目　录

第1章　程序设计和C语言

1.1 程序设计

程序设计(Programming)是给出解决特定问题程序的过程，是软件构造活动中的重要组成部分。程序设计往往以某种程序设计语言为工具，给出这种语言下的程序。程序设计过程应当包括分析、设计、编码、测试、排错等不同阶段。专业的程序设计人员常被称为程序员。

程序设计的步骤

（1）分析问题

对于接受的任务要进行认真的分析，研究所给定的条件，分析最后应达到的目标，找出解决问题的规律，选择解题的方法，完成实际问题。

（2）设计算法

即设计出解题的方法和具体步骤。

（3）编写程序

根据得到的算法，用一种高级语言编写出源程序。 并通过测试。

（4）对源程序进行编辑、编译和连接

（5）运行程序，分析结果

运行可执行程序，得到运行结果。能得到运行结果并不意味着程序正确，要对结果进行分析，看它是否合理。不合理要对程序进行调试，即通过上机发现和排除程序中的故障的过程。

（6）编写程序文档

许多程序是提供给别人使用的，如同正式的产品应当提供产品说明书一样，正式提供给用户使用的程序，必须向用户提供程序说明书。内容应包括：程序名称、程序功能、运行环境、程序的装入和启动、需要输入的数据，以及使用注意事项等。

1.2 C语言的产生与发展

C语言是1972年由美国的Dennis Ritchie设计发明的,并首次在UNIX操作系统的 DEC PDP-11计算机上使用。它由早期的编程语言BCPL (Basic Combind Programming Language)发展演变而来。在1970年,AT&T贝尔实验室的Ken hompson根据BCPL语言设计出较先进的并取名为B的语言,最后导致了C语言的问世。

随着微型计算机的日益普及,出现了许多C语言版本。由于没有统一的标准, 使得这些C语言之间出现了一些不一致的地方。为了改变这种情况,美国国家标准研究所(ANSI)为 C 语言制定了一套ANSI标准,成为现行的C语言标准。

1.3 C语言的特点

C语言发展如此迅速，而且成为最受欢迎的语言之一,主要因为它具有强大的功能。许多著名的系统软件，如PC-DOS,DBASE Ⅳ都是由C语言编写的。用C语言加上一些汇编语言子程序，就更能显示C语言的优势了。归纳起来C语言具有下列特点:

1 C语言是中级语言

C语言把高级语言的基本结构和语句与低级语言的实用性结合起来。C语言可以象汇编语言一样对位、字节和地址进行操作,而这三者是计算机最基本的工作单元。

2 C语言是结构式语言

结构式语言的显著特点是代码及数据的分隔化,即程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰,便于使用、维护以及调试。C 语言是以函数形式提供给用户的,这些函数可方便的调用,并具有多种循环、条件语句控制程序流向,从而使程序完全结构化。

3 C语言功能齐全

C 语言具有各种各样的数据类型,并引入了指针概念,可使程序效率更高。另外 C 语言也具有强大的图形功能,支持多种显示器和驱动器。而且计算功能、逻辑判断功能也比较强大,可以实现决策目的。

4 C语言适用范围大

C语言还有一个突出的优点就是适合于多种操作系统,如DOS、UNIX,也适用于多种机型。

1.4 面向对象的程序设计语言

在C的基础上，一九八三年又由贝尔实验室的Bjarne Strou-strup推出了C++。 C++进一步扩充和完善了C语言，成为一种面向对象的程序设计语言。C++目前流行的最新版本是 Borland C++, Symantec C++和Microsoft VisualC++。

C++提出了一些更为深入的概念，它所支持的这些面向对象的概念容易将问题空间直接地映射到程序空间，为程序员提供了一种与传统结构程序设计不同的思维方式和编程方法。因而也增加了整个语言的复杂性，掌握起来有一定难度。

1.5 C和C++

C是C++的基础，C++语言和C语言在很多方面是兼容的。因此，掌握了C语言，再进一步学习C++就能以一种熟悉的语法来学习面向对象的语言，从而达到事半功倍的目的。

比尔•盖茨

姓名：威廉•亨利•盖茨
昵称：比尔•盖茨
出生：1955 年 10 月 28 日
身高：180cm
体重：78kg
星座：天蝎座
绰号：电脑神童、左撇子
爱好：桥牌、乒乓球、围棋、电脑软件.....
最喜欢的歌星：JOHN LENNON
最喜欢的食物：汉堡包和可乐
崇拜的人：约翰洛克菲勒 拿破仑
格言：“我是王”“我能赢”
口头禅：“That's good"
理想：把世界引向未来时速之路
社交：不愿主动与人接触，但在压抑哈哈论
信仰：基督教
情绪：突发性的惊慌愤怒
观念：二十一世纪，电脑和因特网会使各地财富和权利分配更加平均
管理：“螺旋桨头脑”和“达尔文式管理（适者生存，不适者淘汰）”是运作微软最有效的手段。数据神经系统是未来管理的必然模式。

比尔•盖茨（Bill Gates），全名威廉•亨利•盖茨（William Henry Gates），美国微软公司的董事长。他与保罗•艾伦一起创建了微软公司，曾任微软 CEO 和首席软件设计师，并持有公司超过 8%的普通股，也是公司最大的个人股东。1995 年到 2007 年的《福布斯》全球亿万富翁排行榜中，比尔•盖茨连续 13 年蝉联世界首富。2008 年 6 月 27 日正式退出微软公司，并把 580 亿美元个人财产尽数捐到比尔与美琳达•盖茨基金会 。2011 年 9 月，《福布斯》美国富豪榜发布，盖茨以 590 亿美元居次席。

比尔•盖茨是一个天才，13 岁开始编程，不到 20 岁便写出 BASIC 语言，并预言将在 25 岁成为百万富翁。31 岁便成为世界首富，并连续 13 年登上福布斯榜首的位置。

第 2 章　dev c++下载和安装

1.1　dev c++下载

在百度上输入关键词“dev c++”,即可找到 dev c++的安装程序，下载后得到一个压缩包文件。如下图。（请下载 dev c++5.4.0 及以上的版本）

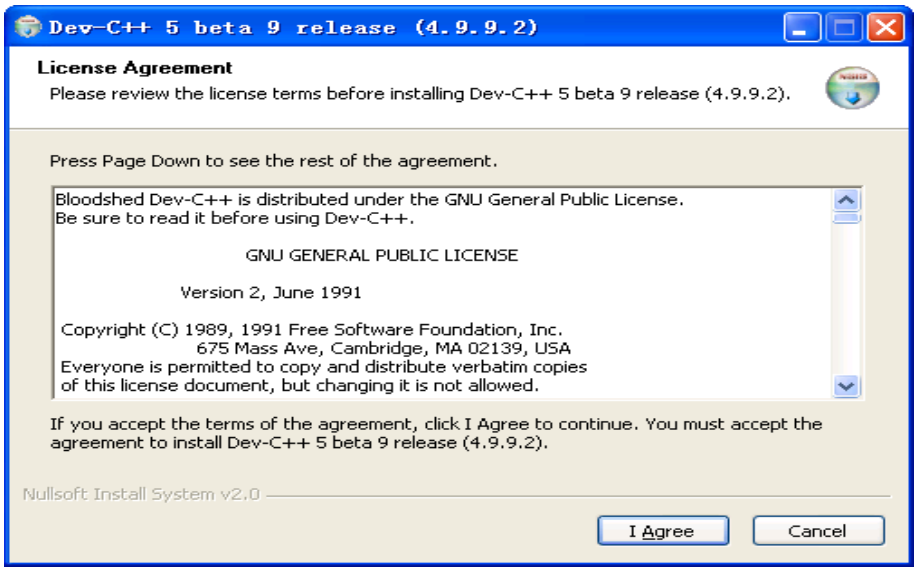
1.2　dev c++ 安装

安装按如下图示进行即可。

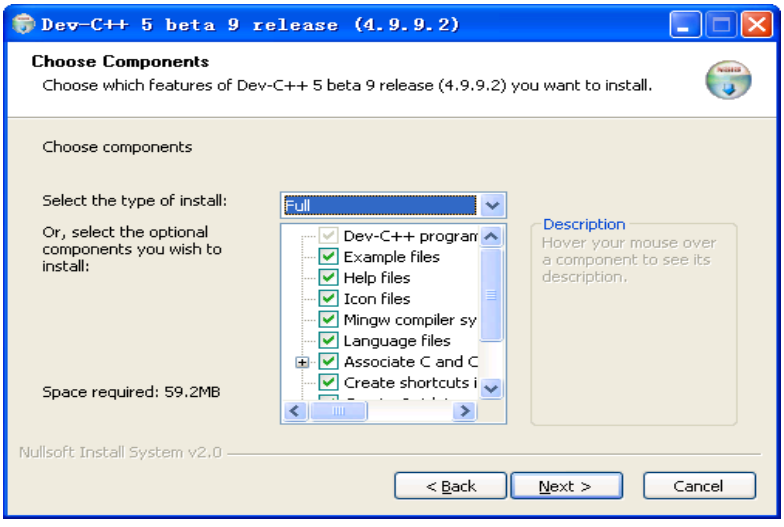
1）双击下载的压缩包文件，打开如下图。

2)双击可执行文件 devcpp-4.9.9.2 后，如下图。

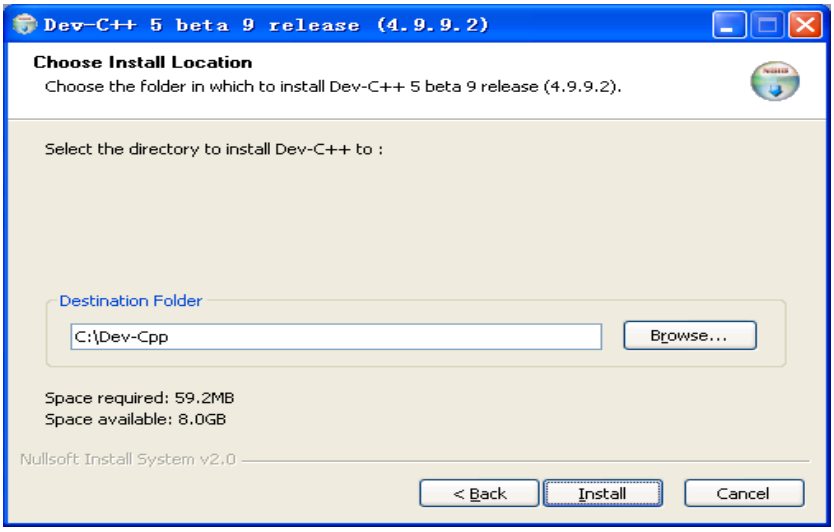
3）选择“English”，单击“OK”



4）单击“I Agree”

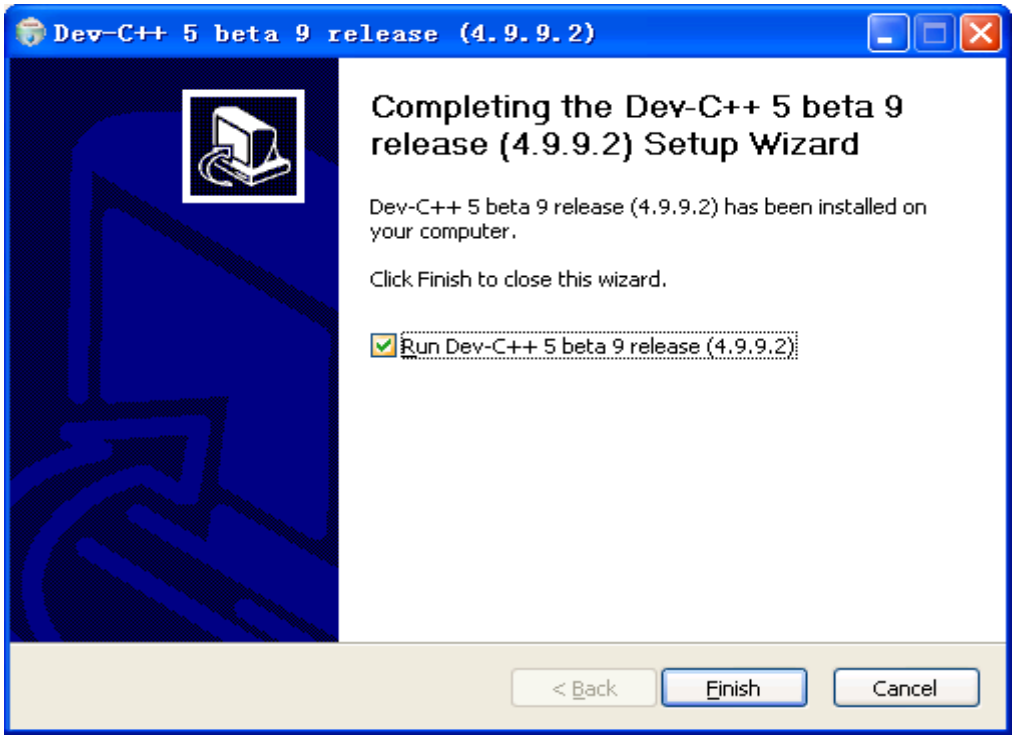


5)不改动参数，单击“Next”

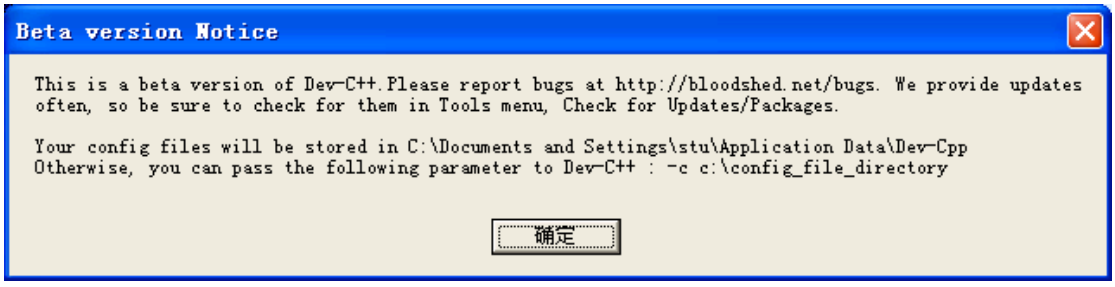


6)可以更改安装目录，当然也可不更改，单击“Install”。

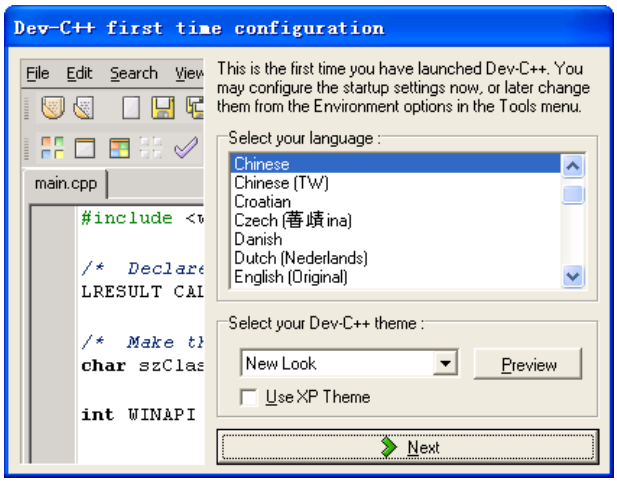
7）如果安装的Dev-c++所有用户可用，单击“是”，否则单击“否”。



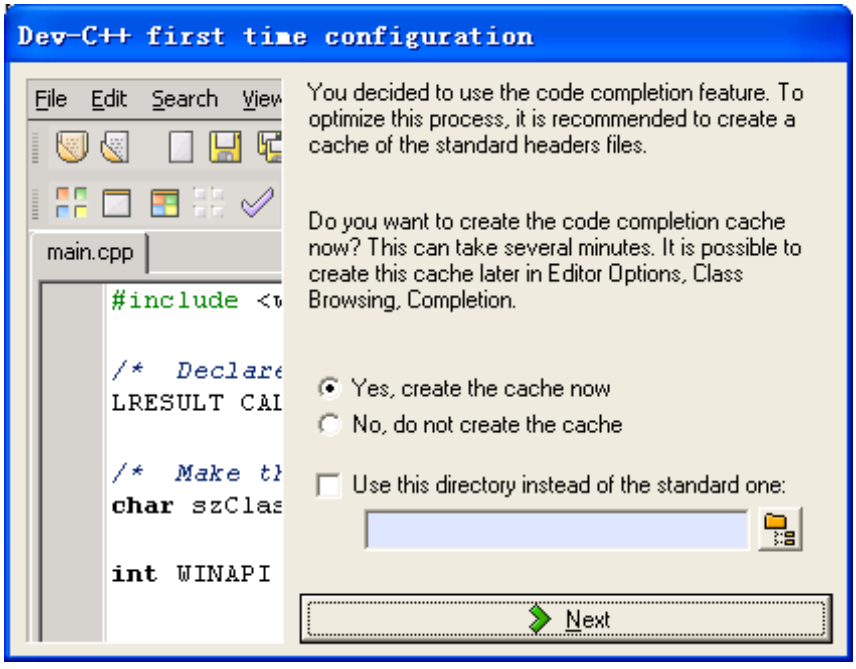
8）单击“Finish”完成，开始运行。



9）单击“确定”

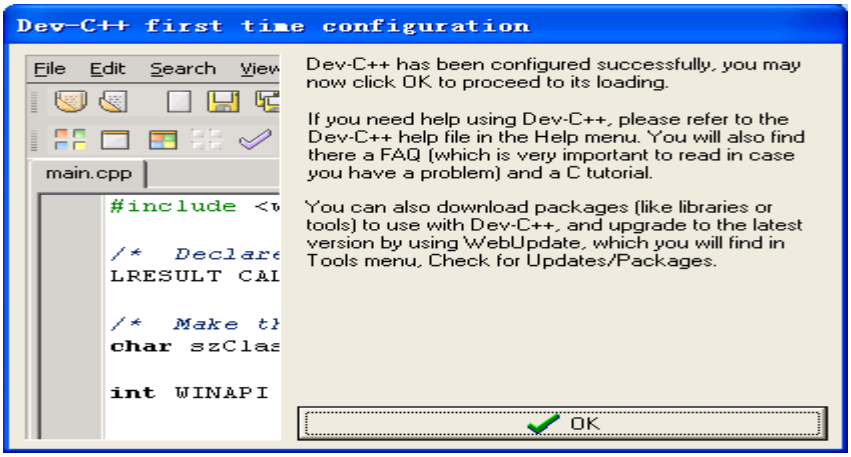


10）选择“Chinese”，单击“Next”。此处选“Chinse”，汉化成中文版。

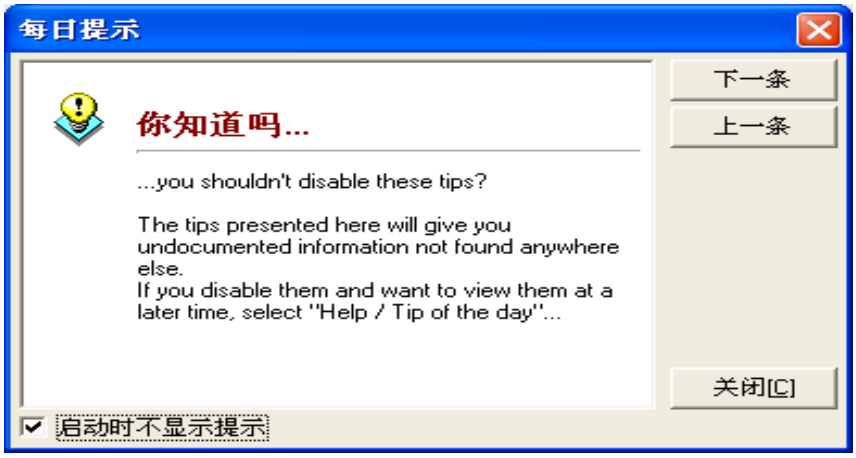


11）单击“Next”。

这里需要量几分钟时间，具体需要的时间与计算机的性能有关。

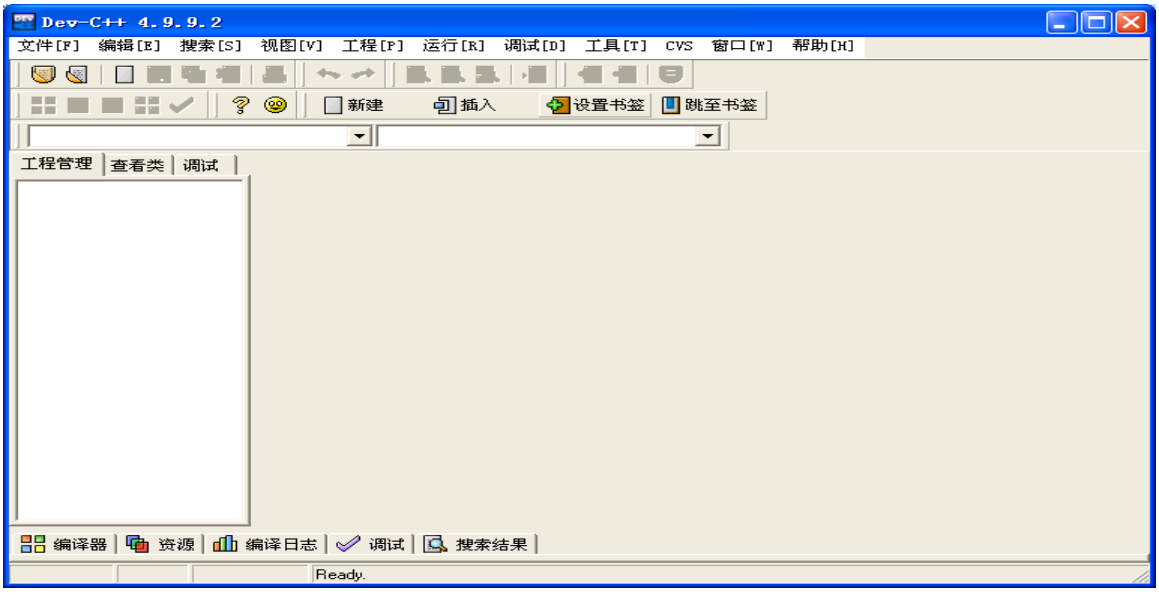


12）单击“OK”



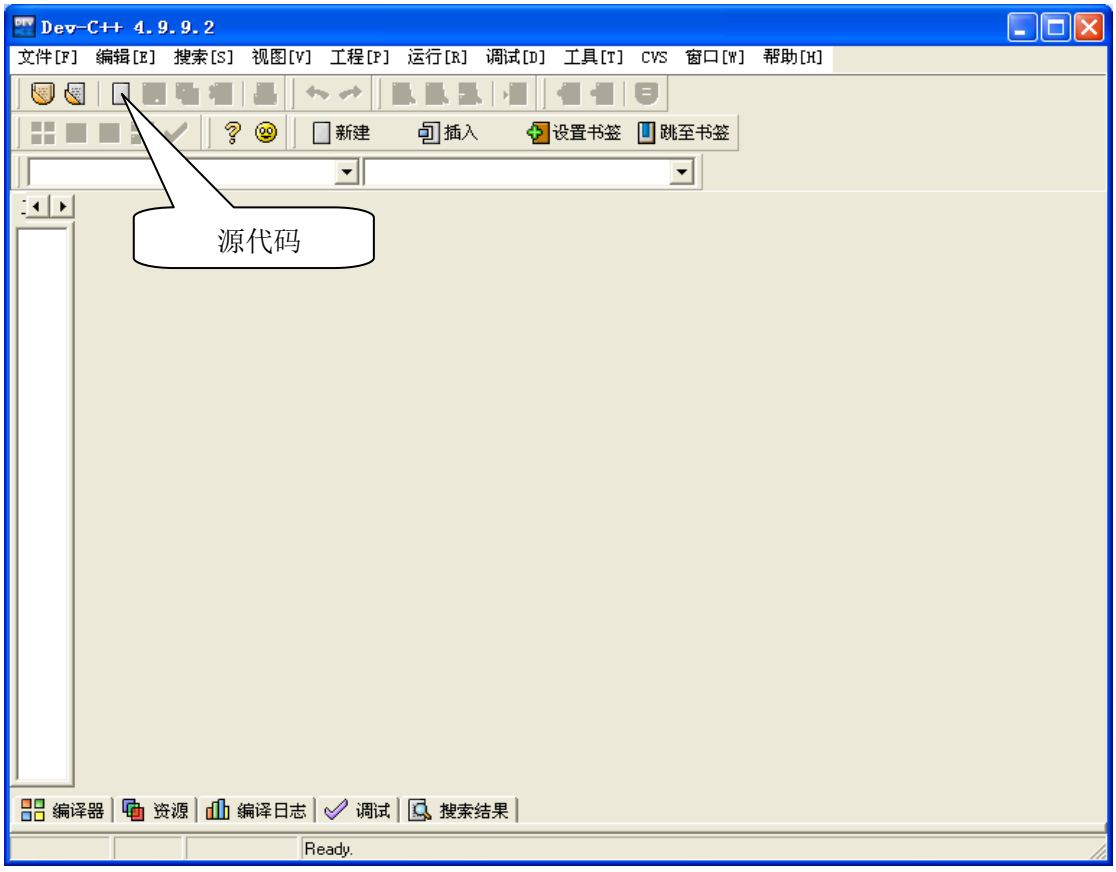
13）选择“启动时不显示提示”，然后单击“关闭”，下次运行就不会出现“每日提示”。

14）编程界面（如上图）。



1.3 进入编程界面及运行程序

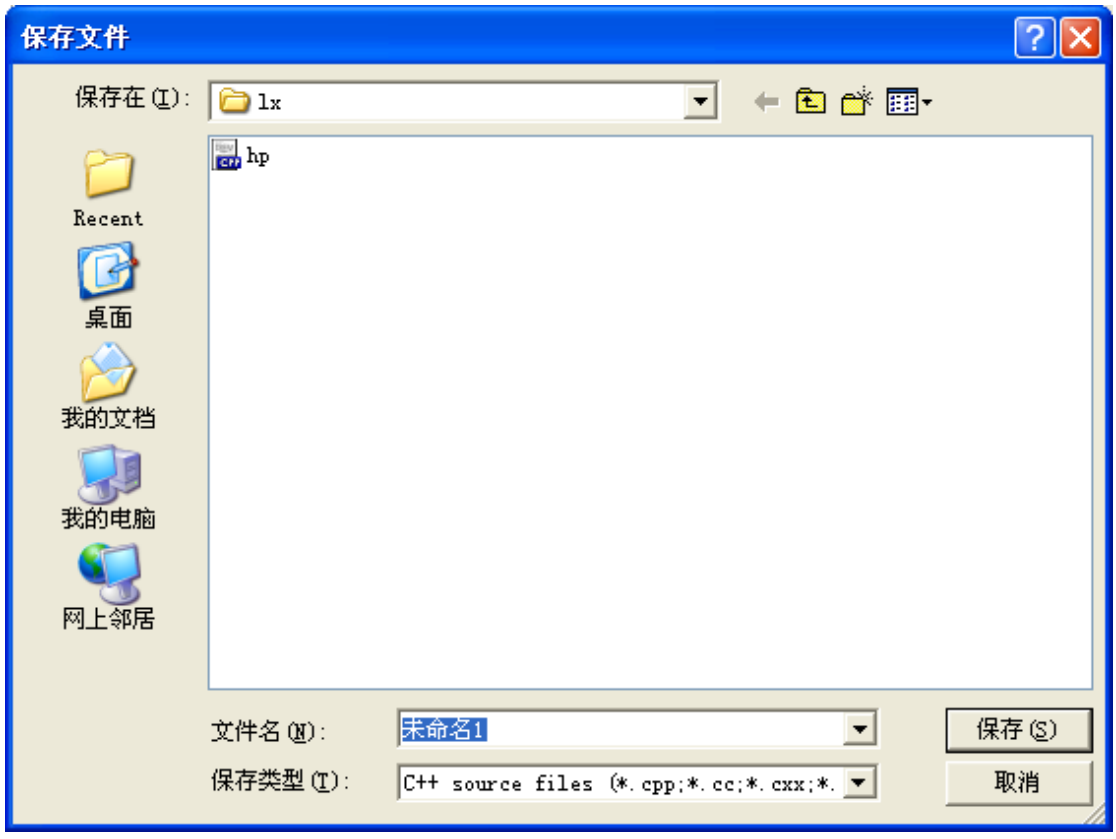
1）单击“源代码”

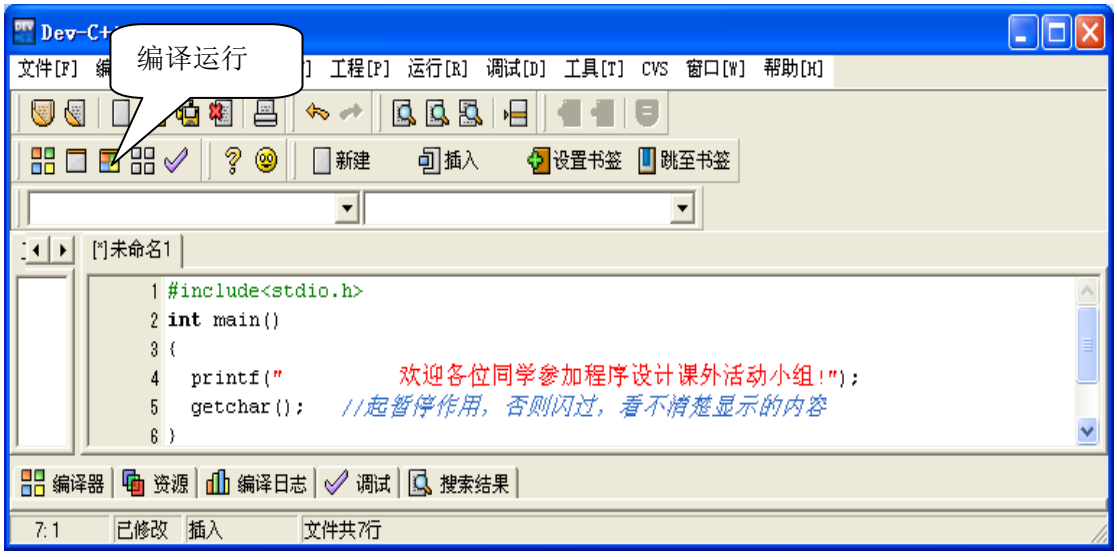


2）出现程序窗口，可以输入程序（可以按照如图输入代码）

3）编译运行

单击“编译运行”，出现如下图，选择文件保存位置，给编写的程序代码取文件名，然后单击“保存”。

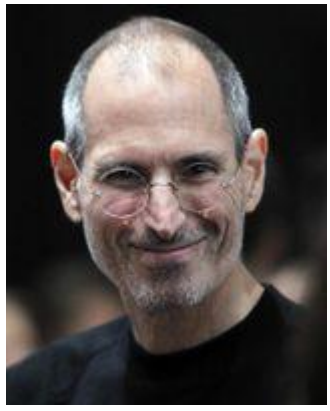

保存文件
保存在(I):
lx
hp
Recent
桌面
我的文档
我的电脑
网上邻居
文件名(N):
未命名1
保存(S)
保存类型(T):
C++ source files (*.cpp;*.cc;*.cxx;*.
取消


编译运行
文件[F]
工程[P]
运行[R]
调试[D]
工具[T]
CVS
窗口[W]
帮助[H]
新建
插入
设置书签
跳至书签
[*]未命名1
#include<stdio.h>
int main()
{
printf(" 欢迎各位同学参加程序设计课外活动小组!");
getchar(); //起暂停作用，否则闪过，看不清楚显示的内容
}
编译器
资源
编译日志
调试
搜索结果
7:1
已修改
插入
文件共7行

4）运行结果


C:\Documents and Settings\OWNER\桌面\lx.exe
欢迎各位同学参加程序设计课外活动小组!

