编辑和编译调试集成在一块，形成了**集成开发环境**

（Integrated Development Environment，IDE）。IDE对人们学习编程提供了极大的方便。

为了学习方便，本章将在附录A中简单介绍DEV C++的IDE使用方法。当然大家选择C++语言的其他IDE也可，使用方法也与Dev- C++的IDE类似。

1.1.2 程序结构

为了弄清C++的编程规则，首先我们看C++程序的基本结构，如图1.1所示。

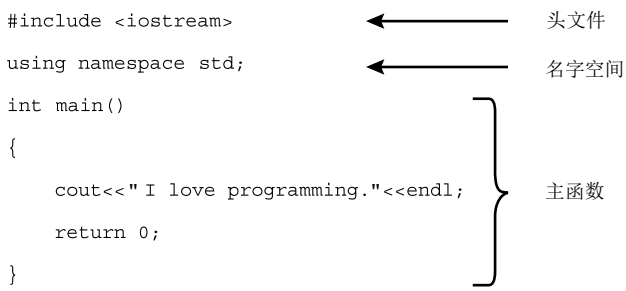


图1.1 C++语言基本结构

由图1.1可以看出，C++程序由头文件、名字空间和主函数组成。

1. 头文件

头文件是C++程序对其他程序的引用。头文件作为一种包含功能函数、数据接口声明的载体文件，用于保存程序的声明。include的英文含义是“包括”。格式为：#include <引用文件名>或#include “引用文件名”。

2. 名字空间

指明程序采用的名字空间。采用名字空间是为了在C++新标准中，解决多人同时编写大型程序时名字产生冲突问题。比如A、B两个班都有叫张三的人，你要使用A班的张三，必然要先指明是A班这个名字空间(namespace)，然后你对张三的所有命令才能达到你的预想，不会叫错人。

“using namespace std”表示这个程序采用的全部都是std（标准）名字空间，std是英文单词standard（标准）缩写。若不加这句，则该程序中cout和endl都需指明其名字空间的出处。cout语句必须写成

```
std::cout<<"I love programming."<<std::endl;
```

3. 主函数

日常生活中，我们要完成一件具有复杂功能的事，总是习惯把“大功能”分解为多个“小功能”来实现。在C++程序的世界里，“功能”可称为“函数”，因此“函数”其实就是一段实现了某种功能的代码，并且可以供其他代码调用。

一个程序，无论复杂或简单，总体上都是一个“函数”，这个函数称为“main函数”，也就是“主函数”。比如有个“做菜”程序，那么“做菜”这个过程就是“主函数”。在主函数中，根据情况，你可能还需要调用“买菜、切菜、炒菜”等子函数。main函数在程序中大多数是必须存在的，程序运行时都是找main函数来执行。

每个函数内的所有指令都需用花括号“{}”括起来。一般每个函数都需要有一个返回值，用return语句返回。

练　习

（1）阅读附录A中的 Dev-C++集成开发环境，尝试运行图1.1的程序。

（2）熟悉C++程序的基本结构，尝试改变程序，输出你想说的句子。

1.2　整数算术运算

【例1.2】把20张画平均分给7个同学，每人分得几张，还剩几张？

分析：每个人分得的张数是20除以7的商，剩余的张数是20除以7的余数。

程序如下：

```
1  //exam1.2
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      cout<<20/7<<endl;
7      cout<<20%7<<endl;
8      return 0;
9  }
```

运行结果：

```
2
6
```

对于这个解决问题的程序，大家的疑问会是什么呢？cout语句能做什么事？如何表达两个整数相除的商和余数？为了回答这些问题，我们需要学习cout语句的格式和使用，学习算术表达式的计算机表示方式。

1.2.1　cout语句

cout是C++的输出语句，C++的输出和输入是用“流”(stream)的方式实现的。

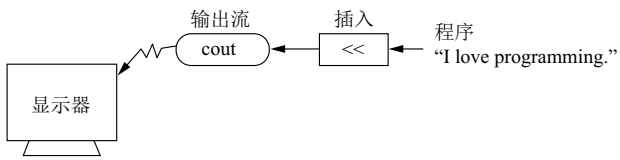


图1.2　C++的输出流

在定义流对象时，系统会在内存中开辟一段缓冲区，用来暂存输入输出流的数据。在执行cout语句时，先把数据存放在输出缓冲区中，直到输出缓冲区满或遇到cout语句中的endl或'\n'为止，此时将缓冲区中已有的数据一起输出，并清空缓冲区。输出流中的数据在系统默认的设备（一般为显示器）输出。输出遇到endl或'\n'换行。

cout语句的一般格式为：

cout<<项目1<<项目2<<…<<项目n;

功能：

（1）如果项目是表达式，则输出表达式的值。

（2）如果项目加引号，则输出引号内的内容。

1.2.2　算术运算符

在例1.2中运用cout语句输出算术表达式的值，C++语言为算术运算提供了5种基本算术运算符号：加（+）、减（−）、乘（×）、除（/）还有模（%）。如表1.1所示。

表 1.1　基本算术运算符

运算符	含　义	说　明	例　子
+	加法	加法运算	5+1=6
–	减法	减法运算	13–5=8
*	乘法	乘法运算	5*4=20
/	除法	两个整数相除的结果是整数，去掉小数部分	3/2=1
%	模	模运算的结果符号取决于被除数的符号	8%3=2

上述运算符的优先级与数学中相同，*、/、%高于＋、－。

表 1.1 中特别值得注意的是“/”号和“%”，对于“/”号，当参与运算的数含有实数，运算结果是两数相除的值，当参与运算的两个数都是整数，运算结果是两数相除的商，如：例 1.2 程序中第 6 行求 20/7 的商为 2；对于“%”是求两个整数相除的余数，如：例 1.2 程序中第 7 行求 20%7 的余数为 6。

【例 1.3】阅读下列程序和运行结果，学习表达式的书写格式，了解程序中表达式运算先后顺序和数学习惯的数学表达式运算先后顺序的关系。

```
1  //exam1.3
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      cout<<9/8<<4*(6+3)%5<<(4*6+3)%5<<endl;  //输出算式值
7      return 0;
8  }
```

运行结果：

```
112
```

说明：由于 9 和 8 是整数，因此，9/8 结果为 1；对于式子 4*(6+3)%5，先计算 4*(6+3) 的值为 36 然后 %5，结果为 1；对于式子 (4*6+3)%5，先计算 (4*6+3) 的值为 27 然后 %5，结果为 2。表达式的运算先后顺序与数学习惯相同。

【例 1.4】在例 1.3 中 3 个表达式的运行结果紧挨在一起，希望改变输出方式，要求：（1）每个表达式值隔开一个空格；（2）在结果前提示表达式。

分析：对于问题(1)，在输出表达式间输出一个空格。

对于问题（2），在输出表达式前输出加引号的表达式。

程序如下：

```
1  //exam1.4
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6       cout<<9/8<<4*(6+3)%5<<(4*6+3)%5<<endl;// 输出算式值
7       cout<<9/8<<" "<<4*(6+3)%5<<" "<<(4*6+3)%5<<endl;
                                          // 空格加引号，输出空格
8       cout<<"9/8="<<9/8<<"4*(6+3)%5="<<4*(6+3)%5<<"(4*6+
        3)%5="<<(4*6+3)%5<<endl;// 加引号的输出项，输出引号内的内容
9       return 0;
10 }
```

运行结果：

```
112
1  1  2
9/8=1 4*(6+3)%5=1 (4*6+3)%5=2
```

说明：在输出语句中，如果要输出项目本身的符号而不是表达式的值，书写格式是对输出项目加双引号。程序中第7行，要输出空格，对空格加双引号。程序中第8行，要输出表达式，对表达式加双引号。

实验：如果将程序中的endl删除，运行结果会怎样？

【例1.5】将8000秒表示成小时分钟秒的形式。

分析：

（1）1小时为3600秒，那么8000除以3600的商（8000/3600）即为小时。

（2）将转换小时后剩余的秒数，即8000除以3600的余数（8000%3600）转为分钟，1小时为60分钟，那么，（8000%3600）除以60的商（(8000%3600)/60）即为分钟。

（3）转换为小时和分钟后剩余的8000%3600%60即为秒数。

程序如下：

```
1  //exam1.5
2  #include<iostream>
```

```
3  using namespace std;
4  int main()
5  {
6      cout<<"8000秒=";
7      cout<<8000/3600<<"小时";
8      cout<<(8000%3600)/60<<"分钟";
9      cout<<8000%3600%60<<"秒"<<endl;
10     return 0;
11 }
```

运行结果：

```
8000秒=2小时13分钟20秒
```

思考：程序中运用“/”和“%”实现将秒转换为小时、分钟、秒表示，是否还有其他的运算表达方式？如果想出来了，请修改上述程序并实现之。

练　习

对于下列问题，写出数学解决的步骤，然后用程序描述数学解决过程，让计算机运行得到问题的解。

（1）有3台拖拉机3天耕地90公顷，照这样计算，5台拖拉机6天耕地多少公顷？

（2）有5辆汽车4次可以运送100吨钢材，如果用同样的7辆汽车运送105吨钢材，需要运几次？

（3）体育室里有58根跳绳，如果要平均分给8个班且无剩余，最少要去掉几根跳绳？每个班分到几根跳绳？

1.3　实数算术运算

【例1.6】4个工人3天铺了90m²地板砖，照这样计算，5个工人6天能铺多少平方米地板砖？

分析：

（1）求1个工人1天铺多少平方米地板砖：$90 \div 3 \div 4 = 7.5$（m²）。

（2）求5个工人6天铺多少平方米地板砖：$7.5 \times 5 \times 6 = 225$（m²）。

列成综合算式：$90 \div 3 \div 4 \times 5 \times 6 = 7.5 \times 30 = 225$（m²）。

程序如下：

```
1  //exam1.6
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      cout<<"5 个工人 6 天能铺 ";
7      cout<<90.0/3/4*5*6<<" 平方米地板砖。"<<endl;
8      return 0;
9  }
```

运行结果：

```
5个工人6天能铺225平方米地板砖。
```

对于这个解决问题的程序，大家的疑问会是什么呢？程序中 90 为什么要写成 90.0，如果 90 不写成 90.0 结果正确吗？为了回答这些问题，我们需要理解实数与整数的区别。

在日常生活中，我们对于 5 和 5.0 的区别并不是很在意的，但在计算机中，它们表示不同意义的数，一个是整数，另一个是实数。对于一个算术表达式，整数之间的运算结果为整数，实数之间的运算结果为实数。

C++ 中参与运算的数若存在实数，运算过程按实数运算，即如果希望运算过程按实数运算，表达式中至少有一个数表示为实数，如：例 1.6 虽然结果是整数，但程序中第 7 行相除运算需要按实数运算，所以需要将一个数表示为实数。

【例 1.7】分析下面的程序运行结果。

```
1  //exam1.7
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6    cout<<15*3/2<<endl;
7    cout<<15*3/2.0<<endl;
8    return 0;
9  }
```

运行结果：

```
22
22.5
```

说明：表达式15*3/2中参与运算的数均为整数，按整数方式运算，“/”代表求两数相除的商，其值为22。表达式15*3/2.0中数2.0为实数，按实数方式运算，“/”代表两数相除的结果，其值为22.5。

实验：将例1.6程序中的90.0改为90后运行程序，结果是多少？为什么？

【*例1.8】阅读下列程序和运行结果，理解实数运算和整数运算的区别，理解fixed <<setprecision(8)格式函数的作用。

```
1  //exam1.8
2  #include<iostream>
3  #include <iomanip>
4  using namespace std;
5  int main()
6  {
7      cout<<"9/8="<<9/8<<"9.0/8="<<9.0/8<<"9/8.0="<<9/
       8.0<<"9.0/8.0="<<9.0/8.0<<endl;
8      cout<<"10.0/6.0="<<10.0/6.0<<endl;
9      cout<<"10.0/6.0="<<fixed<<setprecision(8)<<10.0/6.0
         <<endl;
10     return 0;
11 }
```

运行结果：

```
9/8=1 9.0/8=1.125 9/8.0=1.125 9.0/8.0=1.125
10.0/6.0=1.66667
10.0/6.0=1.66666667
```

说明：程序中fixed<<setprecision(8)是格式函数，其作用是让其后面的输出值保留小数后8位，使用格式函数需要头文件#include <iomanip>。

实验：

（1）删除程序中#include<iomanip>头文件，编译运行程序，说明其作用。

（2）尝试改变表达式，如：10.0/6.0改为100.0/6.0，运行程序，观察结果的保留位数。

（3）尝试改变fixed<<setprecision(8)格式函数括号中的数字，运行程序，体验格式函数的作用。

练　习

对于下列问题，写出数学解决的步骤，然后用程序描述数学解决过程，让计算机运行得到问题的解。

（1）买5支铅笔要0.6元钱，买同样的铅笔16支，需要多少钱？

（2）服装厂原来做一套衣服用布3.2m，改进裁剪方法后，每套衣服用布2.8m。原来做791套衣服的布，现在可以做多少套？

（3）两块瓷砖，一个长方形，长10cm，宽8cm，另一个是正方形，长方形瓷砖面积大于正方形瓷砖面积16cm^2，正方形瓷砖边长为多少厘米？

附录A　Dev-C++集成开发环境

这里介绍Dev-C++5.2.0.3集成开发环境的简单使用方法，该IDE适合于Windows操作系统平台。

对于初学者，会希望IDE帮助我们做什么呢？

（1）创建程序文件。

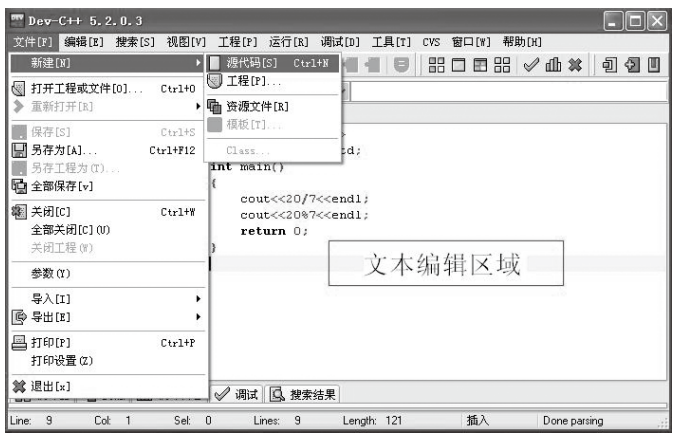
（2）编译和运行程序。

（3）如果程序有问题，借助调试手段帮助找到问题。

（4）设置个性化的界面。

A.1　新建、保存、打开程序文件

打开Dev-C++ 5.2.0.3，熟悉文件菜单，如图A.1所示。



图A.1

方法 1：

（1）打开“文件”菜单，选择“新建”下的“源代码”，在文本编辑区域输入和编辑程序。

（2）用“文件”菜单下的“保存”或“另存为”，保存程序。

（3）用“文件”菜单下的“打开工程或文件”打开程序文件。

方法 2：使用菜单中提示的快捷操作键实现相关的操作。

方法 3：使用菜单下方的快捷图标实现相关的操作。

A.2　编译、运行程序

程序运行前需要：

（1）存储程序。

（2）选择“运行”菜单下的“编译”，弹出编译窗口，如图 A.2 所示。如果程序语法有错，将显示错误位置；如果程序语法正确，完成编译。

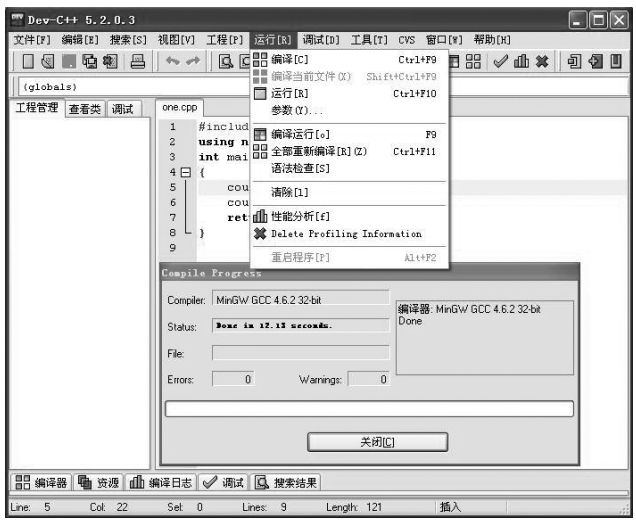


图 A.2

（3）完成编译后，选择“运行”菜单下的“运行”，如果输出是指向显示器，将弹出运行结果窗口，如图 A.3 所示。



图 A.3

（4）可以选择“运行”菜单下的“编译运行”。

A.3　调试程序

当发生程序运行结果不正确，又不知道在哪里发生错误时，可借助 IDE 的调试工具，对可能发生错误的程序段实施单步执行。点击图 A.4 界面下方“调试”按钮，弹出调试程序界面，通过单步执行过程中观察变量的变化，找到问题。

Dev-C++ 5.2.0.3 提供了很清晰的调试程序界面，如图 A.4 所示。

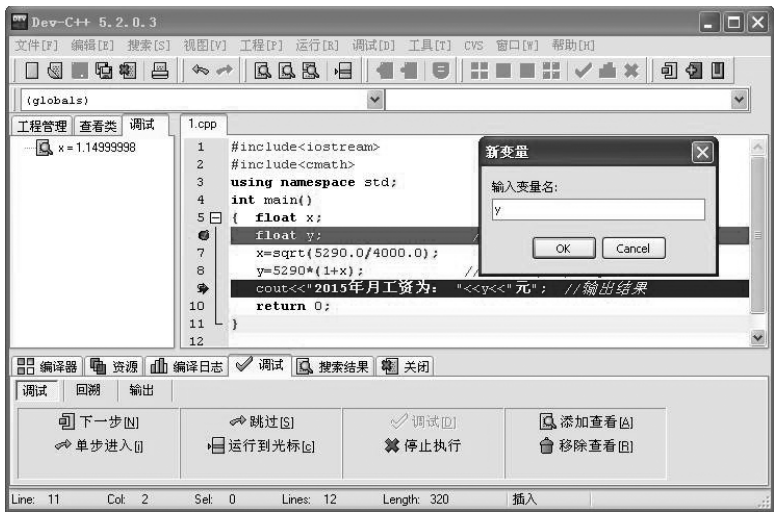


图 A.4

（1）设置断点：将光标移到程序想要暂停执行的一行，按 F4 键在该行设置断点，该行将变成红色，表示程序将运行到该行处暂停。

（2）观察变量：通过“添加查看”，在窗体左边添加希望观察的变量，

如图 A.4 中左窗和“新变量”对话框。

（3）调试程序：可以通过窗体下方“调试”选项中的不同按钮的操作帮助调试程序，也可以按 F7 执行当前行，并跳到下一行。Ctrl+F7 跳到下一个断点。Shift+F4 跳到光标所在行，并在该行设置断点。

A.4　设置自己喜欢的 IDE 界面

Dev-C++5.2.0.3 提供了许多选项的设置功能，可以让用户进行个性化界面设置。

选择“工具”菜单下的“编辑器选项”命令，弹出“编辑器属性”对话框，如图 A.5 所示。在“编辑器属性”对话框下，可以对不同选项进行设置，达到设置自己喜欢的 IDE 界面效果。

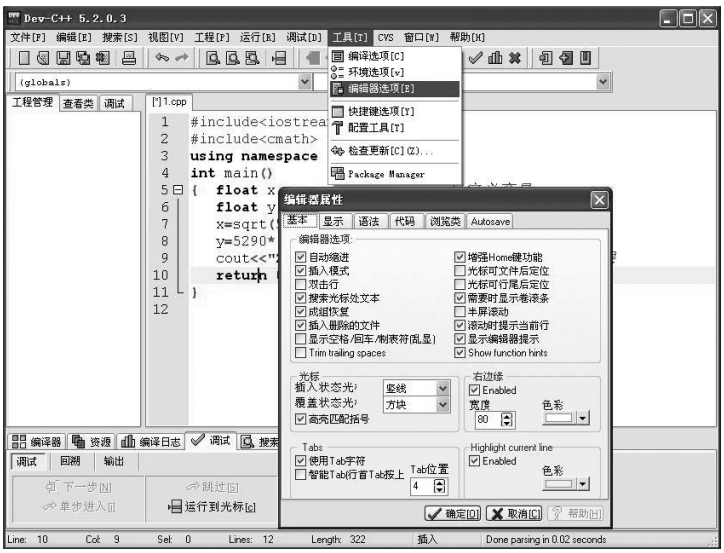


图 A.5

A.5　程序设计与调试建议

良好的程序设计行为与调试习惯，有助于培养学习兴趣和深入学习。

（1）透彻分析问题，给出能够实施的具体解决问题的方法和步骤，即设计完整的算法和数据结构，写程序过程即用语言规定的语句描述解决问题的方法和步骤。不可只有一个大致的想法就匆忙开始写程序。正如写一篇文章，关键是文章的构思，有了具体的构思，可以中文、英文等语言进行描述，如果只有一个大致的想法写出来的文章一定不是好文章。对程序

而言，一点点的错误导致的结果可能是很严重的。

（2）分析问题过程中，给出问题可能存在的各种数据状态和结果，如边界数据等，帮助有效得到正确的解决方案，测试程序的正确性。

（3）编译运行程序前，先进行静态查错，所谓**静态查错**，即认真阅读一遍所写的程序，检查是否正确表达所设计的算法、数据结构、程序模块。特别关注细节表达，如：变量名、数据类型、数据边界、变量初值、数据传递等。

（4）编译运行程序，先利用能够预见的可能存在的各种数据状态和结果测试程序，再设计大数据测试程序。

（5）调试程序尽量不依赖调试工具。

第2章　数据的存储和读入

大家有没有想过计算机是如何存储和管理数据这个问题？存储器是计算机存放数据的空间。生活中大多数的空间需要通过规划或分配或申请等手续后才能使用，计算机存储空间也是需要通过类似的规划、分配和申请方能使用。空间的使用要遵从一定的规则，如：我们到宾馆入住，首先要按需预定房间的类型，如果预定1间双人间，那么最多只能入住两人，否则就违规。同理，数据入住内存中，如何才能合理入住呢？首先需要对数据进行分类，依据不同类型数据申请能够存放相应大小的空间。程序设计的一个重要环节是合理分配数据存储类型和存储空间。

2.1　变量和变量的类型

【例2.1】将整数65存储到计算机内存中，并且输出。

程序如下：

```
1  //exam2.1
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      int a;              //定义整型变量a
7      a=65;               //将整数65存入a中
8      cout<<a<<endl;      //输出a的值
9      return 0;
10 }
```

运行结果：

```
65
```

从运行程序结果我们可以看到，数据存储到a中，并且输出了a的值。那么a代表什么？程序中int a的作用是什么？为了回答这些问题，我们先学习变量和变量的类型。

2.1.1　变量和变量类型概念

在各学科的学习中，当求解一个问题时，对于数据我们并没有想得太多，写在纸上，爱怎么写就怎么写。然而，当把数据存储到计算机中时，计算机需要硬件实现数据的存放，这个硬件就是计算机的内存储器（简称内存）。那么，应该将数据存放到内存的什么位置呢？计算机高级语言中通常用变量名标识数据放在存储器的位置，同时需要指明给变量名所在位置开辟多大的空间。那么，应该依据什么开辟空间的大小呢？我们自然会想应该依据放入变量中数据可能出现的大小，为了能够规范地开辟空间，高级语言把数据进行了分类，称之为数据类型，在使用变量前，需要定义变量的数据类型，系统依据定义的数据类型，给变量开辟对应大小的存储空间来存放数据。

例 2.1 程序中第 6 行 int a 表示在内存中开辟一个变量名叫做 a，数据类型为整型的空间，该整型空间占用 4 字节，允许存放在 a 中的数据为 –2147483648~2147483647 范围内的整数。

【例 2.2】阅读下列程序的运行结果，说一说变量 a 的作用。

程序如下：

```
1  //exam2.2
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6       int a;              //定义整型变量 a
7       a=65;               //将整数 65 存入 a 中
8       cout<<a<<endl;      //输出 a 的值
9       a=100;              //将整数 100 存入 a 中
10      cout<<a<<endl;      //输出 a 的值
11      return 0;
12 }
```

运行结果：

```
65
100
```

说明：当新的数据存入变量空间时，变量的值改变为新值，这是存储器的特点——喜新厌旧。程序第 7 行 a 值为 65，第 9 行把 100 存入 a 后，a

值即为100。

2.1.2 变量名

变量是个多义词，在计算机语言中变量表示某个存储数据空间的名称，因此，命名时要遵守一定的规则。

C++语言变量命名规则如下：

（1）变量名中只能出现字母（A~Z,a~z）、数字(0~9)或下划线。

（2）第一个字符不能是数字，例如2Server不是一个合法的C++变量。

（3）不能是C++关键字。所谓关键字，即C++中已经定义好的有特殊含义的单词。

（4）区分大小写，例如1A和1a是两个不同的变量。

为了便于阅读，变量的命名最好用有含义的英文单词或英文单词组合。变量名不宜太长，太长容易写错，一般长度控制在15个字符之内。

【例2.3】以下合法的变量名是：

（A）int （B）_10days （C）my_book （D）us$D.count

分析：正确答案是（C）。合法的变量名应该是字母开头，允许含有下划线，但下划线不能作为开头，所以（B）是错误的。int是关键字不可以作变量名，所以（A）是错误的。（D）中包含非法字符，所以是错误的。

2.1.3 变量的定义

变量定义的作用是，在内存开辟一个类型标识符指定类型的空间，用变量名标识。

C++语言中，数据存入变量前，首先要定义变量。

变量的定义格式如下：

类型标识符 变量名1，变量名2，…，变量名n;

【例2.4】将实数65.5存储到计算机内存变量a中，并且输出。

程序如下：

```
1  //exam2.4
2  #include <iostream>
3  using namespace std;
4  int main()
```

```
5  {
6       float a;              //定义浮点型变量
7       a=65.5;               //将实数 65.5 存入 a 中
8       cout<<a<<endl;        //输出 a 的值
9       return 0;
10 }
```

运行结果：

```
65.5
```

说明： 程序中 float a 表示在内存中开辟一个变量名叫做 a，数据类型为浮点型的空间，该浮点型的空间占用 4 字节，允许存放在 a 中的数据为 –3.4E+38~3.4E+38(7 位有效数字) 范围内的实数。

【例 2.5】将字符“A”存储到计算机内存变量 a 中，并且输出。

程序如下：

```
1  //exam2.5
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6       char a;               //定义字符型变量
7       a='A';                //将字符 A 存入 a 中
8       cout<<a<<endl;        //输出 a 的值
9       return 0;
10 }
```

运行结果：

```
A
```

说明： 程序中 char a 表示在内存中开辟一个变量名叫做 a，数据类型为字符型的空间。该字符型的空间占用 1 字节，允许存放在 a 中的数据是编码为 –128~127 范围内对应的字符，但一般情况，我们更多使用字符类型存放键盘字符。

【例 2.6】求长为 7.5cm、宽为 10.6cm 的矩形面积，要求先将矩形长和宽数据分别存储到变量 x、y 中。

程序如下：

```
1  //exam2.6
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      float x=7.5;          //定义浮点型变量并给 x 赋初值
7      float y=10.6;         //定义浮点型变量并给变量 y 赋初值
8      cout<<"Area of a rectangle:"<<x*y<<endl;
                             //求面积并输出
9      return 0;
10 }
```

运行结果：

```
Area of a rectangle:79.5
```

说明：变量定义的同时，可以给变量赋一个初始值。

在例2.2、例2.4、例2.5中，我们认识了整型、浮点型、字符型的三种数据类型，更多的数据类型参见附录B。

变量定义的两个关键要素是数据类型和变量名，其选择来自问题，要依据问题中的数据的性质选择合适的类型和变量名。

练　习

（1）下列变量名中哪些是合法的，哪些是不合法的，请说明原因。

3zh ant _3cq my friend Mycar my_car all 55a a_abc while daf-32 x.13 Var(3) maxn max&min

（2）模仿例2.1、例2.3、例2.4，查阅附录B，使用更多的数据类型，将你想的数据存入某些变量中并且输出。

（3）模仿例2.5，寻找学习中类似的问题，尝试用程序实现。

2.2 赋值语句和数学表达式

【例2.7】求半径为7cm的圆面积。

程序如下：

```
1  //exam2.7-1
```

```
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6       float radius;      //定义存放半径变量为浮点型
7       float area;        //定义存放面积变量为浮点型
8       radius=7;          //半径值为 7
9       area=3.14159265*radius*radius;        //求圆面积
10      cout<<"Circular area ="<<area<<endl;
                           //输出面积
11      return 0;
12 }
```

运行结果：

```
Circular area =153.938
```

程序中第 8 行和第 9 行的表达方式大家非常熟悉，可能会说这不是数学等式吗？式子的表现形式是等式，但不能用熟悉的等式概念去理解，这样的语句称之为赋值语句，那么在程序中应该如何理解赋值语句呢？C++ 语言中的赋值语句还有怎样的表达方式？在程序第 9 行，我们自然而然地用了算术运算符表示数学表达式，那么，如何正确地将数学表达式转换为程序能接受的表达式？为了回答这些问题，我们将学习 C++ 语言的赋值语句和数学表达式。

2.2.1　赋值语句

赋值语句的格式为：

变量　赋值运算符　表达式

赋值语句的意思是，将运算的结果放到变量中存储起来。

例 2.7 的第 8 行表示将 7 存入变量 radius 中，第 9 行表示通过运算器先计算右边表达式的值，然后将运算结果存入变量 area 中。

赋值运算符用于对变量进行赋值，分为简单赋值 (=)、复合算术赋值 (+=、-=、*=、/=、%=) 和复合位运算赋值 (&=、|=、^=、>>=、<<=) 三类共 11 种。

【例 2.8】阅读下列程序，理解赋值语句。

```
1  //exam2.8
```

```
#include<iostream>
using namespace std;
int main()
{
    int a=5;          //定义变量并赋初值
    cout<<a<<endl;    //输出a的值
    a=a+2;            //让a值加2
    cout<<a<<endl;    //输出a的值
    a=a+5;            //让a值加5
    cout<<a<<endl;    //输出a的值
    return 0;
}
```

运行结果：

```
5
7
12
```

说明：程序中第8行和第10行，如果理解成数学等式显然是不成立的，然而赋值的概念是将右边式子值存入左边变量中，由于a的初值是5，a+2的值是7，因此，运行程序第8行后，a值为7。对于程序第10行，a+5的值为12，运行后a值为12。

【例2.9】阅读下列程序，说一说复合算术赋值的功能。

```
//exam2.9
#include<iostream>
using namespace std;
int main()
{
    int a,b;          //定义变量
    a=b=3;            //a、b值赋为3
    a+=b;             //相当于a=a+b
    cout<<a<<endl;    //输出a值
    cout<<b<<endl;    //输出b值
    return 0;
}
```

运行结果：

```
6
3
```

说明：C++ 语言支持连等号赋值的表达形式，程序中 a=b=3 表示将 a 和 b 的值赋为 3。程序中 a+=b 等效于 a=a+b，但是前者执行速度比后者快，表示先计算 a+b 的值为 6，然后赋值给 a。其他 -=、*=、/=、%= 的用法和 += 类似。

关于复合位运算赋值，将在第 4 章中学习。

【例 2.10】编程实现两个变量 x、y 之间值的交换。

方法 1：生活中，如何实现两个杯子中的东西交换呢？很自然想到借助一个空杯。那么，可以用同样的方法实现两个变量 x、y 之间值的交换，引入一个中间变量 t，把 x、y、t 看成 3 个杯子，t 是个空杯子，现要把 x、y 两杯子的东西交换，怎么做呢？首先 x 倒入 t 中，t 装了 x 的内容，然后 y 倒入 x 中，这样 x 装了 y 的内容，最后 t 倒入 y 中，这样 y 装了 x 的内容，实现了交换。

值得注意的是赋值语句是将右边值赋给左边变量，即 x 倒入 t，用赋值语句表达写成 t=x。

程序如下：

```
1  //exam2.10-1
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      float x,y,t;                  //定义变量
7      x=10.5;                       //给 x 变量赋一个值
8      y=30.6;                       //给 y 变量赋一个值
9      cout<<x<<""<<y<<endl;         //输出 x、y 的值
10     t=x;x=y;y=t;                  //交换 x、y 的值
11     cout<<x<<""<<y<<endl;         //输出交换后 x、y 的值
12     return 0;
13 }
```

运行结果：

```
10.5   30.6
30.6   10.5
```

说明：程序第 10 行实现变量内容交换的方法在解决问题中经常用到。

方法2：利用内存变量只有存入新值时才会改变旧值性质,实现变量交换。

（1）先将x+y值放入x中，则x值为两数之和，y为原值。

（2）将x-y值放入y中，则x值还是为两数之和，y为x的原值。

（3）接着再求一次x-y放入x中，则x为y的原值，y为x的原值，实现两数交换。

程序如下：

```
1  //exam2.10-2
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      float x,y;
7      x=10.5;
8      y=30.6;
9      cout<<x<<""<<y<<endl;
10     x+=y;y=x-y;x-=y;          // 交换x、y的值
11     cout<<x<<""<<y<<endl;
12     return 0;
13 }
```

运行结果：

```
10.5   30.6
30.6   10.5
```

思考： 程序中第10行利用赋值语句性质通过运算实现变量交换，还有其他求两数互换的方法吗？请用程序实现你的方法。

2.2.2 变量的自增和自减

C++语言中，整型或浮点型变量的值加1可以使用自增运算符“++”。有两种用法：

用法1： 变量名++；

用法2： ++变量名；

这两种用法都能使变量的值加1，但它们是有区别的。

【例2.11】 阅读程序和程序运行结果，说一说变量自增两种用法的共同点和它们的区别。

```
1  //exam2.11
```

```
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int n1,n2=5;          //定义变量，并给 n2 变量赋初值
7      n2++;                                    //自增 n2
8      cout<<"n2="<<n2<<endl;                   //输出 n2 的值
9      ++n2;                                    //自增 n2
10     cout<<"n2="<<n2<<endl;                   //输出 n2 的值
11     n1=n2++;                                 //n1 值为自增 n2 的值
12     cout<<"n1="<<n1<<" n2="<<n2<<endl;       //输出 n1、n2 的值
13     n1=++n2;                                 //n1 值为自增 n2 的值
14     cout<<"n1="<<n1<<" n2="<<n2<<endl;       //输出 n1、n2 的值
15     return 0;
16 }
```

运行结果：

```
n2=6
n2=7
n1=7 n2=8
n1=9 n2=9
```

说明：从程序运行结果可以看出，单独使用自增，如程序中第 7、9 行，n2++ 和 ++n2，两种用法结果都一样，使变量 n2 的值加 1。当在赋值语句中用时，两种用法结果就不同了，如程序中第 11 行，n1=n2++，先将 n2 值赋给 n1，然后 n2 再加 1，执行后 n1 和 n2 值不同，而程序中第 13 行，n1=++n2，则 n2 先加 1 后赋给 n1，执行后 n1 和 n2 值相同。

整型或浮点型变量的值减 1 可以使用自减运算符“--”，其用法与自增类似。

2.2.3　程序中的数学表达式

数学表达式由数据、变量、运算符、数学函数、括号组成，程序中的数学表达式需要用语言能够接受的运算符和数学函数表示。

【例 2.12】已知 a=5.5、b=6.7、c=9.3，编程求式子 $\frac{-b+4ac}{2a}$ 的值。

程序如下：

```
1  //exam2.12
2  #include<iostream>
```

```
3  using namespace std;
4  int main()
5  {
6      float a,b,c,f ;              //定义存放半径变量为整型
7      a=5.5;                       //给变量a赋值
8      b=6.7;                       //给变量b赋值
9      c=9.3;                       //给变量c赋值
10     f=(-b+4*a*c)/(2*a);          //求f的值
11     cout<<f<<endl;               //输出f的值
12     return 0;
13 }
```

运行结果：

```
17.9909
```

说明：为了保证运算顺序的正确，有时需要适当加入括号。程序第10行的数学表达式中，在分子和分母部分各加了括号保证了数学式子运算顺序的正确性。

【例2.13】为了激励员工稳定工作，小计所在的公司每年都在元月一次性提高员工的当年的月工资。小计2012年的月工资为4000元，在2014年时他的月工资增加到5290元，他在2015年的月工资按2012到2014年的月工资的平均增长率继续增长，那么小计2015年的月工资是多少？

分析：设小计2012到2014年的月工资的平均增长率为x，那么

$$4000(1+x)^2 = 5290$$

$$x = \sqrt{5290/4000} - 1$$

设小计2015年的月工资y，那么

$$y = 5290(1+x)$$

程序如下：

```
1  //exam2.13
2  #include<iostream>
3  #include<cmath>
4  using namespace std;
5  int main()
6  {
7      float x;                                  //定义变量
8      float y;                                  //定义变量
```

```
9       x=sqrt(5290.0/4000.0)-1;            //求平均增长率
10      y=5290*(1+x);          //求 2015 年的月工资
11      cout<<"2015 年月工资为："<<y<<" 元 ";   //输出结果
12      return 0;
13 }
```

运行结果：

```
2015年月工资为： 6083.5元
```

说明：cmath 是 C++ 数学函数库，一些数学计算公式的具体实现放在 cmath 里，常用的数学函数表见附录 C。程序第 9 行用到了开平方根函数，函数的书写按常用数学函数表中提供的方式书写。当程序中使用到数学函数时，头文件需要加 #include<cmath>。

实验：删除上述程序中的头文件 #include<cmath>，编译运行程序看到什么错误？说明头文件 #include<cmath> 的作用。

【* 例 2.14】一列火车在某地时的速度为 v_0=40km/h，现以加速度 a=0.15m/s^2 行驶，求 2min 后的速度 v 和距开始点的距离 s。

分析：根据物理知识：

$$v_t = v_0 + at$$

$$S = v_0 t + at^2/2$$

问题中已知的 v_0、a 和 t 的单位不一致，在运用公式求解过程中，需要先将单位变换统一，即：

$$v_0 = 40\text{km/h} = 40 * 1000/3600\text{m/s}$$

$$t = 2\text{min} = 2 * 60\text{s}$$

在现实中，速度、距离一般为实数，时间以秒为单位一般为整数，因此，问题中的变量定义，速度、距离选择 float 类型比较合适，时间可以选择 int 类型。

程序如下：

```
1  //exam2.14
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      float v0=(40/3.6),a=0.15; //定义变量并赋初值
7      int t=120;                //定义变量并赋初值
```

```
8       float vt,s;                                  //定义变量
9       vt=v0+a*t;                                   //求2min后的速度
10      s=v0*t+0.5*a*t*t;                            //求2min后的距离
11      cout<<"vt="<<vt<<" s="<<s<<endl;  //输出结果
12      return 0;
13 }
```

运行结果：

```
vt=29.1111   s=2413.33
```

说明：对于程序中的第9、10行，由于平时数学书写习惯，初学者很容易忘记乘号，在程序中式子中的乘法运算一定要用“*”表示，不可忽略。

2.2.4 常量定义

在例2.7求圆的面积中，圆周率π是一个常数，对于常数可以定义一个常量来存储。所谓**常量**，即常量的值在程序中不能发生变化。

定义或说明一个常量格式如下：

<类型说明符> const <常量名>

如：int const X=2;

或者

const <类型说明符><常量名>

如：const int X=2;

π用常量定义后，例2.7的程序可改写如下：

```
1  //exam2.7-2
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6       const float PI=3.14159265;  //定义常量PI存放π的值
7       float radius;               //定义存放半径变量为浮点型
8       float area;                 //定义存放面积变量为浮点型
9       radius=7;                                  //半径值为7
10      area=PI*radius*radius;                     //求圆面积
11      cout<<"Circular area ="<<area<<endl; //输出面积
12      return 0;
13 }
```

说明：在程序中使用常量具有以下优势：

（1）修改方便。无论程序中出现多少次定义的常量，只要在定义语句中对定义的常量值进行一次修改，就可以全改。

（2）可读性强。常量通常具有明确的含义，如上述程序中定义的PI，一看到PI就会想到它代表的是圆周率。

（3）为了区别常量与变量，通常程序中常量名用大写字母表示。

练　习

（1）下列属于合法的表达式的是（　　）。

（A）a+b*|c|　（B）bct÷9　（C）pi*r*r　（D）$\alpha+\beta*\gamma$

（2）把下列数学式子写成计算机可以识别的数学表达式。

（A）$y=mx+b$　（B）$m=\dfrac{a+b+c}{e*f}$　（C）$a=\sqrt{(x-3y)z}$

（D）$a=\dfrac{2x-y}{x+y^2}$　（E）$m=\dfrac{x-yz}{\dfrac{2}{c}}$

（3）下列赋值语句正确的有哪些？

（A）$-m=1$　（B）$m=4n$　（C）$m=2*m$

（D）$a+2=b-3$　（E）$xx=y*y$　（F）$a=b2$

（4）假定下面每个表达式中整型变量x的值均为10（假设各表达式互不影响），求x和y的值。

表达式	值	表达式	值
++x		x++	
--x		x--	
y=x++		y=5*x++	
y=--x		y=x--*2+3	

（5）写出下列程序的运行结果

```
//test(5)-1
#include<iostream>
using namespace std;
int main()
{
   int a=3,b=3,c=3;
```

```
    cout<<++a<<""<<b++<<""<<c+1<<endl;
    cout<<a<<""<<b<<""<<c<<endl;
    return 0;
}
```

输出：

```
//test(5)-2
#include<iostream>
using namespace std;
main()
{
    int x,y,z;
    x=y=1;
    z=x++-1;
    cout<<x<<""<<z<<endl;
}
```

输出：

（6）查阅附录C，阅读下列程序写出运行结果后运行程序验证，分析ceil函数、floor函数、pow函数、sqrt函数的作用以及ceil函数和floor函数的区别。

```
//test(6)
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
      cout<<"ceil(3.14)="<<ceil(3.14)<<" floor(3.14)="<<floor(3.14)<<endl;
      cout<<"4^3.0="<<pow(4,3.0)<<endl;
      cout<<"sqrt(9)="<<sqrt(9)<<endl;
      return 0;
}
```

（7）540棵树苗分给五、六年级同学去种，五年级有120人，六年级有150人，如果按照人数进行分配，每个年级各应分得多少棵树苗？用程序实现。

（8）模仿例2.14给出一个学习中用学到的知识能够解决的问题，编程实现，与同学们分享。

*2.3　数据类型转换

【例2.15】已知三角形的底为23，高为51，求三角形的面积。

程序如下：

```
1  //exam2.15
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6       int a,h;              //定义变量
7       float s;              //定义变量
8       a=23;
9       h=51;
10      s=a*h/2.0;            //求三角形面积
11      cout<<s<<endl;        //输出面积
12      return 0;
13 }
```

运行结果：

```
586.5
```

对于程序中的第10行，认真分析我们还会发现赋值语句的“=”右边的变量a和h是整型，而“=”左边的变量s是浮点型，在运算和赋值过程中遇到不一样的数据类型时，程序语言按什么规则实现转换呢？为了回答这些问题，我们将学习C++语言中的数据类型转换。

数据类型转换就是将数据（变量、表达式的结果）从一种类型转换到另一种类型。例如，为了保存小数可以将int类型的变量转换为float类型。数据类型转换有自动类型转换，也可以强制类型转换。

2.3.1　自动类型转换

在不同数据类型的混合运算中，编译器会隐式地进行数据类型转换，称为**自动类型转换**。

自动类型转换遵循下面的规则：

（1）若参与运算的数据类型不同，则先转换成同一类型，然后进行运算。

（2）转换按数据长度增加的方向进行，以保证精度不降低。例如int类型和long类型运算时，先把int类型转成long类型后再进行运算。

即当参加算术或比较运算的两个操作数类型不统一时，将简单类型向复杂类型转换。

```
char(short)-->int(long)-->float-->double
```

（3）在赋值运算中，赋值号两边的数据类型不相同时，将把右边表达式值的类型转换为左边变量的类型。如果右边表达式值的数据类型长度比左边长时，将丢失一部分数据。

（4）在赋值语句中，赋值号两边数据类型一定是相兼容的类型，如果等号两边数据类型不兼容，语句在编译时会报错。

在例2.15程序中的第10行，表达式a*h/2.0中，a、h是整型，而2.0是浮点型，运算过程中类型自动转换为浮点型。

2.3.2 强制类型转换

当自动类型转换不能实现目的时，可以显式进行类型转换，称为**强制类型转换**。

强制类型转换的一般形式为：

（类型名）（表达式）

（类型名）变量

如：(double)a是将a转换成double类型，(int)(x+y)是将x+y的值转换成整型，(float)(5%3)是将5%3的值转换成float型。

需注意的是，无论是强制转换还是自动转换，都只是为了本次运算的需要而对变量的数据长度进行的临时性转换，不改变数据说明时对该变量定义的类型。

【例2.16】求三个整数的和。

分析：设a,b,c分别存放三个整数，其和存放在s中，4个变量设为int类型。

程序1：

```
1  //exam2.16-1
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int a,b,c,s;
7      a=1562345672;    // 给 a 变量赋一个 int 范围内的整数
```

```
8       b=1456789343;    // 给 b 变量赋一个 int 范围内的整数
9       c=1234567832;    // 给 c 变量赋一个 int 范围内的整数
10      s=a+b+c;         // 求和
11      cout<<"s="<<s<<endl;
12      return 0;
13 }
```

