晫 爸 编 程
PROGRAMMING

信奥编程讲义

（函数结构体）

陈 琰 宏 编著

微信：13989476922

邮箱：zufe_graphics@163.com

目录

第一讲 函数定义

“函数”这个名词的英文原文是“function”，而“function”的原意是“功能”。顾名思义，函数就是用来实现某个功能的，而且通常只实现一个功能(而不会把多个功能糅合到一个函数里)。也就是说，在程序设计语言里引入函数的概念，就是为了进行功能分解。

例如:要输出100~200之内的素数,可以用一个二重循环实现。但如果有一个函数prime，能够实现判断一个整数是否为素数。其调用形式是：prime(m)。该函数调用返回值如果为1，则m为素数，如果为0，则为合数。因此我们只需要用如下的代码就可以输出100~200之内的素数：

```
for( int m=100; m<=200; m++ )
{
    if( prime(m) )
        printf( "%d\n", m );
}
```

把“输出100~200之内所有素数”的功能需求进行分解，把“判断一个整数是否为素数”的功能用prime函数去实现。这就是函数的功能所在。

通过函数，可以提高代码的清晰度，可以模块化编程，提高开发效率。如图所示为一张户型图，客厅、餐厅、厨房、卫生间等功能明确，如果把一个家比喻成一个程序，那么在客厅会客、在餐厅吃饭、在卧室休息等等就是一个个函数。

1.1 函数定义

```
返回值类型 函数名（参数列表）{
        函数体
        return 0;
}
```

其中，第一行称为函数头部。**函数名**是标识这个函数的合法标识符。**返回值类型**是指一个函数结束后返回给调用者的一个“返回值”的数据类型。有些函数的功能是执行一系列操作，而不返回任何值，这种情况下，返回值类型是关键字 void。**参数列表**是当函数被调用时，调用者向函数传递的各种“参数”，此处的参数称为形式参数，参数列表包括参数的数据类型和参数名，参数是可选的，没有参数就是“无参”函数，但是括号不能省略。

大括号之间的部分称为“**函数体**”，主要包括变量说明语句、表达式语句等。如果有返回值，则函数体内至少有一条语句“return 表达式”。在执行函数体的过程中，一旦遇到 return 语句，执行完就立刻退出函数，不再执行后续的语句。无返回值函数不需要 return 语句。

练习：在 1-5 处填写函数的每个部分。

```
_1________  _2______ ( _3________ ){
          __4_________________
          __5_________________


          }
```

如下为求素数函数：

```
int prime(int x){
    int i;
    for(i=2;i<x;i++)
        if(x%i==0)
            return 1;
    return 0;
}
```

1.2 参数

在函数的定义五个部分中，第三个部分为参数列表，参数是函数与函数之间实现通信的数据“接口”。函数调用的过程就是调用者带着实际参数（如果有）执行函数，将实际参数“传递”给形式参数，执行完函数体后再将计算得到的返回值传递给调用者（如果有）。在未调用函数前，函数中的形式参数并不分配内存空间。只有在被调用执行时，才被分配临时存储空间。函数调用结束后，形式参数的内存空间将被操作系统立刻收回。

大多数函数都是带参数的，但也可以不带参数。

1.2.1 无参函数

无参函数就是形参列表中没有参数，即没有发生参数传递。

例 1 定义两个函数，分别输出一行字符，在 main 函数中调用这两个函数。

```
#include<bits/stdc++.h>
using namespace std;
void printstar( )
{
    printf( "******************************\n" );
}
```

```
void printmessage(  )
{
    printf( " Welcome to C/C++!\n" );
}
int main( )
{
    printstar( );        //调用 printstar 函数
    printmessage( );     //调用 printmessage 函数
    printstar( );        //调用 printstar 函数
    Return 0;
}
```

1[1019]累加求和

用无参函数实现累加 100~1000 之内能被 3 整除的偶数。

普通写法：	**改成用无参函数：**

#include<iostream>
using namespace std;
int main()
{
int sum=0,i=100;
while(i<=1000){
if((i%3==0)&&(i%2==0))
sum=sum+i;
i++;
}
cout<<sum;
return 0;
}
	#include<iostream>
	using namespace std;
	
	_____
	_____
	_____
	_____
	_____
	_____
	_____
	_____
	int main()
	{
	sum();
	return 0;
	}

2 [2195] 求累加和

用无参函数实现输入一个正整数 n，计算 1+2+.....+n
样例输入 10
样例输出 55

普通写法：	**改成用无参函数：**
##include<iostream>	#include<iostream>

using namespace std;
 	
int main(){
int n,sum=0,i;
cin>>n;
for(i=1;i<=n;i++)
sum=sum+i;
cout<<sum;
return 0;
}
	using namespace std;
	_____
	_____
	_____
	_____
	_____
	_____
	_____
	_____
	int main()
	{
	_____
	return 0;
	}

3[2624] 计算函数值

定义无参函数，实现计算函数值 f(x) = $4x^2$ - 9x + 7，在 main 函数中根据输入的 x，调用 f 函数求函数值 y。

```
__1__ {
    double x,y;
    __2__
    y= 4*x*x-9*x+7;
    __3__
}
int main(){
__4__
    return 0;
}
```

1.2.2 有参函数

大多数函数都是带参数的。参数由两种：**形式参数和实际参数**。在定义函数时函数名后面括号中的变量名称为形式参数（formal parameter，简称形参）在主调函数中调用一个函数时，函数名后面括号中的参数(可以是一个表达式)称为实际参数（actual parameter，简称实参）。

调用有参函数时，系统将会根据形参的类型为其分配存储空间，而存储空间中的内容（即形参的值）则来自调用函数所提供的实参。因此，调用时必须要提供与形参相匹配的实参。所谓匹配是指实参与形参的个数相等，对应实参与形参的类型相同或赋值兼容。

1 [2195] 求累加和

用有参函数输入一个正整数 n，计算 1+2+.....+n
样例输入 10
样例输出 55

普通写法：

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

int main(){
    int n,sum=0,i;
    cin>>n;
    for(i=1;i<=n;i++)
        sum=sum+i;
    cout<<sum;
    return 0;
}
```

改成有参函数：

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

_____
_____
_____
_____
_____
_____
}
int main(){
    int n;
    cin>>n;
    cout<<sum(n);
    return 0;
}
```

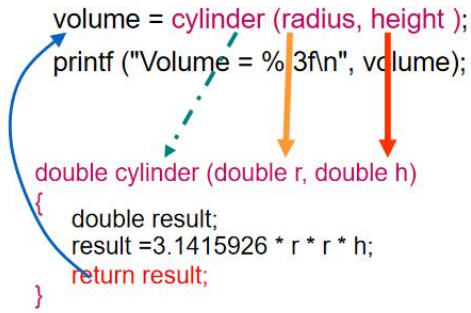
2 [2624] 计算函数值

定义有参函数，实现计算函数值 f(x) = $4x^2$ - 9x + 7，在 main 函数中根据输入的 x，调用 f 函数求函数值 y。

```
#include<iostream>
using namespace std;
  1
{
    double y;
    y = 4*x*x - 9*x + 7;
     2
}
int main( )
{
    double x, y;
    scanf( "%lf", &x );
    y =  3
    printf( "y = %.4f\n", y );
}
```

1.3 函数的返回值

函数的返回值是通过函数中的 return 语句获得。 **return 语句将被调用函数中的一个确定值带回主调函数中去。**一个函数中可以有一个以上的 return 语句，执行到哪个 return 语句，哪个语句起作用。执行到 return 语句，函数就执行完毕了，之后的语句就不执行了。函数值应属于某个确定的类型。如果函数值的类型和 return 语句中表达式的值不一致，则以函数类型为准，即函数类型决定返回值的类型。



4 [3308] 分段函数

通过自定义函数的方式实现如下的分段函数：

$$y=\begin{cases} x^2 & (x<0) \\ \sqrt{x} & (0\le x<9) \\ x-6 & (x\ge 9) \end{cases}$$

```
#include<iostream>
using namespace std;
double f( double x1 )
{
    if( x1<0 )  return x1*x1;
    else if( x1<9 )  return sqrt(x1);
    else  return x1 - 6;
}
int main( )
{
    double x, y;
    printf( "Please input x : " );
    scanf( "%lf", &x );
    y = f( x );
    printf( "y = %.4f\n", y );
    return 0;
}
```

思考：能够改成用一个 return 语句实现？

```
#include<iostream>
using namespace std;
double f( double x1 )
{




}
int main( )
{
    double x, y;
    printf( "Please input x : " );
    scanf( "%lf", &x );
    y = f( x );
    printf( "y = %.4f\n", y );
    return 0;}
```

1.4 函数原型声明

函数声明是指说明函数的类型和参数的情况，以保证程序编译时能判断对该函数的调用是否正确。函数必须先定义后调用，将主调函数放在被调函数的后面，就像变量先定义后使用一样。如果自定义函数在主调函数的后面，就需要在函数调用前，加上函数原型声明。

```
函数类型 函数名(参数表);
double cylinder (double r, double h);
void pyramid (int n);
```

例：阅读程序，写出程序的运行结果，体会函数的“提前声明”。

```
#include<iostream>
using namespace std;
int big(int x,int y);// 函数的提前声明
int main(){
    int x,y,z;
    cin >> x >> y >> z;
    cout << big(big(x,y),z); // 函数调用的返回值又作为其他函数调用的实际参数
    return 0;
}
int big(int x,int y){// 函数定义
    if(x > y) return x;
    else return y;
}
```

1.5 案例讲解

5 [2254] 水仙花数

题目描述

输入两个正整数 m 和 n（1<=m,n<=1000）,输出 m~n 之间的所有满足各位数字的立方和等于它本身的数。要求定义并调用函数 is(number)判断 number 的各位数字之立方和是否等于它本身。

样例输入：100 400

样例输出

153

370

371

分析：写一个函数求解水仙花数。确实函数名、函数参数和返回值类型。

```
#include<bits/stdc++.h>
using namespace std;
  1
        int a,b,c;
        a=i%10;b=i/10%10;c=i/100;
```

```
        if(a*a*a+b*b*b+c*c*c==i)
                2_____________________;
}
int main(){
        int m,n;
        cin>>m>>n;
        for(int i=m;i<=n;i++)
        if(__3_____________________  )
                cout<<i<<endl;
        return 0;
}
```

6 [7736] 数学公式计算快递费

设计一个程序,输入一个值，可以计算专门计算快递费,快递费用计算方式已经用下列数学式子 f(x)来表达,
其中 x 为用重量(千克)(x 可能为小数,且<=1000)，f(x)为快递费
其中 f(x)的表达式为

$$f(x)=\begin{cases}12, & 0\le x\le 4\\ (x-4)*4+12, & 4<x\le 15\\ (x-15)*3+(15-4)*4+12, & 15<x\le 100\\ (x-100)*2+(100-15)*3+(15-4)*4+12, & x>100\end{cases}$$

输入 x 为用重量(千克)(x 可能为小数,且<=1000)
输出快递费
样例输入 500
样例输出 1111

```
#include <iostream>
using namespace std;
double f(double x)
{
        if (x <= 4)
                return 12;
        if (x <= 15)
                return (x-4)*4 + 12;
        if (x <= 100)
                return (x - 15) * 3 + (15 - 4) * 4 + 12;
        return (x - 100) * 2 +(100 - 15) * 3 + (15 - 4) * 4 + 12;
}
int main()
{
        double x;
        cin >> x;
```

```
        cout << f(x);
        return 0;
}
```

7* [1057] 求素数

输入两个正整数 m 和 n(m<n)，求 m 到 n 之间(包括 m 和 n)所有素数的和，要求定义并调用函数 isprime(x)来判断 x 是否为素数(素数是除 1 以外只能被自身整除的自然数)。
输入 m n 输出素数和
样例输入 2 3
样例输出 5

```
#include<bits/stdc++.h>
using namespace std;
int prime(int x)
{
    
}
int main()
{
    int s=0;
    int n,m,i;
    scanf("%d%d",&m,&n);
    for(i=m;i<=n;i++)
    {
        if(__________________) s=s+i;
    }
    printf("%d\n",s);
    return 0;
}
```

课堂作业列表

[2624] 计算函数值　　[3308] 分段函数
[1463] 输出最大的数　　[2621] 素数个数
[2254] 水仙花数　　[1057] 求素数
[3312] 函数计算第几天　[1456] 输出一个四位数

[3313] 2261 拆分实数

课前练习

```
#include <iostream>
using namespace std;
int main()
{
    int n[3],i,j,k;
    for(i=0;i<3;i++)
    n[i]=0;
    k=2;
    for (i=0;i<k;i++)
    for (j=0;j<k;j++)
       n[j]=n[i]+1;
    printf("%d\n",n[1]);
    return 0;
}
```

结果_____

```
#include <iostream>
using namespace std;
int main()
{
    int p[8]={11,12,13,14,15,16,17,18},
i=0, j=0;
  while(i++<7)
     if(p[i]%2)  j+=p[i];
   printf("%d\n",j);
   return 0;
}
```

结果 _____

习题

1. 以下说法中错误的是(　　)
   A、C 程序中可以只包含一个 main 函数
   B、C 程序由一个 main 函数和若干其他函数构成
   C、C 程序中可以没有 main 函数，但至少应包含一个其他函数
   D、C 程序由函数组成，函数是构成程序的基本单位
2. 以下说法中正确的是(　　)
   A、main 函数和其他函数间可相互调用
   B、main 函数可以调用其他函数，但其他函数不能调用 main 函数
   C、因为 main 函数可不带参数，所以其后的参数小括号能省略
   D、根据情况可以不写 main 函数
3. C++语言规定，函数返回值的类型是(　　)
   A、由 return 语句中表达式的类型所决定
   B、由调用该函数的主调函数所具有的类型决定
   C、由定义该函数时所指定的函数类型决定
   D、有系统随即决定
4. 以下关于函数调用的描述中错误的是(　　)
   A、实参可以是常量、表达式或有确定值的变量
   B、实参和形参共用同一内存单元
   C、实参与形参的类型、个数必须一致
   D、只有发生函数调用时，系统才为形参分配存储空间
5. 以下正确的函数定义形式是（）。

A.double fun(int x, int y)　　B.double fun(int x ; int y)
C.double fun(int x, int y);　　D.double fun(int x, y)

6. C 语言规定，简单变量做实参时，它和对应的形参之间的数据传递方式是（　　）。
A.地址传递　　B.值传递
C.由实参传给形参，再由形参传给实参　　D.由用户指定传递方式

7.有以下程序：

```
int fun(int n)
{  int x=1;
   x=x+n;
  return x;
}
main( )
{   printf("%d\n",fun(1)+fun(2));  }
```

程序运行后的输出结果是(　　)
A)4　　B)5　　C)6　　D)7

8. 以下 fun 函数的功能是计算机 x^y，则调用 fun 函数计算 $x=(1.2)^5+(a+4)^2$ 的语句是(假设变量 a 和 x 以正确定义的赋值)。_____

```
double fun(double x, int y)
{    int i;
     double mul=1;
     for( i=1; i<=y; i++)   mul=mul*x
     return mul;
}
```

9.有以下程序，程序运行后的输出结果是_____

```
int fun(int x,int y)
{   a=x+y;
   return 1;
}
int main( )
{   int a=3,b=4;
    c=fun(a,b);
    printf("%d\n",c);
}
```

10.调用函数 f(27)的输出结果是(　　)

```
void f(int n)
{  if(n<5)
     printf("%d", n);
   else{
     printf("%d", n%5);
     f(n/5);
   }
}
```

A. 102　　B. 201　　C. 21　　D. 20

11.执行下面程序，输出是 _____

```
void f(int v, int w) {
  int t;
  t = v; v = w; w = t;
}
int main()
{
  int x = 1, y = 3, z = 2;
  if (x > y)f(x, y);
  else if (y > z)f(y, z);
  else f(x, z);
  printf("%d, %d, %d\n", x, y, z);
  return 0;
}
```

12.以下程序的功能是输出数列 0，0，1，1，2，4，7，13，24，44..的前 20 项(即从第 4 项起每一项的值均为其前 3 项之和)。 请填空。

```
void sub( int a,int b,int c )
{
    int n, d;
    for( n=4;n<=20;n++ )
    {   d=__1_______;
        printf("%d ",d);
        a=b;__2_____ ; c=d;
    }
}
main( )
{   int a=0,b=0,c=1;
    printf("%d %d %d ",a,b,c);
    sub(a,b,c);
}
```

第二讲 参数传递

参数是函数与函数之间**实现通信的数据“接口”**。函数调用的过程就是调用者带着实际参数执行函数，将实际参数“传递”给形式参数，执行完函数体后再将计算得到的返回值传递给调用者。在未调用函数前，函数中的形式参数并不分配内存空间。只有在被调用执行时，才被分配临时存储空间。函数调用结束后，形式参数的内存空间将被操作系统立刻收回。

大多数函数都是带参数的。参数由两种：**形式参数和实际参数**。在**定义函数时函数名后面括号中的变量名称为形式参数**（简称形参）在**主调函数中调用一个函数时，函数名后面括号中的参数称为实际参数**（简称实参）。

（1）形式参数

定义函数时，写入函数圆括号内的参数称为形式参数，又称为形参或虚参。在函数被调用前，形参并没有被分配存储空间，也没有具体的值。

（2）实际参数

调用函数时，写入函数圆括号内的参数称为实际参数，又称为实参。实参可以是常量、变量或表达式，有具体的值。对于实参变量而言，它已经被分配了相应的存储空间。

（3）函数之间参数的传递

调用有参函数时，系统将会根据形参的类型为其分配存储空间，而存储空间中的内容（即形参的值）则来自调用函数所提供的实参。因此，调用时必须要提供与形参相匹配的实参。所谓**匹配是指实参与形参的个数相等，对应实参与形参的类型相同**。此外，当被调用函数执行完毕返回调用函数时，形参的存储空间又被系统收回，形参的值也就不复存在了。

根据不同的应用需求，函数参数的传递方式，或者说函数参数的调用方式分为三种：**传值（调用），传址（调用）和 引用（调用）**。

2.1 值传递

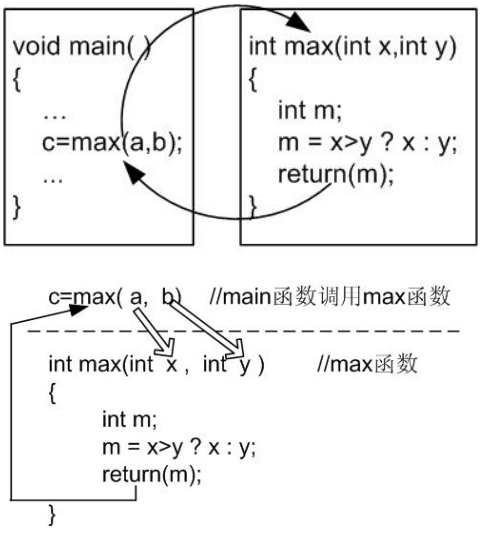
值传递就是主调函数将实际参数“传递”给形式参数。在传递过程中，**实参把值复制给形参传递过去以后**，实参跟形参就没有了联系。因此，值传递是单向传递，第一讲的例子都是属于单向传递。

这种参数传递的过程被形象地称为“虚实结合”的过程，下面结合例子具体分析参数传递的过程。

1 [7535] 两个最大值

编写一个程序，输入 a b 两个值，输出其中最大值。
样例输入 20 30
样例输出 30

```
#include <iostream>
using namespace std;
int max(int x,int y)
{
    int m;
    m=x>y?x:y;
```



```
    return m;
}
int main()
{
    int a,b,c;    .....1
    cin>>a>>b;
    c=max(a,b);
    cout<<c;
}
```

在 main 主函数调用 max 函数时，实参 a,b 把值分别拷贝给 x,y；x,y 得到 a,b 的值后，求解得到最大值 m；然后把 m 的值返回给 c。

把上面的函数改成如下写法，**思考:1 处、2 处、3 处的 a,b 有什么关系？**

```
int max(int a,int b).....2
{
    int m;
    m=a>b?a:b;.....3
    return m;
}
```

