

C++程序结构

C++是众多编程语言中的一种，也是最受欢迎的语言之一，使用极其广泛，关于其语言背景，在此先略过不谈，有兴趣者请自行搜索吧。以下直接就 C++程序结构开始讲起。

对于入门者而言，以下是 C++最基本的程序结构。

```
1. #include<iostream> //引用头文件
2.
3. using namespace std; //声明命名空间
4.
5. int main(){ //主函数
6.     int a,b,c;
7.     cin>>a>>b;
8.     c=a+b;
9.     cout<<c;
10. }
```

接下来，我们简要介绍这三部分。

```
#include<iostream> //引用头文件
```

#include<>是 C++中引用头文件的格式，方括号中加入头文件的名字。为什么要引用头文件呢？头文件就像一个抽屉，里面放着各种工具，这些工具在程序中叫做函数（如果你不理解什么是函数，没关系，反正我们现在也用不到），如果我们要使用这些工具，就得先打开抽屉，而引用头文件就是告诉程序打开这个抽屉，我要从这个抽屉中拿工具。在以上代码的示例，**#include<iostream>**是告诉程序打开一个叫 **iostream** 的抽屉，然后我们从这个抽屉中拿出了 **cin** 和 **cout** 这两个工具，这两个工具在主函数中使用。如果不先引用 **iostream** 头文件，主函数中 **cin** 和 **cout** 工具将不能使用。当然，C++中有很多抽屉（头文件），每个抽屉中都放着各种工具，这在我们以后的学习中会慢慢接触到的。

```
using namespace std; //声明命名空间
```

声明命名空间，其实这句话不是必须的，但是使用它，可以使我们的程序变得简洁一点。为什么要声明命名空间呢？打个比方，假设在学校中，三年 1 班有一个同学叫小明，4 年 2 班也有一个同学叫小明，如果我们在广播中叫小明，就不知我们具体叫的是哪一班的小明，除非我们加上班级，如三年 1 班的小明。但是，我们可以先做个声明：以下我所读的名字都是三年 1 班的同学，接着我读到小明，尽管这里没有加限定，但是大家都知道我所指的是三年 1 班的小明。同理，C++中有很多工具（函数），工具名难免相同，所以在使用工具的时候我就得声明是哪个抽屉中的工具，比如用 **cin**，我就得写成 **std::cin**，用 **cout** 我就得写成 **std::cout**，这就是给 **cin** 和 **cout** 加限定条件，而使用命名空间，告诉程序，我使用的工具都是 **std** 这个命名空间内的，我说的 **cin**，就是 **std::cin**，我说的 **cout**，就是 **std::cout**，这样，我们就可以把工具前的限定条件 **std::** 省略掉了。当然，如果这个比喻不能让你理解，也没关系，毕竟这个语句也不影响我们的编程学习。

接下来是 main 函数

```
1. int main(){
2.     int a,b,c;
3.     cin>>a>>b;
4.     c=a+b;
5.     cout<<c;
6. }
```

以上是主函数的格式，每个 C++ 程序必须有个 main() 函数才能运行。在详细解释 main 函数这段代码之前，我们先运行下这段代码。

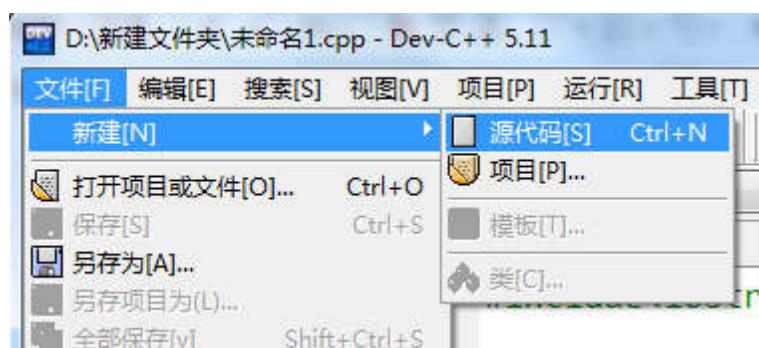
输入代码。

理论上你可以用任何文本工具敲写代码，在这里推荐大家使用基于 windows 平台的 dev C++。Dev C++ 是免费软件，下载安装都很方便，随便百度搜索下都下载得到。如下图。



mac 平台也可以使用类似 CLion 或者 Xcode 等软件编写 C++ 程序。

打开 Dev C++ 软件，从“文件”菜单中新建一个“源代码”文件



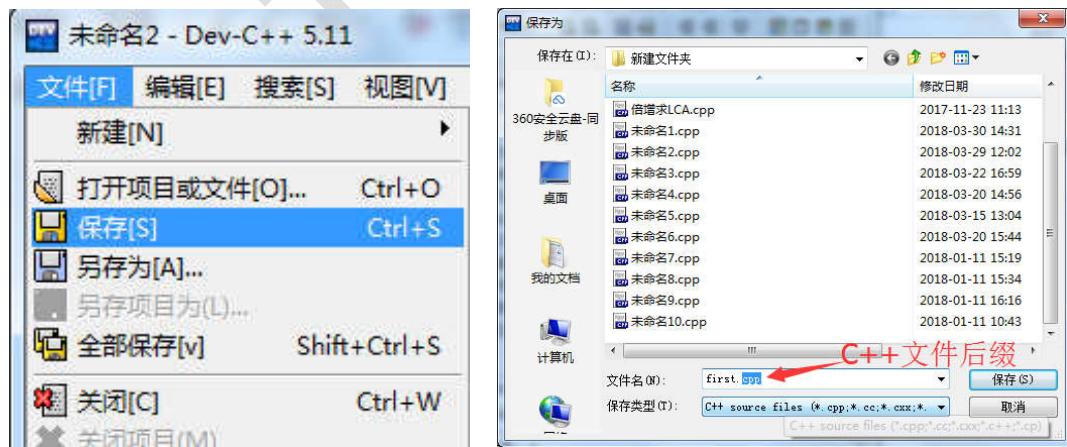
在新建的源代码文件中输入以下代码，特别注意标点符号不能错。

```

1 #include<iostream>
2
3 using namespace std;
4
5 int main(){
6     int a,b,c;
7     cin>>a>>b;
8     c=a+b; // Line 8 highlighted in light blue
9     cout<<c;
10 }
11
12

```

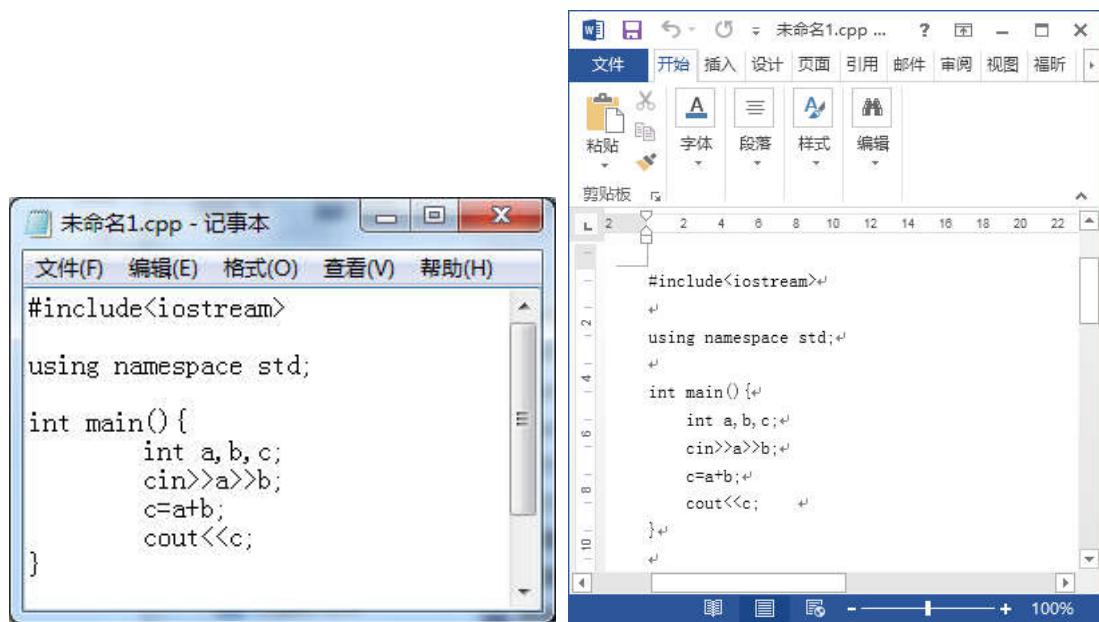
然后，点击菜单栏“文件”->“保存”，保存代码。注意，C++程序文件的默认后缀为 cpp。



编译程序

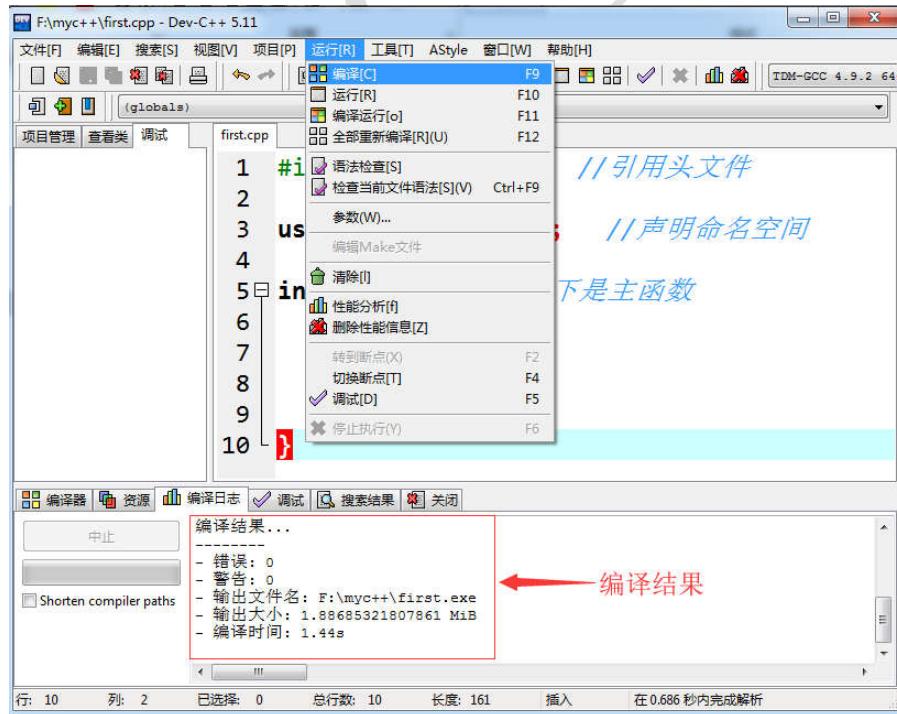
如果你写好了上述代码，恭喜你完成首次 C++ 编程，但是，这仅仅是代码，它还不是程序。程序是指能够被电脑直接执行的文件，比如后缀为 exe 的文件。保存代码后你再瞧瞧，这个

程序的后缀是 **cpp**, 这个文件可以被任何文本处理软件打开, 可以用记事本打开, 也可以用 word 打开 (如下图), 打开之后, 也就只有上述代码。



那么, 如何将代码变成程序呢? 打个比方, 要将一堆面粉变成香喷喷热乎乎的面包, 其中还需要一个烘焙的过程, 在这里, 代码就好比面粉, 程序就好比面包, 要将代码变成程序, 还需要一个烘焙的过程, 这个过程, 我们叫做编译。编译就是将代码转换成程序的过程。

在 Dev C++ 中编译程序, 在菜单栏中选择“**运行**”->“**编译**” (如下图), 进行编译程序。如果代码没有错误, 将能正确生成一个 **exe** 文件, 可以在 **cpp** 所在同级目录中查找。



如果编译出错, 则可以结合出错信息在出错行附近查找错漏, 留心每行语句是否用了分号; 结尾, 头文件名字是否打错等。

The screenshot shows the Dev-C++ IDE interface. The code editor window displays a file named 'first.cpp' with the following content:

```

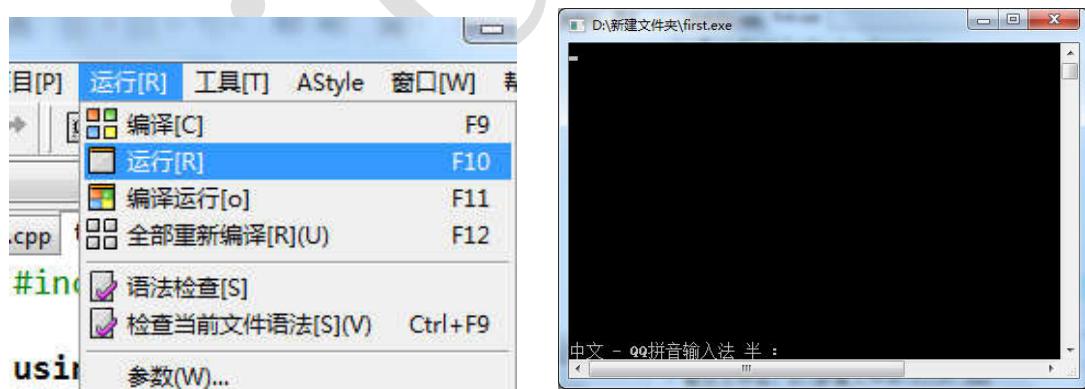
1 #include<iostream> //引用头文件
2
3 using namespace std //声明命名空间
4
5 int main(){ //以下是主函数
6     int a,b,c;
7     cin>>a>>b;
8     c=a+b;
9     cout<<c;
10 }

```

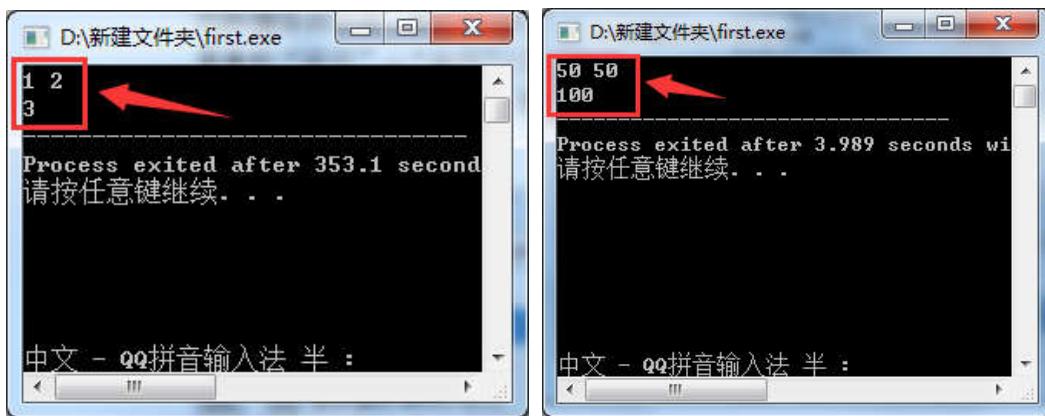
A red arrow points from the text '出错行' (Error Line) to the opening brace '{' at line 5, column 1. Another red arrow points from the text '出错信息' (Error Message) to the status bar at the bottom, which shows the message '[Error] expected ';' before 'int''. The status bar also displays other information like '行: 5' and '列: 1'.

运行程序

代码编译成功后，我们就可以运行程序了，这就像面包烘烤完成，我们当然是开吃啦。点击菜单栏“运行”->“运行”（如下左图），运行程序，此时你会看到如下右图的一个黑乎乎的窗口，里面的光标在闪动，似乎在等待着什么。是的，它在等待你输入数据。



这时，你可以尝试输入两个数字，数字与数字之间用空格隔开，然后按回车键，看看发生什么。多次运行程序，输入不同的两个数字，看看程序输出的数据有什么变化。



没错，这个程序实现了两个数的加法。看看你的程序，这么简短的几行代码就实现了两个数的加法。兴奋吧！

先别急着高兴，这里有几个问题需要你做下实验验证下：

- 1、你成功写出了两个数的加法，你能否改造你的程序求两个数的减法、乘法、除法？
- 2、在运行程序时，你输入的最大的数是多大？如果输入两个比较大的数，比如 10 的 18 次方级别的两个数，程序还能返回正常结果吗？
- 3、在进行两个数的运算时，你是否只测试了整数，如果输入小数程序能正常工作吗？

仔细观察，对于以上的后两个问题，我们找到了否定的答案。以下是实验截图，有图有真相。



太诡异了，为什么会出现这种现象呢？

数据类型

上述问题，跟数据类型有关系。数据类型是程序的基础，它告诉我们数据的意义以及我们能在数据上执行的操作。**C++**语言支持广泛的数据类型。它定义了几种基本内置类型（如字符、整型、浮点数等），同时也为程序员提供了自定义数据类型的机制。这里，我们先认识两种基本的类型：整型和浮点型。

整型

整型，其实也是整数，C++中用 int 来标识。与数学上整数的概念不同的地方，计算机中的整型是一个有限的概念，数学上的整数是一个无穷的概念，不存在一个最大的整数，但是在 C++ 中，int 所能表示的整数是有一个范围的，在 -2147483648 至 2147483647 之间，超过这个范围，就不是 int 所能表示的了。当然，在 C++ 中，关于整型的标识符可不止 int 一个，还有 short, long long 等。如果我们把整型比喻成一个物品类，比如衬衣，那么 int 就相当于衬衣中的中码，short 就是小码，long long 就是大码，当然还有加大码，不过现在这三个整型标识符就足够我们用了。记住，每个整型标识符（short、int、long long）的表示范围都是有限的。以下是关于整型更详细的范围介绍。

数据类型	定义标识符	占字节数	数值范围	数值范围
短整型	short [int]	2(16 位)	-32768~32767	$-2^{15} \sim 2^{15}-1$
整型	[long] int	4(32 位)	-2147483648~2147483647	$-2^{31} \sim 2^{31}-1$
长整型	long [int]	4(32 位)	-2147483648~2147483647	$-2^{31} \sim 2^{31}-1$
超长整型	long long [int]	8(64 位)	-9223372036854775808~-9223372036854775807	$-2^{63} \sim 2^{63}-1$
无符号整型	unsigned [int]	2(16 位)	0~65535	$0 \sim 2^{16}-1$
无符号短整型	unsigned short [int]	2(16 位)	0~65535	$0 \sim 2^{16}-1$
无符号长整型	unsigned long [int]	4(32 位)	0~4294967295	$0 \sim 2^{32}-1$
无符号超长整型	unsigned long long	8(64 位)	0~18446744073709551615	$0 \sim 2^{64}-1$

认识整型概念了，我们要让我们的程序支持更大整数的加法，可以很容易的实现，如下图所示。但是，尽管改成了大码的整型 long long，它也是有一个范围限制的。

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     long long a,b,c;    //这里 int 改成了 long long
5.     cin>>a>>b;
6.     c=a+b;
7.     cout<<c;
8. }
```



浮点型

浮点型是用来表示小数的数据类型，在C++中浮点型的标识符有float、double和long double，这三个类型也好比衣服界的小码、中码、大码，表示的数据范围不同。具体范围如下：

数据类型	定义标识符	数值范围	占字节数	有效位数
单精度实型	float	-3.4E-38~3.4E+38	4(32位)	6~7位
双精度实型	double	-1.7E+308~1.7E+308	8(64位)	15~16位
长双精度实型	long double	-3.4E+4932~1.1E+4932	16(128位)	18~19位

表中的数值范围用的是科学计数法，如3.4E+38是指 3.4×10^{38}

同理，认识浮点型后，我们也很容易将我们的程序改造成支持小数运算，方法如下：

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     double a,b,c;    //这里将int类型改成了double类型
5.     cin>>a>>b;
6.     c=a+b;
7.     cout<<c;
8. }
```



变量

其实数据类型，指的是一种规范，一种标准，好比我们说大码的衣服，指的是这件衣服领口的标准、袖长的标准，并没有具体指哪一件衣服，我们说整型 int，指的是一个能够表示整数，且数据范围在 -2147483648 至 2147483647 之间的一种规范，并没有具体告诉程序怎样来存储这样的数据。

正如我们可以按照大码的尺寸来生产衣服一样，我们也可以按照 int 的标准来制定存储空间，这样所制定的存储空间，我们称为变量。变量提供一个具名的、可供程序操作的存储空间，C++ 中的每个变量都有其数据类型，数据类型决定着变量所占的内存空间大小、存储值的范围以及变量能参与的运算。声明变量的格式如下：

类型说明符 变量名列表;