史蒂夫·乔布斯



史蒂夫·乔布斯（Steve Jobs，1955-2011），发明家、企业家、美国苹果公司联合创办人、前行政总裁。1976 年乔布斯和朋友成立苹果电脑公司，他陪伴了苹果公司数十年的起落与复兴，先后领导和推出了麦金塔计算机、iMac、iPod、iPhone 等风靡全球亿万人的电子产品，深刻地改变了现代通讯、娱乐乃至生活的方式。2011 年 10 月 5 日他因病逝世，享年 56 岁。乔布斯是改变世界的天才，他凭敏锐的触觉和过人的智慧，勇于变革，不断创新，引领全球资讯科技和电子产品的潮流，把电脑和电子产品变得简约化、平民化，让曾经是昂贵稀罕的电子产品变为现代人生活的一部分。

1955 年 2 月 24 日，乔布斯生于美国旧金山。

1972 年毕业于加利福尼亚州洛斯阿图斯的 Homestead 高中，后入读俄勒冈州波特兰的里德学院，六个月后退学。

1976 年，乔布斯与斯蒂夫·沃兹尼亚克成立苹果公司。

1985 年，乔布斯获得了由里根总统授予的国家级技术勋章。

1997 年，成为《时代周刊》的封面人物；同年被评为最成功的管理者，是声名显赫的“计算机狂人”。

2007 年，史蒂夫·乔布斯被《财富》杂志评为了年度最有影响力的商人（most powerful businessman-Fortune） 。

2009 年，被财富杂志评选为这十年美国最佳行政总裁，同年当选《时代周刊》年度风云人物之一。

乔布斯的生涯极大地影响了硅谷风险创业的传奇，他将美学至上的设计理念在全世界推广开来。他对简约及便利设计的推崇为他赢得了许多忠实追随者。乔布斯与沃兹尼亚克共同使个人计算机在 70 年代末至八十年代初流行开来，他也是第一个看到鼠标的商业潜力的人。

1985 年，乔布斯在苹果高层权力斗争中离开苹果并成立了 NeXT 公司，瞄准专业市场。

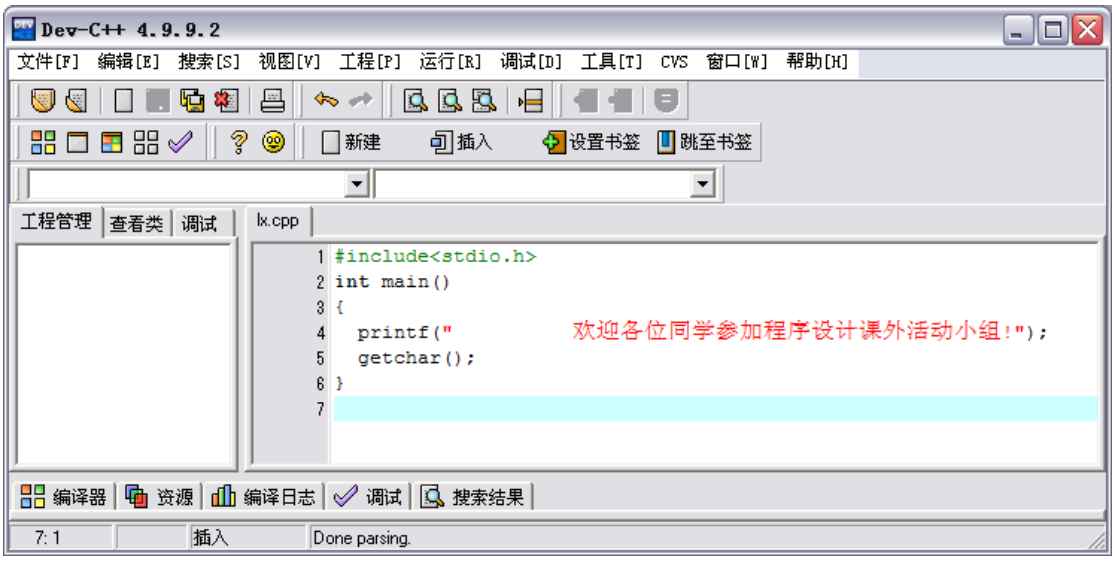
1997 年，苹果收购 NeXT，乔布斯回到苹果接任行政总裁（CEO）。

2011 年 8 月 24 日，乔布斯辞去苹果公司行政总裁职位。

2011 年 10 月 5 日逝世，终年 56 岁。

第 3 章　　顺序结构

例 3.1 在屏幕上输出如下信息：
欢迎各位同学参加程序设计课外活动小组！



参考程序：

```
#include<stdio.h>
int main()
{
  printf("        欢迎各位同学参加程序设计课外活动小组!");
  return 0;
}
```

运行结果：

说明：（1）printf()输出小括号内的内容。字符串（用双引号括起来的一串字符）原样照印，即输出双引号""内的内容。这里双引号只能是英文输入法下的，不能是中文双引号。

（2）//后面的内容表示注释，帮助我们理解程序，对程序运行不会有任何影响，所以运行程序时可以不输入//及其后的内容。

（3）要输出空格，空格也要用双引号""括起来。

（4）每个语句一定有一个分号“;”，输入英文状态下的“;”而非中文状态下的“；”，要注意区分。

（5）"　　欢迎各位同学参加程序设计课外活动小组!"，这里是字符串常量，就是用双引号（英文双引号）括起来的一串字符（不包括双引号本身）。输出时，字符串原样照印。

例 3.2 在屏幕上输出如下信息：

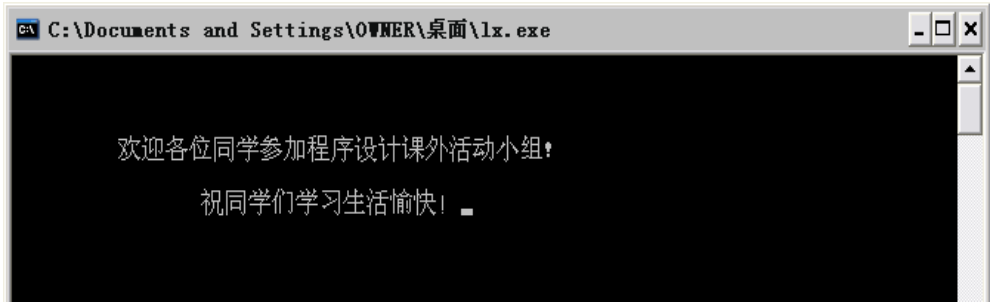
欢迎各位同学参加程序设计课外活动小组

祝同学们学习生活愉快！

参考程序：

```
#include<stdio.h>
int main()
{
  printf("\n\n\n");
  printf("    欢迎各位同学参加程序设计课外活动小组!\n\n");
  printf("              祝同学们学习生活愉快！");
  return 0;
}
```

运行结果：



说明："\n"是换行符，一个"\n"就输出一个换行。

例 3.3 在屏幕上输出如下信息：

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

参考程序：

```
#include<stdio.h>
int main()
{
  printf("\n\n\n");
  printf("                *\n");
  printf("               ***\n");
  printf("              *****\n");
  printf("             *******\n");
  return 0;
}
```

说明：语句 printf("　　*\n");的作用是输出双引号内的空格和*后换行，下一个输出语句从下一行输出。

至此，我们已经运行的 3 个 c 程序了,对 c 程序中使用的单词和 c 程序结构也有一个大致了解。

现在强调以下几点：

（1）经常使用的几个单词：

include 包括、包含

main 主要的

stdio=standard input&output 标准输入、输出

stdio.h 标准输入、输出库

int=integer 整数

printf=print function 打印、输出函数

scanf=scan function 扫描、输入函数

getchar=get a character 得到一个字符、读入一个字符

（2）程序框架结构

```
#include<stdio.h>
int main()
{
  ……
  return 0;
}
```

每个语句后有分号“;”，表示一小节完了。一定要通过多练习熟练掌握程序的框架结构。

（3）由于初次开始运行程序，学生一定要仔细观察教师演示操作全过程。

（4）学生要多练习，如输出“Hello,my name is…!” ，熟能生巧。

例 3.4 编程计算 12345+67890 的值

参考程序：

```
#include<stdio.h>
int main()
{
   printf("%d",12345+67890);
   return 0;
}
```

运行结果：80235

说明：

（1）printf("%d",12345+67890);输出字符中的%d,用后面整数表达式 12345+67890 的值来替换，所以输出 80235。

（2）数 12345 和 67890 整型常量。常量是程序运行过程中不可改变的量。比如，前面用到的字符串常量、整型常量，后面将会用到实型常量、字符常量。

（3）本程序中没有写“getchar();”语句。运行程序时，在“return 0;”语句之前加上。以后我们的程序中均不写，运行程序时加上即可。

（4）“return 0;”语句的作用是返回 0。c 语言程序的最后都应该加上该语句。

例 3.5 输入两个整数，输出这两个数的和。

分析：输入两个正数 a，b; （1）**输入已知**

计算 a+b-->c （2）**处理（由已知求结果）**

输出 c　　（3）**输出结果**

程序设计的三大步

参考程序

```
#include<stdio.h>
int main()
{
  int a,b,c;                  //int 是整数类型，定义整型变量 a,b,c
   //c 语言变量是先定义后使用，所以后面才可能使用变量 a,b,c
  scanf("%d%d",&a,&b);    //输入已知
   //输入两个整数送到变量 a 和 b 地址存放,所以每个变量前要加&(取地址运算符)
  c=a+b;                  //处理（由已知求结果）
  printf("%d",c);           //输出结果
  return 0;
}
```

（1）输入 324 432434 输出 432758

（2）输入　　输出

-4342　　338101

342443

说明：

（1）变量即其值可以改变的量。每一个变量，实际上是对应在内存中分配的存储单元，该存储单元中的值可以改变的。c 语言中，**变量是先定义后使用**，即使用这个变量之前要先定义，定义了变量后才能使用。

定义变量的格式：类型标识 变量名表

如 int a,b; 即定义的两个整型变量 a 和 b，变量 a 和 b 只能保存整数。

后面学了其它数据类型，就可能定义其它类型的变量了。

（2）“c=a+b;”是赋值语句。

赋值表达式：变量=表达式

功能：把赋值号“=”右边表达式的值计算出来，保存到左边的变量中。

方向性：变量←表达式。

赋值表达式后加上“;”构成赋修值语句。

（3）scanf("%d%d",&a,&b);输入两个整数，可以采取两种方式输入，第一种方式：一行输入两个整数，用空格隔开；第二种方式：两行输入，每行输入一个数。

（4）c 语言中的算术运算符

名称	算术运算符		示例	
	数学	C 语言	数学	C 语言
加	+	+	a+b	a+b
减	-	-	a-b	a-b
乘	×或·或不写	*	a×b 或 a·b 或 ab	a*b
除	÷或/	/	a÷b 或 a/b 或 $\frac{a}{b}$	a/b
求余数		%	整数 a 除以整数 b 的余数	a%b

注意：

1）两个整数相除结果为整数；被除数和除数只要有一个是实数，其结果为实数。

2）取余数（%）只是针对整数而言。

（5）本程序在语句“return 0;”之前加一个“getchar();”，还是暂停不下来，那就再加一个“getchar();”即可。

例 3.6 输入两个整数，输出它们的商。

分析：输入两个正数 a,b;　　（1）**输入已知**

计算 a/b-->c　　（2）**处理（由已知求结果）**

输出 c　　（3）**输出结果**

程序设计的三大步

参考程序

```
#include <stdio.h>
int main()
{
   int a,b,c;
   scanf("%d,%d",&a,&b);//输入的两个整数用“,”隔开
   c=a/b;
   printf("%d",c);
   return 0;
}
```

输入 2,3　输出 0

说明：

（1）两个整数相除，结果为整数。

（2）整数 int、long　实数 double、float

（3）printf(格式控制,输出表列),该函数包含于 stdio.h 库中,所以程序最前面要写#include <stdio.h>

（4）scanf(格式控制,地址表列），该函数包含于 stdio.h 库中,所以程序最前面要写#include <stdio.h>

（5）输入输出格式 整数 %d　实数 %lf

%nd 表示输出的**整数**占 n 列宽，如果该数不足 n 列前面补空格，如果该数超过 n 个字符，则不受 n 的限制。

%n.mlf 表示输出的**实数**占 n 列宽，其中小数位，四舍五入占 m 位。如果该数不足 n 列前面补空格，如果该数超过 n 个字符，则不受 n 的限制。

数据类型：

名称	类型符	数的范围	输入输出格式符
整数	int 或 long	-2147483648~2147483647	%d
	long long	-9223372036854775808~9223372036854775807	%lld
实数	float	1.2×10^{-38}~$1.2\times10^{+38}$ -1.2×10^{-38}~$-1.2\times10^{+38}$	%f

	double	2.3×10^{-308}~$1.7\times10^{+308}$ 2.3×10^{-308}~$1.7\times10^{+308}$	%lf
字符	char	-128~127	%c
布尔	bool	只有真和假 0—假　非0—真	一般不输入输出

实际上，一般不需要知道数的具体范围，如知道 int 在-21 亿~+21 亿，long long 在-9 万多万亿~+9 万多万亿，就足够了。

例 3.7 输入语文、数学、外语 3 科成绩，输出总分和平均分

样例　输入　91 92 95　　输出　总分 278.0 平均分 92.67

分析：已知 3 个数 yw,sx,wy;

　　计算总分 zf、平均分 pj

　　输出 zf，pj

参考程序：

```
#include<stdio.h>
int main()
{
  double yw,sx,wy,zf,pj;//成绩,总分,平均分可能有小数,是实数类型
  scanf("%lf%lf%lf",&yw,&sx,&wy);
  zf=yw+sx+wy;
  pj=zf/3;
  printf("总分%0.1lf 平均分%0.2lf",zf,pj);// %0.2lf 四舍五入保留 2 位小数
  return 0;
}
```

输入 134 125 141.5　输出 总分 400.5 平均分 133.50

说明：

　定义变量之前，想清楚变量要保存什么类型的数据，是整数，是实数，还是字符。

例 3.8　输入圆的半径，输出圆的周长和面积（四舍五入结果保留 2 位小数）。

分析：已知半径 r

　　计算周长 c、面积 s

　　输出 c、s

参考程序：

```
#include<stdio.h>
#define PI 3.1415926     //指定了一个符常量 PI，行尾无“;”
int main()
{
  double r,c,s;
  scanf("%lf",&r);
  c=2*PI*r;
  s=PI*r*r;
```

```
    printf("周长=%0.2lf 面积=%0.2lf",c,s);
    return 0;
}
```

输入 4　输出 周长=25.13 面积=50.27

说明：#define PI 3.1415926 是符号常量，在程序其它位置使用 PI 编译运行程序时，都会替换成 3.1415926。

符号常量，即用#define 指令，指定一个符号名称代表一个常量

（1）指定符号常量，注意行尾没有“;”

（2）一般在程序开始指定，不能在 int main( )之内指定。

例 3.9 输入一个三位正整数，输出各位的数字，各数字之间用“,”隔开。

分析：已知一个三位正整数 a

求出这个三位正整数的百位 b,十位 s,个位 g

输出 b,s,g

关键是怎么求出百位 b,十位 s,个位 g 呢?

b=a/100 (两个整数相除，结果是整数)

s=a%100/10 或 a/10%10,还记得吧，%是取余数

g=a%10

参考程序：

```
#include<stdio.h>
int main()
{
    int a,b,s,g;
    scanf("%d",&a);
    b=a/100;
    s=a%100/10;
    g=a%10;
    printf("%d,%d,%d",b,s,g);
    return 0;
}
```

输入 203　输出 2,0,3

说明：给定一个位数确定的整数，分离出各位的数字。这是我们经常会用到的，一定要在理解的基础上掌握。

例 3.10 你知道华氏温度吗？

华氏温标（Fahrenheit temperature scale）符号℉，1724 年，荷兰人华伦海特制定了华氏温标，他把一定浓度的盐水凝固时的温度定为 0℉，把纯水凝固时的温度定为 32 ℉，把标准大气压下水沸腾的温度定为 212℉，中间分为 180 等份，每一等份代表 1 度，这就是华氏温标，用符号 F 表示，这就是华氏温度。

摄氏温度，冰点时温度为 0 摄氏度，沸点为 100 摄氏度，而华氏温度把冰点温度定为 32 华氏度，沸点为 212 华氏度 ，所以 1 摄氏度（1 等份）等于 9/5 华氏度。

$$\text{摄氏度}=\frac{5}{9}(\text{华氏度}-32)$$

任务：编程完成：输入一个华氏温度，输出摄氏温度

分析：已知华氏温度 F

求摄氏温度 C＝5/9(F-32)

输出 C

参考程序：

```
#include<stdio.h>
int main()
{
  double F,C;  //c 语言中区分大小写
  scanf("%lf",&F);
  C=5.0/9*(F-32);  //写成 5/9*(F-32)呢？
  printf("%0.2lf\n",C);
  return 0;
}
```

输入 212 输出 100.00

说明

（1）c 语言中是要区分大小写字母的。int a,A;是定义了两个不同的变量。

（2）“printf("%0.2lf\n ",C); ”格式控制符最后有一个'\n'表示输出内容后会换行。

例 3.11 输入三角形的三边长，输出三角形面积。

注：三角形的三边 a,b,c，则该三角形的面积公式为：$s=\sqrt{p(p-a)(p-b)(p-c)}$

其中 p = (a+b+c)/2

该公式称为海伦公式。

数学函数

数学	例子	c 语言	包含于头文件
$\sqrt{x}$	$\sqrt{4}$ ＝2	sqrt(x)	math.h
$\lvert x\rvert$	$\lvert\pm 2.3\rvert$ =2.3	fabs(x)	math.h

分析：

已知 a、b、c

P=(a+b+c)/2

t=p(p-a)(p-b)(p-c)

$s=\sqrt{t}$

求 s

参考程序：

```
#include <stdio.h>
#include <math.h>  //sqrt()函数包含于 math.h 库中
int main()
{
  double a,b,c,p,s;
  scanf("%lf%lf%lf",&a,&b,&c);
```

```
    p=(a+b+c)/2;       //先算出 p 值,才能在下一个语句计算面积
    s=sqrt(p*(p-a)*(p-b)*(p-c)); //注意括号配对,表达式改变运算顺序,只能使用小括号();
    printf("s=%0.3lf\n",s);
    return 0;
}
```

输入 3 4 5 输出 s=6.000

例 3.12 输入一元二次方程的系数 a、b、c 的值，输出实根，四舍五入保留 4 位小数（输入数据保证方程有实根）。

分析：已知 a、b、c（输入）

利用求根公式计算 x1、x2（处理）

求 x1、x2（输出）

参考程序：

```
#include <stdio.h>
#include <math.h>
int main()
{
    double a,b,c,d,x1,x2;
    scanf("%lf%lf%lf",&a,&b,&c);
    d=b*b-4*a*c;
    x1=(-b+sqrt(d))/2/a;
    x2=(-b-sqrt(d))/2/a;
    printf("x1=%0.4lf,x2=%0.4lf\n",x1,x2);
    return 0;
}
```

输入 1 2 -3 输出 x1=1.0000,x2=-3.0000

说明：

输入一元二次方程系数一定要满足判别式 d>=0，否则运行结果将会有误。如输入 1 2 3，输出 x1=-1.#IND,x2=-1.#IND，这是因为 d<0，该一元二次方程无实根。

例 3.13 输入一个大写字母，输出对应的小写字母。

ASCII 表

ASCII 值	控制字符	ASCII 值	控制字符	ASCII 值	控制字符	ASCII 值	控制字符
0	NUT	32	(space)	64	@	96	、
1	SOH	33	!	65	A	97	a
2	STX	34	”	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	,	71	G	103	g

8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DCI	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	X	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	—	127	DEL

分析：已知一个大写字母存于变量 c1 中

变成小写字母存于变量 c2 中，即 c2=c1+32

输出变量 c2

说明：c 语言中保存字符实际上是保存字符的 ASCII 值。同一个英文字母，小写字母 ASCII 值与大写字母 ASCII 值相差 32,所以 c2=c1+32

参考程序：

```
#include<stdio.h>
int main()
{
  char c1,c2;
  scanf("%c",&c1);
  c2=c1+32;
  printf("%c",c2);
  return 0;
}
```

输入 B 输出 b

说明：

（1）c 语言中保存字符是保存字符的 ASCII 值。如 char x='a'，实际上 x 中保存的是 97。

（2）**字符'a'的 ASCII 值是 97**,符'b'的 ASCII 值是 98,...,符'z'的 ASCII 值是 122。

字符'A'的 ASCII 值是 65,符'B'的 ASCII 值是 66,...,符'Z'的 ASCII 值是 9……

字符'0'的 ASCII 值是 48,符'1'的 ASCII 值是 49,...,符'9'的 ASCII 值是 57。

所以只需记得'a'是 97, 'A'是 65, '0'是 48, 就可推算出其它英文字母和数字的 ASCII 值。

（3）运行程序时，我们要在“return 0;”之前加两个“getchar();”，其原因：运行本程序输入'B'后，按回车。实际上是输入了两个字符：B 和回车符。“scanf("%c",&c1);”读字符'B'，第一个“getchar();”读回车符，第二个“getchar();”没有对应的字符可读，所以停下来了。

例 3.14 输入两个整数存于变量 a,b 中，交换这两个变量的值然后输出。

分析：已知两个整数 a,b

交换 a,b 之值

输出 a,b

关键是交换 a,b 之值怎么进行呢？交换两满杯的饮料是怎么进行的呢？借助第三个杯子。我们这里就借助第三个变量，设为 t。下表演示了交换过程。

	a	b	t
交换前	1	2	
t=a	1	2	1
a=b	2	2	1
b=t	2	1	1

参考程序：

```
#include<stdio.h>
int main()
{
  int a,b,t;
  scanf("%d%d",&a,&b);
  t=a;
  a=b;
  b=t;
  printf("%d %d",a,b);
  return 0;
}
```

输入 2 3　输出 3 2

说明：交换两个变量 a,b 之值：t=a;a=b;b=t;即从第三个变量开始，首尾相连，一定要掌握，后面经常用到。

例 3.15 阅读程序写出运行结果。

```
#include<stdio.h>
int main()
{ int a,b;
  a=3;
```

```
  b=5;
  printf("%d %d\n",a,b);
  a=a+b;
  b=a-b;
  a=a-b;
  printf("%d %d\n",a,b);
  return 0;
}
```

分析：该程序中a、b的值在不断变化，可以列一张表来观察变量的变化情况。

执行语句	a	b
a=3; b=5;	3	5
a=a+b;	8	5
b=a-b;	8	3
a=a-b;	5	3

由此可以看出，三个语句的作用也是交换二个变量的值。

所以，该程序的运行结果：

3 5

5 3

说明：交换两个变量a、b之值有两种方法：

（1）引入中间变量t：t=a;a=b;b=t;

（2）不用中间变量

```
  a=a+b;    //现a变为原a、b之和
  b=a-b;    //现b变为(原a、b之和 )a-原b即为原a
  a=a-b;    //现b变为(原a、b之和 )a-现b即为原b
```

这两种交换两变量之值，第（1）种方法一定要在理解的基础上牢固掌握，第（2）种方法能理解就行了。

习 题

3.1 输入一个字符，输出它的ASCII码。

3.2 编程完成：用“*”拼出“中国”两个汉字。

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

3.3 输入两个整数，输出商和余数。

3.4 输入两个实数存入变量a和b中，交换a、b之值，并输出a、b之值。

3.5 输入一个三位整数，反向输出这个三位整数。如输入123，输出321。

3.6 输入三个字符并分别存入变量 a，b，c 中，请依次将 b 赋给 a，c 给赋 b，c 最后应为 a 最初的值。

约翰·冯·诺依曼

约翰·冯·诺依曼（ John von Neumann，1903－1957），美籍匈牙利人，1903 年 12 月 28 日生于匈牙利的布达佩斯。在发明电子计算机中冯·诺依曼所起到关键性作用，他被西方人誉为“计算机之父”. 而在经济学方面，他也有突破性成就，被誉为“博弈论之父”。在物理领域，冯·诺依曼在 30 年代撰写的《量子力学的数学基础》已经被证明对原子物理学的发展有极其重要的价值。在化学方面也有相当的造诣，曾获苏黎世高等技术学院化学系大学学位。与同为犹太人的哈耶克一样，他无愧是上世纪最伟大的全才之一。

第 4 章　选择结构

例 4.1　某航空公司对旅客随身携带的行李收费标准为：2 0 公斤以下（包括 20 公斤）不收费，超过 20 公斤的部分每公斤收 20 元。编一程序，输入旅客行李重量，如果重量超标，则输出收费总额，如果重量不超标，则打印 0 （表示免费）。

分析：

已知行李重量为 x 公斤

如果 X 超过 20　Y=(x-20)*20

否则 Y=0

求费用 Y 元

这需要用条件语句：

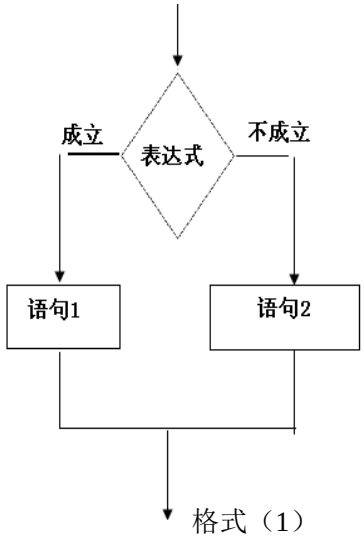
if 语句

格式：

（1） if（表达式）语句 1

　　else 语句 2

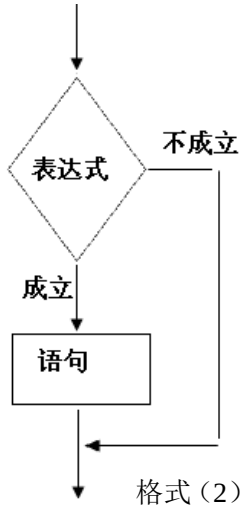
功能：如果表达式成立，则执行语句 1，否则执行语句 2。执行过程如右图



格式（1）

（2）if（表达式）语句

功能：如果表达式成立，则执行语句。执行过程如右图。



格式（2）

（3）if（表达式 1）语句 1

　　else　if（表达式 2）语句 2

　　…

　　else　if（表达式 n）语句 n

　　else　语句 n+1

功能：如果表达式 1 成立，则执行语句 1，否则（在表达式 1 不成立的情况下） 如果表达式 2 成立，则执行语句 2，……，（在表达式 n-1 不成立的情况下）如果表达式 n 成立，则执行语句 n，否则执行语句 n+1。

参考程序

```
#include<stdio.h>
int main()
{
  double x,y;
  scanf("%lf",&x);
  if (x>20)       //表达后不能写";"
    y=(x-20)*20; //后面有";"
  else
    y=0;          //后面有";"
  printf("y=%0.1lf",y);
```

```
  return 0;
}
```

①输入 15　输出 y=0.0　②输入 32.3　输出 y=246.0

说明：

（1）if (x>20)

```
      y=(x-20)*20;
    else
      y=0;
```

这是一个条件语句。

（2）条件语句的表达式一定要用( )括起来。

（3）语句 1 和语句 2 一定包括“;”。

（4）不能表达式后面加上“;”。因为单独一个“;”是一空语句。如果加了“;”，表达式后面就有两个语句，这样运行就会出错。

```
    if (x>20) ;  //此处多了一个“;”，成了一个空语句。出错!
      y=(x-20)*20;
    else
      y=0;
```

例 4.2 输入一个实数，输出其绝对值。

分析：

已知实数 a，若 a<0 则 a=-a，否则 a 不变。

参考程序：

```
#include<stdio.h>
int main()
{
  double a;
  scanf("%lf",&a);
  if (a<0) a=-a;
  printf("%lf\n",a);
  return 0;
}
```

① 输入 2.75　输出 2.750000　② 输入 -32.3　输出 32.300000

例 4.3 输入两个整数，输出这两个数的最小值。

分析：

已知两个整数 a,b

如果 a<b 输出 a

否则输出 b

参考程序

```
#include<stdio.h>
int main()
{
```

```
    int a,b;
    scanf("%d%d",&a,&b);
    if (a<b) printf("%d",a);
    else printf("%d",b);
    return 0;
}
```

说明：

（1）也可以先把最小值求出来保存的变量 c 中，再输出。如下。

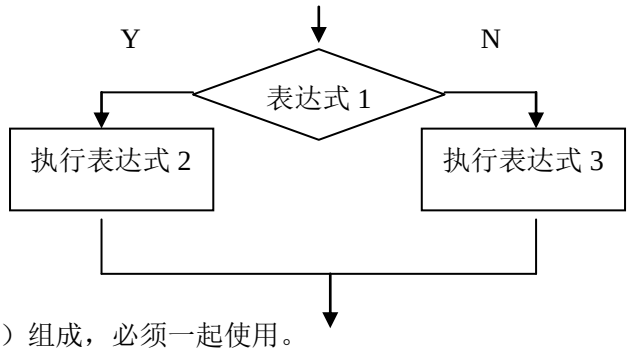
```
#include<stdio.h>
int main()
{
    int a,b,c;
    scanf("%d%d",&a,&b);
    if (a<b) c=a;
    else c=b;
    printf("%d",c);
    return 0;
}
```

（2）c 语言中条件表达式

条件表达式的一般格式：

表达式 1?表达式 2:表达式 3

功能：如右图所示。



说明：条件运算符由两个符号（?和:）组成，必须一起使用。

将参考程序的 if 语句换成“a<b?printf("%d",a):printf("%d",b);”，也可以将说明（1）中的 if 语句换成“c=a<b?a:b;”，其作用完全一样。

例 4.4 输入两个实数，按代数值由小到大的顺序输出这两个数

方法一

分析：

已知两个实数 a,b

如果 a<b 输出 a,b

否则输出 b,a

参考程序：

```
#include<stdio.h>
int main()
{
    double a,b;
    scanf("%lf%lf",&a,&b);
    if (a<b)
      printf("%lf  %lf",a,b);
    else
      printf("%lf  %lf",b,a);
```

```
        //可将 if 语句换成
        //a<b? printf("%lf  %lf",a,b): printf("%lf  %lf",b,a);
        return 0;
    }
```

方法二

分析：

已知两个实数 a,b

如果 a>b 交换 a,b 之值

输出 a,b

还记得怎么交换两个变量的值吗？t=a;a=b;b=t;

但问题来了，交换两个变量是三个语句，而 if (表达式)后只能是一个语句，怎么办呢？那就把三个语句“捆”成一个语句—复合语句。

```
{
t=a;
a=b;
b=t;
}
```

参考程序：

```
#include<stdio.h>
int main()
{
    double a,b,t;
    scanf("%lf%lf",&a,&b);
    if (a>b) //下面的语句一定要用{}括起来做成一个语句（复合语句），因为条件成立只能执行一个语句。
    {
      t=a;
      a=b;
      b=t;
    } //这里用不着“;”，加了也不起作用，相当多一个空语句
    printf("%lf  %lf",a,b);
    return 0;
}
```