运行结果：

```
s=-41264449
```

实验：运行后发现结果是错的，为什么呢？程序中a、b、c三个数本身没有超过int类型规定的数据范围，但当它们相加时其结果超过了int类型规定的数据范围，那么将存放和的s变量的类型改为long long是否可以呢？

程序2：

```
1  //exam2.16-2
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6       int a,b,c;;
7       long long s;          //s 定义为长整型
8       a=1562345672;
9       b=1456789343;
10      c=1234567832;
11      s=a+b+c;
12      cout<<"s="<<s<<endl;
13      return 0;
14 }
```

运行结果：

```
s=-41264449
```

说明：运行后发现结果还是错的，为什么呢？因为a、b、c三个数都是int类型，因此，运算结果类型还是int类型，赋值语句左边的变量类型不影响运算结果。解决方法是将其中的一个变量如a变量强制转换为long long类型，让运算过程按长整型进行运算。

程序3：

```
1  //exam2.16-3
2  #include<iostream>
```

```
3  using namespace std;
4  int main()
5  {
6      int a,b,c;
7      long long s;
8      a=1562345672;
9      b=1456789343;
10     c=1234567832;
11     s=(long long)a+b+c;              //强制类型转换
12     cout<<"s="<<s<<endl;
13     return 0;
14 }
```

运行结果：

```
s=4253702847
```

说明：表达式运算过程的类型变化导致结果出错是程序设计中容易犯的错误，但是由于这种错误在程序编译时不报错，同时要在数据比较大的情况下才发生，不容易被发现，因此，数据类型变换是程序设计中需要注意的细节。

思考：

（1）将程序3中b、c变量改为short类型，运行程序结果正确吗？为什么？

（2）如果将问题改为求程序中给定的3个数平均值，程序的变量类型如何设计呢？

2.3.3　字符型和整型的转换

将一个字符存放到内存单元时，实际上并不是把该字符本身放到内存单元中去，而是将该字符相应的ASCII代码放到存储单元中。如果字符变量c1的值为'a'，c2的值为'b'，则在变量中存放的是'a'的ASCII码97，'b'的ASCII码98，如图2.1（a）所示，实际上在内存中是以二进制形式存放的，如图2.1（b）所示。

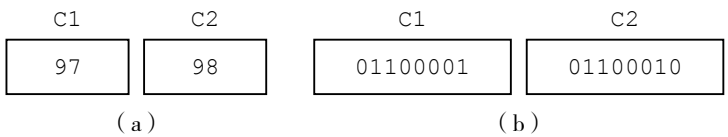


图2.1　内存的存放示意图

既然字符数据是以ASCII码存储的，它的存储形式就与整数的存储形式类似。这样，在C++中字符型数据和整型数据之间就可以通用。一个字符数据可以赋给一个整型变量，反之，一个整型数据也可以赋给一个字符变量。也可以对字符数据进行算术运算，此时相当于对它们的ASCII码进行算术运算。

【例2.17】将字符赋给整型变量。

程序如下：

```
1  //exam2.17
2  #include <iostream>
3  using namespace std;
4  int main( )
5    {
6      int i,j;   //i 和 j 是整型变量
7      i='A';     // 将一个字符常量赋给整型变量 i
8      j='B';     // 将一个字符常量赋给整型变量 j
9      cout<<i<<''<<j<<'\n';
                  // 输出整型变量 i 和 j 的值，'\n' 是换行符
10     return 0;
11 }
```

运行结果：

```
65 66
```

说明： i和j为整型变量，在程序第7和第8行中，将字符'A'和'B'分别赋给i和j，即把'A'和'B'字符的ASCII编码赋给i和j变量。

【例2.18】通过字符数据与整数进行算术运算将小写字母转换为大写字母。

```
1  //exam2.18
2  #include <iostream>
3  using namespace std;
4  int main( )
5  {
6      char c1,c2;                  // 定义字符型变量
7      c1='a';                      //c1 值为字符 a 的 ASCII 码
8      c2='b';                      //c2 值为字符 b 的 ASCII 码
```

```
9       c1=c1-32;                   //将c1的编码值-32
10      c2=c2-32;                   //将c2的编码值-32
11      cout<<c1<<' '<<c2<<endl;    //输出转换编码后的字符
12      return 0;
13 }
```

运行结果：

```
A B
```

说明：'a'的ASCII码为97，而'A'的ASCII码为65，'b'为98，'B'为66。从ASCII代码表中可以看到每一个小写字母比它相应的大写字母的ASCII代码大32。C++允许字符数据与数值直接进行算术运算，'a'-32得到整数65，'b'-32得到整数66。将65和66存放在c1、c2中，由于c1、c2是字符变量，因此用cout输出c1，c2时，得到字符'A'和'B'。

练 习

（1）写出下列程序的运行结果并上机运行验证，解释程序中变量数据类型的转换。

```
//test(1)-1
#include<iostream>
using namespace std;
int main()
{
   short a,b;
   double c;
   a=b=3;
   cout<<a<<""<<b<<endl;
   b=b+5;
   cout<<a<<""<<b<<endl;
   a+=b;
   cout<<a<<""<<b<<endl;
   c=a/b;
   cout<<"c="<<c<<endl;
   c=(double)a/b;
   cout<<"c="<<c<<endl;
   return 0;
```

```
}

//test(1)-2
#include<iostream>
using namespace std;
int main()
{
   int m,n,num;
   int t=48;
   char th;
   double dou_1,dou_2,dou_3;
   m=5;n=326;
   num=t/((float)m/n);
   dou_1=(double)(n/m);
   dou_2=n/m;
   dou_3=(double)n/m;
   th=(double)n/m;
   cout<<num<<","<<dou_1<<","<<dou_2<<","<<dou_3<<","<<th<<endl;
   return 0;
}
```

（2）神龙数码公司设计了一个加密算法：用a代替z，用b代替y，用c代替x，……，用z代替a。现要求输入一个字符，对其进行加密输出。

（3）模仿例2.18，设计并实现字符变换。

2.4　变量的读入

【例2.19】从键盘上输入一个三位数，然后将它反向输出。例如输入673，输出应为376。

分析：设x为输入的三位数，y为x的反向输出。先求出x的百位、十位、个位数：

$$x1 = x/100$$

$$x2 = (x - x1 * 100)/10$$

$$x3 = x\%10$$

则：

$$y = x3 * 100 + x2 * 10 + x1$$

程序如下：

```
1  //exam2.19
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int x,y,x1,x2,x3;
7      cin>>x;                  //读入一个三位数存入x变量中
8      x1=x/100;                //求百位数
9      x2=(x-x1*100)/10;        //求十位数
10     x3=x%10;                 //求个位数
11     y=x3*100+x2*10+x1;       //逆序组成新数
12     cout<<y<<endl;           //输出
13     return 0;
14 }
```

运行结果：

```
673
376
752
257
```

在此之前学习的程序对应问题运行的结果是唯一的，而这个解决问题的程序使用了cin语句，能够依据输入不同的三位数得到相应的结果。那么cin语句能做什么事？如何表达和使用？为了回答这些问题，我们需要学习cin语句的格式和使用。

2.4.1 cin语句格式

cin是C++的输入语句，与cout语句一样，C++的输入是用“流”(stream)的方式实现的。图2.2表示C++通过流进行输入的过程。

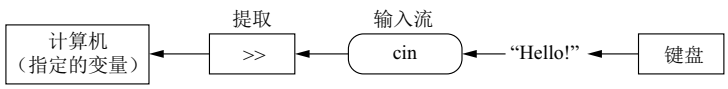


图2.2 cin的输入流程

为了叙述方便，常常把由cin和流提取运算符“>>”实现输入的语句称为输入语句或cin语句。

cin语句的一般格式为：

cin>>变量1>>变量2>>……>>变量n;

输入流中的数据通过系统默认的设备（一般为键盘）读入赋给变量。

与cout类似，一个cin语句可以分写成若干行，如

```
cin>>a>>b>>c>>d;
```

也可以写成

```
cin>>a;
cin>>b;
cin>>c;
cin>>d;
```

以上书写变量值均可以从键盘输入：1 2 3 4↙

也可以分多行输入数据：

```
1↙
2 3↙
4↙
```

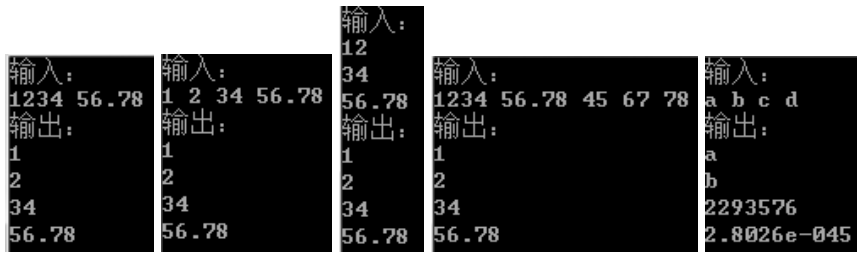
2.4.2　cin语句的使用

在用cin输入时，系统会根据变量的类型从输入流中提取相应长度的字节。

【例2.20】依据五组输入数据和运行结果分析cin数据读入方式。

```
1  //exam2.20
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      char c1,c2;                  //定义字符型变量
7      int a;
8      float b;
9      cout<<"输入："<<endl;        //提示输入
10     cin>>c1>>c2>>a>>b;           //读入数据
11     cout<<"输出："<<endl;        //输出变量的值
12     cout<<c1<<endl;
13     cout<<c2<<endl;
14     cout<<a<<endl;
15     cout<<b<<endl;
16     return 0;
17 }
```

运行结果：



（1）　（2）　（3）　（4）　（5）

说明：变量c1和c2是char类型，分别接受一个键盘字符，因此，对于第（1）组输入数据中的第1个数据1234，1赋给了c1，2赋给了c2，剩下的34赋给了a变量，第2个数据56.78赋给了b变量，得到第（1）组的输出结果。第（2）组输入数据间加了空格，将数据分别赋给了符合长度类型的四个变量，得到与第（1）组相同的输出结果。第（3）组数据间加了回车符，得到与第（1）组相同的输出结果。第（4）组输入数据个数超过变量个数，多余数据程序自动忽略，也得到与第（1）组相同的输出结果。第（5）组数据头两字符分别存入字符类型变量中，输出正确的结果，而后两字符存入到整型和实型变量中，输出意想不到的结果（不同编译器输出不同的结果）。

对应不同的输入方式，分析程序的运行结果，可以得出如下结论：

（1）cin语句把空格字符和回车换行符作为分隔符，不输入给变量。如果想将空格字符或回车换行符(或任何其他键盘上的字符)输入给字符变量，可以用后面学习的getchar函数。

（2）cin语句忽略多余的输入数据。

（3）在组织输入流数据时，要仔细分析cin语句中变量的类型，按照相应的格式输入，否则容易出错。

实验：对例2.20中的程序，以不一样的方式输入数据，设想运行结果，然后，运行程序，比对设想的运行结果和程序运行结果，总结读入语句的数据输入方式。

【例2.21】某两城市间公路用A~Z单字符来标识路段，输入某路段的路程(km)和汽车平均速度(km/h)，每公升汽油可以维持的公里数，以及每公升汽油价格(元)，求汽车经过该路段所花费的时间和费用。

输入样例：

A
200　80　6　15.6

输出样例：

A
time =2.5
totalcost=520

分析：设路段路程为d，汽车平均速度为s，每公升汽油可以维持的公里数为k，每公升汽油价格为cost。则：

（1）汽车运行时间time=d/s。

（2）所用汽油立升数l=d/k。

（3）旅行总共花费totalcost=l*cost。

问题中需要输入和输出路段标识符，因此，设一个字符类型的变量guidepost存储路标。

程序如下：

```
1  //exam2.21
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      char guidepost;           //定义路标变量
7      double d,s,k,cost;        //定义输入变量
8      double time,l,totalcost; //定义求值变量
9      cin>>guidepost;           //输入路标
10     cin>>d>>s>>k>>cost;       //输入路程、速度、公里数、油价
11     time=d/s;                 //汽车运行时间
12     l=d/k;                    //所用汽油立升数
13     totalcost= l*cost;        //求花费
14     cout<<guidepost<<endl;    //输出路标
15     cout<<"time="<<time<<endl;            //输出时间
16     cout<<"totalcost="<<totalcost<<endl; //输出费用
17     return 0;
18 }
```

运行结果：

```
A
200 100 8 5.2
A
time=2
totalcost=130
```

```
H
600 110 8 5.3
H
time=5.45455
totalcost=397.5
```

说明：程序中第9、10行利用cin语句实现对变量的输入，给问题的解决带来灵活和方便。

实验：能否对cin变量输入一个算式？

练 习

（1）将输入的华氏温度转换为摄氏温度。

*（2）输入三角形三边长a、b、c（保证能构成三角形），输出三角形面积（已知三角形的三条边求面积公式，请大家从网上查询）。

（3）有一个“就是它”的猜数游戏，步骤如下：请你任意输入的一个三位数x，在这三位数后重复一遍，得到一个六位数，467→467467，把这个数连续除以7、11、13，最后的商y就是你输入的三位数。请加以验证。

*2.5 C语言中的scanf语句和printf语句

C++语言兼容C语言中的基本语句语法，scanf语句和printf语句是C语言中的输入输出语句，在C++语言环境中亦可使用。对于大数据的输入输出，使用scanf语句和printf语句比C++的输入输出流cin和cout效率高、速度快。

scanf和printf分别称为格式输入函数和格式输出函数，其关键字最末一个字母f即为“格式”(format)之意，其意义是按指定的格式输入输出值。

scanf和printf是标准库函数，对于不同数据类型变量和数据的输入与输出，有严格对应的配对格式，使用前需要在头文件部分使用#include <cstdio>。

2.5.1 printf格式输出函数

printf函数调用的一般形式为：

printf(“格式控制字符串”，输出列表)

其中格式控制字符串用于指定输出格式，可由格式字符串和非格式字符串两种组成。格式字符串是以%开头的字符串，在%后面跟有各种格式字符，以说明输出数据的类型、形式、长度、小数位数等，如：

- “%d”表示按十进制整型输出。
- “%ld”表示按十进制长整型输出。
- 如果“”内为非格式字符串，则原样输出，在显示中起提示作用。

格式字符串和各输出项在数量和类型上应该一一对应。printf 函数的格式符见表 2.1。

表 2.1　printf 函数的格式符

格式字符	意　义
d	以十进制形式输出带符号整数（正数不输出符号）
o	以八进制形式输出无符号整数（不输出前缀 0）
x，X	以十六进制形式输出无符号整数（不输出前缀 0x）
u	以十进制形式输出无符号整数
f，lf	以小数形式输出单、双精度实数
e，E	以指数形式输出单、双精度实数
g，G	以 %f 或 %e 中较短的输出宽度输出单、双精度实数
c	输出单个字符
s	输出字符串

【例 2.22】阅读下列程序和程序运行结果，说一说“%d”格式控制字符串和输出列表的表达方式。

```
1  //exam2.22
2  #include<cstdio>
3  using namespace std;
4  int main()
5  {
6      printf("%d%d%d\n",9/8,4*(6+3)%5,(4*6+3)%5);
7      printf("%d  %d  %d\n",9/8,4*(6+3)%5,(4*6+3)%5);
8      printf("9/8=%d 4*(6+3)%5=%d (4*6+3)%5=%d\n",
             9/8,4*(6+3)%5,(4*6+3)%5);
9      printf("%d %d %d\n",41%6,41%(-6),(-41)%6);
10     return 0;
11 }
```

运行结果：

```
112
1  1  2
9/8=1  4*(6+3)=1  (4*6+3)=2
5 5 -5
```

实验：

（1）删除例2.22程序printf中的\n，编译运行程序，说明\n的作用。

（2）将例2.22程序printf(“%d %d %d\n”,9/8,4*(6+3)%5, (4*6+3)%5)中的空格改成逗号，编译运行程序，说明printf函数中双引号内的书写格式对输出显示内容的影响。

【例2.23】阅读下列程序和程序运行结果，说一说“%f”格式控制字符串和输出列表的表达方式。

```
1  //exam2.23
2  #include<cstdio>
3  using namespace std;
4  int main()
5  {
6      printf("9/8=%d 9.0/8=%f 9/8.0=%f 9.0/8.0=%f \n",
              9/8,9.0/8,9/8.0,9.0/8.0);
7      printf("10.0/6.0=%f\n",10.0/6.0);
8      printf("10.0/6.0=%.3f\n",10.0/6.0);
9      printf("10.0/6.0=%9.3f\n",10.0/6.0);
10     return 0;
11 }
```

运行结果：

```
9/8=1 9.0/8=1.125000 9/8.0=1.125000 9.0/8.0=1.125000
10.0/6.0=1.666667
10.0/6.0=1.667
10.0/6.0=    1.667
```

实验：

（1）将例2.23程序中的f符号变换为d符号，编译运行程序，说明数据输出时如何正确使用两符号。

（2）改变例2.23程序printf(“10.0/6.0=%9.3f\n”,10.0/6.0)中%与f间的数据，说明其对数据输出格式的影响。

【例2.24】阅读下列程序和程序运行结果，理解“%c”格式控制字符串和输出列表的表达方式。

```
1  //exam2.24
2  #include<cstdio>
3  using namespace std;
```

```
4  int main()
5  {
6       int a=88,b=89;
7       printf("%d %d\n",a,b);
8       printf("%d,%d\n",a,b);
9       printf("%c,%c\n",a,b);
10      printf("a=%d,b=%d",a,b);
11      return 0;
12 }
```

运行结果：

```
88 89
88,89
X,Y
a=88,b=89
```

2.5.2　scanf格式输入函数

scanf函数调用的一般形式为：

scanf(“格式控制字符串”，地址表列)；

其中，格式控制字符串的作用与printf函数相同，但不能显示非格式字符串，也就是不能显示提示字符串。地址表列中给出各变量的地址。地址是由地址运算符“&”后跟变量名组成的。

例如：&a、&b分别表示变量a和变量b的地址。

变量的地址和变量值的关系可以这样理解，在赋值表达式中给变量赋值，如：

a=567;

则，a为变量名，567是变量的值，&a是变量a的地址。

scanf函数在本质上也是给变量赋值，但要求写变量的地址，如：&a。这两者在形式上是不同的，&是一个取地址运算符，&a是一个表达式，其功能是求变量的地址。

表2.2　scanf函数的格式符

格式符	说　明
d,i	用于输入十进制整数
u	以无符号十进制形式输入十进制整数
o（字母）	用于输入八进制整数

续表

格式符	说 明
x	用于输入十六进制整数
c	用于输入单个字符
s	用于输入字符串（非空格开始，空格结束，字符串变量以‘\0’结尾）
f	用于输入实数（小数或指数均可）
e	与f相同（可与f互换）

表2.3 附加格式说明符

附加格式	说 明
l（字母）	l用于double型实数（%lf,%le）
域宽（一个整数）	指定输入所占列宽
*	表示对应输入量不赋给一个变量

【例2.25】阅读下列程序和程序运行结果，理解“%d”格式控制字符串和输入列表变量关系。

```
1  //exam2.25
2  #include<cstdio>
3  using namespace std;
4  int main()
5  {
6      int a,b,c;
7      printf("input a,b,c\n");
8      scanf("%d%d%d",&a,&b,&c);
9      printf("a=%d,b=%d,c=%d",a,b,c);
10     return 0;
11 }
```

运行结果：

```
input a,b,c
4 5 6
a=4,b=5,c=6
```

```
input a,b,c
4
5
6
a=4,b=5,c=6
```

程序中，由于scanf函数本身不能显示提示串，故先用printf语句在屏幕上输出提示，请用户输入a、b、c的值。执行scanf语句，等待用户输

入。在scanf语句的格式串中由于没有非格式字符在两个“%d”之间作输入时的间隔，因此在输入时要用一个以上的空格或回车键作为每两个输入数之间的间隔。如：

```
7  8  9
```

或

```
7
8
9
```

实验：

（1）将程序中的scanf语句改为scanf("%4d%2d%3d",&a,&b,&c)；输入：123456 123456，程序运行结果是什么？

（2）将程序中的scanf语句改为scanf("a=%d,b=%d,c=%d",&a,&b,&c)；输入：3 4 5，程序运行结果是什么？输入：a=3,b=4,c=5，程序运行结果是什么？

【例2.26】阅读下列程序和程序运行结果，理解不同“格式控制字符串”、输入列表变量和输入方式的关系。

```
1  //exam2.26
2  #include<cstdio>
3  using namespace std;
4  int main()
5  {
6      int a;
7      double b;
8      char c;
9      scanf("%c%d,%lf",&c,&a,&b);
10     printf("结果是：\n");
11     printf("%c %d %.2lf",c,a,b);
12     return 0;
13 }
```

运行结果：

```
x 5,34.5
结果是：
x 5 34.50
```

实验：

（1）输入x 5 34.5，程序运行结果是什么？

（2）输入x

5

34.5

程序运行结果是什么？

【例2.27】阅读下列程序和程序运行结果，分析“*”格式符的作用。

```
1  //exam2.27
2  #include<cstdio>
3  using namespace std;
4  int main()
5  {
6      int a,b;
7      scanf("%d%*d%d",&a,&b);
8      printf("a=%d,b=%d\n",a,b);
9      return 0;
10 }
```

输入：1 2 3回车

输出：a=1,b=3

scanf函数一些注意事项：

（1）scanf函数中没有精度控制，如：“scanf("%5.2f",&a);”是非法的。

（2）scanf中要求给出变量地址，如给出变量名则会出错，如“scanf("%d",a);”是非法的，“scanf("%d",&a);”才是合法的。

（3）在输入多个数值数据时，若格式控制串中没有非格式字符作输入数据之间的间隔，则可用空格、TAB或回车作间隔。C编译在碰到空格、TAB、回车或非法数据（如对“%d”输入“12A”时，A即为非法数据）时即认为该数据结束。

（4）在输入字符数据时，若格式控制串中无非格式字符，则认为所有输入的字符均为有效字符。

（5）如果格式控制串中有非格式字符，则输入时也要输入该非格式字符。

（6）如输入的数据与输出的类型不一致，虽然编译能够通过，但结果将不正确。

实验：

对例2.22~例2.27的程序，删除“using namespace std;”句子，程序是否可以编译并运行出正确的结果？为什么？

练　习

（1）阅读下列程序，给出结果。

```
//test(1)-1
#include<cstdio>
using namespace std;
int main()
{
     int a=202;
     double b=2323.34345;
     printf("a=%d\n",a);
     printf("2*a=%d\n",2*a);
     printf("a=%2d\n",a);
     printf("%3lf\n",b);
     printf("%20.2lf\n",b);
     printf("%-20.2lf\n",b);
     printf("%.2lf\n",b);
     return 0;
}
```

运行结果：

```
//test(1)-2
#include<cstdio>
using namespace std;
int main()
{
     int  i=1;
     int  j=123;
     printf("%d,%2d,%03d,%1d,%-4d,
     %05d",i,i,i,j,j,j);
     return 0;
}
```

运行结果：

2.6 顺序结构程序设计实例

【例2.28】鸡兔同笼，共有35个头，94条脚，求鸡和兔子各有多少只。

分析：设所求的鸡数是x只，兔子数是y只，已知笼子里的头数是a，脚数是b，依题意，得到如下的方程组：

$$\begin{cases} x + y = a \\ 2x + 4y = b \end{cases}$$

解方程组得：$x = 2a - b/2$，$y = b/2 - a$

程序如下：

```
1  //exam2.28
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int  a,b,x,y;                              //定义变量
7      a=35;b=94;                                 //变量赋初值
8      x=2*a-b/2;                                 //求鸡的只数
9      y=b/2-a;                                   //求兔子的只数
10     cout<<"x="<<x<<"  y="<<y<<endl;            //输出结果
11     return 0;
12 }
```

运行结果：

```
x=23   y=12
```

说明：程序中第7行给变量赋初值，第8行求鸡的只数，第9行求兔子的只数，最后输出结果。这种按顺序从上至下求解过程，称为顺序结构程序设计。

思考：还有其他求鸡兔只数的方法吗？请用程序实现你的方法。

【例2.29】公交车公司要统计公交车从始发站到末站所花费的时间。已知公交车于a时b分从始发站出发，并于当天的c时d分到达终点站（以上表述均为二十四小时制）。公交车从始发站到终点站共花了e小时f分钟（0<=f<60），要求输出e和f的值。

输入样例：

12 5 13 19

输出样例：

公交车从首站到末站共用了1小时14分钟

分析：公交车路途花费的时间=到达时间－出发时间，问题已知的是小时和分钟，需要将时间统一转化为分钟实现相减运算，再将相减的结果转换为分钟，得到如下解决问题的步骤，简称算法。

算法描述如下：

（1）输入a、b、c、d。

（2）求路途花费多少分钟时间timepast=60*c+d-(60*a+b)。

（3）将timepast转化为小时和分钟。

（4）输出结果。

程序如下：

```
1  //exam2.29
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6       int a,b,c,d,e,f,timepast;        //定义变量
7       cin>>a>>b>>c>>d;                 //读入数据
8       timepast=60*c+d-(60*a+b);        //求路途花费多少分钟时间
9       e=timepast/60;                   //将 timepast 转化为小时
10      f=timepast%60;                   //将 timepast 转化为分钟
11      cout<<" 公交车从首站到末站共用了 "<<e<<" 小时 "<<f<<" 分钟 "
            <<endl;                      //输出
12      return 0;
13 }
```

运行结果：

```
12 5 13 19
公交车从首站到末站共用了1小时14分钟
12 58 14 5
公交车从首站到末站共用了1小时7分钟
```

思考：本问题有否其他解决的方法？如果有，写出相应的算法，尝试用程序实现。

【例2.30】计算机随机生成一道加减混合运算题，输出题目和运算结果。参加运算的数据为1~1000的随机整数。

分析：问题的关键是如何生成随机数。C++语言生成随机数的方法：

（1）使用rand()函数返回从[0,MAX)之间的随机整数，这里的MAX由所定义的数据类型而定，需要在头文件处使用#include<cstdlib>。

（2）使用srand(time(NULL))或srand(time(0))设置当前的系统时间值为随机数种子，由于系统时间是变化的，那么种子也是变化的。需要在头文件处使用#include<cstdlib>和#include<ctime>。

随机数种子的作用是使rand()函数每次生成随机数据，如果不用随机数种子或用固定数随机种子，rand()函数每次生成相同随机数据。

产生一定范围随机数的通用表示公式：

- 要取得[a,b)的随机整数，使用(rand()%(b-a))+a。
- 要取得[a,b]的随机整数，使用(rand()%(b-a+1))+a。
- 要取得(a,b]的随机整数，使用(rand()%(b-a))+a+1。
- 通用公式：a+rand()%n。

其中的a是起始值，n是整数的范围。

要取得a到b之间的随机整数，另一种表示

```
a+(int)b*rand()/(RAND_MAX+1)
```

要取得0～1之间的浮点数，可以使用

```
rand()/double(RAND_MAX)
```

程序如下：

```
1  //exam2.30
2  #include <iostream>
3  #include <ctime>
4  #include <cstdlib>
5  using namespace std;
6  int main()
7  {
8      int x,y,z;
9      srand(time(0));          //随机种子
10     x=rand()%1000+1;         //产生随机数
11     y=rand()%1000+1;
12     z=rand()%1000+1;
13     cout<<x<<"+"<<y<<"-"<<z<<"="<<x+y-z<<endl;
                                //输出随机式子
14     return 0;
15 }
```

运行结果：

```
975+860-376=1459
168+656-786=38
```

实验：模仿上述程序设计产生多个不同范围的随机整数，将他们构建成一个数学表达式输出，并输出数学表达式的结果。

【*例2.31】输入四个正整数a、b、c、n（a、b、c均小于200，n<=6），求$a^n+b^n+c^n$。

输入样例：

34　56　7 5

输出样例：

S=596184007

分析：本问题的算法比较简单：

（1）输入a、b、c、n。

（2）求$a^n+b^n+c^n$的值。

（3）输出结果。

问题的关键是如何求a^n、b^n、c^n，查附录C常用数学函数可以知道利用pow(x，y)函数求指数方值，由于pow(x，y)是数学函数，因此，需要在头文件处使用#include<cmath>。

程序如下：

```
1  //exam2.31
2  #include<iostream>
3  #include<cmath>
4  #include <iomanip>
5  using namespace std;
6  int main()
7  {
8      double a,b,c,n;                          //定义变量
9      double s;
10     cin>>a>>b>>c>>n;                         //读入数据
11     s=pow(a,n)+pow(b,n)+pow(c,n);            //求a^n、b^n、c^n
12     cout<<setprecision(15)<<"s="<<s<<endl;      //输出
13     return 0;
14 }
```

运行结果：

```
34 56 7 5
s=596184007
```

实验：

（1）删除头文件#include<iomanip>，运行程序会出现什么问题，说明该头文件的作用。

（2）删除程序第12行中的“setprecision(15)<<”，运行程序，查看结果显示形式，说明其作用。

【*例2.32】给定整数等差数列的首项a和末项b以及项数n，求等差数列各项的总和（0<=a,b<=10^9,n<=200）。

输入样例：

5 10005 5

输出样例：

25025

分析：本问题需要数学数列知识，利用等差数列求和公式，得到如下的算法：

（1）输入a、b、n。

（2）利用等差数列求和公式（a+b）*n/2求数列和sum。

（3）输出结果。

在程序实现过程中需要特别注意问题中给出的输入数据范围。按照题意，a、b、n可以设置成int类型，然而数列和sum可能超出int范围，应设置成long long类型。对于公式(a+b)*n/2，如果a、b、n为int类型，则运算结果为int类型，可能发生错误，因此，需强制转换。

程序如下：

```
1  //exam2.32
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int a,b,n;
7      long long sum;                    //定义长整型
8      cin>>a>>b>>n;                     //读入数据
9      sum=((long long)a+b)*n/2;         //强制转换数据类型求值
10     cout<<"等差数列的和为"<<sum<<endl;      //输出
11     return 0;
```

```
12 }
```

运行结果：

```
5 10005 5
等差数列的和为25025
```

说明：程序第 9 行利用强制类型转换保证运算结果的正确性。在大数据的运算过程需要特别注意数据类型的变化。

练　习

（1）已知矩形的大小为 n × m，现用 a × a 的正方形填充该矩形。输入三个整数 n,m,a（n,m,a ≤ 10^9），求至多能填入多少正方形？（正方形可以正好碰到矩形边界，但不能超出矩形外）

（2）按年利率 R% 存入本钱 X，存入 P 年后的本利合计为 Y=X（（100+R）/100）^{P}，求 Y 的值。

（3）憨厚的老农夫昨天捡到了 3 块小石头，他想再去捡一个小石头，让这 4 个石头正好一共重 20 斤，请问他应该去捡一个多少斤的石头？

输入格式：

三个整数 a,b,c，是这三个石头的重量（斤）。

输出格式：

一个数，表示农夫应该去捡一个多少斤的石头。

输入样例：

3　5　7

输出样例：

5

（4）计算两个双精度浮点数 a 和 b 相除的余数，a 和 b 都是正数。这里余数（r）的定义是：a=k*b+r，其中 k 是整数，0<=r<b。

（5）设计算术题目形式，利用随机函数编程实现题目的输出，与同学们分享。

附录 B　基本数据类型

C++ 语言提供了丰富多彩的数据类型。常用的**基本数据类型**如表 B.1

所示。

表B.1　常用的基本数据类型

数据类型	类型标识符	所占字节数	取值范围
短整型	short[int]	2	–32768~32767
无符号短整型	unsigned short [int]	2	0~65535
整型	int	4	–2147483648~2147483647
无符号整型	unsigned int	4	0~4294967295
长整型	long long	8	-2^{63}~$2^{63}-1$
无符号长整型	unsigned long long	8	0~$2^{64}-1$
单精度浮点数	float	4	–3.4E+38~3.4E+38(7位有效数字)
双精度浮点数	double	8	–1.79E+308~1.79E308(15位有效数字)
高精度浮点数	long double	12	3.4E–4932~1.1E+4932(19位有效数字)
字符型	char	1	–128~127
	signed char	1	0~255
布尔型	bool	1	0或1

表中int、double、short、char、unsigned int等标识符都是类型名，C++中的类型名可以由用户定义，这将会在后面进一步学习。

表中“所占字节数”表示存储器分配给对应类型的空间大小，“取值范围”对该类型数据的取值范围进行了规定，如：short类型，其数据值只能是在–32768~32767范围中，若在运算过程中超出了对应数据类型的数值范围，会造成数据的溢出（overflow）错误。请注意，数据的溢出在编译和运行时并不报错，经常会让编程者不知道在哪儿发生错误。编程者需要特别细心和认真对待数据类型。

表中3.40E+38为科学计数法表示形式，为3.40×10^{38}。

类型的所占字节数可以用sizeof函数来测试，比如sizeof(long long)=8。

附录C　常用数学函数

C++中一些常用的数学函数如表C.1所示。

表 C.1　常用数学函数

绝对值、取整	
int abs(int i)	返回整型参数 i 的绝对值
double fabs(double x)	返回双精度参数 x 的绝对值
long labs(long n)	返回长整型参数 n 的绝对值
double ceil (double)	取上整
double floor (double)	取下整
指数、对数、开方	
double log(double x)	返回自然对数 ln(x) 的值
double log10(double x)	返回常用对数 lg(x) 值
double pow(double x,double y)	返回 x 的 y 次幂
double pow10(int p)	返回 10 的 p 次幂
double sqrt(double x)	返回 x 的平方根
三角函数	
double acos(double x)	求 x 的反余弦函数的值，x 为弧度
double asin(double x)	求 x 的反正弦函数的值，x 为弧度
double atan(double x)	求 x 的反正切函数的值，x 为弧度
double atan2(double y,double x)	求 y/x 的反正切函数的值，y 和 x 为弧度
double cos(double x)	返回 x 的余弦 cos(x) 值，x 为弧度
double sin(double x)	返回 x 的正弦 sin(x) 值，x 为弧度
double tan(double x)	返回 x 的正切 tan(x) 值，x 为弧度
double cosh(double x)	返回 x 的双曲余弦 cosh(x) 值，x 为弧度
double sinh(double x)	返回 x 的双曲正弦 sinh(x) 值，x 为弧度
double tanh(double x)	返回 x 的双曲正切 tanh(x) 值，x 为弧度

第3章　程序的选择执行

当我们走出家门时，会自觉不自觉地看一下外面的天气，如果下雨就带上雨伞。当我们走到十字路口需要过马路时，会依据红绿灯信息选择停下来还是过马路。上课时，我们会依据课程表选择上课教室。学校运动会，我们每个人要选择是否参加项目比赛，若选择参加，要参加哪个项目。购买衣服时，我们会依据自己的喜好选择衣服款式和色彩……生活中需要我们依据不同的条件和情况选择做不同事情的现象到处存在，在计算机语言中，同样需要引入选择结构（或称分支结构）来描述选择事件的解决过程。

3.1　if语句和关系表达式

【例3.1】星星公司致力于信件快递业务，收费标准为：500克以内6元，超过500克9元。

分析：这是一个选择问题，快递员依据信件的重量W值选择收取费用C，用数学表达式表示如下：

$$C = \begin{cases} 6 & w <= 500 \\ 9 & w > 500 \end{cases}$$

程序如下：

```
1  //exam3.1
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      int c;                  //定义变量
7      float w;                //定义变量
8      cout<<"w=";             //提示输入
9      cin>>w;                 //读入快件重量w
10     if(w<=500) c=6;         //如果w<=500，收费6元
11     else c=9;               //否则收费9元
```

```
12      cout<<"c="<<c<<endl;
13      return 0;
14 }
```

运行结果：

```
w=67
c=6
w=609.67
c=9
w=600
c=9
```

问题中用了if…else…语句，表示依据信件的重量W值，选择如何求费用C的值，对于选择问题，依据的条件是问题的关键，那么，如何表达条件？if…else…能解决怎样的问题？为了回答这些问题，我们将学习C++语言的if语句和关系表达式。

3.1.1　if语句格式

格式1：

if（表达式）　语句

功能：当条件成立即表达式值为真时，执行“语句”，否则执行**if语句**下方的语句。执行流程如图3.1所示。

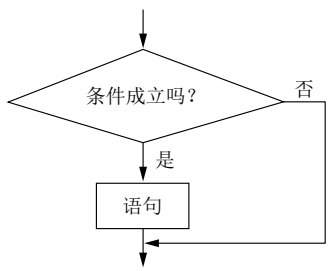


图3.1　if语句执行过程

【例3.2】读入一个整数，输出该数，如果该数是负数，在输出该数前加个提示“注意负数!”。

分析：对于负数需要输出“注意负数!”的提示。设n存放读入的数，当n<0时，输出“注意负数!”。最后输出n的值。

程序如下：

```
1  //exam3.2
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
```

```
6       int n;                          //定义变量
7       cout<<"n=";                     //提示输入
8       cin>>n;                         //读入数据
9       if(n<0)                         //如果n小于0,输出“注意负数!”
10      cout<<"注意负数!"<<endl;
11      cout<<n<<endl;                  //输出n值
12      return 0;
13 }
```

运行结果：

```
n=-5
注意负数!
-5
n=5
5
```

说明：程序中第9行，在写条件时，条件式要加括号，第10行是满足条件要做的事。第11行是正常执行的语句。

格式2：

```
if（表达式）
    语句1;
else
    语句2;
```

功能：当条件成立即表达式值为真时，执行“语句1”，否则执行“语句2”。执行流程如图3.2所示。

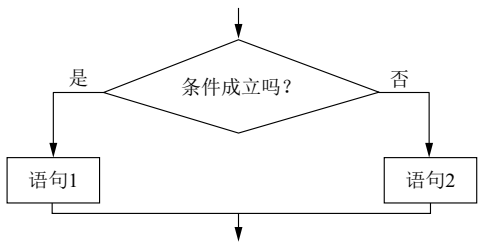


图3.2　if…else…语句执行过程

【例3.3】读入一个整数，判定其是偶数还是奇数。

分析：一个整数如果是偶数，那么该数除2的余数为0。设n存放读入的数，那么，如果n%2为0，则为偶数，否则为奇数。

程序如下：

```
1  //exam3.3
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int n;                          //定义变量
7      cin>>n;                         //读入数据
8      if(n%2==0)                      //判断 n 除 2 的余数是否为 0
9      cout<<n<<" 是偶数 "<<endl;      //条件式成立输出偶数
10     else cout<<n<<" 是奇数 "<<endl; //条件式不成立输出奇数
11     return 0;
12 }
```

运行结果：

```
191
191是奇数
234
234是偶数
```

说明：程序第 8 行表示判断 n 除以 2 的余数是否等于 0。特别注意，条件式中是否等于的书写是“==”，而不是“=”，条件式书写时加括号，第 9 行是条件成立时要执行的语句，第 10 行是条件不成立时要执行的语句。

【例 3.4】星星音乐社团招收社员，依据音乐成绩发放不同的广告，音乐成绩不高于 80 分的同学发的广告单内容是“欢迎你参加音乐社”，其他同学发的广告单内容是“非常欢迎你参加音乐社”。

方法 1：打印广告单时，可以这么考虑，依据输入的音乐成绩 M，当 M>=80 时，打印“非常欢迎你参加音乐社”，否则，打印“欢迎你参加音乐社”。

程序如下：

```
1  //exam3.4-1
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int m;                      //定义变量
7      cout<<"m=";                 //提示输入
8      cin>>m;                     //读入音乐成绩
9      if(m>=80)cout<<" 非常欢迎你参加音乐社 ";
```

```
                                        // 依据音乐成绩发放不同广告内容
10      else  cout<<" 欢迎你参加音乐社 ";
11      return 0;
12 }
```

方法 2：也可以这么考虑，分析广告词的特点，音乐成绩高于 80 分的人的广告词比其他人多了“非常”两字，即当 M>=80 时，先打印“非常”，然后跟所有人一样打印“欢迎你参加音乐社”。

程序如下：

```
1  //exam3.4-2
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6       int m;                          // 定义变量
7       cout<<"m=";                     // 提示输入
8       cin>>m;                         // 读入音乐成绩
9       if(m>=80)cout<<" 非常 ";        // 依据音乐成绩发放不同广告内容
10      cout<<" 欢迎你参加音乐社 ";
11      return 0;
12 }
```

运行结果：

```
m=67
欢迎你参加音乐社
m=89
非常欢迎你参加音乐社
```

思考：对于同一个条件，方法 1 有两个分支，方法 2 只有一个分支，请思考它们的区别。

3.1.2 关系表达式

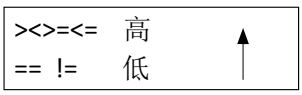
选择结构问题的一个关键点是选择条件的描述，即 if 语句中 (表达式) 的具体表达。

在前面的例子中，我们很自然地用上了数学的比较符描述 if 语句中的条件。与数学比较符相对应的是关系运算符。

关系表达式运算符：

等于	不等于	大于	小于	大等于	小等于
==	!=	>	<	>=	<=

优先级别：



用关系运算符将两个表达式连接起来的式子，称为**关系表达式**。关系表达式的一般形式可以表示为：

表达式　关系运算符　表达式

其中的“表达式”可以是算术表达式，也可以是关系表达式、逻辑表达式、赋值表达式、字符表达式。

关系表达式的值是一个逻辑值，即“真”或“假”，如果为“真”，则表示条件成立；如果为“假”，则表示条件不成立。例如，关系表达式“1==3”的值为“假”，“3>=0”的值为“真”。在C++中用数值1代表“真”，用0代表“假”。

【例3.5】设a=5,b=6,c=7。写出下面关系表达式的值。

关系表达式	值	分　析
a>b	0	因为a=5,b=6，所以条件不成立
a+b>b+c	0	因为a+b=11，b+c=13，所以条件不成立
(a==3)>=(b==5)	1	因为a==3不成立值为0，b==5不成立值为0，所以两者相等成立
'a'<'b'	1	字符'a'的ASCII码小于字符'b'的ASCII码，所以条件成立
(a>b)>(b<c)	0	a>b值为0，b<c值为0，所以条件不成立

【例3.6】为了学生的卫生安全，学校给每个住宿生配一个水杯，每只水杯3元，大洋商城打八八折，百汇商厦“买八送一”。输入学校想买水杯的数量，请你当“参谋”，算一算：到哪家购买较合算？输出商家名称。

分析：设变量cup存放读入的水杯数量，变量a为到大洋商城购买水杯的费用，变量b为到百汇商厦购买水杯的费用。那么：

```
a=cup*3*0.88
b=(cup-cup/8)*3
```

式中，cup/8是求cup除以8的商，即“买八送一”送的杯子数量。

如果a<b，那么到大洋商城购买，否则到百汇商厦购买。

程序如下：

```
1  //exam3.6
```

```
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int cup;                  //定义变量
7      float a,b;
8      cout<<"cup=";             //提示输入
9      cin>>cup;                 //输入购买的杯子数量
10     a=cup*3*0.88;             //求大洋商城购买水杯的费用
11     b=(cup-cup/8)*3;          //求百汇商厦购买水杯的费用
12     if(a<b) cout<<"大洋商城"<<endl;
                                 //比较a、b值，输出购买商家
13     else cout<<"百汇商厦"<<endl;
14     return 0;
15 }
```

```
cup=85
大洋商城
cup=180
百汇商厦
```

说明：问题中的关系表达式是到两商场购买杯子的费用比较，为了方便关系式的书写，程序中第10、11、12行，采用先求出购买的费用，再进行比较的方法。当需要比较的表达式比较复杂时，问题提供了比较好的解决方式。

【例3.7】学校开发了一片区域准备种果树，依据校友捐款选择树种，如果捐款小于10万，只种梨树，每棵梨树500元；捐款大于等于10万，30%用于种梨树，50%用于种桃树，每棵桃树600元，20%用于种苹果树，每棵苹果树800元。输入捐款，输出各种果树种多少棵。

分析：设变量money存放读入的捐款，如果money小于10万，求种梨树数量并输出；否则，求种梨树数量并输出、种桃树数量并输出、种苹果树数量并输出。

程序如下：

```
1  //exam3.7
2  #include<iostream>
3  #include<cmath>
4  using namespace std;
```

```
5  int main()
6  {
7       int money;          //定义变量
8       cout<<"money=";     //提示输入
9       cin>>money;         //输入捐款
10      if(money<100000)cout<<"梨树 ="<<money/500<<" 棵 "<<endl;
                            //捐款小于 10 万，输出梨树棵数
11      else                //捐款大于等于 10 万，输出各种树种的棵数
12      {
13      cout<<" 梨树 ="<<floor(money*0.3/500)<<" 棵 "<<endl;
14      cout<<" 桃树 ="<<floor(money*0.5/600)<<" 棵 "<<endl;
15      cout<<" 苹果树 ="<<floor(money*0.2/800)<<" 棵 "<<endl;
16      }
17      return 0;
18 }
```

```
money=80000
梨树=160棵
```

```
money=120000
梨树=72棵
桃树=100棵
苹果树=30棵
```

说明：当 if 和 else 后面有多个要操作的语句时，如：程序中第 13、14、15 表示在一个分支下要完成的操作，要用花括号“{}”括起来，将几个语句括起来的语句组合称为**复合语句**。

练　习

（1）运行下列程序，分别输入三组数据：

5 5

5 6

6 5

观察运行结果，说明 if 语句中“=”和“==”的区别。

```
//test(1)-1
#include<iostream>
using namespace std;
int main()
{
     int a,b;
     cout<<"a,b=";
     cin>>a>>b;
     if (a = b) cout<<a;
     else cout<<"Unequal";
     return 0;
}
```

```
//test(1)-2
#include<iostream>
using namespace std;
int main()
{
     int a,b;
     cout<<"a,b=";
     cin>>a>>b;
     if(a==b)cout<<a;
     else cout<<"Unequal";
     return 0;
}
```

（2）运行下列程序，分别输入两组数据：

5 7

7 6

观察运行结果，说明分支语句中{}的作用。

```
//test(2)-1
#include<iostream>
using namespace std;
int main()
{
     int a,b,c=0,d=0;
     cout<<"a,b=";
     cin>>a>>b;
     if(a>b)
     {
          c=a/b;
          d=a%b;
     }
     cout<<c+d;
     return 0;
}
```

```
//test(2)-2
#include<iostream>
using namespace std;
int main()
{
     int a,b,c=0,d=0;
     cout<<"a,b=";
     cin>>a>>b;
     if(a>b)
        c=a/b;
        d=a%b;
     cout<<c+d;
     return 0;
}
```

（3）输入一个三位数n，判断是否为水仙花数，如果是，则输出“该数是水仙花数”；不是，则输出“该数不是水仙花数”。**水仙花数**：是指一个3位

数，它的每个位上的数字的3次幂之和等于它本身。（例如：$1^3+5^3+3^3=153$）

3.2　逻辑表达式和条件表达式

【例3.8】班级评选先进个人，其中一个条件是语文成绩不低于75分且数学成绩不低于85分，输入语文和数学成绩，输出该生是否有资格参选。

分析：解决问题的关键是如何表达语文成绩不低于75分且数学成绩不低于85分这样的条件，可以用if嵌套实现，也可以用逻辑表达式实现。用逻辑表达式实现比较直观。

设cmark存放读入的语文成绩，mmark存放读入的数学成绩，C++使用&&符号连接两个条件需要同时满足的条件。

程序如下：

```
1  //exam3.8
2  #include<iostream>
3  #include<cmath>
4  using namespace std;
5  int main()
6  {
7       int cmark,mmark;
8       cin>>cmark>>mmark;   // 读入成绩
9       if(cmark>=75&&mmark>=85) cout<<" 有资格 "<<endl;
                 // 依据语文和数学成绩是否同时满足条件输出有无参选资格
10      else cout<<" 无资格 "<<endl;
11      return 0;
12 }
```

运行结果：

```
85 90
有资格
60 90
无资格
```

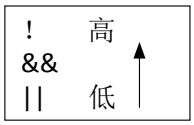
问题中条件表示用了逻辑与“&&”，那么，逻辑与的运算规则是什么？逻辑运算符有哪些？如何应用逻辑运算符？为了回答这些问题，我们将学习C++语言的逻辑表达式。

3.2.1　逻辑运算和逻辑表达式

逻辑运算符：

逻辑与	逻辑或	逻辑非
&&	||	!