这里，我们举例来说明：

```
int a;
```

这句话的意思是告诉程序，我需要分配一个能装 int 类型数据的内存空间，并且给这个内存空间取名为 a。这里的内存空间就像一个容器一样，它能装 int 类型的数据，就意味着它只能装整数，而且整数的范围在 -2147483648 至 2147483647 之间。简单的理解就是，变量是一种能装特定数据类型的容器，具体能装什么数据类型，在声明的时候就确定了。以下是更丰富的变量声明方式：

1. `int sum; // 分配一个能装 int 数据类型的内存空间，给它取名为 sum`
2. `float price, cost, pay; // 声明三个 float 类型的变量，分别取名为 price, cost, pay`
3. `int a=5, b=10, c=1; // 声明三个 int 类型的变量 a、b、c，并且给它们赋初值`

这里还需要特别强调的是，声明变量时，变量名是由我们自己取名的，取名的时候需要遵守以下规范：

- C++ 的标识符 (identifier) 由字母、数字和下划线组成，其中必须以字母或下划线开头。
- 标识符的长度没有限制，但是对大小写字母敏感。
- C++ 语言保留了一些名字供语言本身使用，这些名字不能被用作标识符。

练习：请指出下面的名字中哪些是非法的？

1. `int double=3.14;`
2. `int _;`
3. `int catch-22;`
4. `int 1_or_2=1;`
5. `double Double=3.14;`

下一页看答案。

```

1. int double=3.14; //非法, double 是数据类型关键字
2. int _; //合法, 下划线开头,但不建议这样取变量名
3. int catch-22; //非法, 包含非法字符‘-’,变量名只能是数字、字母和下划线
4. int 1_or_2=1; //非法, 必须以字母和下划线开头
5. double Double=3.14; //合法, Double 和 double 是不一样的, 区分大小写

```

赋值语句

刚刚, 我们说了变量是一个容器, 可以往里面装数据, 但是怎么装呢? 这里就需要用到我们的赋值语句。赋值语句的作用就是将一个数据存储到变量中, 赋值语句的一般形式:

变量=表达式; //不要漏了末尾的分号哦

在 C++语言中, “=”作为赋值运算符, 而不表示“等于”判断。赋值语句是由赋值表达式再加上分号构成的表达式语句。样例分析

```

1. a=3; //将数值 3 存储在左边的变量 a 中
2. b=4; //将数值 4 存储在左边的变量 b 中
3. c=a+b; //计算 a+b 的结果, 将结果存储在左边的变量 c 中

```

注意, 赋值运算符的左侧运算对象必须是一个变量。以下赋值语句是非法的。

```

1. 1024=k; //非法赋值, 1024 不能是变量名
2. i+j=k; //非法赋值,i+j 不能是变量名

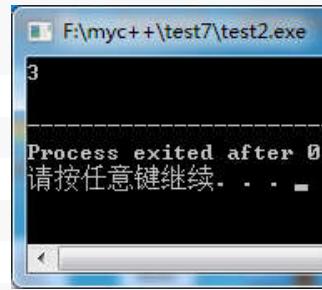
```

在进行赋值运算时, 如果赋值运算符两边的数据类型不同, 系统将会自动进行类型转换, 即将赋值运算符右边的数据类型转换成左边的变量类型。当左边是整型而右边是实型时, 将去掉小数部分并截取该整型对应的有效位数。如下程序所示:

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a=4;
5.     float b=3.1415624;
6.     a=b;
7.     cout<<a<<endl;
8. }

```



输出语句

在了解了这么多概念后, 我们继续来看什么是输出语句。

```

1. #include<iostream> //引用头文件
2. using namespace std; //声明命名空间

```

```

3. int main(){ //主函数
4.     int a,b,c; //声明三个 int 类型的变量，命名为 a,b,c
5.     cin>>a>>b;
6.     c=a+b; //赋值语句，将 a+b 的结果存储到 c 变量中
7.     cout<<c; //输出语句， (σ°∀°)σ...:☆ 我是主角哦
8. }

```

一个程序可以没有输入，但不能没有输出（好吧，我说的是信息学竞赛的题目），我们辛辛苦苦编写一个程序，不就是让计算机计算之后告诉我们答案嘛。

在 C++ 中，输出语句由 `cout` 和流插入运算符 `<<` 结合在一起使用，可向显示器屏幕输出数据，其基本格式如下：

`cout<<表达式;`

输出语句可以把表达式的值输出到屏幕上，该表达式可以是各种基本类型的常量、变量或者由它们组成的表达式。输出时，程序根据表达式的类型和数值大小，采用不同的默认格式输出，大多数情况下可满足要求。若要输出多个数据，可以连续使用流插入运算符。

以下是示例

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     double a,b,c;
5.     a=3;
6.     b=4;
7.     c=3+4;
8.     cout<<c; //输出变量的值
9. }

```



`cout` 的注意事项：

- 每当我们输出字符串常量的时候，必须用双引号把字符串引起来，以便将它和变量名明显的区分开来。下面两个语句是不同的：

```

1. cout << "Hello" ;           //打印字符串 Hello 到屏幕上
2. cout << Hello ;           //把变量 Hello 存储的内容打印到屏幕上

```

- 必须注意，`cout` 并不会自动在其输出内容的末尾加换行符，因此下面的语句：

`cout << "This is a sentence." ;`

`cout << "This is another sentence." ;`

将会有如下内容输出到屏幕：

`This is a sentence.This is another sentence.`

为了在输出中换行，可以用操作符 `endl` (`endline`) 来换行，例如：

```
cout << "First sentence." << endl;
cout << "Second sentence." << endl;
将会输出:
First sentence.
Second sentence.
```

练习：如何让我们两个数的加法输出结果更友好一些，比如输出结果如下



这里是参考代码：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,b,c;
5.     cin>>a>>b;
6.     c=a+b;
7.     cout<<a<<"+"<<b<<"="<<c;    //注意这一句
8. }
```

输入语句

我们来讲本节的最后一个内容，输入语句。

```
1. #include<iostream> //引用头文件
2. using namespace std; //声明命名空间
3. int main(){ //主函数
4.     int a,b,c; //声明三个 int 类型的变量，命名为 a,b,c
5.     cin>>a>>b; //输出语句，(σ °∀°)σ..:☆ 轮到我做主角了
6.     c=a+b; //赋值语句，将 a+b 的结果存储到 c 变量中
7.     cout<<c; //输出变量 c 的值
8. }
```

我们的程序用了输入语句，才能实现交互，也才能解决各种不同数据的问题。输入语句由 `cin` 和流读取运算符 `>>` 结合在一起使用，其功能是程序运行后遇到输入语句停下来，从键盘读取数据并将其赋给“变量”。以下是输入语句的格式

`cin>>变量 1>>变量 2>>变量 3;`

以下是示例：

The screenshot shows a C++ development environment. On the left, the code editor displays the following C++ code:

```

1 #include<iostream>
2 using namespace std;
3 int main(){
4     int a,b,c;
5     cin>>a>>b;
6     c=a+b;
7     cout<<c;
8 }
9

```

A red arrow points from the terminal window to the line of code `cin>>a>>b;`. The terminal window on the right has a black background and contains the text "等待输入" (Waiting for input).

以上代码编译运行后，遇到 `cin` 语句，程序暂停运行，等待用户输入数据。这时，用户需要输入和 `cin` 语句后面所接的变量个数相同，变量类型相同的数据，数据输入后，会依次赋值给变量。如下图所示，因为 `cin>>a>>b;` 这里有两个变量 `a` 和 `b`，所以需要输入两个数值，因为 `a` 和 `b` 是整型，所以要求输入的数据类型是整型，数据输入结束后（按回车表示输入结束），`34` 会赋值给变量 `a`，`26` 会赋值给变量 `b`。

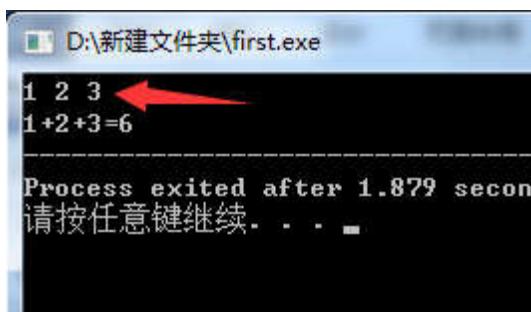
The screenshot shows a C++ development environment. On the left, the code editor displays the same C++ code as before. A red arrow points from the terminal window to the line of code `cin>>a>>b;`. The terminal window on the right has a black background and contains the text "等待" (Waiting). Above the terminal window, there is a status bar with the text "34 26". Red arrows point from the numbers "34" and "26" to the terminal window.

On the right side of the terminal window, there is explanatory text in red: "输入个数相同、类型相同的数据，数据依次赋值给变量" (Input the same number of elements of the same type, and the data will be assigned to variables sequentially).

输入语句的注意事项：

- 1、在使用 `cin` 输入的时候必须考虑后面的变量类型。如果你要求输入一个整数，在 `>>` 后面必须跟一个整型变量，如果要求一个字符，后面必须跟一个字符型变量。
- 2、可以连续使用 `>>`，实现从键盘对多个变量输入数据。这要求从键盘输入的数据的个数、类型与变量相一致。从键盘读取数据时，各数据之间要有分隔符，分隔符可以是一个或多个空格键、回车键等。

练习：输入三个整数，实现三个整数的加法。如下所示：



参考代码如下：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,b,c,d;
5.     cin>>a>>b>>c;
6.     d=a+b+c;
7.     cout<<a<<"+"<<b<<"+"<<c<<"="<<d;
8. }
```

=====网站练习=====

学习编程，不仅仅是知道程序的相关知识，更重要的是自己动手编写程序，这就好比学游泳一般，光有理论不行，必须下水实践才能学会这项技能。

本课程每一节都精挑了 6 道题目进行巩固练习，你需要登录 <http://www.oiClass.com> 上完成。

ID	Name	Status	Private	Creator
1000	程序结构	运行中 剩余 3470天18小时54分36秒	公开	chxulong
1052	表达式	运行中 剩余 3470天18小时51分36秒	公开	chxulong
1053	if语句	运行中 剩余 3470天18小时51分36秒	公开	chxulong
1054	for语句1	运行中 剩余 3470天18小时51分36秒	公开	chxulong
1055	for语句2	运行中 剩余 3470天18小时51分36秒	公开	chxulong
1056	for语句3	运行中 剩余 3470天18小时51分36秒	公开	chxulong
1057	while语句1	运行中 剩余 3470天18小时51分36秒	公开	chxulong
1058	while语句2	运行中 剩余 3470天18小时51分36秒	公开	chxulong
1059	一维数组1	运行中 剩余 3470天18小时51分36秒	公开	chxulong
1060	一维数组2	运行中 剩余 3470天18小时51分36秒	公开	chxulong
1093	一维数组3	运行中 剩余 3553天16小时47分36秒	公开	chxulong

A+B Problem

【题目描述】

输入两个整数，输出两个整数的和

【输入】

两个整数

【输出】

一个整数

【样例输入】

23 4

【样例输出】

27

【分析】

本题意在让使用者熟悉 oiiclass 网站的在线评测功能。oiiclass 网站采用黑盒测试，对输入数据有严格的格式要求，对输出数据进行全文本匹配来判断对错，所以，程序的输出要严格按照题目要求。使用 oiiclass 进行在线评测，需要遵从以下步骤：

1、认真阅读题意，理解题目的输入格式和输出格式，在本地利用 dev C++软件进行编写程序，程序可以正常编译，并且利用输入样例也能输出和输出样例一样的结果时，再准备提交程序。

2、提交程序。在题目页点击提交按钮（如下图）

问题 : A+B Problem

时间限制: 1 Sec 内存限制: 128 MB
提交: 348 解决: 244
[命题人:chxulong][Edit][TestData]

提交 状态 讨论版

题目描述

输入两个整数，输出两个整数的和

输入

两个整数

输出

一个整数

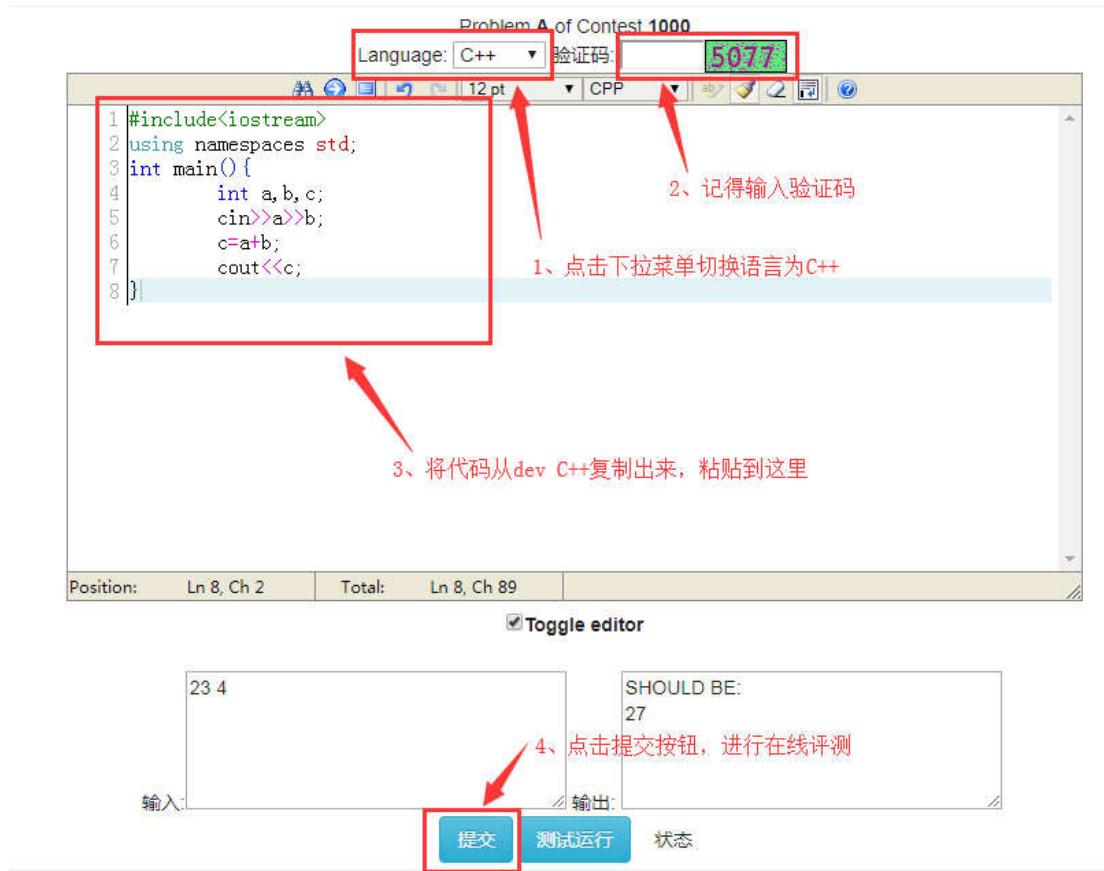
样例输入

23 4

样例输出

27

页面会跳转到提交代码页，如下图，注意设置提交的语言，输入验证码（验证码注意刷新），然后将代码粘贴到代码框，点击提交按钮就可以进行在线评测了。



3、评测结果。如果你的代码正确，网站会显示你的答案正确。如果显示答案错误，你可以点击答案错误按钮，看看自己的输出和标准输出的异同。

题目编号:	用户:	语言:	结果:	内存	耗时	语言	代码长度	提交时间	判题机
提交编号	用户	问题	结果						
36138	chxulong	A	正确	1708	0	C++/Edit	102 B	2018-08-28 22:31:46	172.18.48.146
36137	chxulong	A	编译错误	0	0	C++/Edit	103 B	2018-08-28 22:31:10	172.18.48.146
18950	chxulong	A	正确	1088	0	C++/Edit	71 B	2018-07-08 13:29:54	172.18.48.146
9498	chxulong	A	正确	1708	0	C++/Edit	102 B	2018-04-17 20:55:40	172.18.48.146
8481	chxulong	B	正确	1708	0	C++/Edit	88 B	2018-04-07 16:09:42	172.18.48.146
8434	chxulong	A	正确	1708	0	C++/Edit	109 B	2018-04-07 11:02:57	172.18.48.146
8433	chxulong	A	答案错误	1708	0	C++/Edit	125 B	2018-04-07 11:01:40	172.18.48.146
8055	chxulong	F	正确	1708	0	C++/Edit	147 B	2018-04-03 11:21:03	172.18.48.146

点击“答案错误”按钮，你可以看到错误的相关信息。结合这些信息，思考你的输出为什么会和标准输出不一样，然后重新修改程序，再次提交即可。

```
===== [plus2.out] =====
----- test in top 100 lines -----
325 234
----- test out top 100 lines -----
559 ← 标准输出
----- user out top 100 lines -----
325+234=559 ← 你的输出
===== [plus1.out] =====
----- test in top 100 lines -----
1 1
----- test out top 100 lines -----
2
----- user out top 100 lines -----
1+1=2
=====
```

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,b,c; // 定义变量
5.     cin>>a>>b; // 输入数据
6.     c=a+b; // 数据处理
7.     cout<<c; // 输出数据
8. }
```

航班管理

【题目描述】

某国家有 N 个城市。每个城市都有一个机场。但现在这些机场之间都没有航班通行。你需要安排航班连通这些机场。安排航班必须满足下面 3 个规则：

1. 每个航班是双向服务，直接连通两个机场。
2. 两个机场之间只有一个航班飞行。
3. 保证可以从一个城市到达其它任意城市（直接到达或者转乘其他航班到达）

你的任务是在城市之间设计最少的航班数来保证上面的规则成立。

【输入】

一行，一个整数 N ($2 \leq N \leq 50$) 表示城市的个数。

【输出】

输出一个整数，表示最少的航班数。

【样例输入】

3

【样例输出】

2

【样例解释】

可以在城市 1 与城市 2 建立一个航班，城市 1 与城市 3 建立一个航班，这样，1 与 2、1 与 3 可以直航，2 与 3 可以通过城市 1 转乘到达。

连接方案可以有多种。

【分析】

本题目其实是一个思维题，即给出 n 个点，求最少连多少条边可以把这些点都连接起来，答案显然是 $n-1$ 条边。即，输入 n ，输出 $n-1$ 就行。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n;
5.     cin>>n;
6.     cout<<n-1;
7. }
```

小明的时间

【题目描述】

小明是个非常忙碌的人，因此，他经常以秒为单位计算时间。但是，生活中习惯以小时、分钟、秒钟来计算时间，这使得他非常不习惯。因此，他希望学编程的你编写一个程序，将时间转化为秒数，你能帮到他吗？

【输入】

输入三个整数 h , m , s , 分别代表小时、分钟、秒数

【输出】

输出一个整数，表示转换后的秒数

【样例输入】

0 31 4 (表示 0 小时 31 分钟 4 秒)

【样例输出】

1864

【分析】

本题给出时、分、秒，要求换算成总秒数。假如有 h 小时， m 分钟， s 秒，已知一小时有 3600 秒，一分钟有 60 秒，所以总秒数有 $h*3600+m*60+s$ ，输出结果就行。

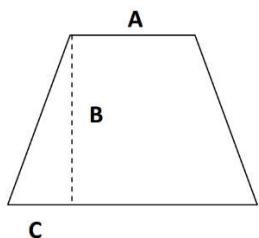
【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int h,m,s,tot;
5.     cin>>h>>m>>s;
6.     tot=h*3600+m*60+s;
7.     cout<<tot;
8. }
```

面积

【题目描述】

最近楠楠的数学老师在教他计算梯形面积，由于测验时他不小心计算错了一道求等腰梯形面积的题目，楠楠的老师要罚他计算 N 道这样的题，这太痛苦了！楠楠求你编个程序来帮助他快速计算面积。已经知道每道题给如下图形状的等腰梯形中的 A, B, C 三个参数，请编程计算它的面积。



【输入】

一行 3 个正整数 A, B, C ，分别表示等腰梯形的上底宽、高和左边突出的长度（即下底宽为 $A+2C$ ）。

【输出】

一行，一个整数，表示梯形的面积。

【样例输入】

2 3 1

【样例输出】

9

【数据范围】

10 个数据： A, B, C 的范围是 $[1 \cdots 20]$ 。

【分析】

本题求梯形面积。设梯形的上底为 a ，下底为 c ，高为 b ，则梯形的面积为 $(a+c)*b/2$ ，但是在本题中，我们已知上底，但是下底只有突出的一半 c ，幸好是一个等腰梯形，则下底为 $c+a+c$ ，所以整个梯形的面积为 $(a+c+a+c) * b / 2$ ，可以将公式化简为 $(a+c) * b$ 。

【代码】