① 输入 1.2 4 输出 1.200000 4.000000

② 输入 32.45 -435 输出 -435.000000 32.450000

说明：书写程序时应遵循以下的规则，以养成良好的编程风格。

书写程序时应遵循的规则：

从书写清晰，便于阅读，理解，维护的角度出发，在书写程序时应遵循以下规则：

1.一个语句占一行。

2.用{} 括起来的部分，通常表示了程序的某一层次结构。{}一般与该结构语句的第一个字母对齐，并单独占一行。

3.低一层次的语句比高一层次的语句缩进若干格后书写。以便看起来更加清晰，增加程序的可

读性。

例 4.5 输入三个数，输出这三个数中的最大值。

分析：已知三个数，设为 a,b,c，求出这三个数的最大值。

关键是怎么求出最大值呢？

只有一个数 a，a 是最大的，max=a;

max 与 b 进行比较，如果 max<b，更新 max 的值为 b ,max=b（此时 max 的值为原来 a,b 中的最大值）;

max 与 c 进行比较，如果 max<c，更新 max 的值为 c,max=c（此时 max 的值为原来 a,b,c 中的最大值)。具体操作过程如下表。

设 a=3;b=2;c=5;

操作	max	说明
max=a;	3	只有一个数 a，最大值是 a
If (max<b) max=b;	3	条件不成立，不更新 max
If (max<c) max=c;	5	条件成立，更新 max=c

于是我们得到算法为：

(1)把第一变量 a 存入 max 中。

(2)if (max<b) max=b;

(3)if (max<c) max=c;

参考程序

```
#include<stdio.h>
int main()
{
  double a,b,c,max;
  scanf("%lf%lf%lf",&a,&b,&c);
  max=a;   // 只有一个数时，它就是最大
  if (max<b) max=b; //目前最大值 max 和 b 相比，如果不是最大，则更新 max 为 b
  if (max<c) max=c;//目前最大值 max 和 c 相比，如果不是最大，则更新 max 为 c
  printf("max=%0.2lf",max);
  return 0;
}
```

(1)输入 1 2 3 输出 max=3.00 (2)输入 2 1 3 输出 max=3.00

(3)输入 3 1 2 输出 max=3.00

说明：请大家深刻理解从几个数中，找出最大值（最小值）的方法。这种程序思维相当重要。

例 4.6 输入 3 个数 a,b,c,要求按由小到大的顺序输出。

分析：

已知 3 个实数 a,b,c

如果 a>b 交换 a,b 之值

如果 a>c 交换 a,c 之值

如果 b>c 交换 b,c 之值

输出 a,b,c

参考程序：

```
#include<stdio.h>
int main()
{
  double a,b,c,t;//没有说明是整数，定义成实数类型
  scanf("%lf%lf%lf",&a,&b,&c);
  if (a>b) {t=a;a=b;b=t;} //复合语句内三个语句短小,写在一行
  if (a>c) {t=a;a=c;c=t;}
  if (b>c) {t=b;b=c;c=t;}
  printf("%0.1lf %0.1lf %0.1lf",a,b,c);
  return 0;
}
```

（1）输入 1 2 3 输出 1.0 2.0 3.0（2）输入 1 3 2 输出 1.0 2.0 3.0

（3）输入 2 1 3 输出 1.0 2.0 3.0（4）输入 2 3 1 输出 1.0 2.0 3.0

（5）输入 3 1 2 输出 1.0 2.0 3.0（6）输入 3 2 1 输出 1.0 2.0 3.0

说明：上面给出的三个不相等的数的 6 种排列，运行结果均正确。**要养成习惯：**检验满足题意的所有情况，程序是否都能得出正确结果。

上面几个题中均涉及到了关系表达式，下面进行介绍。

关系运算符和关系表达式

数学	c 语言	数学条件	c 语言表达式	优先级别
＝	＝＝	x 与 3 相等	x==3	低
≠	！＝	a 不等于 b	a!=b	
＜	＜	a 比 b+3 小	a<b+3	高
＞	＞	x 大于 1	x>1	
≤	<=	a 不大于 3	a<=3	
≥	>=	x 与 y 的和大于等于 3	x+y>=3	

注意：＝、≠、≤、≥、这四个关系运算符的写法不同于数学上，应写成= =、！＝、<= 、>=，其中要特别注意= =的写法，经常想的是判断相等却写成了＝，而且还不易察觉。

例 4.7 有一函数：$y=\begin{cases}-1.....(x<0)\\0.....(x=0)\\1.....(x>0)\end{cases}$，编程输入一个值，输出 y 值。

分析：已知 x

根据 x 值情况，计算出 y

```
If (x>0) y=1;
Else
  If (x<0) y=-1;
  Else y=0;
```

输出 y

参考程序：

```
#include<stdio.h>
```

```
int main()
{
    double x;
    int y;
    scanf("%lf",&x);
    if (x>0) y=1;
    else
      if (x==0) y=0;
      else y=-1;
    printf("%d\n",y);
    return 0;
}
```

也可写成三个独立的 if 语句

```
if (x>0) y=1;
if(x==0) y=0;
if (x<0) y=-1;
```

说明：实际上这里用到了 if 语句的嵌套，即 if 语句中又包含了 if 语句。If 语句的嵌套要注意 if 和哪个 else 配对。

配对原则：if 和最近还没有配对的 else 配对。

例如：

```
if (x>0) y=1;
else
  if (x==0) y=0;
  else y=-1;
```

```
if (x>=0)
  if (x==0) y=0;
  else y=1;
else y=-1;
```

上面这两种情况，程序是正确的。

例 4.8 给出一个百分制成绩，90 分以上输出’A’,80-89 分输出‘B’,70-79 分输出‘C’，60-69 分输出‘D’，60 分以下输出‘E’。

分析：已知成绩 cj;

根据成绩求出等级 dj

输出 dj

参考程序：

```
#include <stdio.h>
int main()
{
  double cj;
  char dj;
  scanf("%lf",&cj);
  if (cj>=90) dj='A';
  else if (cj>=80)dj='B';
  else if (cj>=70)dj='C';
  else if (cj>=60)dj='D';
  else dj='E';
  printf("%c\n",dj);
```

```
  return 0;
}
```

例 4.9

输入 a+b 或 a-b 或 a*b 的形式，输出 a+b 或 a-b 或 a*b 的值

输入　　输出

1.2+3.2　　1.20+3.20＝4.40

1.2-3.2　　1.20-3.20＝-2.00

1.2*3.2　　1.20*3.20＝3.84

即结果保留 2 位小数。

分析：已知 a,f,b

如果 f 是+，c=a+b;

如果 f 是-，c=a-b;

如果 f 是*，c=a*b;

求 c(a,b 经过运算得到的值)

参考程序：

```
#include <stdio.h>
int main()
{
  double a,b,c;
  char f;      //注意运算符是字符，输入输出格式用%c
  scanf("%lf%c%lf",&a,&f,&b);
  if (f=='+') c=a+b;
  if (f=='-') c=a-b;
  if (f=='*') c=a*b;
  printf("%0.2lf%c%0.2lf=%0.2lf",a,f,b,c);
  return 0;
}
```

例 4.10　输入三个正数，如果能构成三角形输出“Yes”,不能则输出“NO”。

分析：已知三个正数 a,b,c;

if 能构成三角形 则输出“Yes”,不能则输出“NO”。

能构成角形的判断条件：任意两边之和大于第三边。怎么来表示任意两边之和大于第三边呢？这里实际上就是三种情况：a+b>c、a+c>b、b+c>a。这三种情况要同时成立，就可写成 a+b>c 且 a+c>b 且 b+c>a。

逻辑运算符及逻辑表达式

逻辑运算符		逻辑表达式	
数学	c 语言	数学	c 语言
并且	&& （逻辑与）	x 大于等于 2 小于等 10	x>=2&&x<=10
或	|| （逻辑或）	x 小于 2 或 x 大于 10	x<2||x>10
非、不	! （逻辑非）	x 不大于 2	!(x>2)

参考程序：

```
#include<stdio.h>
int main()
{
    double a,b,c;
    scanf("%lf,%lf,%lf",&a,&b,&c);
    if  (a+b>c&&a+c>b&&b+c>a)
        printf("YES\n");
    else
        printf("NO\n");
    return 0;
}
```

（1）输入 1,2,3 输出 NO （2）输入 3,2,4 输出 YES

例 4.11 输入一个字符，如果是英文大写字母，转换成小写字母；如果不是大写字母，则不转换。输出最后得到的字符。

分析：已知一个字符 a，求出对应的字符 b。

```
if  a 是大写字母 b=a+32;
else  b=a;
```

大写字母条件：a>='A'&&a<='Z'

也可改成条件表达式来完成： b= a>='A'&&a<='Z'?a+32:a;

参考程序：

```
#include<stdio.h>
int main()
{
    char a,b;
    scanf("%c",&a);
    b=a>='A'&&a<='Z'?a+32:a;
    printf("%c",b);
    return 0;
}
```

说明：大写字母的条件：a>='A'&&a<='Z'；小写字母的条件：a>='a'&&a<='z'；数字的条件：a>='0'&&a<='9'。

例 4.12 写一个程序，输入一个年份，输出该年是否是闰年。

如 2010 年不是闰年，输出“2010 年不是闰年!”，

2008 年是闰年，输出“2008 年是闰年!”。

分析：输入一个年份 year;

```
if  year 是闰年
  输出“是闰年”
Else
  输出“不是闰年”
```

关键问题是闰年怎么判断？分成两种情况：

（1）能被 4 整除的非整百年　year%100!=0&& year%4==0
（2）能被 400 整除年份　year%400==0
所以条件为：year%4==0&& year%100!=0|| year%400==0
多种运算符从高到低的运算顺序：
（1）! → *、/、% → +、- →<、<=、>、>=→ = =、!= →&&→||→ =
（2）同级别的运算，从左到右依次运算。
参考程序：

```
#include<stdio.h>
int main()
{
    int year;
    scanf("%d",&year);
    if  (year%4==0&&year%100!=0||year%400==0)
       printf("%d 年是闰年！",year);
    else
       printf("%d 年不是闰年！",year);
    return 0;
}
```

（1）输入 2012 输出 2012 年是闰年！
（2）输入 2011 输出 2011 年不是闰年！
（3）输入 2100 输出 2100 年不是闰年！

例 4.13 输入年和月，输出该月有多少天。
分析：已知年 year 和月 month
　　求这个月的天数 days
　怎么求天数 days 呢
可以这样考虑
方法一：分成闰年和平年两种情况处理

```
If 闰年
  {
   If  month==2
      Days=29
   Else if  month==4、6、9、11
      Days=30
   Else days=31
  }
  Else
   {
   If  month==2
      Days=28
   Else if  month==4、6、9、11
      Days=30
```

```
   Else days=31
  }
```

参考程序：

```
#include<stdio.h>
int main()
{
  int year,month,days;
  scanf("%d%d",&year,&month);
  if (year%100!=0&&year%4==0||year%400==0)
    {
      if (month==2) days=29;
      else if (month==4||month==6||month==9||month==11) days=30;
      else days=31;
    }
  else
    {
      if (month==2) days=28;
      else if (month==4||month==6||month==9||month==11) days=30;
      else days=31;
    }
  printf("%d\n",days);
  return 0;
}
```

（1）输入 2012 2 输出 29 （2）输入 2011 2 输出 28

（3）输入 2012 3 输出 31 （2）输入 2013 4 输出 30

方法二：方法一中重复处理月份是 1、3-12 月，先判断月份，分成 2 月和不是 2 月两种情况分别处理。

```
  if (month==2)
    {
      if (year%100!=0&&year%4==0||year%400==0) days=29;
      else days=28;
    }
  else
    {
      if (month==4||month==6||month==9||month==11) days=30;
      else days=31;
    }
```

方法三：分成（1）闰年 2 月；（2）其它情况。

```
 if ((year%100!=0&&year%4==0||year%400==0)&&month==2)
    days=29;
  else
    {
```

```
    if (month==2) days=28;
    else if (month==4||month==6||month==9||month==11) days=30;
    else days=31;
}
```

方法四：先都按平年处理得到天数，然后，修正闰年 2 月的结果

```
  if (month==2) days=28;
  else if (month==4||month==6||month==9||month==11) days=30;
  else days=31;
    if ((year%100!=0&&year%4==0||year%400==0)&&month==2) days=29;
```

说明：方法二、三、四的参考程序只须将方法一的参考程序中的 if 语句改成每种方法的 if 语句即可。

例 4.14 求 $ax^2+bx+c=0$ 方程的解。

分析：已知 a,b,c

根据 a 的情况进行处理，如下

```
If a==0
      {
       处理 a==0 的情况：即输出“不是一元二次方程!”
      }
Else
     {
       处理 a≠0 的情况：
       d=b*b-4ac
       If d>=0
         {
           计算 x1,x2
           输出 x1,x2
         }
       Else
         输出“一元二次方程无解!”
    }
```

参考程序：

```
#include<stdio.h>
#include<math.h>
int main()
{
  double a,b,c,d,x1,x2;
  scanf("%lf%lf%lf",&a,&b,&c);
  if (a==0)
    printf("不是一元二次方程! ");
  else
  {
```

```
    d=b*b-4*a*c;
    if (d>=0)
    {
     x1=(-b+sqrt(d))/(2*a);
     x2=(-b-sqrt(d))/(2*a);
     printf("x1=%lf  x2=%lf",x1,x2);
    }
    else
    printf("无解！");
  }
return 0;
}
```

说明：初一、初二学生最好不讲此题，他们还没有学习一元二次方程和算术方根函数。

例 4.15 输入一个不多于 3 位的正整数，要求

（1） 求出它是几位数

（2） 分别输出每一位数字

（3） 按逆序输出各位数字，例如原数为 321，应输出 123

分析：已知一个不多于 3 位的正整数 a

百位：x3=a/100　十位：x2=a%100/10　个位：x1=a%10

位数 d：根据 x3,x2,x1 的情况进行判断

输出位数 d，

根据位数,输出每一位数字 x3,x2,x1，逆序输出这个数数字 x1,x2,x3

参考程序：

```
#include <stdio.h>
int main()
{
    int a,x1,x2,x3,d;
    scanf("%d",&a);
    x3=a/100;
    x2=a%100/10;
    x1=a%10;
    if (x3!=0) d=3;
    else  if (x2!=0) d=2;
    else  d=1;
    printf("%d 是%d 位数\n",a,d);
    if (d==3)
     {
       printf("%d 的数字分别为%d,%d,%d\n",a,x3,x2,x1);
       printf("%d 的逆序数字分别为%d,%d,%d\n",a,x1,x2,x3);
    }
  else
```

```
  if (d==2)
   {
      printf("%d 的数字分别为%d,%d\n",a,x2,x1);
      printf("%d 的逆序数字分别为%d,%d\n",a,x1,x2);
   }
  else
   {
      printf("%d 的数字为%d\n",a,x1);
      printf("%d 的逆序数字分别为%d\n",a,x1);
   }
  return 0;
}
```

说明：语句

```
if (x3!=0) d=3;
else  if (x2!=0) d=2;
else  d=1;
```

不能写成三个独立的 if 语句

```
if (x3!=0) d=3;
if (x2!=0) d=2;
if (x1!=0) d=1;
```

因为十位数不为 0 不能说明就是两位数，如 123。

例 4.16 按照考试成绩的等级输出百分制分数段，A 等 85 分以上，B 等为 70-84 分，C 等为 60-69 分，D 等为 60 分以下。

分析：已知等级 DJ

根据等级情况输出分数段。

这里用 c 语言中的开关语句来完成：根据等级情况输出分数段。

开关语句

格式：

```
switch(表达式)
{
   case 常量 1：语句 1
   case 常量 2：语句 2
    …
   case 常量 n：语句 n
   default :语句 n+1
}
```

功能：首先计算出表达式的值，如表达式的值和 case 后常量相等，就从该常量后的那个语句开始执行。如果表达式的值找不到相等的常量，就执行 default 后面的语句。

注意：

（1）表达式只能为可数类型（整数、字符）；

（2）switch 的所有情况语句一定有{}括起来；

（3）执行了一种情况后，不希望接着执行下一种情况，一定要用 break 退出 switch 语句。

（4）每种情况与后面的语句之间用“:”隔开。

于是得到该题的程序如下：

参考程序

```
#include<stdio.h>
int main()
{
    char dj;
    scanf("%c",&dj);
    switch (dj)
    {
      case 'A':printf("85 分以上");break;//break;退出 switch 语句
      case 'B':printf("70-84 分");break;
      case 'C':printf("60-69 分");break;
      case 'D':printf("60 分以下");break;
      default :printf("输入等级错误！");
    }
    return 0;
}
```

例 4.17 输入一个等级（A—G），输出是否过关。

如果输入 A—C，则输出“过关”，D—G，则输出“未过关”，其它字符，则输出“输入等级错误!”。(要求用 switch 完成)

参考程序

```
#include<stdio.h>
int main()
{
    char dj;
    scanf("%c",&dj);
    switch (dj)
    {
      case 'A':
      case 'B':
      case 'C':printf("过关");break; //A、B、C 等级都“过关”
      case 'D':
      case 'E':
      case 'F':
      case 'G':printf("不过关");break;
      default :printf("输入等级错误！");
    }
    return 0;
}
```

习　　题

4．1　水费（www.ssxdzx.net/bas　试题编号　1003）

【题目描述】FRACK 去美国做交换教师，在新居他拿到了新的收费标准。其中水费的计算让 FRACK 很恼火。

当地用水紧张，政府为了提倡节约用水，规定用水 20 立方米以下(含 20)每吨 2 美元，如超过 20 立方米，超过部分每一吨加收 1 美元资源浪费税。

于是他想请你帮帮忙，替他计算一下这个月要交多少水费。

【输入】一行，一个整数 n（1<=n<=5000）

【输出】一行，一个整数为要交的水费

【输入样例】50

【输出样例】130

4．2 输入 3 个字母，按字母表顺序从小到大输出这 3 个字母。

4．3 有一函数：$y=\begin{cases}-1.....(x<0)\\0.....(x=0)\\1.....(x>0)\end{cases}$　编程输入一个值，输出 y 值。

有的学生设计出如个程序，请分析是否能够得出正确结果。

(1)

```
#include<stdio.h>
int main()
{
    double x;
    int y;
    scanf("%lf",&x);
    y=-1;
    if (x!=0)
      if (x>0)
        y=1;
    else
      y=0;
    printf("%d\n",y);
    return 0;
}
```

(2)

```
#include<stdio.h>
int main()
{
    double x;
    int y;
    scanf("%lf",&x);
    y=0;
```

```
    if (x>=0)
      if (x>0)
        y=1;
    else
      y=-1;
    printf("%d\n",y);
    getchar();
    getchar();
    return 0;
}
```

4．4 有一个函数 $y=\begin{cases} x.....(x<1) \\ 2x-1.....(1\le x<10) \\ 3x-11.....(x\ge 10) \end{cases}$

写一个程序，输入 x 的值，输出 y 的值。

4．5 对一批货物征收税金。价格在 1 万元上以的货物征税 5%，在 5000 元~1 万（不含 1 万）元的货物征税 3%，在 1000 元~5000（不包含 5000）元的货物征税 2%，1000 元以下的货物免税。写一程序，读入货物价格，计算并输出税金。

4．6　输入一个整数 n，按下表输出。

n	输出内容	n	输出内容
1	今天是星期一	5	今天是星期五
2	今天是星期二	6	今天是星期六
3	今天是星期三	7	今天是星期天
4	今天是星期四	其它值	输入错误！

阿兰·麦席森·图灵

阿兰·麦席森·图灵 Alan Mathison Turing （1912.6.23-1954.6.7），生于英国伦敦，英国著名的数学家和逻辑学家，被称为计算机科学之父、人工智能之父，是计算机逻辑的奠基者，提出了“图灵机”和“图灵测试”等重要概念。人们为纪念其在计算机领域的卓越贡献而设立“图灵奖”。

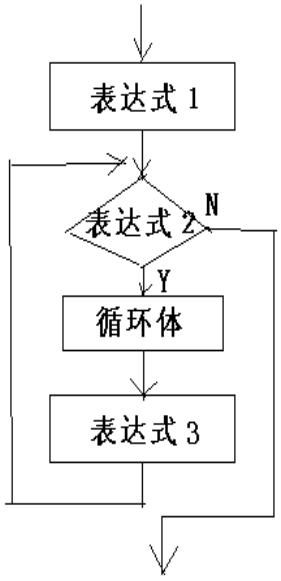
第 5 章　循环结构

例 5.1 输入一个整数 N，输出 1-N 之间所有数的平方。

分析：已知整数 N

输出 1^2，2^2，…,N^2

这里要用到循环语句来解决问题。



for 语句格式

for（表达式 1；表达式 2；表达式 3）

循环体;

其执行过程如右图。

（1）首先计算表达式 1;

（2)判断表达式 2 是否成立,若成立则执行循环体,若不成立则转（5)。

（4）求解表达式 3 转（2）

（5）循环结束

如下面的循环语句：

```
for (i=1;i<=4;i=i+1)
  printf("%d ",i*i);
```

执行过程：

首先执行 i=1;

循环次数	表达式 2	循环体	表达式 3	i 的值
	i<=4	printf("%d ",i*i);	i=i+1	变为
1	成立	输出 1	执行	2
2	成立	输出 4	执行	3
3	成立	输出 9	执行	4
4	成立	输出 16	执行	5
i<=4,条件不成立结束循环				

由此可知，这三个表达式的作用：

表达式 1 —初始化

表达式 2 —条件

表达式 3 —调整

参考程序

```
#include <stdio.h>
int main()
{
    int i,n;
    scanf("%d",&n);
    for (i=1;i<=n;i=i+1) //( )后面没有“;”
      printf("%d ",i*i);
    return 0;
```

```
}
```

说明：

（1）本题中用到了循环语句，一定要把循环执行的过程理解清楚。

（2）for (i=1;i<=n;i=i+1)

```
        printf("%d ",i*i);
```

“for (i=1;i<=n;i=i+1) ”最后没有“;”，即不能写成“for (i=1;i<=n;i=i+1);”，如果加了“;”，相当于循环体为一个空语句，而“printf("%d ",i*i);”不再是循环体了。

例 5.2 输入一个正整数 N，输出 1 + 2 + 3 +..+N 的值

分析：已知一个正整数 N，求 1 + 2 + 3 +..+N

可能有同学知道公式 N（1+N）/2，但我们这里不用此方法，而是使用计算机中常用的一种方法——“累加”。要明白“累加”是怎么回事，先看捐款过程。

我们为希望工程捐款的过程：首先，班长拿出一个捐款箱(空的)，第一个同学将捐款放入捐款箱内，第二个同学将捐款放入捐款箱内，…，最后捐款箱内就是同学们捐款的总数。

班长拿出一个捐款箱(空的)：s=0;

第一个同学将捐款数 X 放入捐款箱内，s=s+x

第二个同学将捐款数 X 放入捐款箱内，s=s+x

…,最后捐款箱内就是同学们捐款的总数。

用程序把这个过程表示出来：

```
s=0;
for (i=1;i<=n;i++)    //i++相当于 i=i+1
  s=s+x;              //每次 x 的值可能不一样。
```

参考程序

```
#include <stdio.h>
int main()
{
    int s,i,n;
    scanf("%d",&n);
    s=0; //赋初值 0,如果去掉这个语句，运行结果可能不正确!
    for (i=1;i<=n;i++)  //i++自增运算,这里的作用相当于 i=i+1
      s=s+i;    //循环体内出现 s=s+x 的形式
    printf("%d\n",s);
    return 0;
}
```

说明：

（1）关于累加和累乘

	累加	累乘
循环前	s=0	s=1
循环内	s=s+x	s=s*x

大家一定要理解的基础上去记忆累加和累乘的形式。

（2）s=0；这一语句必不可少。到目前为止，我们定义的变量都是定义在 int main()这个函数内部的。

在函数内部定义的变量称为局部变量，其默认的初值不一定是 0，所以 s=0;这句不可少。

在函数外部定义的变量称为全局部变量，其默认的初值一定是 0。如果在函数内部去掉定义变量 s，在函数外部进行定义，不赋初值 s=0;就没有问题，不妨试一下。

```
#include <stdio.h>
int s;  //定义的全局变量 s，默认初值为 0。
int main()
{
    int i,n;  //定义局部变量 i 和 n，这里没有定义 s
    scanf("%d",&n);
    for (i=1;i<=n;i++)
      s=s+i;
    printf("%d\n",s);
    return 0;
}
```

（3）自增运算符（++）和自减运算符（--）

作用是使用变量的值加 1 或减 1。假设 i 的初值为 1。

表达式	变量 i 的值	表达式的值
i++	2	1
++i	2	2
i--	0	1
--i	0	0

例 5.3 输入一个整数 N，输出 1 /1+1/ 2 +1/ 3 +..+1/N 的值

分析：把一串数加起来，显然是累加。

```
已知 n
s=0;
for (i=1;i<=n;i++)
  s=s+1.0/i;    //这里可以写成 s+=1.0/i
输出 s
```

参考程序

```
#include<stdio.h>
int main()
{
    int i,n;
    double s=0;  //注意变量 s 的类型
    scanf("%d",&n);
    for (i=1;i<=n;i++)
      s+= 1.0/i;    // 1.0/i 不能写成 1/i, 想想为什么?
    printf("%0.4lf\n",s);
    return 0;
}
```

输入 1000 输出 7.4855

说明：s+= 1.0/i;这里用到了复合赋值运算符“+=”，即在赋值号“=”前加一个其它运算符就构成了复合赋值运算符。

复合赋值运算符	表达式	等价表达式
+=	a+=b	a=a+b
-=	a-=b	a=a-b
*=	a*=b	a=a*b
/=	a/=b	a=a/b
%=	a%=b	a=a%b

此处 a 一定是变量，b 是一个表达式。如 a/=4+c 等价于 a=a/(4+c)，而不是 a=a/4+c。

例 5.4 输入一个整数 N，输出 1 /1-1/ 2 +1/ 3 -…±1/N 的值，

分析：和例 3 进行比较，相加的是一项为正，紧接着下一项为负。我们可以设一个变量 t 来表示正负。开始 t=1,表示为正，累加一项以后就将其变成相反数（t=-t），这样不断进行，t 的值依次为 1,-1,1,-1,…。

参考程序：

```
#include<stdio.h>
int main()
{
    int i,n,t=1; //代表第一项的正号
    double s=0;
    scanf("%d",&n);
    for (i=1;i<=n;i++)
     {
       s+=1.0/i*t;  //注意 t 的作用
       t=-t;        //将 t 变成下一项的正负
     }
    printf("%0.4lf\n",s);
    return 0;
}
```

输入 1000 输出 0.6926

说明：程序设计中解决正负相间，这个正负号的解决方法。用一个变量 t 来控制正负号。t=1 表示正负，t=-1 表示负号。正负相互转换用语句：t=-t;

例 5.5 用 $\frac{\pi}{4}$≈1-1/3+1/5-1/7+…公式求π 的近似值，直到某一项的绝对值小于 10^{-6}为止（该项不累加）。

分析：本题也是累加方法，求出 $\frac{\pi}{4}$ 的值，然后算出π 的值。

例 5.4 求 1 /1-1/ 2 +1/ 3 -…±1/N 的值。其程序的关键代码为:

```
    for (i=1;i<=n;i++)
     {
       s+=1.0/i*t;
       t=-t;
     }
```

本题只需将循环条件改为1.0/i>=1e-6（该项的绝对值>=10^{-6}），将i++改为i=i+2。

参考程序

```
#include<stdio.h>
int main()
{
  int i,t=1;
  double s=0;
  for (i=1;1.0/i>=1e-6;i=i+2) //注意表达式2和表达式3的变化
  {
    s+=t*1.0/i;
    t=-t;   //t的作用正负相间
  }
  s=4*s;  //得到π的值
  printf("%0.10lf",s);
  return 0;
}
```

输出 3.1415906536

说明

（1）1e-6这是c语言中的科学记数法，表示1×10^{-6}。一般地$m\times10^{n}$，用科学记数法表示为men。

（2）运行出来的结果为 3.1415906536，不太精确。这是因为计算的项数还不够多，如果把条件改为1.0/i>=1e-8，其结果为3.1415926336，则更精确但耗时。

例5.6 输入一个整数N，输出N!=1×2×3×…×N的值

分析问题：已知N，求1×2×3×…×N，显然是累乘。

参考程序

```
#include <stdio.h>
int main()
{
    int s,i,n;
    scanf("%d",&n);
    s=1;  //累乘赋初值1
    for (i=1;i<=n;i++)
      s*=i;  //循环体内出现s=s*x的形式
    printf("%d\n",s);
    return 0;
}
```

（1）输入 3 输出 6 （2）输入 10 输出 3628800

说明：

（1）N!就表示 1 × 2 × 3 ×…×N;

（2）N 的值不能太大。N 如果太大,N!会超过 int 数的范围（约-21 亿~21 亿）。超过范围以后，计算出的结果是错的。如输入 20 输出-2102132736。

例 5.7 寻找“水仙花数”。水仙花数是一组三位的正整数，它们满足条件：每一位上数字的立方之和恰好等于它自己。例如：153 就是一个水仙花数，1^3+5^3+3^3=153，请找出所有的水仙花数。

分析问题

我们可以尝试通过列举所有的三位正整数，如果是水仙花数，则输出，这种方法称为穷举法或枚举法。

于是得到思路框架如下

```
for (i=100;i<=999;i++)
  if    i 是水仙花数   输出 i;
```

关键问题怎么判断水仙花数

设三位数为 i

百位数 b=i /100

十位数 s=i % 100 /10

个位数 g= i % 10

参考程序

```
#include<stdio.h>
int main()
{
   int i,b,s,g;
   for (i=100;i<=999;i++)
    {
       b=i/100;
       s=i%100/10;
       g=i%10;
       if (i==b*b*b+s*s*s+g*g*g)  printf("%d ",i);
    }
  return 0;
}
```

输出 153 370 371 407

例 5.8 输入整数 N，表示接下来将会输入 N 个数，求出接下来的 N 个数的和。例如输入：3 5 4.5 1 输出 10.50

分析：我们先来看回顾例 3 求 1 + 2 + 3 +…+N 的值，程序的关键代码：

```
s=0;    //累加赋初值 0
for (i=1;i<=n;i++)
  s=s+i;
```

而所求的不是 1 + 2 +…+N，而是输入的 N 个数，循环体是不是应该：（1）输入一个数，（2）累

加一个呢？回答是肯定的。

也就是说循环体应该有两个部分：输入数 X，累加

循环体只能是一个语句，而这是两个语句，显然要做成一个语句--复合语句，才能一起做为循环体。

```
{
scanf(“%lf”,&x);
s=s+x;
}
```

参考程序：

```
#include <stdio.h>
int main()
{
    int i,n;
    double s,x;
    scanf("%d",&n);
    s=0;
    for (i=1;i<=n;i++)  //循环体是一个复合语句
    {
        scanf("%lf",&x);
        s=s+x;
     }
    printf("%lf\n",s);
    return 0;
}
```

输入 5 3.1 4.3 5.6 1.05 10.99 输出 25.040000

例 5.9 输入整数 N，表示接下来将会输入 N 个数，求出接下来的 N 个数的和、积和平均数。

分析：

本题就是把前面的累加和累乘结合在一起了。

参考程序：

```
#include <stdio.h>
int main()
{
    int i,n;
    double s,x,m;
    scanf("%d",&n);
    s=0;  //累加赋初值 0
    m=1;  //累乘赋初值 1
    for (i=1;i<=n;i++)
    {
      scanf("%lf",&x);
      s=s+x;
```

```
      m=m*x;
    }
    printf("%0.2lf  %0.2lf  %0.2lf\n",s,s/n,m);
    return 0;
}
```