2 [1463] 三个最大值

用函数实现，从三个数中找出最大的数。输入 3 个实数输出最大的数，用函数求解。 保留 3 位小数。

```
#include<iostream>
using namespace std;
  1                    //对 max 函数进行声明
int main( )
{
    int a, b, c;
    cin>>a>>b;
    c= 2
    cout<<c;
    return 0;
}
int max(int x,int y)
{//划线可不写满
    _____
    _____
    _____
    _____
    _____
}
```

3 [2027] 求最大值

给你两个数 a,b，函数求 a+b,a-b,a*b 的最大值
样例输入 4 9
样例输出 36

```
#include <iostream>
using namespace std;
int max(int x,int y)
{
    int m;
    m=x>y?x:y;
    return m;
}
int main()
{//划线可不写满
    _____
    _____
    _____
    _____
    _____
    _____
}
```

2.2 地址传递

阅读下面的程序，写出程序运行结果 _____

```
#include <iostream>
using namespace std;
void swap(int a,int b);          //函数原型声明
int main()
{
    int x,y;
    x=10;
    y=20;
    printf("Before swapping:x=%d y=%d\n",x,y);
    swap(x,y);                   //调用函数
    printf("After swapping:x=%d y=%d\n",x,y);
    return 0;
}
void swap(int a,int b)           //定义函数
{
    int t;
```

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

程序运行结果如下：

```
Before swapping:x=10 y=20
After swapping:x=10 y=20
```

该程序通过调用 swap 函数来交换两个变量的值，但这样是不成功的。**为什么不成功？**

答：“值传递”是一种单向的“值传递”。**函数调用时仅仅是将实参变量的值复制了一份交给形参，形参与对应实参的存储空间完全不同，在函数调用过程中对形参的改变，根本不会影响到实参的值。**例子中，将实参 x 和 y 的值分别传给形参 a 和 b。(在值传递之后，a 和 x 就没有任何关联了)

思考：如果把上题的交换函数的 a 换成 x，b 换成 y,能实现交换吗？_____

```
void swap(int x,int y)
{
    int t;
    t=x;x=y;y=t;
}
```

思考：加粗的 x,y 是同一个吗？

```
#include<iostream>
using namespace std;
double f(double x){
    double y;
    y= 4*x*x-9*x+7;
    return y;
}
int main(){
    double x,y;
    cin>>x;
    y=f(x);
    printf("%.4lf",y);
    return 0;
}
```

该程序的运行示例如下：

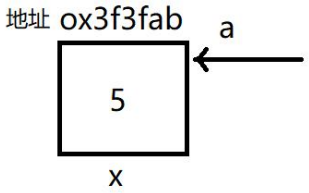
```
2.7↙
y = 11.8600
```

答：**f 函数中的变量 x 和 y，与 main 函数中的变量 x 和 y 是不同的变量**。特别是定义 f 函数时的参数 x 和调用 f 函数时的参数 x，有不同的含义。

如果函数的定义出现在函数调用之后，则必须在函数调用之前对函数进行声明，否则编译时会提示：标识符(函数名)未定义。函数声明是指在函数尚未定义的情况下，事先将该函数的有关信息通知编译系统，以便使编译能正常进行。

为了能实现 x,y 的交换，讲上述代码做如下修改：

```
#include <iostream>
using namespace std;
void swap(int *a,int *b)          //定义函数
{
    int t;
    t=*a;*a=*b;*b=t;
}
int main()
{
    int x=10,y=20;

    printf("Before swapping:x=%d y=%d\n",x,y);
    swap(&x,&y);                  //调用函数
    printf("After swapping:x=%d y=%d\n",x,y);
    return 0;
}
```



此时参数传递的内容是地址，**其传递过程如下：**

1. 初始 x=10,y=10。编译器给变量 x、y 在内存中分别分配一块大小为 4 个字节的存储空间。如图所示，变量 x 在内存中的首地址为 ox3f3fab，地址用&x 表示。

2. 当主函数运行到 swap(&x,&y)时，把&x 传递给 a。a 是一个指针变量，可以用来表示地址，即 a 得到 x 的地址，a=&x。

3. *a，表示地址 a 指向的存储单元的值，即*a=5。当执行 t=*a;*a=*b;*b=t;时，地址 a 存储的值 x 与地址 b 存储的值 y 就发生了交换。

4. 虽然地址传递也是单向传递，但是因为所有的操作是发生在同一块存储空间，等价于进行了双向值传递。

传址的好处：

(1)能在函数内部通过实参地址间接地改变实参的值(但不能改变实参的指向)。

(2)当所传实参内容比较庞大时，传址只是复制了整个实参的地址过去，指针依据同一个地址访问实参变量。而传值就会将实参内容整个拷贝过去，形参会跟实参占一样大的内存，栈空间是有限的。当然了，在弱小的程序中，传址的这个优点不会被体现出来。

因为数组名就是地址，当需要通过函数去修改实参的值时，通常通过数组实现。

4[3313] 10 个元素反序

将数组 array 中的 10 个元素按相反顺序存放。用函数实现。

题目要求用函数实现，定义 reverse 函数，实现将数组 a 中的 10 个元素按相反顺序存放。在主函数中输入数组 array 各元素值，然后调用 reverse 函数实现逆序存放，最后在主函数中输出 array 各元素的值。

样例输入 1 2 3 4 5 6 7 8 9 10

样例输出 10 9 8 7 6 5 4 3 2 1

```
#include <iostream>
using namespace std;
void reverse( int *a, int n ); //函数声明
int main( )
{
    int array[10], i;
    for( i=0; i<10; i++)
        cin>>array[i];// scanf( "%d", &array[i] );
    reverse( array, 10 );
    for( i=0; i<10; i++)
        cout<<array[i]<<" ";//printf( "%d ", array[i] );
    printf( "\n" );
    return 0;
}
//reverse 函数用于将 a 数组中的 n 个元素按逆序存放
void reverse( int *a, int n )
{
    int i, j;       //循环变量
    int temp;   //交换 a[i]和 a[j]时用到的临时变量
    for( i=0; i<n/2; i++ )
    {
        j = n-1-i;
        temp = a[i]; a[i] = a[j];  a[j] = temp;    //交换 a[i]和 a[j]
    }
}
```

2.3 函数设计

设计函数时面临两个问题：

(1) 函数是否有参数，有几个参数，是否有返回值？

(2) 函数要处理的数据是哪些，函数形参的作用是什么，形参的值是在什么时候被“赋予”的。

第(1)问题：程序设计者希望采用怎样的形式去调用函数，这种函数调用形式里有几个参数，分别是什么类型，以此来确定函数的形参个数和类型；程序设计者希望函数执行以后是否得到一个结果，这个结果是什么类型的，是什么含义，是否需要返回到主函数中，以此来确定函数的返回值及其类型、含义等。

第(2)问题：函数形参是在函数调用时，通过实参与形参之间的数据传递，从而“被赋予”了值。

例 输出 100~200 间的全部素数。要求如下：

1) 定义一个函数 prime，用于判断 x 是否为素数，如果为素数，返回 1，否则返回 0。

2) 在主函数中调用 prime 函数，用于判断 100~200 之间的每个数是否为素数。

分析一：

根据题目的意思，如果要通过调用 prime 函数判断 199 是否为素数，函数的形式是 prime(199)。如果为素数则返回 1，否则返回 0。因此，prime 函数的原型为：

```
int prime( int x );
```

分析二：

在 prime 函数里，是要判断形参 x 是否为素数，这个 x 的值不是在 prime 函数里通过输入语句输进去的，也不是采用赋值的方式“赋予”给它的。而是主调函数在调用 prime 函数时，如 prime(199)，把实参 199 的值传递给形参的，因此这时执行 prime 函数时，形参 x 的值就是 199，prime 函数就是要判断 199 是否为素数。

5 [2621] 素数个数

编程求 2～n(n 为大于 2 的正整数)中有多少个素数。

输入输入 n(2≤n≤50000)。

输出素数个数。

分析：

(1) 函数是否有参数，有几个参数？_____

(2) 是否有返回值？_____

```
#include<bits/stdc++.h>
using namespace std;
int prime( int x )  //如果 x 为素数；则返回 1，否则返回 0
{
    int flag=0;
    int k = sqrt(x);
    if(x==2)_1_____________
    for( int i=2; i<=k; i++ )
    {
        if( x%i==0 )  {
            _2_____________;//x 为合数，返回 0
        }
    }
    _3_____________ //x 为素数，返回 1

}
int  main()
{
    int n,m,sum=0;
    cin>>n;
    for( int m=2; m<=n; m++ )
    {
        if( _4_____________ )   //调用 prime 函数，判断 m 是否为素数
                sum++;
    }
```

```
    cout<<sum;
}
```

6 [2716] 回文三位数

如果一个数从左边读和从右边读都是同一个数，就称为回文数。例如 6886 就是一个回文数，求出所有的既是回文数又是素数的三位数。

输出 所有的既是回文数又是素数的三位数。一个数一行。

分析：判断一个数即是回文数，又是素数，写两个函数，一个函数判断是否是回文数，一个函数判断是否是素数

(1) 函数是否有参数，需要几个参数？_____

(2) 函数是否有返回值？_____

```
#include <bits/stdc++.h>
using namespace std;

 1                          //定义判断素数函数
{
        for(int i=2;i*i<=x;i++)
        {
                if(x%i==0) 2
        }
       3
}
 4                          //定义判断回文函数 bool
{
        int a=x/100,b=x%10,c=(x-a*100)/10;
        if(a==b)
                5
        6


}
  int main()
{
        for(int i=100;i<=999;i++)
        {
                if( 7                              )
                {
                        cout<<i<<endl;
                }
        }

        return 0;
  }
```

7* [3312] 函数计算第几天

用函数求解：输入一个日期（年、月、日），计算出该日期是该年的第几天。
在本题中，可以用一个一维数组存储平年中 12 个月份的天数。另外，本题还需要设计两个函数：
1) sum_day 函数用来计算输入日期中的某月某日在平年里是第几天，方法是把该月份前每个月份的天数累加起来，再加上年月日中的日；
2) leap 函数用来判断输入的年份是否为闰年。
样例输入
2018 5 26
样例输出
146

分析
（1）函数是否有参数，需要几个参数？_____
（2）函数是否有返回值？_____

```
#include<iostream>
#include<cmath>
using namespace std;
int leap(int y){//考虑参数如何设置？考虑返回值的类型？
        if((y%4==0)&&(y%100!=0)||(y%400==0))
                return 1;
        else
                return 0;
}
int sum_day(int y,int m,int d){
        int sum=0;//表示总天数
        int a[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
        for(int i=1;i<m;i++)//2018 5 26
                sum=sum+a[i];//把足月的月份求和
                sum=sum+d;//把当前月份的天数加起来
        if(leap(y)&&m>2)sum++;//如果是闰年 且 是 3 月及其以上的月份 需要加 1 天
        return sum;
}
int main(){
        int year,month,day;
        scanf("%d %d %d",&year,&month,&day);
        cout<<sum_day(year,month,day);
        return 0;
}
```

课前练习

```
#include <iostream>
using namespace std;
int main()
{
    int x, y, s;
    cin >> x >> y;
    int t = x % y;
    switch (t)
    {
    case 1:
    case 2:
        s = 1;
        break;
    case 0:
        s = 3;
        break;
    default:
        s = 2;
    }
    cout << s << endl;
    return 0;
}
```

输入 1：

12 3

输出 1：_____

输入 2：

123 8

输出 2：_____

```
#include <iostream>
using namespace std;
int main()
{
    int m, n, i, j, s;
    int d[101];
    cin >> n;
    for (m = 10; m <= n; m++)
    {
        s = m * m;
        j = 0;
        while (s > 0)
        {
            j = j + 1;
            d[j] = s % 10;
            s = s / 10;
        }
        i = 1;
        while (d[i] == d[j] && (i < j))
        {
            i = i + 1;
            j = j - 1;
        }
        if (i >= j)
            cout << m << endl;
    }
    return 0;
}
```

输入：30

输出：_____

习题

1. 以下正确的说法是（　）。

A.实参与其对应的形参共同占用一个存储单元

B.实参与其对应的形参各占用独立的存储单元

C.只有当实参与其对应的形参同名时才占用一个共同的存储单元

D.形参是虚拟的，不占用内存单元 1

2.对于以下递归函数 f，调用 f(4)，其返回值为（　　）。

```
int f(int n)
{    if (n)  return f(n - 1) + n;
```

```
    else return n;
}
```

若函数定义如下，int fun(float a) {float b=a+3; return b; } 假设将常数 3.6 传给 a，则函数的返回值是（　　）。

A.3　　B.6.6　　C.5　　D.6

3.下列程序执行后输出的结果是(　　)

```
int d=1;
void fun (int q)
{ int d=5;
  d+=q++;
  printf("%d ",d);
}
int main(void)
{ int a=3;
  fun(a);
  d+=a++;
  printf("%d\n",d);
  return 0;
}
```

4.有以下函数定义：void fun(int n, double x){……}。若以下选项中的变量都已正确定义并赋值，则对函数 fun()的正确调用语句是（　　）

A.fun(int y, double m);　　B.k=fun(10, 12.5);

C.fun(x, n);　　D.void fun(n, x);

5. 以下（ ）函数的定义是错误的? （　　）

A.void f(int i) { return i+1; }　　B.void f( ) { }

C.void f(int i) { }　　D.int f( ) { return 0; }

6. 下列计算三角形面积函数的声明中，（　　）是正确的。

A.double area(double a, double b, double c);

B.double area(int a, b, c);

C.int area(double a, double b, double c)

D.double area(double a, b, c)

7.执行下面程序，输出是 _____

```
int x=5, y=7;
void swap( ){
    int z;
    z=x;  x=y;  y=z;
}
int main( ){
    int x=3, y=8;
    swap( );
    printf("%d, %d\n", x, y);
    return 0;
}
```

8. 有以下程序：

```
int fun(int  x,int  y)
{   return  x+y; }
main( )
{   printf("%d\n",fun(fun(1,2),fun(3,4)));}
```

程序运行后的输出结果是(　　)

A、3　　B、7　　C、10　　D、编译错误

9.有以下程序：

```
int fun(int n)
{  int f=1;
   f=f*n;
   return f;
}
main( )
{   int i,a[ 5];
   for (i=1; i<=5; i++)  cout<<fun(i-1);
}
```

程序运行后的输出结果是(　　)

A)12345　　B)54321　　C)01234　　D)43210

10.有以下程序：

```
int fun(int n)
{  int x=1;
  x=x+n;
  return x;
}
main( )
{   printf("%d\n",fun(1)+fun(2));  }
```

程序运行后的输出结果是(　　)

A)4　　B)5　　C)6　　D)7

11. 以下程序判断输入的正整数的各位数字之和是否为质数并打印相应的结果”。

```
#include <bits/stdc++.h>
using namespace std;
int isPrime(int num){  //判断 num 是否为质数
    for(int i=2; _____1____; i++)
            if( _____2____) return 0;
    return 1;
}
int sumDigits(int num){  //返回 num 各位数字之和
    int sum;
    for( _____3____ )
        sum +=  _____4____;
    return sum;
}
int main(){
    int num;
```

```
    scanf("%d",&num); //输入一个整数，并假设输入的数大于 1
    if( _____5___== 1 ) printf("各位数字之和是素数！\n");
    else printf("各位数字之和不是素数！\n");
    return 0;
}
```

第三讲 数组与函数

根据不同的应用需求，函数参数的传递方式，或者说函数参数的调用方式分为三种：**传值（调用），传址（调用）和 引用（调用）**。传址（调用）传的是地址，可以通过数组和指针实现。

（1）指针传递与引用传递的区别：

指针参数传递本质上是值传递，它所传递的是一个地址值。值传递过程中，被调函数的形参作为被调函数的局部变量处理，会在栈中开辟内存空间以存放由主调函数传递进来的实参值，从而形成了实参的一个副本（替身）。值传递的特点是，被调函数对形参的任何操作都是作为局部变量进行的，不会影响主调函数的实参变量的值（形参指针变了，实参指针不会变）。

引用参数传递过程中，被调函数的形参也作为局部变量在栈中开辟了内存空间，但此时**存放的是由主调函数放进来的实参变量的地址**。被调函数对形参（本体）的任何操作都被处理成间接寻址，即通过栈中存放的地址访问主调函数中的实参变量（根据别名找到主调函数中的本体）。因此特点，被调函数对形参的任何操作都会影响主调函数中的实参变量。

【注:】指针与引用的另一个重要的不同是指针可以被重新赋值以指向另一个不同的对象。但是引用则总是指向在初始化时被指定的对象，以后不能改变。

3.1 指针作为参数

思考：

```
#include<bits/stdc++.h>
int change(int a, int b){
    int t;
    t=a,a=b,b=t;
}
using namespace std;
int main(){
    int x=3,y=4;
    change(x,y);
    cout<<a<<" "<<b;

}
```

输出的结果是_____

可以将函数的形参定义成指针类型，在调用这类函数时，指针类型的形参所对应的实参应该为同一基类型的变量地址。下面结合例 1 说明指针变量作参数时，函数的调用过程。

例 1　指针变量作参数。

程序代码如下。

```
#include<bits/stdc++.h>
using namespace std;
int change(int *a, int *b){
```

```
    int t;
    t=*a,*a=*b,*b=t;
}
int main(){
    int x=3,y=4;
    change(&x,&y);
    cout<<a<<" "<<b;

}
```

思考与讨论：

1)change 函数中使用两个基类型为整型的指针变量 a 和 b 作为形参，main 函数中调用 change 函数时，将两个整型变量 x 和 y 的地址&x 和&y 作为实参传递给形参 a 和 b。

2）指针变量作形参时，参数传递的形式仍然为值传递。调用函数时，形参 a、b 与 main 函数中变量 x、y 占据着不同的存储空间，形参存储空间中存放的是对应变量的地址。在函数调用过程中，通过对 main 函数中的变量 x 和 y 的间接访问，交换了它们的值。当函数调用完毕后，形参的存储空间仍然被收回，但此时变量 x 和 y 的存储空间中已经保留了改变后的值，从而解决了在被调用函数中改变调用函数中变量值的问题。

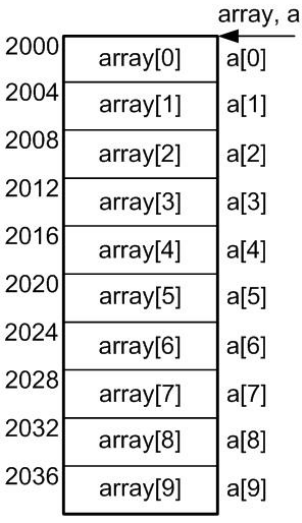
注意：不能试图通过调用以下函数来交换被调用函数中变量的值。

程序代码如下。

```
void change(int *a,int *b)
{   int *t;
    t=a;
    a=b;
    b=t;
}
```

3.2 数组名作参数

当程序中需要处理一批相关数据时，往往使用数组。同样，如果实现函数功能时需要获取一批相关数据作为原始信息，则应该考虑使用数组参数。数组名也可以作实参和形参，传递的是数组的起始地址，通过数组传递可以实现双向传递。



1[3313]　10 个元素反序

将数组 array 中的 10 个元素按相反顺序存放。用函数实现。

样例输入 1 2 3 4 5 6 7 8 9 10

样例输出 10 9 8 7 6 5 4 3 2 1

分析一：将数组各元素按相反顺序存放，只需要把 array[0]和 array[9]交换，array[1]和 array[8]交换，…，array[4]和 array[5]交换。题目要求用函数实现，所以定义 reverse 函数，实现将数组 a 中的 n 个元素按相反顺序存放。在主函数中输入数组 array 各元素值，然后调用 reverse 函数实现逆序存放，最后在主函数中输出 array 各