```

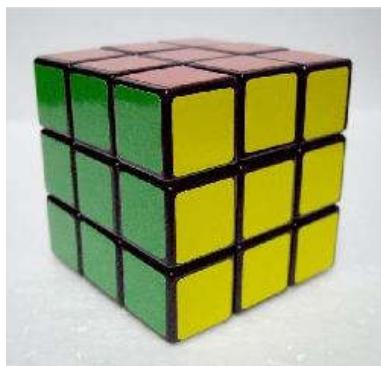
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,b,c,s;
5.     cin>>a>>b>>c;
6.     s=(a+c)*b;
7.     cout<<s;
8. }
```

魔方

【题目描述】

魔方大家都玩过吧？

常见的魔方，每边上有 3 个小正方体，如下图所示：



我们把魔方每边上的小正方体数量，叫魔方的“阶”，所以，常见的魔方叫“3 阶魔方”。

不过，魔方可不是只有 3 阶的，还有 2、4、5……阶的呢，如下图所示：



观察所有的魔方，你会发现，我们可以把魔方表面上的小正方体分为三类：

第一类：有三个面露在外面的；

第二类：有两个面露在外面的；

第三类：有一个面露在外面的。

当然，这三类小正方体的数量会随着魔方阶的不同而不同。你的任务就是计算一下，对于给定阶数的魔方，这三类小正方体分别有多少个。

【输入】

只有一个整数，表示魔方的阶数，已知 $2 \leq n \leq 100$ 。

【输出】

每行一个整数，分别表示对于阶的魔方，第一类、第二类、第三类的小正方体的数量。

【样例输入】

3

【样例输出】

8

12

6

【分析】

这道题，其实就是找规律啊。仔细观察，对于魔方表面的小正方体，三个面都露出来，只在角的位置，这个数目固定为 8。对于两个面都露出来的小正方体，只在正方体的棱上，对于每条棱，去掉头尾的角，有 $n-2$ 个小正方体露出，一共有 12 条棱，所有第二类的数目为 $(n-2) * 12$ 。对于第三类，只露出一个面的小正方体，则只存在一个面上，一个面上有 $n*n$ 个小正方体，去除外围一圈的棱和角，余下小正方体的个数为 $(n-2) * (n-2)$ ，一共有 6 个面，所以第三类的总个数有 $(n-2)*(n-2)*6$ 。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,a,b,c;
5.     cin>>n;
6.     a=8;      //第一类数目
7.     b=(n-2)*12;    //第二类数目
8.     c=(n-2)*(n-2)*6;    //第三类数目
9.     cout<<a<<endl;   //注意换行，否则会被判错
10.    cout<<b<<endl;
11.    cout<<c<<endl;
12. }
```

零花钱

【题目描述】

小洪妈妈每个星期一早上就会给她 s 元的零花钱，每到星期五晚上她就要告诉妈妈还 剩下多少钱，并且把剩余的钱上交。已知她星期一至星期五每天花费的金额，现在请你帮忙算出小洪星期五晚上还剩下多少钱。

【输入】

第一行：一个整数 s ，代表星期一早上母亲给的金额。 $(0 < s \leq 100)$ 第二行：五个整数，分别代表小洪每天的消费金额。

【输出】

一行：一个整数，代表星期五晚上小洪还剩下多少钱。

【样例输入】

50
5 5 11 10 5

【样例输出】

14

【分析】

本题侧重在于学会多个数据的读入。题目共需要读入 6 个数字，我们设 s , d_1, d_2, d_3, d_4, d_5 存储我们读入的数据，设变量 r 为最后剩余的钱，则 $r=s-d_1-d_2-d_3-d_4-d_5$ ，输出 r 就行。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int s,d1,d2,d3,d4,d5,r;
5.     cin>>s>>d1>>d2>>d3>>d4>>d5;
6.     r=s-d1-d2-d3-d4-d5;
7.     cout<<r;
8. }
```

表达式

我们在认识程序结构的时候接触过表达式，但是我们没有展开说明，还记得在哪里吗？

赋值语句的格式：

变量=表达式；

在这里，我们要更充分地展开对 C++ 表达式的理解。

表达式，简单地讲，就是由通过运算符连结运算对象的式子。特别的，单个常量或者单个变量也是表达式。请看以下例子：

1. `a=3; //单个常量 3 是表达式`
2. `a=b; //单个变量 b 是表达式`
3. `a=3+4; //运算符+号将运算对象 3 和 4 连结起来`
4. `a=b+3; //运算符+号将运算对象变量 b 和常量 3 连结起来`
5. `a=b+c; //运算符+号将运算对象变量 b 和变量 c 连结起来`

表达式的结果是一个确定的值，如 `3+4` 的值就是 7。

认识表达式，关键在于认识运算符。在 C++ 中运算符的类型非常多，在这里我们主要讲三类运算符：**算式运算符、关系运算符和逻辑运算符**。

算式运算符

算式运算符我们比较熟悉，常见的有 ‘+’、‘-’、‘*’、‘/’ 号，还有我们不太熟悉的 ‘%’。

算式运算符	功能	用法
+	加法	<code>Expr + expr</code>
-	减法	<code>Expr - expr</code>
*	乘法	<code>Expr * expr</code>
/	除法	<code>Expr / expr</code>
%	求余	<code>Expr % expr</code>

特别的 ‘/’ 号：C++ 中，‘/’ 号有两种功能，一个是整除，当运算对象都是整型时，‘/’ 号表示整除，比如 `5/2` 的结果是 2，即求 `5/2` 的商，自动忽略小数部分。另一个是普通意义的除，当运算对象有一个是浮点型时，运算的结果即为浮点型，比如 `5/2.0` 的结果是 2.5，或者 `5.0/2` 的结果也是 2.5。‘/’ 的这两种功能，需要特别弄清楚，否则写程序就会出现神奇的 bug。

特别的 ‘%’ 号：取余符号在数学中用得比较少，但在计算机编程中却经常用到。‘%’ 表示两个整数相除后的余数，如 `5%2` 的结果为 1。需要注意的是 ‘%’ 两边的运算对象类型必须

是整型，否则编译会出错，比如 `5.0%2` 就是错误的写法，编译不通过。

练习：计算以下表达式的值：

- (1) `-30*3+21/5`
- (2) `-30+3*21/5`
- (3) `30/3*21%5`

结果分别是 `-86, -18, 0`，跟你的答案一样吗？

有时候，我们需要将数学的表达式转换成 C++ 表达式，在这个过程需要注意区分，C++ 表达式中只能用小括号 () 进行嵌套，改变表达式的运算优先级，多个小括号可以层层嵌套，但是不能使用中括号和大括号。数学表达式中常见在变量之间省略运算符，而 C++ 则不允许这样做。

练习，将下列数学表达式转换成 C++ 表达式

$$\frac{5+b}{\frac{a+6}{b+5}-c \cdot d} \quad \frac{p \cdot q \cdot (r+1)^2}{(r+1)^2 - 1}$$

以下是参考答案，跟你写得一样吗？

`(5+b)/((a+6)/(b+5)-c*d)`
`p*q*(r+1)*(r+1)/((r+1)*(r+1)-1)`

例题：

输入一个三位自然数，分离出三位数的个位、十位、百位，将它们倒序输出

输入样例： 583

输出样例： 3 8 5

分析：

假如我们输入的一个三位数存储在变量 `a` 中，如何在 `a` 中分离出个位、十位、百位呢？对于任意位数的 `a`，将其整除 10 后，它的余数就是个位，比如 `123%10` 的结果就是 3，所以个位数可以写成 `a%10`。而 `123/10` 的结果则是 12，即整除 10 以后的结果相当于将该数的个位删除。那么如果求 10 位数呢？`123/10=12`，12 中的个位即是原数的十位数，所以，十位数可以写成 `a/10%10`。同理，百位数可以写成 `a/10/10`，简写成 `a/100`。我们将求出的各个位数存到变量 `ge`、`shi`、`bai` 中，即：

```
ge=a%10;
shi=a/10%10;
bai=a/100;
```

则以上题目的程序很容易就可以实现：

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,ge,shi,bai;
5.     cin>>a; //输入一个三位数
6.     ge=a%10; //分离个位
7.     shi=a/10%10; //分离十位
8.     bai=a/100; //分离百位
9.     cout<<ge<<" "<<shi<<" "<<bai<<endl;
10. }
```

思考：如何分离一个四位数的个位、十位、百位、千位呢？

关系运算符

关系运算符用来比较两种运算对象的关系，包括相等、不相等、大于、小于、大于等于、小于等于。关系表达式的结果是一种布尔类型，即运算结果只能是 `true` 或者 `false`。关系成立结果则为 `true`，关系不成立结果则为 `false`。

运算符	运算	运算对象	结果类型
<code>==</code>	等于	简单类型	布尔型
<code>!=</code>	不等于	简单类型	布尔型
<code><</code>	小于	简单类型	布尔型
<code>></code>	大于	简单类型	布尔型
<code><=</code>	小于等于	简单类型	布尔型
<code>>=</code>	大于等于	简单类型	布尔型

观察以下表达式的值：

```

1. 2<3 //结果是 true
2. 345.5<=100 //结果是 false
3. 12 != 10 //结果是 true
```

逻辑运算符

逻辑运算符用来关联两种条件的关系，常见的运算符有逻辑与 (`&&`)、逻辑或 (`||`) 和逻辑非 (`!`)，逻辑运算符关联的运算对象必须是布尔类型，其运算结果也是一个布尔类型。

逻辑与 (`&&`)：表示两个条件都成立结果才成立。比如：妈妈给小明制定的奖励规则是，小明的语文成绩 (`yw`) 大于 90 分并且数学成绩 (`sx`) 大于 95 (`yw>90&&sx>95`) 才给予奖励。

那么，如果小明的数学考了 100 分而语文考了 89 也是不奖励的。

逻辑或 (||): 表示两个条件只要有一个成立结果就成立。比如：公园收费规则为 10 岁以下或者 60 岁以上免费，只要符合两个条件之一就可以享受公园免费规则。

逻辑非 (!): 单目运算符，取反操作，就像电灯开关一样，如果开则变成关，关则变成开。

以下是三个逻辑运算符的真值表：

a	b	! a	a && b	a b
false	false	true	false	false
false	true	true	false	true
true	false	false	false	true
true	true	false	true	true

练习：根据以下信息写出表达式

1、整数 x 是偶数

2、写出一个数 x 既能被 3 整除又能被 5 整除的表达式

3、判断 y 是闰年的表达式（闰年的判断方法：能够被 4 整除且不能被 100 整除，或者能被 400 整除）

参考答案，注意，以下不是唯一写法：

1、 $x \% 2 == 0$

2、 $(x \% 3 == 0) \&\& (x \% 5 == 0)$

3、 $(y \% 4 == 0) \&\& (y \% 100 != 0) \mid\mid (y \% 400 == 0)$

好了，在这里我们介绍了三类运算符，需要注意的是三类运算符不是单独分离的，而是可以互相结合组成复杂的表达式。算式表达式的结果可以成为关系运算符的运算对象，而关系表达式的运算结果可以成为逻辑运算符的运算对象。以上面第 2 题为例， $x \% 3$ 是算式表达式，在 $x \% 3 == 0$ 中做关系运算符 == 的运算对象，而 $x \% 3 == 0$ 是关系表达式，在 $(x \% 3 == 0) \&\& (x \% 5 == 0)$ 中做逻辑运算符 && 的运算对象。根据这一规则，可以组成更复杂的表达式，如上面第 3 题。

表达式的注意事项：

1、整型、实型、字符型数据间可以混合运算。在这种情况下，需要将不一致的数据类型转换成一致的数据类型，然后进行运算。为了保证运算精度，系统在运算时的转换规则是将存储长度较短的运算对象转成存储长度较长的类型，然后再进行处理。这种转换是系统自动进行的。如： $3+4.5*2-7$ 返回的结果为浮点型

2、在求复合表达式的值需要考虑运算符的优先级。如果不明确运算符的优先级，可以使用括号将表达式的某个局部括起来使其得到优先运算。以下是各个运算符的优先级：

优先级	运算符	说明	结合性
1	()	括号	自左向右
2	!	逻辑非/按位取反	自右向左
3	* / %	乘法/除法/取余	
4	+ -	加号/减号	
5	< <=	小于/小于等于	
	> >=	大于/大于等于	自左向右
6	== !=	等于/不等于	
7	&&	与运算	
8		或运算	
9	=	赋值	自右向左

在上表中可以观察到算是运算符的优先级高于关系运算符，关系运算符的优先级高于逻辑运算符。

思考，求闰年的表达式，如果将式子的括号去掉，写成以下样子，结果会一样吗？

y % 4 == 0 && y % 100 != 0 || y % 400 == 0

=====网站练习=====

追及问题

【问题描述】

甲、乙两人环绕周长是 300 米的跑道跑步，甲每秒跑 6 米，乙每秒跑 4 米，如果两人是从同一地点同向出发，假设两人永远保持体力可以一直均速跑下去，我们知道，甲可以第一次、第二次、……、第 n 次追上乙。

求第 n 次相遇的时间($n \leq 10000$)，单位为秒。

【输入格式】

一个整数 $n(n \leq 10000)$ 。

【输出格式】

求第 n 次相遇的时间，单位为秒。

【输入样例】

1

【输出样例】

150

【分析】

小学奥数题，题目告诉我们第一次相遇时间是 150 秒，后面每 150 秒一个周期相遇，所以第 n 次相遇的时间为 $n * 150$ 。

【代码】

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n;
5.     cin>>n;
6.     cout<<n*150;
7. }

```

青蛙跳荷叶

【问题描述】

在一个荷塘里，有 8 个荷叶按照正八边形的形状生长着，我们给荷叶依次编号为 0——7。在荷叶 0 的位置上，有一只青蛙，青蛙可以每次沿顺时针方向跳过 a 个荷叶而停在下一个荷叶上，请问青蛙经过 k 次跳跃后，它会停在哪朵荷叶上。

【输入格式】

两个整数，a 和 k，其中 $0 < a \leq 5$, $0 < k \leq 100$ 。

【输出格式】

一个整数，表示青蛙最终所在荷叶的编号。

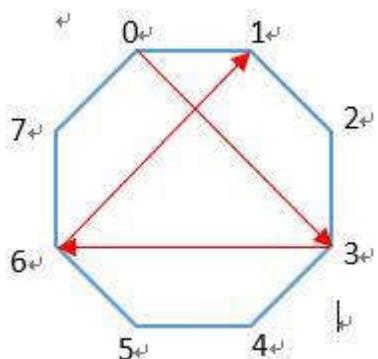
【输入样例】

2 3

【输出样例】

1

【注释】



【分析】

此题有点思维难度。不妨假设青蛙在一条线上跳跃，则第 k 次跳跃所在的距离为 $(a+1) * k$ ，现在把线变成一个环，则所在位置在 0——7 的周期上，即 $(a+1) * k \% 8$ 。考点是对取余符号的认识。

【代码】

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,k;
5.     cin>>a>>k;

```

```
6.     cout<<(a+1)*k%8;
7. }
```

小矮人

【问题描述】

最初出现“七个小矮人”的是德国著名童话集《格林 童话》之中的《白雪公主》。原文讲述了一个可爱 美丽的公主因为被恶毒后母嫉妒其美貌而被迫逃 到森林，在缘分安排下偶遇善良的七个小矮人。最后在他们帮助下，破解了后母的诅咒，找到了王子 的真爱的故事。七个小矮人的姓名分别是： 万事通、害羞鬼、瞌睡虫、喷嚏精、开心果、迷糊鬼、爱生气。白雪公主经常为这七个小矮人讲故事。白雪公主还为这七个小矮人安排了学号，学号分别是 1 至 7 号。有一次，白雪公主又邀请七个小矮人来听她讲故事，但是只来了六个小矮人，那么缺席的那个小矮人是谁呢？

【输入格式】

一行，6 个学号，空格分开，表示来听故事的六个小矮人的学号。

【输出格式】

没来听故事的小矮人的学号。

【输入样例 1】

3 5 2 1 7 4

【输入样例 2】

7 3 2 4 1 6

【输出样例 1】

6

【输出样例 2】

5

【分析】

可以根据数学方法得，1 至 7 的总和为 28，将 28 减去已出现的 6 个数，则剩下的是最后一个没出现的数。涉及多个变量的读入。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a1,a2,a3,a4,a5,a6,sum;
5.     cin>>a1>>a2>>a3>>a4>>a5>>a6;
6.     cout<<28-a1-a2-a3-a4-a5-a6;
7. }
```

温度转换

【问题描述】

编一程序，将摄氏温度换为华氏温度。公式为： $f=9/5*c+32$ 。其中 f 为华氏温度，c 是摄氏温度。

【输入格式】

输入一行，只有一个整数 c。

【输出格式】

输出只有一行，包括 1 个实数。(保留两位小数)

【输入样例】

50

【输出样例】

122.00

【注释】

如何在 C++ 输出中保留两位小数。以下为演示代码：

```
1. #include<iostream>
2. #include<iomanip> //setprecision 和 fixed 函数所需的头文件
3. using namespace std;
4. int main(){
5.     double a;
6.     cin>>a;
7.     cout<<setprecision(2)<<fixed<<a; //输出实数变量 a 且保留 2 位小数
8. }
```

【分析】

本题主要考验对浮点型数据的处理。

【代码】

```
1. #include<iostream>
2. #include<iomanip>
3. using namespace std;
4. int main(){
5.     double f,c;
6.     cin>>c;
7.     f=9.0/5*c+32; //思考，这里能否写成 9/5*c+32
8.     cout<<setprecision(2)<<fixed<<f; //保留两位小数
9. }
```

小明买箱子

【问题描述】

放暑假了，小明准备去旅游，当然啦，旅行之前要整理行李。于是，小明紧锣密鼓地准备买行李箱装行李（网购形式），但是，他看到卖家都是以英寸来标示行李箱的大小，这让小明很头痛。在他知道 1 英寸=2.54 厘米后，他决定编写一个程序，将他需要的尺寸（厘米）转化成英寸，以便在网上选到合适的箱子。

【输入格式】

一个整数，表示多少厘米，数据小于等于 1000000000。

【输出格式】

一个实数，保留小数点后两位，表示转化后的英寸。

【输入样例】

100

【输出样例】

39.37

【分析】

本题与上题相似，考验对公式的逆推。

【代码】

```
1. #include<iostream>
2. #include<iomanip>
3. using namespace std;
4. int main(){
5.     double cm,inch;
6.     cin>>cm;
7.     inch=cm/2.54;
8.     cout<<setprecision(2)<<fixed<<inch;
9. }
```

大象喝水

【问题描述】

一只大象口渴了，要喝 20 升水才能解渴，但现在只有一个深 h 厘米，底面半径为 r 厘米的小圆桶(h 和 r 都是整数)。问大象至少要喝多少桶水才会解渴。

【输入格式】

输入有一行：包含两个整数，以一个空格分开，分别表示小圆桶的深 h 和底面半径 r ，单位都是厘米。

【输出格式】

输出一行，包含一个整数，表示大象至少要喝水的桶数。

【输入样例】

23 11

【输出样例】

3

【注释】

如果一个圆桶的深为 h 厘米，底面半径为 r 厘米，那么它最多能装 $\pi * r * r * h$ 立方厘米的水。(设 $\pi=3.14159$)

1 升 = 1000 毫升

1 毫升 = 1 立方厘米

【分析】

本题主要涉及到实型数据与整型的转换。

【代码】

```
1. #include<iostream>
2. using namespace std;
```

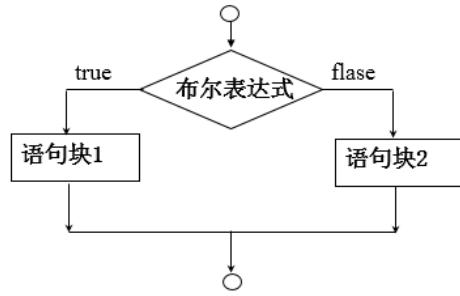
```
3. int main(){
4.     double h,r;
5.     int ans;
6.     cin>>h>>r;
7.     ans=20*1000/(r*r*3.14159*h); //将实型数据赋值给整型，去除小数
8.     cout<<ans+1;
9. }
```

if 语句

在前面，我们学习了赋值语句、输入语句、输出语句，现在，我们来一个新的语句：if 语句。程序中如果没有 if 语句，我们的程序就不能进行逻辑判断，或者说，只有结合 if 语句，才能使我们的程序显得智能一点。

if 语句的格式