例 5.10 输入整数 N，表示接下来将会输入 N 个数，求出接下来的 N 个数中的最大值

分析：已知整数 N 及 N 个数，求最大值

回想一下三个数存入变量 a,b,c 中找出最大值的算法：

(1)把第一变量 a 存入 max 中。

(2)if (max<b) max=b;

(3)if (max<c) max=c;

最后 max 中就是最大值。

于是得到本题算法：

（1）输入整数 N

（2）输入第一个数存入 max //认为第一个数最大

（3）for (i=1;i<=n-1;i++) //将剩下的 n-1 个数进行比较

{ 输入数 x;

X 与 max 进行比较，大数存入 max 中

}

（4）输出 max

参考程序：

```
#include <stdio.h>
int main()
{
  int n,i;
  double max,x;
  scanf("%d",&n);  //表示后面将会输入 n 个数
  scanf("%lf",&max);  //假设第 1 个数是最大的，存入 max 中
  for (i=1;i<=n-1;i++)  //将剩下的 n-1 个数 max 的值进行比较
   {
     scanf("%lf",&x);
     if (x>max) max=x; //当前的值比 max 中的值，进行调整
   }
  printf("max=%lf\n",max);
  return 0;
}
```

输入 4 3.2 9.5 4.3 6.7 输出 max=9.500000

思考（1）同时要求出最大值和最小值呢？

参考程序：

```
#include <stdio.h>
int main()
```

```
{
  int n,i;
  double max,x,min;
  scanf("%d",&n);
  scanf("%lf",&max);
  min=max;
  for (i=0;i<n-1;i++)
   {
     scanf("%lf",&x);
     if (x>max)  max=x;
     else  //此处可以不加 else 吗?
      if (x<min) min=x;
   }
  printf("max=%0.3lf\nmin=%0.3lf\n",max,min);
  return 0;
}
```

思考（2）输入整数 N，表示接下来将会输入 N 个数，求出接下来的 N 个数中的最大值与和；

参考程序：

```
#include <stdio.h>
int main()
{
  int n,i;
  double max,x,s;
  scanf("%d",&n);
  scanf("%lf",&max);
  s=max;     //第一个数放到和 S 中，初值不再是 0。
  for (i=1;i<=n-1;i++)
   {
     scanf("%lf",&x);
     if (x>max)
        max=x;
     s=s+x;
   }
  printf("max=%lf  和＝%lf\n",max,s);
  return 0;
}
```

例 5.11 按正序和反序输出 26 个英文小写字母。

分析：从'a'~'z'顺序依次输出字母；

换行；

从'z'~'a'顺序依次输出字母；

本题的关键是怎么输出字母：我们的想法：

让变量 i 取'a',输出 i,再让变量 i 取'b'，再输出 i,直到输出'z';为止。

```
  for (i='a';i<='z';i++)
       printf("%c",i);
```

循环变量可以取字符吗？回答是肯定的。

然后，再同样的方法输出'z'~'a'。

参考程序：

```
#include <stdio.h>
int main()
{
  char i;
  for (i='a';i<='z';i++)  //输出'a'~'z'
     printf("%c",i);
  printf("\n");       //换行
  for (i='z';i>='a';i--)  //输出'z'~'a'
     printf("%c",i);
  return 0;
}
```

输出

```
abcdefghijklmnopqrstuvwxyz
zyxwvutsrqponmlkjihgfedcba
```

说明：循环变量不仅可以是整数、字符，也可以是实数。

例 5.12 Fibonacci 数列

1202 年，意大利数学家斐波那契出版了《算盘全书》。他在书中提出了一个关于兔子繁殖的问题：

如果一对大兔子每月能生一对小兔（一雄一雌），而每对小兔在出生后的第二个月里能长成大兔，第三个月又能开始生一对小兔，假定在不发生死亡的情况下，第一个月有 1 对出生的小兔，问第 50 个月后会有多少对兔子？

在第 1 个月时，只有 1 对小兔子；第 2 个月，长成大兔，也还是 1 对兔子；第 3 个月时便生下 1 对小兔子，这时有 2 对兔子；第 4 个月时，新生下的小兔 1 对+第 3 个月的兔子 2 对，即为 3 对。如此推算下去，我们便发现一个规律：

时间(月)	兔子总数(对)	说明
1	1	
2	1	
3	2	1+1=2
4	3	1+2=3
5	5	2+3=5
6	8	3+5=8
7	13	5+8=13
…	…	…

由此可知，从第一个月开始以后每个月的兔子总数是:1,1,2,3,5,8,13,21,34,55,89,144,233…

若把上述数列继续写下去，得到的数列便称为斐波那契数列。数列中每个数便是前两个数之和，而数列的最初两个数都是 1。

设第 n 个月的兔子对数为 a_n，则有 $a_n=a_{n-1}$+第 n 个月新生下的兔子数

而第 n 个月新生下的兔子数是多少呢？这就要知道第 n-1 个月有多少对大兔，而第 n-1 个月的大兔数就是第 n-2 个月的兔子，因为第 n-2 个月的兔子不管是大兔还是小兔，到第 n-1 个月都会是大兔。于是我们得到：

$a_1=1$

$a_2=1$

$a_n=a_{n-2}+a_{n-1}$ (n>2)

任务：求 fibonacci 数列 $a_0,a_1,a_2,\ldots,a_{20}$。

分析一：从上面可以看出该数列是第 n 项是靠近前两项之和。为了设计程序方便，我们用 a_0,a_1,a_2 表示这三项,则有 $a_2= a_0+ a_1$，为了下次计算方便，计算一次后，我们可以把 $a_0=a_1$; $a_1= a_2$,则又可以用 $a_2= a_0+ a_1$ 计算下一项

a_0	a_1	$a_2=a_0+a_1$
0	1	1
1	1	2
1	2	3
2	3	5

….

参考程序一

```
#include <stdio.h>
int main()
{
  int a0,a1,a2,i;
  a0=0;a1=1;
  printf("%d %d",a0,a1);    //输出 0、1 两项
  for (i=2;i<=20;i++)     //求出 2-20 项
    {
      a2=a0+a1;    //计算第 i 项
      printf(" %d",a2); //输出第 i 项
      a0=a1;
      a1=a2;    //以上两个语句是作用是平移，便于计算下一项
    }
  return 0;
}
```

分析二

$a_0=0$	$a_1=1$
a_0 (=a_0+a_1)	a_1 (=a_0+a_1)
1	2
3	5
8	13

…

于是得到了本题的算法：

参考程序二

```
#include <stdio.h>
int main()
{
  int a0,a1,i;
  a0=0;a1=1;
  printf("%d %d",a0,a1); //输出 0、1 两项
  for (i=1;i<=9;i++)  //求出 2-19 项，每次计算两项
    {
      a0=a0+a1;
      a1=a0+a1;
      printf(" %d %d",a0,a1); 输出两项
    }
  printf(" %d\n",a0+a1);  //输出第 20 项
  return 0;
}
```

说明：如果只要出现..., 1,1,2,3,5,8,...就要看是否是 **fibonacci 数列。**

例 5.13 输入 n 及 n 个学生的成绩（0-100），分别统计成绩在 85-100、60-85（不包括 85）、60 分以下，各分数段中的人数。

分析：

首先，输入人数 n.

设 num1、num2、num3 分别表示在 85-100、60-85（不包括 85）、60 分以下的人数，每个变量置 0。

读一个成绩 score，就判断 score 值在哪一个分数段，相应的变量增加 1。重复 n 次。

于是得到程序框架

```
For (i=1;i<=n;i++)
{
  输入一个成绩 score;
  判断 score 值在哪一个分数段，相应的变量增加 1;
}
```

最后输出 num1、num2、num3。

参考程序：

```
#include <stdio.h>
int main()
{
  int n,i,num1,num2,num3;
  double score;
  scanf("%d",&n);
  num1=0;num2=0;num3=0;
```

```
  for (i=1;i<=n;i++)
    {
     scanf("%lf",&score);
     if (score>=85)
         num1=num1+1;
     else
        if (score>=60)
          num2=num2+1;
        else
          num3=num3+1;
    }
  printf("100-85:%d   85-60:%d  <60:%d\n",num1,num2,num3);
  return 0;
}
```

输入 6 56 89 70 89 75 90 输出 100-85:3 85-60:2 <60:1

说明：本题是分类进行统计：循环之前 num1=0；在循环内 num1=num1+1；称为计数语句。实际上是特殊的累加，每一次累加 1。

例 5.14 输入一个正整数 N，如果是素数，则输出“YES!”，否则输出“NO!”

分析：素数即质数。只有因子 1 和它本身的数。

我们可用 i=2~n-1 去检验 i 是否 n 的因子，如果 i 是 n 的因子，则已判断出不是素数，不再去检验 i+1 了。如 n=7

i	2	3	4	5	6	7	2~6 都不是 7 的因子，所以 7 是素数
i 是否 n 的因子	否	否	否	否	否	是	

如 n=9

i	2	3	由于 3 是 9 的因子，9 不是素数，也没有必要去检验 4 ~ 8 是否 9 的因子了。
i 是否 n 的因子	否	是	

当检验结束，我们可以通过 i 的值去判断 n 是否素数。

参考程序一

```
#include <stdio.h>
int main()
{
  int n,i;
  scanf("%d",&n);
  for (i=2;i<=n-1;i++)
    if (n%i==0) break;   //有因子，退出循环
  if (i==n) printf("YES!\n");
  else printf("NO!\n");
  return 0;
}
```

运行输入 2000000011，但上面的程序要 10 多秒才能出结果，下面介绍一种快速的方法来判断。

定理：对于整数 N（>2），除了 1 和 N 本身外，如果没有小于等于$\sqrt{N}$ 的因子，就没有大于$\sqrt{N}$ 的因子。

证明：设 X 是 N 的因子且 X>$\sqrt{N}$ ，则必然存在一个 Y，使得 N=XY,而 X>$\sqrt{N}$,则有 Y<$\sqrt{N}$ 。所以只要有大于$\sqrt{N}$ 的因子，就必有小于$\sqrt{N}$ 的因子。

有了上面的定理，我们就可以只判断 2~$\sqrt{N}$ 中，是否有 N 的因子。如果没有就是素数。

于是有

```
for (i=2;i<=√N ;i++)
  If  (n%i==0) break;
```

条件 i<=$\sqrt{N}$ ，实际上就是 i*i<=N。

参考程序二

```
#include <stdio.h>
int main()
{
  int n,i;
  scanf("%d",&n);
  for (i=2;i*i<=n;i++)
    if (n%i==0) break;  //有因子，退出循环。
  if (i*i>n) printf("YES!\n");
  else printf("NO!\n");
  return 0;
}
```

（1）输入 2000000011 输出 YES! （2）输入 8 输出 NO!

说明：

（1）**判断一个数 N 是否素数，只需判断 2~$\sqrt{N}$ 中是否有 N 的因子**。上面的参考程序二判断素数 2000000011,就瞬间出结果了。这是因为 for (i=2;i*i<=2000000011;i++)循环的次数不会超过$\sqrt{2000000011}$ (<50000)次。

（2）break 和 continue 改变循环执行的状态

break 有两个作用

1）退出 switch 语句；

2）提前结束整个循环，不再判断循环的条件是否成立。

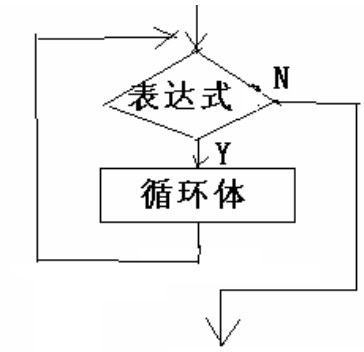
continue 的作用：提前结束本次循环，接着执行下次循环。

请看下面的程序

```
#include <stdio.h>
int main()
{
  int i;
  for (i=1;i<=20;i++)
  {
    if (i%3==0) continue; //i 是 3 的倍数，忽略循环体后半部分 printf("%d ",i);
    printf("%d ",i);
  }
  return 0;
```

}

输出 1 2 4 5 7 8 10 11 13 14 16 17 19 20



前面熟悉了 for 循环，下面再介绍两种循环语句。

while 语句

格式：

while (表达式)

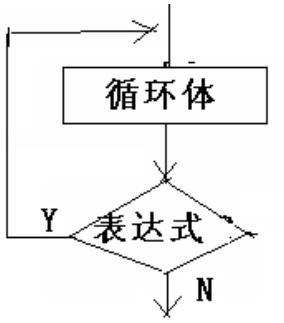
　循环体;

执行过程（如右图）：

（1）计算表达式

（2）当表达式成立，执行循环体，转（1）；表达式不成立转（3）

（3）结束循环



do … while 语句

格式：

do

　循环体;

while (表达式)

执行过程（如右图）：

（1）执行循环体

（2）当表达式成立，转（1）；表达式不成立转（3）

（3）结束循环

While 语句和 do…while 语句区别：

(1)while 语句先判断条件，条件成立才执行循环体，而 do …while 语句执行了一次循环体后再判断条件；

(2)while 语句可能一次循环也不执行，而 do …while 语句至少执行一次循环；

(3)while 语句又叫当型循环，do …while 语句叫直到型循环。

while 语句	do … while 语句

i=0;s=0;
while (i<0)
{
i++; s=s+i;
}
结果 s=0
i=0,先判断条件 i<0,不成立，结束循环。
一次循环也没执行。
	i=0;s=0;
	do
	{
	i++; s=s+i;
	}
	while (i<0);
	结果 s=1
	先执行循环，则 i=1,s=1 了，再判断条件 i<0，条件不成立，则结束循环。
	只执行了一次循环。

例 5.15 求 $s=1+\frac{1}{2}+...+\frac{1}{n}$,当 s>10 的最小 n 值。

分析：本题也是一个累加。循环的条件是 s<=10。

参考程序一(for 语句)：

```
#include<stdio.h>
int main()
{
    double n,s=0;
    for (n=1;s<=10.0;n++)
      s=s+1.0/n;
    printf("%0.0lf\n",n-1); //想一想，为什么要输出 n-1?
    return 0;
}
```

参考程序二（while 语句)：

```
#include <stdio.h>
int main()
{
    int n;
    double s;
    s=0;
    n=1;
    while (s<=10)
      {
        s=s+1.0/n;
        n++;
      }
    n--;  //想一想，为什么要 n--?
    printf("%d\n",n);
    return 0;
}
```

参考程序三（do …while 语句)：

```
#include<stdio.h>
int main()
{
  double s,n;
  s=0;n=0;
  do
  {
    n++;
    s=s+1/n;
  }
  while (s<=10);
  printf("%0.0lf\n",n); //想一想，为什么要输出 n?
  return 0;
```

```
}
```

输出 12367

说明：

（1）三个程序中提出了“三个为什么”，要回答这个问题，必须深刻的理解这循环的执行过程。

一般来说，一个程序对一串数进行操作（这里是累加），我们要看程序对这一串数两头（第 1 个和最后 1 个）操作是否正确，中间的一般不易出错。

（2）三种循环 for 语句、while 语句、do...while 语句，可以相互替代。

例 5.16 输入两个正整数 m 和 n,输出它们的最大公约数和最小公倍数。

分析：已知 m 和 n,求最大公约和最小公倍数。

首先考虑最大公约数。

最朴素的方法：

min 取 m 和 n 小的数，从 min 向下一个一个开始检查是否同时是 m 和 n 的因子，如果是就找到了最大公约数。

于是得到了

```
for (i=min;i>=1;i--)  //向下找到的第 1 个公约数就是最大公约数
  if (i%n==0&&i%m==0) break;
```

这时 i 就是最大公约数。

最小公倍数就是 m*n/i

参考程序

```
#include<stdio.h>
int main()
{
  int m,n,min,i,x;
  scanf("%d%d",&m,&n);
  if (m>n) min=n;//可以不考虑 m、n 的大小
  else min=m;
  for (i=min;i>=1;i--) //向下找到的第 1 个公约数就是最大公约数
    if (m%i==0&&n%i==0) break;
  x=m/i*n;
  printf("gcd=%d  lcm=%d",i,x);
  return 0;
}
```

输入 16 24 输出 gcd=8 lcm=48

说明：

（1）参考程序中是从两个数中小数向下检查。但实际上不一定从小数向下检查，任选一个数向下检查也是可以的。如果选到大数不会影响结果。因为大数到小数这一段数中不可能是公约数。

所以，参考程序关键代码可改写为：

```
  for (i=n;i>=1;i--)
    if (m%i==0&&n%i==0) break;
```

（2）求最小公倍数写成的是 m/i*n，而一般不写成 m*n /i。如果 m 和 n 的最小公倍数 21 亿，m*n 的结果可能很大，超过整数的范围。一般都要避免中间结果过大。所以，写成 m/i*n。

（3）求最大公约数的时间复杂度为O(n)，当n达到2.1×10^9时，1秒内是运行不出结果的。那我们得另寻他法——辗转相除法求最大公约数。

例5.17 利用辗转相除法求正整数m和正整数n的最大公约数。

辗转相除法

（1）求出m除以n的余数r;

（2）如果r=0，则最大公约数是n,转（4），r≠0转（3）

（3）m=n,n=r 转（1）

（4）结束

设m=16,n=24,求最大公约的过程如下：

m	n	r=m%n		
16	24	16	r≠0	执行m=n,n=r
24	16	8	r≠0	执行m=n,n=r
16	8	0	r=0	最大公约数就是n，结束

为什么这样求出来的就是最大公约数呢？

我们只要能证明以下两个问题

（1）m和n的公约数与n和r的公约数是一样的。

∵r=m%n ∴可设m=pn+r

一方面，设m和n的公约数为x,则有m%x=0,n%x=0

∴（pn+r）%x=0 又∵pn%x=0

∴r%x=0

∴x是n和r的公约数。

另一方面，设n和r的公约数为x,则有n%x=0,r%x=0

∴pn%x=0

∴(pn+r)%x=0

∴m%x=0

∴x是m和n的公约数。

（2）一个正整数n和0的最大公约数就是这个正整数n。

∵n%n=0, 0%n=0

又∵n没有比n大的因子。

∴n和0的最大公约数为n。

至此，我们说明了命题的正确性。

分析：已知m和n,求最大公约数。

```
r=m%n
while (r!=0)
 {
   m=n;n=r;
   r=m%n;
 }
```

参考程序

```
#include<stdio.h>
int main()
```

```
{
  int m,n,r;
  scanf("%d%d",&m,&n);
  r=m%n;
  while (r!=0)
  {
    m=n;n=r;
    r=m%n;
  }
  printf("gcd=%d",n);
  return 0;
}
```

左边代码可替换为以下代码

```
for (r=m%n;r!=0;r=m%n)
   {
     m=n;
     n=r;
   }
```

输入 2000000011 2000000012 输出 gcd=1

说明：

（1）辗转相除法求最大公约数的方法，一定要牢固掌握。在竞赛中对时间复杂度要求较高。

（2）如果要求最小公倍数，用辗转相除法求最大公约数之前，要把 m 和 n 的值保存起来，因为在求解的过程中 m 和 n 的值都在改变。

例 5.18 输入一行字符，输出其中字母个数、数字个数和其它字符个数。

分析一：已知一行字符，就是以换行结束'\n'。

算法过程：

用 n1,n2,n3 来统计三类字符的个数，它们赋初值 0

```
读一个字符 x
while  x 不是'\n';
   {
      x 是哪类字符，相应的变量+1;
      读一个字符到变量 x;
   }
 输出 n1,n2,n3
```

参考程序一：

```
#include <stdio.h>
int main()
{
  int n1,n2,n3;
  char x;
  n1=0;n2=0;n3=0;
  scanf("%c",&x);
  while (x!='\n')
   {
     if (x>='a'&&x<='z'||x>='A'&&x<='Z') n1++;
     else
       if (x>='0'&&x<='9') n2++;
```

```
        else n3++;
      scanf("%c",&x);
    }
  printf("letter:%d  number:%d   other:%d",n1,n2,n3);
  return 0;
}
```

输入 abc@#$@#12AABB? 输出 letter:7 number:2   other:6

分析二：设 n1、n2、n3 分别表示字母、数字、其它字符的个数，每个变量置 0。

首选，读一个字符 x

```
for (;x!= '\n';)
{
```

判断 x 是哪类字符，相应的变量增加 1;

读入下一个字符;

```
}
```

最后输出 n1、n2、n3。

参考程序二

```
#include<stdio.h>
int main()
{
    char x;
    int n1=0,n2=0,n3=0;
    scanf("%c",&x);  //先读一个字符, for 循环条件判断 c!='\n'。
    for (;x!='\n';)      //for 语句中省略了表达式 1 和表达式 3
    {  //判断 x 是哪类字符，相应的变量增加 1;
      if (x>='a'&&x<='z'||x>='A'&&x<='Z') n1++;
      else
      if (x>='0'&&x<='9')n2++;
      else n3++;
      scanf("%c",&x); //读下一个字符
    }
    printf("letter:%d  number:%d   other:%d",n1,n2,n3);
    return 0;
}
```

输入 abc@#$@#12AABB? 输出 letter:7 number:2   other:6

说明：

（1）for（表达式 1；表达式 2；表达式 3）

循环体；

for 语句中表达式 1、表达式 2、表达式 3 都可以省略，但相隔的“;”不能省略。省略的表达式是一个空表达式。

（2）本题参考程序：循环之前有一个语句“scanf("%c",&c); ”，循环体内也有一个“scanf("%c",&c)”;语句。这两个语句一定要思考清楚它们的作用，有利于培养严谨的程序思维。

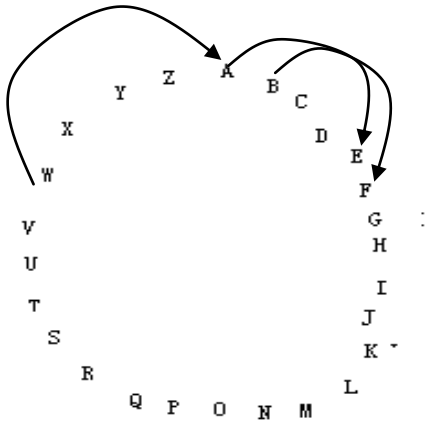
例 5.19 译密码

为使电文保密，往往按一定规律将其转换成密码，收报人再按约定的规律将其译回原文。例如，按下规律将电文变成密码。

'a'→'e', 'b'→'f',…., 'u'→'z', 'w'→'a', 'x'→'b', 'y'→'c', 'z'→'d', 'A'→'E', 'B'→'F',…., 'U'→'Z', 'W'→'A', 'X'→'B', 'Y'→'C', 'Z'→'D'。

输入一行字符,如果是字母,将字母按以下规则变换,其它字符不变。

分析：上述的规律：将大写英文字母排成一圈，变成其后的第 4 个字母。小写英文字母规律一样。



于得到处理思路：

```
读入一个字符 a
While (a 不是回车)
{
   if  a 是字母 转换成相应的字母 a;
  输出字符 a
  输入 a
}
```

参考程序一

```
#include<stdio.h>
int main()
{
    char a;
    scanf("%c",&a);
    while (a!='\n')
    {  //'a'~'v'前进 4 个字符
      if (a>='a'&&a<='v'||a>='A'&&a<='V')
      {
        a=a+4;
      }
      //'w'~'z'实际上是后退 22 个字符
      if (a>='w'&&a<='z'||a>='W'&&a<='Z')
      {
       a=a-22;
      }
      printf("%c",a);
      scanf("%c",&a);
    }
  return 0;
}
```

参考程序二:

```
#include<stdio.h>
int main()
```

```
{
    char a;
    scanf("%c",&a);
    while (a!='\n')
    {  //处理 'a'~'z'
      if (a>='a'&&a<='z')
      {
        a=a+4;
        if (a>'z') a=a-26;  //超过'z',后退 26 个字符
      }
      //处理 'A'~'Z'
     if (a>='A'&&a<='Z')
      {
        a=a+4;
        if (a>'Z') a=a-26; //超过'Z',后退 26 个字符
      }
      printf("%c",a);
      scanf("%c",&a);
    }
  return 0;
}
```

参考程序三:

```
#include<stdio.h>
int main()
{
  char c;
  scanf("%c",&c);
  while (c!='\n')
  {
    if (c>='A'&&c<='Z'||c>='a'&&c<='z')  //处理大小写字母
      {
         c=c+4;
         if (c>'Z'&&c<='Z'+4||c>'z') //大写字母后面有小写字母
           c=c-26;
      }
    printf("%c",c);
    scanf("%c",&c);
  }
  return 0;
}
```

输入 31abzw&%cABYZ? 输出 31efda&%gEFCD?

说明：三个参考程序中对密码规则处理方法不一样，参看程序中的注释。要学会对于同一个问

题多角度的思考来锻炼自己的大脑。

例 5.20 验证哥德巴赫猜想：任一充分大的偶数 n（>=4），可以用两个素数 p、q 之和表示。例 4＝2+2　6＝3+3　8＝3+5　…　98＝19+97

哥德巴赫猜想是一个古老而著名的数学难题。它的理论证明很麻烦，迄今未得到证明。在这方面中国数学家陈景润的研究成果处理世界领先地位。

陈景润之前的数学家，只证明出：对任一充分大的偶数 n，可以找到 4 个素数，p、q、r、s。使得 n=p+q.r.s，这就是 1+3 问题。

陈景润证明出：对任一充分大的偶数 n，可以找到 3 个素数，p、q、r。使得 n=p+q.r，这就是 1+2 问题。

而真正的哥德巴赫猜想是：对任一充分大的偶数 n，可以找到 2 个素数 p、q，使得 n=p+q，这就是 1+1 问题。

分析：已知偶数 n，求素数 p 和 q,有 n=p+q。（假设 p<=q）

枚举 p=2 开始去找 p 和 q，于是有

```
for (p=2;p<=n/2;p++)
{
   q=n-p;
   if p 和 q 都是素数 找到 break;
}
```

判断素数 P 是否素数：枚举 2~$\sqrt{p}$ 中是否有因子。

参考程序

```
#include<stdio.h>
int main()
{
    int n,p,q,i,j;
    scanf("%d",&n);
    for (p=2;p<=n/2;p++)
    {
      q=n-p;
      for(i=2;i*i<=p;i++) //i 检验 p 是否素数
        if (p%i==0) break;
      for(j=2;j*j<=q;j++)//j 检验 q 是否素数
        if (q%j==0) break;
      if (i*i>p&&j*j>q) break; //p 和 q 都是素数 break
    }
    printf("%d=%d+%d",n,p,q);
    return 0;
}
```

输入 100000 输出 100000=11+99989

说明：

（1）for (p=2;p<=n/2;p++)可以省略表达式 2 写成 for (p=2; ;p++)，因为给定一个偶数，素数 p 和 q 是存在的。

（2）要输出所有的 n=p+q（p 和 q 是素数）的情况，程序应该怎么改呢？

例 5.21 输出如下的图形，要求：共 n 行，n 从键盘输入。

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

分析：

输入 n; scanf("%d",&n);

```
for (i=1;i<=n;i++)
  {
    输出 i 行字符;
    换行; printf("\n");
  }
```

怎么输出第 i 行字符呢？（假设图案位于屏幕中间，每行中间约是 40 列）。

先输出空格再输出“*”号，那么 i 行前面空格输出多少个，“*”又有多少个呢？

i 行	空格个数	“*”个数
1	39	1
2	38	3
3	37	5
…	…	…
i	40-i	2*i-1

下面，我们来说明怎么得到第 i 行的空格数是 40-i,*数是 2*i-1。

通过上面的列表可以看出：

（1）行增加 1（i 增加 1），空格数减少 1，于是可设 i 行的空格数为 x-i。而 i=1 时，空格数为 39，有 x-1=39,x=40 所以空格数为 40-i。

（2）行增加 1（i 增加 1），*数增加 2，于是可设 i 行的*数为 x+2i。而 i=1 时，*数为 1 ，有 x+2=1,x=-1 所以*数为 2i-1。

于是，我们得到输出图案的一般模式：

```
for (i=1;i<=行数;i++)  //共输出多少行
  {
      for (j=1;j<=i 行空格个数; j++)  //输出 i 行前面空格
         printf(" ");
      for (j=1;j<=i 行符号个数; j++)  //输出 i 行的*
         printf("* ");
      printf("\n");                  //换行
  }
```

参考程序：

```
#include <stdio.h>
int main()
{
  int i,j,n;
  scanf("%d",&n);
  for (i=1;i<=n;i++)
  {
      for (j=1;j<=40-i;j++)
         printf(" ");
      for (j=1;j<=2*i-1;j++)
        printf("*");
      printf("\n");
  }
  return 0;
}
```

在上题输出的图形基础上，再增加 n-1 行，构成如下的图形：

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

可以认为下半图形也是 n 行（将上面的 n--即可）。有了输出图案的模式，按照模式写就行了。关键是要确定求出空格个数和*个数。

通过观察，每增加 1 行，空格增加 1 个，可设 i 行的空格数为 x+i，当 i=n 时，空格数为 39，于是 39=x+n，x=39-n，所以空格数为 39-n+i；

每增加 1 行，*数减少 2 个，可设 i 行的*数为 x-2i，当 i=n 时，*为 1 个，于是 1=x-2n，x=2n+1，所以*数为 2n+1-2i。

参考程序如下

```
#include <stdio.h>
int main()
{
  int n,i,j;
  scanf("%d",&n);
  for (i=1;i<=n;i++)
  {
```

```
        for (j=1;j<=40-i;j++)
           printf(" ");
        for (j=1;j<=2*i-1;j++)
          printf("*");
        printf("\n");
    }
    n--; //下半图形少了1行
    for (i=1;i<=n;i++)
    {
        for (j=1;j<=39-n+i;j++)
           printf(" ");
        for (j=1;j<=2*n+1-2*i;j++)
          printf("*");
        printf("\n");
    }
    return 0;
}
```

说明：掌握打印图案的一般模式，懂得计算空格个数和符号（*）个数，编写这一类打印图案的程序就容易多了。

例5.22 数字统计（noip2010普及组第一题）

【问题描述】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】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】输出共1 行，表示数字2 出现的次数。

【输入输出样例1】输入 2 22　　输出 6

【输入输出样例2】输入 2 100　　输出 20

【数据范围】1 ≤ L ≤ R≤ 10000。

分析：已知L和R，统计[L，R]区内所有整数中数字2出现的次数。

枚举L—R中的每一个数，分别统计每个数中2的个数。于是得到

```
for (i=L;i<=R;i++)
 {
   将i中2的个数加入到答案ans中。
 }
```

怎么求出数i中2的个数呢？这里有两种方法。

（1）i最多为5位数,可以把i当成5位数，但万位上不可能出现2，所以分离出个位ge、十位shi、百位bai、千位qian,有几个数是2，答案中就增加几。于是得到参考程序一：

参考程序一

```
#include<stdio.h>
int main()
{
```

```
    int L,R,ge,shi,bai,qian,wan,i,ans=0;
    scanf("%d%d",&L,&R);
    for(i=L;i<=R;i++)
    {
       ge=i%10;
       shi=i/10%10;
       bai=i/100%10;
       qian=i/1000%10;
       if(ge==2) ans++;
       if(shi==2) ans++;
       if(bai==2) ans++;
       if(qian==2) ans++;
    }
    printf("%d\n",ans);
    return 0;
}
```

（2）可以把i保存到一个变量n中，每次将n个位g取出，如果是2，答案ans中加1，然后n=n/10,就把n变小为原来的十分之一（只取整数），如果n!=0,说明还有位数没有取到，继续。例如：n=123

g=n%10=3 n=n/10=12

g=n%10=2 n=n/10=1

g=n%10=1 n=n/10=0 不再进行了。

于是得到参考程序二。

参考程序二

```
#include<stdio.h>
int main()
{
    int L,R,ans=0,i,n;
    scanf("%d%d",&L,&R);
    for(i=L;i<=R;i++)
    {
       n=i;
       while (n!=0)
        {
          if(n%10==2) ans++;
          n=n/10;
        }
   }
  printf("%d",ans);
  return 0;
}
```

说明：可以在www.rqnoj.cn 上提交评测（试题编号为602）。

习　题

5.1 计算并输出$1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+..+\frac{1}{9999}-\frac{1}{10000}$的值

5.2 利用下列公式计算并输出π的近似值　$\frac{\pi}{4}\approx 1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-...$

前1000000项的和，结果保留8位小数。

5.3 输出1-999中能被3整把除，且至少有一位数字是5的所有整数。

5.4 求 2-1000 中的守形数（若某数的平方，其低位与该数本身相同，则称该数为守形数。如25，25^2＝625，625的低位与原数相同，则称25为守形数。）

5.5 一个数如果恰好等于它的真因子（不包括它本身）之和，这个数称为完数。例如， 6的因子为1， 2， 3，而6 ＝ 1＋2＋3，因此6是完数。编程找出1000之内的所有完数。

5.6 求数列 $a_0, a_1, a_2, \cdots, a_{30}$。

$a_0=0, a_1=1, a_2=1$

$a_3=a_0+2a_1+a_2$

$a_4=a_1+2a_2+a_3$

…

5.7 有一分数序列

$\frac{2}{1}, \frac{3}{2}, \frac{5}{3}, \frac{8}{5}, \frac{13}{8}, \frac{21}{13}, \cdots$

输入一个正整数n，输出这一序列的前n项之和。

5.8 立方数的分解（www.ssxdzx.net/bas，试题题号1011）

【题目描述】任何一个正整数的立方都可以写成一串奇数之和,这就是著名的尼科梅彻斯定理。

1^3=1、2^3=3+5=8、3^3=7+9+11=27、4^3=13+15+17+19=64。

给出n,求n^3是哪n个连续奇数之和?