优先级别：



逻辑运算符中的“&&”和“||”低于关系运算符，“!”高于算术运算符。

将两个关系表达式用逻辑运算符连接起来的表达式，称为**逻辑表达式**，逻辑表达式的一般形式可以表示为：

表达式　逻辑运算符　表达式

逻辑表达式的值是一个逻辑值。在C++中，整型数据可以出现在逻辑表达式中，在进行逻辑运算时，根据整型数据的值是0或非0，把它作为逻辑值“假”或“真”，然后参加逻辑运算。

下面给出逻辑运算真值表，约定：A、B为两个条件，值为0表示条件不成立，值为1表示条件成立。

1．逻辑非

真值表如下，经过逻辑非运算，其结果与原来相反。

A	!A
0	1
1	0

2．逻辑与

真值表如下，若参加运算的某个条件不成立，其结果为不成立，只有当参加运算的条件都成立，其结果才成立。

A	B	A&&B
0	0	0
0	1	0
1	0	0
1	1	1

3．逻辑或

真值表如下，若参加运算的某个条件成立，其结果就成立，只有当参

加运算的所有条件都不成立，其结果才不成立。

A	B	A||B
0	0	0
0	1	1
1	0	1
1	1	1

【例 3.9】根据问题写出条件表达式。

问　题	条件表达式
判断一个数n是否可以同时被2与3整除	`if (n%2==0&&n%3==0)` 或者 `if(n%6==0)`
判断一个数x是否在区间[1，5]之内	`if(x>=1&&x<=5)` 或者 `if(!(x<1||x>5))`
判断一个数x是否等于0	正向判断：`if(x==0)` 反向判断：`if(x!=0)` 或者 `if(!x)`

【例 3.10】输入年份year，输出该年是否为闰年。

分析：设变量year存放读入的年份。闰年的条件是：年份能被4整除但是不能被100整除或者能被400整除。

表示“年份能被4整除但是不能被100整除”的逻辑表达式为：

```
(year%4==0&&year%100!=0)
```

表示“年份能被400整除”的条件表达式为：

```
year%400==0
```

两个条件式构成“或”的关系，逻辑表达式表示如下：

```
(year%4==0&&year%100!=0)||year%400==0
```

当表达式值为真时，则year为闰年，否则year为非闰年。

程序如下：

```
1  //exam3.10
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int year;
7      cin>>year;
8      if((year%4==0&&year%100!=0)||year%400==0)   // 闰年判断
9      cout<<year<<" 是闰年 "<<endl;               // 输出闰年
10     else cout<<year<<" 不是闰年 "<<endl;        // 输出不是闰年
```

```
11      return 0;
12 }
```

运行结果：

```
2016
2016是闰年
2014
2014不是闰年
```

思考：对于程序第8行的逻辑表达式有否其他的表达方式？

【*例3.11】输入三角形的三条边a、b、c的值，判断是否构成三角形，若构成三角形，则求三角形的面积。

分析：根据数学定理，构成三角形三边满足任意两边之和大于第三边的条件。逻辑表达式表示为：

a+b>c && b+c>a && a+c>b

当表达式值为真时，构成三角形，求三角形面积，否则，不构成三角形。

也可以说成，只要某两边之和小等于第三边就不构成三角形，其逻辑表达式表示：

a+b<=c || b+c<=a || a+c<=b

当表达式值为真时，不构成三角形，否则构成三角形，求三角形面积。

程序如下：

```
1  //exam3.11-1
2  #include<cmath>
3  #include<iostream>
4  using namespace std;
5  int main()
6  {
7       float a,b,c;
8       float p;
9       float s;
10      cin>>a>>b>>c;
11      if(a+b>c&&a+c>b&&b+c>a)
12      {
13        p=(a+b+c)/2;
14        s=sqrt(p*(p-a)*
          (p-b)*(p-c));
```

```
1  //exam3.11-2
2  #include<cmath>
3  #include<iostream>
4  using namespace std;
5  int main()
6  {
7       float a,b,c;
8       float p;
9       float s;
10      cin>>a>>b>>c;
11      if(a+b<=c||a+c<=b||b+c<=a)
12        cout<<" 不能构成三角形 ";
13      else
14      {
```

```
15        cout<<" 三角形面积为:
          "<<s;
16      }
17      else
18      cout<<" 不能构成三角形 ";
19      return 0;
20  }
```

```
15        p=(a+b+c)/2;
16        s=sqrt(p*(p-a)*(p-b)*
          (p-c));
17        cout<<" 三角形面积为: "<<s;
18      }
19      return 0;
20  }
```

运行结果：

```
3 4 5
三角形面积为：6
8 45 36
不能构成三角形
```

思考：两程序分别用逻辑与和逻辑或来解决相同的问题，那么，逻辑与和逻辑或的表达式能否相互转换呢？

3.2.2　逻辑变量

逻辑变量用类型标识符 bool 来定义，它的值只有 true（真）或 false（假）两种。

由于逻辑变量是用关键字 bool 来定义的，因此又称为布尔变量。

C++ 编译系统在处理逻辑型数据时，将 false 处理为 0，将 true 处理为 1。因此，逻辑型数据可以与数值型数据进行算术运算。

如果将一个非零的整数赋给逻辑型变量，则按“真”处理。

【例 3.12】阅读下列程序和运行结果，说一说逻辑变量的特点。

```
1  //exam3.12
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6       bool found, flag=false;
       // 定义逻辑变量 found 和 flag，并使 flag 的初值为 false
7       found=true;                  // 让逻辑变量 found 值为 true
8       cout<<flag<<" "<<found<<endl;
9       flag=5;                      // 赋值后 flag 的值为 true
10      found=0;                     // 赋值后 found 的值为 false
11      cout<<flag<<" "<<found<<endl;
```

```
12      return 0;
13 }
```

运行结果：

```
0 1
1 0
```

说明：程序第6行和第7行表示0代表false，1代表 true。程序的第9行表示非0的逻辑变量值为1。

*3.2.3　条件表达式

格式：

<表达式1>？<表达式2>：<表达式3>

条件表达式要求有3个操作对象，“?”和“:”一起出现在条件表达式中，称三目(元)运算符，它是C++中唯一的一个三目运算符。

条件表达式的运算规则：

（1）计算表达式1的值。

（2）若表达式1的值为真（或非0），则只计算表达式2，并将其结果作为整个表达式的值。

（3）反之，若表达式1的值为假（或为0），则只计算表达式3，并将其结果作为整个表达式的值。

【例3.13】解释下列条件表达式的作用。

条件表达式	作　用
`int maxn=(a>b)?a:b;`	赋值，将两个变量的较大值赋予maxn整型变量中
`cout<<((num%2==0)?"num is even" : "num is odd")<<endl;`	在num为偶数时，输出even，奇数时，输出odd
`y=(x>0)? 1:-1;`	在x>0时，将1赋给y，在x<=0时，将–1赋给y

思考：对表中第2个条件表达式去除最外层括号写成：“cout<<(num%2==0)?"num is even":"num is odd"<<endl;”是否可行？为什么？

【例3.14】输入一个字符，判别它是否为大写字母，如果是，将它转换成小写字母；如果不是，不转换。然后输出最后得到的字符。

分析：设ch存放输入的字符，当满足条件ch>='A'&&ch<='Z'时为大写字母，将它转换成小写字母，大写字母ASCII编码加32即为小写字母。

程序如下：

```
1  //exam3.14
2  #include <iostream>
3  using namespace std;
4  int main( )
5  {
6       char ch;
7       cin>>ch;
8       ch=(ch>='A' && ch<='Z')?(ch+32):ch;
          // 判别 ch 是否为大写字母，是，则转换
9       cout<<ch<<endl;
10      return 0;
11 }
```

运行结果：



说明：C++语言表达能力强，其中一个重要方面就在于它的表达式类型丰富，运算符功能强，因而使用灵活，适应性强。

练　习

（1）在社会实践活动中有三项任务分别是：种树、采茶、送水。依据小组人数及男生、女生人数决定小组接受什么任务，人数小于10人的小组负责送水，人数大于等于10人且男生多于女生的小组负责种树，人数大于等于10人且男生不多于女生的小组负责采茶。输入小组男生人数、女生人数，输出小组接受的任务。

（2）某邮局对邮寄包裹有如下规定：若包裹的重量超过30千克，不予邮寄；对可以邮寄的包裹每件收手续费0.2元，再加上根据下表按重量wei计算的结果：

重量(千克)	收费标准(元/千克)
wei<=10	0.80
10<wei<=20	0.75
20<wei<=30	0.70

请你编写一个程序，输入包裹重量，输出所需费用或“无法邮寄”。

（3）有一个正方形，四个角的坐标（x，y)分别是（1，-1），（1，1），（-1，-1），（-1，1），x 是横轴，y 是纵轴。写一个程序，判断一个给定的点（x0，y0）是否在这个正方形内（包括正方形边界）。

（4）一密码变换规则如下：一个正整数对应一个字符；如果该数模 123 的值在 97-122 范围，则变换为小写字符；如果变换不了小写字符，则将该数模 91，若余数在 65-90 范围，则变换为大写字符；如果变换不了大小写字符，则变换为“*”。输入一个正整数，输出变换后的字符。

3.3 嵌套 if 语句

【例 3.15】输入年份 year，输出该年是否为闰年。

分析：在例 3.10 中用逻辑表达式表示闰年的条件，对于年份能被 400 整除或者能被 4 整除但是不能被 100 整除的闰年条件，也可以用图 3.3 结构表示。

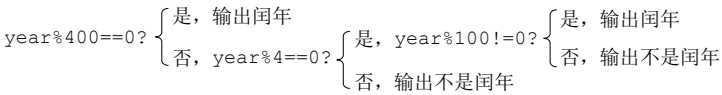


图 3.3 闰年分支结构图

程序如下：

```
1  //exam3.15
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int year;
7      cin>>year;
8      if(year%400==0) cout<<year<<" 是闰年 "<<endl;
                                                    // 第 1 层分支
9      else
10       if(year%4==0)                              // 第 2 层分支
11         if(year%100!=0) cout<<year<<" 是闰年 "<<endl;
                                                    // 第 3 层分支
12         else cout<<year<<" 不是闰年 "<<endl;
13       else cout<<year<<" 不是闰年 "<<endl;
```

```
14      return 0;
15 }
```

运行结果：

```
2000
2000是闰年
```

```
1997
1997不是闰年
```

对应于图3.3的闰年分支结构图，在程序表达过程中，我们自然而然地用到了嵌套if语句。那么，使用嵌套if语句需要注意什么？如何正确应用嵌套if语句解决问题？为了回答这些问题，我们将学习嵌套if语句的使用。

3.3.1　嵌套if语句

嵌套if语句是指在if…else分支中还存在if…else语句。

在例3.15程序中，用了3层if嵌套语句。在用嵌套if语句表达问题时，最重要的是先把问题的分支逻辑关系分析清楚，如图3.3所示，把判断闰年的分支逻辑关系梳理清楚，接着用if表达分支逻辑关系就顺理成章了。

在使用嵌套if语句时，需要特别注意if与else的配对关系，else总是与它上面最近的、且未配对的if配对。

【例3.16】分析下面两个程序的区别？

程序1：

```
1  //exam3.16-1
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6       int n;
7       cin>>n;
8       if(n%3==0)
9       if(n%5==0) cout<<n<<" 是15的倍数 "<<endl;
10      else cout<<n<<" 是3的倍数但不是5的倍数 "<<endl;
11      cout<<" 结束 "<<endl;
12      return 0;
13 }
```

运行结果：

```
30
30是15的倍数
结束
```

```
18
18是3的倍数但不是5的倍数
结束
```

```
13
结束
```

说明： 程序中的 else 与第二个（第 9 行）if 配对，语句只针对被 3 整除的数，判定输出“是 15 的倍数”还是“是 3 的倍数但不是 5 的倍数”。

程序 2：

```
1  //exam3.16-2
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int n;
7      cin>>n;
8      if (n%3==0)
9      {
10       if(n%5==0) cout<<n<<" 是 15 的倍数 "<<endl;
11     }
12     else cout<<n<<" 不是 3 的倍数 "<<endl;
13     cout<<" 结束 "<<endl;
14     return 0;
15 }
```

运行结果：

```
30
30是15的倍数
结束
```

```
18
结束
```

```
13
13不是3的倍数
结束
```

说明： 语句中的 else 与第一个（第 7 行）if 配对，程序对于数 n，判定是否被 3 整除，被 3 整除时，再判断是否被 5 整除，若是，则输出“是 15 的倍数”的数，不被 3 整除时，则输出“不是 3 的倍数”。

两个程序的差别虽然仅在于一对“{}”，但逻辑关系却完全不同。一些数据的运行结果也不相同。

为了清晰表达嵌套if语句，通常程序中采用缩进方式表示，让同层的if与else对齐。

3.3.2　嵌套if语句应用

【例3.17】某商场优惠活动规定，某商品一次购买5件以上（包含5件）10件以下（不包含10件）打9折，一次购买10件以上（包含10件）打8折。设计程序根据单价和客户的购买量计算总价。

分析：设price表示商品单价，count表示购买商品数量，discount表示折扣，amount客户付费。根据题意，折扣与商品数量的关系表示如下：

$$discount=\begin{cases}1 & count<5\\ 0.9 & 5<=count<10\\ 0.8 & count>=10\end{cases}$$

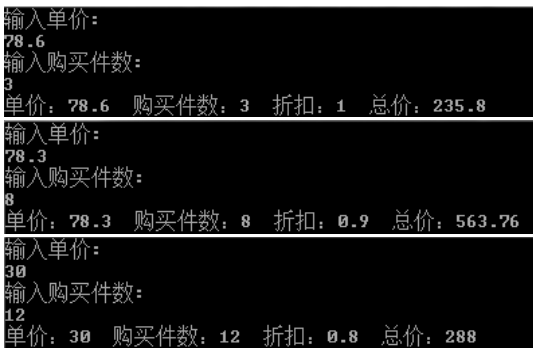
付费：

amount=price*count*discount

程序如下：

```
1  //exam3.17
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      float price,discount,amount;   // 定义单价、折扣、总价
7      int count;                     // 定义购买数量
8      cout<<" 输入单价："<<endl;
9      cin>>price;
10     cout<<" 输入购买件数："<<endl;
11     cin>>count;
12     if(count<5)  discount=1;       // 购买数量小于5件，没有折扣
13     else if(count<10) discount=0.9;  // 购买5件以上10件以下，9折
14     else  discount=0.8;            // 购买10件以上，8折
15     amount=price*count*discount;   // 求付费总价
16     cout<<" 单价："<<price<<"  购买件数："<<count<<"  折扣：
       "<<discount<<" 总价："<<amount<<endl;       // 输出结果
17     return 0;
18 }
```

运行结果：



思考：对于分支程序，应该如何设计测试数据，验证程序的正确性？

【例3.18】输入三个数，输出其中最大的数。

方法1：设maxn用于存放三个数中最大的数，输入的三个数存放在a、b、c中，那么如果a比b和c大，则最大数是a，否则，如果b比a和c大，则最大数是b，否则，最大数是c。

程序如下：

```
1  //exam3.18-1
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      float a, b, c, maxn;
7      cout<<"输入三个数：";
8      cin>>a>>b>>c;
9      if(a>b&&a>c)  maxn=a;          //判断a是否最大
10     else if(b>a&&b>c)  maxn=b;    //判断b是否最大
11     else  maxn=c;
12     cout<<"最大数为："<<maxn<<endl;
13     return 0;
14 }
```

方法2：设maxn用于存放三个数中最大的数，输入的三个数存放在a、b、c中，初值maxn=a，即假设a为最大，那么如果b>maxn，则此时的最大数应该是b即maxn=b，如果c>maxn，则最大数应该是c即maxn=c。

程序如下：

```
1  //exam3.18-2
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      float a, b, c, maxn;
7      cout<<"输入三个整数：";
8      cin>>a>>b>>c;
9      maxn=a;
10     if(b>maxn) maxn=b;          //maxn 为 a,b 中的最大值
11     if(c>maxn) maxn=c;          //maxn 为 a,b,c 中的最大值
12     cout<<"最大数为："<<maxn<<endl;
13     return 0;
14 }
```

运行结果：

```
输入三个整数:8.5 9.6 12.5
最大数为：12.5
输入三个整数:9.6 12.5 8.5
最大数为：12.5
输入三个整数:12.5 9.6 8.5
最大数为：12.5
```

思考：

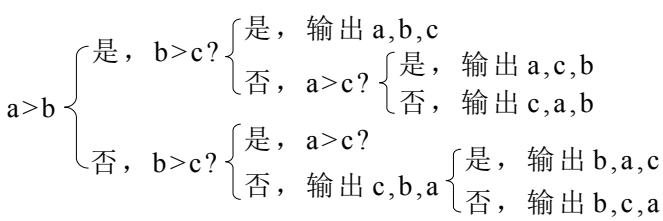
（1）方法 2 比方法 1 有哪些优点？

（2）运行结果中 3 组测试的作用。

（3）方法 1 中程序使用嵌套 if 语句，方法 2 中程序使用并列 if 语句，在程序运行过程中它们有何区别？

【例 3.19】输入三个数，按从大到小的顺序输出。

方法 1：输入的三个数存放在 a、b、c 中，那么存在下面几种情况。



程序如下：

```
1  //exam3.19-1
```

```
#include<iostream>
using namespace std;
int main()
{
    float a,b,c;
    cin>>a>>b>>c;
    if(a>b)      // 确定了 a 在 b 前面的顺序
      if(b>c)    //c 在 b 之后
        cout<<a<<","<<b<<","<<c;
      else
        if(a>c) //c 在 a 与 b 中间
          cout<<a<<","<<c<<","<<b;
        else     //c 在 a 之前
        cout<<c<<","<<a<<","<<b;
    else       // 确定了 a 在 b 之后的顺序，接着讨论 c 所在的位置
    if(b>c)
    if(a>c)
        cout<<b<<","<<a<<","<<c;
      else
        cout<<b<<","<<c<<","<<a;
      else
        cout<<c<<","<<b<<","<<a;
    return 0;
}
```

方法 2：输入的三个数存放在 a、b、c 中，设想让 a 为三数中最大数，怎么做呢？如果 a<b，那么让 a 与 b 的值交换，保证了 a>=b；如果 a<c，那么让a与c的值交换，保证了a>=c。设想让b为第二大的数，c为第三大的数，怎么做呢？如果 b<c，那么让 b 与 c 的值交换，保证了 b>=c，最后，输出 a,b,c。

程序如下：

```
//exam3.19-2
#include<iostream>
using namespace std;
int main()
{
    int a,b,c,temp;
    cin>>a>>b>>c;
    if(a<b)        // 保证 a 大于等于 b
```

```
9       {
10        temp=a; a=b; b=temp;
11      }
12      if(a<c)          // 保证 a 大于等于 c，则 a 为最大数
13      {
14          temp=a;a=c;c=temp;
15      }
16      if(b<c)          // 保证 b 大于等于 c
17      {
18          temp=b;b=c;c=temp;
19      }
20      cout<<a<<" "<<b<<" "<<c<<endl;
21      return 0;
22  }
```

思考：

（1）对该程序要检验程序的正确性，应该设计多少组数据？数据应具有怎样的特点？

（2）比较方法 1 和方法 2，哪种方法更好？好在哪里？

从例 3.18 和例 3.19 看到，一个问题可以有不同的解决方法即可以有不同的算法，不同的算法对解决问题的局限性、程序表达的便捷性以及以后学习中还会涉及的程序效率等带来不一样的效果。一个问题的解决从程序设计角度来看，没有标准答案，只有更好的方案，这也是程序设计的魅力之一。

练　习

（1）下面程序段，对于给定的几组数据，输出怎样的结果？

（A）x=3，y=2　（B）x=2，y=3　（C）x=3，y=4

（D）x=2，y=2　（E）x=3，y=3

程序段 1：

```
//test (1) -1
if(x>2)
      {
        if (y>2)
        {
          int z=x+y;
          cout<<"z ="<<z<<endl;
        }
      }
else
      cout<<"x="<<x<<endl;
```

程序段 2：

```
//test (1) -2
if(x>2)
     if(y>2)
        {
          int z=x+y;
          cout<<"z ="<<z<<endl;
        }
      else
        cout<<"x="<<x<<endl;
```

（2）输入三个正整数，判断能否构成三角形的三边，如果不能，输出不构成三角形。如果能构成三角形，判断构成什么三角形？按等边、直角、一般三角形顺序，输出对应的三角形类型。

（3）输入某学生成绩，根据成绩好坏输出相应评语。如果成绩大于等于90分，则输出“优秀”；如果成绩大于等于80分且小于90分，则输出“良好”；如果成绩大于等于60分且小于80分，则输出“及格”；成绩小于60分，则输出“不及格”。

3.4 switch 语句

【例3.20】一个最简单的计算器支持+,–,*, / 四种运算。输入只有一行：两个参加运算的数和一个操作符（+,–,*,/）。输出运算表达式及结果。考虑下面两种情况：

（1）如果出现除数为0的情况，则输出：Divided by zero!

（2）如果出现无效的操作符(即不为 +, –, *, /之一)，则输出：Invalid operator!

输入样例：

34 56 +

输出样例：

34+56=89

分析：设num1、num2存放两个参加运算的操作数，op存放操作符。

●当op为“+”号时，实现加法操作。

●当op为“-”号时，实现减法操作。

●当op为“*”号时，实现乘法操作。

●当op为“/”号时，判断b值，如果不为0，则实现乘法操作，如果为0，则输出：Divided by zero!。

●当op不是上面四种操作符时，输出：“Invalid operator!”。

●程序可以用if语句来实现，然而，C++提供了解决此类问题更直观便捷的语句。

程序如下：

```
1  //exam3.20
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      float num1,num2;
7      char op;
8      cin>>num1>>num2>>op;
9      switch(op)
10     {
11       case'+':cout<<num1<<op<<num2<<"="<<num1+num2
         <<endl; break;
12       case'-':cout<<num1<<op<<num2<<"="<<num1-
         num2<<endl; break;
13       case'*':cout<<num1<<op<<num2<<"="<<num1*num2
         <<endl; break;
14       case'/':if(num2!=0)cout<<num1<<op<<num2<<"="
         <<num1/num2<<endl;
15       else cout<<"Divided by zero!"<<endl;break;
16       default:cout<<"Invalid operator!";
17     }
18     return 0;
19 }
```

运行结果：

```
34 56 +
34+56=90
56 23 -
56-23=33
```

```
67 23 *
67*23=1541
56 23 /
56/23=2.43478
34 0 /
Divided by zero!
23 45 [
Invalid operator!
```

程序中使用switch语句描述分支问题，那么，switch语句与if语句有何区别？什么样的问题更适合使用switch语句？为了回答这些问题，我们将学习switch语句的使用。

3.4.1 switch语句格式

基本格式如下：

```
switch（表达式）
{
    case常量表达式1:[语句组1][break;]
        ……
    case常量表达式n:[语句组n][break;]
    [default:语句组n+1]
}
```

功能： 首先计算表达式的值，case后面的常量表达式值逐一与之匹配，当某一个case分支中的常量表达式值与之匹配时，则执行该分支后面的语句组，然后顺序执行之后的所有语句，直到遇到break语句或switch语句的右括号“}”为止。如果switch语句中包含default，default表示表达式与各分支常量表达式的值都不匹配时，执行其后面的语句组，通常将default放在最后。

规则：

（1）合法的switch语句中的表达式，其取值只能是整型、字符型、布尔型或枚举型。

（2）常量表达式是由常量组成的表达式，值的类型与表达式的类型相同。

（3）任意两个case后的常量表达式值必须各不相同，否则将引起歧义。

（4）“语句组”可以是一个语句也可以是一组语句。

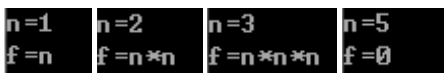
（5）基本格式中的[]表示可选项。

【例3.21】阅读下面两个程序和运行结果，说明两个程序的区别。

程序 1：

```
1  //exam3.21-1
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      int n;
7      cout<<"n=";
8      cin>>n;
9      switch(n)
10     {
11       case 1:cout<<"f=n"<<endl; break;
12       case 2:cout<<"f=n*n"<<endl; break;
13       case 3:cout<<"f=n*n*n"<<endl; break;
14       default:cout<<"f=0";
15     }
16     return 0;
17 }
```

运行结果：

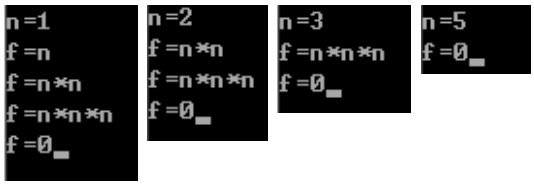


程序 2：

```
1  //exam3.21-2
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      int n;
7      cout<<"n=";
8      cin>>n;
9      switch(n)
10     {
11       case 1:cout<<"f=n"<<endl;
12       case 2:cout<<"f=n*n"<<endl;
13       case 3:cout<<"f=n*n*n"<<endl;
14       default:cout<<"f=0";
15     }
```

```
16      return 0;
17 }
```

运行结果：



说明：程序 2 与程序 1 的区别在于每个 case 分支有没有 break 语句，程序 1 每个 case 分支都有 break 语句，因此每个分支的运行结果很明确。程序 2 每个 case 分支没有 break 语句，即当执行匹配分支后，由于没有 break 语句，所以继续执行下面分支所有的语句，也就是产生多个常量表达式在这种情况下无效的现象，利用这一点可以精简程序，但往往容易产生歧义甚至错误结果！因此，对于分支明确的问题，switch 语句常用如下格式，避免出现分支执行混乱。

switch 语句一般使用如下格式：

```
switch (表达式)
{
    case 常量表达式 1：语句组 1；break；
        ……
    case 常量表达式 n：语句组 n；break；
    [default：语句组 n+1]
}
```

实验：

（1）对于程序 1 和程序 2，调整各 case 分支和 default 分支在程序中的先后顺序，说明其对程序运行结果的影响并说明为什么？

（2）如果将变量 n 类型改为 float 类型，编译运行结果会出现什么现象？为什么？

【例 3.22】阅读下面程序和运行结果。

```
1  //exam3.22
2  #include <iostream>
3  using namespace std;
4  int main()
```

```
5  {
6       char score;
7       cout<<"score=";
8       cin>>score;
9       switch (score)
10      {
11          case'A': case'a': cout<<"excellent"; break;
12          case'B': case'b': cout<<"good"; break;
13          default: cout<<"general";
14      }
15      return 0;
16 }
```

运行结果：

```
score=A
excellent
score=a
excellent
score=B
good
score=b
good
score=D
general
```

说明：运行程序看到，当score='A'和score='a'时，执行同一语句组；当score='B'和score='b'时，执行同一语句组，也就是switch语句支持多个常量表达式共用同一语句组。

【例3.23】恩格尔系数是德国统计学家恩格尔在19世纪提出的反映一个国家和地区居民生活水平状况的定律，计算公式为：

N=人均食物支出金额 ÷ 人均总支出金额 × 100%

联合国根据恩格尔系数的大小，对世界各国的生活水平有一个划分标准，即一个国家平均家庭恩格尔系数大于等于60%为贫穷；50%~60%为温饱；40%~50%为小康；30%~40%属于相对富裕；20%~30%为富裕；20%以下为极其富裕。

分析：设x表示人均食物支出金额，y表示人均总支出金额，n表示恩格尔系数，则：n=x/y*100。

方法1：使用if分支语句，按照题目叙述，依据n的值显示不同的生活水平。

程序如下：

```
1  //exam3.23-1
2  #include<cstdio>
3  int main()
4  {
```

```
5       float n;
6       float x,y;
7       scanf("%f %f",&x,&y);
8       n=100*x/y;        //求恩格尔系数
9       if(n>=60)  printf("贫穷\n");
10      else if(n>=50)  printf("温饱\n");
11      else if(n>=40)  printf("小康\n");
12      else if(n>=30)  printf("相对富裕\n");
13      else if(n>=20)  printf("富裕\n");
14      else printf("极其富裕\n");
15      return 0;
16  }
```

方法2：这是一个多重选择的问题，使用switch语句来描述问题的解决过程更为直观，然而，x/y*100的值是一个实型数，如果作为switch表达式，需要先转换为整型数。设n为整型变量，求x/y*100四舍五入的值为：

n= x/y*100+0.5

这里n的值分布在0~100之间，作为switch表达式寻找与之匹配的case值范围太大也不合适，进一步分析题意可以发现，n值是以10为间隔改变生活水平，因此，可以用n/10作为switch表达式。

程序如下：

```
1   //exam3.23-2
2   #include<cstdio>
3   int main()
4   {
5       int n;
6       float x,y;
7       scanf("%f %f",&x,&y);
8       n=100*x/y+0.5;//求恩格尔系数，对小数后一位四舍五入取整
9       switch (n/10) //依题意，可用整除10后的个位数表示相应范围
                        的恩格尔系数
10      {
11          case 0:case 1: printf("极其富裕\n");break;
12          case 2: printf("富裕\n");break;
13          case 3: printf("相对富裕\n");break;
14          case 4: printf("小康\n");break;
15          case 5: printf("温饱\n");break;
16          default: printf("贫穷\n");break;
```

```
17        }
18     return 0;
19 }
```

说明：程序中输入和输出参阅2.5节。

实验：运行程序，用不同的测试数据，获得程序的每一分支的运行结果。

思考：为什么两种方法的程序中对n变量类型设置不同？

【*例3.24】春节来临，小计想用自己的零花钱购买一些书送给贫困山区的小朋友，他来到书店挑了4本书，每本书的价格分别为6元、13元、15元、20元，小计想把钱用光同时尽量书本数量最多，输入小计的零花钱，输出每种价格书购买的数量（小计的零花钱为大于等于35元的整钱）。

样例输入：

3　6

样例输出：

6元：6　13元：0　15元：0　20元：0

分析：设小计购买6元、13元、15元、20元4种书的数量分别为a、b、c、d。

小计想要把钱用光的同时尽量使书本数量最多，则尽可能买价格为6元的书。

设小计有x元钱，不妨先买6元书的数量为a=x/6，剩余的零钱值为：0、1、2、3、4、5中的一个，我们发现13%6=1、15%6=3、20%6=2，也就是说它们的余数可以组合成0、1、2、3、4、5，当零钱不为0时，减少最少的6元书数量，与零钱组合换购其他种书，其他种书每种最多购买1本。

（1）如果零花钱x能被6整除，即x%6=0，则全买6元的书，那么b=0、c=0、d=0。

（2）如果剩1元，即x%6=1，则从a中退出2本，加上剩余的1元钱，买一本13元的书，那么a=a-2、b=1、c=0、d=0。

（3）如果剩2元，即x%6=2，则从a中退出3本，加上剩余的2元钱，可买一本20元钱的书，那么a=a-3、b=0、c=0、d=1。

（4）如果剩3元，即x%6=3，则从a中退出2本，加上剩余的3元钱，可买一本15元的书，那么a=a-2、b=0、c=1、d=0。

（5）如果剩4元，即x%6=4，则从a中退出4本，加上剩余的4元钱，

可买一本 13 元的书和一本 15 元的书，那么 a=a-4、b=1、c=1、d=0。

（6）如果剩 5 元，即 x%6=5，则从 a 中退出 5 本，加上剩余的 5 元钱，可买 1 本 15 元的书和一本 20 元的书，那么 a=a-5、b=0、c=1、d=1。

程序如下：

```
1  //exam3.24
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int x,a,b,c,d;
7      cin>>x;
8      a=x/6;
9      switch(x%6)
10     {
11       case 0: b=0,c=0,d=0;break;
12       case 1:a=a-2,b=1,c=0,d=0;break;
13       case 2:a=a-3,b=0,c=0,d=1;break;
14       case 3:a=a-2,b=0,c=1,d=0;break;
15       case 4:a=a-4,b=1,c=1,d=0;break;
16       case 5:a=a-5,b=0,c=1,d=1;break;
17     }
18     cout<<"6 元: "<<a<<" "<<"13 元: "<<b<<" "<<"15 元:
       "<<c<<" "<<"20 元: "<<d;
19     return 0;
20 }
```

运行结果：

```
36
6元:6  13元:0  15元:0  20元:0
```

```
49
6元:6  13元:1  15元:0  20元:0
```

```
64
6元:6  13元:1  15元:1  20元:0
```

```
51
6元:6  13元:0  15元:1  20元:0
```

```
71
6元:6  13元:0  15元:1  20元:1
```

思考：

（1）能否证明解决本问题的决策是最优的？

（2）什么样的问题解决方案用switch语句描述比较好？

练　习

（1）阅读例3.20的程序，解释程序的运行过程。

（2）输入数字1~7表示星期一至星期日，输出对应的星期几的英文名称。

（3）输入年份与月份，求该月共有多少天。

*（4）编程实现以下功能：查询水果的单价。有4种水果：苹果（apples）、梨（pears）、橘子（oranges）和葡萄（grapes），单价分别是3.00元/千克，2.50元/千克，4.10元/千克和10.20元/千克。

在屏幕上显示以下菜单(编号和选项)：当用户输入编号1～4，显示相应水果的单价(保留1位小数)；输入0，退出查询；输入其他编号，显示价格为0。

输入样例（括号内是说明）：

```
3(oranges的编号)
```

输出样例：

```
[1]apples
[2]pears
[3]oranges
[4]grapes
[0]Exit
price = 4.1
```

*3.5　分支结构程序设计实例

【例3.25】在劳动技术课上，老师拿来了不同长度铁丝，给每个同学发一根铁丝，要求用手上的铁丝制作固定面积的矩形框，小计想利用计算机帮助大家求出矩形的长和宽，这样就能快速完成任务。当然，解决问题的算法是要大家共同分析完成的。

输入样例：

18　20

输出样例：

矩形的长和宽分别为：5，4

输入样例：

8　5

输出样例：

找不到这样的矩形

分析：设铁丝长度为l，矩形面积为s，这是已知的。设所求矩形的宽为x，则长为l/2−x。那么：

$x(l/2-x)=s$

化简得：

$x^2-1/2lx+s=0$

问题转换为，已知l、s，求$x^2-1/2lx+s=0$一元二次方程的解。

根据数学求解的步骤，我们可以写出解决问题的算法如下：

（1）输入铁丝的长度l。

（2）输入矩形面积s。

（3）计算一元二次方程判别式d=l*l/4−4*s。

（4）如果d<0，则输出“找不到这样的矩形!”。

（5）否则，如果d=0，则求解x1=x2=l/4，输出解，如果d>0，求解x1=(l/2+sqr(d))/2，x2=(l/2−sqr(d))/2，输出解。

（6）结束。

程序如下：

```
1  //exam3.25
2  #include<iostream>
3  #include<cmath>
4  using namespace std;
5  int main()
6  {
7      double l,s,x1,x2;
8      double d;
9      cin>>l>>s;
10       d=l*l/4-4*s;                    // 计算一元二次方程判别式
11       if(d<0)
12         cout<<" 找不到这样的矩形! "<<endl;
13       else                            // 求方程的解
```

```
14        {
15          if(d==0)
16            x1=x2=l/4;
17          else
18          {
19            x1=(l/2+sqrt(d))/2;
20            x2=(l/2-sqrt(d))/2;
21          }
22        cout<<"矩形的长和宽分别为："<<x1<<","<<x2<<endl;
23        }
24        return 0;
25 }
```

说明：本题把一个实际问题用数学的一元二次方程求解，提供了一种把朴素的数学分析转换为用数学方程表示的思维方法。

思考：给出怎样不同的测试数据，能够测试程序的正确性。

【例3.26】小计买了一箱苹果共有n个，很不幸的是买完时箱子里混进了一条虫子。虫子每x小时能吃掉一个苹果，假设虫子在吃完一个苹果之前不会吃另一个，那么经过y小时这箱苹果中还有多少个苹果没有被虫子吃过？输入n、x、y，输出答案。

输入样例：

3　2　1

输出样例：

2

分析：根据题意，被虫子吃过的苹果个数为y/x值的向上取整，如果其值大于n，说明被虫子吃光。那么，没有被虫子吃过苹果个数res值为：

$$res=\begin{cases} n-[y/x] & [y/x]<n \\ 0 & \text{其他} \end{cases}$$

程序1：

```
1  //exam3.26-1
2  #include<iostream>
3  #include<cmath>
4  using namespace std;
5  int main()
6  {
```

```
7       int n,x,y,t,rest;
8       cin>>n>>x>>y;
9       t=ceil((double)y/x); //将y强制转换为实数求y除x的值后
                               向上取整
10      if (t<n) rest=n-t;
11      elserest=0;
12      cout<<rest<<endl;
13      return 0;
14  }
```

程序2：

```
1   //exam3.26-2
2   #include<iostream>
3   using namespace std;
4   int main()
5   {
6       int n,x,y,rest;
7       cin>>n>>x>>y;
8       if (y%x==0) rest=y/x>=n?0:n-y/x; //虫子完整吃完苹果
9       else rest=y/x>=n-1?0:n-1-y/x;    //虫子没吃完苹果
10      cout<<rest<<endl;
11      return 0;
12  }
```

说明：本题提供了对于取整问题的不同的程序表达方法。

思考：给出怎样不同的测试数据，能够说明时间和苹果被虫子吃可能出现的情况。测试上述程序的正确性？

【例3.27】在大学校园里，由于校区很大，没有自行车上课办事会很不方便。但实际上，并非去办任何事情都是骑车快，因为骑车总要找车、开锁、停车、锁车等，这要耽误一些时间。假设找到自行车、开锁并骑上自行车的时间为27秒，停车锁车的时间为23秒，步行每秒行走1.2米，骑车每秒行走3.0米。输入距离（单位：米），输出是骑车快还是走路快。分别用Walk、The same、Bike表示走路快、一样快、自行车快。

输入样例：

90

输出样例：

Walk

分析：设距离为 dis，则走路所需时间为：t1=dis/3+27+23，骑车所需时间为：t2= dis/1.2，那么比较结果为：

$$res=\begin{cases}\text{Walk} & t1>t2\\ \text{The same} & t1=t2\\ \text{Bike} & t1<t2\end{cases}$$

程序 1：

```
1  //exam3.27-1
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      double dis;
7      double t1,t2;
8      cin>>dis;
9      t1=dis/3+27+23;    // 计算走路所需时间
10     t2=dis/1.2;        // 计算骑车所需时间
11     if(t1>t2)          // 比较两时间
12     cout<<"Walk"<<endl;
13     else if(t1==t2)
14     cout<<"The same"<<endl;
15     else
16     cout<<"Bike"<<endl;
17     return 0;
18 }
```

运行结果：

```
12
Walk
```

```
120
Bike
```

```
100
Bike
```

说明：对于第 3 组数据，当距离为 100 时，计算结果 t1=250/3、t2=250/3，应该得到“The same”的结果，然而程序运行结果却是 Bike，为什么呢？因为程序第 9 行表达式中计算 dis/3 和程序第 10 行表达式中计算 dis/1.2 时，由于除不尽，浮点数的运算存在误差，导致计算出的 t1 不等

于t2，如何解决呢？对于实数运行的结果要特别注意误差问题，关于误差问题，根据实际情况可以有不同的解决方案，对于本问题，将t1和t2都乘以6，则t1= dis*2+(27+23)*6，t2= dis*5把问题转换为乘法运算，减少因为除不尽而产生的误差。

程序2：

```
1  //exam3.27-2
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      double dis;
7      double t1,t2;
8      cin>>dis;
9      t1=dis*2+(27+23)*6;        //计算走路所需时间
10     t2=dis*5;                  //计算骑车所需时间
11     if(t1>t2)
12     cout<<"Walk"<<endl;
13     else if(t1==t2)
14     cout<<"The same"<<endl;
15     else
16     cout<<"Bike"<<endl;
17     return 0;
18 }
```

运行结果：

```
100
The same
```

【例3.28】运输公司对所运货物实行分段计费。对于重量为w的货物，每千米每吨基本运费为p，折扣为d，运输里程为m，当里程处于不同里程阶段s时，折扣不同，如下表，每阶段运费f的计算公式为：f=p*w*s*(1-d)。设计程序，当输入p、w和m后，计算总运费f。

阶段里程s	折　扣
s<250	不打折扣
250<=s<500	2%折扣
500<=s<1000	5%折扣

续表

阶段里程s	折　扣
1000<=s<2000	8%折扣
2000<=s<3000	10%折扣
3000<=s	15%折扣

分析：根据题意，总运费为每一阶段里程运费之和。先找到里程m所处的折扣里程阶段，计算处于该折扣里程阶段的运费，然后，累加剩余的不同里程阶段的运费。如m=1250，所处的折扣里程阶段是8%折扣，在该里程阶段s=m-1000，f=p*w*s*(1-d)，使s=1000表示处于低一个里程阶段5%折扣，然后，累加f+=p*w*(s-500)*(1-d)，使s=500进入2%折扣里程阶段，累加f+=p*w*(s-250)*(1-d)，使s=250，进入不打折扣里程阶段，累加f+=p*w*s*(1-d)。