```
if(布尔表达式) {
    语句块 1;
} else {
    语句块 2;
}
```



if 语句格式说明：

- 1、if 语句中的布尔表达式可以是由关系运算符连结的关系表达式，也可以是逻辑运算符连结的逻辑表达式，这两类表达式的结果类型都是布尔类型，其值不是 true 就是 false。特别注意的，在 C++ 中，数值也可以转换成布尔类型，0 转换为 false，非 0 转换为 true。
- 2、如果括号中布尔表达式的值为 true，则执行“语句块 1”，如果括号中布尔表达式的值为 false，则执行“语句块 2”，因为布尔表达式的值不可能既为 true 又为 false，所以语句块 1 和语句块 2 不可能同时被执行，两者只可能执行其一。

例题 1：输入一个整数，判断它是否为偶数，如果是，则输出“yes”，否则输出“no”。

分析：学习 if 语句的第一个难点是如何写出正确的布尔表达式。在这一题中，关键在于如何写出表示一个整数 a 是偶数的表达式。根据偶数的定义，能够被 2 整数的数即为偶数，所以不难得出该表达式为 $a \% 2 == 0$ ，有了这个表达式，套进 if 语句就成。

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a;
5.     cin>>a;
6.     if(a%2==0){
7.         cout<<"yes";
8.     }else{
9.         cout<<"no";
10.    }
11. }
```

练习：

- ▶ 1、输入一个数，如果该数可以被 3 整除或者可以被 5 整除，则输出“YES”，否则输

出“NO”。

- ▶ 2、输入一个年份，判断是不是一个闰年，如果是闰年，则输出“leap year”，否则输出“nonleap year”。
- ▶ 3、输入三条边 a, b, c ，判断其能否组成三角形，如果可以则输出 yes，否则输出 no。

讲解：

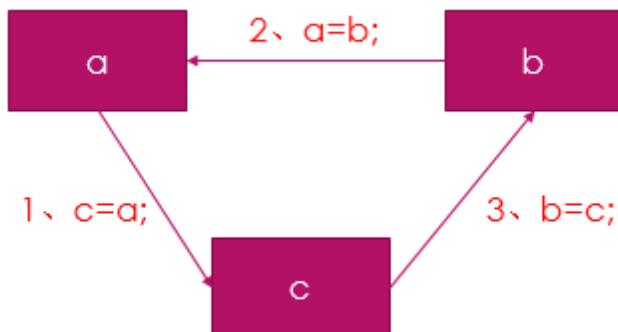
1、表示一个数 a 可以被 3 整除或者可以被 5 整除的表达式： $a \% 3 == 0 || a \% 5 == 0$

2、一个年份 y 是闰年的表达式： $y \% 4 == 0 \& \& y \% 100 != 0 || y \% 400 == 0$

3、判断三条边能否组成三角形的表达式： $a + b > c \& \& a + c > b \& \& b + c > a$

例题 2：输入两个数 a, b ，将大数存于 a 中，小数存于 b 中，再输出变量 a, b 的值。

分析：本题的要点在于掌握如何交换两个变量的值。不能将本题理解成输入两个数，先输出大数再输出小数。如何交换两个变量的值呢？首先要明确变量的两个特点，第一是变量的唯一性，一个变量同一时间只能保存一个数据，该变量一旦被赋值，则之前的变量会被擦写掉，第二是变量的复制性，一个变量赋值给其他变量，并不会改变这个变量的值。在这里交换两个变量，借助第三个变量，为了便于理解，我们打个比方，交换两个瓶子中的液体。这里设变量 a, b, c 表示三个瓶子，瓶子中的液体表示这个变量的值，交换 a, b 的液体，我们可以先将 a 中的液体倒到 c 中 ($c=a$)，然后将 b 中的液体倒到 a ($a=b$)，最后将 c 中液体倒到 b 中 ($b=c$)，这样就实现了 a, b 两个瓶子中的液体进行交换。如下图：



那么，本题的判断则是，当 $a < b$ 时，进行两个变量的交换。

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,b,c;
5.     cin>>a>>b;
6.     if(a<b){ //关系表达式
7.         c=a; //以下三个语句是一个整体，用{}包围起来，表示交换两个变量的操作
8.         a=b;
9.         b=c;
10.    }
11.    cout<<a<<" "<<b<<endl;
12. }
  
```

提示：交换两个变量的方法有多种，借助第三个变量实现是一种比较容易理解的方法，也可以不用借助多余变量，如以下三个语句同样实现了两个变量的交换，你能理解吗？

```
1. a=a+b;  
2. b=a-b;  
3. a=a-b;
```

练习：

1、读入三个不同的整数，编程按由小到大的顺序排列打印出来。

2、某车站行李托运收费标准是：10 公斤或 10 公斤以下，收费 2.5 元，超过 10 公斤的行李，按每超过 1 公斤增加 1.5 元进行收费。试编一程序，输入行李的重量，算出托运费（结果保留两位小数）。

输入样例 1: 5 输出样例 1: 2.50
输入样例 2: 11 输出样例 2: 4.00

3、某超市举行购物优惠活动：所购物品不超过 100 元时，按九折付款，如超过 100 元，超过部分按六折收费。请你编一程序完成超市自动计费的工作。输入只有一个整数 M，表示购物的款数。输出打折后的款数（结果保留两位小数）。

输入样例 1: 23.31 输出样例 1: 20.98
输入样例 2: 120 输出样例 2: 102.00

讲解：

1、第一题有两种做法，其一是将三个数的大小排列枚举出来，然后逐一输出。第二种做法是进行变量比较，先将 a 与 b 和 c 比较，如果 a 比 b 大则交换变量的值，如果 a 比 c 大则交换变量的值，两次比较后，使得 a 是三个数中最小的那个，接着将 b 和 c 比较，如果 b 比 c 大，则交换 b 和 c 的值，使得 b 比 c 小，最后输出 a、b、c。推荐使用第二种方法做。以下是参考代码：

方法一：

```
1. #include<iostream>  
2. using namespace std;  
3. int main(){  
4.     int a,b,c;  
5.     cin>>a>>b>>c;  
6.     if(a<b&&b<c)cout<<a<<" "<<b<<" "<<c;  
7.     if(a<c&&c<b)cout<<a<<" "<<c<<" "<<b;  
8.     if(b<a&&a<c)cout<<b<<" "<<a<<" "<<c;  
9.     if(b<c&&c<a)cout<<b<<" "<<c<<" "<<a;  
10.    if(c<a&&a<b)cout<<c<<" "<<a<<" "<<b;  
11.    if(c<b&&b<a)cout<<c<<" "<<b<<" "<<a;  
12. }
```

方法二：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,b,c,t;
5.     cin>>a>>b>>c;
6.     if(a>b){
7.         t=a;
8.         a=b;
9.         b=t;
10.    }
11.    if(a>c){
12.        t=a;
13.        a=c;
14.        c=t;
15.    }
16.    if(b>c){
17.        t=b;
18.        b=c;
19.        c=t;
20.    }
21.    cout<<a<<" "<<b<<" "<<c;
22. }
```

2、第二题参考代码：

```
1. #include<iostream>
2. #include<iomanip>
3. using namespace std;
4. int main(){
5.     double w,p;
6.     cin>>w;
7.     if(w<=10){
8.         p=2.5;
9.     }else{
10.        p=2.5+(w-10)*1.5;
11.    }
12.    cout<<setprecision(2)<<fixed<<p;
13. }
```

3、第三题参考代码：与上题代码类似。

if 语句的嵌套

在 if 语句中，如果语句块 1 或语句块 2 的语句又是一个 if 语句，叫做 if 语句的嵌套。

形式 1:

```
if(布尔表达式){
    if(布尔表达式){}else{}
}else{
    语句块;
}
```

形式 2:

```
if(布尔表达式){
    语句块;
}else{
    if(布尔表达式){}else{}
}
```

if 语句的嵌套可以满足一些复合条件的判断，在多重 if 语句嵌套的时候，要特别注意其隐含的逻辑判断条件。

例题 3： 编程计算函数 $f(x)$ 的值。

$$f(x) = \begin{cases} 10, & x = 1 \\ 40, & x = 2 \\ 50, & x! = 1 \text{ 和 } x! = 2 \end{cases}$$

分析：本题涉及三个判断，我们可以用三个判断语句来写。

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int x;
5.     cin>>x;
6.     if(x==1)cout<<10;
7.     if(x==2)cout<<40;
8.     if(x!=1&&x!=2)cout<<50;
9. }
```

也可以用 if 语句的嵌套来写，注意使用 if 语句嵌套时，一些逻辑条件被隐形包含了。

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int x;
5.     cin>>x;
6.     if(x==1){
7.         cout<<10;
8.     }else{ //以下包含 x!=1 的情况
9.         if(x==2){
10.             cout<<40;
11.         }else{ //在 x!=1 的情况下且 x!=2 的情况，条件被隐式包含了
12.             cout<<50;
13.         }
14.     }
```

```
15. }
```

练习：

- 输入某学生的成绩，若成绩在 85 分以上，输出 `very good`，若成绩在 60 分到 85 分之间，输出 `good`，若成绩低于 60 分，输出 `no good`。
- 输入两个正整数 `a`，`b`。`b` 最大不超过三位数，`a` 不大于 31。使 `a` 在左，`b` 在右，拼接成一个新的数 `c`。例如：`a=2, b=16`，则 `c=216`；若 `a=18, b=476`，则 `c=18476`。
提示：求 `c` 的公式为：`c=a×K+b`，其中：

$$K = \begin{cases} 10 & \text{当 } B \text{ 为一位数时}(0 < b < 10) \\ 100 & \text{当 } B \text{ 为二位数时}(10 \leq b < 100) \\ 1000 & \text{当 } B \text{ 为三位数时}(100 \leq b < 1000) \end{cases}$$

输入样例： 2 16

输出样例： 216

讲解：

- 第一题参考代码，这里使用嵌套写法，注意 `if` 语句包含的隐式条件。

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int score;
5.     cin>>score;
6.     if(score>85){
7.         cout<<"very good";
8.     }else{
9.         if(score>=60){
10.             cout<<"good";
11.         }else{
12.             cout<<"no good";
13.         }
14.     }
15. }
```

- 第二题参考代码：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,b,c,k;
5.     cin>>a>>b;
6.     if(0<b&&b<10)k=10;
7.     if(10<=b&&b<100)k=100;
```

```
8.     if(100<=b&&b<1000)k=1000;  
9.     c=a*k+b;  
10.    cout<<c;  
11. }
```

=====网站练习=====

找零钱

【题目描述】

现在假设你是个店员,为了方便、准确、最优的找零钱,你设计了一个程序。该程序应该实现如下功能:

第一行输入客户所给你金额

第二行输入客户消费的总金额

第三行输出应找的总零钱是多少。如果客户给的钱不够, 应该输出 ‘Not Enough!’ (不含引号部分)

注:为了简单,假设上述中的金额都是整数, 范围[1..100000000]。

【输入】

第一行输入一个整数(表示客户所付的金额),如 100

第二行输入一个整数(表示商品的总计金额),如 25

【输出】

第一行输出 应找的零钱,如 75

【样例输入 1】

100

25

【样例输入 2】

95

100

【样例输出 1】

75

【样例输出 2】

Not Enough!

【题目讲解】

本题是 if 语句的模板题, 设客户的钱为 m, 客户消费的钱为 p, 根据表达式 $m \geq p$ 进行判断, 特别注意, 不能少了 = 的条件。

【代码】

```
1. #include<iostream>  
2. using namespace std;  
3. int main(){  
4.     int m,p;  
5.     cin>>m>>p;  
6.     if(m>=p){
```

```
7.         cout<<m-p;
8.     }else{
9.         cout<<"Not Enough!";
10.    }
11. }
```

判奇偶

【题目描述】

小洪对偶数很感兴趣，现在有一个整数，他想对这个数进行如下操作：如果这个数是偶数的话就除以 2，如果这个数是奇数的话就乘以 2，那最后的结果是多少呢？

【输入】

只有 1 个正整数（小于 1000000）。

【输出】

一个正整数，表示处理后结果。

【样例输入】

90

【样例输出】

45

【题目讲解】

本题跟练习题判断一个数是否为偶数类似，不同在于如果是偶数则将该数除以 2，否则乘以 2。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a;
5.     cin>>a;
6.     if(a%2==0){
7.         cout<<a/2;
8.     }else{
9.         cout<<a*2;
10.    }
11. }
```

小米

【题目描述】

小米同学现在读四年级，小米同学想知道自己成年后的身高大概是多少。于是小米同学上网查找资料，终于找到了一条计算公式：

- 1、如果小米是男生，那么成年后身高 = （父亲身高+母亲身高+13 厘米） div 2
- 2、如果小米是女生，那么成年后身高 = （父亲身高+母亲身高-13 厘米） div 2

温馨提示：题目中的 `div` 是表示整除，`A div B` 表示的意义是 `A` 除以 `B` 的商，忽略余数。例如：`10 div 2 = 5`，因为 `10` 除以 `2` 的商是 `5`。`9 div 2 = 4`，因为 `9` 除以 `2` 的商是 `4`。因此，本题只需要用到整型，不要用实型。

【输入】

一行，三个整数：`father`、`mother`、`me`。其中 `father` 是父亲身高，`mother` 是母亲身高，`me` 如果是 `1`，则代表小米是男生；`me` 如果是 `0`，则代表小米是女生。

【输出】

一个整数，表示小米同学成年后的身高。

【输入样例 1】

174 162 0

【输入样例 2】

180 165 1

【输出样例 1】

161

【输出样例 2】

179

【样例解释 1】

父亲身高 `174`，母亲身高 `161`，小米是女生，因此身高是 $(174+162-13) \text{ div } 2 = 323 \text{ div } 2 = 161$

【样例解释 2】

父亲身高 `180`，母亲身高 `165`，小米是男生，因此身高是 $(180+165+13) \text{ div } 2 = 358 \text{ div } 2 = 179$

【题目讲解】

本题增加了输入数据，只需根据性别输出不同计算身高的表达式即可。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int father,mother,me;
5.     cin>>father>>mother>>me;
6.     if(me==0){
7.         cout<<(father+mother-13)/2;
8.     }else{
9.         cout<<(father+mother+13)/2;
10.    }
11. }
```

小明坐车

【题目描述】

小明的生活非常忙碌，有时为了赶时间不得不打的。细心的他发现广州市出租车公司规定是：2.5 公里及 2.5 公里以内为起步价 10 元，若超过 2.5 公里，超过部分按每公里 2.6 元收费。为了验算这个规定，他决定自己计算车费。

【输入】

输入一个实数，表示小明的乘坐路程

【输出】

输出一个整数，表示总共的车费。如果路费为小数，则四舍五入到个位。

【样例输入】

5.8

【样例输出】

19

【题目讲解】

本题与课堂练习题目类似。需要注意地方在于输入是一个实数，不能用整型存储。其次是如何四舍五入到个位。可以利用数学函数来实现，也可以简易实现。以下代码中，我们加结果加上 0.5，然后将整个数值赋值个整型，则自动去除小数部分，因为加上了 0.5，则可以达到四舍五入的效果。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     double d;
5.     int p;
6.     cin>>d;
7.     if(d<=2.5){
8.         p=10;
9.     }else{
10.        p=10+(d-2.5)*2.6+0.5;
11.    }
12.    cout<<p;
13. }
```

求天数

【题目描述】

输入年份和月份，求当月有多少天。注意，闰年的判断方法为能被 4 整除且不能被 100 整除，或者能被 400 整除。

【输入】

输入两个数 yy 和 mm，代表年份和月份。

【输出】

输出当月的天数。

【样例输入】

2010 2

【样例输出】

28

【分析】

本题重在考察 if 语句的嵌套。对每个月进行分别判断其天数即可。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int y,m,d;
5.     cin>>y>>m;
6.     if(m==1||m==3||m==5||m==7||m==8||m==10||m==12){
7.         d=31;
8.     }
9.     if(m==4||m==6||m==9||m==11){
10.        d=30;
11.    }
12.    if(m==2){
13.        if(y%4==0&&y%100!=0||y%400==0){
14.            d=29;
15.        }else{
16.            d=28;
17.        }
18.    }
19.    cout<<d;
20. }
```

成绩分级

【题目描述】

信息技术期末考试完了，学校要求根据学生的考试成绩 M 分为 A、B、C、D、E 四级，其中对应关系为：A 级 ($90 \leq M \leq 100$)、B 级 ($80 \leq M < 90$)、C 级 ($70 \leq M < 80$)、D 级 ($60 \leq M < 70$)、E 级 ($M < 60$)。

【输入】

输入一个实数，表示学生的成绩 M ($M \leq 100$)。

【输出】

输出 学生等级（用大写字母表示）。

【样例输入】

输入样例 1:

96

输入样例 2:

59

【样例输出】**【输出样例】**

输出样例 1:

A

输出样例 2:

E

【分析】

本题同样考察 if 语句的嵌套。可以逐层嵌套解决，如代码 1；也可以分多条语句判断，如代码 2。

【代码 1】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     double m;
5.     cin>>m;
6.     if(m>=90){
7.         cout<<"A";
8.     }else{
9.         if(m>=80){
10.             cout<<"B";
11.         }else{
12.             if(m>=70){
13.                 cout<<"C";
14.             }else{
15.                 if(m>=60){
16.                     cout<<"D";
17.                 }else{
18.                     cout<<"E";
19.                 }
20.             }
21.         }
22.     }
23. }
```

【代码 2】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     double m;
5.     cin>>m;
6.     if(m>=90&&m<100)cout<<"A";
7.     if(m>=80&&m<90)cout<<"B";
8.     if(m>=70&&m<80)cout<<"C";
9.     if(m>=60&&m<70)cout<<"D";
10.    if(m<60)cout<<"E";
11. }
```

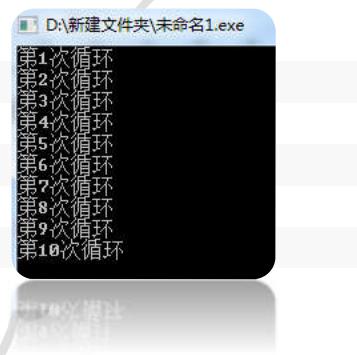
for 语句 1——认识 for 语句

计算机最擅长的事情是可以快速重复地做一系列运算，要发挥计算机这个特长，我们需要用到编程中的循环语句，其中 **for** 循环语句是我们使用最频繁的语句之一。我们先来认识下 **for** 语句的格式：

```
for (控制变量初始化表达式; 条件表达式; 增量表达式) {
    语句块;
}
```

语句格式稍显复杂，我们通过例子来认识这个格式，请先编写以下代码：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     for(int i=1;i<=10;i++){
5.         cout<<"第"<<i<<"次循环"<<endl;
6.     }
7. }
```



在上述代码中，特别关注 **for** 语句括号中的三个表达式。

① **控制变量初始化表达式:** `int i=1`

在这里定义一个循环控制变量（虽说是循环控制变量，其实就是一个变量）`i`，并给 `i` 赋值为 1，这是循环的初始值，初始值也可以是其他任意整数值。

② **条件表达式:** `i<=10`

这是循环的结束条件。当表达式 `i<=10` 的值为 `true` 时，继续循环，直到 `i<=10` 的值为 `false`，则退出循环。

③ **增量表达式:** `i++`

这是循环变量的增加方式。每循环一次 `i` 增加 1。`i++` 是自增语句，等价于 `i=i+1`（这种形式的赋值叫做累加语句）。当然，我们也可以改变变量 `i` 每次增加的值，比如 `i=i+2`。

④ **循环语句块:** `cout<<"第"<<i<<"次循环"<<endl;`

循环语句块是每循环一次就会执行一次的语句块，上述代码中，这个语句块被执行了 10 次。

把循环拆解成上述四个部分（用①②③④表示），其执行顺序是这样的：

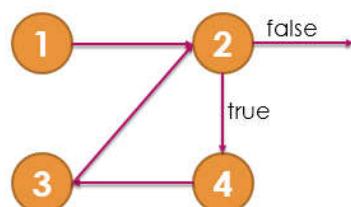
1、执行①，定义循环控制变量，并赋初值。

2、执行②，判断表达式的值，如果表达式的值为 `true`，继续执行循环语句块④，如果值为 `false`，退出循环。

3、执行④，执行循环语句块。

4、执行③，循环控制变量自增。

5、回到第 2 步，重新执行②



思考：改造程序，使其输出效果如下图：



参考代码 1:

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     for(int i=5;i<=15;i++){ //改变循环的初值和终值
5.         cout<<"第"<<i<<"次循环"<<endl;
6.     }
7. }
```

参考代码 2:

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     for(int i=1;i<=20;i=i+2){ //改变循环的增量方式
5.         cout<<"第"<<i<<"次循环"<<endl;
6.     }
7. }
```

例题: 计算 $1+2+3+\dots+99+100$ 的和

分析:

在小学阶段, 我们求类似的等差数列, 会用到高斯的等差公式, 利用(首项+末项)*公差/2来直接计算。这是方法一。在编程中, 我们可以用最直觉的方式来操作, 让每一项值都加起来。问题是, 我们怎么来表示1到100这100个数呢? 利用循环, `for(int i=1;i<=100;i++)`, 在这个语句中, 循环变量*i*的初值为1, 然后它每循环一次就会加1, 一直循环到100, 所以这里的*i*就可以表示1到100这100个数。那么, 每得到这么一个数, 我们应该怎么将这个数加起来呢? 我们设一个变量`sum`, 赋值为0, 每得到一个*i*, 我们将*i*的值累加到`sum`中, 即`sum=sum+i`, `sum+i`的意思是将`sum`(即前*i*-1项的和)加上*i*, 所得结果重新存储到`sum`中, 这时的`sum`其实表示的就是前*i*项的和。

`sum=sum+i` 这种形式的语句我们叫做累加语句，`sum` 这样的变量我们叫做累加器，累加语句还可以用 `sum+=i` 的语句形式表示。由此推广，还有累减，累乘、累除。

原句	等价于
<code>a=a+b;</code>	<code>a+=b;</code>
<code>a=a-b;</code>	<code>a-=b;</code>
<code>a=a*b</code>	<code>a*=b;</code>
<code>a=a/b</code>	<code>a/=b;</code>

对于每次自增值为 1 的情况，即 `a=a+1`，可以写成 `a+=1`，但在编程中，我们更习惯写成 `a++`，表示 `a` 自增 1。对应的，也有 `a--`，表示 `a` 自减 1。

代码：

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int sum=0;
5.     for(int i=1;i<=100;i++){
6.         sum+=i;
7.     }
8.     cout<<sum;
9. }
```

思考：

- 1、将上述例题修改为输入 `n`，求出 $1+2+3+\dots+n$ 的和。
- 2、将上述例题修改为输入 `n`，求出 $1+2+3+\dots+n$ 的奇数和。

参考代码 1：

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,sum=0;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){ //注意条件表达式的写法
7.         sum+=i;
8.     }
9.     cout<<sum;
10. }
```

参考代码 2：

```

1. #include<iostream>
2. using namespace std;
```

```
3. int main(){
4.     int n,sum=0;
5.     cin>>n;
6.     for(int i=1;i<=n;i+=2){ //注意增量表达式的用法
7.         sum+=i;
8.     }
9.     cout<<sum;
10. }
```

练习：

1、输入 N，求 N! ($N!=1*2*3*\dots*N$) ，这里 N 不大于 10，输出 N! 的值

2、编程计算 $12+22+32+\dots+1002$

3、编程计算并输出 $1+1/2+1/3+\dots+1/100$ 的和

参考代码 1：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,sum=1;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         sum*=i; //累乘
8.     }
9.     cout<<sum;
10. }
```

参考代码 2：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,sum=0;
5.     for(int i=12;i<=1002;i+=10){ //注意循环的初值、终值和增量
6.         sum+=i;
7.     }
8.     cout<<sum;
9. }
```

参考代码 3：

```
1. #include<iostream>
```

```
2. using namespace std;
3. int main(){
4.     int n;
5.     double sum=0; //累加器定义为浮点型
6.     for(int i=1;i<=100;i++){
7.         sum+=1.0/i; // 1.0/i 表示第 i 项的值
8.     }
9.     cout<<sum;
10. }
```

=====网站练习=====

立方和

【题目描述】

我们先研究如下规律：

$$\begin{aligned}1^3+2^3 &= 9 = 3^2 = (1+2)^2 \\1^3+2^3+3^3 &= 36 = 6^2 = (1+2+3)^2 \\1^3+2^3+3^3+4^3 &= 100 = 10^2 = (1+2+3+4)^2 \\\dots\dots\end{aligned}$$

现在要你求 $1^3+2^3+3^3+4^3+\dots\dots+n^3$ 的值。

【输入】

一个整数 $n(n \leq 300)$.

【输出】

一个整数

【样例输入】

4

【样例输出】

100

【分析】

本题有两种做法，一种是按 $1^3+2^3+3^3+4^3+\dots\dots+n^3$ 来求，第二种是求出前 n 项的和，再输出和的平方。

【代码】

方法 1：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,sum=0;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         sum+=i*i*i; //累加 i 的 3 次方
8.     }
```

```
9.     cout<<sum;
10. }
```

方法 2：

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

小明的阶乘

【题目描述】

大家都知道阶乘这个概念，举个简单的例子： $5! = 1*2*3*4*5$ 。但是，小明是个标新立异的人，他准备重新定义新的阶乘概念：将原来的每个数相乘变为 i 不大于 n 的所有奇数相乘。例如： $5!!=1*3*5$ ($5!!$ 表示新定义的阶乘)。现在你明白这种阶乘的意思了吧！

可是小明的计算能力很差，需要你帮忙计算 $n!!$ 的值。

【输入】

输入一个整数 n ($2 < N \leq 20$)

【输出】

输出一个整数，表示 $n!!$ 的值

【样例输入】

5

【样例输出】

15

【分析】

本题是求 1 至 n 之间的奇数项的乘积。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,sum=1;
5.     cin>>n;
6.     for(int i=1;i<=n;i+=2){
7.         sum*=i;
8.     }
9.     cout<<sum;
```

```
10. }
```

国王与麦子

【题目描述】

传说古代印度有个喜欢下棋的国王叫舍罕，而宰相达依尔是个聪明的大臣，发明了国际象棋。国王玩得爱不惜手，决定奖赏宰相。达依尔说：陛下，我别无他求，请你在这张棋盘的第一个格子里赏我一粒麦子；在第 2 个格子里赏我 2 粒麦子；在第 3 个格子里赏我 4 粒麦子；在第 4 个格子里赏我 8 粒麦子……依此类推直到 64 个格子，按这张棋盘上各格应赏的麦子全赏给我吧。

国王听了，觉得达依尔的要求并不高，说道：你能如愿以偿的。然而，国王却不知道这个数字是多么巨大啊！

你能帮助国王算算第 n 个格子的麦子数量吗？

【输入】

输入正整数 n ($n < 60$)

【输出】

输出第 n 个格子的麦子数量，注意不能以科学记数法表示。

【样例输入】

输入样例一：

5

输入样例二：

40

【样例输出】

输出样例一：

16

输出样例二：

549755813888

【分析】

本题是求 2 的 n 次方。只需要循环 n 次，每次累乘 2 就行。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     long long x=1,n;
5.     cin>>n;
6.     for(int i=2;i<=n;i++){
7.         x*=2;
8.     }
9.     cout<<x;
10. }
```

流感

【题目描述】

流感来了！流感有很强的传染性，一个流感病患每轮会传染 x 个人。假设现在有一个人得了流感，试问 n 轮传染后有多少人被传染了？

【输入】

两个数 x 和 n 。 $0 < x, n \leq 10$ 。

【输出】

一个数：被传染的人数。

【样例输入】

10 2

【样例输出】

121

【分析】

假设人数为 p ，则新一轮感染的总人数为 p （已感染的人数） $+p*x$ （新感染的人数），所以新的感染总人数 $p=p+p*x$ 。循环处理 n 天就可以了。注意， p 应该设置为 long long 类型。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     long long x,n,p=1;
5.     cin>>x>>n;
6.     for(int i=1;i<=n;i++){
7.         p=p+p*x;
8.     }
9.     cout<<p;
10. }
```

猴子吃桃问题

【题目描述】

有一堆桃子不知数目，猴子第一天吃掉一半，又多吃了1个，第二天照此方法，吃掉剩下桃子的一半又多一个，天天如此，到第 m 天早上，猴子发现只剩一只桃子了，问这堆桃子原来有多少个？

【输入】

输入一个整数 m ($m < 29$)

【输出】

输出桃子的总个数

【输入样例 1】 3

【输入样例 2】 11

【输出样例 1】 10

【输出样例 2】 3070

【分析】

本题可以逆推不好实现，可以用递推的方式实现，即猴子一开始有 1 个桃子，第二天有 $(1+1) * 2 = 4$ 个桃子，第三天又有 $(4+1) * 2$ 个桃子，一直推到第 m 天，可以用循环递推实现。

【代码】

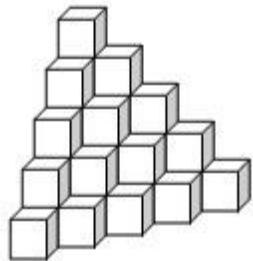
```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int m,s=1;
5.     cin>>m;
6.     for(int i=2;i<=m;i++){
7.         s=(s+1)*2;
8.     }
9.     cout<<s;
10. }
```

小明数木块

【题目描述】

小明家里堆放着一堆完全相同的正方体小木块，如右图所示：



因为木块堆得实在是太有规律了，所以只要知道它的层数就可以计算所有木块的数量了。但是，小明的计算能力真得很弱，所以，他有得再请你写个程序帮忙：给你一堆木块的层数，求出这堆木块的数量。

【输入】

只有一个整数 $n(1 \leq n \leq 20)$ ，表示这堆小木块的层数

【输出】

输出一个整数，表示这堆小木块的总数量

【样例输入】

5

【样例输出】

35

【样例注释】 第 1 层 1 个，第 2 层 3 个，第 3 层 6 个，第 4 层 10 个，第 5 层 15 个，所以结果是 $1+3+6+10+15=35$

【分析】

本题可以用三种方法来做。

方法 1：将每一层展开，第一层有 1 个，第二层有 $1+2$ 个，第三层有 $1+2+3$ 个，所以总和有 $1+(1+2)+(1+2+3)+\dots+(1+2+3+\dots+n)$ 个，该式子等价于 $1*n+2*(n-1)+3*(n-2)+\dots+n*1$ ，

那么，对于第 i 项，它的值为 $i^*(n-i+1)$ ，循环 n 次，每次累加 $i^*(n-i+1)$ 即可，详见代码 1。

方法 2：可以用双层循环来做，外循环枚举每一层，对于第 i 层，循环计算这一层的个数，即求 $1+2+3+\dots+i$ 的和，然后再累加起来，详见代码 2。

方法 3：对方法 2 进行优化，循环枚举每一层，对于第 i 层，设上一层的方块数为 t ，则这一层的方块数为 $t+i$ ，将 $t+i$ 赋值给 t ，进行迭代计算即可。详见代码 3。

【代码】

代码 1：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,sum=0;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         sum+=i*(n-i+1);
8.     }
9.     cout<<sum;
10. }
```

代码 2：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,sum=0;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         for(int j=1;j<=i;j++){
8.             sum+=j;
9.         }
10.    }
11.    cout<<sum;
12. }
```

代码 3：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,sum=0,t=0;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         t+=i;
8.         sum+=t;
```

```
9.      }
10.     cout<<sum;
11. }
```

oiClass

for 语句 2——枚举+筛选

使用 **for** 语句，我们可以很容易的枚举任意区间的整数，我们可以对这些整数进行累加、累乘等操作，但是，很多时候，并不是所有的整数都符合我们的要求，往往我们需要对枚举的整数进行筛选。本节课，我们主要来认识 **for** 语句的一种常用用法，枚举+筛选，利用 **for** 语句来枚举任意区间的整数，利用 **if** 语句进行筛选，常见格式如下：

```
for (枚举任意区间整数 i) {  
    if(i 是符合条件的整数){  
        对 i 进行操作（累加、计数、输出）;  
    }  
}
```

例题 1: 编程输出 1000 以内（包括 1000）所有能被 7 整除的自然数。

分析：用 **for** 语句可以很容易枚举出 1 至 1000 之间的所有整数 *i*，但并不是所有的整数 *i* 都能被 7 整除，所以在可以使用选择语句 **if** 来判断当前的 *i* 是否能被 7 整除，如果可以，则输出，如果不可以，则进入下一次循环。

代码：

```
1. #include<iostream>  
2. using namespace std;  
3. int main(){  
4.     for(int i=1;i<=1000;i++){  
5.         if(i%7==0){  
6.             cout<<i<<" ";  
7.         }  
8.     }  
9. }
```

【练习】

- 1、在自然数中，如果一个三位数等于自身各位数字之立方和，则这个三位数就称为是水仙花数。如: $153=1^3+5^3+3^3$ ，所以 153 是一个水仙花数。输出所有的水仙花数。
- 2、一个两位数 *x*，将它的个位数字与十位数字对调后得到一个新数 *y*，此时 *y* 恰好比 *x* 大 36，请编程求出所有这样的两位数。

【参考代码 1】

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

```

4.     for(int i=100;i<1000;i++){
5.         int ge=i%10;
6.         int shi=i/10%10;
7.         int bai=i/100;
8.         if(ge*ge*ge+shi*shi*shi+bai*bai*bai==i){
9.             cout<<i<<" ";
10.        }
11.    }
12. }
```

【参考代码 2】

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     for(int x=10;x<100;x++){
5.         int ge=x%10;
6.         int shi=x/10;
7.         int y=ge*10+shi;
8.         if(y-x==36){
9.             cout<<x<<" ";
10.        }
11.    }
12. }
```

例题 2: 编程输入一个自然数 $x(x \leq 10000)$, 求这个自然数的所有约数（包括 1 和 x 本身）的个数。

分析：对于输入整数 x , 判断其约数的方法为：枚举 1 至 x 之间的所有整数 i (思考：是不是一定要从 1 枚举到 x)。如果 x 能被 i 整数，则 i 是 x 的约数，累计其个数

代码：

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int x,ans=0;
5.     cin>>x;
6.     for(int i=1;i<=x;i++){
7.         if(x%i==0)ans++;
8.     }
9.     cout<<ans;
10. }
```

【练习】

1、输入一个整数，判断其是否为素数，如果是则输出‘YES’，不是则输出‘NO’（不含引号）

【参考代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int x,ans=0;
5.     cin>>x;
6.     for(int i=2;i<x;i++){
7.         if(x%i==0)ans++;
8.     }
9.     if(ans==0)cout<<"YES";
10.    else cout<<"NO";
11. }
```

=====网站练习=====

明明的幸运数

【题目描述】

明明是 2003 年 3 月 3 日出生的，他出生的年份最后一个数字是 3，月数、日期都是 3，他认为数字 3 是他的幸运数，甚至凡是 3 的倍数的数他都非常喜欢，认为都是他的幸运数，现在给出一个正整数 N，请你帮明明统计出 1 至 N 之间（包括 1 和 N）有几个数字是明明的幸运数。

【输入】

只有一个数 N (N<=32767)。

【输出】

只有一个数，就是 1 至 N 之间明明的幸运数的个数。

【样例输入】

10

【样例输出】

3

【分析】

本题与例题 1 类似，枚举 1 至 n 的所有整数，如果这个数能够被 3 整除，则累计个数。本题还有另一种做法，求 n 以内能被 3 整除的个数，直接 n 整除 3 即可求出。

【代码 1】

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

```
4.     int n,ans=0;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         if(i%3==0){
8.             ans++;
9.         }
10.    }
11.    cout<<ans;
12. }
```

【代码 2】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,ans=0;
5.     cin>>n;
6.     cout<<n/3;
7. }
```

玫瑰花数

【题目描述】

如果一个四位数等于它的每一位数字的 4 次方之和，则称这个四位数为玫瑰花数，例如：8208

【输入】

不用输入

【输出】

输出所有满足条件的玫瑰花数，每个数输出一行。

【分析】

本题与水仙花数类似。枚举 1000 至 9999 之间的所有整数 i，对 i 分离个位、十位、百位、千位，判断每一位数字的 4 次方之和是否等于原数即可。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     for(int i=1000;i<10000;i++){
5.         int ge=i%10;
6.         int shi=i/10%10;
7.         int bai=i/100%10;
8.         int qian=i/1000;
9.         if(ge*ge*ge*ge+shi*shi*shi*shi+bai*bai*bai*qian*qian*qian==i){
10.            cout<<i<<endl;
11.        }
12.    }
13. }
```

```
11.         }
12.     }
13. }
```

守形数

【题目描述】

求 2 到 n 之间的守形数（若某数的平方，其低位与该数相等，则称守形数。如：25 的平方等于 625，625 的低位与 25 相同，故 25 是守形数。）

【输入】

输入一个整数 n (n<=1000)。

【输出】

输出符合条件的守形数数，共一行，每两个数之间用空格隔开。

【样例输入】

10

【样例输出】

5 6

【分析】

枚举 2 至 n 之间的所有整数 i，对 i 进行分类讨论，如果 i 是 1 位数，则与该数平方的个位比较，如果 i 是 2 位数，则与该数平方的末两位数比较，如果 i 是 3 位数，则与该数平方的末三位数比较。

【代码 1】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n;
5.     cin>>n;
6.     for(int i=2;i<=n;i++){
7.         int k;
8.         if(i<10)k=10;
9.         if(i>=10&&i<100)k=100;
10.        if(i>=100&&i<1000)k=1000;
11.        if(i==i*i%k)cout<<i<<" ";
12.    }
13. }
```

【代码 2】

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

```
6.     for(int i=2;i<=n;i++){
7.         if(i==i*i%10||i==i*i%100||i==i*i%1000){
8.             cout<<i<<" ";
9.         }
10.    }
11. }
```

数字王国的二等公民

【题目描述】

数字王国“数”满为患了，国王决定清理掉数字王国里的二等公民。国王是这样定义二等公民的，对于任意数字，如果它可以由其他数字相乘获得，那么它就是二等公民，比如 8，可以由 2 和 4 相乘获得，所以 8 是二等公民。作为数字王国的警卫官，你需要灵敏的判断一个数字是否是二等公民。

【输入】

输入一个整数 n ($n \leq 30000$)

【输出】

如果该数字是二等公民，输出 “YES”，否则输出 “NO”

【样例输入】

8

【样例输出】

YES

【分析】

本题其实是判断一个数是否为素数，如果不是素数，则输出 YES，如果是，则输出 NO。判断素数除了可以通过统计该数约数的个数来判断，还可以利用一个布尔变量来标识，详见代码 2。

【代码 1】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,ans=0;
5.     cin>>n;
6.     for(int i=2;i<n;i++){
7.         if(n%i==0)ans++;
8.     }
9.     if(ans==0){
10.        cout<<"NO"<<endl;
11.    }else{
12.        cout<<"YES"<<endl;
13.    }
14. }
```

【代码 2】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n;
5.     bool flag=true;
6.     cin>>n;
7.     for(int i=2;i<n;i++){
8.         if(n%i==0)flag=false;
9.     }
10.    if(flag==true){
11.        cout<<"YES"<<endl;
12.    }else{
13.        cout<<"NO"<<endl;
14.    }
15. }
```

表达式求值

【题目描述】

请你计算以下表达式的值：

$(1-1/2)+(1/3-1/4)+(1/5-1/6)+\dots\dots+(1/(n-1)-1/n)$

【输入】

输入一个整数 n , $n \leq 1000$, n 保证是偶数。

【输出】

输出以上表达式的值, 结果保留 8 位小数。

【样例输入】

4

【样例输出】

0.58333333

【分析】

通过观察发现, 该题是将奇数项相加, 偶数项相减。枚举 1 至 n , 如果该项是奇数项, 则累减, 如果该项是偶数项, 则累减。

【代码】

```
1. #include<iostream>
2. #include<iomanip>
3. using namespace std;
4. int main(){
5.     int n;
6.     double sum=0;
7.     cin>>n;
8.     for(int i=1;i<=n;i++){
9.         if(i%2!=0){
10.             sum+=1.0/i;
```

```
11.         }else{
12.             sum-=1.0/i;
13.         }
14.     }
15.     cout<<setprecision(8)<<fixed<<sum;
16. }
```

子数整数

【题目描述】

对于一个五位数 $a_1a_2a_3a_4a_5$, 可将其拆分为三个子数:

$sub1=a_1a_2a_3$
 $sub2=a_2a_3a_4$
 $sub3=a_3a_4a_5$

例如, 五位数 20207 可以拆分成

$sub1=202$
 $sub2=020 (=20)$
 $sub3=207$

现在给定一个正整数 K , 要求你编程求出 10000 到 30000 之间所有满足下述条件的五位数, 条件是这些五位数的三个子数 $sub1$, $sub2$, $sub3$ 都可被 K 整除。

【输入】

输入仅一行, 为正整数 K ($0 < K < 1000$)。

【输出】

输出到文件, 输出文件的每一行为一个满足条件的五位数, 要求从小到大输出。不得重复输出或遗漏。如果无解, 则输出 “No”。

【样例输入】

15

【样例输出】

22555
25555
28555
30000

【分析】

枚举 10000 至 30000 之间的数 i , 分离 i 的三个子数, 判断三个子树能否同时被 k 整除, 如果可以, 则输出。对于无解的情况, 我们用一个变量 cnt 统计满足条件的个数, 如果最终 cnt 还为 0, 则说明无解。