【输入】一行,n(1<=n<=100)

【输出】连续奇数相加等于立方的算式。

例如当n=1,输出:1=1 ； 当n=3,输出:7+9+11=27

【输入样例】3

【输出样例】7+9+11=27

5.9 求 $s_n=a+aa+aaa+...+\overbrace{aa...a}^{n个a}$ 之值，其中a是一个数字，n表示a的个数。n(1<=n<=9)和a(1<=a<=9)由键盘输入。

5.10 利用下列公式计算并输出π的近似值　$\frac{\pi}{4}\approx 1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-...$

直到某一项的绝对值小于10^{-6}为止，结果保留8位小数。

5.11 输入整数n及n个字符，统计英文字母、数字和其它字符的个数。

5.12 输入一个正整数n,输出1~n内的所有素数。

5.13 求2－100中，每个数的质因子，输出如下形式：

2＝2

3＝3

4＝2*2

…

100=2*2*5*5

提示：质因子即素数因子。在找质因子的过程中，可以不必判断它是否为素数，只要算法合适，所求的因子必然是质因子。

5.14 求2－10000中的亲密数对（如果a的真因子和等于b,b的真因子和等于a,且a≠b，称a,b为亲密数对）。

5.15 哥德巴赫猜想：对任一充分大的偶数n，可以找到2个素数p、q，使得n=p+q。请输出所有可能的p和q，使得n=p+q。

5.16 要将一张100元的大钞票，换成等值的10元、5元、2元、1元的小钞票。要求每次换成40张小钞票，每种至少一张。编程输出所有可能的换法。

5.17 编程输出下列图形

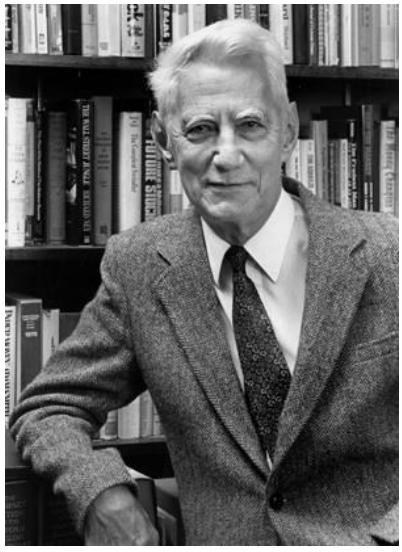
```
                a
              a   b
            a   b   c
          a   b   c   d
        .               .
      .                   .
    .                       .
  z   b   c   ...       y   z
```

注意分析：每一行上相邻两这字母间的空格数。

5.18 打印数字三角形

```
                1
              1 2 1
            1 2 3 2 1
          1 2 3 4 3 2 1
        1 2 3 4 5 4 3 2 1
      . . . . . . . . . .
    . . . . . . . . . . . .
1 2 3 . . . 8 9 8 . . .3 2 1
```

克劳德·香农



美国数学家、信息论的创始人。

克劳德·香农(Claude Elwood Shannon，1916-2001)，1916年4月30日出生于美国密歇根州的加洛德(Petoskey)，1936年毕业于密歇根大学并获得数学和电子工程学士学位，1940年获得麻省理工学院（MIT）数学博士学位和电子工程硕士学位。1941年他加入贝尔实验室数学部，工作到1972年。1956年他成为麻省理工学院（MIT）客座教授，并于1958年成为终生教授，1978年成为名誉教授。香农博士于2001年2月26日去世，享年84岁。

第6章 数 组

数组，顾名思义就是一组同类型的数。将数组和循环结合起来，可以有效地处理大批量的数据，大大提高了工作效率，十分方便。

例6.1 输入10个整数，然后反序输出。

分析：已知10个整数，反序输出。

根据前面的知识，这里我们要定义10个整型变量，保存输入的10个整数，然后反序输出。这10个变量可以定义为x1,…x10,要是10000个数呢？前面使用变量的方法就望尘莫及了。这里我们就要使用数组来解决问题了。

一维数组的声明：类型符 数组名[整型常量表达式]

例如 int a [10];

说明：

（1）类型标识符是int 表明定义的数组中只能储存一批整数；

（2）int a[10] 定义了一个有10个整型元素数组a,这10个元素变量分别是a[0],a[1],…,a[[9]（从0开即编号），不能使用a[10];double b[1000]定义了一个有1000个实型变量的数组b;char c[10000]定义了一个有10000个字符变量的数组c。

（3）使用这些变量只需要下标改变就对应不同的变量。如a[i],i=1表示变量a[1],i=5表示变量a[5]。

（4）int a[10]定义的这10整数变量在内存中是连续存储的，每个整数占4个字节，所以连续区域的大小是4×10＝40个字节。

（5）声明数组时，[ ]内是整型常量表达式，该表达式的值只能是整数并且只能是常量，也就是该表达式内不能含有变量。

于是得到本题的算法：

输入10个数存于数组a中

反序输出这10个数。

参考程序

```
#include<stdio.h>
int main()
{
int i,a[10];
for(i=0;i<=9;i++) scanf("%d",&a[i]);
for(i=9;i>=0;i--) printf("%d ",a[i]);
return 0;
}
```

输入：1 2 3 4 5 6 7 8 9 10

输出：10 9 8 7 6 5 4 3 2 1

例6.2 对一维数组初始化，依次赋值为1,3,5,7,9,11，然后输出。

分析：数组可以在定义是就进行初始化，int a[]={1,2,3,4};定义的数组 a,有 4 个元素，a[0]=1,a[1]=2,a[2]=3,a[3]=4。

此处能使用变量a[4]吗？一定不能使用a[4]。因为只声明了a[0]-a[3]共4个元素。

参考程序

```
#include<stdio.h>
int main()
{
int a[]={1,3,5,7,9,11}; //自动确定为6个元素，下标为0-5
for(int i=0;i<=5;i++) printf("%d ",a[i]);
return 0;
}
```

输出：1 3 5 7 9 11

定义数组时赋初值要注意以下几点：

（1）可以只给部分元素赋初值。

double a[100]={1.5,2},即 a[0]=1.5,a[1]=2,一般情况下，a[2],…,a[99]均是0；

int b[10]={0,1,2},即b[1]=1,b[2]=2,其余的元素为0。

（2）int a[3]={1,2,3,4};运行会出错，原因是数组元素只有3个，但有4个初值。也就是初值的个数不能超过元素的个数。

（3）初值之间用“,”隔开。

例6.3 输入10个数，输出每个数与这10个数平均值之差。

分析：由先要求出这10个数平均值，才能依次求这10个数与平均值的差。所以这10个数保存到数组中。于是得到算法：

（1）输入10个数存于数组a中，并求出平均值；

（2）依次输出这10个数与平均值的差。

参考程序

```
#include<stdio.h>
int main()
{
    double a[11],ave=0;//定义为实数，平均数一般可能有小数
    for (int i=1;i<=10;i++)
    {
      scanf("%lf",a+i); //注意此处是a+i
      ave+=a[i];
    }
    ave/=10;
    for (int i=1;i<=10;i++)
     printf("%0.2lf ",a[i]-ave);
    return 0;
}
```

输入 1 2 3 4 5 6 7 8 9 10

输出 -4.50 -3.50 -2.50 -1.50 -0.50 0.50 1.50 2.50 3.50 4.50

说明：

（1）语句“scanf("%lf",a+i);”中用“a+i”代表a[i]的地址。

（2）a是一维数组名，a代表一维数组在内存中的起始位置，即数组a的第一个元素存放的地址。a+i代表a[i]内存中的地址。所以不能&(a+i)。设a[0]的地址为2000，由于每个double实数占用8个字节，所以a[1]的地址为2008。具体情况如下表：

	第0个	第1个	第2个	第3个	……	第8个	第9个
	a[0]	a[1]	a[2]	a[3]	……	a[8]	a[9]
地址	2000	2008	2016	2024	2032. ……	2072	2080
地址	a+0	a+1	a+2	a+3	a+4 ……	a+8	a+9

a+i并不是单纯的“+”，而是a+i*每个元素字节数，得到第i个元素的地址。

例6.4 将一个表格中的数据按四舍五入保留1位小数按要求输出。

1.289	1.081	1.35	4.28
3.54	5.62	6.38	5.48
6.32	8.37	9.40	8.34

输出

```
1.3  1.1  1.4  4.3
3.5  5.6  6.4  5.5
6.3  8.4  9.4  8.3
```

分析：要解决此问题，要用到二维数组。一组数组可以看成是直线，其元素可以看成直线上的点，只要一个下标就决定一个元素在数组中的位置（直线上的点）；二维数组可以看成平面，其元素可以看成平面上的点，要两个下标才能决定一个元素在数组中的位置（平面上的点）。

二维数组的声明：类型符 数组名[整型常量表达式][整型常量表达式]

```
int  a[4][2];
```

说明：

（1）定义了一个4×2（4行2列）的二维整型数组a,每一维用[ ]括起来；

（2）每一维下标从0开始。行：0－3，列：0－1；

（3）a可以看成一个维数组，它有4个元素：a[0],a[1],a[2],a[3],而每个元素又是一个包含2个元素的一维数组：

a[0]--a[0][0],a[0][1];

a[1]--a[1][0],a[1][1];

a[2]--a[2][0],a[2][1];

a[3]--a[3][0],a[3][1]

也就是说a[0],a[1],a[2],a[3]也可看成一维数组名。

（4）一般来说，计算机语言中二维数组是按行存储的。设a[0][0]存储的内存起始地址为2000，每个整数占4个字节，第0行元素在内存中存储的地址2000－2007，第1行元素在内存中存储的地址2008－2015。具体表示方式如下表：

	第0行元素		第1行元素		第2行元素		第3行元素	
	a[0][0]	a[0][1]	a[1][0]	a[1][1]	a[2][0]	a[2][1]	a[3][0]	a[3][1]
地址	2000	2004	2008	2012	2016	2020	2024	2028
地址	a[0]		a[1]		a[2]		a[3]	
地址	a[0]+0	a[0]+1	a[1]+0	a[1]+1	a[2]+0	a[2]+1	a[3]+0	a[3]+1

（5）二维数组的初始化。

①分行赋初值：int a[4][2]={{1,2},{3,4},{5,6},{7,8}};赋值后如下

a[0][0]	a[0][1]	a[1][0]	a[1][1]	a[2][0]	a[2][1]	a[3][0]	a[3][1]
1	2	3	4	5	6	7	8

②一起赋初值：int a[4][2]={ 1,2,3,4,5,6,7,8};赋值后如下

a[0][0]	a[0][1]	a[1][0]	a[1][1]	a[2][0]	a[2][1]	a[3][0]	a[3][1]
1	2	3	4	5	6	7	8

③部分元素赋初值：int a[4][2]={{1},{4},{5},{7,8}};赋值后如下

a[0][0]	a[0][1]	a[1][0]	a[1][1]	a[2][0]	a[2][1]	a[3][0]	a[3][1]
1	0	4	0	5	0	7	8

未赋初值的元素，系统默认为0。

现在，回到例3，其解决思路为：

（1）先把表格中的初值存入二维3×4数组a中；

（2）输出数组a中的值。

输出一个二维数组，一行一行地输出（从0行开始）。于是有

```
for (i=0;i<=2;i++)
{
  输出第i行数据;
  换行;
}
```

回顾一下，这是不是和例5.22输出图案的方法差不多呢？

输出第i行数据，当然是一个一个的输出。

```
for (j=0;j<=3;j++)
  printf("%0.1lf ",a[i][j]);
```

至此，我们可以写出该题的程序了。

参考程序

```
#include<stdio.h>
int main()
{
  double a[3][4]={{1.289,1.081,1.35,4.28},
                  {3.54,5.62,6.38,5.48},
                  {6.32,8.37,9.40,8.35}};
  //最好写成上面的形式赋初值，对应关系明确。
  int i,j;
  for (i=0;i<=2;i++)
  {  //输出第i行的元素
    for (j=0;j<=3;j++)
      printf("%0.1lf ",a[i][j]);
    printf("\n");   //换行
```

```
  }
  return 0;
}
```

输出

```
1.3 1.1 1.4 4.3
3.5 5.6 6.4 5.5
6.3 8.4 9.4 8.3
```

说明：在教学过程中，经常有学生不知道怎么输出这种二维的形式。实际上只要注意按下面的模式是一行一行的输出，就可以了。

```
for (i=1;i<=行数;i++)
{
   输出第 i 行数据;
   换行;
}
```

例 6.5 将一个表格 a 的数据行与列交换，存入另一个表格 b 中,如下图所示。

表格 a

1.3	2.1	3.9
4.5	5.3	6.7

表格 b

1.3	4.5
2.1	5.3
3.9	6.7

分析：可以先把表格 a 中初值存入二维 2×3 数组 a，输出 a 数组，再按对应关系，存入 3×2 数组 b，输出数组 b。

参考程序：

```
#include<stdio.h>
int main()
{
  double a[2][3]={{1.3,2.1,3.9},{4.5,5.3,6.7}},b[3][2];
  int i,j;
  //以下是输出数组 a 各元素的值
  for (i=0;i<=1;i++)
   {
     for (j=0;j<=2;j++)
       printf("%0.2lf ",a[i][j]);
     printf("\n");
   }
  //以下是将数组 a 中各元素行与列交换存入数组 b 中
  for (i=0;i<=1;i++)
    for (j=0;j<=2;j++)
      b[j][i]=a[i][j];
  printf("\n");  // 增加一个空行，将两个数组分隔开
    //以下是输出数组 a 各元素的值
```

```
  for (i=0;i<=2;i++)
   {
     for (j=0;j<=1;j++)
       printf("%0.2lf ",b[i][j]);
     printf("\n");
   }
  return 0;
}
```

输出：

```
1.30 2.10 3.90
4.50 5.30 6.70

1.30 4.50
2.10 5.30
3.90 6.70
```

例 6.6 一个学习小组有 5 个人，每个人有三门课的考试成绩。求每人平均成绩、每科平均成绩，并找出所有成绩中的最高分。要求成绩通过键盘输入。

	张	王	李	赵	周
语文	80	61	59	85	76
数学	75	65	63	87	77
外语	92	71	70	90	85

分析：可设一个二维数组 a[4][6]存放五个人三门课的成绩（不使用 0 行 0 列元素），max 存放最高分。

参考程序：

```
#include<stdio.h>
int main()
{
  int a[4][6],i,j,max=0;//max 赋为 0，成绩不可能为负。
  double kp,rp;
  //以下是读入数据，并求出最大值
  for (i=1;i<=3;i++)
    for (j=1;j<=5;j++)
      {
       scanf("%d",&a[i][j]);
       if (max<a[i][j]) max=a[i][j];//当前的成绩，比先前的最高成绩还要高，更新最高分
      }
  //以下是求出每一科的平均分，并输出
  for (i=1;i<=3;i++)
```

```
    {
      kp=0;
      for (j=1;j<=5;j++)
       kp+=a[i][j];
      kp/=5;
      printf("%0.2lf ",kp);
    }
    printf("\n");
    //以下是求出每人的平均分，并输出
   for (i=1;i<=5;i++)
    {
      rp=0;
      for (j=1;j<=3;j++)
       rp+=a[j][i];  //注意此处不能写成 a[i][j],想一想为什么？
      rp/=3;
      printf("%0.2lf ",rp);
    }
   printf("\nmax=%d",max);
   return 0;
}
```

输入：

```
80 61 59 85 76
75 65 63 87 77
92 71 70 90 85
```

输出：

```
72.20 73.40 81.60
82.33 65.67 64.00 87.33 79.33
max=92
```

例 6.7 打印杨辉三角形（要求 10 行）

```
                    1
                 1     1
              1     2     1
           1     3     3     1
        1     4     6     4     1
     1     5    10    10     5     1
```

分析：这个种形状的数据，不好保存。一般都向左靠齐如下：

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
```

1 5 10 10 5 1

设以上的数据保存在数组a中，可以看出:

第1列的数全是1（a[i][1]=1),

对角线上的数也是1(a[i][i]=1),

其它位置的数据（下三角）a[i][j]=a[i-1][j-1]+a[i-1][j]。

这个数据就容易生成了(见参考程序)。

剩下的问题就是打印图案，用前面讲的打印图案模式编写即可。

参考程序

```
#include<stdio.h>
int a[15][15];
int main()
{
  a[1][1]=1;a[2][1]=1;a[2][2]=1;
  for (int i=3;i<=10;i++)
  {
    a[i][1]=1;a[i][i]=1;
    for (int j=2;j<=i-1;j++)
      a[i][j]=a[i-1][j-1]+a[i-1][j];
  }
  for (int i=1;i<=10;i++)
  {
    for (int j=1;j<=42-i*3;j++)
      printf(" ");
    for (int j=1;j<=i;j++)
      printf("%6d",a[i][j]);//一个数占6列
    printf("\n");
  }
  return 0;
}
```

说明：

（1）杨辉三角形，又称贾宪三角形，帕斯卡三角形。

（2）观察下面的式子

$(x+y)^0=1$

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

$(x+y)^2=x^2+2xy+y^2$

$(x+y)^3=x^3+3x^2y+3xy^2+y^3$

$(x+y)^4=x^4+4x^3y+6x^2y^2+4xy^3+y^4$

……

上面这些式子称为二项展开式，其系数称为二项式系数，这些二项式系数就是杨辉三角形中的数。

例6.8 求fibonacci数列$a_1,a_2,\ldots,a_{20}$。

$a_1=1$

$a_2=1$

$a_n=a_{n-2}+a_{n-1}$ (n>2)

分析:前面我们用的平移法来解决此问题。现在可以使用数组，直接使用 a[i]=a[i-2]+a[i-1] 即计算出第 i 项,这样比前面的平移法简单多了。

参考程序

```
#include<stdio.h>
int main()
{
int i,a[25]={0,1,1};
for(i=3;i<=20;i++) a[i]=a[i-2]+a[i-1];
for(i=1;i<=20;i++) printf("%d ",a[i]);
return 0;
}
```

输出：1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765

说明：

（1）int a[25]={0,1,1};此处把 0 赋给 a[0],a[1]=1,a[2]=1,此处 0 起占位作用。

（2）int a[25]={0,1,1}; 本题只需要 20 项，而我们定义了 25 个元素，使用数组一般多定义几个元素，避免因为不小心下标超界。

例 6.9 输入年和月，输出该月有多少天。

分析：已知年 year 和 month

求这个月的天数 days

实际上除了 2 月，其它月份都是不变的，我们可以用一个数组把平年每一个月的天数保存下来，如果是闰年再把 2 月的天数改成 29 天。

参考程序：

```
#include<stdio.h>
int main()
{
int month,year;
int days[15]={0,31,28,31,30,31,30,31,31,30,31,30,31};
scanf("%d%d",&year,&month);
if (year%100!=0&&year%4==0||year%400==0) days[2]=29;
printf("%d",days[month]);
return 0;
}
```

用上面的思路来解决，比以前的方法简单得多吧！主要思路是：先假设是几种情况中的一种（先假设是平年，2 月是 28 天），然后再来修正。即不是假设的情况（即为闰年），改变成现在的状态（改成 2 月 29 天）。这种方法希望同学们细细品味，我们经常会用到。

例 6.10 输入 10 个整数，从小到大排序后输出

分析：这种问题称为排序。先把 10 个数存入数组 a 中，然后从小到大排序后，输出即可。

这里介绍一种冒泡排序法,为演示方便，设有 5 个数。

如下图所示，

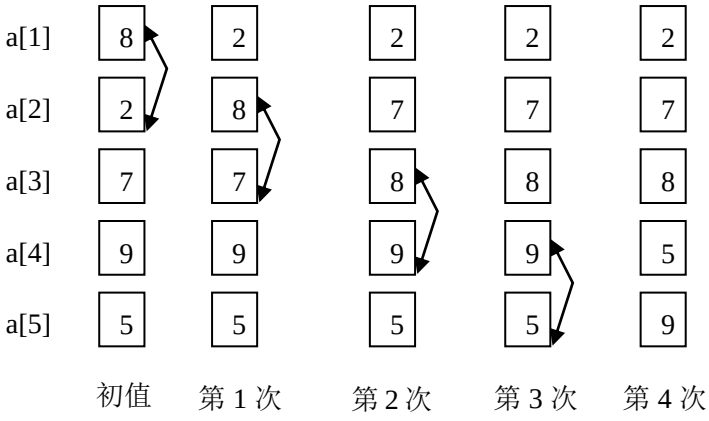
第 1 次：a[1]>a[2]，交换 a[1]与 a[2]的值；

第 2 次：a[2]>a[3]，交换 a[2]与 a[3]的值；

第 3 次：a[3]<a[4]，不交换 a[3]与 a[4]的值；

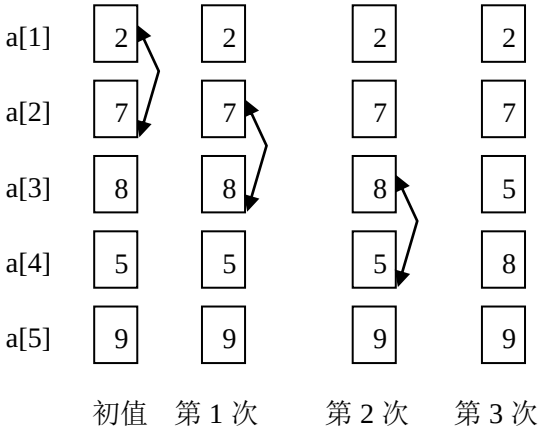
第 4 次：a[4]>a[5]，交换 a[4]与 a[5]的值。

这称为一趟比较，比较了 4 次，至此，a[5]中已是最大值。



第 1 趟比较

接下来，对剩下的前 4 个数进行和上面一样的比较如下图



第 2 趟比较

如上图所示，

第 1 次：a[1]<a[2]，不交换 a[1]与 a[2]的值；

第 2 次：a[2]<a[3]，不交换 a[2]与 a[3]的值；

第 3 次：a[3]>a[4]，交换 a[3]与 a[4]的值。

这一趟比较了 3 次，至此，把余下的最小值放入到 a[4]中。

请想一想，总共要几趟，每趟比较多次，能完成所有数的排序？

5 个数要 4 趟，因为每一趟都把剩下数中的最大值放到合适的位置上。第 1 趟比较了 4 次，第 2 趟比较了 3 次，第 3 趟比较了 2 次，第 4 趟比较了 1 次。

每一趟比较都是大数向下“沉”，小数向上“升”，所以我们形象称这种排序为冒泡排序。

于是得到了冒泡排序的思路:

```
for (i=1;i<=4;i++)
  for (j=1;j<=5-i;j++)
    if (a[j]>a[j+1]) 交换 a[j]、a[j+1]
```

参考程序

```
#include<stdio.h>
int a[15];
int main()
{
  int i,j,t;
  for (i=1;i<=10;i++)
    scanf("%d",&a[i]);
  for (i=1;i<=9;i++)     //10 个数比较 9 趟
    for (j=1;j<=10-i;j++) //每趟都从第 1 个与第 2 个比较开始，进行 10-i 次比较
      if (a[j]>a[j+1]) //前面的大交换
      { //交换 a[j]与 a[j+1]
        t=a[j];
        a[j]=a[j+1];
        a[j+1]=t;
      }
  for (i=1;i<=10;i++)  //输出排好序的结果
    printf("%d ",a[i]);
 return 0;
}
```

输出:9 2 7 8 3 2 1 5 4 0 输出:0 1 2 2 3 4 5 7 8 9

说明：（1）设有 n 个数存到数组 a 中，要求从小到大排序的模式:

```
  for (i=1;i<=n;i++)     //n 个数比较 n-1 趟
    for (j=1;j<=n-i;j++) //每趟都从第 1 个与第 2 个比较开始，进行 n-i 次比较
      if (a[j]>a[j+1])  //相邻的两个数，前面的大于后面的，所以交换
      { //交换 a[j]与 a[j+1]
        t=a[j];
        a[j]=a[j+1];
        a[j+1]=t;
      }
```

如果要求从大到小排序呢？只需要 a[j]>a[j+1]改成 a[j]<a[j+1]即可。要牢记上面冒泡排序的模式。

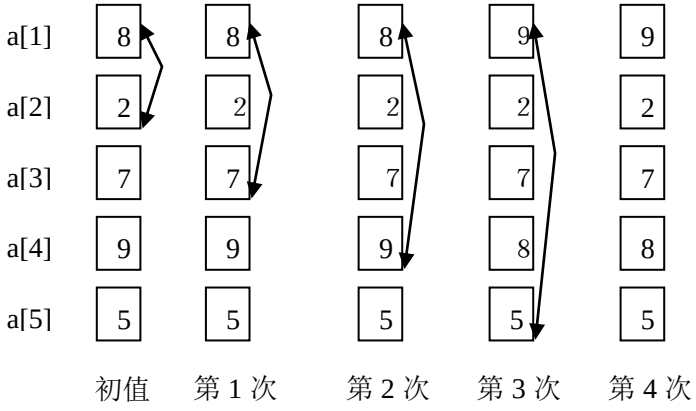
（2）此种排序的时间复杂度为 $O(n^2)$,如果 n 太大，超过 5000，在 1 秒钟内可能出不了结果，这就是我们通常所说的超时。

例 6.11 输入正整数 n(<=3000)，接下来输入 n 个整数，要求把这 n 个整数从大到小排序后输出(用选择排序完成)

分析：算法如下

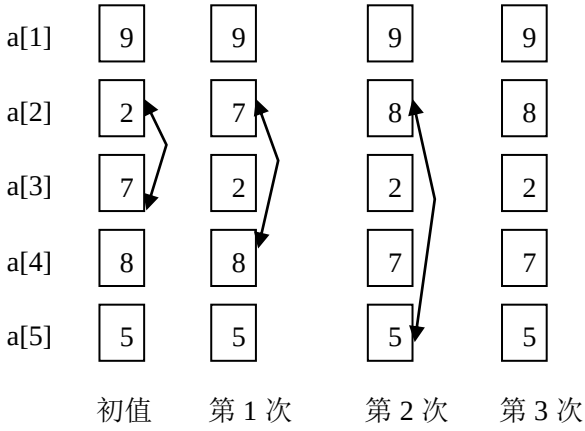
（1）输入 n 个数存入数组 a 中；

（2）从大到小进行选择排序；
（3）输出这 n 个数。
下面介绍选择排序，为演示方便，设有 5 个数。



第 1 趟比较

如上图所示，
第 1 次：a[1]>a[2]，不交换 a[1]与 a[2]的值；
第 2 次：a[1]>a[3]，不交换 a[1]与 a[3]的值；
第 3 次：a[1]<a[4]，交换 a[1]与 a[4]的值；
第 4 次：a[1]>a[5]，不交换 a[1]与 a[5]的值。
这称为一趟比较，比较了 4 次，至此，a[1]中已是最大值。
接下来，对剩下的后 4 个数进行和上面一样的比较如下图



第 2 趟比较

如上图所示，
第 1 次：a[2]<a[3]，交换 a[2]与 a[3]的值；
第 2 次：a[2]<a[4]，交换 a[2]与 a[4]的值；
第 3 次：a[2]>a[5]，不交换 a[2]与 a[5]的值。
这一趟比较了 3 次，至此，把余下的最大值放入到 a[2]中。
请想一想，总共要几趟，每趟比较多次，能完成所有数的排序？
5 个数要 4 趟，因为每一趟都把剩下数中的最大值放到合适的位置上。第 1 趟比较了 4 次，第 2

趟比较了 3 次，第 3 趟比较了 2 次，第 4 趟比较了 1 次。

第 i 趟比较都是选择第 i 个元素——第 n 元素中的最优值（最大值或最小值）放到第 i 个位置，称这种排序为选择排序。

于是得到了选择排序的思路:

```
For (i=1;i<=4;i++)
  For (j=i+1;j<=5;j++)
    if (a[i]<a[j]) 交换 a[i]、a[j]
```

参考程序

```
#include<stdio.h>
int a[3005];
int main()
{
  int i,j,t,n;
  scanf("%d",&n);
  for (i=1;i<=n;i++)
    scanf("%d",&a[i]);
  for (i=1;i<=n-1;i++)     //n 个数比较 n-1 趟
    for (j=i+1;j<=n;j++) //第 i 趟是第 i 与第 i+1 个比较开始，直到第 n 个进行比较
      if (a[i]<a[j]) //前面的小交换
      { //交换 a[i]与 a[j]
        t=a[i];
        a[i]=a[j];
        a[j]=t;
      }
  for (i=1;i<=n;i++)  //输出排好序的结果
    printf("%d ",a[i]);
return 0;
}
```

说明：

（1）选择排序模式(n 个数已存入数组 a 中，从大到小排列)