元素的值。

分析二：reverse 函数的原型为：

```
void reverse( int x[ ], int n );
```

reverse 函数的第 1 个形参为数组名的形式

```
#include<bits/stdc++.h>
using namespace std;
void reverse( int a[ ], int n ){
    int i, j;        //循环变量
    int temp;   //交换 a[i]和 a[j]时用到的临时变量
    for( i=0; i<n/2; i++ )
    {
        j = n-1-i;
        temp = a[i];  a[i] = a[j];  a[j] = temp;    //交换 a[i]和 a[j]
    }
}
int  main( )
{
    int array[10], i;
    for( i=0; i<10; i++)  scanf( "%d", &array[i] ); //输入
    reverse( array, 10 );
    for( i=0; i<10; i++)  printf( "%d ", array[i] ); //输出
    printf( "\n" );
    return 0;
}
```

main 函数中调用 reverse 函数中后，数组各元素的值发生了变化，比如原来 array[0]的值为 1，reverse 函数执行后，array[0]的值变为 9 了。

思考：这一点是否跟“实参对形参的数据传递是单向的，只由实参传给形参，不能由形参传回来给实参”是否有矛盾？

数组名表示整个数组所占存储空间的首地址，因此在 reverse 函数调用语句：

```
reverse( array, 10 );
```

中，**数组名作函数实参，传递给形参 a，这样，形参 a 也表示这段存储空间的首地址**。如图所示。在 reverse 交换 a[i]和 a[j]的值，实际上交换的就是 array[i]和 array[j]的值。

关于用数组名作函数参数的两点说明：

(1) 如果函数实参是数组名，形参也应为数组名(或指针变量)，形参不能声明为普通变量(如 int a)。

(2) 形参采用数组名的形式，但实际上，声明形参数组并不意味着真正建立一个包含若干元素的数组，在调用函数时也不会分配数组存储单元（只为形参指针变量分配存储空间；注意，形参数组名实际上是一个指针变量），只是用 a[ ]这样的形式表示 array 是一维数组名，以接收实参传来的地址。因此 a[ ]中方括号内的数值并无实际作用，编译系统对一维数组方括号内的内容不予处理。形参一维数组的声明中可以写元素个数，也可以不写。

```
void reverse ( int a[10], int n )   //指定元素个数与实参数组相同
void reverse ( int a[  ], int n )   //不指定元素个数
```

```
void reverse ( int a[5], int n )    //指定元素个数与实参数组不同
```

3.3 引用传递

C++ 通过取地址运算符“&”引入了一种新的数据类型，即引用类型。引用类型定义格式为：

类型 & 变量；

引用就是某一变量的一个别名，对引用的操作与对变量直接操作完全一样。

例：char ch;

char &rp=ch;

1)引用仅是变量的别名,而不是实实在在地定义了一个变量,因此引用本身并不占用内存,而是和目标变量共同指向目标变量的内存地址。

2）表达式中的取地址符&不再是取变量的地址，而是用来表示该变量是引用类型的变量。

3)定义一个引用时，必须对其初始化。

阅读程序，写出程序的运行结果，体会函数的引用调用

```
#include<iostream>
using namespace std;
void swap(int &a,int &b){
    int temp;
    temp = a;
    a = b;
    b = temp;
    cout << a << “ ” << b << endl;
}
int main(){
    int a = 10,b = 50;
    swap(a,b);
    cout << a <<“ ”<< b << endl;
    return 0;
}
```

3.4 案例讲解

2[1053] 查找最大值

编制函数，其功能是在 double 类型一维数组中查找最大值、最小值，并将它们返回到调用程序。 输出保留两位小数

输入：n 及 n 个浮点数　　输出：最大值 最小值

样例输入

10

1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0

样例输出：10.00 1.00

分析

（1）函数是否有参数，需要几个参数？_____

（2）函数是否有返回值？_____

```
#include<bits/stdc++.h>
using namespace std;
double a[100],min1=1000,max1=-1000;
_1__________ slove(_2_________________)
{
    int i;
    min1=100,max1=-100;
    for(i=1;i<=n;i++){
        if(a[i]<=min1)
            min1=a[i];
        if(a[i]>=max1)
           max1=a[i];
    }

}
int main()
{
    int i,n;
    cin>>n;
    for(i=1;i<=n;i++)
        cin>>a[i];
    __3_____________
    cout.precision(2);
    cout<<fixed<<max1<<" "<<min1;
    return 0;
}
```

3 [2258] 交换数组中最小值的函数

写一个函数 array_min(int a[ ], int n)，将最小值与第一个位置的值做交换。在主函数中，输入一个正整数 n(1<n<=10)，再输入 n 个整数作为数组元素，输出交换以后的整个数组。

输入

输出

样例输入

5

1 2 5 4 0

样例输出

0 2 5 4 1

```
#include <bits/stdc++.h>
using namespace std;
void swap_min( __1_____________________ ){
        int min=b[1],k=1;
        for(int i=1;i<=n;i++){
                if(b[i]<min){
                        min=b[i];
                        k=i;
                }
                int t;
                t=b[1],b[1]=b[k],b[k]=t;
        }
}
int main()
{
        int a[100],n;
        cin>>n;
        for(int i=1;i<=n;i++)
                cin>>a[i];
          __2_____________________;
        for(int i=1;i<=n;i++)
                cout<<a[i]<<"   ";
        return 0;
}
```

4 [2262] 在数组中查找指定元素

题目描述

在数组中查找指定元素。输入一个正整数 n(1<n<=10)，然后输入 n 个整数存入数组 a 中。再输入一个整数 x ，在数组 a 中查找 x，如果找到则输出相应的下标，否则输出“Not found”。要求定义并调用函数 search(list, n, x)，它的功能是在数组 list 中查找元素 x ，若找到则返回相应的下标，否则返回-1。

样例输入

```
3
1  2  -6
2
```

样例输出

```
1
```

分析

（1）函数是否有参数，需要几个参数？_____

（2）函数是否有返回值？_____

```
#include<iostream>
using namespace std;
int search(int list[], int _1_____, int _2_____){
    for (int i = 0; i < n; i++){
        if (list[i] == x)
            return _3_____;
    }
    return _4_____;
}
 int main(){
    int n, x;
    int list[11];
    cin >> n;
    for (int i = 0; i < n; i++){
        cin >> list[i];
    }
    cin >> x;
    int y;
    y=__5_____________________;
    if (y == -1)
        cout << "Not found" << endl;
    else
        cout << y << endl;
    return 0;
}
```

5 [1468] 移动 n 个数

有 n 个整数(n<100)，使前面各数顺序向后移 m 个位置，最后 m 个数变成前面 m 个数。写一函数：实现以上功能，在主函数中输入 n 个数和输出调整后的 n 个数。（如果不是函数体，就不用函数）

输入输入数据的个数 n 个整数 移动的位置 m

输出移动后的 n 个数

样例输入

10

1 2 3 4 5 6 7 8 9 10

2

样例输出

9 10 1 2 3 4 5 6 7 8

分析

（1）函数是否有参数，需要几个参数？_____

（2）函数是否有返回值？_____

```
#include<bits/stdc++.h>
using namespace std;
void slove(int a[],int n,int m){
    int b[105];
    for(int i=1;i<=n-m;i++)
        b[i+m]=a[i];//a 的前 n-m 的元素赋值到 b 的后面
    for(int i=1;i<=m;i++)
         b[i] =a[n-m+i] ;//a 的后面 m 个元素赋值给 b 的前面
    for(int t=1;t<=n;t++)
        cout<<b[t]<<" ";
}
int main()
{
    int a[105],n,m;//  生成新数组新数组 b
    cin>>n;
    for(int i=1;i<=n;i++)
        cin>>a[i];
    cin>>m;
    slove(a,n,m);
    return 0;
}
```

6[2259] 排序

写一个函数 array_min(int a[ ], int low, int hig)，其中 low<=hig，求出 a[low]到 a[hig]的最小值并与 a[low]交换。在主函数中，输入一个正整数 n(1<n<=10)，再输入 n 个整数，将它们按从小到大排序后输出。

样例输入

```
5
8  2  5  1  4
```

样例输出

```
1  2  4  5  8
```

```
#include<iostream>
using namespace std;
void array_min(int a[],int low,int hig)
{
1    int i,min,index=low,temp;
2    min=a[low];
3    for(i=low+1;i<=hig;i++)
    {
4        if(a[i]<min)
5       {//找到当前最小的数和第一个作交换
6           min=a[i];
```

```
7           index=i;
        }
    }
8   temp=a[low];a[low]=a[index];a[index]=temp;
}
int main()
{

    int n,a[10],i;
    scanf("%d",&n);
    for(i=0;i<n;i++)scanf("%d",&a[i]);
9    for(i=0;i<n-1;i++)
10       array_min(a,i,n-1);
    for(i=0;i<n;i++)printf("%d  ",a[i]);
        return 0;
}
```

课堂作业列表

[3313] 10 个元素反序　　[1053] 查找最大值
[2262] 在数组中查找指定元素　　[2259] 排序
3556 孪生素数　　3338 回文质数

课前练习

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string a, t;
    int i, j;
    a = "NOIP2013";
    j = 0;
    for (i = 1; i <= 7; i++)
        if (a[j] > a[i])
            j = i;
    j = j - 2;
    for (i = 0; i <= j; i++)
        cout << a[i];
    return 0;
}
```

输出：_____

```
#include <iostream>
using namespace std;
int s, i;
int f1(int n)
{
    int j, t=1;
    for (j = n; j >= 1; j--)
        t = t * j;
    return t;
}
int main()
{
    s = 0;
    for (i = 1; i <= 5; i++)
        s += f1(i);
    cout << "s=" << s << endl;
    return 0;
}
```

输出：_____

习题

1. 当调用函数时，实参是一个数组名，则向函数传送的是（　　）

　　A） 数组的长度　　B） 数组的首地址

　　C） 数组每一个元素的地址　D）数组每个元素中的值

2.以下程序

```
#include<iostream>
using namespace std;
void fun(int a[], int n)
{ int i, t;
for(i=0; i<n/2; i++) {t=a[i]; a[i]=a[n-1-i]; a[n-1-i]=t;}
}
main()
{ int k[10]={1,2,3,4,5,6,7,8,9,10}, i;
fun(k,5);
for(i=2; i<8; i++) printf("%d", k[i]);
printf("\n");
}
```

程序的运行结果是（　　）。

A）345678　　B）876543　　C）1098765　　D）321678

3. 有以下程序

```
#include<iostream>
using namespace std;
void f(int b[])
{int i;
for(i=2;i<6;i++) b[i]*=2;
}
main()
{int a[10]={1,2,3,4,5,6,7,8,9,10},i;
f(a);
for(i=0;i<10;i++)
    printf("%d ",a[i]);
}
```

程序运行后的输出结果是（　　）

A）1，2，3，4，5，6，7，8，9，10，　　B）1，2，6，8，10，12，7，8，9，10

C）1，2，3，4，10，12，14，16，9，10，　D）1，2，6，8，10，12，14，16，9，10，

4. 有以下程序

```
  void change(int k[ ]){k[0]=k[5];}
  main()
  {int x[10]={1,2,3,4,5,6,7,8,9,10},n=0;
  while(n<=4) {change(&x[n]);n++;}
  for(n=0;n<5;n++) printf("%d",x[n]);
```

```
  printf("\n");
  }
```
程序运行后输出的结果是(　　)

A）678910　　B）13579　　C）12345　　D）62345

5. 以下程序运行输出后的结果是(　　)

```
main()
{  int a,num=16118,count=0;
  do
  {   a=num%10;
      if(a==1)  count++;
      num=num/10;
    while   (num);
    printf("count=%d",count);
}
```

6. 执行以下程序段后，变量 m 中的值是(　　)

```
int  x[10]={ 77,34,85,74,12,21,64,90,101,9 },i,m;
m=0;
for ( i=1;i<10;i++ )  if( x[i]>x[m] ) m=i;
```

A)101　　B)9　　C)10　　D)8

7.倒置数组，n 为元素个数，如输入 1,2,3,4,5　输出 5,4,3,2,1

```
#include<bits/stdc++.h>
using namespace std;
void trans(_1_____________)
{   int i,t;
    for(i=_2__________;i++)
    {   t=a[i];
        _3___________
        _4____________=t;
    }
}
int main()
{   int a[10],i;
    for(i=0;i<10;i++)
        a[i]=i;
    trans(5_____________);
    for(i=0;i<10;i++)
        printf("%d ",a[i]);
    return 0;
}
```

8 执行如下程序段，打印输出的内容是：1 3

```
#include<iostream>
void fun (int c, int *d) {
    c++;
    (*d)++;
```

```
}
int main ( ){
    int a=1, b=2;
    fun(a, &b);
    printf("%d, %d", a, b);
    return 0;
}
```

9、填空：在数组中查找指定元素。输入一个正整数 n(1<n<=10)，然后输入 n 个整数存入数组 a 中。再输入一个整数 x ，在数组 a 中查找 x，如果找到则输出相应的下标，否则输出“Not found”。

样例输入

```
3
1  2  -6
2
```

样例输出

```
1
```

```
#include<stdio.h>
int f(int a[ ],int n,int x);
int main()
{
        int n,a[10],x;
        scanf("%d",&n);
        for(int i=0;i<n;i++)
                scanf("%d",&a[i]);
        scanf("%d",&x);
  int c=__1_____________
          if(_____2_____   )
                printf("%d",c);
        else
                printf("Not found");
        return 0;
}
____3_____________
{
        for(int i=0;i<n;i++){
                if(x==a[i])
                        _________4____
        }
        return -1;
}
```

第四讲 变量范围

变量按其在程序中的作用范围，分为全局变量和局部变量。全局变量是指定义在任何函数之外的变量，也就是不被任何“｛函数体｝”所包含，可以被源文件中其他函数所共用，用静态数据区存储，作用域（有效范围）是从定义变量的位置开始到源文件（整个程序）结束。局部变量是指在一个函数（包括 main 函数）内部定义的变量，它只在本函数内部有效，其他函数不能使用这些变量，用动态数据区存储，函数的参数也是局部变量。

4.1 局部变量

在一个函数内部定义的变量被称为局部变量，也叫做内部变量。形式参数也是一种局部变量。局部变量只在定义它的函数范围内有效，即在此函数之外无法使用这些变量。

例 变量作用域示例。
下列程序试图在主函数中随机产生一维数组的 10 个元素并打印输出，调用 trans 函数将数组倒置，再在主函数打印输出。

```
#include<bits/stdc++.h>
using namespace std;
void trans()                    //倒置数组
{   int i,t;
    for(i=0;i<5;i++)
    {   t=a[i];
        a[i]=a[9-i];
        a[9-i]=t;
    }
}
int main()
{   int a[10],i;
    for(i=0;i<10;i++)
    {   a[i]=i;
        printf("%d ",a[i]);
    }
    trans();
    for(i=0;i<10;i++)
        printf("%2d",a[i]);
    return 0;
}
```

在 dev c++环境下编译时出现如图所示的错误信息。

21	11	C:\Users\win10\Desktop\未命名2.cpp	[Error] 'a' was not declared in this scope

从编译出错信息，可以看出是符号 a 在 trans 函数中未定义的错误。虽然在 main 函数

中已经定义了数组 a，但它无法在 trans 函数中使用，即数组 a 的作用域未包含 trans 函数。

如果在 trans 函数中增加一条“int a[10];”的定义语句，虽然编译时不再提示出错，但仍然无法得到正确的运行结果。为什么会出现这种情况？因为 main 函数和 trans 函数中的数组 a 是两个不同的数组，它们对应不同的存储空间，作用域分别局限于各自所在的函数。这就能够解释为什么前面的程序无法得到正确的运行结果了。

在一个函数内部，还可以在复合语句中定义变量，其作用域只局限于该复合语句，例如，

```
for(i=0;i<5;i++)
{   int t;          //t 作用域的开始
    t=a[i];
    a[i]=a[9-i];
    a[9-i]=t;       //t 作用域的结束
}
```

一般要求将局部变量的定义放在函数的开始，尽量不要在函数中间定义局部变量。复合语句中的局部变量一般用于小范围内的临时变量。因为**函数中的局部变量只在本函数中有效，所以可以在不同函数中定义同名的局部变量而不会带来混乱**。正如每个班级中都有一位班长，他们的职责各自各不相同，互不干涉对方；但同一函数中不能定义两个同名的局部变量，正如同一个班级中不能有两个班长一样，因为这将带来管理上的混乱。

4.2 全局变量

全局变量又称为外部变量，它是在函数之外定义的变量，其作用范围为从定义的位置开始到本源程序文件的结束。通过全局变量可以在多个函数间共享数据。可以将例 2 的程序作如下修改。

```
#include<bits/stdc++.h>
using namespace std;
int a[10];                      //a 作用域开始
void trans()                    //倒置数组
{   int i,t;
    for(i=0;i<5;i++)
    {   t=a[i];
        a[i]=a[9-i];
        a[9-i]=t;
    }
}
int main()
{   int a[10],i;
    for(i=0;i<10;i++)
    {   a[i]=i;
        printf("%d ",a[i]);
    }
    trans();
    for(i=0;i<10;i++)
```

```
        printf("%d ",a[i]);
    return 0;
}                               //a 作用域结束
```

全局数组 a 首先在 main 函数中被赋值，然后在 trans 函数中进行倒置操作，最后再在 main 函数中输出倒置后的结果。显然，全局变量增加了函数之间的数据传递通道，使得函数之间的数据联系不只局限于参数传递和 return 语句，通过全局变量可以从函数中得到一个以上的返回值。**在全局变量的作用域中，所有函数均可直接访问全局变量，也就是说，如果在一个函数中改变了全局变量的值，就能影响到其他函数。**这也使得程序容易出错，且很难清楚判断出错原因。因此，建议读者慎用全局变量。

全局变量就像一柄双刃剑，它既增强了函数之间的数据联系通道，同时也降低了函数的通用性。因为函数的执行依赖于全局变量，为了将函数移植到其他文件中，就必须将所涉及的全局变量一并移植，此时有可能出现原有全局变量与新文件中的全局变量同名的问题，从而降低了程序的可靠性。事实上，设计一个好的函数应当遵循“黑箱”观点，即所有的输入是以参数的形式传递给函数的，所有的输出是以函数值的形式返回，函数中所使用的变量都是局部变量，与调用者没有任何关系。

按照此原则，上例的程序最好改写成如下形式。

```
#include<bits/stdc++.h>
using namespace std;
void trans(int a[],int n)   //倒置数组，n 为元素个数
{   int i,t;
    for(i=0;i<n/2;i++)
    {   t=a[i];
        a[i]=a[n-i-1];
        a[n-i-1]=t;
    }
}
int main()
{   int a[10],i;
    for(i=0;i<10;i++)
    {   a[i]=i;
        printf("%d ",a[i]);
    }
    trans(a,10);
    for(i=0;i<10;i++)
        printf("%d ",a[i]);
    return 0;
}
```

此外，全局变量可以与某个函数中的局部变量同名，此时，在该函数内部局部变量的作用域将“覆盖”同名全局变量的作用域。

练习：

以下程序的结果是_____。

```
#include<iostream>
int a,b;
void fun()
{
  a=100;
  b=200;
}
int main()
{
    int  a=5,b=7;
    fun();
    printf("%d%d\n",a,b);
    return 0;
 }
```

执行以下程序，打印输出的内容是：_____

```
#include<iostream>
int x=5, y=7;
void swap( ){
    int z;
    z=x;  x=y;  y=z;
}
int main( ){
    int x=3, y=8;
    swap( );
    printf("%d, %d\n", x, y);
    return 0;
}
```

以下程序的输出的结果是____。

```
#include<stdio.h>
int x=3;
void incre();
int main()
{
    int i;
    for (i=1;i<x;i++)  incre();
    return 0;
}
 void incre()
{
     static  int  x=1;
     x*=x+1;
     printf("  %d",x);
}
```

执行以下程序，打印输出的内容是：____。