方法1：用if语句实现，程序如下：

```
1  //exam3.28-1
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      double s;
7      double p,w,d,f,m;
8      double f1;
9      cout<<" 输入运输单价 p, 重量 w 和里程 m:"<<endl;
10     cin>>p>>w>>m;
11     f=0;
12     s=0;
13     f1=p*w;
14     if(m<250)
15       f=f1*m;              // 里程小于 250 千米的运费
16     else
17     {
18       s+=f1*250;           // 里程大于 250 千米，先累计 250 千米
                                产生的运费
19       d=0.02;              // 设置下一个里程阶段的折扣
20         if(m<500)
21           f=s+f1*(1-d)*(m-250);    // 里程 250~500 千米的运费
22         else
```

```
23          {
24            s+=f1*250*(1-d);          // 里程大于 500 千米，先累计
                                           500 千米产生的运费
25            d=0.05;                   // 设置下一个里程阶段的折扣
26            if(m<1000)
27              f=s+f1*(1-d)*(m-500); // 里程 500~1000 千米的运费
28            else
29            {
30              s+=f1*(1-d)*500;        // 里程大于 1000 千米，先累计
                                           1000 千米产生的运费
31              d=0.08;                 // 设置下一个里程阶段的折扣
32              if(m<2000)
33                f=s+f1*(1-d)*(m-1000);         // 里程 1000~2000
                                                    千米的运费
34              else
35              {
36                  s+=f1*(1-d)*1000; // 里程大于 2000 千米，先累计
                                           2000 千米产生的运费
37                  d=0.1;              // 设置下一个里程阶段的折扣
38                  if(m<3000)
39                    f=s+f1*(1-d)*(m-2000);  // 里程 2000~3000
                                                    千米的运费
40                  else
41                  {
42                      s+=f1*(1-d)*1000;
                    // 里程大于 3000 千米，先累计 3000 千米产生的运费
43                      d=0.15;          // 设置下一个里程阶段的折扣
44                      f=s+f1*(1-d)*(m-3000);
                    // 里程大于 3000 千米的运费
45                  }
46              }
47            }
48         }
49       }
50       cout<<" 折扣后总运费："<<f<<endl;
51       return 0;
52   }
```

方法 2：本问题是一个多阶段问题，可以用switch语句实现。分析阶段里程表，可以发现是以250倍数分隔的，设c=m/250作为switch语句表

达式，当c值在0-11范围时对应于3000千米以内不同的里程阶段，需要特别注意的是，当c值不在0~11范围时，即为大于3000千米，作为default处理，因此，对于本问题default放在switch语句最上面。

程序如下：

```
1  //exam3.28-2
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      int c,s;
7      double p,w,d,f,m;
8      cout<<"输入运输单价p，重量w和里程m:"<<endl;
9      cin>>p>>w>>m;
10     f=0;
11     c=m/250;
12     s=m;
13     switch(c)
14     {
15       default:d=0.15;f+=p*w*(s-3000)*(1-d);s=3000;
16       case 8:case 9:case 10:case 11:d=0.1;f+=p*w*
         (s-2000)*(1-d);s=2000;
17       case 4:case 5:case 6:case 7:d=0.08;f+=p*w*(s-1000)*
         (1-d);s=1000;
18       case 2: case 3: d=0.05;f+=p*w*(s-500)*(1-d);s=500;
19       case 1: d=0.02;f+=p*w*(s-250)*(1-d);s=250;
20       case 0: d=0;f+=p*w*s*(1-d);
21     }
22     cout<<"折扣后总运费："<<f<<endl;
23     return 0;
24 }
```

说明：本题提供了利用case语句后不加break语句实现一类问题解决的技巧。

【例3.29】小计知道一个“先有鸡还是先有蛋的答案”，他和n个人说了答案，不过，对其中的x个人故意告诉了错误的答案。然后，有一个人问了这n个人问题的答案，有m个人说先有蛋，n-m个人说先有鸡，已知其中有y个人故意说了小计告诉他们的相反的答案。现在给n、m、x、y，

问是否能推测出答案，或者多解，或者无解。如果答案是鸡，输出“The chicken”；如果答案是蛋，输出“The egg”；如果两个答案都满足条件，输出“Ambiguous”；如果都不满足，输出“The oracle is a lie”。

输入样例：

10 10 0 0

输出样例：

The egg

输入样例：

60 40 0 30

输出样例：

The oracle is a lie

输入样例：

60 20 5 25

输出样例：

The chicken

输入样例：

20 10 5 5

输出样例：

Ambiguous

分析：根据题意，对于给定的数据n、m、x、y，我们需要推测出是先有鸡还是先有蛋、多解或无解，从已知数据推出答案有一定的难度。我们试着反过来分析，因为答案只有4种，不妨先假设是其中的一个答案，然后根据已知数据判断该答案是否正确。这种解决问题的方法称为判定性问题求解法。

假设答案是鸡，根据数据推理如下：

（1）根据题意，n个人中，n-x人被告知是鸡，有x人被告知是蛋。

（2）这n个人说出答案的过程，有y个人说出的答案与其被告知的相反，假设鸡说成蛋的为a人，蛋说成鸡的为b人，那么：a+b=y（式一）。

（3）说出蛋的人数m=被告知是蛋的人数x+被告知是鸡的人数中“叛变”的人数a-被告知是蛋的人中“叛变”的人数b，即：m=x-b+a（式二）。

（4）联立式一和式二得：a=(y+m-x)/2，b=(y-m+x)/2。

（5）若a和b都为正整数（若a为整数，b也一定是整数），且0<=a<=n-x、0<=b<=x，则说明“答案是鸡”的假设是正确。

（6）上述条件表达为：

(y+m-x)%2==0&&n-x>=(y+m-x)/2&&
(y+m-x)/2>=0&&x>=(y-m+x)/2&&(y-m+x)/2>=0

（7）满足上述条件，只能证明“答案是鸡”的假设是正确，但不能证明其他假设是错误的。

同理，假设“答案是蛋”，成立的条件为：

0<=(y+n-m-x)/2&&(y+n-m-x)/2<=n-x&&(m+x+y-n)/2>=0&&(m+x+y-n)/2<=x&&(m+x+y-n)%2==0

程序如下：

```
1  //exam3.29
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int x,y,m,n;
7      bool chicken,egg;          //用于标识是鸡还是蛋
8      cin>>n>>m>>x>>y;
9      chicken=0;                 //设初值不是鸡
10     egg=0;                     //设初值不是蛋
11     if(n-x>=(y+m-x)/2&&(y+m-x)/2>=0&&x>=(y-m+x)/2&&
       (y-m+x)/2>=0&&(y+m-x)%2==0)
12     chicken=1;                 //“答案是鸡”的假设正确
13     if(0<=(y+n-m-x)/2&&(y+n-m-x)/2<=n-x&&
         (m+x+y-n)/2>=0&&(m+x+y-n)/2<=x&&(m+x+y-n)%2==0)
14     egg=1;                     //“答案是蛋”的假设正确
15     if(chicken==0&&egg==0) cout<<"The oracle is a lie";
16     if(chicken==0&&egg==1) cout<<"The egg";
17     if(chicken==1&&egg==0) cout<<"The chicken";
18     if(chicken==1&&egg==1) cout<<"Ambiguous";
19     cout<<endl;
20     return 0;
21 }
```

说明：虽然本题的分析过程比较难，但提供了一种逻辑推理的方法。

练　习

（1）某市的IC卡电话计费标准如下：首次为0.5元/3分钟（不足3分钟按3分钟计费），之后是0.2元/1分钟。已知某人打一次电话花费为x元，问这个人有可能打了多长时间的电话？（精确到分钟）

（2）编写程序，计算下列分段函数y = f(x)的值。

$$f(x)=\begin{cases} x+2.5; & 0 <= x < 5 \\ 2-1.5(x-3)(x-3); & 5 <= x < 10 \\ x/2-1.5; & 10 <= x < 20 \end{cases}$$

（3）输入一个不多于四位的正整数，求出它是几位数，并分别打印出各位上的数字。

（4）企业发放的奖金根据利润分段计算提成。当利润I低于或等于10万元的，奖金可提10%；利润高于10万元，低于20万元，低于10万元的部分按10%提成，高于10万元的部分，可提成7.5%；20万到40万的，低于20万的部分仍按上述办法提成（下同）。高于20万元的部分按5%提成；40万到60万时，高于40万的部分按3%提成；60万到100万时，高于60万的部分按1.5%提成；I>=100万元时，超过100万元的部分按1%提成。输入当月利润I，求应发奖金总数。

第4章　程序段的反复执行

生活中经常会遇到一些重复性的工作。例如将书包里的十本书，摆放到书架的指定位置上。要做的工作就是重复十次：拿起书——找位置——摆好书。如果用程序的方式描述这个工作，那就是十个重复的程序段。可是，真的要写出十段同样的程序吗？如果十段还不算麻烦，那么，一百段呢？N段呢？

事实上，反复执行多次同样的操作，就是循环的思想。应用循环思想编写的程序，就是循环结构程序。

在C++语言中，为表现循环思想，提供了for、while、do-while三种不同格式的循环语句。而上述提到的所有重复的内容，就是循环语句的循环体。

4.1　for语句

【例4.1】对于给定的任意正整数n，输出1~n的平方数。

分析：对于任意正整数i，其对应的平方数就是i*i。解决本题，就要做n次重复操作：i的取值从1变化到n，对于i的每一个取值，输出其平方数。

程序如下：

```
1  //exam4.1
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int i,n;
7      cin>>n;
8      for(i=1;i<=n;i++)          // 重复操作
9        cout<<i*i<<" ";          // 输出 i 的平方数
10     return 0;
11 }
```

运行结果：

```
20
1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361 400
```

程序中第 8 行使用 for(i=1;i<=n;i++) 语句，控制第 9 行的输出语句“cout<<i*i<<"";”执行 n 次，避免了重复书写 n 段相同程序段的问题。

那么，for 语句的使用规则如何？什么样的重复操作适合使用 for 语句？for 语句又是如何解决重复操作的呢？

本节，我们就将学习 C++ 语言的 for 语句。

4.1.1　for 语句的格式与功能

1. 格式

格式 1：

```
for(循环变量初始化；循环条件；循环变量增量)
    语句;
```

格式 2：

```
for(循环变量初始化；循环条件；循环变量增量)
{
    语句 1;
    语句 2;
    ……
}
```

无论是格式 1 中的“语句”还是格式 2 中的“{语句 1；语句 2；……；}”，都是要重复执行的内容，即 for 语句的循环体。

2. 功能

对于使循环条件成立的每一个循环变量的取值，都要执行一次循环体。

for 循环语句的执行流程如图 4.1 所示。

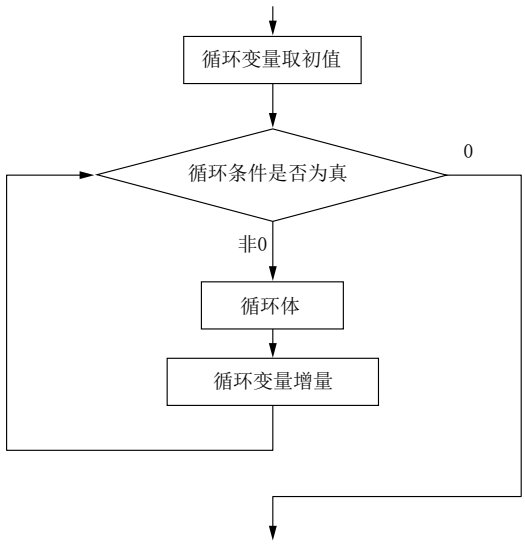


图4.1 for循环语句的执行流程

【例4.2】写出下面程序的运行结果。

```
1  //exam4.2
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6       int i;
7       for (i=1; i<=5; i++)
8         cout<<i;
9       return 0;
10 }
```

运行结果：

```
12345
```

说明：循环初始化的作用是使循环变量获得初值；循环条件一般用来约束循环变量的范围，当循环变量超出范围的时候，这个循环就会结束；而循环变量增量则是计算循环变量的语句。

程序中第7行的“i=1”使循环变量i取得初值1；“i<=5”约束了循环变量i的取值范围不能超过5；“i++”决定了循环变量i的增量是每次递增1。

因此，程序运行时，变量i的取值以及循环体的执行情况如下表所示。

续表

循环变量i的取值	循环体的执行情况
1	输出1
2	输出2
3	输出3
4	输出4
5	输出5
6	退出循环

【*例 4.3】写出下面程序的运行结果。

```
1  //exam4.3
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int i,j;
7      for(i=0,j=10;i<j;i++,j--)
8          cout<<i<<" "<<j<<endl;
9      return 0;
10 }
```

运行结果：

```
0 10
1 9
2 8
3 7
4 6
```

说明：在循环变量的初始化部分和增量部分，可以使用逗号分隔的语句序列，来进行多个动作。

程序的第 7 行使用逗号分隔的语句序列，同时确定了两个循环变量 i 和 j。在初始化部分，i 和 j 分别被初始化为 0 和 10；在增量部分，分别设定了 i 和 j 的变化情况为增 1 和减 1。

因此，程序运行时，变量i、j的取值以及循环体的执行情况如下表所示。

循环变量i的取值	循环变量j的取值	循环体的执行情况
0	10	输出0 10
1	9	输出1 9
2	8	输出2 8

续表

循环变量i的取值	循环变量j的取值	循环体的执行情况
3	7	输出3 7
4	6	输出4 6
5	5	退出循环

【*例4.4】写出下面程序的运行结果。

```
1  //exam4.4
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int i,sum;
7      sum=0;
8      for(;i<=100;)
9      {
10         sum+=i;
11         i++;
12     }
13     cout<<i<<" "<<sum<<endl;
14     return 0;
15 }
```

运行结果：

```
101 5050
```

说明： for语句中的初始化、条件和增量，可部分或全部省略，但两个分号不能省略。

在程序第8行的for语句中省略了循环变量的初始化i=0（或i=1）和增量部分i++。因此，程序的功能是在求前100个自然数之和。

实验： 将程序第8行的for语句分别改写成如下写法，并对比输出结果。

（1）for(i=0; i<=100; i++)。

（2）for(i=1; i<=100; i++)。

4.1.2 for循环语句的应用

【例4.5】输出100以内的所有偶数。

方法 1：从题目看，很明显是要重复输出 100 以内的 50 个偶数。循环变量的初值、终值和增量都很清楚，所以，适合应用 for 循环语句来解决问题。

我们可以想到对于 1~100 之间的 100 个数字 i，直接重复进行判断：如果 i 是偶数，则输出 i 的值。

程序如下：

```
1  //exam4.5-1
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int i;
7      for(i=1;i<=100;i++)            // 对于 i 取得 1 至 100 之间的
                                         每一个整数，都重复操作
8        if (i%2==0) cout<<i<<" ";   // 如果 i 为偶数，则输出 i 的值
9      return 0;
10 }
```

运行结果：

```
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70
 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100
```

程序中，将 1~100 之间的所有数字都列举出来，然后一一判断，符合偶数条件的，就输出。这种思想，本质上是穷举。穷举法保证在求解的过程中，所有可能解都会判断到，不会丢解。当然缺点就是有时候效率不高。关于穷举法，在后续学习中，还会有所接触。

方法 2：在上述分析的基础上，再进一步分析：我们都知道，相邻偶数之间的差值为 2，所以，我们还可以设置变量的初值为 2、增量为 2 的 for 循环，使得循环次数减少为 50 次。

程序如下：

```
1  //exam4.5-2
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int i;
7      for(i=2;i<=100;i+=2)       // 对于 i 取得 2 至 100 之间的每一
                                     个偶数，都重复操作
```

```
8       cout<<i<<" ";              // 输出 i 的值
9       return 0;
10 }
```

运行结果：

```
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70
 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100
```

对比上述两个程序，不难发现，其不同之处在于第7、8行：

（1）对于同一问题，可以设置不同的循环变量初值、条件、增量。

（2）不同的for语句设置，循环体的内容和执行次数也会有所不同。

思考：解决这个问题，还可以设置其他的初值、条件、增量吗？

【例4.6】分别计算1~100中偶数和奇数之和。

方法1：根据例4.5的分析，我们很容易找到所有的偶数和奇数，继而计算其和，也并非难事。但是，如何使用程序来计算和呢？

这里，我们引进一个累加的概念。假设用变量sum1和sum2分别存放偶数与奇数和，累加就是在sum1或sum2的基础上，加上一个数字，改变累加变量的值；再加上一个数字，改变累加变量值；……；如此重复下去。这就如同我们向储蓄罐里存钱，每存一次，储蓄罐里的钱数就在原有基础上发生一次变化。

下面，就在程序exam4.5-1的基础上，稍微改变，来完成累加操作。

程序如下：

```
1  //exam4.6-1
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6       int i,sum1=0,sum2=0;      //sum1、sum2 分别存放偶数和、奇
                                    数和，均初始化为 0
7       for(i=1;i<=100;i++)       // 对于 i 取得 1 至 100 之间的每一
                                    个整数，都重复操作
8         if(i%2==0)  sum1+=i;    // 偶数累加到 sum1 中
9         else sum2+=i;           // 奇数累加到 sum2 中
10      cout<<" 偶数和 "<<sum1<<"  "<<" 奇数和 "<<sum2;
11      return 0;
12 }
```

运行结果：

```
偶数和2550  奇数和2500
```

实验：观察下面的程序，它与exam4.6-1程序有何区别？上机实验一下，它的运行结果如何？

```
#include<iostream>
using namespace std;
int main()
{
  int sum1=0,sum2=0;
  for(int i=1;i<=100;i++)
    if (i%2==0) sum1+=i;
    else sum2+=i;
  cout<<" 偶数和 "<<sum1<<"  "<<" 奇数和 "<<sum2;
  return 0;
}
```

*** 方法 2：**参考例 4.3，for 语句的循环变量初始化和循环变量增量两部分都可以使用逗号语句序列。那么我们可否同时产生偶数和奇数呢？

偶数从 2 开始每次递增 2，奇数从 1 开始，每次递增 2。这个规律很明确，也很适合同时写在 for 语句中控制循环变量。

程序如下：

```
//exam4.6-2
#include<iostream>
using namespace std;
int main()
{
  int i,j,sum1=0,sum2=0;
  for(i=2,j=1;i<=100;i+=2,j+=2)     // 同时生成偶数和奇数
  {
      sum1+=i;                      // 偶数 i 累加入 sum1
      sum2+=j;                      // 奇数 j 累加入 sum2
  }
  cout<<" 偶数和 "<<sum1<<"  "<<" 奇数和 "<<sum2;
  return 0;
}
```

运行结果：

```
偶数和2550  奇数和2500
```

实验：由于题目的特殊性，1~100之间的整数，不是偶数，就是奇数，所以，还可以用如下方法，完成计算：

第一步，计算1~100之间的所有整数和S。

第二步，计算1~100之间的所有偶数和S1。

第三步，分别输出S1和S-S1的值。

根据上述分析，编写程序并上机实验。

*本题还可以直接使用等差数列的求和公式，分别计算出1~100之间偶数和奇数之和。这种方法，留给同学们进一步探究实践。

所谓**等差数列**，就是数列中任意相邻两项的差值都相等，这个差值称为公差。

一个以a1为首项，公差为d的等差数列，各项数值为a1,a1+d,a1+2d,a1+3d,……

这个数列的第n项为a1+(n-1)d；前n项的和为na1+n(n-1)d/2。

【例4.7】已知n个人的身高值，求出其中的最大值。

分析：我们联想一下日常排队找排头的方式：假设一个最高者作为排头，然后让其他人和他比身高，如果有人比当前排头还高，就取代当前排头成为新排头；后面的人继续和排头比身高，如果比当前排头还高，就成为新排头，……重复比身高，直到全部人都比较、调整过。这时的排头，一定最高。

类比这种思想，我们可以先假设max存放了身高最大值（这里可以将其初始化为0）。读入第1个人的身高值，若比max大，则将其值存入max；继续读入第2个人的身高，若比max大，则将其值存入max，……直到读入第n个人的身高值，若比max大，则将其值存入max。不难看出，这个过程就是一个循环的过程。

程序如下：

```
1  //exam4.7
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6    int i,n;
7    float x,max=0;          //将身高最大值初始化为0
8    cin>>n;
```

```
9     for(i=1;i<=n;i++)
10      {
11        cin>>x;
12        if(x>max) max=x;          //判断身高是否比当前最高者高
13      }
14    cout<<max<<endl;
15    return 0;
16 }
```

运行结果：

```
5
1.79 1.75 1.69 1.80 1.78
1.8
```

思考：程序第 7 行中，max=0 的作用是什么？

实验：

（1）如果将程序中 max 初始化为第 1 个人的身高值，可以吗？调整 max 的初始值，自己编程求最大身高值。

（2）结合例题和思考结果，编程求最小身高值。

【例 4.8】Fibonacci 数列是一个特殊的数列：数列的第一项和第二项分别为 0 和 1，从第三项开始，每一项是其前面两项之和。即 0，1，1，2，3，5，8，……请编程输出该数列的前 40 项 (每十项一行，每两项之间用空格分隔)。

分析：根据题目描述，对于数列中的第 i（i ≥ 3）项 c，可以表示为其前面两项 a 和 b 之和，即 c=a+b。

由于题目只要求输出，我们就不必将数列的前 40 项全部存放下来，只需要边计算边输出即可。

（1）对于 a、b 分别初始化为 0、1，并输出它们的值。

（2）计算第三项 c，令 c=a+b，输出第三项的值。

（3）计算第四项。此时，它的前两项分别是当前的 b 和 c，而当前 a 的值不再需要被保存。为保持描述的连贯性，我们不妨做以下更新：

```
a=b;
b=c;
```

这样，第四项的值，又可以表示为 c=a+b，输出 c 值即可。

（4）求解其后各项的值，都可以用类似（3）的方法，先更新 a 和 b 的值，计算 c=a+b，然后输出 c 值。

分析至此，我们可以发现：解决本题，从第三项开始用到了重复操作，

即计算c=a+b并输出，更新a和b的值。

程序如下：

```
1  //exam 4.8
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6    int i,a=0,b=1,c;
7    cout<<a<<" "<<b;              //输出第一项和第二项的值
8    for(i=3;i<=40;i++)            //求解第三项至第四十项的值
9    {
10     c=a+b;                      //求第i项的值
11     cout<<c<<" ";
12     if(i%10==0)cout<<endl;      //每十个数据一行
13     a=b;b=c;                    //更新a和b的值
14   }
15   return 0;
16 }
```

运行结果：

```
0 1 1 2 3 5 8 13 21 34
55 89 144 233 377 610 987 1597 2584 4181
6765 10946 17711 28657 46368 75025 121393 196418 317811 514229
832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986
```

*程序中，用了重复更新a、b值的方法来求得数列每一项的值。这种思想，本质是一种迭代。关于迭代，留在后续学习中深入研究。

思考：程序中a、b、c三者的关系如何，例如i=10时，这三个变量的值各是多少？

实验：

（1）程序中，a、b、c的数据类型均定义为int类型，那么这个程序可以正确输出Fibonacci数列的多少项呢？

（2）是否可以用两个变量求解该问题，如果可以，编程实现之。

练　习

（1）读程序，写结果。

```
//test(1)-1
#include<iostream>
using namespace std;
int main()
{
     int i,n=0;
     for(i=1;i<=100;i=i*3)
       if(i %7==0) n++;
     cout<<n<<endl;
     return 0;
}

//*test(1)-2
#include<iostream>
using namespace std;
int main()
{
     int i,j;
     for(i=20,j=0;i<=50;i++,j=j+5)
       if(i==j) cout<<i<<endl;
     return 0;
}

//test(1)-3
#include<iostream>
using namespace std;
int main()
{
   int n;
   for(n=1;n<=5;n++)
     switch(n%5)
     {
          case 0:cout<<n<<"@"<<endl;break;
          case 1:cout<<n<<"#"<<endl;break;
          default:cout<<n<<endl;
          case 2:cout<<n<<"$"<<endl;
     }
   return 0;
}
```

```
//test(1)-4
#include<iostream>
using namespace std;
int main()
{
     int n,i;
     for(i=1;i<=100;i++)
     {
          n=i;
          if(++n%2==0)
          if(++n%3==0)
          if(++n%7==0)
          cout<<i<<":"<<n<<endl;
     }
     return 0;
}
```

（2）根据下述表达式，计算并输出S的结果。

（A）S=1+2-3+4-5+6-……+100

（B）S=1+(1+2)+(1+2+3)+……+(1+2+3+……+10)

（C）S=1/1-1/2+1/3-1/4+1/5-1/6+……-1/100

（3）[1]班上有学生若干名，给出每名学生的年龄（整数），求班上所有学生的平均年龄，保留到小数点后两位。

输入包括若干行,第1行有一个整数n（1≤n≤100），表示学生的人数，其后n行每行有一个整数，表示每个学生的年龄，取值为15到25。

输入样例：

2

18

17

输出样例：

17.50

（4）[2]给定一个长度为n的非负整数序列，请计算序列的最大跨度值（最大跨度值 = 最大值减去最小值）。

输入一共两行，第1行为序列的个数n（1 <= n <= 1000），第2行为

1）本题选自NOI题库，题号1716。

2）本题选自NOI题库，题号8182。

序列的n个不超过1000的非负整数，整数之间以一个空格分隔。

输入样例：

6

3 0 8 7 5 9

输出样例：

9

（5）相传古印度宰相达依尔是国际象棋的发明者。有一次，国王因为他的贡献要奖励他，问他想要什么。达依尔说："只要在国际象棋棋盘的64个格子里摆上麦子：第1格一粒，第2格两粒，……，后面一格的麦子总是前一格麦子数的两倍，摆满整个棋盘，我就感恩不尽了。"国王一想，这还不容易。于是令人扛来一袋麦子，可很快用完了，又扛来一袋，还是很快用完了……国王对此很奇怪。

现在，请你编程计算所需要麦子的体积（1立方米的麦子约为1.42×10^8粒）。

（6）对于任意给定的一个正整数，计算其因数个数。

输入样例：

6

输出样例：

4

说明：1、2、3、6都是6的因数。因此，输出4。

（7）[1]2008年北京奥运会，A国的运动员参与了n天的决赛项目(1≤n≤17)，现在要统计一下A国所获得的金、银、铜牌数目及总奖牌数。

输入n＋1行，第1行是A国参与决赛项目的天数n，其后n行，每一行是该国某一天获得的金、银、铜牌数目，以一个空格分开。

输出仅1行，包括4个整数，为A国所获得的金、银、铜牌总数及总奖牌数，以一个空格分开。

输入样例：

3

1 0 3

3 1 0

0 3 0

输出样例：

1）本题选自NOI题库，题号6174。

4 4 3 11

（8）在一次运动会方队表演中，学校安排了十名老师进行打分。对于给定的每个参赛班级的不同打分（百分制整数），按照去掉一个最高分、去掉一个最低分，再算出平均分的方法，得到该班级的最后得分。

输入样例：

90 89 92 90 93 95 88 90 89 88

输出样例：

90.13

（9）[1)] 陶陶家的院子里有一棵苹果树，每到秋天树上就会结出10个苹果。苹果成熟的时候，陶陶就会跑去摘苹果。陶陶有个30cm高的板凳，当她不能直接用手摘到苹果的时候，就会踩到板凳上再试试。

现在已知10个苹果到地面的高度，以及陶陶把手伸直的时候能够达到的最大高度，请帮陶陶算一下她能够摘到的苹果的数目。假设她碰到苹果，苹果就会掉下来。

输入两行数据：第1行只包括一个100到120之间（包含100和120）的整数（以厘米为单位），表示陶陶把手伸直的时候能够达到的最大高度；第2行包含10个100到200之间（包括100和200）的整数（以厘米为单位），分别表示10个苹果到地面的高度，两个相邻的整数之间用一个空格隔开。

输出格式：1行，只包含一个整数，表示陶陶能够摘到的苹果的数目。

输入样例：

110

100 200 150 140 129 134 167 198 200 111

输出样例：

5

4.2 while语句

【例4.9】考试结束后，老师想计算全体学生的平均分，你能帮助老师吗？现在无法知道参考人数，但是知道参加考试的人都不是0分。所以，提供给你的若干个考试成绩，以0作为计算的结束标志。

分析：计算平均分，需要知道总分和总人数。所以，本问题转化为先

1）本题改编自NOIP2005普及组试题。

计算总分和总人数。我们用变量tot存放总分，pep来存放人数。现在要做的就是：每读入一个非0的成绩，就将其加入tot，并且将pep加1。这个读成绩、累加、计数的过程一直重复到读入的成绩为0为止。

程序如下：

```
1  //exam4.9
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      float score,tot=0; // 总分 tot 初始化为 0
7      int pep=0;         // 计数器 pep 初始化为 0
8      cin>>score;
9      while(score!=0)    // 当成绩非 0，就重复执行
10     {
11           pep++;
12           tot+=score;
13           cin>>score;
14     }
15     cout<<tot/pep<<endl;
16     return 0;
17 }
```

运行结果：

```
90 0
90
90 90 0
90
90 89 90 0
89.6667
```

程序中，使用while语句（第9行）完成了重复操作的任务。while语句的使用规则如何？while语句是怎样解决重复操作的呢？while语句与for语句又有哪些区别呢？

带着这些问题，本节，我们就将学习C++语言的while语句。

4.2.1　while语句的格式与功能

1. 格式

格式1：

```
while(表达式)
  语句;
```

格式2：

```
while(表达式)
{
  语句1;
  语句2;
  ……
}
```

2. 功能

当表达式的值非0时，不断地执行循环体中的语句。所以，用while循环语句实现的循环被称为“当型循环”。

while循环语句的执行过程见图4.2。

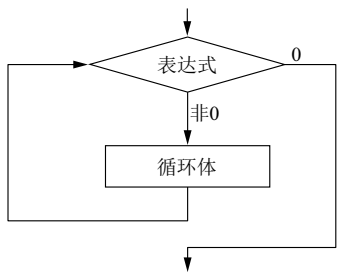


图4.2 while循环语句的执行过程

【例4.10】写出下面程序的运行结果。

```
1  //exam4.10
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6       int i=1;
7       while(i<=5)
8       {
9            cout<<i;
10           i++;
11      }
12      return 0;
13 }
```

运行结果：

```
12345
```

说明：只有表达式的值非 0，循环体才被执行。首次执行 while 的表示式也不例外（若表达式最初的值为 0，则根本不执行循环体）。

根据上述说明，程序的第 6 行，对 i 进行初始化为 1，判断第 7 行的条件 i<=5，表达式成立，则执行循环体：输出 i 的值，i 自增，再判断，输出，i 自增，……

我们可以用下表模拟一下程序的执行过程。

i的取值	表达式i<=5的结果	循环体的执行情况
1	非0	输出1；i更新为2
2	非0	输出2；i更新为3
3	非0	输出3；i更新为4
4	非0	输出4；i更新为5
5	非0	输出5；i更新为6
6	0	退出循环

思考：对比本例与上一节中用 for 语句输出 1 至 5 之间的所有整数的程序段，二者在程序功能上是一致的，但是在描述上却有所不同。你能说出用 while 语句替换 for 语句的一般规律吗？

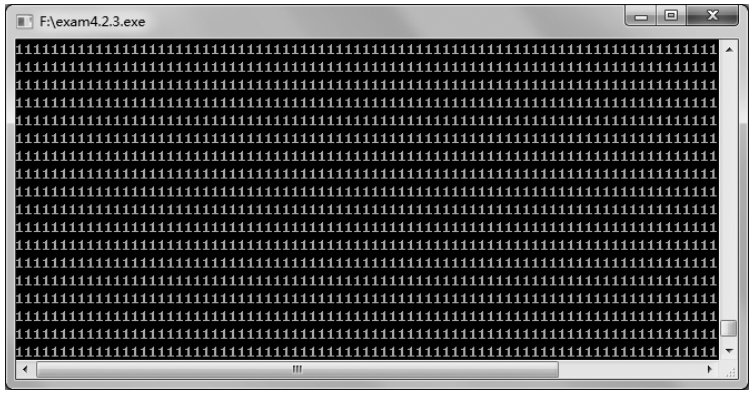
实验：将程序第 6 行（i 的初始化语句）修改成如下写法，上机实验，程序的运行结果将有什么变化？

```
int i=6;
```

【例 4.11】写出下面程序的运行结果。

```
1  //exam4.11
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int i=1;
7      while (i<=5)
8      {
9          cout<<i;
10     }
11     return 0;
12 }
```

运行结果：



说明：为了使循环能终止，循环体中一定要有影响表达式值的操作，否则该循环是一个死循环。

相对于上例，本程序的循环体中少了“i++;”语句，i的值始终不变，所以，while表达式的值始终非0，程序就一直输出i的值1，无法停止。

我们可以用下表模拟一下程序的执行过程。

i的取值	表达式i<=5的结果	循环体的执行情况
1	非0	输出1
1	非0	输出1
1	非0	输出1
1	非0	输出1
1	非0	输出1
1	非0	输出1
……	……	……

4.2.2 while循环语句的应用

【例4.12】在银行取款时，我们需要输入密码（由六位数字组成）。密码正确，才可以进行取款操作；若连续三次输入密码错误，就会冻结账号。现在，请你编写一个程序，模拟输入密码的过程。

输入格式：

每次输入六位数字

输出格式：

给出提示信息：正确、错误、冻结

分析：对于取款用户来说，做的就是反复输入六位数字的操作。每输入一次密码，可能有三种情况：

（1）输入错误，但输入不超过三次，输出“错误”。

（2）错误输入超过三次，退出循环，输出“冻结”。

（3）输入未超过三次，密码正确，退出循环，输出“正确”。

综上，我们可以设置循环的条件为：输入次数不超过三次且输入不正确。

程序如下：

```
1  //exam 4.12
2  #include <iostream>
3  using namespace std;
4  int main( )
5  {
6      int mima=201611;          // 预设一个密码
7      int x=0,n=0;              //x 接受密码，n 统计输入次数
8      while( n<3&&x!=mima) // 重复操作
9      { n++;
10       cin>>x;
11       if(x!=mima) cout<<" 错误 "<<endl;     // 提示错误
12     }
13     if(x==mima) cout<<" 正确 "<<endl;       // 提示正确
14     else if (n==3) cout<<" 冻结 "<<endl;    // 提示冻结
15     return 0;
16 }
```

运行结果：



思考：程序第 7 行的 x、n 初始化部分可以省略吗？与 while 的条件式有关系吗？

实验：将程序中的第 13、14 行改成如下写法，输出结果会怎样？

```
if(n==3)cout<<" 冻结 "<<endl;
else if(x==mima)cout<<" 正确 "<<endl;
```

【例 4.13】判断给定正整数 n（保证在正整数范围内）是否为质数，是，则输出 Yes，否则，输出 No。

分析：根据数学知识，质数是这样约定的：除了1和它本身不再有其他约数的数，就是质数。

因此，我们可以从除数为2开始试除，除数没有超过n-1并且没有出现整除，就将除数加1，反复试除。在重复的过程中，一旦出现整除现象，就说明n非质数；如果直到除数超过了n-1，也没有出现整除现象，那么n一定是质数。

不难看出，要用循环结构来解决问题，只是重复的次数未知。但是，重复进行的条件可以确定下来：那就是没出现整除并且除数未超过n-1。所以，可以用while语句来解决问题。

程序如下：

```
1  //exam4.13
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int n,i;
7      cin>>n;
8      i=2;
9      while(n%i!=0 && i<=n-1)  // 当n不能被i整除且i未超过
                                   n-1时，就重复操作
10       i++;
11     if(i>n-1)cout<<"Yes"<<endl; // 退出循环时，若i超过了n-1，
                                   则说明n是质数
12     else cout<<"No"<<endl;    // 退出循环时，若i未超过n-1，
                                   则说明n非质数
13     return 0;
14 }
```

运行结果：

```
100
No
137
Yes
```

思考：

（1）上述程序能解决大多数质数的判定。从严谨的角度考虑，程序中还需要加上哪些处理，才能正确判断任意正整数是否为质数？

（2）程序中，除数的上界设定为n-1。这个数字可否缩小？你能设计一个合适的数字吗？

实验：将程序的第 11、12 行改写成如下形式，运行结果将会怎样？

```
if(n%i==0) cout<<"No"<<endl;
else cout<<"Yes"<<endl;
```

【例 4.14】输入一个正整数，输出其位数。

分析：当输入的正整数不是一位数时，我们要用累计的方法完成位数统计：设定计数器 num，取出正整数的个位数字，num 加 1，从正整数中去掉个位数字，对剩余数位上数字所组成的新数重复计数。这个计数的过程是一个当型循环，可以用 while 语句来解决。

程序如下：

```
1  //exam4.14
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int n,num=1;        //num 存放数字的位数，初始化为 0
7      cin>>n;
8      while (n>10)        // 当 n 不是一位数，就重复操作
9      {
10         num++;
11         n/=10;          // 产生要去计数的新整数
12     }
13     cout<<num<<endl;    // 输出整数的位数
14     return 0;
15 }
```

运行结果：

```
5   123   1234567890
1   3     10
```

实验：对于任意正整数 n，n/10 的结果是什么？

在程序第 11 行后插入以下语句，程序的运行结果会怎样？根据结果，你能回答前面的问题了吗？

```
cout<<n<<" ";
```

【例 4.15】输入任意两个自然数，求它们的最大公约数。

方法 1：求任意两个自然数 a 和 b 的公约数，可以想到其最大的可能就

是两个数中的较小者min,最小的可能是1。所以，可以设最大公约数gcd从min开始进行判断，若gcd>1并且没有同时被a和b整除，就将gcd-1，重复判断是否整除。

程序如下：

```
1  //exam4.15-1
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      int a,b,gcd;
7      cin>>a>>b;
8      gcd=a>b?b:a;                    // 注意此处的特殊写法
9      while (gcd>1&& (a%gcd!=0 || b%gcd!=0))
                                       // 重复操作，寻找最大公约数
10       gcd--;
11     cout<<gcd<<endl;                // 输出最大公约数
12     return 0;
13 }
```

运行结果：

```
1 8
1
8 1
1
8 12
4
88 35
1
8 40
8
```

方法2：这里，我们来学习一个数学技巧——辗转相除法。

辗转相除法即欧几里德算法。对于任意两个自然数a和b，如果q和r是a除以b的商和余数，那么a和b的最大公约数等于b和r的最大公约数。

具体的求解过程描述如下：

（1）a ÷ b=q……r1。

（2）若r1=0，则a和b的最大公约数为b。

（3）若r1 ≠ 0，则继续做除法：b ÷ r1=q……r2。

（4）若r2=0，则a和b的最大公约数为r1。

（5）若r2 ≠ 0，则继续做除法：r1 ÷ r2=q……r3。

（6）如此重复下去，直到出现整除为止。余数为0时的除数就是a和b最大公约数。

根据辗转相除法的思想，求解a和b的最大公约数，就是一个重复做除法的过程，我们可以使用while语句来模拟上述思想。

程序如下：

```
1  //exam4.15-2
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6       int a,b,tmp;
7       cin>>a>>b;
8       while (tmp = a %b)          // 注意此处条件的特殊写法
9       {
10        a = b;
11        b = tmp;
12      }
13      cout<<b<<endl;              // 输出最大公约数
14      return 0;
15 }
```

运行结果：



实验：在使用循环解决实际问题时，具体使用哪一种语句，并没有硬性规定，往往与个人习惯和理解程度有关。

请尝试使用for语句解决求最大公约数问题，并对比不同语句的程序，与同学交流你的心得。

练　习

（1）读程序，写结果。

```
//test(1)-1
#include <iostream>
using namespace std;
int main( )
{
   int n,s=0;
   cin>>n;
   while(n)
   { s = s * 10 + n % 10;
     n /= 10;
```

```
    }
    cout<<s<<endl;
    return 0;
}
```

分别输入：（1）0　（2）1024　（3）1234567890

```
//test(1)-2
#include <iostream>
using namespace std;
int main( )
{
    int n;
    cin>>n;
    while(n!=0)
    {
      cout<<n%2;
      n/=2;
      }
    return 0;
}
```

分别输入：4　0

```
//test(1)-3
#include<iostream>
using namespace std;
int main()
{
    int i,n;
    cin>>n;
    i=n-1;
    while(i>1 && n%i!=0)
      i--;
    cout<<i;
    return 0;
}
```

分别输入：100　79　2

（2）对于任意输入的整数，计算其各个数位上的数字之和。

输入样例1：

1234

输出样例1：

10

输入样例2：

0

输出样例2：

0

（3）输入两个正整数m和n，判断m和n是否互质（即最大公约数为1），是，则输出Yes，否则，输出No。

输入样例：

36 55

输出样例：

No

（4）[1]请统计某个给定范围[L, R]的所有整数中，数字2出现的次数。

比如给定范围[2,22]，数字2在数2中出现了1次，在数12中出现1次，在数20中出现1次，在数21中出现1次，在数22中出现2次，所以数字2在该范围内一共出现了6次。

输入共1行，为两个正整数L和R，之间用一个空格隔开。

输出共1行，表示数字2出现的次数。

输入样例1：

2 22

输出样例1：

6

输入样例2：

2 100

输出样例2：

20

（5）[2]已知Sn＝1+1/2+1/3+……+1/n。显然对于任意一个整数k，当n足够大时，Sn大于k。现给出一个整数k(1≤k≤15)，计算出一个最小的n，使得Sn>k。

（6）在传递信息的过程中，为了加密，有时需要按一定规则将文本转换成密文发送出去。

1）本题选自NOIP2010普及组复赛试题。

2）本题选自NOIP2002普及组复赛试题。

有一种加密规则是这样的：

（A）对于字母字符，将其转换成其后的第 3 个字母。例如：A→D，a→d，X→A，x→a。

（B）对于非字母字符，保持不变。

现在，请你根据输入的一行字符，输出其对应的密码。

输入样例：

I(2016)love(08)China(15)!

输出样例：

L(2016)oryh(08)Fklqd(15)!