```
  for (i=1;i<=n-1;i++)     //n 个数比较 n-1 趟
    for (j=i+1;j<=n;j++) //第 i 趟是第 i 与第 i+1 个比较开始，直到第 n 个进行比较
      if (a[i]<a[j]) //前面的小交换
      { //交换 a[i]与 a[j]
        t=a[i];
        a[i]=a[j];
        a[j]=t;
      }
```

（2）冒泡排序与选择排序的比较（n 个数）

	冒泡排序	选择排序
比较趟数	n-1	n-1
第 i 趟最优值存放位置	第 n+1-i 位置（后面）	第 i 个位置（前面）

第 i 趟比较过程	第 1 个元素~第 n-i 个元素依次与后面相邻的元素比较	第 i 个元素与第 i+1~n 个元素依次比较
时间复杂度	O（n^2）	O（n^2）
空间复杂度	O（n）	O（n）

例 6.12 输入正整数 n(<=3000)，接下来输入 n 个整数，要求把这 n 个整数从小到大排序后输出(用插入排序完成)

分析：假设有 5 个整数（分别是：8、2、7、6、5），排序结果存放在数组 a 中。

插入排序是在数组中已有数据有序的基础上进行的。数组中已有数据是有序的，将一系列数 x 依次插入到数组中，使所有数据有序。

假设数组中存有 i-1(=3)个数，且有序（从小到大）。现有 x(=6)要插入数组中。下面是插入过程：

a[1]	a[2]	a[3]	a[4]	a[5]
2	7	8		

（1）找到数 x(=6)可插入数组中的位置前一个位置 k=1;

（2）将 a[k+1]~a[m]中数据都向后平移一位;

a[1]	a[2]	a[3]	a[4]	a[5]
2	7	7	8	

（3）a[k+1]=x;

a[1]	a[2]	a[3]	a[4]	a[5]
2	6	7	8	

这时数组中 i 个数也已有序。

第(1)步，从最后一个数（a[i-1]）向前查找，找到第一个 a[k]<x，则 x 该插入到 k+1 位置上。

向前查找的过程中，有可能找到 0 号位置(x 比 a[1]~a[i-1]中所有的数都小)。为了避免超过 0 号位置，我们可以把 a[0]设为“哨兵”元素（其值设为比输入数据都小的数，如-2100000000）。

第（2）步，平移从后向前开始。

于是得到程序框架

```
for (int i=1;i<=n;i++)
  {
    scanf("%d",&x);
    for (k=i-1;a[k]>x;k--) ;  //查找插入的位置 k+1
    for (int j=i-1;j>=k+1;j--) //将 k+1~i-1 位置上的数向后平移到 k~i 位置上
       a[j+1]=a[j];
    a[k+1]=x;              //x 放到正确的位置上
  }
```

参考程序

```
#include<stdio.h>
int a[3005];
int main()
{
  int n,x,k;
  scanf("%d",&n);
```

```
    a[0]=-210000000;  //设置“哨兵”元素
    for (int i=1;i<=n;i++)
    {
      scanf("%d",&x);
      for (k=i-1;a[k]>x;k--) ;  //查找插入的位置 k+1
      for (int j=i-1;j>=k+1;j--) //将 k+1~i-1 位置上的数平移
         a[j+1]=a[j];
      a[k+1]=x;              //x 放到正确的位置上
    }
    for (int i=1;i<=n;i++)  //输出排好序的结果
      printf("%d ",a[i]);
      return 0;
}
```

说明：

（1）插入排序就是依次把数插入到已经有序的序列中，使新序列依然有序。

（2）插入排序分为三步：

第一步，查找可插入的位置；

第二步，平移，即空出插入位置；

第三步，插入该数。

例 6.13 输入一个正整数 n(2<=n<=10^6),输出 1-n 之间的素数。

输入 10 输出 2 3 5 7

分析：前面介绍了判断数 i 是否素数的方法是判断 2～$\sqrt{i}$ 中是否有 i 的因子，如果没有约就是素数。于是思路为

```
for (i=2;i<=n;i++)
{
  判断 i 是否素数；
  如果 i 是素数，输出 i;
}
```

但这个时间复杂度为 O(nlogn),当 n=10^6时，运行次数达到 10^8，显然会超时。

下面介绍筛法求素数的方法。

先设所有的素数都是素数，用一个数组 a 来标记，所有元素都设为 0,表示都设为素数。设 n=10,如下表。

	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]
初值	0	0	0	0	0	0	0	0	0
2 是素数	0	0	1	0	1	0	1	0	1
3 是素数	0	0	1	0	1	0	1	1	1
5 是素数	0	0	1	0	1	0	1	1	1
7 是素数	0	0	1	0	1	0	1	1	1

从 i=2 开始,i 从小到大,不断重复下面的操作。

找到 a[i]==0（i 一定是素数），然后把 i 的倍数（2 倍及以上），标志设为 1（它们都不是素数，

有因子 i)。

于是得到思路：

```
for (i=2;i<=n;i++)
{
  if (a[i]==0)
  {输出 i;
   把 i 的倍数的标记设为 1;
  }
}
```

参考程序

```
#include<stdio.h>
int a[1000005]={0};//先假所有的素为素数，标记为 0
int main()
{
  int n;
  scanf("%d",&n);
  for (int i=2;i<=n;i++)
  {
    if (a[i]==0)//i 是素数
    {
     printf("%d ",i);
     for (int j=2*i;j<=n;j+=i) //筛掉 i 的倍数
       a[j]=1;
    }
  }
  return 0;
}
```

输入 20 输出 2 3 5 7 11 13 17 19

说明：该算法的时间复杂度接近 O(n),有些同学可能会问，一个二重循环，怎么时间复杂度接近 O(n)呢？因为一般来说某个范围内的素数不是太多：100 内素数 25 个，1000 内 168 个，10000 内 1229 个，100000 内 9592 个，1000000 内 78948 个，….,n 比较大时，素数的个数不足$\frac{1}{10}$，也就是说内层循环执行的次数相当少。

例 6.14 将“Chinese”保存到数组中，并输出。

分析：用一个数组 c 来保存“Chinese”。而“Chinese”是由字符构成，所以这里要涉及到**字符数组**。

字符数组也允许在定义时作初始化赋值。

例如：char c[7]={'C','h','i','n','e','s','e'};

说明：

（1）数组各元素的值如右表。

（2）字符一定要用''括起来，注意此处

c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]
'C'	'h'	'i'	'n'	'e'	's'	'e'

是两个单引号，如'a'。

（3）当对全体元素赋初值时也可以省去长度说明。

char c[ ]={'C','h','i','n' ,'e','s','e'};这时c数组的长度自动定为7。

（4）{}内字符个数不能大于数组的长度；{}内字符个数小于数组的长度，后面的元素系统自动定为空字符（即' \0'）。

如char c[4]={'C','h','i','n','s','e'};出现语法错误；

如char c[8]={'C','h','i','n','s','e'};各元素的值如下表。

c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]
'C'	'h'	'i'	'n'	'e'	's'	'e'	' \0'

好了，我们回到刚才问题上来，定义一个字符数组，每个元素赋初值依次为'C','h','i','n','s','e'。然后输出这个字符数组的内容即。

参考程序一：

```
#include<stdio.h>
int main()
{ char c[7]={'C','h','i', 'n','e','s','e'};
  for (int i=0;i<=6;i++)
    printf("%c",c[i]);
  return 0;
}
```

说明：char c[7]={'C','h','i', 'n','e','s','e'};是定义在 int main()函数内的，如果用后面讲到的puts()将c作为字符串输出，输出的内容：“Chinese”可能还跟有其它字符。

下面讲解字符串有知识

字符串

实际上，我们可以把“Chinese”,作为字符串来处理。在C语言中没有专门的字符串变量，通常用一个字符数组来存放一个字符串。当把一个字符串存入一个字符数组时，在最后会存入一个空字符'\0'，并以此作为该字符串是否结束的标志。有了'\0'标志后，就不必再用字符数组的长度来判断字符串的长度了。

C语言允许用字符串的方式对数组作初始化赋值。例如：

char c[]={'C','h','i','n', 'e','s','e'};可写为：char c[]={"Chinese"};

或去掉{}写为：char c[]="Chinese";

c在内存中的实际存放情况如下表：

c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]
'C'	'h'	'i'	'n'	'e'	's'	'e'	' \0'

说明：

（1）用字符串方式赋值比用字符逐个赋值要多占一个字节， 用于存放字符串结束标志'\0'。'\0'是由C编译系统自动加上的。

（2）采用了'\0'标志，所以在用字符串赋初值时一般无须指定数组的长度， 而由系统自行处理。

（3）字符串输入

1）逐个字符的输入：scanf("%c",&c[0]);读入一个字符存入 c[0]中。

2）将整个字符串一次输入：

scanf("%s",c);将一串字符存入字符数组 c 中，同时在最后加上'\0'。这里不能写成&c，因为前面已经说明数组名表示这个数组的首地址。

gets(c);读入一个字符串到字符数组 c 中，并且得到一个函数值，这个函数值是字符数组的起始地址。

两种方式都能读入一个字符串到数组中，它们的**主要区别**：

scanf("%s",c)，识别字符串结束是对空格或换行符，也就是说字符串中不能有空格。而 gets(c)识别字符串结束只是换行符，也就是说以行结束来标志字符串结束。

（4）字符串输出

用 printf("%s",c);和 puts(c);均可，其作用是将一个字符串 c（以'\0'结束）输出。

puts（）输出字符串后会换行。

于是我们可以改写刚才的程序为：

参考程序二：

```
#include<stdio.h>
int main()
{
  char c[]="chinese";//实际上 c[7]= '\0'
  puts(c);
  return 0;
}
```

例 6.15 有一个同学的成绩表如下表，编程输出成绩最高的科目。

Chinese	Math	English	Computer
136	125	143	128

要求，科目和成绩按以下方式输入，每行一个科目名和该科成绩，之间用一个空格隔开。

```
Chinese 136
Math 125
English 143
Computer 128
```

分析：要保存科目名，定义一个字符数组 km,应该是二维字符数组，每一行保存一科目名。而科目名的字符最多为 8 个，所以每一行至少要存 9 个字符（最后一个保存'\0'），如果不使用 0 行，要定义 5 行，即 char km[5][10],就能保存下所有的科目名。

再定义一个整型数组保存成绩。Int cj[5];

那么，现在枚举每一个 cj[i],找出最高成绩的位置。记住其位置，设为 k,最后输出 km[k]即可。

参考程序

```
#include<stdio.h>
char km[5][10];
int cj[5],max=0,k=0;//现假设最高成绩为 0,因最高成绩不可能小于 0,当前最高成绩的位置 k 也设 0;
int main()
{
```

```
  for (int i=1;i<=4;i++)
    scanf("%s%d",km[i],&cj[i]);  //想一想为什么此处要用 scanf 来读字符串？
  for (int i=1;i<=4;i++)
    if (max<cj[i])
      {
        max=cj[i];
        k=i;
      }
  puts(km[k]);
  return 0;
}
```

输出 English

说明：“scanf("%s%d",km[i],&cj[i]);”语句中 km[i]未用取地址符&，km[i]代表二维数组 km 的第 i 行的首地址。

字符串函数

下面介绍的字符串函数包含于头文件 string.h 中，所以在程序开头要加上#include<string.h>，才能使用。

1. strcat 函数——字符串连接函数

格式： strcat (字符数组 1，字符串 2)

strcat 是 string catenate(字符串连接)的简写。

功能：把字符串 2 连接到字符数组 1 中字符串的后面。函数返回值是字符数组 1 地址。

```
char s1[100]="My name is ",s2[100]="Jason.";
strcat(s1,s2);
strcat(s2," is a student.");
```

s1 的值是"My name is Jason. "，s2 的值是"Jason. is a student. "

说明：

（1）字符数组 1 足够长，以便容纳连接后的新字符串。

（2）字符串 2，可以是字符串常量、字符串数组。

（3）连接前两个字符串后面都有'\0',字符数组后面的'\0'去掉，只在新串后加'\0'。

2. strcpy 和 strncpy 函数--字符串拷贝函数

格式：strcpy (字符数组 1，字符串 2)

strcpy 是 stringcopy(字符串复制)的简写。

功能：把字符串 2 中的字符串拷贝到字符数组 1 中。字符串结束标志“\0”也一同拷贝。

```
char s1[100]="My name is ",s2[15]="Jason.";
strcpy(s1,s2);
strcpy(s2,"Marry.");
```

s1 的值是"Jason."，s2 的值是"Marry."

说明：

（1）字符数组 1 足够长，以便容纳连接后的新字符串。

（2）字符串 2，可以是字符串常量、字符串数组。

（3）字符串不能用赋值语句进复制。s2="Marry.";这是不合法的。赋值语句只能对字符、整数、

实数、布尔型、结构体（后面将会介绍）进行赋值，注意只能是一个元素。如 char c=s[1]。

（4）strncpy (字符数组 1，字符串 2,n)

将字符串 2 中前面 n 个字符复制到字符数组 1 中去。注意只改变字符数组 1 的前 n 个字符为字符串 2 的前 n 个字符。

char s1[15]="123456789",s2[20]="ABCD";

strncpy(s1,s2,2);

s1 的值为"AB3456789"。

3. strcmp 函数——字符串比较函数

格式： strcmp(字符串 1，字符串 2)

strcmp 是 string compare(字符串比较)的简写。

功能：比较两个字符串，并由函数返回值返回比较结果。

$$\text{strcmp(字符串1, 字符串2)} = \begin{cases} -1 & \text{字符串1} < \text{字符串2} \\ 0 & \text{字符串1} = \text{字符串2} \\ 1 & \text{字符串1} > \text{字符串2} \end{cases}$$

说明：

（1）字符串的比较规则：

将两个字符串从左到右逐个字符相比（按照 ASCII 码值大小比较），直到出现不同的字符或遇到'\0'为值。

1）全部字符相同，则相等。

2）出现不同字符 ASCII 码值大的字符所在串大。

如"AB34" <"AC","AB34"<"a"。

（2）两个字串比较不能使用关系运算符。

char s1[10]="AB34",s2[10]="AC";

则 if (s1<s2) printf("Yes");不合法。应写成 if (strcmp(s1,s2)==-1) printf("Yes");

4. strlen 函数——测字符串长度函数

格式：strlen(字符串)

strlen 是 string length(字符串长度)的简写。

功能：测字符串的实际长度(不含字符串结束标志‘\0’) 并作为函数返回值。

char s[10]="AB34";

strlen(s)的值是 4,strlen("12%$*yz")的值是 7。

5. strlwr 函数——转换为小写字母的函数

格式：strlwr(字符数组)

strlwr 是 string lowercase(字符串小写)的简写。

功能：将字符数组中的英文大写字母换成小写字母，并存在原位置，函数返回值字符串的地址。

char s[10]="AB34";

执行 strlwr(s);后，s 的值是"ab34"。

说明：strlwr()的参数不能是字符串常量。

6. strupr 函数——转换为大写字母的函数

格式：strupr(字符数组)

strupr 是 string uppercase(字符串大写)的简写。

功能：将字符数组中的英文小写字母换成大写字母，并存在原位置，函数返回值字符串的地址。

char s[10]="ab34";

执行 strupr(s);后，s 的值是"AB34"。

说明：strupr()的参数不能是字符串常量。

7.memset（ ）函数。如：memset(a,0,sizeof(a));//将a数组所有元素全部清“0”

例6.16 输入一行字符（不超过1000个字符）存入数组，然后把他们反序存入到同一数组中。

分析:将一行字符存入数组a中。测出a的长度len。然后a[0]与a[len-1]交换,a[1]与a[len-2]交换，直到该数组中间为止。

于是得到框架结构

```
For (i=0;i<=len/2-1;i++)
  交换a[i]与a[len-1-i];
```

参考程序

```
#include <stdio.h>
#include<string.h>
char a[1005],t;
int main()
{
    gets(a);
    int len=strlen(a);//测试字符串长度
    for (int i=0;i<=len/2;i++)
      {
        t=a[i];
        a[i]=a[len-1-i];
        a[len-1-i]=t;
      }
    puts(a);
    return 0;
}
```

输入 342652rwef%#$^#$%@1abRETHFGewr

输出 rweGFHTERba1@%#$^$#%fewr256243

例6.17 输入一行字符（不超过1000个字符），如果是回文，输出“回文”，否则输出“不是回文”。回文就是正读和反读都一样。

分析:将一行字符存入数组a中,测出长度len。然后a[0]与a[len-1]是否相等,a[1]与a[len-2]是否相等，如果检验该数组中间都还相等则是回文。

于是得到框架结构

```
For (i=0;i<=len/2;i++)
  if a[i]!=a[len-1-i]  break;
if i>len/2 则是回文
else 不是回文
```

参考程序

```
#include<stdio.h>
#include<string.h>
char a[1005];
```

```
int main()
{
    int len,i;
    gets(a);  //不能用 scanf("%s",a[i]);可能有空格。
    len=strlen(a);
    for (i=0;i<=len/2;i++)
      if (a[i]!=a[len-1-i]) break;  //不是回文，提前结束
    if (i>len/2)
      printf("回文");
    else
      printf("不是回文");
    return 0;
}
```

（1）输入 abcdefg1234567890987654321gfedcba 输出 回文

（2）输入#@423asdfdasfawer 输出 不是回文

说明：for (i=0;i<=len/2;i++)，这里实际上多循环了一次。因为字符串的长度是 len，而循环从 0－len/2。所以多循环了 1 次。虽然，本题不影响结果。但在考虑问题时要想清楚，有时结果可能有误。

例 6.18 输入五个国家的名称（每个国家的名称，不超过 40 个字符）按字典顺序排列输出。输入时每行输入一个国家名称。

分析：五个国家名应由一个二维字符数组 a 来保存。然而 c 语言规定可以把一个二维数组当成多个一维数组处理。 因此本题又可以按五个一维数组处理， 而每一个一维数组就是一个国家名字符串。排序后输出结果即可。

参考程序

```
#include<stdio.h>
#include<string.h>
char a[6][45],b[45];
int main()
{
 for(int i=1;i<=5;i++)
   gets(a[i]);  //不能用 scanf("%s",a[i]);国家名中可能有空格。
 //冒泡排序，字符串排序的模式与数的排序一样，注意比较两个字符串用 strcmp()，而交换两个字符串用 strcpy()
 for (int i=1;i<=4;i++)
   for (int j=1;j<=5-i;j++)
     if (strcmp(a[j],a[j+1])>0)//字符串比较
     {  //字符串比较不能用"=="
      strcpy(b,a[j]);
      strcpy(a[j],a[j+1]);
      strcpy(a[j+1],b);
     }
 for (int i=1;i<=5;i++)
```

```
    puts(a[i]);
 return 0;
}
```

输入	输出
Singapore China France United States of America Poland	China France Poland Singapore United States of America

例 6.19 学号安排

题目描述：每个人在学校都会有一个学号，信息学奥赛也不例外。出于对学生自尊心的保护，决定不按成绩好坏来给一个学生授予学号，而是按姓名拼音的字典序授予学号。

输入：第一行包含一个整数 n，表示一共有 n 个人(n<=100)，接下来 n 行每行一个字符串，表示每个人的姓名（不超过 20 个字符）。

输出：每行一个按原来姓名的先后给出相应的学号

样例输入	样例输出
3 Bbb Aaa Ccc	2 1 3

分析：用一个字符数组 name(二维数组，一行保存一学生的姓名)保存学生的姓名，按字典顺序排序，序号就是学号。但本题增加了难度，要按原来的顺序输出学号。

那怎么办呢？

（1）用一个字符数组 name 保存学生的姓名，输入顺序号再用一个数组 order 保存起来（排序以后才能知道原来顺序号）；

（2）按姓名字典顺序排序（排序 order 中的顺序号一起交换）

（3）按现在的顺序依次授予学号（现在的顺序号）保存于数组 number 中。

（4）按 order 中的顺序号排序

（5）按输入顺序输出学号 number 中的值。

通过上面 5 步是解决了问题，但要经过两次排序，每次排序要交换姓名、顺序号、学号，太麻烦了。

能否把姓名、顺序号、学号一起交换呢？回答是肯定的。但要用到结构体变量。

定义结构体

```
struct 结构体名
{
成员列表
};
```

如 `struct student`

```
{
   char name[25];
   int order,number;
};
```

这里定义的一个结构体类型 student,它包括了 name,order,number 这些成员。我们就可以象使用 int 一样使用 student 了。

student stu; 定义一个结构体变量。这个变量 stu 包含了三个成员 name,orde,number。

说明：

（1）定义结构体类型时{}后面一定有一个“;”。

（2）结构体变量初始化，student stu={"Marry",10,34}，但一定要注意成员的顺序。如 student stu={10, "Marry",34}这样就出问题了。

（3）结构体变量引用：结构体变量名.成员名 如 stu.name stu.order, stu.number 这些变量就和单个变量一样了。

参考程序一

```
#include<stdio.h>
#include<string.h>
struct student  //定义了一个结构体类型 student
{
   char name[21];
   int order,number;
};  //注意此处有";"
student stu[105],x;//定义了一外结构体数组 stu 和一个结构体变量 x
int n;
int main()
{
    scanf("%d",&n);
    for (int i=1;i<=n;i++)
    { scanf("%s",stu[i].name);//保存姓名
      stu[i].order=i;  //记录输入的顺序
    }
    //冒泡排序，按姓名排序
    for (int i=1;i<=n-1;i++)
      for (int j=1;j<=n-i;j++)
        if (strcmp(stu[j].name,stu[j+1].name)==1)//字符串比较
          {
           x=stu[j];stu[j]=stu[j+1];stu[j+1]=x;//结构体变量交换
          }
   for (int i=1;i<=n;i++) //授予学号
     stu[i].number=i;
    //冒泡排序，按输入顺序排序
   for (int i=1;i<=n-1;i++)
      for (int j=1;j<=n-i;j++)
        if (stu[j].order>stu[j+1].order)
          {
           x=stu[j];stu[j]=stu[j+1];stu[j+1]=x;
          }
```

```
  for (int i=1;i<=n;i++)  //输出学号
    printf("%d\n",stu[i].number); //'\n'每一行输出一个学号
  return 0;
}
```

下面介绍该题的另一解决方法。方法是：

用一个字符数组name保存姓名，整型数组来保存学号number。

先假设每个人的学号均为是1号。

下面去枚举每个学生，调整他的学号。于是有框架

```
for (i=1;i<=n;i++)
{调整学生i的学号}
```

关键是怎么调整学生i的学号呢？

第i个学生的姓名和所有的学生姓名进行比较，有1个学生姓名小于第i个学生姓名，第i个学生的学号就增加1个。（想一想，为什么？）

于是有

```
for (i=1;i<=n;i++)
  for (j=1;j<=n;j++)
    if (name[i]>name[j]) number[i]++;
```

参考程序二

```
#include<stdio.h>
#include<string.h>
char name[105][25];
int number[105],n;
int main()
{
  scanf("%d",&n);
  for (int i=1;i<=n;i++)
    { scanf("%s",name[i]);//保存姓名
      number[i]=1;  //设每个人学号均为1
    }
  for (int i=1;i<=n;i++)
    for (int j=1;j<=n;j++)
      if (strcmp(name[i],name[j])==1)
        number[i]++;//一个学生姓名小于name[i],则学号增加1
  for (int i=1;i<=n;i++)
    printf("%d\n",number[i]);
  return 0;
}
```

说明：

（1）此题程序可以到www.ssxdzx.net/bas上（试题题号1022）提交测试。

（2）第二种解法是笔者学生陈昌航想出来的。从编程序上来讲，比第一种解法好多了。只要大家开动脑子，会比老师讲的方法更妙！

例 6.20 校门外的树（noip2005 普及组试题）

【问题描述】某校大门外长度为 L 的马路上有一排树，每两棵相邻的树之间的间隔都是 1 米。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。

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。现在要把这些区域中的树（包括区域端点处的两棵树）移走。你的任务是计算将这些树都移走后，马路上还有多少棵树。

【输入】第一行有两个整数 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。

【输出文件】一行，这一行只包含一个整数，表示马路上剩余的树的数目。

样例输入	样例输出
500 3 150 300 100 200 470 471	298

【数据规模】

对于 20%的数据，区域之间没有重合的部分；

对于其它的数据，区域之间有重合的情况。

分析：在 0,1,2,..,L,这些点均有一棵树，现要挖去很多区间上的树，有些区间可能重合，问最后还有多少棵树？

是把这些区间去掉重合的部分，再挖去吗？这样太麻烦了！

我们可以定义一个数组 tree 来表示这些树的挖了（设为 0）还是没有挖（设为 1），有一个要挖的区间，就把这一段树的状态设为 1，表示挖了。于是得到了本题思路：

```
for (i=1;i<=m;i++)
{
  读入一个区间[a,b];
  将 tree[a]-tree[b]均设为 1;
}
```

最后统计 tree[0]-tree[L]状态为 0 有多少个，即有多少棵树。

参考程序：

```
#include<stdio.h>
int tree[10005]={0},L,m,a,b,ans=0;
int main()
{
    scanf("%d%d",&L,&m);
    for(int i=1;i<=m;i++)
    {
      scanf("%d%d",&a,&b);
      for(int j=a;j<=b;j++)
      tree[j]=1;
    }
```

```
    for(int i=0;i<=L;i++)//注意此下从 0 开始
     if(tree[i]==0) ans++;
    printf("%d",ans);
    return 0;
}
```

输入	输入
1000 5	1000 4
0 100	0 100
101 200	50 200
900 1000	199 900
207 400	909 989
401 899	输出 19
输出 6	

说明：

（1）数组一般定义在函数之外，即在 int main()之前。因为如果数组很大定义在函数内，运行将会出现运行错误。

（2）本题可以到 www.rqnoj.cn 上提交测试（试题编号 13）。

例 6.21 猴子选大王

有 n(1<=n<=100)个猴子需要选出一个大王，选大王的规则是： 先围成一圈，用一个可以随机产生正整数 1 到 12 的骰子，产生一个数 m(1<=m<=12),从编号为 1 的猴子开始报数，报到 m 的出局，这样依次再从后缀位置开始，从剩下的报数，到 m 者出局.....这样剩下的最后一只猴子就是大王。用猴子的编号来描述，猴子所围圈的构成结构是:1,2,3,4...,n-2,n-1,n,1,2,3,...

输入：只有一行两个整数 n 和 m，用一个空格隔开

输出：只有一个数，大王的编号。

分析：已知整数 n 和 m,求大王的编号。

实际上可以理解为出局 n 次，最后一次出局的为大王。

```
For (i=1;i<=n;i++)
{
   从上一次出局猴子的下一只猴子开始报数，报数到 m 的猴子 k 出局;
}
```

那么怎么标记猴子在队列还是在已经出局了呢？我们可设一个数组 flag 来标记猴子是否还在队列中。开始每一只猴子均在队列中，flag[1],…,flag[n]均设为 0，flag[i]==0 表示猴子 i 在队列中，如果猴子 i 出局,则 flag[i]=1

下表演示了出局过程（n=5 m=3）(0 在圈内，1 出局）

	flag [1]	flag [2]	flag [3]	flag [4]	flag [5]
初始	0	0	0	0	0
第 1 次出局	0	0	1	0	0
第 2 次出局	1	0	1	0	0

第 3 次出局	1	0	1	0	1
第 4 次出局	1	1	1	0	1
第 5 次出局	1	1	1	1	1

所以，最后出局的是 4 号猴子。

但表中没有演示报数的过程，实际上出局 1 只猴子后，下一次从刚出局猴子的下一个还未出局的猴子开始报数。

参考程序：

```
#include<stdio.h>
int flag[105]={0};//猴子都在圈内，标志都设为 0，即所有 flag[i]=0
int main()
{
  int n,m,i,k,s;
  scanf("%d%d",&n,&m);
  k=0;  //还没有猴子出局，可以认为上一次是 0 号猴子出局。
  for (i=1;i<=n;i++)  //出 n 次局
  {
     s=0; //本次报数前，初值为 0
     while (s<m)
      {
        k++;//k 是上次报数的猴子编号，所以要 K++
        if (k>n)k=1;// 编号超过 n，接到编号 1
        if (flag[k]==0) s++;//如果编号为 k 的猴子，未出局，则报数
      }
     flag[k]=1;//编号为 k 的猴子报数到 m，出局，标志改为 1
  }
 printf("%d\n",k);
 return 0;
}
```

（1）输入:5 3 输出:4 （2）输入:18 4 输出:9 （3）输入:100 12 输出:81

说明：此题程序可以到 www.ssxdzx.net/bas 上（试题题号 1020）提交测试。

例 6.22 求马鞍数坐标

【题目描述】求一个 5×5 矩阵中的马鞍数，输出它的位置。所谓马鞍数，是指在行上最小而在列上最大的数，其中矩阵中的数最大不超过 32767。

【输入】任意一个 5 行 5 列的矩阵

【输出】马鞍数的数组下标值,若有多个，换行输出

样例输入	样例输出
9 5 4 7 8 3 4 3 2 1 7 6 5 8 9	3 3

6 3 2 9 0 1 2 1 4 8	

分析

（1）输入数存入到数组 a 中。

（2）找出马鞍数并输出。

马鞍数是行上最小列上最大。

```
For (i=1;i<=5;i++)
{
  找出 i 行的最小值所在列 min;
  找出列 min 上的最大值所在的行 max;
  If (max==i) 输出 i 和 min;
}
```

参考程序一

```
#include<stdio.h>
int a[7][7];
int main()
{
  for (int i=1;i<=5;i++)
    for (int j=1;j<=5;j++)
      scanf("%d",&a[i][j]);  //读入数存入数组 a 中
  for (int i=1;i<=5;i++)
  {  //找出 i 行上最小值列下标
    int min=1;  //假设 i 行上的最小值所在列为 1
    for (int j=2;j<=5;j++)
      if (a[i][min]> a[i][j]) min=j; //有更小的，更新 min
    //找出 min 列上的最大值行下标
    int max=1; //假设 min 列上的最大值所在行为 1
    for (int j=2;j<=5;j++)
      if (a[max][min]<=a[j][min]) max=j; //有更大的,更新 max
    if (max==i) printf("%d %d\n",i,min);
  }
  return 0;
}
```

说明：该程序提交只能得到 60－70 的分。出现错误的原因是一行上可能有多个最小值，列上也有可能有多个最大值。参考程序一只考虑了取出每行最前面最小值，列上取出最上面的最大值。这样可能得不到正确的解。如下面的数据：

9 8 5 5 7
5 4 6 4 7
3 1 7 3 4
6 7 8 2 4
3 2 9 1 5

第一行最小值是5，如果取第3列的5就不是马鞍数，而取第4列的5，就是马鞍数。

算法改进如下：

```
For (i=1;i<=5;i++)
{
  找出i行的最小值所在列min;
  用j枚举i行最小值所在的列
  {
    找出列j上的最大值所在的行max;
    用k枚举和列j上最大值相等的行
       If (k==i) 输出i和j;
  }
}
```

参考程序二