```
#include<iostream>int x=5, y=6;
void incxy( ){
    x++; y++;
}
int main( ){
    int x=3;
    incxy( );
    printf("%d,%d\n", x,y);
    return 0;
}
```

4.3 静态变量

系统在程序运行期间为其分配固定存储空间的变量称为静态存储变量。全局变量就是一种静态存储变量，系统在程序开始执行时为其分配存储空间，直到程序执行完毕才释放，因此整个程序执行期都是全局变量的生存期。

有时可能希望在函数调用结束后保留其局部变量的值，即不释放局部变量所占据的存储空间，从而在下一次调用该函数时可继续使用上一次调用结束时的结果。这就需要**使用 static 关键字将局部变量的存储方式声明为静态存储方式，这种变量称为静态局部变量**。相应地，使用 auto 关键字或缺省定义的局部变量称为动态局部变量。

练习

1. 将一个函数说明为 static 后，该函数将（　　）。

A. 既能被同一源文件中的函数调用，也能被其他源文件中的函数调用

B 只能被同一源文件中的函数调用，不能被其他源文件中的函数调用

C. 只能被其他源文件中的函数调用，不能被同一源文件中的函数调用

D. 既不能被同一源文件中的函数调用，也不能被其他源文件中的函数调用

2. 以下程序的结果是_____。

```
#include<stdio.h>
int f(int    a){
      int    b=0;
      static int c=3;
      b++;
      c++;
      return(a+b+c);
}
 int main()
 {int    a=2, i;for(i=0;i<3;i++)
      printf("%d ",f(a));return 0;
 }
```

3 编程计算$1-\frac{1}{2!}+\frac{1}{3!}-\frac{1}{4!}+\cdots+\frac{(-1)^{n-1}}{n!}$，精度为 0.000001。

编程分析：编写一个 term 函数利用静态局部变量计算 n!。

程序代码如下。

```
#include<iostream>
double term(int n);
void main()
{     int i=1,k=1;
      double s=0.0,t;
      do
      {     t=term(i);
            s=s+k/t;
            i++;
            k=-k;
      }while(1.0/t>1e-6);
      printf("%lf\n",s);
}
double term(int n)
{     static    double t=1.0;                    //静态局部变量的声明与初始化
      t=t*n;
      return(t);
}
```

程序运行结果如下。 0.632121

注意：term 函数中静态局部变量 t 是在声明的同时进行初始化的，不能写成以下 两句。
static double t;
t=1.0;

因为静态局部变量只在编译时赋一次初值，即“static double t=1.0；”语句只执行一次，以后每次调用该函数时都直接使用上一次调用结束后静态局部变量的值。如果改成上面两句，则每调用一次函数都会执行一次“t=1.0；”语句，这显然不是程序设计的初衷。

由此可见，静态局部变量的生存期是从该变量被分配存储空间开始，即定义该变量的函数被调用开始，直到程序结束。但是，静态局部变量的作用域仍然局限于定义它的函数，静态局部变量在上一次调用结束后被保留下来的值，也只能在该函数的下一次调用中使用，而不能被其他函数使用。

模仿上面代码，求解[1055] 和 [1044]

4.4 案例分析

1 [2258] 交换数组中最小值的函数

用全局变量实现：写一个函数 array_min(int a[ ], int n)，将最小值与第一个位置的值做交换。在主函数中，输入一个正整数 n(1<n<=10)，再输入 n 个整数作为数组元素，输出交换以后的整个数组。
输入
输出
样例输入
5
1 2 5 4 0
样例输出
0 2 5 4 1

```
#include <bits/stdc++.h>
using namespace std;
void swap_min(int b[],int n){
        int min=b[1],k=1;
        for(int i=1;i<=n;i++){
                if(b[i]<min){
                        min=b[i];
                        k=i;
                }
                int t;
          t=b[1],b[1]=b[k],b[k]=t;

        }
}
```

全局变量改写：

```
int main()
{
        int a[100],n;
        cin>>n;
        for(int i=1;i<=n;i++)
                cin>>a[i];
        swap_min(a,n);
        for(int i=1;i<=n;i++)
                cout<<a[i]<<"   ";
        return 0;
}
```

2 [1468] 移动 n 个数

有 n 个整数(n<100)，使前面各数顺序向后移 m 个位置，最后 m 个数变成前面 m 个数。写一函数：实现以上功能，在主函数中输入 n 个数和输出调整后的 n 个数。（如果不是函数体，就不用函数）

输入 输入数据的个数 n 个整数 移动的位置 m

输出 移动后的 n 个数

样例输入

10

1 2 3 4 5 6 7 8 9 10

2

样例输出

9 10 1 2 3 4 5 6 7 8

```
#include<bits/stdc++.h>
using namespace std;
void slove(int a[],int n,int m){
    int b[105];
    for(int i=1;i<=n-m;i++)
        b[i+m]=a[i];
    for(int i=1;i<=m;i++)
         b[i] =a[n-m+i] ;
    for(int t=1;t<=n;t++)
        cout<<b[t]<<" ";
}
int main()
{
    int a[105],n,m;
    cin>>n;
    for(int i=1;i<=n;i++)
        cin>>a[i];
    cin>>m;
```

全局变量改写：

slove(a,n,m);
return 0;
}	

3［2679］矩阵交换行

给定一个 5×5 的矩阵(数学上，一个 r×c 的矩阵是一个由 r 行 c 列元素排列成的矩形阵列)，将第 n 行和第 m 行交换，输出交换后的结果。

输入　输入共 6 行，前 5 行为矩阵的每一行元素,元素与元素之间以一个空格分开。第 6 行包含两个整数 m、n，以一个空格分开（1 ≤ m,n ≤ 5）。
输出　输出交换之后的矩阵，矩阵的每一行元素占一行，元素之间以一个空格分开。
样例输入
1 2 2 1 2
5 6 7 8 3
9 3 0 5 3
7 2 1 4 6
3 0 8 2 4
1 5
样例输出
3 0 8 2 4
5 6 7 8 3
9 3 0 5 3
7 2 1 4 6
1 2 2 1 2

#include<bits/stdc++.h>
using namespace std;
int main(){
int a[6][6],i,j;
int n,m,t;
for(i=1;i<=5;i++)
for(j=1;j<=5;j++)
cin>>a[i][j];
cin>>n>>m;
for(i=1;i<=5;i++)
t=a[n][i],a[n][i]=a[m][i],a[m][i]=t;
for(i=1;i<=5;i++){
for(j=1;j<=5;j++)
cout<<a[i][j]<<" ";
cout<<endl;
}
return 0;
	用全局变量和函数改写：

4 [1449] 求最大公约数和最小公倍数

写两个函数，分别求两个整数的最大公约数和最小公倍数，用主函数调用这两个函数，并输出结果两个整数由键盘输入。
样例输入 6 15
样例输出 3 30

分析：如何求解最大公约数？

方法 1：枚举，从大到小一一枚举最大公约数[min(m,n),1].

方法 2：辗转相除法

```
#include<bits/stdc++.h>
using namespace std;
int f(int m,int n)
{// 15 20
    int t;
    while(n!=0){
        int t;
        t=m%n;
        m=n,n=t;
    }
    return m;
}
int main ()
{
    int a,b,s,k;
    cin>>a>>b;
    s=a*b;
    k=f(a,b);
    cout<<k<<" ";
    cout<<s/k<<endl;
    return 0;
}
```

5 [3556] 孪生素数

桐桐把大小之差不超过 2 的两个素数称为一对孪生素数，如 2 和 3、3 和 5、17 和 19 等等。请你帮助桐桐统计一下，在不大于自然数 N 的素数中，孪生素数的对数。
输入 一个自然数 N(N>=4&&N≤10^6)。
输出 一个整数，表示 N 以内孪生素数的对数。
样例输入 20
样例输出 5

分析：

1. 定义一个求解素数的函数
2.

```
#include <bits/stdc++.h>
using namespace std;
bool isprime(int x)
{
        for(int i=2;i*i<=x;i++)
        {
                if(x%i==0) return 0;
        }
        return 1;
}
int main()
{
        int n,ans=0;
        cin>>n;
        for(int i=2;i<=n-2;i++)
        {
                if(i==2)
                {
                        ans++;
                        continue;
                }
                if(_____________________________ )
                {
                        ans++;
                }
        }
        cout<<ans<<endl;
        return 0;
  }
```

作业列表

[2258] 交换数组中最小值的函数　　[1468] 移动 n 个数
[2679] 矩阵交换行　　[1449] 求最大公约数和最小公倍数
[3556] 孪生素数

课前练习

```
#include<iostream>
void fun(int a[], int n)
{
    int i, t;
    for(i=0; i<n/2; i++) {
      t=a[i];
      a[i]=a[n-1-i];
      a[n-1-i]=t;
    }
}
main()
{
  int k[10]={1,2,3,4,5,6,7,8,9,10};
  int i;
  fun(k,5);
  for(i=2; i<8; i++)
       printf("%d", k[i]);
  printf("\n");
}
程序的运行结果是(       )
```

阅读程序

```
#include <iostream>
using namespace std;
int s, i;
int f1(int n)
{
    int j, t;
    t = 1;
    for (j = n; j >= 1; j--)
        t = t * j;
    return t;
}

int main()
{
    s = 0;
    for (i = 1; i <= 5; i++)
        s += f1(i);
    cout << "s=" << s << endl;
    return 0;
}
```

输出：_____

习题

1. 如果在一个函数中的复合语句中定义了一个变量，则该变量（　　）。

A. 在该函数中有效　　B. 只在该复合语句中有效

C. 在该程序范围内均有效　　D. 为非法变量

2. 有以下函数调用语句： Func(rec1 , rec2+rec3 , (rec4,rec5)); 该函数调用语句中含有的实参个数是（ ）

A. 3　　B. 4　　C. 5　　D. 有语法错误

3. 能把函数处理结果的两个数据返回给主调函数，下面的方法中不正确的是（　　）

A. return 数组的首址　　B. 形参用数组

C. 用两个全局变量　　D. return 两个数

4. 文件中定义的全局变量的作用域为（D）。

A. 本程序全部范围　　B. 本文件全部范围

C. 函数内全部范围　　D. 从定义该变量的位置开始到本文件结束

5. 执行下面程序，输出是 _____

```
#include<iostream>
using namespace std;
int x = 5, y = 7;
```

```
void swap ( )
{     int z ;
      z = x ;  x = y ;  y = z ;
}
int main( )
{
      int x = 3, y = 8;
      swap ( ) ;
      printf ( " %d , %d \n", x , y ) ;
      return 0 ;
}
```

6. 以下程序的结果是_____。

```
#include<iostream>int  a,b;
void fun()
{
  a=100;
  b=200;
}
int main()
{
    int  a=5,b=7;
    fun();
    printf("%d%d\n",a,b);
    return 0;
 }
```

7. 以下程序的结果是_____。

```
#include<stdio.h>
int f(int  a)
{
    int  b=0;
    static int c=3;
    b++;
    c++;
    return(a+b+c);
 }
 int main()
 {
    int  a=2, i;
    for(i=0;i<3;i++)
        printf("%d ",f(a));
    return 0;
 }
```

8. 执行下面程序，输出是 _____

```
#include<iostream>int x1 = 30, x2 = 40;
```

```
sub(int x, int y) {
  x1 = x; x = y; y = x1;
}
int main()
{
  int x3 = 10, x4 = 20;
  sub(x3, x4);
  sub(x2, x1);
  printf("%d,%d,%d,%d\n", x3, x4, x1, x2);
  return 0;
}
```

9. 执行以下程序，打印输出的内容是:_____

```
#include<iostream>int x=5, y=7;
void swap( ){
    int z;
    z=x;  x=y;  y=z;
}
int main( ){
    int x=3, y=8;
    swap( );
    printf("%d, %d\n", x, y);
    return 0;
}
```

10. 编程求 2～n(n 为大于 2 的正整数)中有多少个素数。

```
#include<bits/stdc++.h>
using namespace std;
___1___
{
    for(int i=2;i<=sqrt(x);i++)
      if(x%i==0)
          __2_____
   __3_____
}
int main ()
{
    int n,ans=0;
    cin>>n;
    for(int i=2;i<=n;i++)
      if(s(i))
        ans++;
     __4_____
    return 0;
}
```

第五讲 系统函数

c++提供了很多系统函数，常用的包括数学函数、输出输出函数、字符串处理了函数、STL 模板库等。

5.1 数学函数

常用的数学函数包括如下：

```
int abs(int x); //求整数的绝对值
long labs(long x); //求长整型数的绝对值
double fabs(double x); //求实数的绝对值
double floor(double x); //求不大于 x 的最大整数
double celi(double x);//求不小于 x 的最小整数
double sqrt(double x);//求 x 的平方根
double log10(double x);//求 x 的常用对数
double log(double x);//求 x 的自然对数
double exp(double x);//求欧拉常数 e 的 x 次方
double pow(double x.double y);//求 x 的 y 次方
double sin (double);
double cos (double);
double tan (double);
```

具体使用如下：

```
abs(-100.01) = 100.01
fabs(-10.11) = 10.11
floor(12.2) = 12
ceil(32.19) = 33
pow(2, 3) = 8
sqrt(4) = 2
log(100) / log(10) = 2
sin(90) = 0.893997
cos(60) = -0.952413
tan(45) = 1.61978
asin(1) = 1.5708
round(10.45753) = 10
```

5.2 输入输出函数

C++ 标准库提供了一组丰富的输入/输出功能。如果字节流是从设备（如键盘、磁盘驱动器、网络连接等）流向内存，这叫做输入操作。如果字节流是从内存流向设备（如显示屏、打印机、磁盘驱动器、网络连接等），这叫做输出操作。

<iostream>	该文件定义了 cin、cout 对象，分别对应于标准输入流、标准输出流

<iomanip>	该文件通过所谓的参数化的流操纵器（比如 setw 和 setprecision），来声明对执行标准化 I/O 有用的服务

5.2.1 cin 输入

cin 主要用于从标准输入读取数据，这里的标准输入，指的是终端的键盘。

在理解 cin 功能时，不得不提标准输入缓冲区。当我们从键盘输入字符串的时候需要敲一下回车键才能够将这个字符串送入到缓冲区中，那么敲入的这个回车键(\r)会被转换为一个换行符\n，这个换行符\n 也会被存储在 cin 的缓冲区中并且被当成一个字符来计算！比如我们在键盘上敲下了 123456 这个字符串，然后敲一下回车键（\r）将这个字符串送入了缓冲区中，那么此时缓冲区中的字节个数是 7 ，而不是 6。

cin 的常用读取方法

使用 cin 从标准输入读取数据时，通常用到的方法有 cin>>，cin.get，cin.getline。

（1）cin>>的用法

cin 可以连续从键盘读取想要的数据，以空格、tab 或换行作为分隔符。实例程序如下。

```
#include <iostream>
using namespace std;
int main()
{
    char a;
    int b;
    float c;
    cin>>a>>b>>c;
    cout<<a<<" "<<b<<" "<<c<<" "<<endl;
    system("pause");
    return 0;
}
```

当 cin>>从缓冲区中读取数据时，若缓冲区中第一个字符是空格、tab 或换行这些分隔符时，cin>>会将其忽略并清除，继续读取下一个字符，若缓冲区为空，则继续等待。但是如果读取成功，字符后面的分隔符是残留在缓冲区的，cin>>不做处理。

（2） cin.get 读取一个字符

读取一个字符，可以使用 cin.get 或者 cin.get(var)，示例代码如下：

1）从结果可以看出，cin.get()从输入缓冲区读取单个字符时不忽略分隔符，直接将其读取，就出现了如上情况，将换行符读入变量 b，输出时打印两次。

2）cin.get()的返回值是 int 类型，成功：读取字符的 ASCII 码值，遇到文件结束符时，返回 EOF，即-1。

(3）cin.get 和 **cin.getline** 都可以读取一行，cin.getline 与 cin.get 的区别是，cin.getline 不会

将结束符或者换行符残留在输入缓冲区中。cin.getline 从标准输入设备键盘读取一串字符串，并以指定的结束符结束，可以读入空格。

#include <iostream>
using namespace std;
 	
int main()
{
char a;
char b;
a=cin.get();
cin.get(b);
cout<<a<<b<<endl;
return 0;
}
	include <iostream>
	using namespace std;
	
	int main()
	{
	char a;
	char array[20]={NULL};
	cin.get(array,20);
	cin.get(a);
	cout<<array<<" "<<(int)a<<endl;
	return 0;
	}
		#include <iostream>
		using namespace std;
		int main()
		{
		char array[20]={NULL};
		cin.getline(array,20);
		//或者指定结束符
		//，使用下面一行
		//cin.getline(array,20,'\n');
		cout<<array<<endl;
		return 0;
		}

5.2.2 cout 格式控制

有时候，我们需要以不同的进制来输出数字，而默认输出是十进制，其他进制输出方法如下：

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
    int i = 90;
    cout << i << endl;
    cout << dec << i << endl;//dec 是十进制（效果和默认一样）
    cout << oct << i << endl;//oct 是八进制输出
    cout << hex << i << endl;// hex 是十六进制输出（字母默认是小写字母
    cout << setiosflags(ios::uppercase);
    cout << hex << i << endl;
}
```

setiosflags(ios::uppercase) 表示将字母大写输出，包含在库 < iomanip > 中。
以上均包含在 std 命名空间中。这个库包含了对输入输出的控制。

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
    double i = 3333.1415926;
    cout << i << endl;
    cout << setprecision(3) << i << endl;
    cout << setprecision(9) << i << endl;
    cout << setiosflags(ios::fixed);
```

```
    cout << i << endl;
    cout << fixed << setprecision(3) << i << endl;
    cout << setprecision(9) << fixed <<    i << endl;
}
```

以看出，C++默认浮点数输出有效位数是 6 位（若前面整数位数大于 6 位，使用科学计数法输出），而通过以下几种方式可以更改输出精度：

1.使用 setprecision(n) 即可设置浮点数输出的有效位数
（若前面整数位数大于 n 位，使用科学计数法输出）

2.使用 setiosflags(ios::fixed) 或 fixed，表示对小数点后面数字的输出精度进行控制所以，和 setprecision(n) 结合使用即可设置浮点数小数点后面数字的输出精度，位数不足的补零

以上均采用“四舍五入”的方法控制精度，三个控制符均包含在 std 命名空间中。

5.2.3 scanf 与 printf

(1) 格式化输出函数 printf

printf 函数的功能是格式化输出任意数据列表，其一般调用格式为：

printf(格式控制符，输出列表)

例如，printf(“%d, %c \n”,i,c); 表示将变量 i 以整数形式输出，变量以支付形式输出，两个输出项之间用一个逗号隔开。

格式控制符是用双引号括起来的字符串。包括两种信息，一种是普通字符，按原样输出；另一种是格式说明，由%和格式字符组成。如%d 的作用是将输出的数据转换成指定的格式输出，具体如下表所示。

格式字符	说明
%d	以十进制形式输出带符号整数，正号(+)不输出
%o	以八进制形式输出无符号整数，不输出前缀 0
%x，%X	以十六进制形式输出无符号整数，不输出前缀 0x
%u	以十进制形式输出无符号整数
%c	以字符形式输出一个字符
%s	输出字符串
%f，%lf	以小数形式输出单、双精度数，隐含输出 6 位小数
%e，%E	以指数形式输出实数
%l	加在格式符 d、o、x、u 前，用于长整型数据

使用说明：

1） 输出整数形式可以用 %d 或 %md，m 为指定的输出字符的宽度，输出数据右对齐。

2） 输出长整型可以用 %ld，一个 int 型数据可以用 %d 或 %ld 格式输出。

3） 输出字符串用 %s，如“printf （ “ %s ” ， “ hello ” ）；”。

4） “%m.nf”，输出浮点数，占 m 列，其中有 n 位小数，如果数值长度小于 m，则左

阅读并上机调试程序，体会 printf 函数的使用。