4.3 do-while 语句

【例 4.16】对于给定的自然数 n，求使 1+2+3+4+5+……+i ≥ n 成立的最小 i 值。

分析：这个题目的实质是对一个有规律的数值序列求和。但是，数值序列的个数不能直接确定，我们需要以序列和的值作为结束的条件。因此，我们能做的只是不断地累加，直到序列之和≥ n 为止。

程序如下：

```
1  //exam4.16
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int i=1,n,sum=0;
7      cin>>n;
8      do
9      {
10       sum+=i;
11       i++;
12     }
13     while (sum<n);              // 重复执行
14     cout<<i-1<<endl;
15     return 0;
16 }
```

运行结果：



累加求和的过程，至少要做一次加法。程序中使用do-while语句（第8、13行）解决了这个重复操作的过程。

do-while语句是C++语言中用于解决至少执行一次重复操作的循环语句。那么，如何使用do-while语句解决重复操作的问题？什么样的重复操作问题比较适合选择do-while语句呢？

本节，我们将学习C++语言的do-while语句。

4.3.1　do-while语句的格式与功能

1. 格式

格式1：

```
do
    语句;
while（条件表达式）;
```

格式2：

```
do
  {
      语句1;
      语句2;
      ……
  }
while（条件表达式）;
```

2. 功能

重复执行循环体，直到条件表达式的值为0。与while循环相比，do-while循环是先执行循环体，后判断条件表达式的当型循环。

do-while循环语句的执行过程用流程图表示见图4.3。

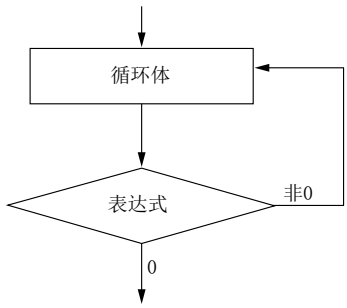


图4.3 do-while循环语句的执行过程

【例4.17】写出下面程序的运行结果。

```
//exam4.17-1
#include<iostream>
using namespace std;
int main()
{
    int i=1;
    do
    {
        cout<<i;
        i++;
    }
    while(i<1);
    return 0;
}
```

```
//exam4.17-2
#include<iostream>
using namespace std;
int main()
{
    int i=1;
    while(i<1)
    {
        cout<<i;
        i++;
    }
    return 0;
}
```

exam4.17-1运行结果：

```
1
--------------------------------
Process exited after 0.2561 seconds with return value 0
请按任意键继续. . .
```

exam4.17-2运行结果：

```
--------------------------------
Process exited after 0.4369 seconds with return value 0
请按任意键继续. . .
```

说明：do-while语句是在执行循环体之后检查条件表达式的值，所以至少要执行一次循环体。

回顾上节内容，while 语句是在进入循环体之前，就检查条件表达式的值，于是可能造成 while 语句的循环体一次也不被执行。

对照 exam4.17-1 和 exam4.17-2，虽然在进入循环之前 i 的值均为 1，并且循环的检查条件均为 i<1，但是前者使用了 do-while 语句，至少执行一次循环体，因此输出了一个 1；而后者使用了 while 语句，先检查条件（为 0），因此不执行循环体，也就没有任何输出。

思考： 如果希望使用 do-while 语句和 while 语句解决相同的循环问题，你能说出一般规律吗？

【例 4.18】 写出下面程序的运行结果。

```
1  //exam4.18-1
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6        int i=1;
7        do
8        {
9          cout<<i;
10         i++;
11       }
12       while(i<=5);
13       return 0;
14 }
```

```
1  //exam4.18-2
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6        int i=1;
7        do
8        {
9          i++;
10         cout<<i;
11       }
12       while (i<=5);
13       return 0;
14 }
```

exam4.18-1 运行结果：

```
12345
```

exam4.18-2 运行结果：

```
23456
```

说明： 在 do-while 语句的条件一样的情况下，循环体中语句的不同顺序，也将影响执行结果。

在上面两个程序中都使用了 do-while 语句来控制循环，但是循环体部分（第 9 行和第 10 行）的语句顺序不同。当 i ≤ 5 时，exam4.18-1 重复执行：先输出 i 的值，i 再自增 1；exam4.18-2 重复执行：是先自增 1，再输出 i 的值。

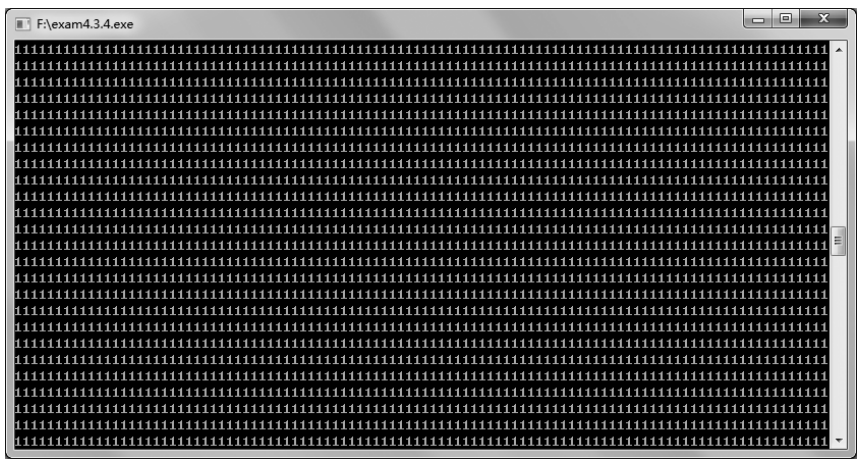
我们分别模拟一下两个程序的执行过程。

i的取值	exam4.18–1循环体的执行情况	exam4.18–2循环体的执行情况
1	输出1；i更新为2	i更新为2；输出2
2	输出2；i更新为3	i更新为3；输出3
3	输出3；i更新为4	i更新为4；输出4
4	输出4；i更新为5	i更新为5；输出5
5	输出5；i更新为6	i更新为6；输出6
6	退出循环	退出循环

【例4.19】写出下面程序的运行结果。

```
//exam4.19
#include<iostream>
using namespace std;
int main()
{
     int i=1;
     do
          cout<<i;
     while (i<=5);
     return 0;
}
```

运行结果：



说明：为了使重复能终止，循环体中一定要有影响条件表达式值的操作，否则该循环是一个死循环。

在这个程序中，循环体中没有改变i值的语句，i的值始终保持进入do-while语句之前获得的值1，而do-while语句的条件是i<=5。因此，这个条件表达式的值始终非0，循环体将无休止地运行下去，反复输出i的值1。

4.3.2　do-while语句的应用

【例4.20】针对计算机随机产生的两个三位数字，用户计算并输入其和，直到计算正确，输出所用的次数。

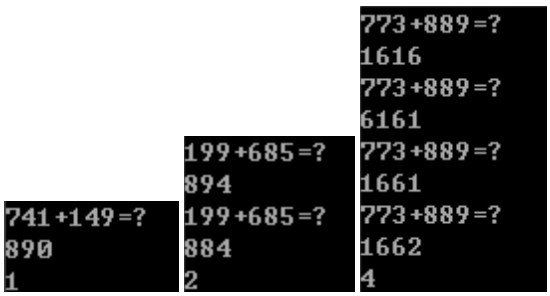
分析：根据题意，输入的数字不正确，就要重复输入。这是很明确的循环思想，而输入数字的不正确就是循环的条件。

因为至少要输入一个数字判断计算是否正确，符合do-while至少执行一次循环体的特征。我们使用do-while语句来编写程序。

程序如下：

```
1  //exam4.20
2  #include <iostream>
3  #include <cstdlib>
4  #include <ctime>
5  using namespace std;
6  int main()
7  {
8      int x,y,n,num=0;
9      srand(time(NULL));          //随机数种子
10     x=100+rand()%(999-100+1);   //产生第一个三位随机数
11     y=100+rand()%(999-100+1);   //产生第二个三位随机数
12     do
13     {
14       cout<<x<<"+"<<y<<"=?\n";
15       cin>>n;
16       num++;
17     }
18     while (n!=x+y);             //输入数字不对，重复操作
19     cout<<num<<endl;            //输出次数
20     return 0;
21 }
```

运行结果：



说明： srand() 是随机种子函数，rand() 是产生随机数函数。函数的具体用法可以查阅第 2 章。

实验： 将程序第 18 行 do-while 语句的条件改写成如下形式，并上机验证程序的正确性：

```
!(n==x+y)
```

【例 4.21】 对于给定的任意正整数，将其各个数位上的数字分离出来，并倒序输出。例如，输入 123，则应输出 321。

分析： 题目中给了一个暗示：从整数的低位开始输出。那么如何产生一个整数 n 的最低位呢？实践一下，234 % 10 的结果是多少？答案是 4。

事实上，n %10 正是产生 n 的最低位的方法。

解决本题，我们需要做的就是进行若干次取出个位数字、更新整数的操作（产生去掉个位数字后的新数，参考 exam4.14）。不同的整数，需要做 % 运算的次数也不同。那么，如何控制循环何时停止呢？当 n=0 时，就无需再做 % 运算了。

分析到此，本题可以应用 while 语句或 do-while 语句。到底用哪一个呢？对于只有一位数字的正整数，我们也做一次 % 运算，那就是至少执行一次循环体，我们选用 do-while 语句。

程序如下：

```
1  //exam4.21
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int n;
7      cin>>n;
8      do
```

```
9       {
10          cout<<n%10;     // 输出最末一位数字
11          n/=10;          // 产生去掉个位数字后的新数
12      }
13      while (n!=0);       // 商非 0，就重复操作
14      return 0;
15  }
```

运行结果：

```
0 5 123
0 5 3 2 1
```

实验：将程序第 13 行 while 的条件部分直接写成 n，即写成如下形式，程序的运行结果如何？

```
while(n)
```

【例 4.22】计算与任意输入的十进制正整数等值的二进制数共有几位数字。

分析：二进制是计算技术中广泛采用的一种数制。二进制数中只有 0 和 1 两个数码，也就是说它的每个数位上只有两个可能，非 0 即 1。二进制的基数为 2，进位的规则是“逢二进一”，借位的规则是“借一当二”。

将一个十进制整数转换为二进制数，通常采用“除二取余，逆序连接”的方法。将十进制的 13 转换成二进制的过程如图 4.4 所示。

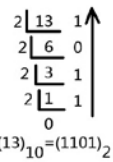


图 4.4　十进制整数转换为二进制

反之，将二进制整数转换为十进制，则采用“权值展开”的方法。

例如，将二进制的 1101 转换成十进制，具体步骤如下：

$$(1101)_2=(1*2^3+1*2^2+0*2^1+1*2^0)_{10}=(1*8+1*4+0*2+1*1)_{10}$$
$$=(8+4+0+1)_{10}=(13)_{10}$$

要解决本题，可以借助将十进制数转换成二进制数的思想，每一次除法产生一个二进制位，我们就可以进行一次位数计数。根据上述说明，转换的过程实质上就是一个在数字（或商）非 0 时，反复除以 2 的操作，我们可以使用 do-while 语句来实现。

程序如下：

```
1  //exam4.22
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6       int n,num=0;
7       cin>>n;
8       do
9       {
10        num++;              // 二进制位数加 1
11        n/=2;
12      }
13      while (n!=0);         // 当数字（商）非 0，就重复做除法
14      cout<<num;
15      return 0;
16 }
```

运行结果：

```
1  2  13  1024
1  2  4   11
```

思考： 在上述说明中，仅介绍了十进制正整数与二进制正整数互相转换的方法。你能想到小数的转换方法吗？

实验： 在程序上做哪些修改，可以输出一个十进制整数对应的二进制数形式？

练　习

（1）读程序，写结果。

```
//test(1)-1
#include<iostream>
using namespace std;
int main()
{
    int n;
    cin>>n;
```

```
    do
    {
        cout<<n%2;
        n/=2;
    }
    while(n!=0);
    return 0;
}
```

分别输入：4　0

```
//test(1)-2
#include<iostream>
using namespace std;
int main()
{
    int i,n;
    cin>>n;
    i=n-1;
    do
      i--;
    while(i>1 && n%i!=0);
    cout<<i;
    return 0;
}
```

分别输入：100　79　2

```
//test(1)-3
#include <iostream>
using namespace std;
int main()
{
  int i=1,s=3;
  do
  {
    s+=i++;
    if (s%7!=0)
    ++i;
  }
  while (s<15);
  cout<<i;
```

```
    return 0;
}
```

（2）圆周率的近似值PI可以用如下公式产生：

PI/4=1-1/3+1/5-1/7....

请你计算当某项的绝对值小于10^{-6}时，PI的近似值是多少？

（3）一小球从200m高处自由落下，每次落地后反跳回原高度的一半，然后再落下，再弹起，……一直到小球弹起的高度不足0.5m时，计算小球一共经过了多少路程？

（4）[1]给定一个整数，请将该数各个数位上的数字反转得到一个新数。新数也应满足整数的常见形式，即除非给定的原数为零，否则反转后得到的新数的最高位数字不应为零。

输入样例1：

123

输出样例1：

321

输入样例2：

-380

输出样例2：

-83

（5）[2]现代数学的著名证明之一是Georg Cantor证明了有理数是可枚举的。他是用下面这一张表来证明这一命题的：

1/1 1/2 1/3 1/4 1/5...

2/1 2/2 2/3 2/4...

3/1 3/2 3/3...

4/1 4/2...

5/1

我们以z字型给上表的每一项编号：第1项是1/1，后是1/2，2/1，3/1，2/2……输入正整数N，输出表中的第N项。

输入样例：

7

输出样例：

1）本题选自NOIP2011普及组复赛试题。

2）本题选自NOIP1999普及组复赛试题。

1/4

（6）角谷猜想又称冰雹猜想，它首先流传于美国，不久传到欧洲，后来由一位叫角谷的日本人带到亚洲，因此被称为角谷猜想。

通俗地讲，角谷猜想的内容是这样的：任意给定一个自然数n，当n是偶数时，将它除以2，即将它变成n/2；当n是奇数时，就将它变成3n+1，……，若干步后，总会得到1。

在上述演变过程中，将每一次出现的数字排列起来，就会出现一个数字序列。

我们现在要解决的问题是：对于给定的n，求出数字序列中第一次出现1的位置。

输入样例：

6

输出样例：

9

说明：数字的变化过程如下：

6→6÷2→3→3×3+1→10→10÷2→5→5×3+1→16→16÷2→8→8÷2→4→4÷2→2→2÷2→1

所形成的数字序列为：

6 3 10 5 16 8 4 2 1

1位于数字数列的第9个位置

4.4　多重循环

【例4.23】对于给定的自然数n，输出1~n之间的全部质数。

分析：要输出1~n之间的全部质数，就要判断每一个数是否为质数。而判断质数的问题，在例4.13中已经解决了。那么，本题就是重复进行n次的质数判断。

程序如下：

```
1  //exam4.23
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
```

```
6       int n,m,i;
7       cin>>n;
8       for(m=2;m<=n;m++)
9       {
10          i=2;
11          while (m%i!=0 && i<=m-1)
12              i++;
13          if(i>m-1) cout<<m<<" ";
14          }
15          return 0;
16 }
```

运行结果：

```
100
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

循环体内可以出现任何语句，当循环体内又出现循环语句时，就构成了多重循环（本程序第8行for语句的循环体中又出现了以第11行while语句开头的循环）。

本节，我们将学习应用多重循环解决实际问题的方法。

4.4.1 多重循环的使用说明

for、while、do-while语句的循环体都可以出现任何一种循环语句。例如，下表都是多重循环的语句形式。

for(……)
{
……
for(……)
{
……
}
……
}
	for(……)
	{
	……
	while(……)
	{
	……
	}
	……
	}
		for(……)
		{
		……
		do(……)
		{
		……
		}while(……);
		……
		}

while(……)
{
……
for(……)
{
……
}
……
}
	while(……)
	{
	……
	while(……)
	{
	……
	}
	……
	}
		while(……)
		{
		……
		do
		{
		……
		}while(……);
		……
		}

do
{
……
for(……)
{
……
}
……
}while(……);
	do
	{
	……
	while(……)
	{
	……
	}
	……
	}while(……);
		do
		{
		……
		do
		{
		……
		}while(……);
		……
		}while(……);

【例 4.24】写出下面程序的运行结果。

```
1  //exam4.24
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int i,j;
7      for(i=1;i<=5;i++)
8      {
9        j=5;
10       while(i<=j)
11       {
12         cout<<i*10+j<<" ";
13         j--;
14       }
15       cout<<endl;
16     }
17     return 0;
18   }
```

运行结果：

```
15 14 13 12 11
25 24 23 22
35 34 33
45 44
55
```

说明：对于多重循环而言，外层循环执行一次，内层循环将执行若干次，直到内层循环的条件不成立，外层循环才去执行下一次操作。

根据上述说明，程序第 7 行 for 语句的循环变量 i 每取得一个值，第 10 行的 while 语句就要执行 i<=j 的所有情况。

模拟程序的执行情况如下表所示。

i的值	j的值	输出情况
1	5	15
	4	14
	3	13
	2	12
	1	11
	0	退出while循环
2	5	25
	4	24
	3	23
	2	22
	1	退出while循环
3	5	35
	4	45
	3	33
	2	退出while循环
4	5	45
	4	44
	3	退出while循环
5	5	55
	4	退出while循环
6	退出for循环	

【例4.25】写出下面程序的运行结果。

```
1  //exam4.25
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6    int i=10,j;
7    do
8    for(j=1;j<i;j++)
9    while(i>5);
10   cout<<i+j;
11   i--;
12   return 0;
```

```
13 }
```

运行结果：出现编辑错误，无法运行。



说明：在多重循环中，内层循环必须在内层结束，不能出现内外循环交叉的情况。

程序中在第 7 行 do 开头的循环语句中，又出现了第 8 行 for 语句，这是一个明显的多重循环。但是，for 语句没有结束的情况下，就出现了第 9 行 while，结束了 do-while 语句，因此，程序出错，无法运行。

实验：如何修改上面的程序，能使其正确运行？

4.4.2　多重循环的应用

【例 4.26】对于给定的自然数 n（n<20），在屏幕上输出仅由“*”构成的 n 行的直角三角形。

例如：当 n = 4 时，输出：

```
*
**
***
*****
```

分析：打印图形总是逐行逐列进行的。对于本题，要重复 n 行操作，对于每一行，又重复若干次输出 * 的操作。于是，构成了一个两层循环：外层循环是 1 至 n 行的处理，而内层循环，则是输出同一行上的每一列。分析样例，不难发现，每一行上“*”的个数恰是行数。因此对于第 i 行，内层循环可以设置重复 i 次。

程序如下：

```
1  //exam4.26
2  #include<iostream>
3  using namespace std;
```

```
4  int main()
5  {
6      int i,j,n;
7      cin>>n;
8      for(i=1;i<=n;i++)          // 外层循环，控制行
9      {
10       for(j=1;j<=i;j++)        // 内层循环，控制每一行上的各列
11         cout<<"*";
12       cout<<endl;              // 每行最后的换行
13     }
14     return 0;
15 }
```

运行结果：



思考：在程序的第11行和第12行分别出现了输出语句：cout<<“*”和cout<<endl，在程序执行的过程中，它们各执行了多少次？

实验：如果删除程序的第9行和第13行，即外层循环的循环体不使用复合语句，写成如下形式，程序的执行结果会怎样？

```
for(i=1;i<=n;i++)
 for(j=1;j<=i;j++)
     cout<<"*";
 cout<<endl;
```

【例4.27】百钱买百鸡问题。鸡翁一，值钱五，鸡母一，值钱三，鸡雏三，值钱一，百钱买百鸡，问鸡翁、鸡母、鸡雏各几何？

方法1：在数学中解决这个问题，我们通常会列出一个方程组，设鸡翁x，鸡母y，鸡雏z，则：

$$x+y+z=100$$

5*x+3*y+z/3=100

同时满足上述两个方程的 x、y、z 值就是所求。

根据这个思路，问题就转化为求解方程组。

我们列举 x、y、z 的所有可能解，然后判断这些可能解是否能使方程组成立。能使方程组成立的，就是真正的解。

再进一步分析，x 的取值范围是 0~100/5，y 的取值范围是 0~0/3，z 的取值范围是 0~3*100。

程序如下：

```
1  //exam4.27-1
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int x,y,z;
7      for(x=0;x<=100/5;x++)           //列举鸡公的所有可能
8        for(y=0;y<=100/3;y++)         //列举鸡母的所有可能
9          for(z=0;z<=3*100;z++)       //列举鸡雏的所有可能
10           if(5*x+3*y+z/3==100 && x+y+z==100)
11             cout<<x<<" "<<y<<" "<<z<<endl;
12     return 0;
13 }
```

运行结果：

```
0 25 75
3 20 77
4 18 78
7 13 80
8 11 81
11 6 83
12 4 84
```

思考：这里用了一个三层循环的程序解决问题。当x取得一个数值时，for y的循环体都要执行遍y的所有取值；对于y的每一个取值，for z的循环体都要执行遍z的所有取值；对于z的每一个取值，if语句都要执行一次。

不难算出，在程序的执行过程中，作为最内层循环体的if语句，将被执行：(1+100/5)*(1+100/3)*(1+3*100)=214914次。

而观察程序的运行结果，问题的解远远小于这个数字——只有7组解。如何减少循环次数呢？

方法2：由于题目的特殊性，鸡公、鸡母、鸡雏共100只，一旦确定了鸡公x和鸡母y的数量，鸡雏便只能购买100-x-y只。这样，我们可以尝试写出一个两层循环的程序，解决这个问题。

程序如下：

```
1  //exam4.27-2
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int x,y,z;
7      for(x=0;x<=100/5;x++          // 列举鸡公的所有可能
8        for(y=0;y<=100/3;y++)       // 列举鸡母的所有可能
9        {
10           z=100-x-y;              // 根据x，y计算鸡雏的数量
11           if(5*x+3*y+z/3==100)    // 判断总钱数是否符合条件
12             cout<<x<<" "<<y<<" "<<z<<endl;
13       }
14     return 0;
15 }
```

运行结果如下：

```
0 25 75
3 20 77
4 18 78
7 13 80
8 11 81
11 6 83
12 4 84
```

说明：对于与本题类似的求解不定方程问题，都可以用循环来求解。为提高效率，可以在程序中进行适当优化，减少循环体的执行次数。

思考：你还能想出哪些办法优化程序，减少循环体的执行次数呢？

实验：鸡公x、鸡母y、鸡雏z中确定了任意两个，第三个都可以用减法的形式来确定。分别设置循环变量为x、z和y、z，再来编写程序，验证算法，并计算在不同的写法中，内层循环体的执行次数各是多少。

【例4.28】编程求出所有的水仙花数。

方法1：所谓水仙花数，在第3章中有过介绍：水仙花数是一类特殊的三位数，它们每一个数位上的数字的立方和恰好等于这个三位数本身。

例如：$153=1^3+5^3+3^3$。153 就是一个水仙花数。

模拟上述描述，我们可以循环产生所有的三位数，将三位数拆分开来，计算立方和并进行条件判断即可。

程序如下：

```
1  //exam4.28-1
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int x,a,b,c;
7      for(x=100;x<=999;x++)              // 列举所有的三位数
8      {
9          a=x/100;                       // 产生百位数字
10         b=(x-a*100)/10;                // 产生十位数字
11         c=x%10;                        // 产生个位数字
12         if(a*a*a+b*b*b+c*c*c==x)       // 判断是否满足条件
13           cout<<x<<endl;
14     }
15     return 0;
16 }
```

运行结果如下：

```
153
370
371
407
```

方法 2：在方法 1 中穷举了所有的三位数，共 900 个，循环体也执行了 900 次，而满足条件的数字仅有 4 个。有没有办法减少重复的次数呢？

现在我们换一个思路：方法 1 是拆分三位数，我们逆其道而行，从每一位数字出发合成三位数，会怎样呢？

令百位数 a 取 1~9，十位数 b 取 0~9，个位数 c 取 0~9，则对应的三位数 x=a*100+b*10+c。

程序如下：

```
1  //exam4.28-2
2  #include<iostream>
3  using namespace std;
4  int main()
```

```
5  {
6      int x,a,b,c;
7      for(a=1;a<=9;a++)                  //列举百位数字
8        for(b=0;b<=9;b++)                //列举十位数字
9          for(c=0;c<=9;c++)              //列举个位数字
10         {
11           x=a*100+b*10+c;              //由各位数字，计算x
12           if(a*a*a+b*b*b+c*c*c==x) //判断是否满足条件
13           cout<<x<<endl;
14         }
15     return 0;
16 }
```

运行结果如下：

```
153
370
371
407
```

思考：对比exam4.28-1与exam4.28-2，你有什么感悟？

【例4.29】把一个合数分解成若干个质因数乘积的形式（即求质因数的过程）叫做**分解质因数**。分解质因数（也称分解素因数）只针对合数。

输入一个正整数n，将n分解成质因数乘积的形式。

输入样例：

36

输出样例：

36=2*2*3*3

分析：将任意的n分解为质因数的乘积，要从最小的质数开始，那么，我们就不妨从2开始试除，能整除，就输出2，再对商继续试除，直到不再含有因子2；然后用下一个质数反复试除，……，再用下一个质数试除，……，一直到商为1，停止操作。

这里，质因数的递增，是一层循环，每一个质因数的反复试除，又是一层循环。因此，本题使用两层循环来解决。

程序如下：

```
1  //exam4.29
2  #include<iostream>
```

```
3  using namespace std;
4  int main()
5  {
6      int n,i=2;
7      cin>>n;
8      cout<<n<<"=";
9      do
10     {
11         while(n%i==0)    //n 能被 i 整除，就重复做除法操作
12         {
13           cout<<i;
14           n/=i;
15           if(n!=1) cout<<"*";
16         }
17         i++;
18     }
19     while(n!=1);         //n 没有除尽，就重复操作
20     return 0;
21 }
```

运行结果：

```
36
36=2*2*3*3
2310
2310=2*3*5*7*11
```

***思考：**你能估算出内层循环体的执行次数吗？

练　习

（1）读程序，写结果。

```
//test(1)-1
#include<iostream>
using namespace std;
int main()
{
   int i,j,n;
   cin>>n;
   for (i=1;i<=n;i++)
   {
     for (j=1;j<=n-i;j++)
```

```
            cout<<" ";
      for (j=1;j<=i;j++)
            cout<<"*";
      cout<<endl;
   }
   return 0;
}
```

输入：4

```
//test(1)-2
#include<iostream>
using namespace std;
int main()
{
   int i,j,n;
   cin>>n;
   for(i=2;i<=n;i++)
   {
       j=i-1;
       while(j>1 && i%j!=0)
          j--;
       cout<<i<<"("<<j<<")\n";
   }
   return 0;
}
```

输入：10

```
//test(1)-3
#include<iostream>
using namespace std;
int main()
{
   int i,m,n=0;
   for(i=1;i<=5;i++)
   {
     m=i%2;
     while(m-->0) n++;
   }
   cout<<m<<","<<n;
}
```

```
//test(1)-4
#include<iostream>
using namespace std;
int main()
{
   int n;
   cin>>n;
   cout<<n<<"=";
   for(int i=2;i<=n;i++)
     for(;n%i==0;)
     {
       n=n/i;
       cout<<i;
       if(n!=1) cout<<"*";
     }
     return 0;
}
```

分别输入：36　　2310

（2）输入一个正整数n，输出高为n的由*组成的等腰三角形。

输入样例：

```
3
```

输出样例：

```
  *
 ***
*****
```

（3）输入一个正整数n，输出用1–n的数字组成的菱形。

输入样例：

```
3
```

输出样例：

```
  1
 123
12345
 123
  1
```

（4）根据给定的n，输出乘法口诀表的前n行。

输入样例：

3

输出样例：

```
1*1=1
1*2=2  2*2=4
1*3=3  2*3=6  3*3=9
```

（5）将任意给定的整百元钞票，兑换成10元、20元、50元小钞票形式。输出兑换方案总数。

输入样例：

100

输出样例：

10

说明：

方案序号	10元张数	20元张数	50元张数
1	0	0	2
2	0	5	0
3	1	2	1
4	2	4	0
5	3	1	1
6	4	3	0
7	5	0	1
8	6	2	0
9	8	1	0
10	10	0	0

（6）数根是这样定义的：对于一个正整数n，将它的各个数位上的数字相加得到一个新数，如果这个数是一位数，我们就称之为n的数根，否则重复处理直到它成为一个一位数。

例如，n=34，3+4=7，7是一位数，所以7是34的数根。

再如，n=345，3+4+5=12，1+2=3，3是一位数，所以3是345的数根。

对于输入数字n，编程计算它的数根。

输入样例：

345

输出样例：

3

（7）[1]输入两个正整数x0，y0（2<=x0<100000，2<=y0<=1000000），求出满足下列条件的P、Q的个数：

（A）P、Q是正整数。

（B）要求P、Q以x0为最大公约数，以y0为最小公倍数。

输入样例：

3 60

输出样例：

4

说明：此时的P和Q分别为：

3和60

15和12

12和15

60和3

所以，满足条件的所有可能的两个正整数的个数共4种。

*4.5　在循环结构中应用位运算

【例4.30】还记得“宰相的麦子”吗？在8*8的棋盘里，第一个格子放一粒麦粒，以后每个格子放前一个格子麦粒数的2倍。将64个格子放满麦粒，可是让国王花费了不小的代价呢！

现在，宰相又出新花招，他要检验一下格子里放的麦粒数（整型范围内）是否正确。他的方式很特别：判断麦粒数是否是2的整数幂。

例如，$1024=2^{10}$，他就认为1024是正确的麦粒数。

编程解决老宰相的问题吧：输入任意一个整数n，如果n是2的整数幂，则输出Yes，否则输出No。

方法1：判断n是否是2的整数幂，最简单的办法，就是在数字（或商）大于0且能被2整除时，就反复除以2。退出循环时，若商不是1，n就不是2的整数幂，否则就是。

程序如下：

```
1  //exam4.30-1
2  #include<iostream>
```

1）本题选自NOIP2001普及组复赛试题。

```
3  using namespace std;
4  int main()
5  {
6      int n;
7      cin>>n;
8      while(n>0 && n%2==0)      //数字(商)大于0且能被2整除,
                                   就重复操作
9        n/=2;                   //产生新的数字
10     if(n!=1) cout<<"No";      //判断结果
11     else cout<<"Yes";
12     return 0;
13 }
```

运行结果如下：

```
1       2       1023    1024
Yes     Yes     No      Yes
```

方法2：解决这个问题还有一个巧妙的方法，就是利用位运算。

程序如下：

```
1  //exam4.30-2
2  #include <iostream>
3  using namespace std;
4  int main( )
5  {
6      int n;
7      cin>>n;
8      if(n&(n-1)) cout<<"No";  //利用位运算完成判断
9      else cout<<"Yes";
10     return 0;
11 }
```

运行结果：

```
1       2       1023    1024
Yes     Yes     No      Yes
```

对比两个程序，很明显exam4.30-2在书写上，更为简洁；在结构上，用第8行的条件n&(n-1)做判断，省掉了exam4.30-2中的循环，避免了重复操作。

数据在计算机中采用二进制形式存储，一个0或一个1称为一个二进制位，简称位。在C++语言中，有一类特殊运算，就是针对位进行的，我

们称之为位运算。

那么，C++ 提供了哪些位运算？应用位运算解决问题有哪些优势呢？下面，我们将结合循环结构学习 C++ 语言的位运算。

4.5.1　位运算符的含义

C++ 语言提供了六种位运算符，见表 4.1。

表 4.1　位运算符

运算符	含　义	说　明	例　子
&	按位与	把参与运算的两个数对应的二进制位相与，只有对应的二进制位均为 1 时，结果的对应位才为 1，否则为 0	9&5 相当于 00001001&00000101，运算结果为 00000001。输出结果是 1
|	按位或	把参与运算的两个数对应的二进制位相或，只要对应的两个二进制位有一个为 1 时，其结果就为 1	9|5 相当于 00001001|00000101，运算结果就是 00001101，输出结果是 13
^	按位异或	把参与运算的两个数对应的二进制位相异或，对应位上的两个二进制数字不同时，结果为 1，否则为 0	1^1=0，1^0=1，0^0=0，0^1=1
~	取反	把运算数的各个二进制位按位求反	~9 相当于 ~（00001001），运算结果 11110110。输出结果为 –10
<<	左移	m<<n 是指把 m 对应的二进制数的各个二进制位向左移 n 位，高位丢弃，低位用 0 补齐	设 a=3，a<<4 指把 00000011 的各二进制位向左移动 4 位，结果为 00110000（十进制 48）
>>	右移	m>>n 是指把 m 对应的二进制数的各二进制位向右移 n 位，低位丢弃，高位用 0 补齐	设 a=15，a>>2 表示把 00001111 右移 2 位，结果为 00000011（十进制 3）

【例 4.31】 对于 n 取得十进制 1~10 之间的每一个数，分别写出表达式 n&(n–1) 的值。

解： 对应于 n=1、2、3、4、5、6、7、8、9、10，表达式的值分别为 0、0、2、0、4、4、6、0、8、8。

说明： 使用位运算时，要将位运算符的操作数转换成为二进制形式再运算。

根据说明，我们首先将 n 的每一个取值转换为二进制形式，然后做 & 运算，具体见下表。

n的值	1	2	3	4	5	6	7	8	9	10
n对应的二进制值	1	10	11	100	101	110	111	1000	1001	1010
n-1对应的二进制值	0	01	10	011	100	101	110	0111	1000	1001
n&(n-1)二进制结果	0	0	10	000	100	100	110	0000	1000	1000
n&(n-1)对应十进制	0	0	2	0	4	4	6	0	8	8

综上，n取1~10之间的每一个数，n&(n-1)结果分别为0、0、2、0、4、4、6、0、8、8。更进一步，我们可以看出，n=1、2、4、8时，n&(n-1)结果为0，而这4个值恰好都是2的整数幂。因此，这个题目，从一定程度上，证明了exam4.30-2的正确性。

思考：如何证明当n>0时，若n&(n-1)=0，n一定是2的整数幂？

【例4.32】写出下面程序的运行结果。

```
1  //exam 4.32
2  #include <iostream>
3  using namespace std;
4  int main( )
5  {
6       int x=3,y=5;
7       x^=y;
8       y^=x;
9       x^=y;
10      cout<<"x="<<x<<",y="<<y;
11      return 0;
12 }
```

运行结果：

```
x=5,y=3
```

说明：^是二进制按位异或操作符，应用^可以解决很多实际问题。例如，对于给定的x和y值，经过x^=y; y^=x; x^=y;三步操作后，x和y的值将发生交换。

上述说明，说出了^的一个巧妙应用。我们可以模拟程序流程，分析其实现过程。

将x和y转换成对应的二进制数：$x=(3)_{10}=(11)_2$；$y=(5)_{10}=(101)_2$。

对应程序第7行x^=y，改变了x的值：x=x^y=11^101=011^101=110。

对应程序第 8 行 y^=x，改变了 y 的值：y=y^x=101^110=011。

对应程序第 9 行 x^=y，改变了 x 的值：x=x^y=110^011=101。

综上，运行程序后，x 和 y 的值分别为二进制 101 和 011，对应的十进制数是 5 和 3。实现了 x 和 y 的交换：x=5，y=3。

【例 4.33】写出下面程序的运行结果。

```
1  //exam4.33
2  #include <iostream>
3  using namespace std;
4  int main( )
5  {
6       int x,y;
7       cin>>x>>y;
8       cout<<(x&y)+((x^y)>>1);
9       return 0;
10 }
```

运行结果：

```
3 5
4
1 8
4
12 0
6
```

说明：^ 是二进制按位异或运算符，& 是二进制按位与运算符，>> 是二进制右移运算符。应用位运算符可以解决很多实际问题。例如，对于给定的 x 和 y 值，(x&y)+((x^y)>>1) 的结果是 x 和 y 的平均值。

上述说明，说出了 ^、& 以及 >> 相结合的一个巧妙应用。我们可以模拟程序流程，分析其实现过程。

对于输入的 x 和 y，以 3 和 5 为例，在程序的第 8 行，将完成多步运算：

（1）将 x 和 y 转换成二进制数：011 和 101。

（2）x&y=011&101=001。

（3）x^y=011^101=110。

（4）(x^y)>>1=(110)>>1=11。

（5）(x&y)+((x^y)>>1)=001+11=001+011=100。

（6）输出二进制数 100 对应的十进制数 4。

思考：你能应用位运算符设计表达式，完成特定的计算吗？写出几个实例。

4.5.2 位运算的应用

【例4.34】学习了一段C++程序设计后，老师组织了验收考核。考核分为笔试和上机两种形式，每位同学也便有了两个成绩。现在老师想知道，参加考核的n人中有多少人只有一个成绩不及格（每场考核满分为100，不低于60分为及格）。

分析： 题目很清楚，是对n人进行条件计数。因此可以写成一个for循环语句，重复处理每一个人的成绩：若笔试成绩为x，上机成绩为y，则两个成绩都不及格的条件就是：x<60 && y<60；只有一个成绩不及格的条件就是：从x<60 || y<60的人数中除去x<60 && y<60的人数。所以，满足题目要求的条件式应写成：(x<60 || y<60) && !(x<60 && y<60)

这个条件式的写法看起来比较复杂，我们再进一步分析。

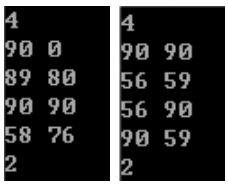
事实上，我们想要的是x<60和y<60中只有一个成立的结果。两个操作数只有一个为1，结果才为1，恰是异或（^）运算的含义。因此，我们可以将上述条件简化为如下形式：

```
x<60^y<60
```

程序如下：

```
1  //exam4.34
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      float x,y;
7      int n,i,ans=0;
8      cin>>n;
9      for(i=1;i<=n;i++)
10     {
11       cin>>x>>y;
12       if(x<60^y<60)ans++;            //用异或操作实现判断
13     }
14     cout<<ans;
15     return 0;
16 }
```

运行结果：



实验：结合本题的“分析”部分，将程序的第12行改写成逻辑表达式，检验程序的正确性。

【例 4.35】在一条笔直的街道上，有无数的街灯，每盏街灯有自己的独立开关。为了检验灯的质量，管理员想出了一个有趣的办法。找若干个人按顺序一个一个地从街道的一侧进入，每个人看到亮着的灯就熄灭，直到看到第一盏关着的灯，将其点亮，完成任务。

如果所有的灯都质量完好，那么第m个人走过后，有多少灯被点亮过？

分析：灯的状态只有两种情况，不妨用0表示灭，1表示亮。下面我们以10个人为例，模拟所有人走过街灯的过程。

状　态	灯的情况（0灭1亮）	被点亮过的灯数
初始状态	……00000000	0
第1人走过	……00000001	1
第2人走过	……00000010	2
第3人走过	……00000011	2
第4人走过	……00000100	3
第5人走过	……00000101	3
第6人走过	……00000110	3
第7人走过	……00000111	3
第8人走过	……00001000	4
第9人走过	……00001001	4
第10人走过	……00001010	4

不难看出，灯的状况恰好构成了一个二进制数。再进一步，被点亮过的灯数，恰好是这个二进制数的位数（不含前导0）。再联系每个人的编号，我们还能发现，十进制编号恰好与灯情况的二进制数相对应。

至此，问题转化成求给定的正整数m所对应的二进制数的位数。

例如，m=10，二进制数是1010，输出位数是4。

那么如何计算一个十进制数对应二进制数的位数呢？4.3节的例4.22已经解决过这个问题，可以自行查阅。

现在，我们借助位运算的思想，再进一步分析。要做除以2的操作，可以利用位运算的右移操作（>>1）来完成：n>>1等价于n/2。

程序如下：

```
1  //exam4.35
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int n,num=0;
7      cin>>n;
8      do
9       {
10         num++;        // 位数加 1
11         n>>=1;        // 用位移操作产生新的数字
12      }
13     while(n!=0);   // 商非 0，就重复操作
14     cout<<num;
15     return 0;
16 }
```

运行结果：

```
1   2   13   1024
1   2   4    11
```

本程序的分析过程很巧妙：

（1）将一个看似复杂的问题，转化为统计二进制数位数的问题，使得编程的复杂度大大降低。

（2）用位运算替代了除法运算。虽然从代码上看，两个程序的长度并没有明显差别。但是，在执行程序时，二者的区别就会体现出来：位运算的效率高于四则运算。

*有研究说，由于位运算的操作单元是位（bit），相对普通操作，其运算效率能提高60%。因此，在写程序时，适当应用位运算，可以大大提高程序的运行速度。编写大型程序的人，对这一点有更深刻的体验。

***实验：**解决m对应二进制数的位数，还有一个数学方法，就是借助

$\log_2 m$的结果。这种方法需要调用cmath中的求对数函数。如果感兴趣，可以自行编程实验。

练　习

（1）读程序，写结果。

```
//test(1)-1
#include <iostream>
using namespace std;
int main()
{
     int a,b,c,d;
     cin>>a>>b;
     do
     {
          c=a^b;
          d=(a&b)<<1;
          a=c;
          b=d;
     }
     while(b!=0);
     cout<<a;
     return 0;
}
```

分别输入：0 9　　12 16　　1024 1023

```
//test(1)-2
#include <iostream>
using namespace std;
int main()
{
     int n=0,x;
     cin>>x;
     for(n=0;x!=0;x&= x-1)
       n++;
     cout<<n;
     return 0;
}
```

分别输入：　0　　1024　　4095

```
//test(1)-3
#include <iostream>
using namespace std;
int main()
{
   int a,b,c,d,e,f;
   cin>>a>>b;
   c=a&b;
   d=a^b;
   while(c)
   {
     e=d;
     f=c<<1;
     c=e&f;
     d=e^f;
   }
   cout<<d;
   return 0;
}
```

分别输入：0 9　12 16　1024 1023

（2）模拟将任意给定的正整数n转换成对应的二进制数的过程：对于输入的任意正整数n，输出若干行“商*，余*”的形式，表示其转换过程。

输入样例：

13

输出样例：

商6，余1
商3，余0
商1，余1
商0，余1

（3）计算a^b的个位数。

输入样例：

2 63

输出样例：

8

（4）又到休息时间了。两个小伙伴要玩扑克牌比大小的游戏：每人分到一种花色的扑克牌，然后每人随机出一张牌，大者胜，最后获胜次数多

的人，赢一轮。

不开心的事发生了，玩了一轮后，发现扑克牌竟然少了一张。到底少了几呢？

你能写个程序，快速找到扑克x吗？

输入1行，包括25个用空格分隔的数字（A表示为1，J、Q、K表示为11、12、13）。

输出数字x，表示缺少的扑克牌数字。

输入样例：

9 1 13 2 6 10 7 8 3 11 4 1 5 9 10 2 4 3 12 13 11 5 8 6 7

输出样例：

12

（5）对于任意给定的n，计算2的n次方。

输入样例：

3

输出样例：

8

（6）[1]有n个小伙伴（编号从0到n−1）围坐成一个圆圈玩游戏。按照顺时针方向给n个位置编号，从0到n−1。最初，第0号小伙伴在第0号位置，第1号小伙伴在第1号位置，……，第n−1号小朋友在第n−1位置。

游戏规则如下：

每一轮第0号位置上的小伙伴顺时针走到第m号位置，第1号位置小伙伴走到第m+1号位置，……，第n−m位置上的小伙伴走到第0号位置，第n−m+1号位置上的小伙伴走到第1号位置，……，第n−1号位置上的小伙伴顺时针走到第m−1号位置。

现在，一共进行了10^k轮，请问x号小伙伴最后走到了第几号位置。

输入1行，包含4个整数n、m、k、x，每两个整数之间用一个空格隔开。

输出一个整数，表示10^k轮后x号小伙伴所在的位置编号。

输入样例：

10 3 4 5

输出样例：

5

1）本题选自NOIP2013提高组复赛试题。

*4.6 循环结构程序设计实例

【例4.36[1)]】津津上初中了。妈妈认为津津应该更加用功学习，所以津津除了上学之外，还要参加妈妈为她报名的各科复习班。另外每周妈妈还会送她去学习朗诵、舞蹈和钢琴。但是津津如果一天上课超过八个小时就会不高兴，而且上得越久就会越不高兴。假设津津不会因为其他事不高兴，并且她的不高兴不会持续到第二天。请你帮忙检查一下津津下周的日程安排，看看下周她会不会不高兴；如果会的话，哪天最不高兴。

输入格式：包括7行数据，分别表示周一到周日的日程安排。每行包括两个小于10的非负整数，用空格隔开，分别表示津津在学校上课的时间和妈妈安排她上课的时间。

输出格式：1行，仅一个数字。如果不会不高兴，则输出0，如果会，则输出最不高兴的是周几（用1, 2, 3, 4, 5, 6, 7分别表示周一，周二，周三，周四，周五，周六，周日）。如果有两天或两天以上不高兴，则输出时间最靠前的一天。

输入样例：

```
5 3
6 2
7 2
5 3
5 4
0 4
0 6
```

输出样例：

```
3
```

分析：将本题抽象出数学模型，实际是求7组数字之和中超过8的最大一组所在位置。因为7天限定了数据的组数不大，而每一组数的和也不会超过20（题目约定在校和课后时间都小于10），求和与比较操作也不麻烦，所以，我们不需要做过多的优化，直接对7组数字重复求和、找最大并记录位置就可以了。

1）本题选自NOIP2004普及组复赛试题。

程序如下：

```
1  //exam4.36
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6    int s1,s2;           //s1,s2 分别表示上学时间和课后学习时间
7    int day=0;           //day 表示不高兴的一天，初始为 0
8    int max=8;           //max 表示最长时间，初始为 8
9    int i;
10   for(i=1;i<=7;i++)    // 重复处理每一组数据
11   {
12       cin>>s1>>s2;
13       if(s1+s2>max)    // 若学习时间超过当前 max，则更新，并记录
                             是周几
14       {
15           max=s1+s2;   // 更新 max
16           day=i;       // 记录周几
17       }
18   }
19   if(max>8) cout<<day;
20   else cout<<"0";
21   return 0;
22 }
```

运行结果：

```
5 3  2 3  4 0
6 2  7 0  7 1
7 2  6 1  2 5
5 3  0 0  4 5
5 4  4 3  6 3
0 4  2 6  0 0
0 6  6 2  0 5
3    0    4
```

【例 4.37[1)]】试计算在区间 1 到 n 的所有整数中，数字 x（0 ≤ x ≤ 9）共出现了多少次？

例如，在 1 到 11 中，即在 1、2、3、4、5、6、7、8、9、10、11 中，

1）本题选自 NOIP2013 普及组复赛试题。

数字1出现了4次。

输入格式：共1行，包含2个整数n、x，之间用一个空格隔开。

输出格式：共1行，包含一个整数，表示x出现的次数。

输入样例：

11 1

输出样例：

4

数据范围：

对于100%的数据，1≤n≤1,000,000，0≤x≤9

分析：对于一个数字i，判断其中出现了几个x，我们可以逐个数位进行比较、计数。拆分数位，可以参考4.2节的例4.14，直接使用while语句来解决。

要对1~n中的每一个数字进行统计，可以使用一个n次的循环，每次循环做一次拆分计数。

至此，本题解决方法很明确了——使用两层循环来完成计数。

程序如下：