```
#include<stdio.h>
int a[7][7];
int main()
{
  for (int i=1;i<=5;i++)
    for (int j=1;j<=5;j++)
      scanf("%d",&a[i][j]);
  for (int i=1;i<=5;i++)
  {
    int min=1;
    for (int j=2;j<=5;j++)
      if (a[i][min]>a[i][j]) min=j;
    for (int j=1;j<=5;j++)  //枚举i行上最小值所在的列
      if (a[i][min]==a[i][j])
      {
        int max=1;
        for (int k=2;k<=5;k++) //找出j列上的最大值
          if (a[k][j]>a[max][j]) max=k;
        for (int k=1;k<=5;k++) //枚举j列上最大值
          if (a[k][j]==a[max][j])
            if (k==i) printf("%d %d\n",k,j);
      }
  }
  return 0;
}
```

说明：

（1）参考程序一在www.ssxdzx.net/bas上（试题题号1031）提交测试60分，参考程序二提交100分。

（2）通过这个问题的解决得到启示提示：考虑问题一定要全面，各种特殊情况都要思考清楚。

矩阵与矩阵运算

一、矩阵

矩阵，是由m×n个数组成的一个m行n列的矩形表格，通常用大写字母A,B,C,…表示，组成矩阵的每一个数，均称为矩阵的元素，通常用小写字母a_{ij},b_{kl},c_{pq}表示元素，其中下标i、j、k、l、p、q都是正整数，他们表示该元素在矩阵中的位置。$A=(a_{ij})_{m\times n}$ 表示一个m×n矩阵，下标表示元素a_{ij}位于该矩阵的第i行、第j列。元素全为零的矩阵称为零矩阵。

特别地，一个 m×1矩阵，也称为一个 m维列向量；而一个1×n矩阵，也称为一个 n维行向量。

$$A=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \qquad B=\begin{pmatrix} a_1 \\ a_2 \\ \cdots \\ a_m \end{pmatrix} \qquad C=(a_1,a_2,...,a_n)$$

m×n矩阵A　　m×1矩阵B　　1×n矩阵C

当一个矩阵的行数m与列数n相等时，该矩阵称为一个 m阶方阵。对于方阵，从左上角到右下角的连线，称为主对角线；而从左下角到右上角的连线称为副对角线。若一个 n阶方阵的主对角线上的元素都是 1，而其余元素都是零，则称为单位矩阵，记为E_n。如一个n 阶方阵的主对角线上（下）方的元素都是零，则称为下（上）三角矩阵。

$$E_n=\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \ddots & \cdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}_{n\times n} \qquad A=\begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \qquad B=\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ 0 & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & b_{mm} \end{pmatrix}$$

单位矩阵E_n　　n 阶下三角矩阵A　　m阶上三角矩阵B

二、矩阵的运算

1、矩阵的加法 ： 如果$A=(a_{ij})$、$B=(b_{ij})$是两个同型矩阵（即它们具有相同的行数和列数，则定义它们的和C=A+B仍为与它们同型的矩阵（矩阵C的元素为A 和B对应元素的和，即$c_{ij}=a_{ij}+b_{ij}$）。

给定矩阵$A=(a_{ij})$，我们定义其负矩阵$-A=(-a_{ij})$。这样我们可以定义同型矩阵 A,B 的减法为:$A-B=A+(-B)$。由于矩阵的加法运算归结为其元素的加法运算，容易验证，矩阵的加法满足下列运算律：

（ 1）交换律： A+B=B+A;

（ 2）结合律： A+(B+C)=(A+B)+C;

（ 3）存在零元： A+0=0+A=A;

（ 4）存在负元： A+(-A)=A-A=0。

2 、数与矩阵的乘法

设λ 是一个数，$\lambda A=(\lambda a_{ij})$，由定义可知：(-1)A=-a。容易验证数与矩阵的乘法满足下列运算律：

（1 ）1A=A ；

（2 ）λ (A+B)= λ A+λ B;

（3 ）(λ +μ)A=λ A+μ A;

（4 ）(λ μ)A=λ (μ A)=μ (λ A)= (λ A) μ =(μ A)λ ；

3 、矩阵的乘法

设 $A=(a_{ij})$ 为 m×n 矩阵， $B=(b_{ij})$ 为 n×p 矩阵，则矩阵 A 可以左乘矩阵 B（注意：矩阵 A 列数与矩阵 B 的行数相等），所得的积为一个 m×p 矩阵 C ，即 C=AB，其中 $C=(c_{ij})$ ，并且

$$c_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+...+a_{in}b_{nj}=\sum_{k=1}^{n}a_{ik}b_{kj}$$

矩阵的乘法满足下列运算律（假定下面的运算均有意义）：

（ 1）结合律：(AB)C=A(BC)

（ 2）左分配律：A(B+C)=AB+AC；

（ 3）右分配律：(B+C)A=BA+CA；

（ 4）数与矩阵乘法的结合律：(λ A)BC=A(λ B)C=AB(λ C)；

（ 5）单位元的存在性： $E_mA_{m\times n}=A_{m\times n}$ ， $A_{m\times n}E_n=A_{m\times n}$ 。

若 A 为 n 阶方阵，则对任意正整数 k，我们定义： $A^k=\underbrace{AAA\cdots A}_{k个}$ ，并规定： $A^0=E$，由于矩阵乘法满足结合律，我们有： $A^kA^l=A^{k+l}$， $(A^k)^l=A^{kl}$ 。

注意： 矩阵的乘法与通常数的乘法有很大区别，特别应该注意的是：

（1 ）矩阵乘法不满足交换律：一般来讲即便 AB 有意义，BA 也未必有意义；倘使 AB、BA 都有意义，二者也未必相等（请读者自己举反例）。正是由于这个原因，一般来讲，

$(A+B)^2\neq A^2+2AB+B^2$， $(AB)^k\neq A^kB^k$ 。

（2 ）两个非零矩阵的乘积可能是零矩阵，即 AB=0 未必能推出 A=0 或者 B=0 （请读者自己举反例）。

（3 ）消去律不成立：如果 AB=BC 并且 A≠0 ，未必有 B=C 。

例 6.23 矩阵乘法

两矩阵相乘得到的是一个新的矩阵，新矩阵的第 i 行第 j 列的元素是第一个矩阵的第 i 行与第二个矩阵的第 j 列对应元素乘积的和。

矩阵的行数与列数均不超过 100，输入的矩阵和运算得到的矩阵元素为整数,其值不超过 2000000000。

输入：第一行为两个整数 M、N、k（1<=M、N、k<=100），之间用一个空格隔开。接下来依次为 2 个矩阵，每两个矩阵之间空一行。第一个矩阵为 M 行 N 列，第二个矩阵为 N 行 K 列。

每个矩阵格式：每行对应矩阵的一行，数之间用一个空格隔开。

输出：为运算得到的矩阵，每个整数之间用一个空格隔开。

input	output
2 3 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	3 3 3 3 3 3 3 3

分析：给两个矩阵 $A=(a_{ij})_{m\times n}$ 和 $B=(b_{ij})_{n\times k}$ ，求这两个矩阵的乘积 $C=(c_{ij})_{m\times k}$ 。

$c_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+...+a_{in}b_{nj}=\sum_{p=1}^{n}a_{ip}b_{pj}$

于是得到算法：

（1）输入矩阵 $A=(a_{ij})_{m\times n}$ 和 $B=(b_{ij})_{n\times k}$

（2）计算矩阵 $C=(c_{ij})_{m\times k}$

（3）输出矩阵 $C=(c_{ij})_{m\times k}$

细化（1）输入矩阵 $A=(a_{ij})_{m\times n}$ 和 $B=(b_{ij})_{n\times k}$：

①矩阵的A是m行n列，只能一行行的输入

```
for (int i=1;i<=m;i++)
  输入第i行的数据;
```

②进一步细化输入第i行的数据:第一行有n个整数，只能一个一个的输入

```
for (int j=1;j<=n;j++)
   输入a[i][j];
```

于是得到输入矩阵A的代码：

```
for (int i=1;i<=m;i++)
   for (int j=1;j<=n;j++)
      输入a[i][j];
```

同样可输入矩阵B。

细化（2）计算矩阵 $C=(c_{ij})_{m\times k}$：

①矩阵的C是m行k列，显然是一行一行的计算

```
for (int i=1;i<=m;i++)
     计算第i行的各元素;
```

②进一步细化计算第i行的各元素，只能一个一个计算;

```
for (int j=1;j<=k;j++)
   计算c[i][j];
```

③再细化计算c[i][j]，c[i][j]是A的第i行元素与B的第j列对应元素乘积的和，是p个积的和，当然用累加;

```
    for (int p=1;p<=n;p++)
     c[i][j]+=a[i][p]*b[p][j]
```

于是得到计算矩阵 $C=(c_{ij})_{m\times k}$ 的代码

```
for (int i=1;i<=m;i++)
  for (int j=1;j<=k;j++)
    for (int p=1;p<=n;p++)
     c[i][j]+=a[i][p]*b[p][j]
```

细化（3））输出矩阵:$C=(c_{ij})_{m\times k}$

①矩阵的C是m行k列，输出只能是一行行的进行，输出一行以后还应换行。

```
for (int i=1;i<=n;i<=m)
 {
  输出第i行的数据;
  换行;
```

```
 }
```

②细化输出第 i 行的数据，由于每一行有 k 个数据，只能一个一个输出。

```
for (int j=1;j<=k;j++)
  输出 c[i][j];
```

于是可以得到输出矩阵 $C=(c_{ij})_{m\times k}$ 的代码：

```
for (int i=1;i<=n;i<=m)
 {
  for (int j=1;j<=k;j++)
    输出 c[i][j];
  换行;
 }
```

参考程序

```
#include<stdio.h>
int a[105][105],b[105][105],c[105][105],m,n,k;
int main()
{
  scanf("%d%d%d",&m,&n,&k);
  for (int i=1;i<=m;i++)//输入矩阵 A
    for (int j=1;j<=n;j++)
      scanf("%d",&a[i][j]);
  for (int i=1;i<=n;i++)//输入矩阵 B
    for (int j=1;j<=k;j++)
      scanf("%d",&b[i][j]);
  for (int i=1;i<=m;i++)//计算矩阵 C
    for (int j=1;j<=k;j++)
      for (int p=1;p<=n;p++)
        c[i][j]+=a[i][p]*b[p][j];
  for (int i=1;i<=m;i++)//输出矩阵 C
   {
    for (int j=1;j<=k;j++)
      printf("%d ",c[i][j]);
    printf("\n");
   }
return 0;
}
```

说明：

（1）矩阵就是一个二维数组；

（2）A 和 B 是同类型矩阵，C=A+B，$c_{ij}=a_{ij}+b_{ij}$，C 与 A、B 也是同类型矩阵。

（3）矩阵 A 的列数与矩阵 B 的行数相等，A、B 才相乘。$A=(a_{ij})_{m\times n}$、$B=(b_{ij})_{n\times k}$，$C=AB=(c_{ij})_{m\times k}$。

$$c_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+...+a_{in}b_{nj}=\sum_{p=1}^{n}a_{ip}b_{pj}$$

即 C_{ij} 是矩阵A的第i行与矩阵B的第j列对应元素的乘积。注意：矩阵乘法只满足结合律，不满足交换律。

习 题

6.1 输入n(<=100)个整数，找出最小数所在的位置，并把它与第一个数对调。

6.2 求除1到m(<=2.1×10^7)之内(含m)能被7或11整除的所有整数放在数组a中。

6.3 输入一串字符（字符个数不超过1000），除首尾字符外，将其个字符按ASCII码降序排列。

6.4 输入一串字符（字符个数不超过1000），将所有奇数位置(第1个字符位置记为1)上的字母转换为大写(若该位置上不是字母，则不转换)。

6.5 输入一行字符密码，编程输出原文。

原文译成密码的规则如下：

A→Z　a→z

B→Y　b→y

C→X　c→x

…　　…

即第1个字母变成第26个字母，第i个字母变成第（26-i+1）个字母，非字母字符不变。

6.6 三角形的个数(www.ssxdzx.net/bas,试题编号1016)

【题目描述】给出n(3<=n<=100)条长度完全不同的线段,从中任选3条,问能组成多少个不同的三角形?

【输入】第一行,线段的条数n;

第二行,n个数,即n条线段的长度(不超过10000),各个数之间用一个空格隔开。

【输出】一行，组成三角形的个数

【输入样例】	【输出样例】
4 1 2 3 4	1

6.7 小辉的决心(www.ssxdzx.net/bas,试题编号1019)

【题目描述】小辉同学是一位非常热爱学习的好同学，但英语成绩总不理想，经常比他的同桌低二三十分。他苦读英语，渴望在半期英语考试中取得高分，他的梦想能实现吗？我们拭目以待。

英语试卷上共有n道大题，每道大题有5个小题，每道小题都有A,B,C,D四个选项。每小题2分。你作为他的同桌兼精神领袖，请在考试后得到标准答案后，告诉他他能得多少分，以及你会比他高多少分。

满分是不定的，每次需要自行计算（事实表明，你的答案每次都与标准答案完全相同）。

【输入】第一行一个整数n(1<=n<=10),代表有n个大题，以下2到n+1行：每行前五个是他的答案，接下来5个是你的标准答案，中间用空格格开。

【输出】两行，第一行是他的得分，第二行是你比他多的分数。

【输入样例】	【输出样例】
4 ABCDA　DACDA	6 34

DABDA CDABC BABDA CCCCC CCABD BBCCA	

6.8 成绩处理(www.ssxdzx.net/bas,试题编号 1018)

【题目描述】给出 n(1<=n<200)个同学 m(1<=m<=5)科的考试成绩,按总分找出前十名是哪些同学。我们规定：总分相同名次相同，且占用后面的名次，例如：第 2 名有两个同学，那么将没有第 3 名，后面的同学将从第 4 名算起。

【输入】第一行为 n 和 m,下面 n 行为每个同学的成绩描述： 每行有 m+1 个数，其中第一个数为学号（小于等于 200），后面为各科成绩，中间用一个空格隔开。

【输出】一行，成绩名次在前 10 的同学学号，不足 10 名全部输出。当第 10 名有多个同学时，输出有可能超过 10 名同学的学号。同名次按学号由小到大输出。输出的各个学号之间用一个空格隔开。

【输入样例】	【输出样例】
17 2 1 7 13 2 18 54 3 47 91 4 65 86 5 44 48 6 92 88 7 80 36 8 76 68 9 54 21 10 65 11 11 61 69 12 78 32 13 0 81 24 93 32 15 42 97 16 28 9 17 65 55	6 4 8 15 3 11 24 17 7 12

6.9 连接最大数(www.ssxdzx.net/bas,试题编号 1033)

【题目描述】给出 n(1<=n<=500)个数,把它们连接成一个数 s,求最大的 s。

【输入】第一行为 n;后面 n 行为不超过 10^9 的正整数。

【输出】输出连接后的最大数 s。

【输入样例】	【输出样例】
4 1 2	4321

4 3	

6.10　矩陈加法

两矩阵相加得到的是一个新的矩阵，新矩阵的每个元素是其对应的两矩阵元素的和。

矩阵的行数与列数均不超过 100，输入的矩阵和运算得到的矩阵元素为整数,其值不超过2000000000。

输入：第一行为两个整数 M、N（1<=M、N<=100）之间用一个空格隔开。接下来为 M 行,每行 N 个数,数之间用一个空格隔开。依次为 2 个矩阵，每两个矩阵之间空一行。

输出：为运算得到的两个矩阵的和，每个整数之间用一个空格隔开。

input	output
2 3 1 1 1 1 1 1 1 1 1 1 1 1	2 2 2 2 2 2

赫伯特•亚历山大•西蒙

赫伯特•亚历山大•西蒙(Herbert Alexander Simon)是20世纪科学界的一位奇特的通才,在众多的领域深刻地影响着我们这个世代。他学识渊博、兴趣广泛，研究工作涉及经济学、政治学、管理学、社会学、心理学、运筹学、计算机科学、认知科学、人工智能等广大领域,并做出了创造性贡献，在国际上获得了诸多特殊荣誉。

第7章　函数

在前面各章的程序中都只有一个主函数main()， 但实用程序往往由多个函数组成。函数是C程序的基本模块， 通过对函数模块的调用实现特定的功能。用户可把自己的算法编成一个个相对独立的函数模块，然后用调用的方法来使用函数。

例7.1 输出如下的图案

```
++++++++++++++++++++++++++++++++++++++++
               I love games!
++++++++++++++++++++++++++++++++++++++++
```

可以写成如下程序。

```
#include<stdio.h>
void print()  //自定义函数print()
{
  printf("++++++++++++++++++++++++++++++++++++++++\n");
}
int main()
{
   print();//调用函数print()
   printf("               I love games!\n");
   print();//调用函数print()
   return 0;
}
```

说明：

（1）void print()自定义了一个函数。这个函数的作用：输出一行+号；

（2）如果函数定义放在调用该函数位置之前，不须进行说明。如果在调用位置之前没有进行定义，则需要进行声明。

```
#include<stdio.h>
int main()
{
   void print(); //还没有定义，进行声明后面进行定义
   print();//调用函数print()
   printf("               I love games!\n");
   print();//调用函数print()
   return 0;
}
void print()  //自定义函数print()
{
  printf("++++++++++++++++++++++++++++++++++++++++\n");
}
```

（3）定义函数的一般形式

类型名 函数名(形式参数表)

{

函数体

}

类型名：指函数返回值类型，即调用函数后得到的结果。返回值类型可以是整数、实数、字符和结构体，也可以没有返回值（其类型用 void 表示）。

调用 print()后，只是输出一行“+”号，没有返回值，所以定义该函数：void print()。

下面定义一个有返回值的函数。定义如下：

```
int max(int x,int y)
{
  int z;
  if (z>y) z=x;
  else z=y;
  return z;
}
```

大家应该很熟悉上面的语句：z 取 x 和 y 中的最大值。

return z;就是把 z 的值返回给函数 max。

该函数的功能是求出两个正整数的最大值。

因为两个正整数的最大值是整数，所以该函数返回值类型为 int。

根据函数有无返回值来划分，函数可分为有返回值函数和无返回值函数。void print()是无返回值函数，int max(int x,int y)是有返回值函数。

函数返回值

1)函数的返回值只能通过 return 语句获得的。

return 语句的一般形式为：

return 表达式； 或者为 return (表达式);

该语句的功能是计算表达式的值，并返回给主调函数。 在函数中允许有多个 return 语句，但每次调用只能有一个 return 语句被执行， 因此只能返回一个函数值。

上述定义的 int max(int x,int y)函数中 return z;也可写成 return (z);

2)类型名和 return 后面表达式值的类型应保持一致。 如果两者不一致，则以函数类型为准，自动进行类型转换。

如上述定义的 int max(int x,int y)函数，类型名和 z 的类型均为 int。

函数名：函数的名字，如上面定义的函数名是 print 和 max

形式参数表：有的函数需要参数，有的不需要参数。函数可分为有参函数和无参函数。

void print()作用是输出一行+号，所以不需要参数，但（）不能省略。

int max(int x,int y)作用求出两个整数的最大值。所以需要把这两个数通过参加带进函数进行处理。

形式参数表：各参数分别定义类型，各参数之间用“,”隔开。

例 7.2 输入两个整数，输出这两个数的最大值（要求自定义函数完成）。

分析：已知两个整数 a、b,输出 a、b 中的最大值。

参考程序一

```
#include<stdio.h>
int max(int x,int y)//函数求出x,y的最大值
{
  int z;
  if (x>y) z=x;
  else z=y;
  return z;//返回最大值
}
int main()
{
   int a,b,c;
   scanf("%d%d",&a,&b);
   c=max(a,b);
   printf("%d",c);
   return 0;
}
```

参考程序二

```
#include<stdio.h>
void printmax(int x,int y)//输出x,y的最大值
{
  int z;
  if (x>y) z=x;
  else z=y;
  printf("%d",z);
}
int main()
{
   int a,b;
   scanf("%d%d",&a,&b);
   printmax(a,b);
   return 0;
}
```

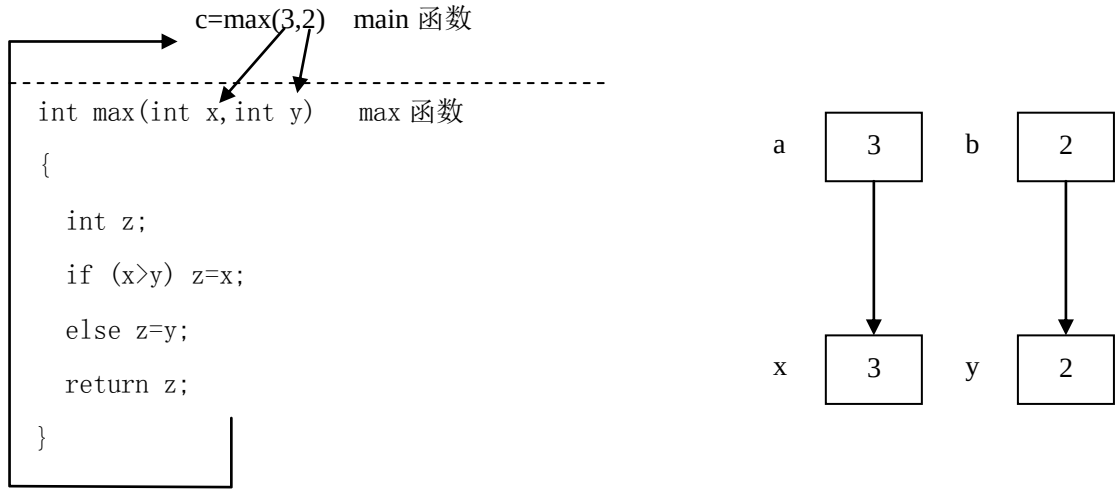
输入 3 2 输出 3

说明：

（1）int max(int x,int y)是有参有返回值函数，void printmax(int x,int y)是有参无返回值函数。

请对比主程序中调用两个函数的情形。一般来说，有返回值函数作为表达式或表达式的一部分来使用,如“c=max(a,b);”，无返回值函数作为语句来使用，如“printmax(a,b);”。

（2）int max(int x,int y)和void printmax(int x,int y)中的int x和int y称为形式参数（简称形参）或（虚拟参数）。调用函数时使用的参数称为实际参数（简称实参）。函数调用时，实际参数把值传递给形式参数，形式参数的值只在函数内部有效。如下图函数的调用过程。



实参向形参值传递过程和 max(3,2)函数调用过程

假设 a=5。

执行 c=max(3,2); c 的值是 3。

执行 printmax(3+a,9); 输出 9。

执行 d=max(3,a*4); d 的值是 20。

函数的形参和实参具有以下特点：

1）形参变量只有在被调用时才分配内存单元，在调用结束时， 即刻释放所分配的内存单元。因此，形参只有在函数内部有效。 函数调用结束返回主调函数后则不能再使用该形参变量。

2）实参可以是常量、变量、表达式、函数等， 无论实参是何种类型的量，在进行函数调用时，它们都必须具有确定的值， 以便把这些值传送给形参。 因此应预先用赋值，输入等办法使实参获得确定值。

3）实参和形参在数量上，类型上，顺序上应严格一致， 否则会发生“类型不匹配”的错误。

3）函数调用中发生的数据传送是单向的。 即只能把实参的值传送给形参（称为“值传递”），而不能把形参的值反向地传送给实参。 因此在函数调用过程中，形参的值发生改变，而实参中的值不会变化。下面和程序可以说明这个问题。

```
#include<stdio.h>
void test(int x)
{
  x++;//x 改变不会影响实参 a 的值
  printf("%d ",x);
}
int main()
{
   int a=10;
   test(a);
   printf("%d",a);
   getchar();
   return 0;
}
```

运行结果 11 10

例 7.3 输入两个正整数 a,b，输出这两个正整数之间的素数（a<=b）。

分析：已知 a,b,输出 a,b 之间的素数。

用 i 枚举 a--b 的所有数，if i 是素数则输出 i。

```
for (i=a;i<=b;i++)
  if i 是素数则输出 i ;
```

我们把“判断 i 是否素数”定义成函数。

参考程序

```
#include<stdio.h>
int prime(int n)  //定义函数 prime()
{
  for (int i=2;i*i<=n;i++)
    if (n%i==0) return 0; //有因子 返回 0
  return 1; //没有因子 返回 1
}
int main()
{
  int a,b;
  scanf("%d%d",&a,&b);
  for (int i=a;i<=b;i++)
    if (prime(i)) printf("%d ",i);//i 是素数输出 i
  return 0;
}
```

输入 20 50 输出 23 29 31 37 41 43 47

说明：

（1）本程序中把判断 i 是否为素数定义成一个函数，在主程序中调用。int main()中就简洁得多了。

（2）一般来说，使用了函数，使程序的层次结构清晰，便于程序的编写、阅读、调试。

例 7.4 输入整数 n（n<=10000），表示接下来将会输入 n 个实数，将这 n 个实数存入数组 a 中。请定义一个数组拷贝函数将数组 a 中的 n 个数拷贝到数组 b 中。

分析：

（1）输入 n,再输入 n 个实数存入数组 a 中

（2）将数组 a 中的 n 个数拷贝到数组 b 中

（3）输出输出 b 中的 n 个数

将（2）定义成函数 arraycopy:

这里只是执行拷贝，所以定义无返回值函数（void）;

arraycopy 的功能是要将一个数组中的多少个数拷贝另一个数组中,这里会涉及到三个形参，设为数组 c、数组 d、整数 m。

定义的 arraycopy 函数见参考程序中。

参考程序

```
#include<stdio.h>
double a[10005],b[10005];
```

```
void arraycopy(double c[],double d[],int m)
{
   for (int i=1;i<=m;i++)
     d[i]=c[i];
}
int main()
{
   int n;
   scanf("%d",&n);
   for (int i=1;i<=n;i++)
     scanf("%lf",&a[i]);
   arraycopy(a,b,n);
   for (int i=1;i<=n;i++)
     printf("%lf ",b[i]);
   return 0;
}
```

输入　4 1.2 5.3 9.456 3.72

输出　1.200000 5.300000 9.456000 3.720000

说明：

（1）定义函数时，数组名作为形参可以不指定数组大小；

（2）在用数组名作函数参数时，不是“值传递”。因为实际上形参数组并不存在，编译系统不为形参数组分配内存。数组名作函数参数时所进行的传递是“地址传递”， 也就是说把实参数组的首地址赋予形参数组名。形参数组名取得该首地址之后，也就等于有了实在的数组。实际上是形参数组和实参数组为同一数组，共同拥有一段内存空间。

（3）当形参数组发生变化时，实参数组也随之变化（参看下面的程序）。因为实际上形参数组和实参数组为同一数组。前面讲的实参变量与形参变量之间的传递是“值传递”，实参变量和形参变量在内存中分配不同的存储单元。

```
#include<stdio.h>
int a[15];
void test(int c[])
{
  for (int i=1;i<=4;i++)
    c[i]=c[i]*10;
}
int main()
{

   for (int i=1;i<=10;i++) a[i]=i;
   test(a);
   for (int i=1;i<=10;i++)
     printf("%d ",a[i]);
   return 0;
```

```
}
```

运行结果：10 20 30 40 5 6 7 8 9 10

例 7.5 输入长方体的长宽高，求体积和全面积（要求用一个函数求出体积和全面积）。

分析：前面已经说明一个函数只有一个返回值，但要求用一个函数求出两个值体积和全面积。这里借助全局变量从函数 vs( )中把全面积带出来。

参考程序

```
#include<stdio.h>
double s;
double vs( double a,double b,double c)
{
  double v;
  v=a*b*c;
  s=2*(a*b+a*c+b*c);
  return v;
}
int main()
{
    double c,k,g,v;
     scanf("%lf%lf%lf",&c,&k,&g);
     v= vs(c,k,g);
    printf("v=%0.2lf  s=%0.2lf\n",v,s);
    return 0;
}
```

局部变量 v 的作用域

局部变量 a,b,c 的作用域

局部变量 c,k g,v 的作用域

全局变量 s 的作用域

输入 1 1 1　　输出 v=1.00 s=6.00

说明：

（1）局部变量

①在函数内部定义的变量称为局部变量；

②局部变量的作用域即在函数内从定义该变量位置起到函数结束止。在定义这个变量的函数之外是不起作用的；

③复合语句内定义的变量也是局部变量，作用域是复合语句内部；

```
if (a>b)
{
  int t=a;
  a=b;
  b=t;
}
```

局部变量 t 作用域

④ 如果 for()语句中循环变量是在语句内部定义的，其作用域就是该循环；

```
for (int i=1;i<=100;i++)
{
  ……
}
```

局部变量 i
作用域

⑤形参变量是局部变量。double vs(double a,double b,double c)，形参变量 a、b、c 是局部变量。

（2）全局变量

①在函数外部定义的变量称为全局变量（又称外部变量或全程变量）；

②全局变量的作用域从定义该变量位置起到本文件结束止；

③C 语言规定函数返回值只有一个， 当需要增加函数的返回数据时，用全局变量是一种很好的方式。

本例中使用了全局变量 s， 在函数 vs 中求得的 s 值在 main 中仍然有效。因此外部变量是实现函数之间数据通讯的有效手段。

④ 在同一源文件中，允许全局变量和局部变量同名。在局部变量的作用域内，全局变量不起作用。全局变量和局部变量同名时，实际上局部变量和全局变量是不同的变量。

例 7.6 输入正整数 n、m、k(n、m、k<=10000),计算下列表达式的值。

$$\sum_{i=1}^{n} i + \sum_{i=1}^{m} i + \sum_{i=1}^{k} i$$

分析：$\sum_{i=1}^{n} i$ =1+2+3+…+n

算法如下

（1）输入正整数 n、m、k,

（2）求 s= $\sum_{i=1}^{n} i + \sum_{i=1}^{m} i + \sum_{i=1}^{k} i$

（3）输出 s

细化（2）：

1)分别求出 $\sum_{i=1}^{n} i$、$\sum_{i=1}^{m} i$、$\sum_{i=1}^{k} i$

2）求三个之各和 s

参考程序

```
#include<stdio.h>
int n,m,k;//全局变量 n,m,k，作用域是整个程序文件。
//int sum(int n)求 1+2+3+...+n
int sum(int n) //局部变量 n，作用域是 sum( )内
```

```
{
  int s=0; //局部变量 s
  for (int i=1;i<=n;i++) //局部变量 i
    s+=i;
  return s;
}
// int totsum(int n,int m,int k)求三个和的和
int totsum(int n,int m,int k) //局部变量 n,m,k
{
  return sum(n)+sum(m)+sum(k);
}
int main()
{
  scanf("%d%d%d",&n,&m,&k);
  int s=totsum(n,m,k);//和会超过 2.1×10^9 吗? s 局部变量
  printf("%d",s);
  return 0;
}
```

（1）输入　1 2 3　　输出　10

（2）输入　10000 1000 100　　输出　50510550

说明

（1）int main()内就是三步：输入、处理、输出。用了函数以后，可以把每一种功能定义成一个函数，在 int main()来调用。这样让 int main()比较“干净”，各部分功能非常明确。

（2）在 C 语言中，所有的函数定义，包括主函数 main 在内，都是平行的。也就是说，在一个函数的函数体内， 不能再定义另一个函数， 即不能嵌套定义。上述程序中 main()、sum()、totsum()都是平行的。

（3）函数的嵌套调用:一个函数调用另一个函数称为函数的嵌套调用。习惯上把调用者称为主调函数。函数 totsum()调用了函数 sum()。

main 函数是主函数，它可以调用其它函数，而不允许被其它函数调用。 因此，C 程序的执行总是从 main 函数开始， 完成对其它函数的调用后再返回到 main 函数，最后由 main 函数结束整个程序。一个 C 程序必须有，也只能有一个主函数 main。

（4）$\sum_{i=1}^{n} i = 1+2+3+\cdots+n$ ，$\prod_{i=1}^{n} i = 1\times 2\times\cdots\times(n-1)\times n = n!$

（5）参考程序中局部变量和全局变量 m、n、k 作用域各不相同。全局变量 m、n、k 的作用域是整个程序文件;函数 int totsum(int n,int m,int k)中局部变量 m、n、k 的作用域是 totsum(int n,int m,int k)内部，在 int totsum(int n,int m,int k)调用的 int sum( )内不起作用。

例 7.7　输入 4 个整数，找出最大的数。（用函数的嵌套调用来处理）

分析：此问题并不复杂，完全可以由 int main()处理。但题意要求用函数嵌套调用来处理。

定义函数 int max2(int a,int b)求出 a,b 的最大值

定义函数 int max4(int a,int b,intc,int d)求出 a,b,c,d 的最大值。

参考程序