```
#include<cstdio>
using namespace std;
int main(){
```

```
    printf( "%d\n" ,1); // 输出 1
    printf( "%o\n" ,8); // 输出 10
    printf( "%c\n" ,49); // 输出 1
    printf( "%s\n" , "1ab11" ); // 输出 1ab11
    printf( "%f\n" ,0.14); // 输出 0.140000
    printf( "%lf\n" ,0.14); // 输出 0.140000
    printf( "%.2lf\n" ,0.141234); // 输出 0.14
    printf("a=%d, b=%.3lf",1.23,1.345678);//a=2061584302, b=1.346
}
```

(2) 格式化输入函数 scanf

scanf 函数的功能是格式化输入任意数据列表，其一般调用格式为：

scanf(格式控制符，地址列表)

格式控制符跟 printf 一样；地址列表可以是变量的地址，也可以是字符串的首地址。

例如：int a,b;

scanf(“%d,%d”,&a,&b);

就表示先在内存中各开辟 4 个字节空间给 a 和 b，当遇到 scanf 语句时，就把键盘上输入的 2 个数依次存入 a、b 所在的空间（及地址中）。“&a”就表示取 a 变量的地址，“&”称为取地址符。简而言之，就是先找地址后放值。

使用 scanf 函数时，需要注意以下几个问题：

（1）如果在格式控制字符串中有其它字符，则运行程序输入数据时，对应的位置也要输入这些相同的字符。

例如：scanf(“%d,%d”,&a,&b);

键盘输入应该是“3,4”，而不能是“3 4”。

（2）scanf 函数输入时可以过滤掉不想读入的字符。

例如：scanf(“%d+%d+%d”,&a,&b,&c);

键盘输入：1+2+3

则 scanf 可以无视“+”，使得 a,b,c 的值分别为 1,2,3。

再如：scanf(“%3d %*3d %2d”,&m,&n);

键盘输入：113 118 69

则 m,n 的值分别为 113,69， * 表示本输入项在读入后不赋值给相应的变量，表示跳过相应数据。

遇到大数据时尽量避免用 cin

noip 比赛中坚决不要写 std::ios::sync_with_stdio(false)来优化 cin，使用 scanf

如果是 double 或输入格式较复杂用 scanf

遇到数据量大的题，且是 long long 或 int，尽量用手写的快速读入来读取

5.3 随机函数

rand()函数是 C++中常用的生成随机数的函数，其用法如下：

1. rand()不需要参数，它会**返回一个从 0 到最大随机数的任意整数**，最大随机数的大小通常是固定的一个大整数，RAND_MAX 的范围最少是在 32767 之间(int)。

2. 如果你要产生 0~99 这 100 个整数中的一个随机整数，可以表达为：int num = rand() % 100;

这样，num 的值就是一个 0~99 中的一个随机数了。

3. 如果要产生 1~100，则是这样：int num = rand() % 100 + 1;
4. 总结来说，可以表示为：int num = rand() % n +a;

其中的 a 是起始值，n-1+a 是终止值，n 是整数的范围。

5. 一般性：**rand() % (b-a+1)+ a ;　就表示　a~b 之间的一个随机整数。**
6. 若要产生 0~1 之间的小数，则可以先取得 0~10 的整数，然后均除以 10 即可得到“随机到十分位”的 10 个随机小数。

若要得到“随机到百分位”的随机小数，则需要先得到 0~100 的 10 个整数，然后均除以 100，其它情况依 此类推。

如下代码为 10000000 个 0~10000 的随机数。

```
for(int i=1;i<=10000000;i++){
        cout << rand()%10000<< " ";
    }
```

5.4　字符串处理函数

C++语言提供了丰富的字符串处理函数，大致可分为字符串的输入、输出、合并、修改、比较、转换、复制、搜索几类。用于输入/输出的字符串函数，在使用前应包含头文件“cstdio”；使用其他字符串函数则应包含头文件“cstring”。下面介绍几个最常用的字符串函数。

5.4.1 字符数组串

（1）**测字符串长度函数 strlen( )**

测字符串长度函数格式：strlen(st)

其中，st 可以是已定义的字符数组名，也可以是指向字符串的指针变量。功能：测字符串 st 的实际长度（不含字符串结束标志）并作为函数返回值。

例如，

```
char s[10]="abcde";
printf("%d\n",strlen(s));
```

则输出结果为　5（不是 6，也不是 10）。

（2）字符串连接函数 strcat( )

字符串连接函数 strcat 的调用格式：strcat(st1，st2)

其中，st1、st2 可以是已定义的字符数组名，也可以是指向字符串的指针变量。

功能：把 st2 中的字符串连接到 st1 中字符串的后面，并删去字符串 st1 后的串结束标志“\0”。本函数返回值是字符串 st1 的首地址。

（3）字符串复制函数 strcpy( )

字符串复制函数的使用格式：strcpy(st1，st2)

其中，st1，st2 可以是已定义的字符数组名，也可以是指向字符串的指针变量。

功能：把 st2 指向的字符串复制到 st1 中，串结束标志'\0'也一同复制。st2 也可以是一个字符串常量。如果 st1 是数组名，st2 是字符串常量，这时相当于把一个字符串赋予一个字符数组。

本函数要求字符数组 st1 应有足够的长度，否则不能全部装入所复制的字符串。在例 5-18 中，通过编程方式实现了 strcpy 函数的功能。

（4）字符串比较函数 strcmp( )

字符串比较函数的使用格式：strcmp(st1，st2)

功能：按照 ASCII 码顺序比较两个数组中的字符串，并由函数返回值返回比较结果。

比较规则：对两个字符串自左向右逐个字符比较，直到出现不相同的字符或遇到'\0'为止。**如果全部字符相同，则认为相同；若出现不相同的字符，则以第一个不相同的字符比较结果为准，返回不相同字符的 ASCII 码之差。**

例如，strcmp("ABCDE"，"ABEA")返回字符'C'的 ASCII 码与字符'E'之差，即“-2”，但实际上在编程中不需要使用具体的值，只需要判断它们的关系即可。

1）字符串 st1＝字符串 st2，返回值＝0。

2）字符串 st1>字符串 st2，返回值> 0。

3）字符串 st1<字符串 st2，返回值< 0。

注意：字符串只能用 strcmp 函数比较，不能用关系运算符“= =”比较。

例如，以下写法正确。

```
if (strcmp(st1,st2) = = 0) printf("yes");
if (!strcmp(st1,st2)) printf("equal");
```

以下写法错误。

```
if (st1 = = st2) printf("yes");
```

5.4.2 string 串

一、初始化

初始化有两种方式，其中使用等号的是拷贝初始化，不使用等号的是直接初始化。

```
string str1 = "hello world";      // str1 = "hello world"
string str2("hello world");       // str2 = "hello world"
string str3 = str1;               // str3 = "hello world"
```

二、获取长度（length、size）

length()函数与 size()函数均可获取字符串长度。

```
string str = "hello world";
cout << str.length() << str.size();     // 11   11
```

当 str.length()与其他类型比较时，建议先强制转换为该类型，否则会意想之外的错误。

比如：-1 > str.length() 返回 true。

三、插入（insert）

基本情况为以下四种，其余变形函数自行摸索即可。

```
string str = "hello world";
//s.insert(pos,n,ch)          在字符串 s 的 pos 位置上面插入 n 个字符 ch
str.insert(6,4,'z');          // str = "hello zzzzworld"
//s.insert(pos,str)           在字符串 s 的 pos 位置插入字符串 str
str.insert(6,str2);           // str = "hello hard world"
```

四、赋值（assign）

赋值也是一种初始化方法，与插入、替换、添加对应理解较为简单。

```
string str;
string temp = "welcome to my blog";
//s.assign(n,ch)          将 n 个 ch 字符赋值给字符串 s
str.assign(10,'h');       // str = "hhhhhhhhhh"
//s.assign(str)           将字符串 str 赋值给字符串 s
str.assign(temp);         // str = "welcome to my blog"
```

五、删除（erase）

```
string str = "welcome to my blog";
//s.erase(pos,n)          把字符串 s 从 pos 开始的 n 个字符删除
str.erase(11,3);          // str = "welcome to blog"
```

六、取子串（substr）

```
string str = "The apple thinks apple is delicious";
//s.substr(pos,n)         得到字符串 s 位置为 pos 后面的 n 个字符组成的串
string s1 = str.substr(4,5);          // s1 = "apple"
//s.substr(pos)                       得到字符串 s 从 pos 到结尾的串
string s2 = str.substr(17);           // s2 = "apple is delicious"
```

七、比较（compare）

两个字符串自左向右逐个字符相比（按 ASCII 值大小相比较），直到出现不同的字符或遇'\0'为止。

若是遇到‘\0’结束比较，则长的子串大于短的子串，如：“9856” > “985”。

如果两个字符串相等，那么返回 0，调用对象大于参数返回 1，小于返回-1。

```
string str1 = "small leaf";
string str2 = "big leaf";
//s.compare(str)                比较当前字符串 s 和 str 的大小
cout << str1.compare(str2);                   // 1
//s.compare(pos,n,str)          比较当前字符串 s 从 pos 开始的 n 个字符与 str 的大小
cout << str1.compare(2,7,str2);               // -1

//s.compare(pos,n0,str,pos2,n)      比较当前字符串 s 从 pos 开始的 n0 个字符与 str 中 pos2 开始的 n 个字符组成的字符串的大小
cout << str1.compare(6,4,str2,4,4);           // 0
```

八、交换（swap）

交换两个字符串的值。

```
string str1 = "small leaf";
string str2 = "big leaf";
//或者 str1.swap(str2) ,输出结果相同
swap(str1,str2);      // str1 = "big leaf"     str2 = "small leaf"
swap(str1[0],str1[1]); // str1 = "ibg leaf"
```

九、反转（reverse）

反转字符串。

```
string str = "abcdefghijklmn";
reverse(str.begin(),str.end());        // str = "nmlkjihgfedcba"
```

十四、查找（find）

十、 find 函数

```
string str = "The apple thinks apple is delicious";     //长度 34
string key = "apple";
//s.find(str)       查找字符串 str 在当前字符串 s 中第一次出现的位置
int pos1 = str.find(key);                    // 4
//s.find(str,pos)    查找字符串 str 在当前字符串 s 的[pos,end]中第一次出现的位置
int pos2 = str.find(key, 10);                // 17
```

5.5 案例分析

1 [2257] 字符串反序

写一个函数，使输入的一个字符串（少于 80 个字符）按反序存放，在主函数中输入输出字符串。

样例输入 reverse a string

样例输出 gnirts a esrever

分析

（1）函数是否有参数，需要几个参数？_____

（2）函数是否有返回值？_____

写法 1：string 写法

```
#include<bits/stdc++.h>
using namespace std;
void rev(string &str){
    string a;
    int len=str.length();
    for(int i=0;i<len/2;i++)
        swap(str[i],str[len-i-1]);

}
int main ()
{
    string str1=" ",str2=" ";
    getline(cin,str1);
    rev(str1);
    for(int
```

写法 2：字符数组写法

```
#include<bits/stdc++.h>
using namespace std;

void rev(char str[]){
    int len=strlen(str);
    for(int i=0;i<len/2;i++)
        swap(str[i],str[len-i-1]);
}
int main ()
{
    char str[200];
    gets(str);
    rev(str);
    for(int i=0;str[i]!='\0';i++)
        cout<<str[i];
```

写法 3：用全局变量实现

```
#include<bits/stdc++.h>
using namespace std;
char s[100],a[100];
int l;
char ch()
{
    for(int i=0;i<l;i++)
        a[i]=s[l-i-1];
}
int main ()
{
    gets(s);
    l=strlen(s);
    ch();
    for(int i=0;i<l;i++)
```

i=0;str1[i]!='\0';i++) cout<<str1[i]; return 0; }	return 0; }	cout<<a[i]; return 0; }

2 [1900] 查找最大元素

对于输入的每个字符串，查找其中的最大字母，在该字母后面插入字符串“(max)”。
输入输入数据包括多个测试实例，每个实例由一行长度不超过 100 的字符串组成，字符串仅由大小写字母构成。
输出对于每个测试实例输出一行字符串，输出的结果是插入字符串“(max)”后的结果，如果存在多个最大的字母，就在每一个最大字母后面都插入"(max)"。
样例输入 abcdefgfedcba
样例输出 abcdefg(max)fedcba

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

__1_____{
    char mx='A';
    int __2_____
    for(int i=0;i<len;i++){
        if(str[i]>mx)3_____
    }
    return mx;
}
int main(){
    char mx_c;
    string str;
    int i,len;
    cin>>str;
    mx_c=__4_____
    for(__5_____){
        cout<<str[i];
        if(str[i]==mx_c)cout<<"(max)";
    }
    cout<<endl;
    return 0;
}
```

3 [1033] 置换加密法

置换加密算法是最简单的加密算法，其原理是将一个字母表中的字符替换成另一个字母表中的字符。这种形式的加密算法，已经存在 2000 多年的历史了。

输入输入文件中第一行是原文用的字母表，第 2 行是密文用的字母表。接下来有若干行，每一行是待加密的原文，每一行都不超过 64 个字符。

输出输出第 1 行是密文用的字母表，第 2 行是原文用的字母表。接下来的每一行是将输入文件中对应行加密后得到的密文字符串，原文字母表中没有的字符不用替换。

样例输入	样例输出
abcedfghijklmnopqrstuvwxyz zyxwvutsrqponmlkjihgfdecba Shar's Birthday: The birthday is October 6th,but the party will be Saturday, October 5. It's my 24th birthday and the first one in some	zyxwvutsrqponmlkjihgfdecba abcedfghijklmnopqrstuvwxyz Sszi'h Brigsvzb: Tsw yrigsvzb rh Oxglywi 6gs,yfg gsw kzigb eroo yw Szgfivzb, Oxglywi 5. Ig'h nb 24gs yrigsvzb zmv gsw urihg lmw rm hlnw

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

```
int main(){
    string str,a,b;
    getline(cin,a);//密码文
    getline(cin,b);//原文
    cout<<b<<endl<<a<<endl;
    while(getline(cin,str)){
        for(int i=0;i<str.length();i++){
          int j=judge(a,str[i]);//找到译文位置
          if(j!=-1)str[i]=b[j];
        }
        cout<<str<<endl;
    }
    return 0;
}
```

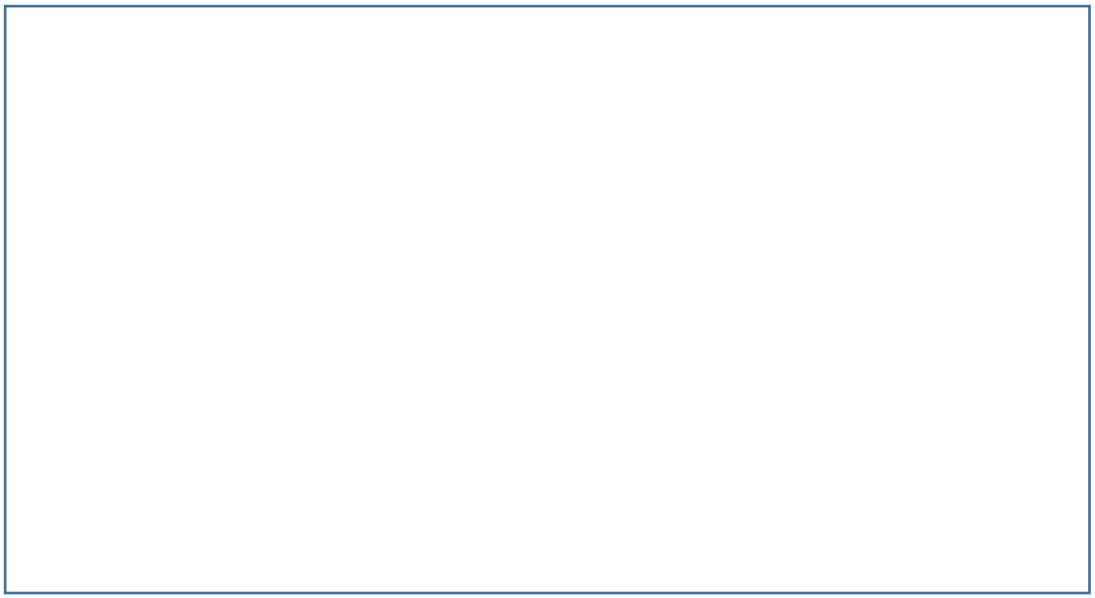
4 [2118] 邮件的烦恼

(看一下提示，注意输入)最近四正的邮箱里有许多未读邮件，但是他又没有那么多的时间一一阅读，现在他只想阅读与 acm 有关的邮件，所以请你帮忙从邮箱中找出题目里带有“acm”（包括 ACM 或者 acm。不包括 Acm,aCm...）的邮件，输出需要查看的邮件数量。
输入
多组测试数据，每组测试数据第一行输入一个整数 n，代表有 n 封邮件，接下来 n 行输入一行字符串，代表邮件的题目（长度不超过 200）。
输出
输出需要查看的邮件数量。
样例输入

```
2
acm TopCoder Special Round Match 600.5!
Amazon.cn
3
aa
Project Euler Email Validation System ACM
aacmm
```

样例输出

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



```
int main()
{
    int n,count=0;
    char str[100];
```

```
    while(cin>>n)
    {
        count=0;//getchar()会输入 cin>>n 后的回车符因为有多组数据，
        getchar();//getchar()吸收输入的回车符
        while(n--)
        {
            gets(str);
            if(getNum(str))
                count++;
        }
         printf("%d\n",count);
   }
   return 0;
}
```

5 [2700] 删除单词后缀

给定一个单词，如果该单词以 er、ly 或者 ing 后缀结尾， 则删除该后缀（题目保证删除后缀后的单词长度不为 0）， 否则不进行任何操作。

输入输入一行，包含一个单词（单词中间没有空格，每个单词最大长度为 32）。
输出输出按照题目要求处理后的单词。

样例输入 referer
样例输出 refer

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

int main()
{
    char a[205];
    cin>>a;
    int len = strlen(a);
    if (a[len - 1] == 'r'&&a[len - 2] == 'e')
        len-=2;
    else if (a[len - 1] == 'y'&&a[len - 2] == 'l')
        len -= 2;
    else if (a[len - 1] == 'g'&&a[len - 2] == 'n'&&a[len - 3] == 'i')
        len -= 3;
    for (int i = 0; i < len; i++)
        cout << a[i];
    cout << endl;
```

```
    return 0;
}
```

6 [2270] 最长单词

一个以'.'结尾的简单英文句子，单词之间用空格分隔，没有缩写形式和其它特殊形式

输入一个以'.'结尾的简单英文句子（长度不超过 80），单词之间用空格分隔，没有缩写形式和其它特殊形式

输出该句子中最长的单词（假设长度不超过 20）。如果多于一个，则输出第一个

样例输入 I am a student of Zhejiang University of Finance and Ecnomics.

样例输出 University

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    char maxs[20]="",word[20];
    int maxlen=0,len;
   do
   {
        cin>>word;
        len=strlen(word);
        if(len>maxlen)
        {
            strcpy(maxs,word);//字符串拷贝
            maxlen=strlen(word);
        }
   }while(word[len-1]!='.');
    cout<<maxs;
    return 0;
}
```

作业列表

[2257] 字符串反序　　[1900] 查找最大元素
[1033] 置换加密法　　[2118] 邮件的烦恼
[2700] 删除单词后缀　　[2270] 最长单词

课前练习

执行下面程序，正确的输出是（　　）。 #include<stdio.h> int x = 3, y = 8;	读程序写结果 fun1(int a,int b) { int c;

```
void swap (){
    int z;
    z = x;
    x = y;
    y = z;
    printf (" %d , %d \n", x, y);
}
int main() {
    int x = 5;
    int y = 7;
    swap ();
    printf (" %d , %d \n", x, y);
    return 0;
}
```

```
    a+=a; b+=b; c=fun2(a,b);
    return  c*c;
   }
fun2(int a,int b)
  { int c;
    c=a*b%3;
    return  c;
  }
main()
  { int x=11,y=19;
    printf("%d\n",fun1(x,y));
  }
```

结果是：_____

习题

1. 执行以下程序，打印输出的内容是：_____

```
#include<iostream>int x=5;
void incx( ){
    x++;
}
int main( ){
    int x=3;
    incx( );
    printf("%d\n", x);
    return 0;
}
```

2. 以下程序运行后的输出结果是（　　）。

```
int fun(int x, int y)
{
    if(x>y)
        return x;
    else
        return y;
}
int main(void)
{
    int x=3,y=8,z=6,r;
    r=fun(fun(x,y),2*z);
    printf("%d\n",r);
    return 0;
}
```