【代码】

```
1. #include<iostream>
2. #include<iomanip>
3. using namespace std;
4. int main(){
5.     int k,cnt=0;
6.     cin>>k;
```

```
7.     for(int i=10000;i<=30000;i++){  
8.         int sub1=i/100;  
9.         int sub2=i/10%1000;  
10.        int sub3=i%1000;  
11.        if(sub1%k==0&&sub2%k==0&&sub3%k==0){  
12.            cnt++;  
13.            cout<<i<<endl;  
14.        }  
15.    }  
16.    if(cnt==0){  
17.        cout<<"No"<<endl;  
18.    }  
19. }
```

for 语句 3——多数据处理

在使用 for 语句时，我们经常会遇到一类题型，它需要输入多个数据，而这些数据可以在线处理（所谓在线处理，是指数据可以输入后即时处理，后续不用重复读取），比如以下例题：

例题 1

问题描述：输入 n 个数，求 n 个数的和，输出其结果。

输入格式：第一行一个整数 n，接下来一行输入 n 个整数

输出格式：输出 n 个整数的和

输入样例：

10

2 6 8 5 9 12 35 -6 55 -20

输出样例

106

分析：首先思考如何输入 n 个数，在这里不能用之前定义变量的方式来输入数据，因为数据会很大，比如 n 达到 1000 或者更大，我们不可能直接定义 1000 个变量，况且这个 n 现在也是不确定的。输入 n 个数，我们可以利用 for 语句来实现，循环 n 次，每次读入一个数，则完成了 n 个数的输入。那么，每读入一个数，我们应该怎么处理呢？因为本题要求总和，我们可以将每读入的一个数累加起来。而这个数，被累加之后，也没有后续的作用了，所以这个存储空间也可以用来存储下一个数。

代码：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,x,sum=0;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         cin>>x; //每次读进来的数存到 x 中
8.         sum+=x; //将 x 累加起来
9.     }
10.    cout<<sum;
11. }
```

例题 2：柏拉图的难题

【问题描述】

有一天，苏格拉底和他的学生柏拉图在田野散步，他们来到一块麦田前。苏格拉底对柏拉图说：我请你穿越这片麦田，去摘一株最大的麦穗，你只能穿越一次，不能回头。面对茫茫的麦田，柏拉图束手无策，他想到了学编程的你，请你帮助他找到最大的那一株麦穗。

【输入格式】

第一行一个整数 n ($0 < n < 10000$)，表示麦穗的数目

接下来一行有 n 个整数，每个整数 x ($0 < x < 32767$)，表示麦穗的大小。

【输出格式】

一个整数，表示能摘到的最大的麦穗。

【输入样例】

```
10  
845 15 155 862 1545 545 53 355 613 2
```

【输出样例】

```
1545
```

分析：问题是求输入 n 个数，打印出最大的数。我们可以设定变量 maxn，将其赋值为最小值。然后、循环读入 n 个数，每读入一个数 x，都将其跟 maxn 变量比较，如果 $x > maxn$ ，则将 x 赋值给 maxn。比较完所有数后，maxn 所存储的则是最大的数，将其输出。

代码：

```
1. #include<iostream>  
2. using namespace std;  
3. int main(){  
4.     int n,x,maxn=0;  
5.     cin>>n;  
6.     for(int i=1;i<=n;i++){  
7.         cin>>x;  
8.         if(x>maxn){  
9.             maxn=x;  
10.        }  
11.    }  
12.    cout<<maxn;  
13. }
```

练习：

1、编写一个评分程序，接受用户输入选手的 10 个得分(0-10 分)，然后去掉一个最高分和一个最低分，求出某选手的最后得分(平均分)。

参考代码：

```
1. #include<iostream>  
2. using namespace std;  
3. int main(){  
4.     double maxn=0,minn=10,sum=0,x;  
5.     for(int i=1;i<=10;i++){  
6.         cin>>x;  
7.         if(x>maxn)maxn=x; //求最大值  
8.         if(x<minn)minn=x; //求最小值  
9.         sum+=x; //求总和  
10.    }  
11.    cout<<(sum-maxn-minn)/8;
```

```
12. }
```

=====网站练习=====

换座位

【题目描述】

期中考试结束了，班主任想给同学们重新安排座位，所以她让同学们按学号 1 至 n 的顺序依次在教室外排好队，然后在队伍中挑选一些同学来改变他们的位置。班主任想知道最终有多少个同学的位置发生了变化。楠楠是个信息学高手，班主任把这个任务交给楠楠来完成。

【输入】

第一行，输入一个整数 n($n \leq 100$)，表示有 n 个同学。第二行，有 n 个整数，依次表示换位后的学号。

【输出】

输出有多少个同学的位置发生了变化。

【样例输入】

```
10  
1 2 5 4 6 3 7 8 9 10
```

【样例输出】

```
3 (样例说明：只有学号为 5, 6, 3 共 3 位同学的位置发生了变化。)
```

【分析】

询问有多少个第 i 个数的值不等于 i，枚举每个数统计输出。

【代码】

```
1. #include<iostream>  
2. using namespace std;  
3. int main(){  
4.     int n,ans=0,x;  
5.     cin>>n;  
6.     for(int i=1;i<=n;i++){  
7.         cin>>x;  
8.         if(x!=i){  
9.             ans++;  
10.        }  
11.    }  
12.    cout<<ans;  
13. }
```

变形虫

【题目描述】

Bessie 是一只变形虫，一开始它的体重是 A。在地板上从左往右依次放着 N 块蛋糕，第 i 块蛋糕的重量是 Wi。变形虫从左边爬到右边，每次遇到一块蛋糕，如果蛋糕的重量恰好等于变形虫当前的重量，那么变形虫就吃掉这块蛋糕，吃完蛋糕后变形虫的重量增加了一倍；如果蛋糕的重量不等于变形虫当前的重量，那么变形虫永远也吃不了这块蛋糕了。变形虫只能从左往右爬，不能吃了某蛋糕后再往左爬。你的任务是计算变形虫的最终体重是多少。

【数据规模】 $1 \leq A \leq 1000000000$, $1 \leq N \leq 200$, $1 \leq W_i \leq 1000000000$ 。

【输入】

第一行，两个整数：A, N。第二行，N 个整数，空格分开，第 i 个整数就是第 i 块蛋糕的重量 Wi。

【输出】

一个整数，变形虫的最终体重。

【输入样例 1】

15

2 1 3 1 2

【输入样例 2】

10 7

1 4 9 16 25 36 49

【输出样例 1】

4

【输出样例 2】

10

提示

【样例解释 1】

变形虫首先会吃掉第 2 块蛋糕，体重变成 2。然后变形虫再吃掉第 5 块蛋糕，体重变成 4。

【样例解释 2】

变形虫吃不了任何蛋糕，体重不变。

【分析】

略

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,n,x;
5.     cin>>a>>n;
6.     for(int i=1;i<=n;i++){
7.         cin>>x;
8.         if(x==a){
9.             a*=2;
10.        }
11.    }
12.    cout<<a;
13. }
```

统计数字

【题目描述】

小洪最近学了正整数和负整数，她现在面对一组整数，想算出这组整数有多少个正整数，多少个负整数，以及正整数的和是多少，负整数的和是多少。

输入格式： 第一行：一个正整数 n ，代表有 n 个整数 ($0 < n \leq 100$)。 第二行： n 个空格分开的整数(小于等于 1000)。

【输入】

第一行：一个正整数 n ，代表有 n 个整数 ($0 < n \leq 100$)。

第二行： n 个空格分开的整数(小于等于 1000)。

【输出】

第一行：两个整数，代表正整数的个数和正整数的和（如果没有正数就输出 0 0）。

第二行：两个整数，代表负整数的个数和负整数的和（如果没有负数就输出 0 0）。

【样例输入】

输入样例一：

6

12 40 15 -20 48 -50

输入样例二：

2

12 20

【样例输出】

输出样例一：

4 115

2 -70

输出样例二：

2 32

0 0

【分析】

枚举统计

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,ans1=0,ans2=0,sum1=0,sum2=0,x;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         cin>>x;
8.         if(x>0){
9.             ans1++;
10.            sum1+=x;
11.        }
12.        if(x<0){
13.            ans2++;
```

```
14.         sum2+=x;
15.     }
16. }
17. cout<<ans1<<" "<<sum1<<endl;
18. cout<<ans2<<" "<<sum2<<endl;
19. }
```

分木块

【题目描述】

小洪今天来到伐木场参观，刚好伐木场的工人遇到一个难题，他们希望小洪帮忙解决。难题是已经知道有 n 棵木头的长度，先将这 n 棵木头截成 100 厘米长度的木块，再将剩余木块截成 10 厘米长度的木块，最后将剩余木块截成 1 厘米长度的小木块，现在是想让你统计出这些木块最终能截成多少条 100 厘米、10 厘米、1 厘米的木块。

【输入】

第一行是一个整数 n ，表示有 n 棵木头需要加工 $n \leq 50$ 。第二行是 n 个整数，表示以厘米为单位的所有木头的长度（所有木头小于 100000）。

【输出】

第一行：一个整数，代表截成 100 厘米长度木块的条数。第二行：一个整数，代表截成 10 厘米长度木块的条数。第三行：一个整数，代表截成 1 厘米长度木块的条数。

【样例输入】

```
2
154 160
```

【样例输出】

```
2
11
4
```

【分析】

见代码

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,x,a=0,b=0,c=0;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         cin>>x;
8.         a+=x/100; //满足整百的数
9.         b+=x%100/10; //累加十位数
10.        c+=x%10; //累加个位数
11.    }
12.    cout<<a<<endl;
13.    cout<<b<<endl;
```

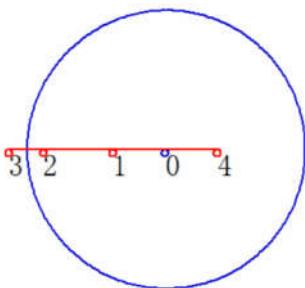
```

14.     cout<<c<<endl;
15. }
```

网络信号

【题目描述】

楠楠来到科技馆参加一个网络信息探测试验活动。试验时中间有一个 WiFi 网络路由器，以路由器为中心半径为 R 的圆内（包括圆周上）的手机都可以收到 网络信号，而圆外的手机就接收不到信号。楠楠拿着手机，一开始在中间，然后 每次向左或向右走动一定距离，停下测试一下网络信号。问有多少次测试是可以 接收到网络信号的。例如： $R=8$ 米，楠楠一共测试了 4 次。第一次向左走 3 米；第二次向左再走 4 米；第三次向左再走 2 米；第四次向右再走 12 米，结果只有第 3 次测试楠楠是接收不到信号的，其它 3 次测试都可以接收到网络信号。



【输入】

第一行，2 个正整数 R 和 N 。分别表示网络路由器的工作半径和楠楠的测试 次数。
第二行， N 个整数。第 i 个整数 D_i 表示楠楠第 i 次向左或向右移动的距离，如果 D_i 是负整数，则表示楠楠第 i 次测试是向左移动了 $|D_i|$ 米的距离。如果 D_i 是非负整数，则表示楠楠第 i 次测试是向右移动了 D_i 米距离。

【输出】

一个整数。表示楠楠有多少次测试是可以接收到网络信号的。

【样例输入】

```
4 5
2 2 2 -8 -8
```

【样例输出】

```
3
```

【提示】

（解释：第 1、2、4 次都半径为 4 的圆内，所以可以接收信号）

数据范围：10 个数据： R,N 的范围是 $[1\cdots 100]$ 。每次移动距离的范围是 $[-100\cdots 100]$ 。

【分析】

略

【代码】

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int r,n,x,p=0,ans=0; //p 表示当前所在的位置
```

```
5.     cin>>r>>n;
6.     for(int i=1;i<=n;i++){
7.         cin>>x;
8.         p+=x;
9.         if(p>=r&&p<=r){
10.             ans++;
11.         }
12.     }
13.     cout<<ans;
14. }
```

摘李子

【题目描述】

六一儿童节就要到了，晨晨学校组织 n 位学生去农场摘李子。为了体现同学友好，大家把摘到的李子集中起来，然后平均分配给学生，剩余的李子就送给老师；另外，为了让老师也更多地分享同学们的快乐，同学们还约定：如果按前面办法分配后老师得到的李子数比每个同学的少，则每位同学再拿一个出来送给老师。

现在晨晨想知道每位同学最后能收获多少个李子？送给了老师多少个李子？

【输入】

第一行：一个整数 n ($1 \leq n \leq 200$)。

第二行： n 个 200 以内的正整数，它们之间用一个空格隔开，代表每人摘到的李子数。

【输出】

第一行：一个整数，代表每位学生最后能收获的李子数。

第二行：一个整数，代表老师最后能收到的李子数。

【样例输入】

输入样例 1：

```
4  
3 5 2 1
```

输入样例 2：

```
10  
95 90 88 92 94 98 96 93 92 94
```

【样例输出】

输出样例 1：

```
2  
3
```

输出样例 2：

```
92  
12
```

【分析】

见代码

【代码】

```
1. #include<iostream>
```

```
2. using namespace std;
3. int main(){
4.     int n,x,sum=0,k,r;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         cin>>x;
8.         sum+=x;    //统计所有李子个数
9.     }
10.    k=sum/n;   //均摊每位同学的李子数
11.    r=sum%n;   //老师获得的李子数
12.    if(r<k){
13.        r+=n;
14.        k--;
15.    }
16.    cout<<k<<endl;
17.    cout<<r<<endl;
18. }
```

while 语句 1

在学习 `for` 语句时，我们可以从循环变量的初值和终值比较直接地知道循环的次数，而我们还会遇到另一类问题，我们知道循环的结束条件，求循环的次数，这类问题没法直接用 `for` 求解，这里就需要用到 `while` 循环语句了。先来看 `while` 语句的格式：

```
while (布尔表达式) {
    语句块;
}
```

`while` 语句用于“当满足某一条件时进行循环”的情况，其执行次序为当布尔表达式为 `true` 时，执行循环体中的语句块，然后继续检查布尔表达式的值，如果该值为 `true`，继续执行循环体，直至布尔表达式的值为 `false` 才退出循环，其流程图如右图所示。特别要注意的是，为了能使 `while` 循环能终止，循环体中一定要有影响布尔表达式的操作，否则该循环就是一个死循环。



例题：求 $s=1+2+3+\dots+n$ ，当加到第几项时， s 的值会超过 1000？

分析：我们之前学过利用 `for` 语句求 $1+2+\dots+n$ 的和，但那时， n 是已知的。现在我们要求的是和超过 1000 时的 n ，显然很难直接套用 `for` 语句的思路来求解。尽管我们不知道 n 是多少，但是我们可以不断试，将 1 加起来，看和是否超过 1000，如果没超过，再把 2 加进来，还是没超过，再加 3，直到什么时候结束呢？直到和超过 1000 就结束。在这里，结束条件是明确的。所以我们就可以写语句 `while(s<=1000){}`，在没有超过时进行循环，每循环一次将一个数累加到 s 中。同时，我们还需要注意，如何让每次累加的数递增呢？我们在累加之前，可以显然数 n 递增 1，即 `n++`，然后再将 n 累加到 s 中。直到循环结束时， n 就是所求的值。

代码：

```

1. #include <iostream>
2. using namespace std;
3. int main (){
4.     int n=0, s=0;
5.     while (s<=1000){
6.         n++;
7.         s+=n; //这个语句使得布尔表达式的值会变成 false, 从而避免死循环
8.     }
9.     cout<<n;
10. }
```

练习：

1、编程求出满足下列式子的 n 的最大值： $2^2+4^2+6^2+\dots+n^2<1500$

2、求恰好使 $s=1+1/2+1/3+\dots+1/n$ 的值大于 10 时 n 的值

3、小明过年收红包收了 A 元，他想把这些钱存银行，银行的年利率是 3.7%，小明采取复利的方式进行存款，即当年的利息不取出继续存在银行算利息，小明想知道 A 元需要存多少年才能翻一倍。

【参考代码 1】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int sum=0, n=0;
5.     while(sum<1500){
6.         n+=2;
7.         sum+=n*n;
8.     }
9.     cout<<n-2;
10. }
```

【参考代码 2】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n=0;
5.     double s=0;
6.     while(s<10){
7.         n++;
8.         s+=1.0/n;
9.     }
10.    cout<<n<<endl;
11. }
```

【参考代码 3】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,n=0;
5.     double p;
6.     cin>>a;
7.     p=a;
8.     while(p<2*a){
9.         p=p*1.037;
10.        n++;
11. }
```

```
11.     }
12.     cout<<n;
13. }
```

=====网站练习=====

加法器

【题目描述】

做一个加法运算器，算出 N 组加数的和。（ $0 \leq N \leq 10$ ）

【输入】

$N+1$ 行，每行两个整数，以 0 0 结束。

【输出】

N 行，每行一个整数，即算出的和。

【样例输入】

3 5

8 1

3 8

0 0

【样例输出】

8

9

11

【分析】

本题的难点在于怎样判断输入结束。我们可以每次读入两个整数，如果这两个整数都不为 0，则输出两个整数的和，继续读入两个整数，直到两个整数为 0 则退出循环。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,b;
5.     cin>>a>>b;
6.     while(a!=0&&b!=0){
7.         cout<<a+b<<endl;
8.         cin>>a>>b;
9.     }
10. }
```

寻找 2 的幂

【题目描述】

数学上把乘方的结果叫 2 的幂，如 4、8、32 等。给定一个整数，请输出距离它最近的那个 2 的幂是多少。如果有两个距离相同，输出那个小的。

【输入】

只有一个整数。

数据范围 ≤ 10000000000000000 （提示：用 qword 存储）

【输出】

只有一个整数，表示距离最近的那个 2 的幂。

【样例输入】

17

【样例输出】

16

【分析】

利用 while 循环求出大于整数 a 的 2 的幂 i，则小于 a 的最大的 2 的幂为 $j=i/2$ ，比较 i 和 j 与 a 的差值，输出差值小的那个。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     long long a,i=2,j;
5.     cin>>a;
6.     while(i<a){
7.         i=i*2;
8.     }
9.     j=i/2;
10.    if(i-a<a-j){
11.        cout<<i;
12.    }else{
13.        cout<<j;
14.    }
15. }
```

验证角谷猜想

【题目描述】

数论中有许多猜想尚未解决，其中有一个被称为“角谷猜想”的问题，该问题在五、六十年代的美国多个著名高校中曾风行一时，这个问题是这样描述的：任何一个大于一的自然数，如果是奇数，则乘以三再加一；如果是偶数，则除以二；得出的结果继续按照前面的规则进行运算，最后必定得到一。现在请你编写一个程序验证他的正确性。

【输入】

一个正整数 N(0<=N<=1 000 000 000)

【输出】

输出验证“角谷猜想”过程中的奇数，最后得到的 1 不用输出；每行中只有两个输出之间才能有一个空格；如果没有这样的输出，则输出：No number can be output!。

【样例输入 1】

18

【样例输入 2】

16

【样例输出 1】

9 7 11 17 13 5

【样例输出 2】

No number can be output!