```
#include<stdio.h>
//int max2(int a,int b)求a,b中的大值
int max2(int a,int b)
{
  return a>b?a:b;  //条件表达式
}
int max4(int a,int b,int c,int d)//求四个数的最大值
{
  return max2(max2(a,b),max2(c,d));
}
int main()
{
  int a,b,c,d,m;
  scanf("%d%d%d%d",&a,&b,&c,&d);
  m=max4(a,b,c,d);
  printf("%d",m);
  return 0;
}
```

输入 10 30 20 40　　输出　40

说明：max4中“return max2(max2(a,b),max2(c,d));”语句，也可改成return max2(max2(max2(a,b),c),d)。

例7.8　进制转换

现给定一个十进制整数N（＜＝2100000000），要求转换成M(2=<M<=36)进制。如果M超过10，用A、B...Z，分别表示数码10、11...35。

输入文件：仅两行，第一行为N，第二行为M。

输出文件：仅为一行，M进制数。

样例

input 291 16　output　123

分析：此题的算法

（1）输入整数n和m；

（2）将n转换成m进制数；

（3）输出m进制数。

关键问题：十进制n转换成m进制。

十进制n转换成m进制的方法：**除以m取余法**。

十进制制数291转换成十六进制的过程如下：

291÷16=18…3

18÷16=1…2

1÷16=0…1

∴$(291)_{10}=(123)_{16}$

除以M取余法：把十进制数N除以M,商再除以M，商再除以M，…，当商为0时不再进行。每次得到的余数的反序就是M进制数。

$(123)_{16}=1\times16^2+2\times16^1+3\times16^0=(291)_{10}$

看到上面的式子就容易理解转换的方法了。

第1次：用$1\times16^2+2\times16^1+3\times16^0$除以16,余数就是十六进制的个位3;

第2次：商$1\times16^1+2\times16^0$除以16,余数就是十六制数的十位2；

第3次：商1×16^0除以16,余数就是十六制数的百位1；

这是商为0，不再进行了。

下面我们来细化算法中的（2）步。

我们余数最多几位呢？首行n不超过2100000000，也就是说不会超过2^{31}，而转换成2进制位数最多，余数不会超过31位。定义一个int a[35]足够了。

```
While (n!=0)
{
  a[++t]=n%m;
  n=n/m;
}
```

参考程序

```
#include<stdio.h>
int a[35],n,m,t=0;//t=0,开始位数为0
//exchange()将n转换成m进制数，结果存在数组a中
void exchange(int a[],int n,int m)
{
   while (n!=0)
   {
     a[++t]=n%m;
     n=n/m;
   }
}
//print()将数组a中的t位，反序输出
void print(int a[],int t)
{
  for (int i=t;i>=1;i--)//反序输出
    if (a[i]>=10) printf("%c",a[i]+55);//超过9,字母表示
    else printf("%d",a[i]);
}
int main()
{
  scanf("%d%d",&n,&m);
  exchange(a,n,m);
  if (t==0) t++;//如果是0位，说明n=0，也得有1位。
  print(a,t);
  return 0;
```

```
}
```

（1）输入 124565445 16　　输出 76CB7C5

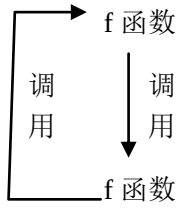
（2）输入 2000000000 36　　输出 X2QXVK

说明：十进制数转换成M进制数的方法，大家一定要熟悉。

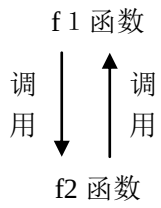
函数的递归调用

一个函数调用中出现直接或间接调用该函数自身称为递归调用。 这种函数称为递归函数。C 语言允许函数的递归调用。在递归调用中， 主调函数又是被调函数。执行递归函数将反复调用其自身。每调用一次就进入新的一层。例如有函数 f,f1,f2 如下：

```
int f (int x)
{
int y,z;
z=f(y);//执行 f 函数的过程中又调用 f 函数。
return z;
}
int f1 (int x)
{
int y,z;
z=f2(y);//执行 f1 函数的过程中又调用 f2 函数。
return z;
}
int f2 (int x)
{
int y,z;
z=f1(y);//执行 f2 函数的过程中又调用 f1 函数。
return z;
}
```



直接调用



间接调用

在调用函数 f 的过程中，又调用 f 函数，这是直接调用函数。

在调用函数 f 1 的过程中，又调用 f2 函数，而调用 f 2 函数过程中又调用函数 f1 本身，就是间接调用 f1。

这个三个函数是一个递归函数。 但是运行该函数将无休止地调用其自身，这当然是不正确的。为了防止递归调用无终止地进行， 必须在函数内有终止递归调用的手段。常用的办法是加条件判断，满足某种条件后就不再作递归调用，然后逐层返回。

关于递归的概念，初学者不好理解，下面举例说明递归调用的执行过程。

例 7.9 有 5 人学生坐成一排。问第 5 个学生多少岁，他说比第 4 个学生大 2 岁。问第 4 个学生多少岁，他说比第 3 个学生大 2 岁。问第 3 个学生多少岁，他说比第 2 个学生大 2 岁。问第 2 个学生多少岁，他说比第 1 个学生大 2 岁。问第 1 个学生多少岁，他说 10 岁。问第 5 个学生多少岁？

分析：要知道第 5 个学生多大，就必须知道第 4 个学生的年龄；要知道第 4 个学生多大，必须知道第 3 个学生的年龄；要知道第 3 个学生多大，就必须知道第 2 个学生的年龄；要知道第 2 个学生多大，就必须知道第 1 个学生的年龄。而第 1 个学生年龄为 10 岁。设第 n 个学生的年龄为 age(n)，于是有

age(5)=age(4)+2

age(4)=age(3)+2

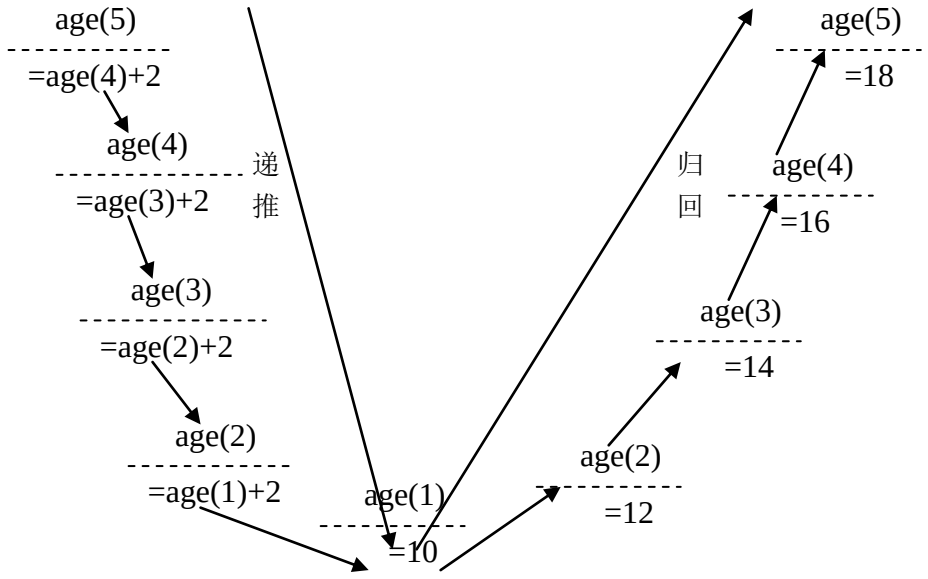
age(3)=age(2)+2

age(2)=age(1)+2

age(2)=10

n>1 时，age(n)的公式一样，都是 age(n)=age(n-1)+2。所以，可用下式表示：

$$age(n)=\begin{cases}10.....n=1\\ age(n-1)+2.....n>1\end{cases}$$



由上图可知，求解过程分为两个阶段：

第 1 阶段是“递推”。要求 age(5)=age(4)+2, age(4)=age(3)+2, …, 直到 age(1)=10（已知）不必再向下推了了。

第 2 阶段是“回归”，age(2)=10+2=12, age(3)=12+2=14, …, age(5)=16+2=18。

至此，整个过程完成，得到结果。

参考程序

```
#include<stdio.h>
int age(int n)  //定义递归函数
{
  if (n==1) return 10;//边界，直接得结果
  else return age(n-1)+2;//age(n)调用 age(n-1)
```

```
}
int main()
{
  printf("%d",age(5));
  getchar();
  getchar();
  return 0;
}
```

输出 18

说明：

（1）int age(int n)函数中可以不写 else,如下定义

```
int age(int n)  //定义递归函数
{
  if (n==1) return 10;//边界，直接得结果
  return age(n-1)+2;//age(n)调用 age(n-1)
}
```

这是因为 n==1 已经返回，不会执行下面的语句“return age(n-1)+2;”，而 n≠1 时才能执行下面的语句。

（2）递归应该满足两个条件：

①具备边界条件，即能直接得出结果（递归的终止条件）；

②能把原问题的规模变小，变成原问题的子问题。

例 7.10 输入整数 n(>=0),用递归法计算 n!。

分析：n!=1×2×…×(n-1)×n,规定 0！＝1

则有 $n!=\begin{cases}1.....0\\(n-1)!\times n.....n>0\end{cases}$

满足递归的两个条件：①边界条件 n=0 时，结果为 1；②n!=(n-1)！×n，规模变小了，是原问题的子问题。

参考程序

```
#include<stdio.h>
int fac(int n)  //定义递归函数
{
  if (n==0) return 1;//边界，直接得结果
  else return fac(n-1)*n;//fac(n)调用 fac(n-1)
}
int main()
{ int n;
  scanf("%d",&n);
  printf("%d",fac(n));
  return 0;
}
```

输入 11 输出 39916800

说明：

（1）本题也可以不用递归的方法来完成。如可以用递推法，即从 1 开始乘以 2，再乘以 3…直到 n。递推法比递归法更容易理解和实现。但是有些问题则只能用递归算法才能实现。如后面将介绍到的 Hanoi 塔问题。

（2）如果 n 过大（>12）,结果将会不正确。因为超过了整数范围（约 21 亿）。

例 7.11 输入整数 n(0<=n<=10000),用递归法计算 n! mod 2012。

分析：n! mod 2012 就是 n!除以 2012 的余数。

因为 n!太大，不可能求出来以后除以 2012，得到余数。

我们知道：n!=(n-1)! ×n，如果求出（n-1）!除以 2012 的余数×n，就不会超界。

定理： (a×b)%c=((a%c)×(b%c))%c

证明：设 a=x1×c+x2,b= y1×c+y2,(x2＝a%c,y2=b%c)。

∴a×b＝（x1×c+x2）×（y1×c+y2）

＝x1× y1×c^2+ x1×y2×c+ x2×y1×c+x2×y2

∴(a×b)%c＝(x1× y1×c^2+ x1×y2×c+ x2×y1×c+x2×y2)%c

＝(x2×y2)%c

＝((a%c)×(b%c))%c

∴(a×b)%c=((a%c)×(b%c))%c

于是 n!%2012=((n-1)!%2012×(n%2012))%2012

$$n!\%2012=\begin{cases}1.....n=0...\\(n-1)!\%2012\times(n\%2012)\%2012.....n>0...\end{cases}$$

参考程序

```
#include<stdio.h>
int fac(int n)  //定义递归函数
{
  if (n==0) return 1;//边界，直接得结果
  else return fac(n-1)*(n%2012)%2012;//调用 fac(n-1)
}
int main()
{ int n;
  scanf("%d",&n);
  printf("%d",fac(n));
  return 0;
}
```

输入 300 输出 44

说明：

（1）对于正整数 a,b,c

(a+b)%c=(a%c+b%c)%c

(a-b)%c=(a%c-b%c+c)%c 两余数相减有可能为负数，所以要+c

(a×b)%c=((a%c)×(b%c))%c

（2）按照正常情况计算结果非常大时，一般就是 mod 某一个数，使结果变小。

“巨型机之父”西蒙·克雷

西蒙·克雷（Seymor Cray，1925年9月－1996年10月5日）曾任CDC公司的电脑总设计师，在20世纪60年代成功开发了第一代超级计算机。后独自创立“克雷研究公司”，专注于巨型机领域。西蒙·克雷(S. Cray)博士研发出符合超级计算机定义产品，后来被西方称为“巨型机之父”。

第 8 章　文件

例8.1 编写程序sum.cpp完成：从文本文件sum.in中读两个整数(<1000)，求出这两个数的和，并输出到文本文件sum.out中。

分析：本题就是求和，相当简单。但要求从文件中读数据，结果并输出到文件中。

C 语言中数据文件可分为 ASCII 文件和二进制文件。一般都使用 ASCII 文件，也就是大家经常使用的文本文件，每一个字节存放一个字符的 ASCII 代码。

要使用文件就要用文件指针变量来指向数据文件。 c 语言中自带了两个已经定义的标准输入输出文件指针变量。

stdin:标准的输入（standard in），指向默认输入设备，键盘。

stdout:标准的输出（standard out），指向默认输出设备，显示器。

对 stdin 和 stdout 重定向

所谓“重定向”，改变文件指针指向的文件。

freopen("sum.in","r",stdin)； 将文件指针 stdin 指向 sum.in 文件并以读的方式打开。其中"r"-read。只能从 stdin 指向的文件中读入数据，不能修改和写入。

freopen("sum.out","w",stdout)； 将文件指针 stdout 指向 sum.out 文件并以写的方式打开。"w"-write。

输入文件以读的文件打开，输出文件要以写的方式打开。

输入输出数据就和以前一样的使用了。

参考程序一

```
#include<stdio.h>
int main()
{
 int a,b,c;
 freopen("sum.in","r",stdin);
 freopen("sum.out","w",stdout);
 scanf("%d%d",&a,&b);
 c=a+b;
 printf("%d",c);
 return 0;
}
```

说明：

（1）程序文件 sum.cpp 和输入 sum.in 必须放在同一文件夹下。因为没有指定路径，默认为当前位置，即程序文件从程序文件所在的目录下的输入文件中读入入数据。

（2）输入文件 sum.in 由我们创建，程序才能读到输入文件中的数据。

现假设程序文件 sum.cpp 保存在桌面，sum.in 也只能创建在桌面。

sum.in 具体创建过程（这里用记事本创建文本文件）：

①鼠标右击桌面空白处→新建→文本文档；

②重命名为sum.in（此时一定要不“隐藏已知文件类型的扩展名”，否则重命后，文件名为sum.in.txt，没有删除原扩展名.txt）；

③双击sum.in文件打开，输入数据（如输入1 2）；

④保存关闭文件。

现在就可以运行sum.cpp程序。程序文件创建sum.out,并把运行结果写入到该文件。

双击文件sum.out 打开，就可看到运行结果了（结果为3）。

（3）输出文件sum.out程序文件创建(因为是写的方式)。

（4）用文件输入输出数据，就不用在“return 0;”前加“getchar();”语句了。

有关文件输入输出数据，参考程序一中采用的是标准输出文件指针变量stdin和stdout。下面我们介绍定义文件指针变量来完成本题的数据输入与输出。

(1)文件指针变量定义：

FILE *fp1,*fp2;定义了文件指针变量fp1，fp2。//注意*

FILE 一定大写。

(2)文件打开

①fp1=fopen("sum.in","r"); 将文件指针fp1指向文件sum.in并以读的方式打开。

只能从fp1指向的文件中读入数据，不能修改和写入。

②fp2=fopen("sum.out ","w"); 将文件指针fp2指向文件isbn.in并以写的方式打开。

输入文件以读的文件打开，输出文件要以写的方式打开。

(3)从文件中读出数据和将数据输出到文件中

①从文件中读出数据

用fscanf(文件变量,格式控制符，地址列表),如fscanf(fp1,"%d%d",&a,&b),从fp1指向的文件中读出两个整数到变量a、b中。

②将数据输出到文件中

用fprintf(文件变量,格式控制符，变量名列表),如fprintf(fp2,"%d",c)，向fp2指向的文件写入变量c的值。

由此可见fscanf( )与fprintf( )函数的用法和scanf( )和printf( )基本一样，差别是格式控制符前加了一个文件指针变量。

参考程序二

```
#include<stdio.h>
int main()
{
 int a,b,c;
 FILE *fp1,*fp2;
 fp1=fopen("sum.in","r");
 fp2=fopen("sum.out","w");
 fscanf(fp1,"%d%d",&a,&b);
 c=a+b;
 fprintf(fp2,"%d",c);
 return 0;
}
```

说明：

（1）关于文件输入与输出，一般使用 freopen( ),只是在程序中加以下两行：

freopen("输入文件名","r",stdin);

freopen("输出文件名","w",stdout);

其它的和以前使用一样。

（2）使用 fopen()打开文件，输入函数用 fscanf()，输出函数用 fprintf(),注意使用格式。把参考程序二多多练习即可掌握。

例8.2 数字反转(noip2011普及组第1题) (reverse.cpp/c/pas)

【问题描述】给定一个整数，请将该数各个位上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（参见样例2）。

【输入】输入文件名 reverse.in。输入共 1 行，一个整数N。

【输出】输出文件名 reverse.out。输出共 1 行，一个整数，表示反转后的新数。

【样例 1】reverse.in 123 reverse.out 321

【样例 2】reverse.in -380 reverse.out -83

【数据范围】-1,000,000,000 ≤ N≤ 1,000,000,000。

分析：

题意：将一个整数 n 反序后输出，负数首先输出“-”,再按正数处理，高位不能有 0 。

把 n 从个位到最高位依次取出来，生成一个新数 s 即可。

数据范围没有超过整数范围绝对值约 21 亿，所以可以用整型变量来保存这个数。

如 n=123,如下表操作得到新数 s

执行操作	n	s
n=123; s=0	123	0
s=s*10+n%10;n=n/10	12	3
s=s*10+n%10;n=n/10	1	32
s=s*10+n%10;n=n/10	0	321
由 n=0,n 从个位到高位已取完，不在重复执行。		

参考程序

```
#include<stdio.h>//reverse.cpp
int n,s=0;
int main()
{  //将文件指针 stdin 指向 reverse.in 并以读的方式打开。
  freopen("reverse.in","r",stdin);
   //将文件指针 stdout 指向 reverse.out 并以写的方式打开。
  freopen("reverse.out","w",stdout);
  scanf("%d",&n);
  if (n<0) {printf("-");n=-n;}//处理-
  while (n!=0)
  { s=s*10+n%10;
    n=n/10;
  }
```

```
    printf("%d\n",s);
    return 0;
}
```

说明

（1）“reverse.cpp/c/pas”表示程序可以用三种计算机语言之一进行编写。

程序设计语言	程序名
c++	reverse.cpp
c	reverse.c
pascal	reverse.pas

（2）输入文件名 reverse.in 表示该程序输入数据只能从文件 reverse.in 中读入，输出文件名 reverse.out 表示程序运行结果输出到文件 reverse.out 中。不能从键盘输入和输出到显示器上。

（3）此程序在 www.rqnoj.cn 上提交（试题编号 660）,注意提交时要去掉文件打开语句：两个“freopen();”。

（4）全国青少年信息学奥林匹克竞赛试题均采用文件输入与输出。切记程序文件名、输入输出文件名一定不能错，一般都是小写字母。如果文件名和输入输出文件名错，这道题就白做了！

例 8.3 ISBN 号码(noip2008 普及组第 1 题) **（isbn.pas/c/cpp）**

【问题描述】每一本正式出版的图书都有一个 ISBN 号码与之对应，ISBN 码包括 9 位数字、1 位识别码和 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），最后一位是识别码，例如 0-670-82162-4 就是一个标准的 ISBN 码。ISBN 码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；第二个分隔之后的五位数字代表该书在出版社的编号；最后一位为识别码。

识别码的计算方法如下：

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，所得的余数即为识别码，如果余数为 10，则识别码为大写字母 X。例如 ISBN 号码 0-670-82162-4 中的识别码 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 作为识别码。

你的任务是编写程序判断输入的 ISBN 号码中识别码是否正确，如果正确，则仅输出“Right”；如果错误，则输出你认为是正确的 ISBN 号码。

【输入】输入文件 isbn.in 只有一行，是一个字符序列，表示一本书的 ISBN 号码（保证输入符合 ISBN 号码的格式要求）。

【输出】输出文件 isbn.out 共一行，假如输入的 ISBN 号码的识别码正确，那么输出“Right”，否则，按照规定的格式，输出正确的 ISBN 号码（包括分隔符“-”）。

【样例 1】

isbn.in	isbn.out
0-670-82162-4	Right

【样例 2】

isbn.in	isbn.out
0-670-82162-0	0-670-82162-4

分析：

竞赛题目往往要描述一定的背景内容，题目一般都相当长。在做竞赛题目时，关键是把题目意思弄明白。如果读一次题意思不明白，可以多读几次，有些题目甚至可以读 4-5 次，并且还要结合样例才能明白题意。

题意：已知 ISBN 号码，判断其识别码是否正确。

（1）首先输入只能以字符串形式输入，设字符串为 a;

（2）按照规则对字符串 a 进行计算和 s;

（3）根据 s mod 11 的结果判断识别码是否正确，并进行相应输出。

细化（2）计算这个和 s。

观察“0-670-82162-4”的计算过程：

s=0×1+6×2+……+2×9=158,可以有下面两种方法。

①从左到右，第 j 个数字就乘以 j,累加起来。不是数字就计数。

②直接写出表达式进行计算：

s=(a[0]-48)*1+(a[2]-48)*2+(a[3]-48)*3+(a[4]-48)*4+(a[6]-48)*5+(a[7]-48)*6+(a[8]-48)*7+(a[9]-48)*8+(a[10]-48)*9

细化（3）时注意：

正确的条件：

余数为 10：s==10&&a[12]=='X'

或余数<10：s==a[12]-48

如果不正确改成正确的串输出字符串即可。

参考程序

```
# include<stdio.h>
char a[25];
int main()
{
    int s=0,j=1;
    FILE *fp;//注意大写 FILE
    fp=fopen("isbn.in","r");//以读方式打开 isbn.in 文件
    fscanf(fp,"%s",a);
    for(int i=0;i<=10;i++)
      {
         if (a[i]=='-');
         else
            {s=s+(a[i]-48)*j;j++;}
      }
    s=s%11;// mod 11 的余数
    fp=fopen("isbn.out","w");//写方式打开 isbn.out 文件
    if ((s==10)&&(a[12]=='X')||(s==a[12]-48))//两种情况
       fprintf(fp,"Right\n");
    else
      {  //不正确，将 a[12]改成正确的输出
        if (s==10) a[12]='X';else a[12]=s+48;
        fprintf(fp,"%s",a);
```

```
        }
      return 0;
}
```

说明

（1）将 for(int i=0;i<=10;i++)

```
      {
         if (a[i]=='-');
         else
            {s=s+(a[i]-48)*j;j++;}
      }
```

改成

```
s=(a[0]-48)*1+(a[2]-48)*2+(a[3]-48)*3+(a[4]-48)*4+(a[6]-48)*5+(a[7]-48)*6+(a[8]-48)*7+
(a[9]-48)*8+(a[10]-48)*9;
```

作用一样，但显得太“笨”了。

（2）此程序在 www.rqnoj.cn 上提交（试题编号 485）,注意提交时不能用文件。

（3）“fprintf(fp,"Right\n");”语句中"Right"一定要和题目要求一致，不能定成"right"。

例 8.3 排座椅(noip2008 普及组第 1 题)（seat.pas/c/cpp）

【问题描述】课的时候总有一些同学和前后左右的人交头接耳，这是令小学班主任十分头疼的一件事情。不过，班主任小雪发现了一些有趣的现象，当同学们的座次确定下来之后，只有有限的 D 对同学上课时会交头接耳。同学们在教室中坐成了 M 行 N 列，坐在第 i 行第 j 列的同学的位置是（i，j），为了方便同学们进出，在教室中设置了 K 条横向的通道，L 条纵向的通道。于是，聪明的小雪想到了一个办法，或许可以减少上课时学生交头接耳的问题：她打算重新摆放桌椅，改变同学们桌椅间通道的位置，因为如果一条通道隔开了两个会交头接耳的同学，那么他们就不会交头接耳了。

请你帮忙给小雪编写一个程序，给出最好的通道划分方案。在该方案下，上课时交头接耳的学生对数最少。

【输入】输入文件 seat.in 的第一行，有 5 各用空格隔开的整数，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】输出文件 seat.out 共两行。

第一行包含 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）。

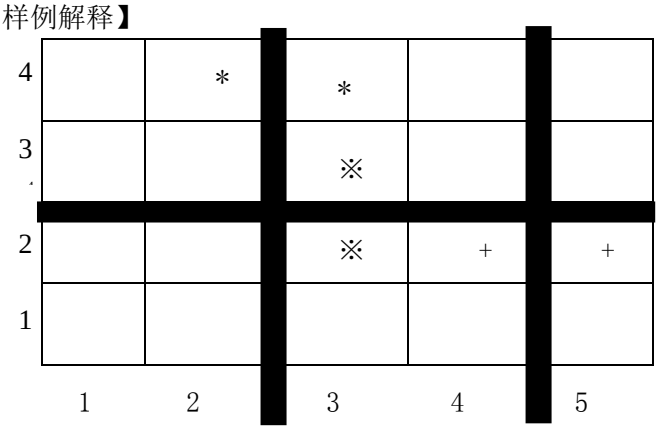
第二行包含 L 个整数，b1b2……bk，表示第 b1 列和 b1+1 列之间、第 b2 列和第 b2+1 列之间、…、第 bL 列和第 bL+1 列之间要开辟通道，其中 bi< bi+1，每两个整数之间用空格隔开（行尾没有空格）。

【输入输出样例】

seat.in	seat.out
4 5 1 2 3 4 2 4 3	2 2 4

2 3 3 3 2 5 2 4	

【输入输出样例解释】



上图中用符号*、※、+ 标出了 3 对会交头接耳的学生的位置，图中 3 条粗线的位置表示通道，图示的通道划分方案是唯一的最佳方案。

分析：

题意：已知有M×N 的方格教室，有 d 对相邻的同学之间要说小话，要在教室内设置 K 条横向的通道和 L 条纵向的通道，将隔开尽量多说话的同学。问怎么设置通道才合理？

下表格内相同的数字表示之间要说话。

4	1	2		3	
3	1	2	4	3	5
2			4	6	5
1				6	

1-2 行之间有 1 对同学说话

2-3 行之间有 2 对同学说话

3-4 行之间有 3 对同学说话

如果设置两条横向通道：显然在 2-3 行、3-4 行设置横向通道。

输出：2 3

一般情况下，设置 k 条横向通道：

（1）分别统计出在 1 行与 2 行之间、2 行与 3 行之间、3 行与 4 行之间，……，M-1 行与M 行之间说的人数（统计在行号小的下面，即 1 行与 2 行之间说话的人数统计在 1 行上）；

（2）按人数从多到少排序，前 K 个数分别对应行与行之间设置通过即可。

（3）但本题要求从行号从小到大输出。所以还得按行号从小到大排序。

（4）输出这K个行号。

这里涉及到两次排序，都可按从大到小排序，输出行号时从后面向前输出即可。

同样求出设置 L 条纵向通道的列号，要涉及两次排序排序，共要 4 次排序。为了方便将排序定成一个函数。用冒泡排序，时间复杂度 O (N^2),N 最大为 1000，O (10^6)，不会超时。

为了排序方便，定义一个结构体

```
struct hl
{
  int w[2];
};
```

再定义hl hang[1005],lie[1005];

hang[i].w[0]=i，hang[i].w[1]统计i行与i+1行之间说话的对数。

对hang[i]按hang[i].w[1]比大到小排序。则前面K个数对应的行之间设置通道。

要求行号从小到大输出，所以能前面K个数按hang[i].w[0]从大到小排序。

从第K个数－第1个数输出行号即可。

同理设置L条纵向通道的方法一样。

参考程序

```
#include<stdio.h>
struct hl
{
  int w[2];//w[0]存行或列号，w[1]统计行或列上说话同学对数
};
hl hang[1005],lie[1005];
//对数组a的前n个元素按w[k]的值排序
void sort(hl *a,int n,int k)
{
   for (int i=1;i<=n-1;i++)
     for (int j=1;j<=n-i;j++)
       if (a[j].w[k]<a[j+1].w[k])
       {
         hl t=a[j];a[j]=a[j+1];a[j+1]=t;
       }
}
int M,N,K,L,D;
int main()
{
  freopen("seat.in","r",stdin);
  freopen("seat.out","w",stdout);
  scanf("%d%d%d%d%d",&M,&N,&K,&L,&D);
  int x,y,p,q;
  for (int i=1;i<=M;i++) hang[i].w[0]=i;
  for (int i=1;i<=N;i++) lie[i].w[0]=i;
  for (int i=1;i<=D;i++)
  {
    scanf("%d%d%d%d",&x,&y,&p,&q);
    if (x==p)//行相等，显然是相邻两列上说话
      if (y<q) lie[y].w[1]++;
      else lie[q].w[1]++;
```

```
      else  //列相等，显然是相邻两行上说话
        if (x<p) hang[x].w[1]++;
        else hang[p].w[1]++;
    }
   sort(hang,M-1,1);//按行上说话人数多少排序
   sort(hang,K,0);  // 按行号从大到小排序
   sort(lie,N-1,1);//按列上说话人数多少排序
   sort(lie,L,0);  //按列号从大到小排序
   for (int i=K;i>=2;i--)//因为输出，最后不能有多余空格
     printf("%d ",hang[i].w[0]);
   printf("%d\n",hang[1].w[0]);
   for (int i=L;i>=2;i--)
     printf("%d ",lie[i].w[0]);
   printf("%d\n",lie[1].w[0]);
   return 0;
  }
```

说明：

（1）此程序在www.rqnoj.cn上提交（试题编号 486）测试通过,注意提交时去掉两个“freopen();”

（2）rqnoi.cn测试系统要求输出结果最后不能有多余空格，所以每一行的最后一个数单独输出。

```
for (int i=K;i>=2;i--)//因为输出，最后不能有多余空格
   printf("%d ",hang[i].w[0]);
 printf("%d\n",hang[1].w[0]); //最后一个数单独输出。
```

8.5 回文数(noip1999 提高组第 1 题，www.noj.cn 试题编号 148)

8.6 进制转换(noip2000 提高组第 1 题，www.noj.cn 试题编号 295)

8.7 选数(noip2002 普及组第 2 题，www.noj.cn 试题编号 67)

8.8 乒乓球(noip2003 普及组第 1 题，www.noj.cn 试题编号 51)

8.9 不高兴的津津(noip2004 普及组第 1 题，www.noj.cn 试题编号 20)

8.10 花生采摘(noip2004 普及组第 2 题，www.noj.cn 试题编号 23)

8.11 陶陶摘苹果(noip2005 普及组第 1 题，www.noj.cn 试题编号 12)

8.12 明明的随机数(noip2006 普及组第 1 题，www.noj.cn 试题编号 1)

8.13 火柴棒等式(noip2008 提高组第 1 题，www.noj.cn 试题编号 394)

8.14 笨小猴(noip2008 提高组第 2 题，www.noj.cn 试题编号 399)

8.15 潜伏者(noip2009 提高组第 2 题，www.noj.cn 试题编号 518)

8.16 数字统计(noip2010 普及组第 1 题，www.noj.cn 试题编号 602)

8.17 机器翻译(noip2010 提高组第 1 题，www.noj.cn 试题编号 598)

8.18 铺地毯(noip2011 提高组 day1 第 1 题，www.noj.cn 试题编号 654)

8.19 计算系数(noip2011 提高组 day2 第 1 题，www.noj.cn 试题编号 659)

高庆狮-中国著名计算机科学家

1957 年毕业于北京大学数学力学系。历任中国科学院计算技术研究所研究员、中科院技术科学部委员。擅长巨型电子计算机总体功能设计、并行算法和人工智能。完成了我国第一台晶体管大型电子计算机的功能总体设计和逻辑设计。是我国自行设计的第一台电子管大型计算机的体系功能设计和逻辑设计负责人之一。负责完成中国第一台每秒十万亿以上的晶体管大型计算机的体系功能设计。1973 年提出纵横加工流水线向量机设计思想，领导完成了我国第一台千万亿次大型向量计算机的系统功能设计，著有《向量巨型机》等。