3. 下列程序的输出结果是_____

```
#include<iostream>
int fun3(int x) {
  static int a = 3;
  a += x;
  return (a);
}
int main()
{
  int k = 2, m = 1, n;
  n = fun3(k); n = fun3(m);
  printf("%d\n", n);
  return 0;
}
```

4. 以下程序的功能是计算 1~n 之间的所有素数之和。请填空。

```
#include <math.h>
double addprime (int) ;
isprime (int) ;
main ()
{
  int n=200;
  printf("1 至%d 之间的素数和为:%lf\n",n, addprime(__1_____));
}
double addprime( int n )
{
  int i;
  double s=0;
  for(  i=1;i<=n; i++)
  if( isprime(___2_____ ) )
  return s;
}
isprime (  int a)
{   int i;
  if( a==1 ) return 0;
  for( i=2; i<=sqrt(a); i++ )
  if(_____3_____) return 0;
  _____4_____;
}
```

5. 阅读程序

```
#include <iostream>
#include <string>
using namespace std;
int x;
void add(int z)
```

```
{
    cout << "z=" << z << endl;
    z = z + 10;
    cout << "z=" << z << endl;
}
int main()
{
    x = 5;
    cout << "x=" << x << endl;
    add(x);
    cout << "x=" << x << endl;
    return 0;
}
```

输出：_____

6. 阅读程序

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;
int j, k;
bool pr(int n)
{
    int i;
    bool t;
    t = true;
    i = 2;
    while (t && (i <= sqrt(n)))
    {
        if (n % i == 0)
            t = false;
        else
            i = i + 1;
    }
    return t;
}
int main()
{
    k = 0;
    j = 11;
    while (j <= 99)
    {
        if (pr(j) && pr(j + 2))
        {
            cout << j << " " << j + 2 << " ";
```

```
            k = k + 1;
        }
        j = j + 2;
    }
    cout << "total:" << k << endl;
    return 0;
}
```

输出：_____

7.
```
#include <iostream>
using namespace std;
int x,y,z;
void silly(int x,int y)
{
    x=7;
    y=17;
    z=18;
    cout<<x<<" "<<y<<" "<<z<<endl;
}
int main()
{
    x=1;
    y=2;
    z=3;
    silly(x, y);
    cout<<x<<" "<<y<<" "<<z<<endl;
    return 0;
}
```

输出：_____

8.辗转相除法求最大公约数

```
#include <iostream>
using namespace std;
int gys(int a,int b){
    if(_1_____ )
        return b;
    else
        return__2_____
}
int  main()
{
int a,b;
cin>>a>>b;
cout<<__3_____<<endl;}
```

第六讲 函数实践

1 [7186] 数字魔术游戏

在一种室内互动游戏中，魔术师要每位观众心里想一个三位数 abc（a、b、c 分别是百位、十位和个位数字），然后魔术师让观众心中记下 acb、bac、bca、cab、cba 5 个数以及这 5 个数的和值。只要观众说出这个和是多少，则魔术师一定能猜出观众心里想的原数 abc 是多少。例如，观众甲说他计算的和值是 1999，则魔术师立即说出他想的数是 443，而观众乙说他计算的和值是 1998，则魔术师说：“你算错了！”。请编程模拟这个数字魔术游戏。

输入 1 个数。

输出有两种情况。如果观众算对了就输出 1 个数，表示魔术师猜的观众心里的原数。如果观众算错了，就输出“The sum you calculated is wrong!”。

样例输入 1999

样例输出 443

分析：枚举求和

```
#include<bits/stdc++.h>
using namespace std;
int Magic(int m);
int main(){
    int sum,abc;
    scanf("%d",&sum);
    abc=Magic(sum);
    if(abc!=-1)
        printf("%d\n",abc);
    else printf("The sum you calculated is wrong!\n");
}
```

```
int Magic(int m){









  }
```

2 [2714] 绝对素数

如果一个自然数是素数,且它的数字位置经过对换后仍为素数,则称为绝对素数,例如 13。输入一个数 n， 输出 n（n<=10000）以内的所有绝对素数。

分析：定义两个函数，一个判断是否素数，一个求回文数。

```
#include<bits/stdc++.h>
using namespace std;
bool pr(int a)
{
    for(int j=2;j<=sqrt(a);j++)
    {
        if(a%j==0)
            return false;
    }
    return true;
}

int rev(int n){
```

```
 }
```

```
int main()
{
    int a,b,c,n;
    cin>>n;
    for(int i=11;i<=n;i+=2)
    {
        if(_____________________)
            cout<<i<<" ";
    }
    return 0;
}
```

3 [3338] 回文质数

因为 151 既是一个质数又是一个回文数(从左到右和从右到左是看一样的)，所以 151 是回

文质数。
写一个程序来找出范围[a,b](5 <= a < b <= 100,000 间的所有回文质数;
输入 第 1 行: 二个整数 a 和 b .
输出 输出一个回文质数的列表，一行一个。
样例输入 5 500
样例输出
5 7 11 101 131 151 181 191 313 353 373 383

4 [6859] 楼层编号

小林在 NOIP 比赛期间住在"新世界"酒店。和其他酒店不一样的是，这个酒店每天都有一个高能的数字 t，这个数字在楼层中是不会出现的，以 t=3 为例，则 3、13、31、33 等楼层是不存在的，楼层编号为 1，2，4，5，…所以实际上的 4 楼才是 3 楼。
已知小林预订了编号为 m 层的房间，并且当天高能数字是 t，现在他想知道房间所在的真实楼层是多少。
[输入格式]一行两个整数 m 和 t，1≤m≤100000，0≤t≤9，保证 m 对 t 合法。
[输出格式]一行一个整数，表示真实楼层。
[输入样例]14 3
[输出样例]12

问题分析

根据题意，只要从 1~m 穷举楼层编号，将所有含高能数字 t 的楼层计数存储在 ans 中，最后的答案就是 m-ans。

```
#include<bits/stdc++.h>
using namespace std;
int check(int x,int t)
{
    
}
int main()
{
        int m,t;cin>>m>>t;
        int ans=m;
        for(int i=1;i<=m;i++)
                if(_____________________) ans--;
        cout<<ans<<endl;
}
```

5 [1034]　校验和

校验和是一种算法，这种算法扫描数据包并返回一个值。这种算法的思想是当数据包被改变时，校验和同样要改变，因此校验和通常用来检查传输错误，用来确认传输内容的正确性。

在本题中，你需要实现一种校验和算法，称为 Quicksum。一个 Quicksum 数据包只允许包含大写字母和空格，并且起始字符和终止字符都是大写字母。除此之外，空格和大写字母允许以任何的组合方式出现，包括连续的空格。

校验和 Quicksum 是数据包中所有字符在**数据包中的位置和它的值的乘积的累加和**。空格的值为 0，其他大写字符的值为它在字母表中的位置，即 A=1,B=2，…，Z=26。

下面是 Quicksum 数据包“ACM”和“MID CENTRAL”的校验和计算方法；

ACM:1*1+2*3+3*13=46

MID CENTRAL:1*13+2*9+3*4+4*0+5*3+6*5+7*14+8*20+9*18+10*1+11*12=650

输入

输入文件中包括一个或多个数据包，输入文件的最后一行为符号#，表示输入文件的结束。每个数据包占一行，每个数据包不会以空格开头或结尾，每个

数据包包含 1~255 个字符。

输出

对每个数据包，输出一行，为它的校验和。

样例输入

ACM

MID CENTRAL

REGIONAL PROGWRAMMING CONTEST

#

样例输出

46

650

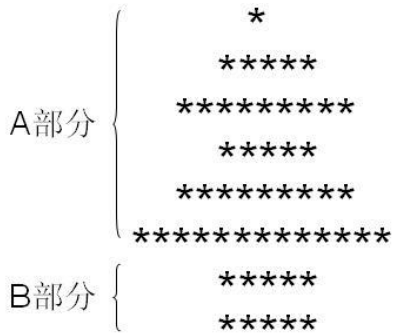
5183

```
#include<bits/stdc++.h>
using namespace std;
int quicksum(char a[]){



}
```

```
int main(){
    char a[256];
    while(gets(a)){
        if(a[0]=='#')break;
        printf("%d\n",quicksum(a));
    }
    return 0;
}
```

6 [7190]圣诞树

圣诞节要到了，不少商家在宣传板上绘制了圣诞树的图案，如图所示。



一棵圣诞树由 A 和 B 两部分组成：

A 是由 n（n≥10）个呈三角形的字符矩阵构成的，每个字符矩阵由三个参数 ai、bi、ci 唯一确定。ai 表示字符矩阵第一行字符的个数；bi 表示字符矩阵从第二行开始每一行与它上面那行的字符数之差均为 bi；ci 则表示字符矩阵的行数。

B 是一个 x 行 y 列的长方形，由 x 和 y 这两个参数唯一确定。

因为圣诞树是中轴对称的，所以根据所有的参数构成的圣诞树是唯一确定的。简单来讲，我们所说的一棵圣诞树就是像图那样的*矩阵，每一行的字符是指若干个连在一起的*。

【说明】

（1）输入数据保证圣诞树不会超出一页纸的范围。

（2）要求圣诞树是轴对称的，并且字符矩阵的第一列至少有一个非空格字符，即圣诞树尽量“顶格写”。在以上要求下，输出的圣诞树矩阵一定是唯一的（不考虑每行行末的空格）。

输入

输入数据分若干行。第一行是一个整数 n，表示 A 部分中字符矩阵的个数。以下 n 行，每行有三个正整数 ai、bi、ci（ai 为奇数，bi 为偶数）。

输入数据的最后一行，有两个正整数 x、y（y 是奇数），表示 B 部分的行数和列数。

输出

对于输入数据给定的圣诞参数，输出与之对应的圣诞树矩阵。

样例输入

3

1 4 3

5 4 3
5 4 4
2 5
样例输出

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

```
#include<bits/stdc++.h>
using namespace std;
char a[40];
int max=0;
int hao(int a,int b,int c)
{
    int i,j,m=0,first=1;
    for(i=1;i<=c;i++)
    {
        if(first==2)m=m+b;
        if(first==1){
            for(j=1;j<=max/2-a/2;j++)printf(" ");
            for(j=1;j<=a;j++)printf("*");
            printf("\n");
            first=0;
        }
        if(first==2){
            for(j=1;j<=max/2-(a+m)/2;j++)printf(" ");
            for(j=1;j<=a+m;j++)printf("*");
            printf("\n");
        }
        first=2;
    }
}
int yang(int d,int e)
{
    int i,j;
```

```
    for(i=1;i<=d;i++){
    for(j=1;j<=max/2-(e/2);j++)printf(" ");
    for(j=1;j<=e;j++)printf("*");
    printf("\n");
    }
}
int main()
{
    int n,ai,bi,ci,x,y,i=0,j;
    int count=1;
    scanf("%d",&n);
    while(count<=n&&scanf("%d%d%d",&ai,&bi,&ci)==3)
    {
        count++;
        a[i]=ai;a[i+1]=bi;a[i+2]=ci;i=i+3;
        if(ai+(ci-1)*bi>max)max=ai+(ci-1)*bi;
    }
    scanf("%d%d",&x,&y);
    for(j=0;j+2<=n*3;j=j+3) hao(a[j],a[j+1],a[j+2]);
    yang(x,y);
    return 0;
}
```

课堂作业列表

[2020] 哥德巴赫猜想　　[3340] 哥德巴赫猜想 2
[6859] 楼层编号　　[1034]　校验和
[6857] 抽奖　　 7187 金蝉素数

课前练习

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int n, m;
    cin >> n >> m;
    n = n % 7;
    m = m % 5;
    if (n > m)
        cout << n << endl;
    else
```

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int n, x, i, temp, j, count=0;
    cin >> n >> x;
    for (i = 1; i <= n; i++)
    {
        temp = i;
        while (temp > 0)
        {
```

```
        cout << m << endl;
    return 0;
}
```
输入：2014 2015
输出：_____

```
            j = temp % 10;
            temp = temp / 10;
            if (j == x)
                count++;
        }
    }
    cout << count << endl;
    return 0;
}
```
输入：100 5
输出：_____

习题

1. 以下程序运行后的输出结果是_____

```
main()
{   int x=11,y=6;
    swap (x,y);
    printf("x=%d,y=%d\n",x,y);
}
swap( int a,int b )
{  int t;
    t=a;   a=b   b=t;
}
```

2.以下程序运行后的输出结果是____.

```
#include<iostream>
void swap(int x, int y)
{
   int  t;
   t=x; x=y; y=t;
   return;
}
int main(void)
{
   int a=3, b=2;
   swap(a, b);
   printf("%d#%d#", a, b);
   return 0;
}
```

3. 将数组 array 中的 10 个元素按相反顺序存放。用函数实现。

```
#include<iostream>
 1                          //函数声明
int main( )
{
        int array[10], i;
        for( i=0; i<10; i++)    scanf( "%d", &array[i] );
       2
        for( i=0; i<10; i++)    printf( "%d ", array[i] );
        return 0;
}
void reverse( int a[ ], int n )
{
```

```
  int i, j,temp;
      for( i=0; i<n/2; i++ )
        {
                j = 3_____________
                temp = a[i];   a[i] = a[j];   a[j] = temp;       //交换 a[i]和 a[j]
        }
}
```

4. 以下程序的输出结果是_____。

```
#include<iostream>int func(int a, int b)
{
  return(a+b);
}
int main()
{
   int  x=2,y=5,z=8,r;
   r=func(func(x,y),z);
   printf("%d\n",r);
   return 0;
}
```

5. 以下程序的输出结果是：_____

```
#include<stdio.h>
void sub1(char a,char b)
{
    char c;
    c=a;
    a=b;
    b=c;
}
void sub2(char* a,char b)
{
    char c;
    c=*a;
   *a=b;b=c;}
void sub3(char* a,char*b)
{
    char c;
    c=*a;
    *a=*b;
    *b=c;
}
int main()
{
```

```
    char a,b;
    a='A';b='B';sub3(&a,&b);putchar(a);putchar(b);
    a='A';b='B';sub2(&a,b);putchar(a);putchar(b);
    a='A';b='B';sub1(a,b);putchar(a);putchar(b);
    return 0;
}
```

6. 下面程序运行后的输出结果是(　　)。

```
int x,y;
void Fun()
{
   int a=18, b=16;
   x=x+a+b;
   y=y+a-b;
}
int main()
{
   int a=9,b=8;
   x=a+b;
   y=a-b;
   Fun();
   printf("%d,%d", x, y);
   return 0;
}
```

7. 给出程序运行的结果：_____

```
int fn(int x,int y)
 {     int z;
       z=(x>y) ? x:y;
       return z;
 }
 int main()
 {   int a,b;
     int z;
     scanf("%d%d",&a,&b);
     z=fn(a,b);
     printf("%d\n",z);
     return 0;
 }
```

样例输入: 20 200 样例输出：_____

第七讲 递归入门

递归是一种重要的算法思想，递归特点是：函数调用它自己本身。其中直接调用自己称为直接递归，而将 A 调用 B，B 以调用 A 的递归叫做间接递归。

7.1 什么是递归

在数学上，求 n 的阶乘，有两种表示方法：

① n! = n×(n-1)×(n-2)×…×2×1

② n! = n×(n-1)!

这两种表示方法实际上对应到两种不同的算法思想。

第①种表示方法中，求 n! 要反复把 1、2、3、…、(n-2)、(n-1)、n 累乘起来，是**循环的思想，要用循环结构来实现**，代码如下：

```
int n, F=1;
scanf( "%d", &n );
for( i=1; i<=n; i++ )  F = F*i;
printf( "%d 的阶乘为%d", n, F );
```

第②种表示方法，求 n! 时需要用到(n-1)！。如果有一个函数 Factorial(int n)能实现求 n 的阶乘，则该函数在求 n!时要使用到表达式：n*Factorial(n-1)，Factorial(n-1)表示调用 Factorial()函数去求(n-1)!;而(n-1)!可以表示为(n-1)*(n-2)!,Factorial(n-2)表示调用 Factorial( )函数去求(n-2)！.....

从以上过程，可以得到：

1. **需要解决的问题 n!可以转化为多个子问题(n-i)!来求解，而这些子问题的求解方法与原问题完全相同，只是在数量规模上不同。**
2. **经过 n 次调用后，把原问题 n!转化为 1!，即调用的次数是有限的。**
3. **有结束递归的条件来终止递归，即当 n=1 时，1！=1。**

一般来说，能够用递归解决的问题应该满足以下三个条件：

- 需要解决的问题可以转化为一个或多个子问题来求解，而这些子问题的求解方法与原问题完全相同，只是在数量规模上不同。
- 递归调用的次数必须是有限的。
- 必须有结束递归的条件来终止递归。

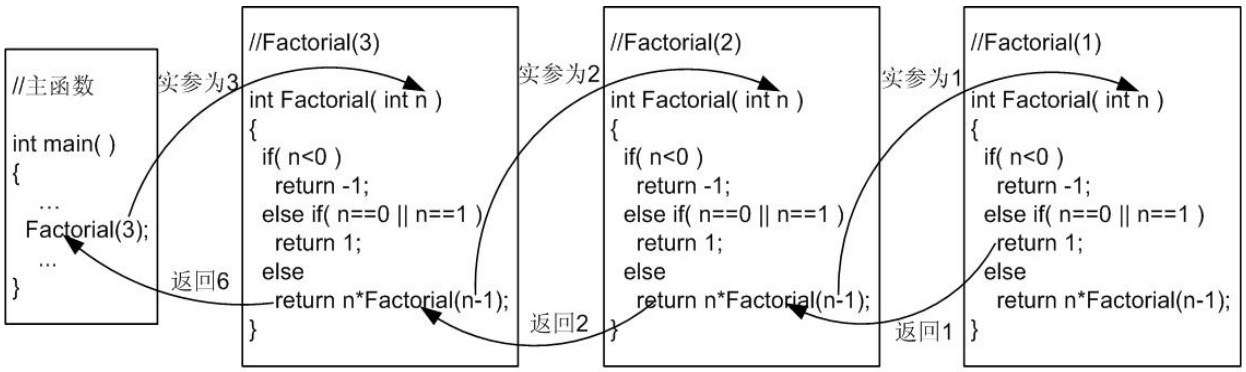
递归的缺点

递归解题相对常用的算法如普通循环等，运行效率较低。因此，应该尽量避免使用递归，除非没有更好的算法或者某种特定情况，递归更为适合的时候。在递归调用的过程当中系统为每一层的返回点、局部量等开辟了栈来存储，因此递归次数过多容易造成栈溢出。

1 [2459] 递归求阶乘

```
#include<iostream>
using namespace std;
int Factorial( int n )
{
    if( n<0 ) return -1;//结束递归的条件            ....    1
    else if( n==0 || n==1 ) return 1;              .....  2
    else return n*Factorial(n-1);   //递归调用 Factorial 函数     ....    3
}
int main( )
{
    int a;
    scanf( "%d", &a );
    printf( "%d!=%d\n", a, Factorial(a) );
    return 0;
}
```

具体执行过程如下：



假设要求 3!，其完整的执行过程如图所示，具体过程为：
①执行 main 函数的开头部分；
②当执行到 Factorial 函数调用“Factorial(3)”时，流程转而去执行 Factorial(3)函数，并将实参 3 传递给形参 n；
③执行 Factorial(3)函数的开头部分；
④当执行到递归调用 Factorial(n-1)函数时，此时 n-1=2，所以要转而去执行 Factorial(2)函数；
⑤执行 Factorial(2)函数的开头部分；
⑥当执行到递归调用 Factorial(n-1)函数时，此时 n-1=1，所以要转而去执行 Factorial(1)函数；
⑦执行 Factorial(1)函数，此时因为 n 的值为 1，所以返回 1，而不是再递归调用下去，即，Factorial(1)函数执行完毕，返回到上一层，即返回 Factorial(2)函数中；
⑧执行完 Factorial(2)函数的剩余语句，返回到 Factorial(3)函数中；
⑨执行完 Factorial(3)函数的剩余语句，返回到 main 函数中；

⑩继续执行 main 函数的剩余部分直到整个程序执行完毕。

思考：如果去掉上述代码第三行的 return，结果会有什么变化？