```
1  //exam4.37
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6    int ans=0,i, n,x,tmp;
7    cin>>n>>x;
8    for(int i=1;i<=n;i++)    // 重复对 1~n 中的每一个数进行计数
9    {
10     tmp=i;
11     while(tmp>=1)          // 对于数字逐位判断、计数
12     {
13       if(tmp%10==x)ans++; // 如果当前的末位数字是 x，则计数器
                                加一
14       tmp/=10;             // 为下一次取得末位数字做准备
15     }
16   }
17   cout<<ans;
18   return 0;
19 }
```

运行结果：

```
8 9
0
11 1
4
987654 3
595336
1000000 0
488895
```

【例 4.38】一个寝室有两张上下铺的床位，第一张床上下铺分别编号为1、2；第二张床的上下铺分别编号为3、4。

现在，A、B、C、D四个人被分到了一个寝室，她们分别有各自的期望。

A 的心里话：“最好我住 3 床，B 住我下铺。”

B 的心里话：“我要住 1 床，D 离我远点，住 4 床才好呢。”

C 的心里话：“我喜欢住下铺，就 4 床吧，D 住我上铺。”

D 的心里话：“我要住 2 床，A 住我上铺。”

结果，老师分配的结果一公布，每个人都是半喜半忧，愿望只实现了一半。

你能推算出老师的分配方案吗？请编程输出四个人的床位编号。

分析：这是一个典型的逻辑推理题目。我们可以将 A、B、C、D 分别分配在 1 到 4 号床位，对于每一种分配方案都判断一次是否满足题目要求，如果满足，则输出四个人的床位号。

分析至此，我们可以用循环穷举 A、B、C、D 的取值，在循环体内完成筛选即可。

但是，还有两个疑问，需要解决：

（1）四个人的期望各实现一半。在程序中如何表示呢？

判断一个人的预期只实现一半，我们可以参考 4.5 节的例 4.34，使用异或（^）运算来实现。

以 A 的预期为例，可以写作：(A==3)^(B==4)。

于是，四个人的预期各实现一半，就可以将四个异或运算写在一起：

```
(A==3)^(B==4)&&(B==1)^(D==4)&&(C==4)^(D==3)
&&(D==2)^(A==1)
```

（2）四张床对应四个人，四个人的床号要保证各不相同。如何表示呢？

首先确定 A；然后在确定 B 的时候，加上条件 B!=A；确定 C 的时候，加上条件 (C=!A)&&(C!=B)；确定 D 的时候，加上条件 (C=!A)&&(D!=B)&&(D!=C)。另一种办法是 A、B、C 都确定，D 的床号可直接算出，D=1+2+3+4−A−B−C。

程序如下：

```
1  //exam4.38
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6    int A,B,C,D;
7    for(A=1;A<=4;A++)              // 枚举 A 的可能床位
8     for(B=1;B<=4;B++)             // 枚举 B 的可能床位
9     if(A!=B)
10     for(C=1;C<=4;C++)            // 枚举 C 的可能床位
11     if(A!=C && B!=C)
12     {
13       D=1+2+3+4-A-B-C;           // 计算 D 的床位
14       if((A==3)^(B==4)&&(B==1)^(D==4)&&(C==4)^(D==3)
         &&(D==2)^(A==1))           // 四个预期各实现一半
15       cout<<A<<B<<C<<D;
16     }
17   return 0;
18 }
```

运行结果：

```
3142
```

【例 4.39[1)]】春春幼儿园举办了一年一度的“积木大赛”。今年比赛的内容是搭建一座宽度为n的大厦，大厦可以看成由n块宽度为1的积木组成，第i块积木的最终高度需要是hi 。

在搭建开始之前，没有任何积木（可以看成n块高度为0的积木）。接下来每次操作，小朋友们可以选择一段连续区间[L,R]，然后将第L块到第R块之间（含第L块和第R块）所有积木的高度分别增加1 。

小M是个聪明的小朋友，她很快想出了建造大厦的最佳策略，使得建造所需的操作次数最少。但她不是一个勤于动手的孩子，所以想请你帮忙实现这个策略，并求出最少的操作次数。

输入包含两行：第1行包含一个整数n，表示大厦的宽度；第2行包含n个整数，第i个整数为hi。

输出仅1行：即建造所需的最少操作数。

1）本题选自NOIP2013提高组试题。

输入样例：

5

2 3 4 1 2

输出样例：

5

说明：其中一种可行的最佳方案，依次选择：[1,5]，[1,3]，[2,3]，[3,3]，[5,5]。

数据范围：

对于 30% 的数据，有 1 ≤ n ≤ 10；

对于 70% 的数据，有 1 ≤ n ≤ 1000；

对于 100% 的数据，有 1 ≤ n ≤ 100000，0 ≤ hi ≤ 10000。

分析：题面描述较抽象，我们不妨将它图形化。本题样例要达到的目标如图 4.5 所示。

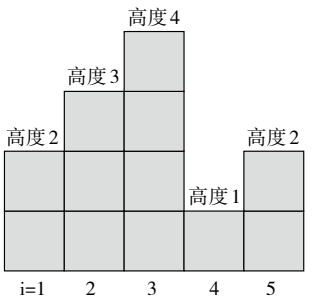


图 4.5　样例对应的大厦

观察图示，我们可以想到：

（1）在同一个高度，如果是相连的区域，那么这个区域的所有位置必然是应用一次操作同时增高而来的，例如 1、2、3、4、5 的第一层，1、2、3 的第二层等。

（2）在同一个高度，如果出现缺口，那么必然需要一次操作，将缺口右侧的层数增高。例如 4 的第二层。

基于上面的分析，我们来模拟样例的搭建过程：

第一步，因为在第 0 层时，位置 1、2、3、4、5 都是平层，所以，对于 1~5 这个连续区域，可以应用一次操作，同时增高一层，达到一层高度。这时，位置 4 达到要求的高度。

第二步，位置 1、2、3 都要求二层以上，三者又是相连区域。所以，

对于1~3，可以应用一次操作，同时增高一层，达到二层高度。这时，第1个位置达到要求目标。

第三步，位置2、3要求三层以上，并且二者相连。所以，对于2~3，可以应用一次操作，同时增高一层，达到三层高度。这时，第2个位置达到要求目标。

第四步，位置3要求四层，而与其相连的位置4已达到目标。所以，单独对3操作一次，增高一层，达到四层高度。这时，位置3达到要求高度。

第五步，位置4已达高度，所以跳过位置4，直接看位置5。单独对5操作一次，增高一层，达到二层高度。这时，位置5达到要求高度。

五步操作，五个位置全部达到要求的高度目标。

再归纳分析：从左到右，第1个位置从平层操作2次，达到目标；位置2比位置1再增加1次操作（位置1操作的同时，已被加高到二层）；位置3比位置2再增加1次操作（位置2操作的同时，已被加高到三层）；位置4无需再操作（已达到目标）；位置5比位置4再增加1次操作（位置4加到一层时，已被加高到一层）。

从样例想开去，我们只需从左到右模拟增高的过程即可。

一般地，当前位置的要求高度高于前一个位置，那么当前位置的操作次数，就是在前一个位置的高度之上增加二者的高度差。

特殊位置有两种情况：一种情况是位置1，可以理解为在高度为0的位置0基础上增加高度；另一种情况，该位置的高度等于或低于前一个位置，则无需操作。

分析至此，我们只需对n个位置一一处理，可以写出一个一层循环的程序。

程序如下：

```
1  //exam4.39
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int i,n,num=0;        //num 为操作次数，初值为 0
7      int s0=0,s1;          //s0 表示前一个位置高度，初值为 0，s1
                               表示当前位置要求高度
8      cin>>n;
9      for(i=1;i<=n;i++)     // 重复对 n 个位置，逐一处理
```

```
10      {
11          cin>>s1;        //读入当前位置要求的高度
12          if(s1>s0)       //若高于前一个位置高度，则增加操作次数
13           num+=s1-s0; //增加操作次数（当前位置与前一位置的高度差）
14          s0=s1;          //更新前一位置高度值，为下一个位置做准备
15      }
16      cout<<num;
17      return 0;
18 }
```

运行结果：

```
5
2 3 4 1 2
5
20
99 27 61 62 17 79 61 22 13 49 71 61 8 81 67 80 47 83 88 30
381
```

从本节的实例可以看出，使用循环结构写程序，代码并不复杂，但是需要缜密的思维。特别是当数据范围较大时，循环操作可能会花费较多时间。这时候，就要特别考虑循环的优化，例如，减少循环的层数，减少循环的次数等。

*在正规竞赛的题目描述中，经常出现数据范围的约定。数据范围对于编程者的意义，在于暗示了算法的优化程度。

如果一台计算机一秒钟计算 10^8 次，在要求程序只能在一秒钟内出解的情况下，对于 n 个数据操作，若 n 的上限是 10^8，那么程序最多只能写成一层循环；若 n 的上限是 10^4，那么程序最多只能写成两层循环。

例如，例 4.39 中 n 的上限为 100000，一层循环就是被允许的。

练　习

（1）[1] 将 1、2、……、9 共 9 个数排成下列形态的三角形。

```
        a
     b     c
   d         e
 f    g    h    i
```

1）本题选自 NOIP1996 普及组复赛试题。

其中，a～i分别表示1，2，……，9中的一个数字，并要求同时满足下列条件：

（A）a<f<i

（B）b<d, g<h, c<e

（C）a+b+d+f=f+g+h+i=i+e+c+a=p

请根据给定的p，计算满足上述条件的三角形的个数。

（2）[1]输入三个自然数n、i、j（1≤i≤n，1≤j≤n），输出在一个有n*n个格子的棋盘中，与格子（i，j）同行、同列、同一对角线的所有格子的位置。

例如：n=4，i=2，j=3，表示在4*4棋盘中，指定位置为第二行第三列。

第一列	第二列	第三列	第四列	
				第一行
		（2，3）		第二行
				第三行
				第四行

当n=4，i=2，j=3时，输出的结果是：

(2,1) (2,2) (2,3) (2,4)　{同一行上格子的位置}

(1,3) (2,3) (3,3) (4,3)　{同列列上格子的位置}

(1,2) (2,3) (3,4)　{左上到右下对角线上的格子的位置}

(4,1) (3,2) (2,3) (1,4)　{左下到右上对角线上的格子的位置}

输入样例：

4 2 3

输出样例：

(2,1)(2,2)(2,3)(2,4)

(1,3)(2,3)(3,3)(4,3)

(1,2)(2,3)(3,4)

(4,1)(3,2)(2,3)(1,4)

（3）[2]已知正整数n是两个不同质数的乘积，试求出其中较大的那个质数。

输入只有1行，包含一个正整数n（对于60%的数据，6≤n≤1000；对于100%的数据，6≤n≤$2*10^9$）。

1）本题选自NOIP1999普及组复赛试题。

2）本题选自NOIP2012普及组复赛试题。

输出只有1行，包含一个正整数 p，即较大的那个质数。

输入样例：

21

输出样例：

7

（4）[1]津津的零花钱一直都是自己管理。每个月的月初妈妈给津津300元钱，津津会预算这个月的花销，并且总能做到实际花销和预算的相同。

为了让津津学习如何储蓄，妈妈提出，津津可以随时把整百的钱存在她那里，到了年末她会加上20%还给津津。因此津津制定了一个储蓄计划：每个月的月初，在得到妈妈给的零花钱后，如果她预计到这个月的月末手中还会有多于100元或恰好100元，她就会把整百的钱存在妈妈那里，剩余的钱留在自己手中。

例如11月初津津手中还有83元，妈妈给了津津300元。津津预计11月的花销是180元，那么她就会在妈妈那里存200元，自己留下183元。到了11月月末，津津手中会剩下3元钱。

津津发现这个储蓄计划的主要风险是，存在妈妈那里的钱在年末之前不能取出。有可能在某个月的月初，津津手中的钱加上这个月妈妈给的钱，不够这个月的原定预算。如果出现这种情况，津津将不得不在这个月省吃俭用，压缩预算。

现在请你根据2004年1月到12月每个月津津的预算，判断会不会出现这种情况。如果不会，计算到2004年年末，妈妈将津津平常存的钱加上20%还给津津之后，津津手中会有多少钱。

输入包括12行数据，每行包含一个小于350的非负整数，分别表示1月到12月津津的预算。

输出仅1行，只包含一个整数。如果储蓄计划实施过程中出现某个月钱不够用的情况，输出-X，X表示出现这种情况的第一个月；否则输出到2004年年末津津手中会有多少钱。

输入样例1：

290
230
280
200

1）本题选自NOIP2004提高组复赛试题。

300
170
340
50
90
80
200
60

输出样例1：

-7

输入样例2：

290
230
280
200
300
170
330
50
90
80
200
60

输出样例2：

1580

（5）小A和小B是一对好朋友，他们的爱好是研究数字。学过除法之后，他们就发明了一个新游戏：两人各说一个数字分别为a和b，如果a能包含b的所有质数因子，那么A就获胜。但是当数字太大的时候，两个朋友的脑算速度就有点跟不上了。

现在，请你写个程序，来判断胜负吧：输入两个正整数，表示a和b（1<=a，b<=10^{18}）。如果a包含了b的所有质数因子，则输出“Yes”，否则输出“No”（输出时没有引号）。

输入样例1：

120 75

输出样例 1：

Yes

输入样例 2：

7 8

输出样例 2：

No

附录 D　break 语句和 continue 语句

在循环结构程序中，有时需要提前终止循环或跳过特定的语句，为此，C++ 语言还提供了一些转移语句。

这里介绍两个转移语句：break 语句和 continue 语句。

D.1　break 语句的格式和用法

1. 格式

break;

2. 功能

中断所在循环体（或 switch…case 语句块），跳出本层循环。

下面看一个 **break 语句**用于循环结构中的实例——将任意大于 4 的偶数 n 表示为两个素数之和。

程序如下：

```
1  //break
2  #include<iostream>
3  #include<cmath>
4  using namespace std;
5  int main()
6  {
7    int n,x,y,i;
8    cin>>n;
9    for(x=3;x<=n/2;x+=2)          // 穷举第一个加数 x
10   {
11     for(i=2;i<=sqrt(x);i++)     // 判断 x 是否为素数
12       if(x%i==0)                // 如果 x 被 i 整除，则 x 非素数，
```

```
13          break;                    // 直接退出for i循环语句，后面
                                         的i就不用判断了
14      if(i>sqrt(x))y=n-x;           // 若i大于sqrt(x)，x一定是
                                         素数，生成第二个加数y
15      else break;                   // 否则，x不是素数，直接退出
                                         for x循环值
16      for(i=2;i<=sqrt(y);i++)       // 至此，已经有了素数加数x，后面
                                         判断y的过程同x
17          if(y%i==0)break;          // 退出for i循环
18      if(i>sqrt(y))cout<<n<<"="<<x<<"+"<<y<<endl;
19    }
20    return 0;
21 }
22
```

运行结果：

```
6
6=3+3
100
100=3+97
4096
4096=3+4093
4096=5+4091
```

需要注意的是，break是用来中断循环的语句，只能用在循环体（switch…case语句块）中，用在其他地方会出错。

D.2　continue语句的格式和用法

1. 格式

continue;

2. 功能

在循环执行的过程中，如遇到**continue语句**，程序将结束本次循环，接着开始下一次的循环。

下面看一个含有continue语句的程序的运行结果。

程序如下：

```
1  //continue
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int i;
7      for(i=10;i>0;i--)
```

```
8       {
9         if(i==5)continue;        //i=5 时，不执行后面的输出，返回
                                     去处理下一个 i 值
10        cout<<i<<endl;
11      }
12      return 0;
13 }
```

运行结果：

```
10
9
8
7
6
4
3
2
1
```

作为对比，我们将程序第 9 行的 continue 语句直接改写成 break 语句，并查看新程序的运行结果。

```
1  // 对比 contine 和 break
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6       int i;
7       for(i=10;i>0;i--)
8       {
9         if(i==5)break;
10        cout<<i<<endl;
11      }
12      return 0;
13 }
```

运行结果：

```
10
9
8
7
6
```

对比两个程序，可以更清晰地看到continue语句并不结束循环，只是从循环体中当前位置跳转到循环的开始处，继续执行循环体（输出结果只是跳过了5）；而break语句直接结束循环（输出结果不仅跳过了5，还跳过5之后的所有数字）。

在使用continue语句时，需要注意：它只能出现在循环体内，出现在其他位置会出错。

第5章　数据的批量存储

分类存放、分类编号的事例在生活中处处可见，如：图书馆中书的排列，相同类别排列在一起，其编号也是归类进行编排；我们的地址，通常用某省、某城市、某区域多少号标识；学生也常常用某学校、某年级、某班级多少号来标识自己，这样编排标识的目的是能够快速并唯一地查找到需要的东西或信息。我们可以发现生活中常见的标识编排通常由两部分组成，一个是名称、一个是编号，通过名称迅速找到方位，通过编号找到确定的目标。当要把一批数据存入计算机时，如何存储和查找使用数据呢？在C++语言中，借鉴了生活中的分类编号的思想，引入数组解决数据的批量存储问题。

5.1　一维数组

【例5.1】读入n个整数将其反向输出（n≤10000）。

输入样例：

5

45 12 34 89 21

输出样例：

21 89 34 12 45

分析：要将读入的数据反向输出，首先要读完全部数据，如何存储这些数据呢？由于n比较大，用前面学习的简单变量存储显然不方便。这里，使用一个特殊变量来完成。

程序如下：

```
1  //exam5.1
2  #include<iostream>
3  using namespace std;
4  const int MAXN=10000;
5  int main()
6  {
7      int a[MAXN];          //定义10000个数组元素
```

```
8       int i,n;
9       cin>>n;                  // 读入输入数据个数
10      for(i=0;i<n;i++)         // 读入 n 个数据存入 a[0]~a[n-1]
                                    数组变量中
11        cin>>a[i];
12      for(i=n-1;i>=0;i--)  // 倒序输出数组变量内容
13        cout<<a[i]<<"";
14      return 0;
15  }
```

运行结果：

```
10
56 43 52 67 89 92 51 90 71 76
76 71 90 51 92 89 67 52 43 56
```

程序中用一维数组解决批量数据存储问题，同时可以方便地引用数组数据。那么，如何定义使用一维数组？一维数组下标有哪些意义？如何使用一维数组灵活解决问题？为了回答这些问题，我们将学习 C++ 语言的一维数组。

5.1.1　一维数组

同类型变量或对象的集合称为数组。

1. 定义

在 C++ 语言中，**一维数组**定义方法如下：

类型名　数组名 [元素个数];

其中，元素个数必须是常数或常量表达式。

数组中的变量称为数组元素，由于数组中每个元素都有下标，因此数组元素也称为下标变量。

数组下标取值从 0 开始，使用数组时下标不能越界。同一数组的所有数组元素在内存中占用一片连续的存储单元。

例如：“int num[10];”定义了一个名字为 num 的数组，它有 10 个元素，每个元素都是一个 int 型变量，下标变量为 num[0] 到 num[9]，num 数组占用了一片连续的、大小为 10*sizeof(int) 字节的空间。

2. 引用

每个数组元素都是一个变量，数组元素可以表示为：

数组名 [下标]

其中，下标可以是任何值为整型的表达式，该表达式里可以包含变量和函数调用。引用时，下标值应在数组定义的下标值范围内。

例如：若i、j都是int型变量，则

```
num[5]
num[i+j]
num[i++]
```

都是合法的元素。

数组的精妙在于下标可以是变量，通过对下标变量值的灵活控制，达到灵活处理数组元素的目的。在例5.1中通过循环控制数组下标值从0变化到n-1，将读入的n个数逐个存入a[0]~a[n-1]中，反向输出只要控制数组下标值从n-1变化到0，逐个输出即达到目标。

【例5.2】斐波那契数列指的是这样一个数列：0、1、1、2、3、5、8、13、21、……求数列的前20项并按从大到小的顺序输出。

分析：在第4章已经研究过这个特殊的数列。对于本问题，由于要求输出是按从大到小的顺序，那么，需要先求得数列的各项再倒序输出。

设数组a存放数列的各项，则：

```
a[0]=0
a[1]=1
a[2]=a[0]+a[1]
……
a[i]=a[i-2]+a[i-1]
```

程序如下：

```
1  //exam5.2
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int a[20];                  // 定义 20 个数组元素
7      int i;
8      a[0]=0;                     // 数列第 1 项初值
9      a[1]=1;                     // 数列第 2 项初值
10     for(i=2;i<20;i++)           // 求数列的第 3~20 项
11       a[i]=a[i-2]+a[i-1];
```

```
12      for(i=19;i>=0;i--)          // 倒序输出数列
13        cout<<a[i]<<"";
14      return 0;
15 }
```

运行结果：

```
4181 2584 1597 987 610 377 233 144 89 55 34 21 13 8 5 3 2 1 1 0
```

说明：程序第 11 行利用数组实现递推求出数列各项，与第 4 章的方法不同。给出了一种利用数组便捷实现递推的思维方式。

【例 5.3】陶陶家的院子里有一棵苹果树，每到秋天树上就会结出 10 个苹果。苹果成熟的时候，陶陶就会跑去摘苹果。陶陶有个 30cm 高的板凳，当她不能直接用手摘到苹果的时候，就会踩到板凳上再试试。

现在已知 10 个苹果到地面的高度，以及陶陶把手伸直的时候能够达到的最大高度，请帮陶陶算一下她能够摘到的苹果的数目。假设她碰到苹果，苹果就会掉下来。

输入格式：包括两行数据，第 1 行包含 10 个 100 到 200 之间（包括 100 和 200）的整数（以 cm 为单位）分别表示 10 个苹果到地面的高度，两个相邻的整数之间用一个空格隔开；第 2 行只包括一个 100 到 120 之间（包含 100 和 120）的整数（以厘米为单位），表示陶陶把手伸直的时候能够达到的最大高度。

输出格式：1 行，只包含一个整数，表示陶陶能够摘到的苹果的数目。

输入样例：

```
100 200 150 140 129 134 167 198 200 111
110
```

输出样例：

```
5
```

分析：在第 4 章我们研究过一个类似的问题，只是数据的输入顺序不同。本题的数据是，先输入 10 个苹果的高度，再输入陶陶把手伸的最大高度。因此，需要定义数组存储 10 个苹果的高度，再对存储数组中的数据进行比较计数。具体实现步骤如下：

（1）设数组 a[10] 用于存储 10 个苹果离地面的高度，h 存储板凳和手伸高度之和，设 n 存储摘到的苹果的数目。

（2）读入 10 个苹果离地面的高度存储到 a[0]~a[9] 数组变量中。

（3）读入手伸高度并加上板凳高度。

（4）循环i从0~9，如果h>=a[i]，则摘下的苹果数目加1。

（5）输出n的值。

程序如下：

```
1  //exam5.3
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int a[10];                  //定义10个数组元素
7      int i,h,n=0;
8      for(i=0;i<=9;i++)           //读入10个苹果离地面的高度存储到
                                     a[0]~a[9]中
9        cin>>a[i];
10     cin>>h;
11     h+=30;
12     for(i=0;i<=9;i++)           //循环
13       if(h>=a[i])n++;           //如果板凳和手伸高度之和>=a[i],
                                     则摘下的苹果数目加1
14     cout<<n;                    //输出
15     return 0;
16 }
```

说明：程序第13行实现在数组中查找满足条件的数，提供了遍历数组查找满足条件的数据的方法。

5.1.2 数组的初始化

在定义一个一维数组的同时，可以给数组中的元素赋初值。

格式：

类型名 数组名[常量表达式]={值1,值2,……}

例如：

```
int a[10]={0,1,2,3,4,5,6,7,8,9}
```

相当于：

```
a[0]=0;a[1]=1;a[2]=2;……;a[9]=9
```

【例5.4】输入年、月、日，输出该天是这一年的第几天。

分析：设 year 表示年，month 表示月，date 表示日，ans 表示求得的第几天，那么，ans值是month之前月份天数之和加date。如果不考虑闰年，则每个月的天数是固定的。设数组 a 存放每个月的天数。

具体实现步骤如下：

（1）初始化数组 a，其值为每个月的天数。

（2）读入 year、month、date。

（3）求 month 之前月份天数之和加上 date，则为平年的第几天。

（4）判断 year 是否闰年，如果是闰年，那么，当 month 是 2 月之后的月份，ans 加上一天。

（5）输出第几天 ans 的值。

程序如下：

```
1  //exam5.4
2  #include<iostream>
3  using namespace std;
4  int year,month,date,ans;
5  int main()
6  {
7      int a[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
                              // 初始化数组 a
8      cin>>year>>month>>date;
9      for(int i=1;i<month;i++) ans+=a[i];
                              // 求 month 之前月份天数之和
10     ans+=date;             // 加上 date
11     if(year%4==0 and year%100!=0 or year%400==0)
                              // 是否闰年
12       if(month>2)ans++; // 闰年及 2 月之后的月份，ans 加上一天
13     cout<<ans;
14     return 0;
15 }
```

思考：程序中第 7 行初始化数组 a 时，为什么多了一个 0？

【例 5.5】下面两程序没有初始化数组，观察程序默认的数组变量初值。

程序1：

```
1  //exam5.5-1
2  #include<iostream>
3  using namespace std;
4  int a[5];
5  int main()
6  {
7    for(int i=0;i<5;i++)
8     cout<<a[i]<<"";
9    return 0;
10 }
```

运行结果：

```
0 0 0 0 0
```

程序2：

```
1  //exam5.5-2
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int a[5];
7      for(int i=0;i<5;i++)
8       cout<<a[i]<<"";
9      return 0;
10 }
```

运行结果：

```
2293592 4252834 4252740 302692880 2292112
```

说明：程序1和程序2的区别在于数组定义放在int main()之外与之内，如果数组定义放在int main()之内，其初始值是随机的。在解决具体问题的程序设计中，需要重视变量初值的设置，否则，很容易造成不易查找的错误。

【例5.6】写出下面程序的运行结果。

程序1：

```
1  //exam5.6-1
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int a[5]={1,2};
7      for(int i=0;i<5;i++)
8      cout<<a[i]<<"";
9      return 0;
10 }
```

运行结果：

```
1 2 0 0 0
```

说明：程序1中只给a[0]、a[1]赋初值，但后面的a[2]~a[4]元素自动赋0值。

思考：结合程序1，当给数组的部分元素赋初值后，其他元素的初值会如何变化呢？

程序 2：

```
1  //exam5.6-2
2  #include<iostream>
3  #include<cstring>
4  using namespace std;
5  int main()
6  {
7      int a[5];
8      memset(a,0,sizeof(a));
9      for(int i=0;i<5;i++)
10       cout<<a[i]<<"";
11     return 0;
12 }
```

运行结果：

```
0 0 0 0 0
```

说明： 程序中用了 memset 函数给数组变量赋初值。使用 memset 函数需要 #include<cstring> 头文件。

思考：

（1）本程序使用 memset 函数设置所有数组变量初值为 0。还有什么方法可以设置所有数组变量初值为 0？

（2）memset 函数使用时需要注意什么？

5.1.3　数组越界

C++ 语言规定，使用数组时，要注意：

（1）数组元素的下标值为正整数。

（2）在定义元素个数的下标范围内使用。

然而，当在程序中把下标写成负数、大于数组元素的个数时，程序编译的时候是不会出错的。例如：

```
int a[10];
a[-3]=5;
a[20]=15;
a[10]=20;
int k=a[30];
```

这些语句的语法是正确的，能够通过程序的编译。然而，它们要访问

的数组元素并不在数组的存储空间内，这种现象叫**数组越界**。

【例5.7】编译运行下面程序，说出下面程序的错误。

```
1  //exam5.7
2  #include<iostream>
3  using namespace std;
4  main()
5  {
6       int i;
7       int array[10];
8       for(i=1;i<=15;i++)
9       {
10           array[i]=0;
11           cout<<i;
12      }
13      return 0;
14 }
```

说明：该程序能够通过编译，也能运行出结果，但程序是无限循环地输出。程序的问题是定义array[10]，使用时数组下标超过了9，然而，出现的问题却是另一种形式。

C++语言中，数组越界访问系统时不会给出任何的提示，也就是说，程序可以超出数组边界进行读/写，从而造成内存的混乱。

数组越界是实际编程中常见的错误，而且这类错误往往难以捕捉。因为越界语句本身并不一定导致程序立即出错，可能在遇到某些数据时才导致错误，有时由于越界，意外地改变了变量或指令，导致在调试器里调试的时候，程序不按照应当的次序运行的怪现象。

发现程序中有否数组越界没有什么好的办法，需要在程序编写时特别的注意，同时，程序编写完成后，加强静查。所谓“**静查**”，即认真阅读程序是否按照设计的要求编写。

练　习

阅读程序写出结果。

```
//test(1)-1
```

```
#include<iostream>
#include<cstring>
using namespace std;
const int SIZE = 100;
int main()
{
   int n,i,sum,x,a[SIZE];
   cin>>n;
   memset(a,0,sizeof(a));
   for(i=1;i<=n;i++){
     cin>>x;
     a[x]++;
    }
   i=0;
   sum=0;
   while(sum<(n/2+1)){
     i++;
    sum+=a[i];
    }
   cout<<i<<endl;
   return 0;
}
```

输入：

```
 11
 4 5 6 6 4 3 3 2 3 2 1
```

输出：

```
//test(2)-2
#include<iostream>
using namespace std;
int main()
   {
     cout<<a[a[1]*a[2]]<<endl;
   }
```

输出：

5.2　活用数组下标

【例 5.8】输入 n 个整数，存放在数组 a[1] 至 a[n] 中，输出最大数所在位

置（n≤10000）。

输入样例：

5

67 43 90 78 32

输出样例：

3

分析：设maxa存放最大值，k存放对应最大值所在的数组位置，maxa的初值为a[1]，k的初值对应为1，枚举数组元素，找到比当前maxa大的数成为maxa的新值，k值为对应位置，输出最后的k值。

程序如下：

```
1  //exam5.8
2  #include<iostream>
3  using namespace std;
4  const int MAXN=10001;
5  int main()
6  {
7      int a[MAXN];                  //定义10001个数组元素
8      int i,n,maxa,k;
9      cin>>n;
10     for(i=1;i<=n;i++)             //读入n个存储到a[1]~a[n]中
11       cin>>a[i];
12     maxa=a[1];                    //赋最大值初值和初始位置
13     k=1;
14     for(i=2;i<=n;i++)             //枚举数组，找到最大数和位置
15         if (a[i]>maxa)
16         {
17           maxa=a[i];
18           k=i;
19         }
20   cout<<k;                        //输出最大数所在数组中的位置
21   return 0;
22 }
```

运行结果：

```
8
12 32 57 98 12 34 45 76
4
```

说明： 程序中第 17、18 行表示找到当前最大数所在的位置。这种思路可以帮助我们确定需要数据所在的位置。

【例 5.9】 有 n 个人，编号为 1~n。开始时，所有人都站着，接着第 2 个人及 2 的倍数的人坐下，然后，第 3 个人及 3 的倍数的人按相反的操作（站的人坐下，坐的人站起来），依此类推，一共操作到第 k 人及 k 的倍数，问最后哪些人站着？输入 n 和 k，输出站着人的编号（k ≤ n ≤ 10000）。

输入样例：

7 3

输出样例：

1 5 6 7

分析： 用 a[1],a[2],……,a[n] 表示编号为 1,2,3,……,n 的人是否站着，初值都为 0，表示都站着，让 i 从 2 到 k 循环，将 a 数组中 i 倍数的元素值取反。输出 a 数组中值为 0 的元素。

程序如下：

```
1  //exam5.9
2  #include<iostream>
3  #include<cstring>
4  const int MAXN=10001;
5  using namespace std;
6  int main()
7  {
8      int a[MAXN];                //0 表示站着，1 表示坐着
9      int n,k;
10     int i,j;
11     memset(a,0,sizeof(a));      // 数组初始化为 0，表示站着
12     cin>>n>>k;                  // 读入数据
13     for(i=2;i<=k;i++)           //i 表示执行到第 i 个人的操作
14       for(j=1;j<=n;j++)         //j 表示第 j 个人
15         if(j%i==0)a[j]=!a[j];//j 是 i 的倍数，则执行相反的操作
16     for(i=1;i<=n;i++)           // 输出最后站着的人
17       if(!a[i])
18        cout<<i<<"";
19     return 0;
20 }
```

说明： 程序中第 15、17、18 行表示借助数组实现操作过程达到找到

结果的目的。这种思路可以帮助我们通过数组模拟实现问题情境的操作过程而获得操作后的最终结果。

【例5.10】学校推出了10名歌手，校学生会想知道这10名歌手受欢迎的程度，设了一个投票箱，让每一个同学给自己喜欢的歌手投票，为了方便，学生会把10个歌手用1~10进行编号，这样，同学们只要用编号进行投票了。现在，学生会找到你，帮助统计一下每个歌手获得的票数。

分析：很直观的想法是，投谁，谁的票数加1。如：投2号，则2号的票数加1，投6号，6号的票数加1。如何表示这么一个操作过程呢？

设变量num[i]表示第i个歌手的票数，相当于用名称num表示票数，然后再对名称进行编号，编号为i的num表示第i个歌手的票数。

这样，投2号，则2号的票数加1，就可以表达为：

```
i=2;num[i]=num[i]+1
```

投6号，则6号的票数加1，就可以表达为：

```
i=6;num[i]=num[i]+1
```

归纳上述分析，具体实现步骤如下：

（1）开辟num[1]~num[10]变量分别存放10个歌手的票数。

（2）读入选票赋给i。

（3）对应选票i的歌手票数加1，即num[i]=num[i]+1。

（4）重复（2）和（3）的操作，直到读完选票为止。

（5）输出每位歌手的票数。

程序如下：

```
1  //exam5.10
2  #include<iostream>
3  #include<cstring>
4  using namespace std;
5  int main()
6  {
7      int num[11];
8      int i;
9      memset(num,0,sizeof(num)); // 数组初始化为 0
10     while(cin>>i)              // 读入选票
11       num[i]=num[i]+1;         //i 表示 i 号歌手，num[i] 表示
                                    i 号歌手的选票数
12     for(i=1;i<=10;i++)         // 输出各位歌手的最终选票数
```

```
13        cout<<" 第 "<<i<<" 号歌手的选票数为：  "<<num[i]<<endl;
14     return 0;
15 }
```

说明： 程序中的第 11 行充分利用数组下标达到统计歌手的选票数的目的。这种思路可以帮助我们明确数组变量和数组下标变量对应问题中的实际意义，更便捷解决问题。

【例 5.11】输入 n 个数，存入数组 a 中，每一个数都是介于 0 到 k 之间的整数，此处 k 为某个整数（n ≤ 100000，k ≤ 1000），按从小到大的顺序输出 a 数组中的数据。

分析： 看了问题，我们直观的想法是将 a 数组中的数据实现从小到大的顺序排序。排序的方法有多种，效率也不一样。但本问题有一个重要的特点就是每一个数都是介于 0 到 k 之间的整数，而且 k 的值很小，我们可以开设一个下标为 0~k 的数组 c，c[0] 记录 a 数组中值为 0 的个数，c[1] 记录 a 数组中值为 1 的个数，……，c[k] 记录 a 数组中值为 k 的个数，那么按从小到大的顺序输出个数为 c 数组元素值的 c 数组下标值，即按从小到大的顺序输出 a 数组中的数据。实现步骤如下：

（1）读入数据存放在 a 数组中，设数组 c，c[x] 存放值为 x 的个数。

（2）对于 a 数组中的 n 个数，统计值为 a[i] 的个数，c[a[i]]=c[a[i]]+1。

（3）循环 i：0~k，输出 c[i] 个 i。

程序如下：

```
1  //exam5.11
2  #include<iostream>
3  #include<cstring>
4  using namespace std;
5  const int MAXN=100010;
6  const int K=1001;
7  int main()
8  {
9      int a[MAXN], c[K];
10     int n;
11     cin>>n;
12     memset(c,0,sizeof(c));    // 数组 c 赋初值 0
13     for (int i=0;i<n;i++)     // 读入数据存放在 a 数组，并用数
                                    组 c 统计 a[i] 的个数
```

```
14      {
15        cin>>a[i];
16        c[a[i]]=c[a[i]]+1;
17      }
18      for (int i=0;i<K;i++)     //输出排序后结果
19        for (int j=1;j<=c[i];j++)
20          cout<<i<<"";
21      return 0;
22  }
```

运行结果：

```
10
2 3 1 2 4 55 3 55 3 2
1 2 2 2 3 3 3 4 55 55
```

说明： 程序中第16行表示把a数组元素按其值归类存放，第19、20行表示按类输出。这种思路可以帮助我们学会如何利用数组值与下标变量实现对数组内容的整理，这种排序方法称为计数排序。

【*例5.12[1)]】给你n根火柴棍，你可以拼出多少个形如“A+B=C”的等式？等式中的A、B、C是用火柴棍拼出的整数（若该数非零，则最高位不能是0）。用火柴棍拼数字0~9的拼法如图5.1所示。

输入整数n（n<=24），输出能拼成的不同等式的数目。



图5.1 火柴棍0~9示意图

注意：

（1）加号与等号各自需要两根火柴棍。

（2）如果A≠B，则A+B=C与B+A=C视为不同的等式（A、B、C>=0）。

（3）n根火柴棍必须全部用上。

输入样例：

18

输出样例：

9

1）本题选自NOIP2008提高组试题。

输出样例解释：

0+4=4

0+11=11

1+10=11

2+2=4

2+7=9

4+0=4

7+2=9

10+1=11

11+0=11

分析：设等式中的三个变量为A、B、C，如果已知A和B，则C=A+B。由于题目给出的火柴数目范围很小，可以穷举A、B值，求得C值，求出构成等式A+B=C的火柴数目，如果火柴数目为输入值，则为问题的一个解，累计解的个数即为本问题的最终答案。

对于本题，我们能否从细节上提高穷举的效率呢？

（1）根据n<=24条件，确定穷举A和B的范围。即如果都使用最少火柴的数字（如数字1）构造表达式中的数，得到最大的数是多少，可以算出该数<1000，因此A和B的范围为0~1000。

（2）在穷举表达式过程中，需要反复计算每个数所用的火柴数目，即相同的数有可能重复计算许多次的火柴数目，显然，需要想办法减少重复计算。先进行预处理，将估算范围内的数所用的火柴数目算出。

（3）穷举过程中如果当前数的火柴数目超过输入限制做剪枝处理。

归纳上述分析，具体实现步骤如下：

（1）预处理：求0~2000数对应的火柴数目。

（2）使用循环枚举A值0~1000，执行下列操作：

①判断A值的火柴数目是否超过限制值，是的，剪枝，跳过该数的处理。

②否，使用循环穷举B值0~1000，执行下列操作：

- 判断A和B值的火柴数目和是否超过限制值，是的，剪枝，跳过该数的处理。
- 否，求C值，若A、B、C值火柴数目和等于限制值，计数方案数。

（3）输出方案数。

程序如下：

```
1  //exam5.12
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      int a[10]={6,2,5,5,4,5,6,3,7,6},ans=0,temp=0,k;
7      int num[2016];
8      int n;
9      cin>>n;
10     num[0]=6;
11     for(int i=1;i<=2000;i++) // 预处理 0~2000 数对应的火柴数目
12     {
13         k=i;
14         while(k)              // 求数 i 的火柴个数
15         {
16             temp+=a[k%10];
17             k/=10;
18         }
19         num[i]=temp;          //num[i] 存放数 i 的火柴个数
20         temp=0;
21     }
22     for(int i=0;i<=999;i++)                  // 枚举 A
23       for(int j=0;j<=999;j++)                // 枚举 B
24        {
25         if(num[i]+num[j]>=n)
26             continue;    // 若 A、B 火柴数目和超过限制值，剪枝
27         else
28         {
29         if(num[i+j]+num[i]+num[j]+4==n)  //C 为 A+B 即 i+j
30         ans++;  // 若 A、B、C 值火柴数目和等于限制值，计数方案数
31         }
32        }
33     cout<<ans;
34     return 0;
35 }
```

说明： 程序中的第 19、29 行表示数组下标变量代表数字，数组名代表数字对应的火柴数，利用这样的表示可以方便地同时表达 A+B=C 和等式火柴数和满足问题的条件。这种思路可以帮助我们学会如何妙用数组下标变量。

【*例 5.13】给出一串长度为 n 的数列，要求从中找出连续的一段来使得总和最大。输入包含两行，第 1 行表示数列长度为 N（N <= 100000），第 2 行包括 N 个整数来描述这个数列，每个整数的绝对值不超过 1000。输出只有一个整数，为最大的连续段总和。

输入样例：

```
5
1 -2 3 1 -4
```

输出样例：

```
4
```

分析：求“最大连续段和”是一个比较基础的问题，可以用多种方法解决，下面讨论三种算法。

方法 1：穷举法，采用三重循环穷举子段起点位置、终点位置、求子段和，比较子段和求最大值。

程序如下：

```
1  //exam5.13-1
2  #include<iostream>
3  const int MAXN=100001;
4  using namespace std;
5  int main()
6  {
7      int a[MAXN];
8      int n,i,j,k;
9      int maxsum;                  //子段和最大值
10     int temp;                    //子段和
11     cin>>n;
12     for(i=1;i<=n;i++)            //输入数据
13       cin>>a[i];
14     maxsum=a[1];
15     for(i=1;i<=n;i++)            //子段起始位置
16       for(j=i;j<=n;j++)          //子段终点位置
17        {
18          temp=0;
19          for(k=i;k<=j;k++)      //求子段和
20            temp=temp+a[k];
21          if(maxsum<temp) maxsum=temp;
22        }
23     cout<<maxsum;
```

```
24      return 0;
25 }
```

方法2：方法1中，在求子段和过程中有许多重复的操作，如：求第3个数到第5个数的和包含在求第2个数到第5个数和之中，显然，重复进行了求和操作。如何解决呢？

解决方法是通过预处理，求出从第1位置到每个位置的和s[i]，而s[i]值等于s[i-1]值加当前位置值，减少了重复求和运算，对于子段[i,j]的子段和则为s[j]-s[i]。

这样，用两重循环穷举子段起点位置i与终点位置j，求得所有子段和，比较子段和求最大值。

程序如下：

```
1  //exam5.13-2
2  #include<iostream>
3  #include<cstring>
4  const int MAXN=100001;
5  using namespace std;
6  int main()
7  {
8      int s[MAXN],a[MAXN];
9      memset(s,0,sizeof(s));
10     int n,i,j;
11     int maxsum;                  // 子段和最大值
12     int temp;                    // 子段和
13     cin>>n;
14     for(i=1;i<=n;i++)            // 输入数据
15       {
16         cin>>a[i];
17         s[i]=a[i]+s[i-1];        // 预处理前缀和
18       }
19     maxsum=a[1];           //maxsum值初始化为数组第一个元素
20     for(i=1;i<=n;i++)            // 子段起始位置
21        for(j=i;j<=n;j++)         // 子段终点位置
22        {
23          temp=s[j]-s[i-1];
24          if(maxsum<temp) maxsum=temp;
25        }
26     cout<<maxsum;
```

```
27      return 0;
28 }
```

方法 3：设 a[i] 为以第 j 个位置为结尾的最大子段和，若 a[i−1] 大于 0，显然，a[i]=a[i−1]+ 当前数，因为，与和大于 0 连续序列构成的序列和一定比本身数构成的序列和大，如果 a[i−1] 小于 0，则 a[i]= 当前数，因为，当前数大于当前数加上一个负数。

这里应用了一个贪心思想，当前数 + 正数 > 当前数，当前数 + 负数 < 当前数。通过求从第 1 个到第 n 个数以本身数为结尾的最大子段和，即枚举了所有数最大子段和。

由于问题只求最大子段和，在求解过程能够求得以当前数为结尾的最大子段和，然后比较记下其中的最大值即可。

设连续子段和为 t，则：

$$\text{当前 t}=\begin{cases} \text{t+a[i]} & \text{t>=0（之前）} \\ \text{a[i]} & \text{t<0（之前）} \end{cases}$$

具体实现步骤如下：

（1）读入序列中的 n 个数存放在 a 数组中。

（2）初值设置，第 1 个数的子段和 t=a[1]，最大子段和 ans=a[1]。

（3）一重循环枚举第 2 个数到第 n 个数，执行下列操作：

①求以当前数为结尾的最大子段和 t。

②如果 t>ans，刷新 ans 值。

（4）输出 ans 值。

程序如下：

```
1  //exam5.13-3
2  #include<iostream>
3  const int MAXN=100001;
4  using namespace std;
5  int main()
6  {
7       int a[MAXN];
8       int n,i;
9       int ans;                    //子段和最大值
10      int t;                      //以当前数为结尾的最大子段和
11      cin>>n;
12      for(i=1;i<=n;i++)           //输入数据
13          cin>>a[i];
```