【分析】

模拟整个过程，判断循环结束的条件：n==1 时结束，则 n!=1 时不断循环，每循环一次，对当前数进行奇偶性判断，如果是偶数则 /2，如果是奇数，则输出，且更新为 *3+1。如何判断没有输出的情况呢？这里采用一个计数器 cnt，每输出一次累加 1，在循环结束后统计，如果 cnt 还是为 0，则没有输出过，那么就输出 No number can be output!。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,cnt=0;
5.     cin>>n;
6.     while(n!=1){
7.         if(n%2==0){
8.             n/=2;
9.         }else{
10.             cout<<n<<" ";
11.             n=n*3+1;
12.             cnt++; //计数器
13.         }
14.     }
15.     if(cnt==0){
16.         cout<<"No number can be output!";
17.     }
18. }
```

圣经数

【题目描述】

人们把 153 叫做“圣经数”，因为 153 具有一个有趣的性质：任写一个 3 的倍数，把各位数字的立方（一个数的立方=三个该数相乘的结果，如 $5^3=5*5*5=125$ ）相加，得出和，再把和的各位数字立方相加，如此反复进行，最后则必然出现 153。

例如：24是3的倍数，按照上述规则，进行变换的过程是：

24→2^3+4^3→72→7^3+2^3→351→3^3+5^3+1^3→153

经过3次变换，153出现了！请编一程序，输入某一3的倍数，输出变换成153所需要的次数。

【输入】

只有一行，有一个整数n（为3的倍数）。

【输出】

只有一行，即为变换成153所需要的次数。

【样例输入】

12

【样例输出】

5

【数据规模】

对于100%数据， $0 < n < 10000$

【分析】

同样需要分析循环的结束条件： $n==153$ ，那么 $n!=153$ 的时候，对n进行数位分离，因为题目限定n为四位数，所以只需分离至千位即可，数位分离后再合成新数。如果新数还不为153，则继续数位分离和合成新数，直至新数的值等于153。每合成一次新数，则用计数器累加1，最后输出计数器的值。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,cnt=0;
5.     cin>>n;
6.     while(n!=153){
7.         int ge=n%10;
8.         int shi=n/10%10;
9.         int bai=n/100%10;
10.        int qian=n/1000%10;
11.        n=ge*ge*ge+shi*shi*shi+bai*bai*bai+qian*qian*qian;
12.        cnt++;
13.    }
14.    cout<<cnt;
15. }
```

小青蛙爬井

【题目描述】

关于小青蛙爬井的故事，你应该早就听过了：井深10尺，小青蛙从井底向上爬，每个白天向上爬3尺，每个晚上又滑下来2尺，然后问你第几天它能爬上来。答案是第8天。

现在，那只著名的小青蛙又回来了，它现在每个白天已经可以向上爬（）尺了，当然，晚上还是要下滑（）尺的。如果告诉你井深（）尺，请计算一下，现在，它第几天可以爬

上来。

【输入】

有三个整数，分别表示白天爬的高度、晚上掉下来的高度、井的高度。

数据范围 ≤ 10000000

【输出】

只有一个整数，表示第几天可以爬上来。

【样例输入】

3 2 10

【样例输出】

8

【分析】

本题需要分两种情况考虑，白天和夜晚，不能直接理解成每天爬的高度为白天-夜晚的差值。所以我们模拟青蛙所在的位置 p ，白天爬的高度为 a ，夜晚掉落的高度为 b ，井的高度为 h 。循环的结束条件为：如果白天的能爬到井外就结束，不然晚上还要掉下来，下一天继续爬。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,b,h,n=1,p=0;
5.     cin>>a>>b>>h;
6.     p=a; //第一天爬的高度
7.     while(p<h){
8.         p-=b; //晚上掉下来
9.         n++; //过了一天
10.        p+=a; //白天继续爬
11.    }
12.    cout<<n;
13. }
```

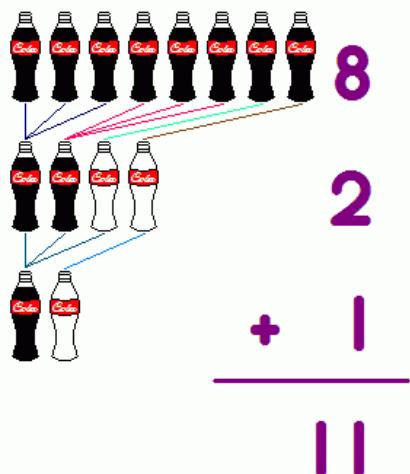
可口可乐

【题目描述】

便利店给出以下的优惠：

“每 3 个空瓶可以换 1 瓶可口可乐。”

现在，您准备从便利店买一些可口可乐 (N 瓶)，您想知道您最多可以从便利店拿到多少瓶可口可乐。下图给出 $N=8$ 的情况。方法是：喝完 8 瓶可乐之后，您有 8 个空瓶；您用 6 只空瓶去换，得到了 2 瓶新的可口可乐；喝完后您有 4 个空瓶子，因此您用 3 个空瓶又换了一瓶新的可乐。最后，您手上有 2 只空瓶，所以您再去换到新的可乐了。因此，您一共获得 $8+2+1 = 11$ 瓶可乐。

**【输入】**

一个整数，这个整数不超过 30000。

【输出】

一个数，表示您可以喝到的最多可乐，你不可以向别人借空瓶子。

【样例输入】

8

【样例输出】

11

【分析】

本题需要分清两个概念：喝的可乐数 k ，和空瓶子数 n 。当空瓶子小于 3 时，就不能继续兑换了。否则，对于 n 个空瓶子，能够兑换的可乐数是 $n/3$ ，这 $n/3$ 瓶可乐又产生 $n/3$ 个空瓶子，所以兑换后的空瓶子数是 $n/3+n\%3$ 。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,k;
5.     cin>>n;
6.     k=n; //一开始喝了 n 瓶可乐,有 n 个空瓶子
7.     while(n>2){
8.         k+=n/3; //换了 n/3 瓶可乐, 所以多喝了 n/3 瓶可乐
9.         n=n/3+n%3; //兑换后的空瓶子数目
10.    }
11.    cout<<k;
12. }
```

while 语句 2

前面我们已经学习了 while 语句的基本用法，接下来，我们学习 while 语句的两个经典应用：数位分离和求最大公约数。

例题 1：

输入一个自然数 M ($m \leq 10^{20}$)，请分离出它各位上的数字，并按个位、十位、百位……顺序输出。

输入样例：

79823

输出样例：

3 2 8 9 7

分析：在之前的练习中，我们能够对特定位数进行数位分离，但在本题中，输入数字的位数是不确定的。我们可以考虑逐位分离的方法。对于数 m，分离个位的操作是 $m \% 10$ ，把个位舍弃的操作是 $m /= 10$ ，重复以上操作，直到 m 为 0，便把各位都分离出来了。

代码：

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int m;
5.     cin>>m;
6.     while(m!=0){
7.         cout<<m%10<<" ";    //直接输出个位
8.         m/=10;    //去除个位
9.     }
10. }
```

例题 2：输入 a 和 b，求 a 和 b 的最大公约数。

分析：求两个数的最大公约数，可以有不同的求法，最高效的求法为辗转相除法。首先理解为什么能辗转相除。我们设 a/b 的商为 k， $a \% b$ 的余数为 r。则 a 可以表示为 $b*k+r$ ，求 a 和 b 的最大公约数，即求 $b*k+r$ 和 b 的最大公约数。设 x 是 b 的约数，如果 x 也是 r 的约数，则 x 是 a 和 b 的公约数，因为 x 肯定能整除 $b*k+r$ 。如果 x 是 b 和 r 的最大公约数，则 x 是 a 和 b 的最大公约数。所以，求 a 和 b 的最大公约数，可以转化为求 b 和 r 的最大公约数，以此迭代，直到余数 r 为 0。那么怎么迭代求两个数的最大公约数呢？以下以 $a=12$, $b=15$ 为例子演示：

	a	b	r
第一次	12	15	12
第二次	15	12	3

第三次	12	3	0

在上例中，我们利用 a 、 b 、 r 三个变量，求 a 和 b 的最大公约数，转换成求 b 和 r 的最大公约数，为了能够迭代计算，我们将 b 和 r 的值赋值回给 a 和 b ，再求 a 和 b 的最大公约数，如果 r 不为 0，则再次迭代。直到 r 为 0，此时 b 就是最大公约数。

代码：

```

1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,b,r;
5.     cin>>a>>b;
6.     r=a%b;
7.     while(r!=0){
8.         a=b;
9.         b=r;
10.        r=a%b;
11.    }
12.    cout<<b;
13. }
```

=====网站练习=====

数字反转

【题目描述】

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】

输入共 1 行，一个整数 N 。 $(-1,000,000,000 \leq N \leq 1,000,000,000)$ 。

【输出】

输出共 1 行，一个整数，表示反转后的新数。

【样例输入 1】

123

【样例输入 2】

-380

【样例输出 1】

321

【样例输出 2】

-83

【分析】

单纯的数字反转，如例题所示，但本题需要多考虑两个问题，一个是负数的问题，另一个是前导零不输出的问题。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,m=0;
5.     cin>>n;
6.     if(n<0){    //判断是否为负数
7.         cout<<"-";
8.         n=-n;
9.     }
10.    while(n!=0){
11.        m=m*10+n%10;    //合成新数，解决前导零输出问题
12.        n/=10;
13.    }
14.    cout<<m;
15. }
```

整数处理

【题目描述】

电脑老师让小明编程做一道题：输入一个正整数 A($A \leq 100000000$)，如果 A 的个位数字是 5，则统计 A 能被 5 整除多少次？否则，统计 A 当中含有多少个“0”？你能做吗？

【输入】

输入一个整数。

【输出】

输出一个整数。

样例输入

【输入样例 1】: 125

【输入样例 2】: 305160

样例输出

【输出样例 1】: 3

【输出样例 2】: 2

【分析】

此题分两种情况处理，如果个位为 5，则统计能被 5 整除多少次，否则统计 0 的个数。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,cnt;
```

```
5.     cin>>n;
6.     if(n%10==5){    //个位为 5 的情况
7.         while(n%5==0){ //如果能够被 5 整除，则继续整除
8.             n/=5;
9.             cnt++;
10.        }
11.    }else{
12.        while(n!=0){ //数位分离，对每位数判断是否等于 0
13.            if(n%10==0){
14.                cnt++;
15.            }
16.            n/=10;
17.        }
18.    }
19.    cout<<cnt;
20. }
```

趣味填空

【题目描述】

小华的寒假作业上，有这样一个趣味填空题：

给出用等号连接的两个整数，如“1234=127”。当然，现在这个等号是不成立的。题目让你在左边的整数中间某个位置插入一个加号，看有没有可能让等号成立。以上面的式子为例，如果写成 123+4=127，这就可以了。

请你编写一个程序来解决它。

【输入】

只有两个正整数，分别表示=左右两边的数，且都不会超过 2000000000。

【输出】

如果存在这样的方案，请输出那个正确的式子，如果加数有前导零，请忽略。如果不存在解决方案，请输出“Impossible!”（引号中的部分）。

【样例输入 1】

1234 127(只需要找到第一种就可以，即加数最小的一种)

【样例输入 2】

12311888100 123118881

【样例输出 1】

123+4=127(只需要找到第一种就可以，即加数最小的一种)

【样例输出 2】

123118881+0=123118881(忽略加数的前导零)

【分析】

本题要把一个整数 n 分离成两半，求其和是否等于 m 。将 n 分离成两半，可以使用过滤器 k ， k 为 10 的幂次方，左边为 n/k ，右边为 $n \% k$ ，分别赋值给 a ， b ，即判断 $a+b$ 是否等于 m 即可。但是难点在于我们需要枚举所有的分离方案，逐一判断。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     long long n,m,k=1,a,b,x;
5.     cin>>n>>m;
6.     x=n;
7.     while(x!=0){    //利用 x 进行数位分离来控制循环的次数
8.         k*=10;      //k 为过滤器
9.         a=n/k;      //分离出左边的数
10.        b=n%k;     //分离出右边的数
11.        if(a+b==m){
12.            cout<<a<<"+"<<b<<"+"<<m<<endl;
13.            return 0;  // 提前结束程序
14.        }
15.        x/=10;
16.    }
17.    cout<<"Impossible!"<<endl;
18. }
```

划分土地

【题目描述】

小明家有一块长 n 单位，宽 m 单位的长方形土地，他想将这块土地都划分成面积相等的小正方形去种番茄（因为小明爱吃番茄）。为了使小正方形的面积最大，小明费劲思考：小正方形的变长是多少才行呢？

【输入】

两个正整数 $n, m (1 \leq n, m \leq 10^9)$

【输出】

一个整数，表示小正方形的边长

【样例输入】

90 60

【样例输出】

30

【分析】

本题为求两个数的最大公约数的模板题。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,m,r;
5.     cin>>n>>m;
```

```
6.     r=n%m;
7.     while(r!=0){
8.         n=m;
9.         m=r;
10.        r=n%m;
11.    }
12.    cout<<m;
13. }
```

最小公倍数

【题目描述】

给出两个正整数 $a, b(1 \leq a, b \leq 10^9)$, 求这两个数的最小公倍数。

【输入】

仅一行, 包含两个正整数 a 和 b , 中间以一个空格隔开

【输出】

仅包含一行, 为 a 和 b 的最小公倍数 $\text{lcm}(a, b)$

【样例输入】

123 321

【样例输出】

13161

【分析】

求两个数的最小公倍数。设 a 和 b 的最大公约数为 gcd , 则 $a=\text{gcd} \cdot x$, $b=\text{gcd} \cdot y$, a 和 b 的最小公倍数 lcm 为 $x \cdot \text{gcd} \cdot y$, 等价于 $a \cdot b / \text{gcd}$ 。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     long long a,b,lcm,r;
5.     cin>>a>>b;
6.     lcm=a*b;
7.     r=a%b;
8.     while(r!=0){
9.         a=b;
10.        b=r;
11.        r=a%b;
12.    }
13.    cout<<lcm/b;
14. }
```

分数的和

【题目描述】

给你 2 个分数，求他们的和 $a/b + c/d$ ，并要求和为最简形式。

【输入】

一行包含四个正整数 a, b, c, d ($0 < a, b < 20, 0 < c, d < 10^9$ (10^9 表示 10 的 9 次方)，以下题目出现类似表达方式，意思相同))，表示两个分数 a/b 和 c/d 。

【输出】

输出两个整数 e 和 f ，表示 $a/b + c/d$ 的最简化结果是 e/f ，每组输出占一行。

【输入样例 1】

1 2 1 3

【输入样例 2】

4 3 2 3

【输出样例 1】

5 6

【输出样例 2】

2 1

【分析】

将 a/b 和 c/d 通分后得 $(a*d+b*c)/(b*d)$ ，令 $e=a*d+b*c$, $f=b*d$ ，则原题转化为求 e 和 f 的最小约分。求出 e 和 f 的最大公约数 x ，则约分后为 e/x , f/x 。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,b,c,d,e,f,r;
5.     cin>>a>>b>>c>>d;
6.     e=a*d+b*c;
7.     f=b*d;
8.     a=e;
9.     b=f;
10.    r=a%b;
11.    while(r!=0){
12.        a=b;
13.        b=r;
14.        r=a%b;
15.    }
16.    cout<<e/b<<" "<<f/b;
17. }
```

多重循环

如果把一个循环放在另一个循环体内，那么就可以形成循环嵌套，循环嵌套既可以是 `for` 循环嵌套 `while` 循环，也可以是 `while` 循环嵌套 `for` 循环，或者 `for` 循环嵌套 `for` 循环，`while` 循环嵌套 `while` 循环，总之，各种类型的循环都可以作为外层循环，各种类型的循环也可以作为内层循环。

两个控制循环的语句

在进行循环的学习中，我们有时会需要提前结束循环或者跳过当前循环的操作，这就需要认识两个控制循环的语句：`break` 和 `continue`。

break 语句：`break` 用于完全结束一个循环，跳出循环体。不管是哪种循环，一旦在循环体中遇到 `break`，系统将完全结束该循环，开始执行循环之后的代码。注意，在多重循环中，`break` 的作用是跳出当前循环，外层循环还是会继续的。以下示例：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     for(int i=1;i<=10;i++){
5.         if(i==5)break;    //注意这个语句
6.         cout<<i<<' ';
7.     }
8. }
```

执行以上语句后，`i` 遇到 5 之后就直接跳出循环，后续语句就不再执行。输出如下：



```
F:\myc++\test7\test.exe
1 2 3 4
Process exited after 0.3114 seconds with return value 0
请按任意键继续... . . .
```

continue 语句：`continue` 的作用是略过当次循环中剩下的语句，重新开始新的循环。以下示例：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     for(int i=1;i<=10;i++){
5.         if(i==5)continue;    //注意这个语句
6.         cout<<i<<' ';
7.     }
}
```

```
8. }
```

执行上述代码后，循环变量 *i* 遇到 5 之后跳过当次循环剩下的语句，不输出 5，但还是会继续进行下次循环。输出如下：



```
F:\myc++\test7\test.exe
1 2 3 4 6 7 8 9 10
-----
Process exited after 0.3004 seconds with return value 0
请按任意键继续...
```

例 1：打印图形

输入 *n*，打印出边长为 *n* 的正方形

输入样例：

```
4
```

输出样例：

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

分析：以上图形，观察得到有 *n* 行，我们可以循环 *n* 次，每次输出一行。对于每一行又如何输出呢？每一行有 *n* 个*号和一个换行，可以循环 *n* 次，每次输出一个*号，最后再输出一个换行即可。

代码：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     for(int i=1;i<=n;i++){ //外循环表示每一行
5.         for(int j=1;j<=n;j++)cout<<'*'; //内循环输出 n 个*号
6.         cout<<endl;
7.     }
8. }
```

练习：

1、编程序打印如下平行四边形，要求输入整数 *n*, *m* (*n*, *m* 为整型范围)，代表图形有 *n* 行，每行有 *m* 个“*”字符。例如输入 4, 6，所形成的图形如下：

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

2、输入 *n*，输出一个 *n* 行的等腰三角形，比如输入 5，输出如下：

```

*
***
***
```

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

【参考代码 1】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,m;
5.     cin>>n>>m;
6.     for(int i=1;i<=n;i++){
7.         for(int j=1;j<=i-1;j++)cout<<" ";
8.         for(int j=1;j<=m;j++)cout<<"*";
9.         cout<<endl;
10.    }
11. }
```

【参考代码 2】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         for(int j=1;j<=n-i;j++)cout<<" ";
8.         for(int j=1;j<=2*i-1;j++)cout<<"*";
9.         cout<<endl;
10.    }
11. }
```

例 2：求 $s=1^1+2^2+3^3+\dots+N^N$

分析：根据题意，第 i 项为 i 的 i 次方，那么，只需要循环 n 次，求出每一项的值累加起来即可。对于循环中的第 i 项，如何求 i 的 i 次方呢，循环 i 次，每次累乘 i ，即为 i 的 i 次方。

代码：

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n;
5.     long long sum=0;
6.     cin>>n;
7.     for(int i=1;i<=n;i++){
8.         long long t=1;
9.         for(int j=1;j<=i;j++)t*=i; //求出第 i 项的值
10.    }
11. }
```

```
10.         sum+=t; //累加第 i 项的值
11.     }
12.     cout<<sum;
13. }
```

练习：输入 $n(n \leq 10)$, 求 $1!+2!+\dots+n!$ 的值

【代码参考】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,sum=0;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         int t=1;
8.         for(int j=1;j<=i;j++)t*=j;
9.         sum+=t;
10.    }
11.    cout<<sum;
12. }
```

=====网站练习=====

直角三角形

【题目描述】

今天数学老师讲了直角三角形的性质，例如直角三角形中两个锐角和是 90 度。老师还介绍了著名的勾股定理：直角三角形两直角边边长平方和等于斜边边长的平方。这一定理在中国的商代就被数学家商高发现了，比西方早了近 500 年。

乐乐想到了用计算机生成一些直角三角形，具体的方法是用 0 到 9 这 10 个数字有规律的组成一个直角三角形的图案。样例中的直角三角形是由乐乐想到的某种确定算法生成的，你能观察出来是那种算法吗？请你写一个程序，对于任意从键盘输入的 1 到 100 的自然数，输出相应的直角三角形。

你的程序必须完整的计算结果，注意在这一算法中数字 ‘0’ 紧跟着数字 ‘9’。

【输入】

输入格式：包含一个整数的单独一行。

【输出】

输出格式：一个像下例一样的数字三角形，数字三角形中没有空格字符。

【样例输入】

7

【样例输出】

1

232
34543
4567654
567898765
67890109876
7890123210987

【分析】

通过观察可以得到，对于第 i 行，一共输出 $2*i-1$ 个数，依次从 i 递增到 $i+i-1$ ，然后再递减到 i 。

【代码 1】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         for(int j=i;j<=i+i-1;j++)cout<<j%10; //从 i 递增到 i+i-1
8.         for(int j=i+i-2;j>=i;j--)cout<<j%10; //从 i+i-2 开始递减到 i
9.         cout<<endl;
10.    }
11. }
```

【代码 2】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n;
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         for(int j=1;j<=i;j++){ //先输出前 i 个数
8.             cout<<(i+j-1)%10;
9.         }
10.        for(int j=i-1;j>=1;j--){ //再输出后 i-1 个数
11.            cout<<(i+j-1)%10;
12.        }
13.        cout<<endl;
14.    }
15. }
```

敲七

【题目描述】

读入一个整数 n(整型范围)，输出 1~n 中 7 和 7 的倍数，还有包含 7 的数字。

例如 (17, 27, 37...70, 71, 72, 73...)