```
#include<iostream>
using namespace std;
int Factorial( int n )
{
    if( n<0 ) return -1;//结束递归的条件            ....    1
    else if( n==0 || n==1 ) return 1;              .....  2
    else  n*Factorial(n-1); //递归调用 Factorial 函数     ....    3
}
int main( )
{
    int a;
    scanf( "%d", &a );
    printf( "%d!=%d\n", a, Factorial(a) );
    return 0;
}
#include<iostream>
using namespace std;

int f1(int n){
    printf("f1 = (int n ) %d\n",n);
    if (n == 1) {
    return 1;
    }
    return n*f1(n-1); //每次返回是给 main 函数接受吗?
}

int main() {
    int result = f1(4);//为什么是 24,执行以次数怎么执行的?
    printf(" result = %d\n",result);
    return 0;
}
```

return 函数执行入栈出栈操作

练习：输入一下程序的运行结果：

```
#include <stdio.h>
void f(int n)
{
    if (n<1)
        return;
    else
```

```
    {
        printf("n=%d\n",n);
        f(n-1);
        printf("n=%d\n",n);
    }
}
int main(){
    f(5);
    return 0;
}
```

2 [2459]猴子吃桃

猴子第 1 天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了一个。第 2 天早上又将剩下的桃子吃掉一半，又多吃了一个。以后每天早上都吃了前一天剩下的一半另加一个。到第 10 天早上想再吃时，就只剩下一个桃子了。求第 1 天共摘了多少个桃子。

分析：

假设 Ai 为第 i 天吃完后剩下的桃子的个数,A0 表示第一天共摘下的桃子,本题要求的是 A0。有以下递推式子：

A0 = 2×(A1+1)　A1：第 1 天吃完后剩下的桃子数

A1 = 2×(A2+1)　A2：第 2 天吃完后剩下的桃子数

……

A8 = 2×(A9+1)　A9：第 9 天吃完后剩下的桃子数

A9 = 1

以上递推过程可分别用循环结构和递归函数实现。

用循环结构实现：

如果 x1，x2 表示前后两天吃完后剩下的桃子数，则有递推关系：x1 = (x2+1)*2。从第 9 天剩下 1 个桃子，反复递推 9 次，则可求第 1 天共摘下的桃子数。这里包含了反复的思想，可以用循环结构来实现，代码如下：

```
#include<iostream>
using namespace std;
int main( )
{
    int day, x1, x2;
    day = 9;
    x2 = 1;
    while( day>0 )
    {
        x1 = (x2+1)*2; // 第 1 天的桃子数是第 2 天桃子数加 1 后的 2 倍
        x2 = x1;
        day--;
    }
    printf( "total=%d\n", x1 );
    return 0;
```

```
}
```

用递归思想实现：

前面所述的递推关系也可以采用下面的方式描述。假设第 n 天吃完后剩下的桃子数为 A(n)，第 n+1 天吃完后剩下的桃子数为 A(n+1)，则存在的推关系：A(n) = (A(n+1) + 1) * 2。这种递推关系可以用递归函数实现，

1. 递归的递归式：_____
2. 递归的退出条件：_____
3. 初始状态：_____

代码如下：

```
#include<iostream>
using namespace std;
int fun(int n)
{
    if(n>=9) return 1;
    else return (2*(fun(n+1)+1));
}
int main( )
{
    printf( "total=%d\n", fun(0) );
    return 0;
}
```

7.2 案例分析

3 [2256] 斐波那契数列

采用递归思想递推 Fibonacci 数列中的每一项，并输出前 20 项的值。

1. 递归的递归式：_____
2. 递归的退出条件：_____

```
#include<iostream>
using namespace std;
int Fibonacci( int n )
{
    if( 1 ) return 1;
    else 2;
}
void main( )
{
    int i;
    int f[20] = { 0 };
```

```
    for( i=0; i<20; i++ )    //调用递归函数求每项
        f[i] = Fibonacci(i);
    for( i=0; i<20; i++ )
    {
        if( i%10==0 ) printf( "\n" );    //每行输出 10 个数据
        printf( "%6d", f[i] );  //每个数据占 6 个字符的宽度
    }
    printf( "\n" );
}
```

7.2 案例讲解

4 [2741] 最大公约数

分析：

数论中有一个求最大公约数的算法称为辗转相除法，又称欧几里德(Euclid)算法。其基本思想及执行过程为(设 m 为两正整数中较大者，n 为较小者)：

(1) 令 u = m，v = n；

(2) 取 u 对 v 的余数，即 r = u%v，如果 r 的值为 0，则此时 v 的值就是 m 和 n 的最大公约数，否则执行第(3)步；

(3) u = v，v = r，即 u 的值为 v 的值，而 v 的值为余数 r。并转向第(2)步。

例如，假设输入的两个正整数为 18 和 33，则 m = 33，n = 18。辗转相除法求最大公约数的过程如下。

```
int gcd( int u, int v ) //求 u 和 v 的最大公约数
{
    if( u%v==0 )  return v;
    else  return gcd(v, u%v);
}
```

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

```
int   main()
{
    int a,b;
    cin>>a>>b;
```

```
    cout<< _____  <<endl;
}
```

5 [2208]　求一个整数的各位数字之和

```
#include<bits/stdc++.h>
using namespace std;
int sum(int n){//12345
    int s;
    if(n==0)return 0;
    else
        s=n%10+sum(n/10);
     return s;
}
int main(){
      int n,s;
      cin>>n;
      cout<<sum(n);
      return 0;
}
```

6* [2764] 放苹果

把 M 个同样的苹果放在 N 个同样的盘子里，允许有的盘子空着不放，问共有多少种不同的分法？（用 K 表示）5，1，1 和 1，5，1 是同一种分法。

输入
第一行是测试数据的数目 t（0≤t≤20）。以下每行均包含二个整数 M 和 N，以空格分开。1≤M，N≤10。
输出
对输入的每组数据 M 和 N，用一行输出相应的 K。
样例输入
1
7 3
样例输出
8

这个问题的关键是递归函数。
m 个苹果放在 n 个盘子中，那么定义函数为 apple(m,n)：
1.m=0，没有苹果，那么只有一种放法，即 apple(0,n)=1
2.n=1，只有一个盘中，不论有或者无苹果，那么只有一种放法，apple(m,1)=1
3.n>m，和 m 个苹果放在 m 个盘子中是一样的，即 apple(m,n)=apple(m,m)
4.m>=n，这时分为两种情况，一是所有盘子都有苹果，二是不是所有盘子都有苹果。不

是所有盘子都有苹果和至少有一个盘子空着是一样的，即=apple(m,n-1)。所有盘子都有苹果，也就是至少每个盘子有一个苹果，m 个苹果中的 n 个放在 n 个盘子中，剩下的 m-n 个苹果，这和 m-n 个苹果放在 n 个盘子中是是一样的，即=apple(m-n, n)。这时，apple(m,n)=apple(m-n, n)+apple(m,n-1)。

```
#include <iostream>
using namespace std;
int apple(int m,int n)
{
  if(m==0 || n==1)
    return 1;
  else if(n > m)
    return apple(m, m);
  else
    return apple(m-n,n) + apple(m,n-1);
}
int main()
{
  int t;
  int m,n;
  cin>>t;
  while(t--)
  {
    cin>>m>>n;
    cout<<apple(m,n)<<endl;
  }
  return 0;
}
```

课堂作业列表

[2741] 最大公约数　　[2749] 汉诺塔问题
7329 元素求和　　2458 阶乘
2459 递推求猴子吃桃　　2719 斐波那契数列
2740 阶乘

课前练习

#include <iostream> using namespace std; int x,y,z; void silly(int x,int y) {	#include <iostream> using namespace std; int n; void change(int n) {

```
    x=7;
    y=17;
    z=18;
    cout<<x<<" "<<y<<" "<<z<<endl;
}
int main()
{
    x=1;
    y=2;
    z=3;
    silly(x, y);
    cout<<x<<" "<<y<<" "<<z<<endl;
    return 0;
}
```
输出：_____

```
    int i,j;
    if(n==0)
        return;
    i=n%8;
    j=n/8;
    change(j);
    cout<<i;
    return;
}
int main()
{
    cin>>n;
    change(n);
    return 0;
}
```
输入：2017
输出：_____

习题

1. 有以下程序：

```
int fun(int  x,int  y)
{   return  x+y; }
main( )
{   printf("%d\n",fun(fun(1,2),fun(3,4)));}
```

程序运行后的输出结果是(　　)

A、3　　B、7　　C、10　　D、编译错误

2. 对于以下递归函数 f，调用 f(4)，其返回值为(　　)

```
int f(int n)
{    if (n)  return f(n - 1) + n;
     else return n;
}
```

A) 10　　B) 4　　C) 0　　D) 以上均不是

3. 以下程序的运行结果是(　　)

```
fun(int k)
{ if(k>0) fun(k-1);
  printf("%d ",k);
}
int main()
{ int w=5;fun(w);printf("\n");}
```

A)5 4 3 2 l　　B)0 1 2 3 4 5　　C)1 2 3 4 5　　D)5 4 3 2 1 0

4. 有以下程序

```
#include<iostream>
```

```
int fun(int a,int b)
{ if(b==0)
return a;
else return(fun(--a,--b));
}
main()
{ printf("%d\n", fun(4,2));}
```

程序的运行结果是(　　)。

A)1　　B)2　　C)3　　D)4

5. 有以下程序

```
int fun(int n)
{
if(n==1) return 1;
else
return (n+fun(n-1));
}
main()
{
int x;
scanf(“%d”,&x); x=fun(x); printf(“%d\n”,x);
}
```

程序执行时，给变量 x 输入 10，程序的输出结果是(　　)

A)55　　B) 54　　C) 65　　D) 45

6. 阅读程序，选择程序的运行结果(　　)

```
int try(int);
int main()
{
   int x;
   x=try(5);
   printf(“%d\n”, x);
}
int try(int n)
{
     if(n>0)
          return(n*try(n-2));
     else
         return(1);
}
```

A) 15　　B)　120　　C) 1　　D)均错误

```
7.#include <iostream>
using namespace std;
char a[11];
void ty(int t)
```

```
{
	if (t == 10)
		cout << a[t];
	else
	{
		ty(t + 1);
		cout << a[t];
	}
}

int main()
{
	int i;
	for (i = 1; i <= 10; i++)
		cin >> a[i];
	ty(1);
	return 0;
}
```

输入：

123459876a

输出：

第八讲 递归理解

一般来说，能够用递归解决的问题应该满足以下三个条件：

- 需要解决的问题可以转化为一个或多个子问题来求解，而这些子问题的求解方法与原问题完全相同，只是在数量规模上不同。
- 递归调用的次数必须是有限的。
- 必须有结束递归的条件来终止递归。

要用到递归方法解决的问题主要可以分为三种情况：定义是递归的、数据结构是递归、问题求解是递归。

8.1 定义是递归的

有许多数学公式、数列等的定义是递归的。例如，求 n!和 Fibonacci 数列等。这些问题的求解过程可以将其递归定义直接转化为对应的递归算法。

1[7716] 递归的简单使用

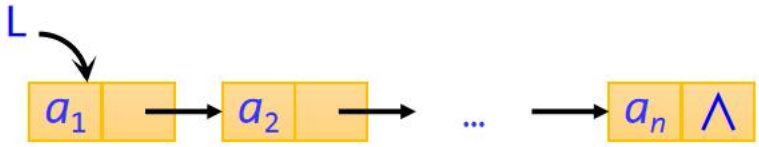
利用递归函数实现求 0.5，0.5*1.5，$0.5*1.5^2$，$0.5*1.5^3$，...，$0.5*1.5^n$，中的某一项。
输入要求的项数。
输出该项数的结果。
样例输入 5

```
#include<iostream>
using namespace std;
double an(int n){
  if(n==1){
     return 0.5;
  }
  else{
    return an(n-1)*1.5;
  }
}
int main(){
  int n;
  cin>>n;
  cout<<an(n);
  return 0;
}
```

8.2 数据结构是递归的

有些数据结构是递归的。例如单链表就是一种递归数据结构，或则数组。我们可以把数

组理解为一个线性表，一个元素一个一个挨着，如 a1 在 a2 的左边，a3 在 a2 的右边。



对于递归数据结构，采用递归的方法编写算法既方便又有效。例如，求数组(单链表)L 的所有数组原则之和的递归算法如下：

```
int Sum(int a[],int n)
{   if (n==0)
        return 0;
    else
        return(a[n]+Sum(a,n-1));
}
```

填空：

```
int Sum(int a[],int n)
{
      if (n==1)
         1_____________________
      else
         2_____________________;
}
```

2 [7329] 元素求和

输出 n 个整数，求和。
输入 n 和 n 个数
输出和

分析：假设 sum(n)表示求 n 个元素的和，采用递归的方式理解，就是 sum(n)=sum(n-1)+a[n]，即把问题转化为对 n-1 个元素求和后，再加上第 n 个元素。

1. 递归的递归式：_____
2. 递归的退出条件：_____
3. 初始状态：_____

#include<bits/stdc++.h>
using namespace std;
 	
int sum(int a[],int n,int s){
  if(n==0)return s;
  else
	#include<bits/stdc++.h>
	using namespace std;
	
	int sum(int a[],int n){
	if(n==1)
	  return a[1];
		#include<iostream>
		using namespace std;
		
		int sum(int a[],int n){
		  int s;
		  if(n==0)

sum(a,n-1,s+a[n]);
}
int main(){
int n,s=0,a[105];
cin>>n;
for(inti=1;i<=n;i++)
cin>>a[i];
cout<<sum(a,n,s);
return 0;
}
	return sum(a,n-1)+a[n];
	}
	int main(){
	int n,a[105];
	cin>>n;
	for(int i=1;i<=n;i++)
	cin>>a[i];
	cout<<sum(a,n);
	return 0;
	}
		return 0;
		else
		s=a[n]+sum(a,n-1);
		}
		int main() {
		int n,s,a[105];
		cin>>n;
		for(int
		i=1;i<=n;i++)
		cin>>a[i];
		cout<<sum(a,n);
		return 0;
		}

3 [1053] 查找最大最小值

分析：假设 fmax(n)表示求 n 个元素的和最大值，采用递归的方式理解，就是

fmax(n)=_____

1. 递归的递归式：_____
2. 递归的退出条件：_____
3. 初始状态：_____

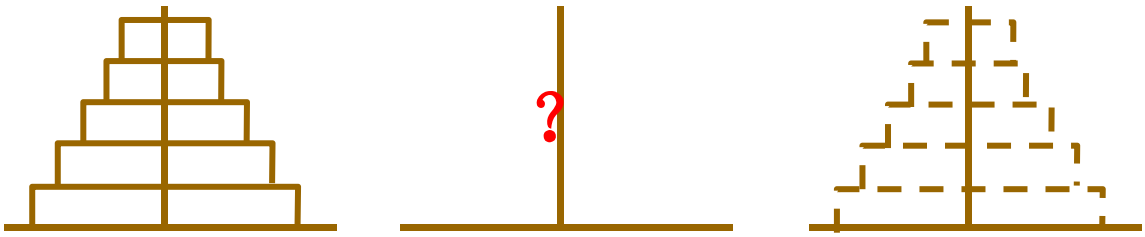
```
#include<bits/stdc++.h>
using namespace std;
double fmax(double a[],int i)
{    if (1_____)
       return a[1];
     else
        2_____;
}

double fmin(double a[],int i)
{    if (i==1)
        return  3_____
     else
        return fmin(fmin(a,i-1),a[i]);
}

int main(){
     int i,n;
     double a[105];
     cin>>n;
     for(int i=1;i<=n;i++)cin>>a[i];
        printf("%.2lf %.2lf",  4_____);
```

```
    return 0;
}
```

8.3 问题求解是递归

有些问题的解法是递归的，典型的有 Hanoi 问题求解。

盘片移动时必须遵守以下规则：每次只能移动一个盘片；盘片可以插在 X、Y 和 Z 中任一塔座；任何时候都不能将一个较大的盘片放在较小的盘片上。



如果盘子数量很少，通过心算便可以直接想出移动盘子的具体步骤。

比如：只有 2 个盘子的情况，此问题就变得相当的简单，只需要 3 步就可以完成整个移动操作。

A—>B（表示将 A 柱 最上面的 1 个盘子移到 B 柱 上）

A—>C（将 A 此时最上面的 1 个盘子移到 C 上）

B—>C（将 B 上的盘子移到 C 上）

假设现在有 64 个盘子，显然是不可能心算的，我们采用递归方式解决。

递归的结束条件：

最后 1 个人（第 64 个人）只需要移动 1 个盘子。

注意：

第 1 个人完成任务的前提是：第 2 个人完成了任务

第 2 个人完成任务的前提是：第 3 个人完成了任务

……

第 63 个人完成任务的前提是：第 64 个人完成了任务

所以：

只有当第 64 个人完成任务后，第 63 个人才能完成任务；只有第 2~64 个人完成任务后，第 1 个人才能完成任务！

典型的递归问题！

将 n 个盘子从 A 移到 C 可以分解为以下 3 个步骤：

将 A 上 n-1 个盘子借助 C 先移到 B 上；

把 A 上剩下的 1 个盘移到 C 上；

将 n-1 个盘从 B 借助 A 移到 C 上。

上面第 1 步和第 3 步，都是把 n-1 个盘子从 1 个座移到另 1 个座上，采用的办法是相同的，只是座的名字不同而已。为使之一般化，可以将第 1 步和第 3 步表示为：

将 one 座上 n-1 个盘子移到 two 座（借助 three 座）。

只是在第 1 步和第 3 步中，one、two、three 和 A、B、C 的对应关系不同。

对第 1 步，对应关系是：one—A，two—B，three—C 。
对第 3 步，对应关系是：one—B，two—C，three—A 。
因此，将上面 3 个步骤分为 2 类操作：
将 n-1 个盘子从 1 个座移到另 1 个座上 （当 n>1 时）；
将最后 1 个盘子从 1 个座上移到另 1 个座上 。

设计程序
分别用 2 个函数实现以上 2 类操作：
设计 hanoi 函数实现第 1 类操作；
函数 hanoi(n, one, two, three) 将实现把 n 个盘子从 one 座借助 two 座移到 three 座的过程；
设计 move 函数实现第 2 类操作；
函数 move(x, y) 将实现把 1 个盘子从 x 座移到 y 座的过程。x 和 y 是代表 A、B、C 座之一，根据每次不同情况分别以 A、B、C 代入。

4[2749] 汉诺塔问题

约 19 世纪末，在欧州的商店中出售一种智力玩具，在一块铜板上有三根杆，最左边的杆上自上而下、由小到大顺序串着由 64 个圆盘构成的塔。目的是将最左边杆上的盘全部移到中间的杆上，条件是一次只能移动一个盘，且不允许大盘放在小盘的上面。
这是一个著名的问题，几乎所有的教材上都有这个问题。由于条件是一次只能移动一个盘，且不允许大盘放在小盘上面，所以 64 个盘的移动次数是：18,446,744,073,709,551,615
这是一个天文数字，若每一微秒可能计算(并不输出)一次移动，那么也需要几乎一百万年。我们仅能找出问题的解决方法并解决较小 N 值时的汉诺塔，但很难用计算机解决 64 层的汉诺塔。假定圆盘从小到大编号为 1, 2, ...

输入为一个整数(小于 20）后面跟三个单字符字符串。整数为盘子的数目，后三个字符表示三个杆子的编号。

输出每一步移动盘子的记录。一次移动一行。
每次移动的记录为例如 a->3->b 的形式，即把编号为 3 的盘子从 a 杆移至 b 杆。
样例输入
2 a b c
样例输出
a->1->c
a->2->b
c->1->b

```
#include <iostream>
using namespace std;
int hannoTa(int n,char one ,char two,char three){
        if(n==1)
                cout<<one<<"->"<<n<<"->"<<two<<endl;
        else
        {
                1
```