```
14      ans=a[1];
15      t=ans;
16      for(i=2;i<=n;i++)
17      {
18         if(t>=0)
19              t=t+a[i];
20         else
21              t=a[i];
22         if(t>ans)ans=t;
23      }
24       cout<<ans;
25       return 0;
26 }
```

说明：求子段和看上去是一个非常普通的问题，三种解决问题的方法得到不同的解决问题的效率。这种思路可以帮助我们学会分析数组中数据性质，学会如何利用数组表示问题的关键的数据而得到更有效的解决问题效率。

练　习

（1）输入n个数，编程将数组中的最小值放到数组第1个位置，数组元素值不变。

（2）围绕着山顶有10个洞，一只狐狸和一只兔子住在各自的洞里。狐狸想吃掉兔子。一天，兔子对狐狸说：“你想吃我有一个条件，先把洞从1~10编上号，你从10号洞出发，先到1号洞找我；第二次隔1个洞找我，第三次隔2个洞找我，以后依次类推，次数不限，若能找到我，你就可以饱餐一顿。不过在没有找到我以前不能停下来。”狐狸满口答应，就开始找了。它从早到晚进了1000次洞，累得昏了过去，也没找到兔子，请问，兔子躲在几号洞里？

（3）有52张扑克牌，使它们全部正面朝上。从第2张牌开始，把凡是2的倍数位置上的牌翻成正面朝下；接着从第3张牌开始，把凡是3的倍数位置上的牌正面朝上的翻成正面朝下，正面朝下的翻成正面朝上；接着从第4张牌开始，把凡是4的倍数位置上的牌按此规律翻转；依次类推，直到第1张要翻的牌是第52张为止。统计最后有几张牌正面朝上，并打印出

它们的位置。

（4）[1]校大门外长度为L的马路上有一排树，每两棵相邻的树之间的间隔都是1m。我们可以把马路看成一个数轴，马路的一端在数轴0的位置，另一端在L的位置，数轴上的每个整数点，即0，1，2，……，L，都种有一棵树并且已知每棵树的高度。

由于马路上有一些区域要用来建地铁。这些区域用它们在数轴上的起始点和终止点表示。已知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。现在要把区域涉及的最小起始点和最大终止点连成的一片树（包括区域端点处的两棵树）移走，然而规定连成的一片长度不能超过100m，也就是说如果连成一片长度超过100m需要修改最大终止点坐标。你的任务是设置一个100个元素的数组，存放移走这片树的每棵树的高度并且输出。

输入格式：第1行有两个整数L（1<=L<=10000）和M（1<=M<=100），L代表马路的长度，M代表区域的数目，L和M之间用一个空格隔开。接下来是L棵树的高度，接下来的M行每行包含两个不同的整数，用一个空格隔开，表示一个区域的起始点和终止点的坐标。

输出格式：1行，移走的每棵树的高度。

输入样例：

```
9 1
1 2 3 4 5 6 7 8 9 10
3 5
```

输出样例：

```
4 5 6
```

*（5）求所有的满足等式x^2=□□□□□□□□□的x，每个□内的数字互不相同，且遍历数字1~9。

5.3　数值排序和查找

5.3.1　数值排序

在现实生活中有各种各样的排序问题，有各种各样实现排序的方法，不同排序方法其效率不一样。我们先来看几种生活中自然而然用的排序方

1）本题改编自NOIP2005普及组复赛试题。

法和程序实现。

【例5.14】输入n个数，将n个数按从小到大的顺序输出（n<=100000）。

输入样例：

5

156 163 178.6 198 123

输出样例：

123 156 163 178.6 198

1. 选择排序

在第3章的例3.19中，我们实现了对三个数按从大到小的顺序输出。如果要对n个数实现排序，需要用到数组。为了方便理解，假设这n个数据是n个人的身高，排序即对这n个人按高矮顺序排队，沿用例3.19方法2的思路，第1次，从队列中选择最矮的一个人与站在第1个位置上的人交换位置，则第1位置上的人为最矮，接下来的排序就不用去管第1人了，即n个人的队列排序问题转换为n-1个人的队列排序问题，第2次，从第2~n人中选择最矮的一个人与站在第2个位置上的人交换位置，n-1个人的队列排序问题转换为n-2个人的队列排序问题，……，第n-1个和第n个人进行比较、交换。经过n-1次选择、交换后，原队列就形成了有序的队列。这种排序方式称为选择排序。

归纳上述分析，具体实现步骤如下：

（1）读入数据存放在a数组中。

（2）在a[1]~a[n]中选择值最小的元素，与第1位置元素交换，则把最小值元素放入a[1]中。

（3）在a[2]~a[n]中选择值最小的元素，与第2位置元素交换，则把第2大值元素放入a[2]中，……

（4）直到第n-1元素与第n个元素比较排序为止。

程序实现方法：用两层循环完成算法，外层循环i控制当前序列最小值存放的数组位置，内层循环j控制从i+1到n序列中选择最小的元素所在位置k。

选择排序算法对数组元素需要进行约n*n次遍历。

```
1  //exam5.14-1
2  #include<iostream>
3  const int MAXN=100001;
```

```
4  using namespace std;
5  int main()
6  {
7      int n,k,i,j;
8      float temp,a[MAXN];
9      cin>>n;
10     for(i=0;i<n;i++)
11     cin>>a[i];
12     for(i=0;i<n;i++)      //i 控制当前序列最小值存放的数组位置
13     {
14       k=i;
15     for(j=i+1;j<n;j++)    //j 控制从 i 之后到 n 序列中选择最小的
                               元素所在位置 k
16       if(a[k]>a[j]) k=j;
17     if(k!=i)
18     {
19         temp=a[i];
20         a[i]=a[k];
21         a[k]=temp;
22     }                     // 交换 a[i] 和 a[k] 将当前最小值放到
                               a[i] 位置
23     }
24     for(i=0;i<n;i++)
25       cout<<a[i]<<"";
26     return 0;
27 }
```

说明：程序第 16 行实现找到数据位置，第 19、20、21 实现把数据交换到对应的位置上。这里用到了遍历数组寻找满足条件数据的方法，用到了第 3 章例题中学到的数据交换的方法。

2. 冒泡排序

冒泡排序的思想：以 n 个人站队为例，从第 1 个开始，依次比较相邻的两个人是否逆序对（高在前矮在后），若逆序就交换这两人，即第 1 个和第 2 个比，若逆序，交换两人，接着第 2 个和第 3 个比，若逆序，交换两人，接着第 3 个和第 4 个比，若逆序，交换两人，……，直到 n–1 和 n 比较，经过一轮比较后，则把最高的人排到最后，即将最高的人像冒泡一样逐步冒到相应的位置。原 n 个人的排序问题，转换为 n–1 个人的排序问题。第二轮从第 1 个开始，依次比较相邻的两个人是否逆序对，若逆序就交换这两

人，直到n-2和n-1比较。如此，进行n-1轮后，队列为有序的队列。

从上述分析中可以看出，每进行一轮的比较之后，n个数的排序规模就转化为了n-1个数的排序规模。

归纳上述分析，具体实现步骤如下：

（1）读入数据存放在a数组中。

（2）比较相邻的前后两个数据，如果前面数据大于后面的数据，就将两个数据交换。

（3）对数组的第0个数据到n-1个数据进行一次遍历后，最大的一个数据就“冒”到数组第n-1个位置。

（4）n=n-1，如果n不为0就重复前面二步，否则排序完成。

程序实现方法：用两层循环完成算法，外层循环i控制每轮要进行多少次的比较，第一轮比较n-1次，第2轮比较n-2次，……，最后一轮比较1次。内层循环j控制每轮i次比较相邻两个元素是否逆序，若逆序就交换这两个元素。

冒泡排序程序1：

```
1  //exam5.14-2
2  #include<iostream>
3  const int MAXN=100001;
4  using namespace std;
5  int main()
6  {
7       float temp,a[MAXN];
8       int n,i,j;
9       cin>>n;                    //读入n
10      for(i=0;i<n;i++)           //读入数据
11         cin>>a[i];
12      for(i=n-1;i>=1;i--)        //进行n-1轮冒泡
13      for(j=0;j<i;j++)           //每轮进行i次的比较
14        if(a[j]>a[j+1])          //相邻元素进行比较，如果大的在
                                     前面，则交换
15        {
16           temp=a[j];
17           a[j]=a[j+1];
18           a[j+1]=temp;
19        }
20      for(i=0;i<n;i++)           //输出排序结果
```

```
21        cout<<a[i]<<"";
22        return 0;
23 }
```

思考：冒泡排序程序中，当数据在不断冒泡调整过程中，可能使得很多数据已经有序了，但还需要继续比较。对于有些序列，经过几轮比较交换后就已经有序了，但需要进行 n−1 轮的比较交换。为此，能否对其进行优化呢？

3．插入排序

插入排序思想：回忆一下打牌时抓牌的情景，为了方便打牌，抓牌时一般一边抓牌一边按花色和大小插入恰当的位置，当抓完所有的牌时，手中的牌便是有序的。这种排序方法即插入排序。

当读入一个元素时，在已经排序好的序列中，搜寻它正确的位置，再放入读入的元素。但不该忽略一个重要的问题：在插入这个元素前，应当先将它后面的所有元素后移一位，以保证前面处理过的元素不被覆盖。

归纳上述分析，具体实现步骤如下：

（1）读入数据存放在 a 数组中。

（2）从第 2 个数开始，取出当前数作为待排序数，逐个与前面的数比较，若小于前面数，则前面数后移，当大等于前面数时，插入当前空出的位置。

（3）直到第 n 个数插入正确位置为止。

程序实现方法：用两层循环完成算法，外层循环 i 控制待排序的数，从第 2 个数到第 n 个数，内层循环 j 控制寻找插入的位置，j 值从 i−1 开始向前扫描，边扫描边将数据后移，寻找到位置，插入当前值。

```
1  //exam5.14-3
2  #include<iostream>
3  const int MAXN=100001;
4  using namespace std;
5  int main()
6  {
7       float a[MAXN];
8       int n,i,j,k,temp;
9       cin>>n;
10      for(i=0;i<n;i++)
11      cin>>a[i];
12      for(i=0;i<n;i++)
```

```
13      {
14        for(j=i-1;j>=0;j--) //为a[i]在前面的有序区间中找一个
                                合适的插入位置
15        if(a[j]<a[i])break; //找到比a[i]数据小的位置，退出，
                                插入其后
16        if(j!=i-1)
17        {
18          temp=a[i];          //将比a[i]大的数据向后移
19          for(k=i-1;k>j;k--)
20          a[k+1] = a[k];
21          a[k+1] = temp;      //将a[i]放到正确位置上
22        }
23       }
24       for(i=0;i<n;i++)
25         cout<<a[i]<<"";
26       return 0;
27 }
```

说明：程序第18、19、20行提供数据批量位移方法。

5.3.2 查找

在现实生活中需要各种各样的查找。我们先来看，在一批数据中查找某一数的问题。

1. 顺序查找

【例5.15】在输入的n个数中，查找输入的数，输出数据存放在数组中的位置，若查找不到，则输出“fail!”（n<=100000）。

分析：设数据存放在a数组中，对输入的查找数，在数组中一个一个比对，若相等，则输出数组下标，若找不到，则输出“fail!”。

程序如下：

```
1  //exam5.15
2  #include<iostream>
3  const int maxn=100001;
4  using namespace std;
5  int main()
6  {
7      int n,k,i,j;
8      bool find;
9      float num,a[maxn];
```

```
10      cin>>n;
11      for(i=0;i<n;i++)
12      cin>>a[i];
13      while(true)                          // 重复循环
14      {
15        cin>>num;                          // 读入要查找的数
16        find=0;
17        for(i=0;i<n;i++)                   // 在数组中查找
18        if(a[i]==num)
19        {
20          cout<<i<<"";                     // 输出找到数的位置
21          find=1;
22        }
23       if(!find)cout<<"fail!"<<endl;       // 查找不到
24       }
25   return 0;
26 }
```

运行结果：

```
8
-5 67 8.9 90 5 23 5 9
5
4 6
-2
fail!
```

说明：程序第 17、18、20 行实现按顺序在数组中查找，这种按顺序查找的方法称为顺序查找。顺序查找的特点是对任意序列都可实现查找，然而，当序列很大时，查找次数很多，显然有许多重复的操作，查找效率低。

2. 二分查找

【例 5.16】这是一个猜数游戏，已知一个 8000 以内的正整数，让你去猜这个数，然后告诉你所猜的答案太大还是太小，直到你猜到为止。你会怎样去猜，使得猜的次数尽量少呢？

分析：一般地，我们会想到用二分的方法，如：要猜的数是 625。猜数过程如下表所示。

次数	1	2	3	4	5	6
猜数	4000	2000	1000	500	750	625
提示	太大	太大	太大	太小	太大	正确

二分猜数过程，实际上是在0，1，2，3，……，8000有序数内查找值为625的数，之所以能够用二分，是因为一个数要么在中间数的左边，要么在右边，如果在中间数的左边，那么，新的查找区间为0~中间数而与右边无关，继续在新区间中二分，可以看到，二分查找算法的时间复杂度为$O(Log_2(N))$。

3. 二分查找算法

如果一批数据已经是有序的数据，那么，在这批数据中查找某一数据就可以用上述的二分思想实现查找。

二分查找算法描述如下：

（1）需要设置三个指针low、high、mid表示查找区间的左端点、右端点和中间位置。

（2）当low <= high时，二分查找：

①求mid = (low + high)/2，需要特别注意，mid设置的数据类型不能出现(low + high)数值越界问题；

②如果目标值大于mid单元值，那么low = mid + 1，继续二分；

③如果目标值小于mid单元值，那么high = mid – 1，继续二分；

④如果目标值等于mid单元值，返回查找结果并退出查找。

（3）返回没有查到。

例：在有序序列[05 13 19 21 37 56 64 75 80 88 92]中查找数据21和85。

①查找数据21。

```
[05  13  19  21  37  56  64  75  80  88  92]
 ↑                   ↑                   ↑
low                 mid                 high
[05  13  19  21  37] 56  64  75  80  88  92
 ↑        ↑        ↑
low      mid      high
 05  13  19 [21 37]  56  64  75  80  88  92
             ↑↑  ↑
         low mid high
            查找成功
```

②查找数据85。

```
[05  13  19  21  37  56  64  75  80  88  92]
↑                   ↑                   ↑
low                 mid                 high
 05  13  19  21  37  56 [64  75  80  88  92]
                         ↑       ↑       ↑
                        low     mid     high
 05  13  19  21  37  56  64  75  80 [88  92]
                                   ↑↑   ↑
                                  low mid high
 05  13  19  21  37  56  64  75  80 [88]  92
                                   ↑↑↑
                                  low mid high
                                 查找失败
```

【例 5.17】中考成绩出来了，许多考生想知道自己成绩排名情况，于是考试委员会找到了你，让你帮助完成一个成绩查找程序，考生只要输入成绩，即可知道其排名及同分数的人有多少。

输入格式：第 1 行一个数 N(N<=10000)；第 2 行一个数 K；第 3 行开始 N 个以空格隔开的从大到小排列的所有学生中考成绩(整数)。接着 K 个待查找的考生成绩。

输出格式：K 行，每行为一个待查找的考生的名次（不同分数的名次）、同分的人数、比考生分数高的人数。查找不到输出“fail!”。

输入样例：

```
10
2
580 570 565 564 564 534 534 534 520 520
564 520
```

输出样例：

```
4 2 3
6 2 8
```

分析：本问题有两个特点，一是给出的原始数据是有序的，因此，查找时可以用二分查找；二是有大量的相同分数，可以利用这一特点节约存储空间，每种分数只保存一个值。由于需要输出待查找的考生的名次、同

分的人数、比考生分数高的人数，因此，在读入数据时要考虑，同分数据及高于当前分数据的同时存储与处理。

归纳上述分析，具体实现步骤如下：

（1）设数组a用于存放不同的分数值，b用于存放相同分数的人数，s用于存放高于此分数的人数。数组下标表示名次。

（2）从高到低读入分数，当读入数与前一个数相同时，对应名次相同分数人数增1，当读入数与前一个数不同时，名次增1，即数组下标增加1，存储当前分数，求高于本分数的人数并存储。

（3）读入需要查找的成绩x，二分查找成绩。

（4）查找到，输出名次、相同分数的人数和高于本分数的人数；查找不到，输出查找不到该分数。

程序如下：

```
1  //exam5.17
2  #include<iostream>
3  #include<cstring>
4  const int MAXN=10010;
5  using namespace std;
6  int main()
7  {
8       int a[MAXN],b[MAXN],s[MAXN];
9       int mark,temp=-1,n,k,x,low,high,mid=0,rank=0;
10      cin>>n>>k;
11      memset(b,0,sizeof(b));
12      for(int i=1;i<=n;i++)    // 预处理
13      {
14         cin>>mark;
15         if(mark==temp)b[rank]+=1;
                 // 当读入数与前一个数相同时，对应名次相同分数人数增 1
16         else                  // 当读入数与前一个数不同时
17        {
18         a[++rank]=mark;       // 名次增一并存储当前分数
19         s[rank]=i-1;          // 求高于本分数的人数并存储
20         b[rank]=1;            // 将自己统计入同分人数
21        }
22         temp=mark;
23     }
24     for(int i=1;i<=k;i++)
```

```
    {
        cin>>x;
        low=1;                              // 二分查找指针初值
        high=rank;
        while(low<=high&&a[mid]!=x)   // 二分查找
        {
          mid=(low+high)/2;
          if(a[mid]<x) high=mid-1;
          else low=mid+1;
        }
        if(a[mid]==x)cout<<mid<<""<<b[mid]<<""<<s[mid]<<endl;
        else cout<<"fail"<<endl;
    }
    return 0;
}
```

说明：程序只是实现在有序序列中采用二分查找算法快速查找数据。二分是一个很有用的思想，在后面学习中我们会遇到许多二分思想的灵活应用。

思考：为什么二分查找比顺序查找效率高，分析两种查找方法适用的场合。

练　习

（1）使用冒泡排序对序列进行升序排列，每执行一次交换操作将会减少 1 个逆序对，因此序列 5，4，3，2，1 需要执行（　）次操作，才能完成冒泡排序。

（A）0　　（B）5　　（C）10　　（D）15

（2）设有 100 个数据元素，采用二分查找时，最大比较次数为（　）。

（A）6　　（B）7　　（C）8　　（D）10

（3）将数组 {8,23,4,16,77,-5,53,100} 中的元素按从大到小的顺序排列，每次可以交换任意两个元素，最少需要交换（　）次。

（A）4　　（B）5　　（C）6　　（D）7

（4）对有序数组 {5,13,19,21,37,56,64,75,88,92,100} 进行二分查找，成功查找元素 19 的查找长度（比较次数）是（　）。

（A）1　　（B）2　　（C）3　　（D）4

（5）阅读下面程序，写出运行结果。

```
//test(5)-1
#include<iostream>
#include<cstring>
using namespace std;
#define maxn 1000
#define maxnum 10000
int main()
{
     int n,k=0;
     int array[maxn],c[maxnum];
     memset(c,0,sizeof(c));
     cin>>n;
     cout<<" 请输入 "<<n<<" 个大等于 0 且小于 10000 的数 "<<endl;
     for(int i=0;i<n;i++)
     {
       cin>>array[i];
       c[array[i]]++;
     }
     for(int i=0; i<10000; i++)
     {
       for(int j=1; j<=c[i]; j++)
         {
            array[k] = i;
            cout<<array[k]<<"";
            k++;
         }
       if(k==n) break;
     }
     return 0;
}
```

输入：

```
10
99 23 56 48 23 56 58 99 14 95
```

输出：

```
//test(5)-2
#include<iostream>
```

```
using namespace std;
const int SIZE=100;
int main()
{
     int n,f,i,left,right,middle,a[SIZE];
     cin>>n>>f;
     for(i=1;i<=n;i++)
     cin>>a[i];
     left=1;
     right=n;
     do
     {
         middle=(left+right)/2;
         if(f<=a[middle])
         right=middle;
         else
         left=middle+1;
     }while(left<right);
     cout<<left<<endl;
     return 0;
}
```

输入：

```
12 17
2 4 6 9 11 15 17 18 19 20 21 25
```

输出：

（6）世博会志愿者的选拔工作正在A市如火如荼地进行。为了选拔最合适的人才，A市对所有报名的选手进行了笔试，笔试分数达到面试分数线的选手方可进入面试。面试分数线根据计划录取人数的150%划定，即如果计划录取m名志愿者，则面试分数线为排名第m*150%（向下取整）名的选手的分数，而最终进入面试的选手为笔试成绩不低于面试分数线的所有选手。

现在就请你编写程序划定面试分数线，并输出所有进入面试的选手的报名号和笔试成绩。

输入格式：第1行，两个整数n，m（5≤n≤5000，3≤m≤n），中间用一个空格隔开，其中n表示报名参加笔试的选手总数，m 表示计划录取的志愿者人数，输入数据保证m*150%向下取整后小于等于n；第2行

到第n+1行，每行包括两个整数，中间用一个空格隔开，分别是选手的报名号k（1000≤k≤9999）和该选手的笔试成绩s（1≤s≤100），数据保证选手的报名号各不相同。

输出格式：第1行，有两个整数，用一个空格隔开，第一个整数表示面试分数线，第二个整数为进入面试的选手的实际人数；从第2行开始，每行包含两个整数，中间用一个空格隔开，分别表示进入面试的选手的报名号和笔试成绩，按照笔试成绩从高到低输出，如果成绩相同，则按报名号由小到大的顺序输出。

输入样例：

```
6 3
1000 90
3239 88
2390 95
7231 84
1005 95
1001 88
```

输出样例：

```
88 5
1005 95
2390 95
1000 90
1001 88
3239 88
```

说明：m*150% = 3*150% = 4.5，向下取整后为4。保证4个人进入面试的分数线为88，但因为88有重分，所以所有成绩大于等于88的选手都可以进入面试，故最终有5个人进入面试。

（7）某小学最近得到了一笔赞助，打算拿出其中一部分为学习成绩优秀的前5名学生发奖学金。期末，每个学生都有3门课的成绩：语文、数学、英语。先按总分从高到低排序，如果两个同学总分相同，再按语文成绩从高到低排序，如果两个同学总分和语文成绩都相同，那么规定学号小的同学排在前面，这样，每个学生的排序是唯一确定的。

任务：先根据输入的3门课的成绩计算总分，然后按上述规则排序，最后按排名顺序输出前五名学生的学号和总分。注意，在前5名同学中，

每个人的奖学金都不相同，因此，你必须严格按上述规则排序。例如，在某个正确答案中，如果前两行的输出数据(每行输出两个数：学号、总分)是：

7 279

5 279

这两行数据的含义是：总分最高的两个同学的学号依次是7号、5号。这两名同学的总分都是279（总分等于输入的语文、数学、英语三科成绩之和），但学号为7的学生语文成绩更高一些。如果你的前两名的输出数据是：

5 279

7 279

则按输出错误处理，不能得分。

输入格式：包含n+1行，第1行为一个正整数n，表示该校参加评选的学生人数；第2到n+1行，每行有3个用空格隔开的数字，每个数字都在0到100之间。第j行的3个数字依次表示学号为j-1的学生的语文、数学、英语的成绩，每个学生的学号按照输入顺序编号为1~n（恰好是输入数据的行号减1），所给的数据都是正确的，不必检验。

50%的数据满足：各学生的总成绩各不相同；

100%的数据满足：6<=n<=300。

输出格式：共有5行，每行是两个用空格隔开的正整数，依次表示前5名学生的学号和总分。

输入样例：

6

90 67 80

87 66 91

78 89 91

88 99 77

67 89 64

78 89 98

输出样例：

6 265

4 264

3 258

2 244

1 237

（8）[1] 明明想在学校中请一些同学一起做一项问卷调查，为了实验的客观性，他先用计算机生成了N个1到1000之间的随机整数（N≤100），对于其中重复的数字，只保留一个，把其余相同的数去掉，不同的数对应着不同的学生的学号。然后再把这些数从小到大排序，按照排好的顺序去找同学做调查。请你协助明明完成“去重”与“排序”的工作。

输入格式：有2行，第1行为1个正整数，表示所生成的随机数的个数N；第2行有N个用空格隔开的正整数，为所产生的随机数。

输出格式：也是2行，第1行为1个正整数M，表示不相同的随机数的个数，第2行为M个用空格隔开的正整数，为从小到大排好序的不相同的随机数。

输入样例：

10

20 40 32 67 40 20 89 300 400 15

输出样例：

8

15 20 32 40 67 89 300 400

（9）平面上有一个大矩形，其左下角坐标（0，0），右上角坐标（R,R)。大矩形内部包含一些小矩形，小矩形都平行于坐标轴且互不重叠。所有矩形的顶点都是整点。要求画一根平行于y轴的直线x=k（k是整数），使得这些小矩形落在直线左边的面积必须大于等于落在右边的面积，且两边面积之差最小。并且，要使得大矩形在直线左边的面积尽可能大。注意：若直线穿过一个小矩形，将会把它切成两个部分，分属左右两侧。

输入格式：第1行是整数R，表示大矩形的右上角坐标是(R,R)(1<=R<=1 000 000)；第2行是整数N（0＜N＜＝100）,表示一共有N个小矩形；再接下来有N行，每行有4个整数，L,T,W和H,表示有一个小矩形的左上角坐标是(L,T),宽度是W，高度是H(0<=L,T<=R,0＜H、W）

输出格式：输出整数n，表示答案应该是直线x=n。如果必要的话，x=R也可以是答案。

输入样例：

1000

2

1 1 2 1

1）本题选自NOIP2006普及组复赛试题。

5 1 2 1

输出样例：

5

（10）在一个非降序列中，查找与给定值最接近的元素。

输入格式：第1行包含一个整数n，为非降序列长度，1<=n<=100000；第2行包含n个整数，为非降序列各元素，所有元素的大小均在0~ 1 000 000 000之间；第3行包含一个整数m，为要询问的给定值个数，1<=m<=10000；接下来m行，每行一个整数，为要询问最接近元素的给定值，所有给定值的大小均在0~1 000 000 000之间。

输出格式：m行，每行一个整数，为最接近相应给定值的元素值，保持输入顺序。若有多个值满足条件，输出最小的一个。

输入样例：

3
2 5 8
2
10
5

输出样例：

8
5

5.4　字符数组

【例 5.18】一串字符如果从左读和从右读完全相同，我们称之为**回文**。请判断键盘输入的一串字符（不超过1000位），是否是回文。是，则输出YES，否则输出NO。

分析：根据回文定义，对于一串n位的字符，如果能确定这串字符的第1位=第n位，第2位=第n–1位，……，一直到中间位置，每一对字符都相等，那么这串n位的字符就是回文，如果中间出现不相等的情况，这串n位的字符就不是回文。

这是一个明显的循环操作。我们要做的就是把n位字符都保存下来，并重复判断。

程序如下：

```
1  //exam5.18
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6    char s[1000];
7    int i=0,n=0;
8    while((s[n]=getchar())!='\n')      // 当读入的字符非换行符，
                                           就重复读入
9    n++;
10   n--;
11   while(s[i]==s[n-i])                // 重复判断字符是否相等
12     i++;
13   if(i>n/2)cout<<"YES";
14   else cout<<"NO";
15   return 0;
16   }
```

运行结果：

```
I
YES
IvI
YES
IyouIyouI
NO
```

程序第6行将数组s的元素类型定义为char。这种将元素类型定义为字符型的数组，我们称之为字符数组。那么字符数组有什么特殊之处呢？具体如何应用呢？

带着这些问题，本节，我们就来学习C++中的字符数组。

5.4.1 字符数组的定义

字符数组的定义格式如下：

char数组名[元素个数];

例如上例中的

char s[1000];

【例5.19】写出下面程序的运行结果。

```
1  //exam5.19
2  #include<iostream>
```

```
3  using namespace std;
4  int main()
5  {
6   char a[6];
7   int i;
8   a[0]='a';
9   for(i=1;i<6;i++)
10  {
11     a[i]=a[i-1]+2;
12     cout<<a[i]<<endl;
13  }
14   return 0;
15 }
```

运行结果：

```
c
e
g
i
k
```

说明：字符数组的每一个元素都可以当作字符使用。

在程序的第 6 行，定义了有 6 个元素的字符数组 a。于是，a 数组的 6 个元素就都可以当作普通字符来使用：首先，在程序的第 8 行给元素 a[0] 赋值为字符'a'，然后在循环体里的第 11 行，对 a[1]~a[5] 分别赋值为其前面元素的值 +2。于是有：

```
a[0]='a'
   ↓
a[1]=a[0]+2='a'+2='c'
   ↓
a[2]=a[1]+2='c'+2='e'
   ↓
a[3]=a[2]+2='e'+2='g'
   ↓
a[4]=a[3]+2='g'+2='i'
   ↓
a[5]=a[4]+2='i'+2='k'
```

【例5.20】写出下面程序的运行结果。

```
1  //exam5.20
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6    char a[6]={"12345"};
7    char b[5]={'1','2','3','4','5'};
8    int i;
9    for(i=0;i<=5;i++)
10     cout<<a[i];
11   cout<<"aOK"<<endl;
12   for(i=0;i<=4;i++)
13     cout<<b[i];
14   cout<<"bOK"<<endl;
15   return 0;
16 }
```

运行结果：

```
12345 aOK
12345bOK
```

说明： 对于字符数组的初始化，可以在定义时对每个元素逐一初始化，也可以在定义时直接使用双引号引起来一串字符来实现。

例如：“char a[10]={'1','2','3','4','5'};”与“char a[10]={"12345"};”是两个等价的字符数组初始化定义。

但是，在使用双引号形式初始化时，字符个数必须比所定义的数组元素个数少一，这是因为这种情况下数组的最后一个位置，要被系统用来存放一个特殊的字符“\0”（更详细的原因，将在字符串一章解释）。

根据上述说明，程序中分别定义了a和b两个字符数组，且均在定义时进行了初始化：在第6行，数组a的元素个数定义为6，以双引号形式进行初始化；而在第7行，数组b的元素个数定义为5，并对元素逐一初始化。

对于数组a的初始化会有一个特殊的占位标志。这个占位标志在输出时，也会有所反应。

5.4.2 字符数组的应用

【例5.21】在应用计算机编辑文档的时候，我们经常遇到替换任务。例

如把文档中的“电脑”都替换成“计算机”。现在请你编程模拟一下这个操作。

输入两行内容，第1行是原文（长度不超过200个字符），第2行包含以空格分隔的两个字符A和B，要求将原文中所有的字符A都替换成字符B。

输入样例：

I love China. I love Beijing.

I U

输出样例：

U love China. U love Beijing.

分析：为完成操作，首先要将给定的原文保存在字符数组里。然后，在原文中，从头开始寻找字符A，找到一个字符A，便将其替换成字符B；继续寻找下一个字符A，找到了就替换，……，直到将原文都处理完。

模拟这个过程，可以写出循环结构的程序。

程序如下：

```
1  //exam5.21
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6    char txt[200];
7    char A,B;
8    int i,ntxt=0;
9    while ((txt[ntxt++]=getchar())!='\n');
                              // 将原文存放到字符数组 txt 中
10   A=getchar();getchar();B=getchar();  // 读取字符 A 和字符 B
11   for(i=0;i<ntxt;i++)    // 在 txt 数组中从头到尾寻找字符 A
12    if(txt[i]==A)         // 找到字符 A，则替换成字符 B 输出
13      cout<<B;
14    else                  // 不是字符 A，就原样输出
15      cout<<txt[i];
16   cout<<endl;
17   return 0;
18 }
```

运行结果：

```
I love China.I love Beijing.
I U
U love China.U love Beijing.
```

实验：

（1）在程序第9行，while语句的循环体为空。你能用非空循环体改写这个语句，使其完成等效的功能吗？

（2）在程序第10行读取字符A和B的语句中间，出现一个“getchar();”语句，可以省略吗？

【例5.22】对于输入的一个英文语句（单词以空格分隔），输出其中最长的单词及其长度。

输入样例：

I love China.

输出样例：

China 5

分析：为寻找一个英文语句中的最长单词，我们可以仿照第4章中寻找最高身高的做法。首先预设一个结果单词及其长度，然后检索句子中的每一个单词，遇到更长的，就更新长度和结果单词，一直到句子检索完毕，输出结果单词和长度即可。

为完成句子的检索，我们需要将句子中的每一个字符都保存下来，这就需要使用字符数组。

程序如下：

```
1  //exam 5.22
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6    char ans[200],txt[200];
7    int  len=0,ntxt=0,tmp=0,i,j;
8    while((txt[ntxt++]=getchar())!='\n');
                                    // 存储句子到 txt 数组中
9    for(i=0;i<ntxt;i++)            // 检索整个 txt 数组
10   {
11   if(txt[i]!=''&&txt[i]!='.')    // 统计当前单词的长度 tmp
12     tmp++;
13   else
14   {
15     if (tmp>len)                 // 若当前单词长度大于 len，则
                                       更新 len 和 ans 数组
```

```
16        {
17          len=tmp;
18          for(j=0;j<tmp;j++)
19            ans[j]=txt[i-tmp+j];   // 更新 ans 数组
20        }
21        tmp=0;
22      }
23    }
24    for(j=0;j<=len-1;j++)
25      cout<<ans[j];                // 输出长度最长的单词
26    cout<<""<<len;                 //输出最长单词的长度
27    return 0;
28 }
```

运行结果：

```
I love China.
China 5
```

实验：程序第 19 行“ans[j]=txt[i-tmp+j]”语句中，txt 数组的下标写成 i-tmp+j 的作用是什么？

在第 18 行 for 语句的循环体里，添加输出 j 和 i-tmp+j 的值的语句（例如用下面的程序段替代程序的第 18、19 行）。观察结果，你能回答前面的问题了吗？

```
for(j=0;j<tmp;j++)
{
     cout<<j<<""<<i-tmp+j<<endl;
     ans[j]=txt[i-tmp+j];
}
```

*事实上，解决与一串字符相关的问题，在 C++ 语言中，除了使用字符数组，还有专门的工具——字符串。字符串可以使一串字符的输入、处理、输出都变得更加方便。如果有精力，可以提前学习并进行实验。

【例 5.23】从键盘输入任意两个整数的加法表达式（表达式长度不超过 5 位），计算并输出结果。

输入样例：

12+23

输出样例：

35

分析：单纯的加法计算本身不难，现在的问题是两个加数的各自位数都不确定，如何确定两个加数呢？我们可以从输入的内容中找到“+”，然后以“+”为界，前面的内容组合成第一个加数，后面的内容组合成第二个加数。

在分析出大致思路后，还有两个细节问题需要处理：

（1）如何将数字字符x转换成对应的数字？可以使用表达式：x-'0'。

（2）如何将一位一位的独立数字生一个整数？使用类似多项式展开的形式，例如,321可以表示为((3*10)+2)*10+1。反过来,对于3、2、1这样的独立数字，可以合成为((3*10)+2)*10+1=321。

程序如下：

```
1  //exam5.23
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6    char s[5];
7    int x=0,y=0,n=0;
8    while ((s[n]=getchar())!='+')     // 找到 + 的位置，其前内容
                                          生成第一个加数
9    {
10     x=x*10+s[n]-'0';                // 计算第一个加数
11     n++;
12   }
13   while ((s[n]=getchar())!='\n')    // 从 + 到换行符，其间内容
                                          生成第二个加数
14   {
15     y=y*10+s[n]-'0';                // 计算第二个加数
16     n++;
17   }
18   cout<<x+y;
19   return 0;
20 }
```

运行结果：

```
0+0
0
12+23
35
123+9
132
```

思考：如果表达式的长度更大，例如20位、100位时，如何完成求和

计算呢？

实验： 在程序第 10 行（x=x*10+s[n]-'0';）和第 15 行（y=y*10+s[n]-'0';）后，分别加上输出 x 和 y 值的语句（例如用下面的程序段替代程序中的第 10~17 行）。程序的输出结果如何？

```
while((s[n]=getchar())!='+')
{
   x=x*10+s[n]-'0';
   cout<<x<<"";
   n++;
}
cout<<endl;
while ((s[n]=getchar())!='\n')
{
   y=y*10+s[n]-'0';
   cout<<y<<"";
   n++;
}
cout<<endl;
```

练　习

（1）写出程序运行结果。

```
//test(1)-1
#include <iostream>
using namespace std;
int main()
{
   char c[27]={"AbcDefGhiJklMnoPqrStuVwxyZ"};
   for(int i = 0; i < 26; ++i)
     putchar(c[i]=='A'?'T':(c[i]=='T'?'A':(c[i]=='C'?'G':'C')));
   cout<<endl;
   return 0;
}

//test(1)-2
#include <iostream>
```

```
using namespace std;
int main()
{
   char s[40];
   int i,len=0;
   while((s[len]=getchar())!='\n')
     len++;
   for(i=0;i<len;i++)
     if(('a'<=s[i]) && (s[i]<='z'))
       putchar(s[i]-32);
     else putchar(s[i]);
   cout<<endl<<len<<endl;
   return 0;
}
```

输入：I have a dream.

```
//test(1)-3
#include <iostream>
using namespace std;
int main()
{
   int num[26]={0};
   char st[100];
   int i,len=0;
   while ((st[len]=getchar())!='\n')
     len++;
   for(i = 0; i < len; i++)
     ++num[st[i]-'a'];
   for(i = 0; i < len; i++)
     if(num[st[i] - 'a']== 1)
       putchar(st[i]);
   return 0;
}
```

分别输入：asdasdf　　asdaodf

```
//test(1)-4
#include <iostream>
using namespace std;
int main()
{
```

```
    int num[26]={0};
    char st[100];
    int i,j,len=0;
    while ((st[len]=getchar())!='\n')
      len++;
    i=0;
    while(st[i]!='.')
    {
      putchar(st[i]);i++;
    }
    if(st[i+6]>='5') st[i+5]++;
    for(j=1;j<=6;j++)
      putchar(st[j+i-1]);
    return 0;
}
```

分别输入：9.87654321　　1.234567890

（2）输入一段由若干个以空格分隔的单词组成的英文文章，求出文章中最短的单词(文章以英文句点“.”结束，且字符总数不超过200)。

输入样例：

We are Oiers.

输出样例：

We

（3）计算仅含有加法计算的表达式的值。该表达式长度不超过250，中间没有空格与括号，并且计算结果在整数范围内。

输入样例：

12+23+21

输出样例：

56

（4）考试的时候老师最讨厌有人抄袭了。自从有了电子评卷，老师要查找雷同卷，就容易多了：只要将两个人的答案输入计算机，进行逐个字符的比对，把相同的位置都找出来，就一目了然了。

输入格式：2行，每行包含一串字符（长度不超过200)。

输出格式：1行，包含若干个以空格分隔的数字，表示出现相同字符的位置。

输入样例：

I am a girl.I was born in 2002.

I am a giel.I was birn in 2012.

输出样例：

1 2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 18 19 21 22 23 24 25 26 27 28 30 31

（5）[1] 若一个数（首位不为零）从左向右读与从右向左读都是一样，我们就将其称之为回文数。例如：给定一个十进制数56，将56加65（即把56从右向左读），得到的121是一个回文数。又如，对于十进制数87：

STEP1：87+78=165　　STEP2：165+561=726

STEP3：726+627=1353　　STEP4：1353+3531=4884

在这里的一步是指进行了一次N进制的加法，上例最少用了4步得到回文数4884。

写一个程序，给定一个N（2≤N≤10，N=16）进制数m，m的位数上限为20。求最少经过几步可以得到回文数。如果在30步以内（包括30步）不可能得到回文数，则输出“impossible”

输入格式：1行，包含以空格分隔的两个数，分别表示m和n。

输出格式：1行，仅一个数，表示步数。若不能得到回文数，输出impossible。

输入样例：

9 87

输出样例：

6

（6）[2] 每一本正式出版的图书都有一个ISBN与之对应，ISBN包括9位数字、1位校验码和3位分隔符，其规定格式如“x-xxx-xxxxx-x”，其中符号“-”是分隔符（键盘上的减号），最后一位是校验码，例如0-670-82162-4就是一个标准的ISBN。ISBN的首位数字表示书籍的出版语言，例如0代表英语；第一个分隔符“-”之后的三位数字代表出版社，例如670代表维京出版社；第二个分隔符之后的五位数字代表该书在出版社的编号；最后一位为校验码。

校验码的计算方法如下：首位数字乘以1加上次位数字乘以2……以此类推，用所得的结果mod 11，所得的余数即为校验码，如果余数为

1）本题选自NOIP1999普及组复赛试题。

2）本题选自NOIP2008普及组复赛试题。

10，则校验码为大写字母X。例如ISBN 0-670-82162–4中的校验码4是这样得到的：对067082162这9个数字，从左至右，分别乘以1，2，……，9，再求和，即0×1+6×2+……+2×9=158，然后取158 mod 11的结果4作为校验码。

你的任务是编写程序判断输入的ISBN中校验码是否正确，如果正确，则仅输出“Right”；如果错误，则输出你认为是正确的ISBN。

输入格式：只有1行，是一个字符序列，表示一本书的ISBN（保证输入符合ISBN的格式要求）。

输出格式：共1行，假如输入的ISBN的校验码正确，那么输出“Right”，否则，按照规定的格式，输出正确的ISBN（包括分隔符“–”）。

输入样例1：

```
0–670–82162–4
```

输出样例1：

```
Right
```

输入样例2：

```
0–670–82162–0
```

输出样例2：

```
0–670–82162–4
```

（7）使用Wifi上网时，通常需要输入正确的密码之后，才能登录。假设输入密码没有次数限制（密码通常为八个字符，假设预置密码为NOIP@CCF），请你编写一个程序，模拟使用Wifi上网的登录过程：用户尝试输入密码，直到自己要求结束或者密码正确。

输入：包含若干行尝试登录信息，每一次尝试对应两行或一行输入，第1行，一个字符“Y”或“N”，表示是否继续登录；第1行为“Y”时，则还需要输入第2行——八位字符，表示要尝试的密码。

输出：仅1行，“Success”或“Sorry”，表示密码是否正确的提示信息

输入样例1：

```
Y
cctv@CCF
Y
NOIP@CC
```

输出样例1：

```
Sorry
```

Success

输入样例2：

Y

cctv@CCF

N

输出样例2：

Sorry

5.5 二维数组

【例5.24】某班级有n人（n<80），期末考试的六门学科分别是语文、数学、英语、物理、化学、生物。考试成绩出来了，现要求每人的成绩总分和各学科的平均分。输入班级人数，每人的座号和各学科成绩，输出每人的座号、成绩和各学科平均分（四舍五入保留1位小数）。

输入样例：

3

1 67 89 93 82 87 90

2 80 98 87 82 89 93

3 78 86 92 90 67 85

输出样例：

1 67 89 93 82 87 90 508

2 80 98 87 82 89 93 529

3 78 86 92 90 67 85 498

75.0 91.0 90.7 84.7 81.0 89.3

分析：问题的关键是如何存储每人的座号和学科成绩，可以用7个一维数组来存储，但这样在处理的过程中比较麻烦。我们借鉴平面坐标的思想，用一维表示人，用另一维表示座号和各学科，使用二维下标的数组存储每人的座号和学科成绩。

设二维数组a[80][8]存储每人的座号、学科成绩和总分，一维数组av[7]存储各学科平均分。如何将数据存入二维数组中呢？一维数组通常用一重循环控制数组下标，对于二维数组就需要用两重循环控制数组下标，一重控制行，这里表示第几人的成绩，一重控制列，这里表示某人的学科成绩。

归纳上述分析，具体实现步骤如下：

（1）读入人数n。

（2）用二重循环读入每人的座号、学科成绩，并求每人的总分，各学科的总分。

（3）求各学科的平均分。

（4）输出结果。

程序如下：

```
//exam5.24
#include<iostream>
#include<cstring>
#include<cmath>
const int MAXN=85;
using namespace std;
int main()
{
  int a[MAXN][8];
  float av[7];
  memset(av,0,sizeof(av));
  memset(a,0,sizeof(a));
  int temp,k,n;
  cin>>n;
  for(int i=0;i<n;i++)          //循环每行（每人）
  {
      cin>>a[i][0];             //读入座号
      for(int j=1;j<=6;j++)     //循环每列
        {
          cin>>a[i][j];         //读入每人各科成绩
          a[i][7]+=a[i][j];     //累加每人成绩求总分
          av[j]+=a[i][j];       //累加每列学科成绩求学科总分
        }
  }
  for(int i=1;i<=6;i++)         //求各学科平均分
  av[i]/=n;
  for(int i=0;i<n;i++)          //按行输出每人的成绩
    {
      for(int j=0;j<=7;j++)
      cout<<a[i][j]<<"";
      cout<<endl;
```

```
32      }
33    for(int i=1;i<=6;i++)
34      cout<<round(av[i]*10)/10<<"";  //输出各科平均分，保留一位小数
35    return 0;
36 }
```

程序中自然而然地用二维数组便捷地解决了问题。那么，二维数组在使用过程中与一维数组有什么异同点？如何应用二维数组解决问题？为了回答这些问题，我们将学习C++语言的二维数组。

5.5.1 二维数组

1. 定义

与一维数组定义方法类似，二维数组定义的一般格式：

类型名 数组名[常量表达式1][常量表达式2];

通常二维数组中的第一维表示行下标，第二维表示列下标。

行下标和列下标都是从0开始的。例如：