【输入】

一个整数 n

【输出】

输出符合要求的整数

【样例输入】

15

【样例输出】

7

14

【分析】

枚举 1 到 n 之间的所有整数，对于每个数 i，首先判断其是否为 7 的倍数，如果是则直接输出，否则再进行数位分离逐位判断。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int n;
4. int main(){
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         if(i%7==0){    //是 7 的倍数直接输出
8.             cout<<i<<endl;
9.         }else{        //不是 7 的倍数，则进行数位分离，逐位判断
10.            int t=i;
11.            while(t!=0){
12.                if(t%10==7){
13.                    cout<<i<<endl;
14.                    break; //一旦找到，即可退出当前循环
15.                }
16.                t/=10;
17.            }
18.        }
19.    }
20. }
```

K 好数

【题目描述】

当且仅当一个数的每一位都不超过 k 时，称这个数是“k 好数”。

给定 n 和 k，输出 1 到 n 中有多少个数是“k 好数”。

【输入】

输入只有一行，包含 2 个用空格隔开的整数 n 和 k。

【输出】

输出只有一行，包含 1 个整数，表示 1 到 n 中“k 好数”的个数。

【样例输入】

25 3

【样例输出】

11

【输入输出样例说明】

1 到 25 中 11 个“k 好数”分别为 1, 2, 3, 10, 11, 12, 13, 20, 21, 22, 23。

【数据范围】

对于 50% 的数据， $1 \leq n \leq 1000$ 。

对于 100% 的数据， $1 \leq n \leq 1000000$, $1 \leq k \leq 9$ 。

【分析】

枚举 1 到 n 的所有整数，对于枚举的整数 i，进行数位分离逐位判断，如果每一位都是小于等于 k，则累加统计。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int n,k,cnt=0;
5.     cin>>n>>k;
6.     for(int i=1;i<=n;i++){
7.         int x=i;
8.         int flag=true; //flag 标记是否为 k 好数
9.         while(x!=0){
10.             if(x%10>k){ //有位数大于 k，则不是 k 好数，标记且退出循环
11.                 flag=false;
12.                 break;
13.             }
14.             x/=10;
15.         }
16.         if(flag==true)cnt++;
17.     }
18.     cout<<cnt;
19. }
```

统计数字

【题目描述】

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。 $(1 \leq L \leq R \leq 10000)$

【输出】

输出共 1 行，表示数字 2 出现的次数。

【样例输入 1】

2 22

【样例输入 2】

2 100

【样例输出 1】

6

【样例输出 2】

20

【分析】

枚举 L 到 R 之间的所有整数，对于每一个整数 i，进行数位分离，逐位判断是否存在 2，如果存在则累加统计。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int L,R,cnt=0;
5.     cin>>L>>R;
6.     for(int i=L;i<=R;i++){
7.         int x=i;
8.         while(x!=0){
9.             if(x%10==2)cnt++;
10.            x/=10;
11.        }
12.    }
13.    cout<<cnt;
14. }
```

自创乘法

【题目描述】

做厌了乘法计算题的贝茜，自创了一种新的乘法运算法则。在这套法则里， $A \times B$ 等于一个取自 A、一个取自 B 的所有数字对的乘积的和。比方说， 123×45 等于 $1 \times 4 + 1 \times 5 + 2 \times 4 + 2 \times 5 + 3 \times 4 + 3 \times 5 = 54$ 。对于 2 个给定的数 A、B ($1 \leq A, B \leq 1,000,000,000$)，你的任务是，用新的乘法法则计算 $A \times B$ 的值。

【输入】

输入只有一行，是 2 个用空格隔开的整数：A、B。

【输出】

输出共一行，输出 1 个整数，即新的乘法法则下 $A \times B$ 的值。

【样例输入】

123 45

【样例输出】

54

【分析】

题目的乘法规则为 A、B 两数的任意位相乘的和累加。先分离 A 的每一位，对于 A 的当前位 x，将 x 与 B 的每一位 y 相乘再累加。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int a,b,ans=0;
5.     cin>>a>>b;
6.     while(a!=0){
7.         int x=a%10;    //a 的当前位
8.         a/=10;
9.         int t=b;
10.        while(t!=0){
11.            int y=t%10;   //b 的当前位
12.            t/=10;
13.            ans+=x*y;   //相乘再累加
14.        }
15.    }
16.    cout<<ans;
17. }
```

连续自然数和

【题目描述】

对一个给定的自然数 M，求出所有的连续的自然数段，这些连续的自然数段中的全部数之和为 M。

例子： $1998+1999+2000+2001+2002 = 10000$ ，所以从 1998 到 2002 的一个自然数段为 $M=10000$ 的一个解。

【输入】

包含一个整数的单独一行给出 M 的值 ($10 \leq M \leq 2,000,000$)

【输出】

每行两个自然数，给出一个满足条件的连续自然数段中的第一个数和最后一个数，两数之间用一个空格隔开，所有输出行的第一个按从小到大的升序排列，对于给定的输入数据，保证至少有一个解。

【样例输入】

10000

【样例输出】

18 142

297 328

388 412

1998 2002

【分析】

枚举这个连续序列的起点 i , i 可以从 1 到 $M/2+1$, 思考为什么是这个边界? 对于当前起点 i , 累加 i 到 j 的和, 如果 i 到 j 的和为 M , 则找到一个解, 如果 i 到 j 的和大于 M , 则从 i 为起点的连续序列不存在可行解, 从下一个 i 开始继续枚举。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int M;
5.     cin>>M;
6.     for(int i=1;i<=M/2+1;i++){    //i 为起点
7.         int sum=i;
8.         for(int j=i+1;j<=M/2+1;j++){
9.             sum+=j;      //sum 表示从 i 累加到 j 的和
10.            if(sum==M){
11.                cout<<i<<" "<<j<<endl;
12.                break;
13.            }
14.            if(sum>M){
15.                break;
16.            }
17.        }
18.    }
19. }
```

一维数组 1

在编程过程中，我们会遇到需要存储大量数据的时候，对于每个数据，我们不可能单独定义一个变量，这个时候，我们就需要用到定义数组的方式来存储大量数据。

数组的定义方式：

类型标识符 数组名[常量表达式]

例如：

```
int a[10]; //这个语句定义了 10 个 int 类型的变量  
double b[100]; //这个语句定义了 100 个 double 类型的变量。
```

注意：

- 1、数组名的命名规则与变量名的命名规则一致。即必须满足由字母或下划线开头，后面接字母、数字、下划线的字符组合。
- 2、常量表达式表示数组元素的个数。如 a[10]，这里的 10 表示有 10 个数组元素。又比如 int a[n];这样的定义方式会出错，因为不知 n 的值是多少。

一维数组元素的引用格式：

数组名[下标]

例如：int a[10];其中，a 是一维数组的数组名，该数组有 10 个元素，依次表示为：

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
------	------	------	------	------	------	------	------	------	------

注意：

- 1、数组的下标是从 0 开始的。这一点经常容易被忽略的，需要认真记牢。在上例中，如果引用了 a[10]，会出现运行时错误。
- 2、下标可以是整型常量或整型表达式，如 a[3], a[i], a[2*3]，如果使用表达式作为下标，就要计算表达式的值以确定下标。
- 3、C++语言只能逐个引用数组元素，而不能一次引用整个数组。例如：int a[100], b[100]; 那么 a=b;这样的写法是非法的，a 和 b 表示两个数组，不能对两个数组直接赋值。
- 4、数组元素可以像同类型的普通变量那样使用，对其进行赋值和运算的操作，和普通变量完全相同。例如： a[3]=34;实现了给 a[3]赋值为 34。

例 1：从键盘输入 10 个数，将这 10 个数逆序输出。

分析：

读入多个数据存储到数组中，我们常常需要结合 for 语句，每循环一次读入一个数，存到 a[i] 中。数组元素的遍历和输出，也需要结合 for 语句进行操作。在此利用 for 语句的递减循环

模式，实现数组元素的逆序输出。

代码：

```
1. #include<iostream>
2. using namespace std;
3. int a[11];
4. int main(){
5.     for(int i=1;i<=10;i++){
6.         cin>>a[i];
7.     }
8.     for(int i=10;i>=1;i--){
9.         cout<<a[i]<<" ";
10.    }
11. }
```

练习：

1、数组元素的查找

【题目描述】给你 m 个整数，查找其中有无值为 n 的数，有则输出该数第一次出现的位置，没有则输出-1。

【输入】第一行一个整数 m : 数的个数 ($0 \leq m \leq 100$) 第二行 m 个整数 (空格隔开) (这些数在 $0-999999$ 范围内) 第三行为要查找的数 n

【输出】 n 的位置或-1

【样例输入】

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

【样例输出】

```
3
```

【代码参考】

```
1. #include<iostream>
2. using namespace std;
3. int a[101],m,n, pos=-1;
4. int main(){
5.     cin>>m;
6.     for(int i=1;i<=m;i++){
7.         cin>>a[i]; //读入数组元素
8.     }
9.     cin>>n;
10.    for(int i=1;i<=m;i++){ //枚举遍历
11.        if(a[i]==n){ //如果相等，即找到
12.            pos=i; //记录找到的位置
13.            break; //退出循环，不需要继续查找
14.        }
15.    }
16.    cout<<"Position: "<<pos;
17. }
```

```
14.         }
15.     }
16.     cout<<pos;
17. }
```

2、数组元素的插入

【题目描述】在一个数组的第 x 个位置插入一个新的数 y

【输入】有四行 第一行有一个整数 n ($5 \leq n \leq 10$) 第二行有 n 个整数 第三行有一个整数 x , 为要插入的位置 第四行有一个整数 y , 为要插入的整数

【输出】更新后的数组

【样例输入】

```
5
7 2 3 4 5
2
9
```

【样例输出】

```
7 9 2 3 4 5
```

【分析】

数组元素的插入，需要先移动数组元素，腾出位置，然后再在该位置插入数值。

【代码参考】

```
1. #include<iostream>
2. using namespace std;
3. int a[15],n,x,y;
4. int main(){
5.     cin>>n;
6.     for(int i=1;i<=n;i++){ //读入数据
7.         cin>>a[i];
8.     }
9.     cin>>x>>y;
10.    for(int i=n;i>=x;i--){ //从 x 位置开始每个元素往后移一位
11.        a[i+1]=a[i];
12.    }
13.    a[x]=y; //在 x 位置插入数值
14.    n++; //数组长度加 1
15.    for(int i=1;i<=n;i++){ //输出数组元素
16.        cout<<a[i]<<" ";
17.    }
18. }
```

3、数组元素的删除

【题目描述】把一个数组的第 x 个位置的元素删除掉

【输入】有三行

第一行有一个整数 n ($n \leq 10$)

第二行有 n 个整数

第三行有一个整数 x , 为要删除的位置

【输出】输出更新后的数组

【样例输入】

```
5  
1 2 3 4 5  
3
```

【样例输出】

```
1 2 4 5
```

【分析】

数组元素的删除, 不是真正意义上的删除, 而是通过右边数据往左边移动实现数据覆盖。

【代码参考】

```
1. #include<iostream>  
2. using namespace std;  
3. int a[11],n,x;  
4. int main(){  
5.     cin>>n;  
6.     for(int i=1;i<=n;i++){  
7.         cin>>a[i];  
8.     }  
9.     cin>>x;  
10.    for(int i=x;i<n;i++){  
11.        a[i]=a[i+1];  
12.    }  
13.    n--;  
14.    for(int i=1;i<=n;i++){  
15.        cout<<a[i]<<" ";  
16.    }  
17. }
```

=====网站练习=====

陶陶摘苹果

【题目描述】

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

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

【输入】

输入两行数据。第一行包含 10 个 100 到 200 之间 (包括 100 和 200) 的整数 (以厘米为单位) 分别表示 10 个苹果到地面的高度, 两个相邻的整数之间用一个空格隔开。第二行只包

括一个 100 到 120 之间（包含 100 和 120）的整数（以厘米为单位），表示陶陶把手伸直的时候能够达到的最大高度。

【输出】

输出包括一行，这一行只包含一个整数，表示陶陶能够摘到的苹果的数目。

【样例输入】

100 200 150 140 129 134 167 198 200 111

110

【样例输出】

5

【分析】

本题属于数组元素的查找，统计有多少个数值在给定的数值以下。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int a[11],n,h,cnt;
4. int main(){
5.     for(int i=1;i<=10;i++){
6.         cin>>a[i];
7.     }
8.     cin>>h;
9.     for(int i=1;i<=10;i++){
10.        if(h+30>=a[i]){
11.            cnt++;
12.        }
13.    }
14.    cout<<cnt;
15. }
```

统计数字

【题目描述】

输出一个整数序列中与指定数字相同的数的个数。

【输入】

输入包含三行：

第一行为 N，表示整数序列的长度($N \leq 100$)；

第二行为 N 个整数，整数之间以一个空格分开；

第三行包含一个整数，为指定的数字 m。

【输出】

输出为 N 个数中与 m 相同的数的个数。

【样例输入】

3

2 3 2

2

【样例输出】

2

【分析】

此题仍然是数组元素的查找，查找数组元素中有多少个数值等于给定数值。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int a[101],n,m,cnt;
4. int main(){
5.     cin>>n;
6.     for(int i=1;i<=n;i++){
7.         cin>>a[i];
8.     }
9.     cin>>m;
10.    for(int i=1;i<=n;i++){
11.        if(a[i]==m)cnt++;
12.    }
13.    cout<<cnt;
14. }
```

十进制转二进制

【题目描述】

二进制是计算机唯一能直接识别的进制形式。在日常的处理中，我们经常需要将十进制转换成二进制。现在请你编写一个程序，将一个十进制整数 n 转换成二进制输出。十进制转二进制的方法可以概括为除 2 取余倒取法，例如将 28 转成二进制：

28/2=14 余数为 0

14/2=7 余数为 0

7/2=3 余数为 1

3/2=1 余数为 1

1/2=0 余数为 1

将余数从下往上输出为 11100，所以 28 的二进制形式为 11100。

【输入】

输入一个正整数 n ($n \leq 10000$)

【输出】

输出 n 的二进制形式

【样例输入】

28

【样例输出】

11100

【分析】

直接模拟。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int a[50],n,len;
4. int main(){
5.     cin>>n;
6.     while(n!=0){
7.         a[++len]=n%2;
8.         n/=2;
9.     }
10.    for(int i=len;i>0;i--){
11.        cout<<a[i];
12.    }
13. }
```

兔子繁殖

【题目描述】

有一种兔子，出生后一个月就可以长大，然后再过一个月一对长大的兔子就可以生育一对小兔子且以后每个月都能生育一对。现在，我们有一对刚出生的这种兔子，那么， n 个月过后，我们会有多少对兔子呢？假设所有的兔子都不会死亡。

【输入】

输入文件仅一行，包含一个自然数 n ($n \leq 46$)。

【输出】

输出文件仅一行，包含一个自然数，即 n 个月后兔子的对数。

【样例输入】

5

【样例输出】

5

【分析】

求斐波那契数列。当前这个月的兔子个数 $f[i]$ ，等于上个月的兔子数 $f[i-1]$ ，加上新出生的兔子数，新出生的兔子数为前两个月的兔子数 $f[i-2]$ ，即 $f[i] = f[i-1] + f[i-2]$ ，递推求解。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int f[47],n;
4. int main(){
5.     cin>>n;
6.     f[1]=1;
7.     f[2]=1;
8.     for(int i=3;i<=n;i++){
9.         f[i]=f[i-1]+f[i-2];
10.    }
11.    cout<<f[n];
```

```
12. }
```

母猪的故事

【题目描述】

话说现在猪肉价格这么贵，著名的网易总裁丁磊也开始了养猪生活。说来也奇怪，丁磊养的猪就是不一样，一出生第二天开始就能每天中午生一只小猪，而且生下来的竟然都是母猪。不过光生小猪也不行，丁磊采用了一个很奇特的办法来管理他的养猪场：

对于每头刚出生的小猪，在他生下第二头小猪后立马被杀掉，卖到超市里。

假设在创业的第一天，丁磊只买了一头刚出生的小猪，请问，在第 N 天晚上，丁磊的养猪场里还存有多少头猪？

【输入】

有一个正整数 N 代表丁磊创业的第 N 天。 $(0 < N < 20)$

【输出】

请在一行里输出第 N 天晚上养猪场里猪的数目。

【输入样例 1】

2

【输入样例 2】

3

【输出样例 1】

2

【输出样例 2】

3

【分析】

本题还是求斐波那契数列。设第 i 天的猪数目为 $f[i]$ ，这个数目应该由昨天的猪全部生一个
小猪为 $2*f[i-1]$ ，又因为前天出生的猪要送去宰割场了，所以应该减去前天出生的猪。前天
出生的猪的数量等于大前天猪的数量 $f[i-3]$ ，所以 $f[i]=2*f[i-1]-f[i-3]$ 。前天出生的猪的数量又
等于昨天减去前天猪的数量的差额，即 $f[i-1]-f[i-2]$ ，所以 $f[i]=2*f[i-1]-(f[i-1]-f[i-2])=f[i-1]+f[i-2]$ 。

【代码 1】

```
1. #include<iostream>
2. using namespace std;
3. int f[21],n;
4. int main(){
5.     cin>>n;
6.     f[1]=1;
7.     f[2]=2;
8.     f[3]=3;
9.     for(int i=4;i<=n;i++){
10.         f[i]=f[i-1]*2-f[i-3];
11.     }
12.     cout<<f[n];
13. }
```

【代码 2】

```
1. #include<iostream>
2. using namespace std;
3. int f[21],n;
4. int main(){
5.     cin>>n;
6.     f[1]=1;
7.     f[2]=2;
8.     for(int i=3;i<=n;i++){
9.         f[i]=f[i-1]+f[i-2];
10.    }
11.    cout<<f[n];
12. }
```

冬眠

【题目描述】

麻雀帕西和青蛙弗洛格是好玩伴，它们经常一起比赛唱歌。但冬天来了，青蛙弗洛格 冬眠了，它的睡眠深度是 D 。麻雀帕西觉得好无聊，于是它想办法要唤醒弗洛格。麻雀帕 西只会唱 N 首歌，第 i 首歌的音量是 S_i 。每听完一首歌，青蛙弗洛格的睡眠深度就会减少，减少的值等于它听到的歌的音量。当青蛙弗洛格的睡眠深度大于 0 的时候，它会继续冬眠，当睡眠深度小于或者等于 0 时，它就会被唤醒了。麻雀帕西会从第 1 首歌开始唱，唱完第 1 首歌后如果弗洛格还没醒就接着唱第 2 首歌，如果唱完第 2 首歌弗洛格还没醒就接着唱第 3 首歌，依次类推，如果唱完第 N 首歌后弗洛格还没醒，那么麻雀帕西又重新从第 1 首歌开始唱，就像循环播放音乐一样，一直到青蛙弗洛格被唤醒为止，那么麻雀帕西总共唱了多少首歌？

【数据规模】 对 80% 的数据， $1 \leq D \leq 10000$, $1 \leq N \leq 50$, $1 \leq S_i \leq 100$ 。另外 20% 的数据， $1 \leq D \leq 2000000000$, $1 \leq N \leq 50$, $1 \leq S_i \leq 3$ 。

【输入】

第一行，两个整数： D 和 N 。 第二行， N 个整数，空格分开，第 i 个整数就是第 i 首歌的音量 S_i 。

【输出】

一个整数，麻雀帕西总共唱了多少首歌后，弗洛格会被唤醒？

【输入样例 1】

13 3

5 2 4

【输入样例 2】

3 3

5 2 4

【输入样例 3】

21 3

2 1 3

【输出样例 1】

4

【输出样例 2】

1

【输出样例 3】

11

【样例解释 1】

麻雀帕西唱完第 1 首歌后，青蛙弗洛格睡眠深度变成 8，麻雀帕西唱完第 2 首歌后，青蛙弗洛格睡眠深度变成 6，麻雀帕西唱完第 3 首歌后，青蛙弗洛格睡眠深度变成 2，麻雀帕西再次唱完第 1 首歌后，青蛙弗洛格睡眠深度变成 -3，青蛙弗洛格会被唤醒。

【样例解释 2】

麻雀帕西唱完第 1 首歌后，青蛙弗洛格睡眠深度变成 -2，青蛙弗洛格会被唤醒。

【分析】

本题如果直接模拟求出，会超时。先求出音量总和 sum，然后可以通过 d/sum 直接计算出需要歌唱多少个整轮，剩余的睡眠深度 $d \% sum$ 可以再模拟求出，详见代码。

【代码】

```
1. #include<iostream>
2. using namespace std;
3. int a[51],sum,d,k,n,ans;
4. int main(){
5.     cin>>d>>n;
6.     for(int i=1;i<=n;i++){
7.         cin>>a[i];
8.         sum+=a[i]; //求出一轮的音量总和
9.     }
10.    ans=(d/sum)*n; //直接计算出需要唱几轮
11.    d=d%sum; //计算还剩余多少睡眠深度需要被唤醒
12.    int i=1;
13.    while(d>0){
14.        d-=a[i++];
15.        ans++;
16.    }
17.    cout<<ans;
18. }
```