```
int num[4][6];
```

相当于定义了一个二维表格，每个表格对应于一个整型变量，下标变化如下表所示。

[0，0]	[0，1]	[0，2]	[0，3]	[0，4]	[0，5]
[1，0]	[1，1]	[1，2]	[1，3]	[1，4]	[1，5]
[2，0]	[2，1]	[2，2]	[2，3]	[2，4]	[2，5]
[3，0]	[3，1]	[3，2]	[3，3]	[3，4]	[3，5]

二维数组num由4行6列组成，共有24个元素，其中每一行都有6个元素，如第1行的6个元素是num[0][0]、num[0][1]、num[0][2]、num[0][3]、num[0][4]、num[0][5]。

2. 引用

二维数组的使用与一维数组类似，引用的格式为：

数组名[下标1][下标2]

使用数组时特别注意下标不能越界。

使用二维数组时，需要区分是处理行数据、列数据,还是处理所有数据的行列下标。

对某一行进行处理，如：在例5.24程序中第18、21行实现对第i行的

第 1~6 列数据求和存入第 i 行的第 7 列位置中。

对某一列进行处理，如：在例 5.24 程序中第 15、22 行实现将每行的数据按列累加到每列和单元中。

遍历二维数组要用二重循环，如：在例 5.24 程序中第 27、29、30 行实现输出二维数组的所有元素。

3. 初始化

二维数组的初始化与一维数组类似。例如对于数组 num[4][3]，可用如下方式初始化：

```
int num[4][3]={{45,67,78},{43,57,68},{15,65,72},{15,47,28}}
```

实验：设计回答下列问题程序。

（1）当给二维数组的部分元素赋初值后，其他元素的初值会如何变化呢？

```
int num[4][3]={{45,67,78}}
```

（2）当给二维数组赋初值时，每行个数不等时，如何分配初值呢？

```
int num[4][3]={{45,67,78},{43,57},{15},{15,47,28}}
```

5.5.2　二维数组的应用

【例 5.25】杨辉三角是一个由数字排列成的三角形数表，一般形式如下：

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
```

输入数据：

一个正整数 n，表示三角形的行数

输出数据：

n 行杨辉三角形

分析：观察杨辉三角形不难看出，数字是有规律的，从第 3 行开始，每行第 1 个和最后一个值为 1，其他值为其上方和左上方数字和。

设二维数组 c[i][j] 存储行坐标为 i−1、列坐标为 j−1 位置上元素值，则

c[i][j]=c[i-1][j-1]+c[i-1][j]

每个元素值由其左上方和上方元素求和得到，因此，可以一行一行地求得元素值。

归纳上述分析，具体实现步骤如下：

（1）定义二维数组C存放杨辉三角形的值。

（2）赋初值：

c[0][0]=1;c[1][0]=1;c[1][1]=1;

（3）用二重循环控制二维数组下标的变化，从上到下，从左到右求元素值。

（4）输出杨辉三角形。

程序如下：

```
1  //exam5.25
2  #include<iostream>
3  #include<cstring>
4  #define maxn 110
5  using namespace std;
6  int main()
7  {
8      int i,j,n,c[maxn][maxn];
9      cin>>n;                                  // 读入 n
10     memset(c,0,sizeof(c));                   // 数组初始化为 0
11     c[0][0]=1;c[1][0]=1;c[1][1]=1;           // 赋初值
12     for(i=2;i<n;i++)
13       {
14         c[i][0]=1;
15         for(j=1;j<=i;j++)
16         c[i][j]=c[i-1][j-1]+c[i-1][j];
                              // 每个元素值由其左上方和上方元素求和
17       }
18     for(i=0;i<n;i++)    // 输出
19       {
20           for(j=0;j<=i;j++)
21           cout<<c[i][j]<<"";
22           cout<<endl;
23       }
24     return 0;
25 }
```

说明：程序第 12、15、16 行利用二重循环实现二维数组的递推。其思考问题方式与例 5.2 中利用一维数组实现递推求数列各项类似。

【例 5.26】在 n*n 的方阵中填入 1,2,3,……,n*n（n<25），要求填成如下形式的蛇形方阵。例如：n=4 时的方阵为：

```
10  11  12  1
9   16  13  2
8   15  14  3
7   6   5   4
```

分析：直接输出蛇形方阵用正常的输出语句是做不到的。如果将数字按蛇形方阵形式存入二维数组中，输出就很容易了。

设二维数组 num[maxn][maxn] 存放方阵，则蛇形方阵数字存放在数组下标 (0,0)~(n-1,n-1) 单元中，设变量 x 表示行，变量 y 表示列，那么从 x=0、y=n-1 开始写入 1 数字，然后写入顺序是下、左、上、右。

归纳上述分析，具体实现步骤如下：

（1）初始化 num 数组值为 0。

（2）设置写入数字 1 的数组下标变量值，x=0、y=n-1，并在该单元中写入 1。

（3）循环写入 2~n*n：

①向下写，行坐标加 1，当加 1 后的行坐标小于 n 且对应的数组单元未填入数字，在该单元中填入当前数字，继续向下写；

②向左写，列坐标减 1，当减 1 后的列坐标大等于 0 且对应的数组单元未填入数字，在该单元中填入当前数字，继续向左写；

③向上写，行坐标减 1，当减 1 后的行坐标大等于 0 且对应的数组单元未填入数字，在该单元中填入当前数字，继续向上写；

④向右写，列坐标加 1，当加 1 后的列坐标小于 n 且对应的数组单元未填入数字，在该单元中填入当前数字，继续向右写。

（4）输出二维数组 num 的内容，每个数字输出宽度为 4。

程序如下：

```
1  //exam5.26
2  #include<cstdio>
3  #include<cstring>
4  using namespace std;
5  const int maxn=26;
```

```
6  int main()
7  {
8      int x,y,n, count=1;
9      int num[maxn][maxn];
10     scanf("%d",&n);                    //输入 n
11     memset(num, 0, sizeof(num));       //数组初始化为 0
12     x=0; y=n-1;                        //第 1 个数的坐标
13     num[x][y] = 1;                     //写入第 1 个数
14     for(;count<n*n;)                   //写入 2--n*n
15     {
16        while(x+1<n&&!num[x+1][y])num[++x][y]=++count;
                                          //向下写
17        while(y-1>=0&&!num[x][y-1])num[x][--y]=++count;
                                          //向左写
18        while(x-1>=0&&!num[x-1][y])num[--x][y]=++count;
                                          //向上写
19        while(y+1<n&&!num[x][y+1])num[x][++y]=++count;
                                          //向右写
20     }
21     for(x=0;x<n;x++)                   //输出蛇形方阵
22     {
23         for(y=0;y<n;y++)
24           printf("%4d",num[x][y]);
25         printf("\n");
26     }
27     return 0;
28 }
```

说明： 程序第 16、17、18、19 行实现按问题给的规律给二维数组赋值而不是常规地从上到下从左至右赋值。提供了一种控制数组下标变化的方法。

【例 5.27】给定一个具有 N 层的数字三角形如图 5.2 所示，从顶至底有多条路径，每一步可沿左斜线向下或沿右斜线向下，路径所经过的数字之和为路径得分，请求出最小路径得分。

2
6 2
1 8 4
1 5 6 8

图 5.2 数字三角形

输入数据：第 1 行，一个正整数 n，表示三角形的行数（n ≤ 1500）；第 2 至 n+1 行，照描述输入三角形，所有数字均为小于 2000000 的整数。

输出数据：最小路径得分，行末有换行。

输入样例：

```
4
2
6 2
1 8 4
1 5 6 8
```

输出样例：

```
10
```

分析：对于样例，我们先尝试人工走一下，按题意从顶至底每一步可沿左斜线向下或沿右斜线向下，如图 5.3（a）所示，从第 1 行到第 2 行经过的数字路径得分分别为 8 和 4，从第 2 行到第 3 行，需要做选择的是到达数字 8 有两条路径，如果从数字 6 下来其路径得分为 14，如果是从数字 2 下来其路径得分为 12，显然选择得分为 12 的路径，同理，可以得到图 5.3（b）中路径数字旁标注的到达每个数字的最小得分，问题的结果是最后一行数字得分中的最小值。

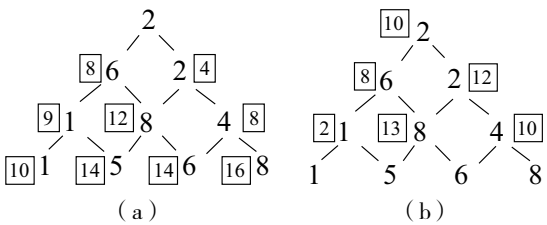


图 5.3　数字三角形

图 5.3（a）的走法是从上向下走，对于路径而言从上往下走与从下往上走是一样的，图 5.3（b）是从下往上走，每次走的时候选择得分小的路径，这样，当走到第 1 层数字时即为最小得分，图 5.3（b）比图 5.3（a）的好处是第一层只有一个元素即为得分最小的路径，无需再去找最小值。

从思考问题角度，我们发现本问题与例 5.25 类似，例 5.25 数组单元的值是从另两个单元求和得到，本问题是从另两个单元中选择最小的得分得到。

设二维数组 minnum[maxn][maxn] 存放经过每个数字时的最小路径得分，则：

```
minnum[i][j]=minnum[i][j]
                +min{minnum[i+1][j], minnum[i+1][j+1]}
```

归纳上述分析，具体实现步骤如下：

（1）读入n。

（2）双重循环，读入三角形数字存入minnum数组中。

（3）循环i:n-2~0。

（4）嵌套循环j:0~i，求

minnum[i][j]=minnum[i][j]
+min{minnum[i+1][j], minnum[i+1][j+1]}

（5）输出minnum[0][0]。

程序如下：

```
1  //exam5.27
2  #include<iostream>
3  #include<cstring>
4  const int maxn=1510;
5  using namespace std;
6  long long minnum[maxn][maxn];      //最大可能值1500*2000000=
                                        3000000000超过了int的范
                                        围，因此要开longlong
7  int main()
8  {
9      int n;
10     cin>>n;// 读入n
11     for(int i=0;i<n;i++)
12     for(int j=0;j<=i;j++)
13         cin>>minnum[i][j];        // 双重循环读入数字三角形
14     for(int i=n-2;i>=0;i--)       // 从下往上走，循环i:n-2~0
15     for(int j=0;j<=i;j++)         // 循环i:n-2~0
16     if(minnum[i+1][j]< minnum[i+1][j+1])
17     minnum[i][j]= minnum[i][j]+minnum[i+1][j];
18     else
19         minnum[i][j]= minnum[i][j]+minnum[i+1][j+1];
                                  // 更新第i+1行数字
20     cout<<minnum[0][0];
21     return 0;
22 }
```

***实验**：将long long minnum[maxn][maxn]开在main函数内，并加一句“memset(minnum,0,sizeof(minnum));”在windows下运行，会出现什么样的情况？为什么？

思考：本节学习了从一维数组拓展到二维数组的应用，那么，对于更多维的数组的使用，你是怎么想的呢？

练　习

（1）阅读下面程序，写出程序运行结果。

```
#include<iostream>
#define maxn 100
using namespace std;
int main()
{
      int n,m,k;
      int a[maxn][maxn];
      cin>>n>>m;
      for(int i=0;i<n;i++)
      {
        k=0;
        if(i%2==0)
        for(int j=m*i+1;j<=m*(i+1);j++)
              a[i][k++]=j;
        else
        for(int j=m*(i+1);j>=m*i+1;j--)
              a[i][k++]=j;
      }
      for(int i=0;i<n;i++)
        {
          for(int j=0;j<m;j++)
          cout<<a[i][j]<<"";
          cout<<endl;
        }
      return 0;
}
```

输入：8　9

输出：

_____

（2）输入4*4方阵，分别求两条对角线上元素之和。

（3）打印一个n*n的数字方阵，例如n=4时：

1　3　4　10

2 5 9 11
6 8 12 15
7 13 14 16

（4）下面是一个3阶的奇数幻方：

6 1 8
7 5 3
2 9 4

它是由1到3^2的自然数组成一个3*3的方阵，方阵的每一行，每一列和两个对角线上各数字之和都相等，且等于$n(n^2+1)/2$（n是方阵的行数或列数）。编程打印出n为10以内的奇数阶幻方。

（5）给定一个5*5的矩阵，每行只有一个最大值，每列只有一个最小值，寻找这个矩阵的鞍点。鞍点指的是矩阵中的一个元素，它是所在行的最大值，并且是所在列的最小值。

例如：在下面的例子中（第4行第1列的元素就是鞍点，值为8 ）。

11 3 5 6 9
12 4 7 8 10
10 5 6 9 11
8 6 4 7 2
15 10 11 20 25

输入格式：输入包含一个5行5列的矩阵。

输出格式：如果存在鞍点，输出鞍点所在的行、列及其值，如果不存在，输出“not found”。

输入样例：

11 3 5 6 9
12 4 7 8 10
10 5 6 9 11
8 6 4 7 2
15 10 11 20 25

输出样例：

4 1 8

（6）给定n*n由0和1组成的矩阵，如果矩阵的每一行和每一列的1的数量都是偶数，则认为符合条件。

你的任务就是检测矩阵是否符合条件，或者在仅改变一个矩阵元素的

情况下能否符合条件。

“改变矩阵元素”的操作定义为0变成1或者1变成0。

输入格式：输入n + 1行，第1行为矩阵的大小n(0<n<100)，以下n行为矩阵的每一行的元素，元素之间以一个空格分开。

输出格式：如果矩阵符合条件，则输出OK；如果矩阵仅改变一个矩阵元素就能符合条件，则输出需要改变的元素所在的行号和列号，以一个空格分开；如果不符合以上两条，输出Corrupt。

输入样例：

```
4
1 0 1 0
0 0 0 0
1 1 1 1
0 1 0 1
```

输出样例：

```
OK
```

（7）输入一个n行m列的黑白图像，将它顺时针旋转90°后输出。

输入格式：第1行包含两个整数n和m，表示图像包含像素点的行数和列数。1<=n<=100，1<=m<=100；接下来n行，每行m个整数，表示图像的每个像素点灰度。相邻两个整数之间用单个空格隔开，每个元素均在0~255之间。

输出格式：m行，每行n个整数，为顺时针旋转90°后的图像。相邻两个整数之间用单个空格隔开。

输入样例：

```
3 3
1 2 3
4 5 6
7 8 9
```

输出样例：

```
7 4 1
8 5 2
9 6 3
```

（8）计算两个矩阵的乘法。n*m阶的矩阵A乘以m*k阶的矩阵B得到的矩阵C是n*k阶的，且C[i][j]=A[i][0]*B[0][j]+A[i][1]*B[1][j]+……

+A[i][m-1]*B[m-1][j](C[i][j]表示C矩阵中第i行第j列元素)。

*5.6　数组的综合应用实例

【例5.28】约瑟夫问题：有n只猴子，按顺序时针方向围成一圈选大王（编号从1~n），从第1号开始报数，一直数到m，数到m的猴子退出圈外，剩下的猴子再接着从1开始报数。就这样，直到圈内只剩下一只猴子时，这只猴子就是猴王。编程完成如下功能：输入n,m后，输出最后猴王的编号。

输入数据：用空格分开的两个整数，第一个是n，第二个是m(0<m,n<10000)。

输出数据：猴王的编号。

输入样例：

6 2

输出样例：

5

分析：有了数组知识，我们可以使用数组模拟猴子选大王过程，设一维布尔数组a[1]~a[n]表示猴子的状态，当a[i]值为0时，表示第i只猴子还在圈中；当a[i]值为1时，表示第i只猴子不在圈中。对于还在圈中的猴子计数到m时，a[i]值设为1，猴子出圈，继续对还在圈中的猴子从1计数到m，直到剩下最后一个猴子即为猴王。

设变量i表示数组下标，变量k用于计数。

归纳上述分析，具体实现步骤如下：

（1）读入n,m。

（2）初始化数组a为0，表示猴子都在圈中，变量l、k初值为0。

（3）循环i：1~n-1，让n-1只猴子出圈：

①设变量l用于逐个枚举圈中的所有位置，如果l>n，l=1回到第1个位置，用数组模拟环状，最后一个与第一个相连；

②如果第l个位置上有人，则k计数；

③如果k计数到m，猴子出圈，k计数清0。

（4）输出最后一只猴子在圈中的位置。

程序如下：

```
1  //exam5.28
```

```
#include<iostream>
#include<cstring>
#define MAXN 100010
using namespace std;
int main()
{
    int n,m,i,l,k;
    int a[MAXN];
    cin>>n>>m;                  //输入 n 只猴子，报数 m
    memset(a,0,sizeof(a));      //初始化数组为 0，表示猴子都在圈中
    k=0;l=0;
    for(i=1;i<=n-1;i++)         //让 n-1 只猴子出圈
    {
      while(k<m)
      {
        l++;                    //枚举圈中的位置
        if(l>n) l=1;            //下标超过 n，回到 1
        if(!a[l]) k++;          //位置有猴子，计数加 1
      }
        a[l]=!a[l];k=0;         //猴子出圈，计数清零
  }
   for(i=1;i<=n;i++)            //输出猴王
   if(!a[i]) cout<<i<<endl;
  return 0;
}
```

思考：用数组模拟解决约瑟夫问题比较直观，但效率不高，有没有更有效率的解决方法呢？

【例 5.29】求 1 到 N 间的素数。

输入格式：一个数 N（1<N<1000000）。

输出格式：以空格隔开的素数。

输入样例：

10

输出样例：

2 3 5 7

分析：在第 4 章应用循环结构解决过这个问题。现在换一个思路。

2 是素数，2 的倍数一定不是素数，这些数就无需进行素数判定，同

理，3是素数，3的倍数一定不是素数，……于是，先把N个自然数按次序排列起来，1不是素数，也不是合数，要划去。第二个数2是素数，留下来，而把2后面所有能被2整除的数都划去。2后面第一个没划去的数是3，把3留下，再把3后面所有能被3整除的数都划去。3后面第一个没划去的数是5，把5留下，再把5后面所有能被5整除的数都划去。这样一直做下去，就会把不超过N的全部合数都筛掉，留下的就是不超过N的全部素数。

这种按顺序输出素数的同时，将素数的倍数筛去的算法，称为**筛法**。用筛法求素数是一种比较快的算法。

使用筛法求素数的算法描述如下：

设布尔型数组a，数组元素a[i]表示数i是否为素数，数组prime按顺序存放筛出来的素数。

方法1：

（1）读入数据范围n。

（2）初始化数组a值都为0，表示素数。

（3）循环i：2~sqrt(n)，实现下列操作：如果当前i的数组元素为素数，从数组中筛去该元素的所有倍数的元素，即a[i*j]=1。

（4）取出未被筛去的a数组元素存放入prime数组中。

（5）输出prime数组元素。

程序如下：

```
1  //exam5.29-1
2  #include<iostream>
3  #include<cmath>
4  #define MAXN 1000010
5  bool a[MAXN];           // 全局变量，初值为0，表示素数。
6  int prime[MAXN];
7  using namespace std;
8  int main()
9  {
10     int n,m;
11     cin>>n;
12     for(int i=2;i<=sqrt(n)+1;i++)
13     if(a[i]==0)
14     for(int j=2;i*j<=n;j++)
15         a[i*j]=1;          // 如果当前i的数组元素为素数，从数组
                                 中筛去该元素的所有倍数的元素，即
                                 a[i*j]:=1
```

```
    m=0;
    for(int i=2;i<=n;i++)
    if(a[i]==0)
        prime[m++]=i;          //取出未被筛去的 a 数组元素存放入
                                  prime 数组中
    for(int i=0;i<m;i++)
        cout<<prime[i]<<"";    //输出素数
    return 0;
}
```

***方法 2：**在上述程序实现中，每次筛去当前获取的素数的倍数，可能存在某些数被重复筛的现象。如：n=25 时，数字 6、12、24 被 2 筛去也被 3 重复筛，如果不重复筛，还能提高效率。

应用数学知识，我们知道任何一个合数都能表示成一系列素数的积。在顺序扫描 i 是否为素数过程中：

（1）如果 i 是素数，那么，素数 i 乘以小于 i 的素数，筛出的数跟之前筛出的是不会重复的。

（2）如果 i 是合数，i 可以表示成递增素数相乘 i=p1*p2*...*pn，pi 都是素数（2<=i<=n），p1 是最小的系数，筛出不大于 p1 的素数 *i，即筛完 p1 等于某个小于 i 素数时，终止筛除。

如：i=2*3*5。此时能筛除 2*i，不能筛除 3*i，如果能筛除 3*i 的话，当 i’=3*3*5 时，筛除就和前面重复了。

为此，我们得到线性筛法具体实现步骤如下：

（1）读入数据范围 n。

（2）初始化数组 a 值都为 0，表示素数。

（3）循环 i：2~n，实现下列操作：

①如果当前 i 的数组元素为素数，存入 prime 数组中。

②循环 j：0~ 当前素数个数 -1，筛去 i 乘 prime 数组的元素，即 a[i*prime[j]]=1（i*prime[j]<=n），筛的过程中如果 i 被 prime[j] 整除，则退出筛数。

（4）输出 prime 数组元素。

程序如下：

```
1  //exam5.29-2
2  #include<iostream>
3  #include<cstring>
```

```
4  using namespace std;
5  #define MAXN 1000010
6  int prime[MAXN],a[MAXN];
7  int main()
8  {
9    int n,i,j;
10    cin>>n;
11    int num=0;                    // 记录素数的个数
12    memset(a,0,sizeof(a));        // 初始化数组为 0，表示全是素数
13    for(i=2;i<n;i++)
14    {
15        if(!a[i])        // 如果 a[i] 为素数标志，存入 prime 数组
16           prime[num++]=i;
17        for(j=0;j<num&&i*prime[j]<n;j++)
                           // 筛去 i*prime 数组的元素
18        {
19           a[prime[j]*i]=1;
20           if(!i%prime[j]) break;    // 如果 i 被 prime 数组元
                                          素整除，则退出
21        }
22     }
23     for( i=0;i<num;i++)              // 输出素数
24     cout<<prime[i]<<"";
25     return 0;
26 }
```

说明：筛法提供了一种利用数组下标标志筛去有规律数而得到问题的解的一种思维方法。

【例5.30】给出一个正整数a，求出a(1 <= a <= 10^12)的因子中最大的数x，使x没有平方因子。

分析：根据数学知识，任何一个大于1的自然数N，都可以唯一分解成有限个质数的乘积N=(P1^a1)*(P2^a2)……(Pn^an)，这里P1<P2<…<Pn是质数，ai是正整数。

那么，可以推出问题中要求的x是a质因子的乘积，设ans表示乘积，初值为1。

归纳分析，具体实现步骤如下：

（1）筛出1到sqrt(a)的质数。

（2）对于这区间的每个质数 p，如果 a 能被 p 整除，则 p 是 a 质因子，那么 ans 乘以 p 并且将 a 一直除以 p 直到不能整除 p 为止。

（3）输出答案 ans。

程序如下：

```
1  //exam5.30
2  # include <cstdio>
3  bool notprime[10000010];       // 是否是素数的标志数组
4  long long prime[1000010], primecnt;
5  int main ()
6  {
7      register long long a, ans = 1, m;
8      scanf("%I64d", &a);
9      m = a;
10     for(register long long p = 2; p * p <= a; ++p)
11         if(!notprime[p])      // 判断 p 是否是素数
12         {
13             prime[++primecnt]=p;
                              // 如果 p 是，加入到 prime 数组中去
14             if(!(a%p))     // 判断 a 是否能被质因子 p 整除
15             {
16                 ans*=p;   // 如果满足，ans 乘以 p，统计到答案
17                 while(!(m%p))
                       // 质因子 p 只统计一次，除到不能整除 p 为止
18                     m/=p;
19             }
20         for(register long long j =p<<1;j*j<=a;j+=p)
                       // 如果当前 p 的数组元素为素数，从数组中筛去
                          该元素的所有倍数的元素
21                 notprime[j] = 1;
22         }
23     if(m>1)
24         ans *= m;
25     printf("%I64d\n", ans);
26   return 0;
27 }
```

【例 5.31[1)]】如图 5.4 所示，A 点有一个过河卒，需要走到目标 B 点。卒

1）本题选自 NOIP2002 普及组复赛试题。

行走规则：可以向下或者向右。同时在棋盘上的任一点有一个对方的马（C点），该马所在的点和所有跳跃一步可达的点称为对方马的控制点。例如图5.4中C点上的马可以控制9个点（图中除A，B之外的黑点）。卒不能通过对方马的控制点。

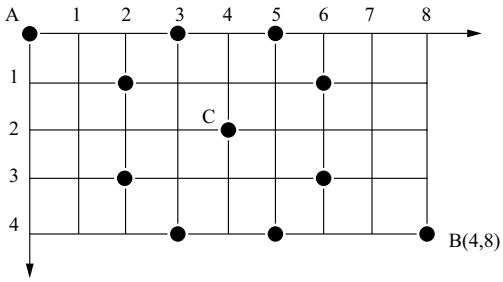


图5.4　示例图

棋盘用坐标表示，A点（0，0）、B点（n,m）(n,m为不超过100的整数)，同样，马的位置坐标是需要给出的（约定：C<>A，同时C<>B）。现在要求你计算出卒从A点能够到达B点的路径的条数。

输入数据：B点的坐标（n,m）以及对方马的坐标（X,Y）。

输出数据：一个整数（路径的条数）。

输入样例：

6 6 3 2

输出样例：

17

分析：由于过河卒只能向下或者向右行走，那么，每点上的路径条数为其上方点和其左方点路径条数之和。设二维数组a[i][j]表示从A点走到第i行第j列位置的路径条数，则a[I][j]具有下面的性质：

（1）行坐标轴和列坐标轴上点的路径条数：A[i][0]=1，A[0][j]=1。

（2）对于没被马控制的坐标点，如图中(1，1)点，只能从(0，1)和(1，0)走到(1，1)点，则：

A[1][1]=A[1][0]+A[0][1]=2

同理：任意一个没被马控制的坐标点(i,j)，只能从(i-1，j)和(i，j-1)走到(1，1)点，则：

A[i][j]=A[i-1][j]+A[i][j-1]

（3）对于被马控制的坐标点，则：

A[i][j]=0

那么，对于已知马点坐标(I,j)，如何标识被马控制的坐标点呢？

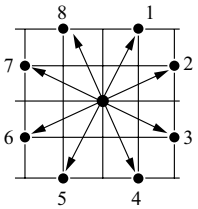


图5.5　马的走步规则

如图 5.5 所示，根据马的走步规则，中心位置马所能控制的点为图中的 1~8 点，设 dx、dy 表示被马控制点相对于马的坐标位移，则：

dx[0~7]=(2,1,-1,-2,-2,-1,1,2)

dy[0~7]=(1,2,2,1,-1,-2,-2,-1)

已知马点坐标(x,y)，求得被马控制点的坐标为：

x=x+dx　　y=y+dy

综合上述分析，具体实现步骤如下：

（1）定义一个二维布尔型 b 数组，标识马点(x,y)及其控制点的坐标，定义一个二维数组 a 存储从 A 点走到各点的路径条数。

（2）读入 B 点的坐标（n,m）及对方马的坐标（x,y）。

（3）初始化 a、b 数组都为 0，将对方马点及控制点设为 1。

（4）a 数组的起点初值设为 1，行坐标轴上从左到右的路径条数初值设为 1，直到遇到被马控制的点为止，列坐标轴上从上到下路径条数初值设为 1，直到遇到被马控制的点为止。

（5）用二重循环 I,j 分别控制从第 1 行到第 n 行，从第 1 列到第 m 列，执行下列操作：

①非马控制点，a[i][j]=a[i-1][j]+a[i][j-1]。

②马控制点，a[i][j]=0。

（6）输出 a[n][m]。

程序设计：

```
1  //exam5.31
2  #include<iostream>
3  #include<cstring>
4  #define MAXN 110
5  using namespace std;
6  int main()
7  {
8      bool b[MAXN][MAXN];
9      long long a[MAXN][MAXN];
10     memset(b,0,sizeof(b));  //b 数组初始化为 0
11     short dx[8]={2,1,-1,-2,-2,-1,1,2};
12     short dy[8]={1,2,2,1,-1,-2,-2,-1};
13     int n,m,x,y;
```

```
14      cin>>n>>m>>x>>y;
15      b[x][y]=1;
16      for(int i=0;i<=7;i++)   // 将对方马点及控制点设为 1
17      if(x+dx[i]>=0&&x+dx[i]<=n&&y+dy[i]>=0&&y+dy[i]<=m)
18      b[x+dx[i]][y+dy[i]]=1;
19      int k=0;
20      while(!b[k][0]&&k<=n)   // 列坐标轴上从上到下路径条数初值
                                   设为 1，直到遇到被马控制的点为止
21      {
22          a[k++][0]=1;
23      }
24      int l=0;
25      while(!b[0][l]&&l<=m)   // 行坐标轴上从左到右的路径条数初值
                                   设为 1，直到遇到被马控制的点为止
26      {
27          a[0][l++]=1;
28      }
29      for(int i=1;i<=n;i++)
30       for(int j=1;j<=m;j++)
31           if(b[i][j]) a[i][j]=0;    // 马控制点：a[i][j]=0
32           else a[i][j]=a[i-1][j]+a[i][j-1];
                      // 非马控制点：a[i][j]=a[i-1][j]+a[i][j-1]
33        cout<<a[n][m];
34      return 0;
35  }
```

说明：程序第 11、12、16、18 行提供了一种平面上从一个点遍历其周边点的方法。

思考：本问题的分析方法与程序表达方式与例 5.19 和例 5.20 有哪些异同点。

【例 5.32】恶魔猎手尤迪安野心勃勃，他背叛了暗夜精灵，率深藏在海底的那加企图叛变：守望者在与尤迪安的交锋中遭遇了围杀，被困在一个荒芜的大岛上。为了杀死守望者，尤迪安开始对这个荒岛施咒，这座岛很快就会沉下去，到那时，岛上的所有人都会遇难：守望者的跑步速度为 17m/s，以这样的速度是无法逃离荒岛的。庆幸的是守望者拥有闪烁法术，可在 1s 内移动 60m，不过每次使用闪烁法术都会消耗魔法值 10 点。守望者的魔法值恢复的速度为 4 点 /s，只有处在原地休息状态时才能恢复。

现在已知守望者的魔法初值M，他所在的初始位置与岛的出口之间的距离S，岛沉没的时间T。你的任务是写一个程序帮助守望者计算如何在最短的时间内逃离荒岛，若不能逃出，则输出守望者在剩下的时间内能走的最远距离。注意：守望者跑步、闪烁或休息活动均以秒(s)为单位。且每次活动的持续时间为整数秒。距离的单位为米(m)。

输入数据：仅1行，包括空格隔开的三个非负整数M，S，T。

输出数据：包含两行，第1行为字符串“Yes”或“No”(区分大小写)，即守望者是否能逃离荒岛；第2行包含一个整数，第一行为“Yes”(区分大小写)时表示守望者逃离荒岛的最短时间，第一行为“No”(区分大小写)时表示守望者能走的最远距离。

输入样例1：

39 200 4

输出样例1：

No

197

输入样例2：

36 255 10

输出样例2：

Yes

6

数据范围：

30%的数据满足：1<=T<=10，1<=S<=100

50%的数据满足：1<=T<=1000，1<=S<=10000

100%的数据满足：1<=T<=300000，0<=M<=1000 1<=S<=10^8

分析：根据题意，守望者要在最短时间走最多的路程，而每秒有三种方法：休息（魔法恢复4），跑步（移动17m），闪烁法术（花费10魔法，移动60m）。可以得到如下信息：

（1）休息和闪烁法术是有关联的（要不然还不如不休息）。

（2）有魔法的情况下，尽量用闪烁法术（因为闪烁法术移动最远）。

（3）在魔法不够的情况下，对休息（等待魔法恢复使用闪烁法术）还是跑步进行选择。

为了理清信息，不妨将跑步和使用闪烁法术分开处理。

设想：

（1）如果“守望者”不会跑步，记第i秒的能到达最大距离为f[i]，则：

$$f[i]=\begin{cases} f[i-1]+60 & \text{当m（魔法）}\geqslant 10\text{时，同时m=m-10} \\ f[i-1] & \text{当m}<10\text{时，同时m=m+4} \end{cases}$$

通过这样一个预处理，解决了闪烁法术的使用。

（2）把跑步的情况加入，则：

f[i] = Max{f[i],f[i-1]+17}（注意：令f[0]=0）

如此，得到了解决问题的递推式。当f[t]（t为限定的时间）<S时，输出“No”及f[t]值，否则，输出“Yes”及最快的离岛时间i。

综合上述分析，具体实现步骤如下：

（1）读入数据M，S，T。

（2）计算只使用闪烁法术时的每秒最大距离。

（3）计算加入跑步选择时的每秒最大距离，如果在某时刻刚好离岛，则输出离岛时间，结束。

（4）如果不能离岛，输出最远距离，结束。

程序如下：

```
1  //exam5.32
2  #include<iostream>
3  #include<cstring>
4  using namespace std;
5  #define MAXN 300010
6  int main()
7  {
8     int m,s,t,i;
9     int f[MAXN];
10    cin>>m>>s>>t;          //读入数据
11    f[0]=0;
12    for(i=1;i<=t;i++)      //计算只用闪烁法术时的每秒最大距离
13    {
14       if(m>9)
15       {
16         f[i]=f[i-1]+60;
17         m=m-10;
18       }
19       else
20       {
21         f[i]=f[i-1];
```

```
22          m=m+4;
23        }
24      }
25      for(i=1;i<=t;i++)     // 计算加入跑步选择时的每秒最大距离
26      {
27          if(f[i]<f[i-1]+17) f[i]=f[i-1]+17;
28          if(f[i]>=s)            // 刚好离岛，输出
29          {
30             cout<<"Yes"<<endl<<i; return 0;
31          }
32      }
33      cout<<"No"<<endl<<f[t];  // 不能离岛，输出
34      return 0;
35 }
```

说明：本题有多种解决问题的方法，然而，在上述分析中很巧妙地运用了分而治之的思想，把原来跑步、魔法、休息交错在一起的问题条件分离开，考虑在只有魔法情况下每秒最远距离，此时，很容易用上问题的贪心条件，能用魔法尽量用上魔法，求只有魔法情况下每秒最远距离的递推式写起来也很简单。接着，考虑跑步的情况，当前秒的跑步距离由上一秒加 17 递推得到，每秒最远距离为跑步距离和魔法距离中的最大值。这是一道很好的题，建议大家用不同方法解决之，然后，从中体会分析问题的方法和技巧。

练　习

（1）小云和朋友们去爬香山，为美丽的景色所陶醉，想合影留念。如果他们站成一排，男生全部在左（从拍照者的角度），并按照从矮到高的顺序从左到右排，女生全部在右，并按照从高到矮的顺序从左到右排，请问他们合影的效果是什么样的（所有人的身高都不同）?

输入格式：第一行是人数 n（2<=n<=40，且至少有 1 个男生和 1 个女生）；后面紧跟 n 行，每行输入一个人的性别（男 male 或女 female）和身高（浮点数，单位米），两个数据之间以空格分隔。

输出格式：n 个浮点数，模拟站好队后，拍照者眼中从左到右每个人的身高。每个浮点数需保留到小数点后 2 位，相邻两个数之间用单个空格隔开。

输入样例：

6
male 1.72
male 1.78
female 1.61
male 1.65
female 1.70
female 1.56

输出样例：

1.65 1.72 1.78 1.70 1.61 1.56

（2）在1949年，印度数学家D. R. Daprekar发现了一类称作Self-Numbers的数。对于每一个正整数n，我们定义d(n)为n加上它每一位数字的和。例如，d(75)=75+7+5=87。给定任意正整数n作为一个起点，都能构造出一个无限递增的序列：n,d(n),d(d(n)),d(d(d(n))),...例如，如果你从33开始，下一个数是33+3+3=39，再下一个为39+3+9=51，再再下一个为51+5+1=57，因此你所产生的序列就像这样：33,39,51,57,69,84,96,111,114,120,123,129,141,...数字n被称作d(n)的发生器。在上面的这个序列中，33是39的发生器，39是51的发生器，51是57的发生器等。有一些数有超过一个发生器，如101的发生器可以是91和100。一个没有发生器的数被称作Self-Number。如前13个Self-Number为1,3,5,7,9,20,31,42,53,64,75,86,97。我们将第i个Self-Number表示为a[i]，所以a[1]=1, a[2]=3, a[3]=5...

输入格式：输入包含整数N、K、s1...sk，其中，1<=N<=10^7，1<=K<=5000，以空格和换行分割。

输出格式：在第1行你需要输出一个数，这个数表示在闭区间[1, N]中Self-Number的数量。第2行必须包含以空格划分的K个数，表示a[s1].. a[sk]，这里保证所有的a[s1].. a[sk]都小于N（例如，如果N=100，sk可以为1~13，但不能为14，因为a[14]=108>100）。

输入样例：

100 10
1 2 3 4 5 6 7 11 12 13

输出样例：

13

1 3 5 7 9 20 31 75 86 97

（3）给出两幅相同大小的黑白图像（用0-1矩阵）表示，求它们的相似度。

说明：若两幅图像在相同位置上的像素点颜色相同，则称它们在该位置具有相同的像素点。两幅图像的相似度定义为相同像素点数占总像素点数的百分比。

输入格式：第1行包含两个整数m和n，表示图像的行数和列数，中间用单个空格隔开，1<=m<=100，1<=n<=100；之后m行，每行n个整数0或1，表示第一幅黑白图像上各像素点的颜色，相邻两个数之间用单个空格隔开；之后m行，每行n个整数0或1，表示第二幅黑白图像上各像素点的颜色。相邻两个数之间用单个空格隔开。

输出格式：一个实数，表示相似度（以百分比的形式给出），精确到小数点后两位。

输入样例：

3 3
1 0 1
0 0 1
1 1 0
1 1 0
0 0 1
0 0 1

输出样例：

44.44

（4）扫雷游戏是一款十分经典的单机小游戏。它的精髓在于，通过已翻开格子所提示的周围格地雷数，来判断未翻开格子里是否是地雷。

现在给出n行m列的雷区中的地雷分布，要求计算出每个非地雷格的周围格地雷数。

注：每个格子周围格有八个：上、下、左、右、左上、右上、左下、右下。

输入格式：第1行包含两个整数n和m，分别表示雷区的行数和列数，1<=n<=100，1<=m<=100；接下来n行，每行m个字符，‘*’表示相应格子中是地雷，‘?’表示相应格子中无地雷，字符之间无任何分隔符。

输出格式：n行，每行m个字符，描述整个雷区。若相应格中是地雷，

则用‘*’表示，否则用相应的周围格地雷数表示，字符之间无任何分隔符。

输入样例：

3 3
*??
???
?*?

输出样例：

*10
221
1*1

（5）一个M行N列的教室座位中，有D对同学总爱凑在一起讲话。现老师要用走廊隔开他们。但只能在行之间加入K条走廊，在列中加入L条走廊，问加在哪里能使效果最佳（一对爱讲话的同学只有左右相邻或上下相邻）。

输入格式：第1行，有5个用空格隔开的整数，分别是M，N，K，L，D（2<=N，M<=1000，0<=K<M，0<=L<N，D<=2000）；接下来D行，每行有4个用空格隔开的整数，第i行的4个整数Xi，Yi，Pi，Qi，表示坐在位置(Xi,Yi)与(Pi,Qi)的两个同学会交头接耳（输入保证他们前后相邻或者左右相邻），输入数据保证最优方案的唯一性。

输出格式：共两行，第1行包含K个整数，a1a2……aK，表示第a1行和a1+1行之间、第a2行和第a2+1行之间、…、第aK行和第aK+1行之间要开辟通道，其中ai< ai+1，每两个整数之间用空格隔开（行尾没有空格）；第2行包含L个整数，b1b2……bk，表示第b1列和b1+1列之间、第b2列和第b2+1列之间、…、第bL列和第bL+1列之间要开辟通道，其中bi< bi+1，每两个整数之间用空格隔开（行尾没有空格）。

输入样例：

4 5 1 2 3
4 2 4 3
2 3 3 3
2 5 2 4

输出样例：

2
2 4

（6）表达式 6×9=42 对于十进制来说是错误的，但是对于十三进制来说是正确的。即 6(13)×9(13)=42(13)，而 42(13)=4×131+2×130=54(10)。你的任务是写一段程序读入 3 个整数 p、q 和 r，然后确定一个进制 B(2 ≤ B ≤ 16) 使得 p×q=r。如果 B 有很多选择，输出最小的一个。例如：p=11,q=11,r=121，则有 11(3) ×11(3)=121(3)，11(10) ×11(10)=121(10)。这种情况下，输出 3。如果没有合适的进制，则输出 0。

输入数据：1 行，3 个 B 进制的正整数 p、q、r（数位 ≤ 7）。

输出数据：使得 p×q=r 成立的最小进制 B，如果没有合适的进制，则输出 0。

输入样例：

6 9 42

输出样例：

13

（7）Bart 的妹妹 Lisa 在一个二维矩阵上创造了新的文明。矩阵上每个位置被三种生命形式之一占据：石头、剪刀、布。每天，上下左右相邻的不同生命形式将会发生战斗。在战斗中，石头永远胜剪刀，剪刀永远胜布，布永远胜石头。每一天结束之后，败者的领地将被胜者占领。

你的工作是计算出 n 天之后矩阵的占据情况。

输入格式：第 1 行包含三个正整数 r，c，n，分别表示矩阵的行数、列数以及天数，每个整数均不超过 100；接下来 r 行，每行 c 个字符，描述矩阵初始时被占据的情况，每个位置上的字符只能是 R，S，P 三者之一，分别代表石头，剪刀，布。相邻字符之间无空格。

输出格式：输出 n 天之后的矩阵占据情况，每个位置上的字符只能是 R，S，P 三者之一，相邻字符之间无空格。

输入样例：

3 3 1
RRR
RSR
RRR

输出样例：

RRR
RRR
RRR

• 参考文献 •

[1] 李文新郭炜余华山. 程序设计导引及在线实践. 北京：清华大学出版社，2012.

[2] 邱桂香，陈颖. 全国青少年信息学竞赛培训教材——C 语言程序设计. 杭州：浙江大学出版社，2010.

[3] 刘汝佳. 算法竞赛入门经典. 北京：清华大学出版社，2012.

[4] http://c.biancheng.net/cpp/biancheng/view/104.html.

[5] http://blog.csdn.net/zhanghaotian2011/article/details/8868577.

[6] http://www.091xue.com/course/view.php?id=5.

索　引

B

编程…… 2
编译…… 2
变量…… 19
break 语句 ……178
布尔型…… 57

C

插入排序…… 208
常量…… 29
程序……1
cin 语句 …… 39
continue 语句 ……179
cout 语句 …… 5

D

等差数列……113
do-while 语句 ……131

E

二分查找……210
二进制数……138
二维数组…… 233

F

翻译…… 2
分解质因数……151
Fibonacci 数列 …… 114
浮点型…… 19
复合语句…… 66
赋值语句…… 21
for 语句 ……105

G

高级语言…… 2
关系表达式…… 63

H

回文…… 220

I

if 语句 …… 60

J

基本数据类型…… 56
集成开发环境（IDE） …… 2
计算机语言……1
解释…… 2

L

流（stream） …… 5
逻辑变量…… 72
逻辑表达式…… 68
逻辑运算…… 69

M

冒泡排序…… 206

名字空间…… 3
模…… 6

P

printf 函数 …… 43

Q

强制类型转换…… 33

S

scanf 函数 …… 46
筛法…… 247
数学表达式…… 26
数组越界…… 190
顺序查找…… 209
算术运算符…… 5
switch 语句…… 83

T

条件表达式…… 73
头文件…… 3

W

位运算…… 156
while 语句 …… 119

X

选择排序…… 205

Y

杨辉三角…… 236
一维数组…… 184

Z

辗转相除法…… 127
整型…… 18
质数…… 142
字符数组…… 220
字符型…… 35
自动类型转换…… 32
最大公约数…… 114

（TP-7438.01）
责任编辑　杨　凯
责任制作　魏　谨
封面设计　杨安安

CCF中学生计算机程序设计教材

全国青少年信息学奥林匹克竞赛

网　址：www.noi.cn　　E-mail：noi@ccf.org.cn

科学出版社互联网入口
销售：（010）64031535
E-mail：boktp@mail.sciencep.com

www.sciencep.com

ISBN 978-7-03-050021-2

定　价：38.00元