```
                cout<<one<<"->"<<n<<"->"<<two<<endl;
                2
        }
}
int main( )
{
        int n;
        char a,b,c;
        cin>>n>>a>>b>>c;
        hannoTa(n, a, b, c);
        return 0;
}
```

8.4 案例分析

5 [2225] 选择排序

思考：选择排序是每次选择一个最大的或者最小的元素和当前第一个元素做交换，那如何转变成递归呢？

1. 递归的递归式：_____
2. 递归的退出条件：_____
3. 初始状态：_____

```
#include<bits/stdc++.h>
using namespace std;
int a[10];
void SelectSort(int a[],int n,int i)
{    int j,k;
     if (i==n-1) return;   //满足递归出口条件
     else
     {
          k=i;//k 记录 a[i..n-1]中最小元素的下标
          for (j=i+1;j<n;j++) //在 a[i..n-1]中找最小元素
               if (a[j]>a[k])k=j;
          if (k!=i)          //若最小元素不是 a[i]
               swap(a[i],a[k]);//a[i]和 a[k]交换
          SelectSort(a,n,i+1);
     }
}

int main(){
     int n;
```

```
    cin>>n;
    for(int i=0;i<n;i++)cin>>a[i];
    SelectSort(a,n,0);
    for(int i=0;i<n;i++)cout<<a[i]<<"   ";
    return 0;
}
```

6 冒泡排序

1. 递归的递归式：_____
2. 递归的退出条件：_____
3. 初始状态：_____

```
#include<bits/stdc++.h>
using namespace std;
void BubbleSort(int a[],int n,int i)
{   int   j;
    bool exchange;
    if (i==n-1) return;   //满足递归出口条件
    else
    {
        exchange=false;     //置 exchange 为 false
        for (j=n-1;j>i;j--)
            if (a[j]<a[j-1])//当相邻元素反序时
            {
                swap(a[j],a[j-1]);
                exchange=true;      //发生交换置 exchange 为 true
            }
        if (exchange==false)//未发生交换时直接返回
            return;
        else //发生交换时继续递归调用
            BubbleSort(a,n,i+1);
    }
}
int main(){
    int a[11];
    for(int i=0;i<10;i++)cin>>a[i];
    BubbleSort(a,10,0);
    for(int i=0;i<10;i++)cout<<a[i]<<" ";
    return 0;
}
```

7 [2867] 集合的划分

设 S 是一个具有 n 个元素的集合，S＝〈 a1，a2，……，an〉 ，现将 S 划分成 k 个满足下列条件的子集合 S1，S2，……，且满足：

1. Si ≠ ∅

2. Si ∩ Sj ＝ ∅ (1≤i，j≤k i≠j)

3. S1 ∪ S2 ∪ S3 ∪ … ∪ Sk ＝ S

则称 S1，S2，……，是集合 S 的一个划分。它相当于把 S 集合中的 n 个元素 a1，a2，……，an 放入 k 个(0＜k≤n＜30)无标号的盒子中,使得没有一个盒子为空。请你确定 n 个元素 a1，a2，……，放入 k 个无标号盒子中去的划分数 S(n,k)。

输入给出 n 和 k。

输出 n 个元素 a1，a2，……，ana1，a2，……，an 放入 k 个无标号盒子中去的划分数 S(n,k)。

样例输入 10 6

样例输出 22827

先举个例子，设 S＝｛1，2，3，4｝，k＝3，不难得出 S 有 6 种不同的划分方案，即划分数 S(4，3)=6，具体方案为：

{1，2} ∪ {3} ∪ {4}

{1，3} ∪ {2} ∪ {4}

{1，4} ∪ {2} ∪ {3}

{2，3} ∪ {1} ∪ {4}

{2，4} ∪ {1} ∪ {3}

{3，4} ∪ {1} ∪ {2}

考虑一般情况，对于任意的含有 n 个元素 a1 ，a2，……，an 的集合 S，放入 k 个无标号的盒子中去，划分数为 S(n，k)，我们很难凭直觉和经验计算划分数和枚举划分的所有方案，必须归纳出问题的本质。

其实对于任一个元素 an，则必然出现以下两种情况：

1、｛an｝是 k 个子集中的一个，于是我们只要把 a1，a2，……，an-1 划分为 k－1 子集，便解决了本题，这种情况下的划分数共有

S(n－1，k－1)个；

2、｛an｝不是 k 个子集中的一个，则 an 必与其它的元素构成一个子集。则问题相当于先把 a1，a2，……，an-1 划分成 k 个子集，这种情况下划分数共有 S(n－1，k)个；然后再把元素 an 加入到 k 个子集中的任一个中去，共有 k 种加入方式，这样对于 an 的每一种加入方式，都可以使集合划分为 k 个子集，因此根据乘法原理，划分数共有

k ＊ S(n－1，k)个。

综合上述两种情况，应用加法原理，得出 n 个元素的集合｛a1，a2，……，an｝划分为 k 个子集的划分数为以下递归公式：

S(n，k)＝S(n－1，k－1) ＋ k ＊ S(n－1，k) (n>k，k>0)。

下面，我们来确定 S(n，k)的边界条件.

首先不能把 n 个元素不放进任何一个集合中去，即 k=0 时，S(n，k)＝0；也不可能在不允许空盒的情况下把 n 个元素放进多于 n 的 k 个集合中去，即 k＞n 时,S(n,k)＝0；再者，把 n 个元素放进一个集合或把 n 个元素放进 n 个集合，方案数显然都是 1，即 k=1 或 k=n 时，S(n,k)=1。

因此，我们**可以得出划分数 S(n，k)的递归关系式**为：

S(n，k)＝S(n－1，k－1) ＋ k ＊ S(n－1，k)　　(n>k，k>0)　　S(n，k)＝0　(n<k)或(k＝0)

S(n，k)＝1　　(k=1)或(k＝n)

```
#include<stdio.h>
int s(int n,int k)
{
  if((n<k)||(k==0))
    return 0;
  else if((k==1)||(k==n))
    return 1;
  return s(n-1,k-1)+k*s(n-1,k);
}
int main()
{
  int n,k;
  scanf("%d %d",&n,&k);
  printf("%d",s(n,k));
  return 0;
}
```

课前练习

阅读程序，选择程序的运行结果 _____

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

int try1(int n)
{
   if(n>0)
       return(n*try1(n-2));
   else
      return(1);
}
int main()
{
  int x;
  x=try1(5);
  printf("%d\n", x);
}
```

下列递归程序的输出结果为 _____

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

int fib(int g)
{   switch(g){
        case 0: return 0;
        case 1:
        case 2: return 2;
   }
    printf("g=%d,", g);
    return fib(g-1) + fib(g-2);
}
int main(void)
{
    int k;
    k = fib(4);
    printf("k=%ld\n", k);
```

	return 0; }

习题

1. 有以下程序：

```
int fun(int  x,int  y)
{   return  x+y; }
main( )
{   printf("%d\n",fun(fun(1,2),fun(3,4)));}
```

程序运行后的输出结果是_____。

A、3　　B、7　　C、10　　D、编译错误

2.输出 1 到 100 之间的所有完数。要求定义和调用函数 is(n)判断 n 是否为完数，若 n 为完数则返回 1，否则返回 0。完数就是因子和与它本身相等的数，6 是完数（6=1+2+3），1 不是完数。

```
#include<iostream>
int is(int n);
int main()
{   int i;
    for(i = 1; i <= 100; i++)
        if( is(i) )
            printf("%d ", i);
    return 0;
}
int is(int n)
{   int i, sum;
    if(n==1) return 0;
    ____1_______
    for(i = 1; i <= n/2; i++)
        if(_____2_______)
            sum = sum + i;
    if(_________3_______)
    return 1;
    else
    return 0;
}
```

```
3 #include <iostream>
using namespace std;
int n;
void change(int n)
{
    int i,j;
    if(n==0)
        return;
```

```
    i=n%8;
    j=n/8;
    change(j);
    cout<<i;
    return;
}
int main()
{
    cin>>n;
    change(n);
    return 0;
}
```

输入：2017

输出：_____

4.用递归法求阶乘

```
#include <iostream>
using namespace std;
int Sx(int n){
1_____
  else
return __2_____________
}
int main(){
        int n;
        cin>>n;
        cout<<  3_____________
        return 0;
}
```

5.输入 10,运行结果是_____

```
int f(int x){
        if(x == 1){
                return 1;
        }else if(x == 2){
                return 3;
        }else{
                return  x + f(x - 1);
        }
}
int main(){
        int n;
        cin>>n;
  cout<<f(n);
      return 0;
}
```

6.

```
#include <iostream>
using namespace std;
int f(int n)
{
    if (n == 0)
    {
        return 1;
    }
    else if (n < 0)
```

```
    {
        return f(n + 1) + 3;
    }
    else
    {
        return f(n - 1) - 2;
    }
}

int main()
{
    cout << f(f(2)) << endl;
    return 0;
}
```

输出：_____

7.

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;
int x, s;
int d(int x)
{
    if (x == 1)
        return 1;
    else
        return d(x - 2) + x;
}
int main()
{
    x = 9;
    s = d(x);
    cout << s << endl;
    return 0;
}
```

输出：_____

8.

```
#include <iostream>
using namespace std;
int n;
int fun(int n)
{
    if(n==1)
```

```
        return 1;
    else if(n==2)
        return 2;
    else if(n==3)
        return 3;
    else
        return fun(n-3)*fun(n-1);
}
int main()
{
    cin>>n;
    cout<<fun(n)<<endl;
    return 0;
}
```

输入：

7

输出：_____

第九讲　结构体

在存储和处理大批量数据时，一般会使用数组来实现，但是每一个数据的类型及含义必须一样。如果需要把不同类型、不同含义的数据当作一个整体来处理，如 1000 个学生的姓名、性别、年龄、体重、成绩等，怎么处理呢？C++ 提供了结构体（struct）来解决这类问题。

9.1 结构体定义

将不同种类型的数据有序地组合在一起，**构造出一个新的数据类型**，这种形式称为结构体。结构体是多种类型组合的数据类型。结构体类型只是一种数据类型，不占内存空间，只有定义结构体类型变量时才开辟内存空间。

使用结构体，必须要**先声明一个结构体类型，再定义和使用结构体变量**。结构体类型的声明格式如下：

```
struct 类型名{
 数据类型 1    成员名 1;
 数据类型 2    成员名 2;
 …
};
```

如：

```
struct   student
  {      int  num;
         char  name[20];    不同数据类型组成的成员
         char  sex;
         char  addr[30];
   };
```

定义结构体后，再定义变量名：

```
struct   student
{    int  num;
     char  name[20];
     char  sex;
     int  age;
     float  score;
     char  addr[30];
};
struct  student    student1,  student2;
```

也可以在定义类型的同时定义变量

```
struct   student
{    int  num;
     char  name[20];
     char  sex;
```

```
        int  age;
        float  score;
        char  addr[30];
    } student1, student2;
```

练习：

定义一只名叫“wangcai”的小狗	定义一个名叫“bigboss”的游戏怪兽	定义一位同学,名字自己定义

9.2 变量的引用

对于结构体类型变量的引用要注意以下几点：

1、 不能对结构体变量整体赋值或输出，**只能分别对各个成员引用**。

```
cin>>student1.num;      student1.num=100;
```

2、同类型的结构体变量之间可以直接赋值。这种赋值等同于各个成员的依次赋值。

3、结构体变量不能直接进行输入输出，它的每一个成员能否直接进行输入输出，取决于其成员的类型，若是基本类型或是字符数组，则可以直接输入输出。

4、结构体变量可以作为函数的参数，函数也可以返回结构体的值。当函数的形参与实参为结构体类型的变量时，这种结合方式属于值调用方式，即属于值传递。（举例说明）

5、结构体类型的变量在内存依照其成员的顺序顺序排列，所占内存空间的大小是其全体成员所占空间的总和。

6、在编译时，仅对变量分配空间，不对类型分配空间。

7、对结构体中各个成员可以单独引用、赋值，其作用与变量等同。

9.3 结构体数组

结构体数组中的每个元素都是一个结构体类型的变量，其中包括该类型的各个成员。数组各元素在内存中连续存放。

一、结构体数组的定义

```
struct   student
   {    int  num;
        char  name[20];
        char  sex;
        int  age;
        float  score;
```

```
        char  addr[30];
} ;
```

struct student stu[30];

或者定义结构体时直接定义数组：

```
struct   student
   {    int  num;
        char  name[20];
        char  sex;
        int  age;
        float  score;
        char  addr[30];
} stu[30];
```

二、结构体数组的初始化

```
struct   student
   {    int  num;
        char  name[20];
        char  sex;
} stu[3]={ {1011, "Li Lin",'M'}, {1012,"Wang Lan",'F'},
                 {1013,"Liu Fang",'F'};
```

练习：

定义 10 只小狗	定义 100 个学生

1 [1472] 打印学生记录

现有有 N 个学生的数据记录，每个记录包括学号、姓名、三科成绩。 编写一个函数 input,用来输入一个学生的数据记录。 编写一个函数 print,打印一个学生的数据记录。 在主函数调用这两个函数，读取 N 条记录输入，再按要求输出。 N<100

输入:学生数量 N 占一行 每个学生的学号、姓名、三科成绩占一行，空格分开。

输出:每个学生的学号、姓名、三科成绩占一行，逗号分开。

样例输入

```
2
a100 zhblue 70 80 90
b200 newsclan 90 85 75
```

样例输出

```
a100,zhblue,70,80,90
```

b200,newsclan,90,85,75

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

struct student{
   char xh[20];
   char name[20];
   int yw,sx,yy;
};

int main(){
   student stu[100];
   int n;
   cin>>n;
   for(int i=1;i<=n;i++){
      cin>>stu[i].xh>>stu[i].name>>stu[i].yw>>stu[i].sx>>stu[i].yy;
      printf("%s,%s,%d,%d,%d\n",stu[i].xh,stu[i].name,stu[i].yw,stu[i].sx,stu[i].yy);
   }
   return 0;
}
```

9.5 案例讲解

2 [7714] 结构体简单使用

输入某个学生的信息（姓名、年龄，五门功课成绩），计算平均成绩并输出。
输入学生名字，年龄，五门课的成绩。
输出名字，年龄，五门课成绩，平均分。
样例输入
张三 20 78 92 83 75 69
样例输出
张三 20
78 92 83 75 69 平均分为 79.4

分析：
1. 定义一个学生结构体
2. 输入相应的信息
3. 计算平均值

```
#include<iostream>
using namespace std;
struct student
{
```

```
    char name[10];
    int age;
    float score[5],ave;
};
 int main()
{
  1                          // 定义结构变量 stu
  int i;
  stu.ave=0;    // 存放平均成绩成员 ave 赋 0
  2                          // 输入学生的姓名及年龄
  for(i=0;i<5;i++)
  {
     cin>>stu.score[i];       // 输入学生的五门功课成绩
     3
  }
  cout<<stu.name<<" "<<stu.age<<endl;  //输出学生信息
  for(i=0;i<5;i++){
     cout<<stu.score[i]<<" ";
  }
  cout<<"平均分为"<<stu.ave;
  return 0;
 }
```

3 [1471] 结构体

题目描述
定义一个结构体变量（包括年、月、日）。计算该日在本年中是第几天，注意闰年问题。
输入年月日
输出当年第几天
样例输入 2000 12 31
样例输出 366

```
#include<bits/stdc++.h>
using namespace std;
```

```
int lun_nian(int year){
        if(year%4==0&&year%100!=0||year%400==0)
                return 1;
        else
```

```
                return 0;
}
int main()
{
        1                       //2000 12 31
        int sum=0,i;
        int mon[12]={31,28,31,30,31,30,31,31,30,31,30,31};
        cin>> 2
        for(i=0;i<d1.month-1;i++)
                sum=sum+mon[i];
        sum=sum+d1.day;
        if( 3                              )
                sum=sum+1;
        cout<<sum<<endl;
        return 0;
}
```

4 [2707] 最高分数的学生姓名

题目描述
输入学生的人数，然后再输入每位学生的分数和姓名，求获得最高分数的学生的姓名。
输入:第一行输入一个正整数 N（N ≤ 100），表示学生人数。接着输入 N 行，每行格式如下：分数 姓名
　　分数是一个非负整数，且小于等于 100；姓名为一个连续的字符串，中间没有空格，长度不超过 20。数据保证最高分只有一位同学。
输出：获得最高分数同学的姓名。

样例输入
5
87 lilei
99 hanmeimei
97 lily
96 lucy
77 jim
样例输出
Hanmeimei

```
#include<bits/stdc++.h>
using namespace std;
struct student{
        int mark;
        char name[20];
};
```

```
int main()
{
        int n,maxx=0;
        cin>>n;
        __1_____________________________
        for(int i=0;i<n;i++){
              __2_____________________
                if(p.mark >maxx)   __3_____________________
        }
        cout<<max.name ;
        return 0;
}
```

5 [1473]　各门课的成绩

有 N 个学生，每个学生的数据包括学号、姓名、3 门课的成绩，从键盘输入 N 个学生的数据，要求打印出 3 门课的总平均成绩，以及最高分的学生的数据（包括学号、姓名、3 门课成绩）
输入:学生数量 N 占一行每个学生的学号、姓名、三科成绩占一行，空格分开。
输出:各门课的平均成绩 最高分的学生的数据（包括学号、姓名、3 门课成绩）
样例输入
2
1 blue 90 80 70
b clan 80 70 60
样例输出
85 75 65
1 blue 90 80 70

```
#include<stdio.h>
struct student{
    char num[20];
    char name[20];
    int s1;
    int s2;
    int s3;
};
int main(){
    int n,c=0,flag,m;
    double t,l=0,w=0,y=0;
    struct student stu[100];
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
scanf("%s %s %d %d %d",stu[i].num,stu[i].name,&stu[i].s1,&stu[i].s2,&stu[i].s3);
        l=l+stu[i].s1;
```

```
        w=w+stu[i].s2;
        y=y+stu[i].s3;
        t=stu[i].s1+stu[i].s2+stu[i].s3;
        if(t>c)c=t,m=i;
    }
    printf("%.0f %.0f %.0f\n",l/n,w/n,y/n);
printf("%s %s %d %d %d",stu[m].num,stu[m].name,stu[m].s1,stu[m].s2,stu[m].s3);
    return 0;
}
```

6 [6874] 校园十佳歌手 2

杭州联合小学“声动杭州”校园十佳歌手大赛即将拉开序幕。选手们都在认真、积极地准备着。选手计分是以歌手大赛的形式进行的，以总分减去一个最高分再减去一个最低分，算平均分。程序首先需算出每位选手的平均分，保存起来，再将这些保存好的数据进行排序。在程序输入时，输入参赛学生的人数，学生的姓名，评委个数（0<sent<=10000），每个评委评的分数，（0<=score<=100），中间用空格隔开以歌手大赛的算分方式算分。保存每个学生名字对应的分数，按照从大到小的顺序排序这些分数。再将结果按顺序输出，并输出名次。

输入第一行为 m n,参赛学生人数和评委个数

后面的 n 行是学生信息

姓名 评委分数

输出

从高到低输出学生信息，选手得分保留小数点 2 位。

名次 姓名 分数

样例输入	样例输出
3 5 李明 89 90 98 96 97 王三 87 86 80 98 77 陈辰 98 78 97 96 95	1 陈辰 96.00 2 李明 94.33 3 王三 84.33

思路：

```
#include<iostream>
using namespace std;
struct singer{
  char name[20];
  double score;//歌手得分
}stu[5500];

int main()
{
  int n,m;
  double s[20];
```

```
    cin>>n>>m;//n 个同学参赛  m 个评委
    for(int i=1;i<=n;i++)
    {
       double sum=0,max=0,min=100;
       cin>>stu[i].name;
       for(int j=1;j<=m;j++) {
          cin>>s[j];//m 个评委的得分
          sum=sum+s[j];
          if(s[j]<min)min=s[j];
          if(s[j]>max)max=s[j];
       }
       stu[i].score=(sum-max-min)/(m-2);
    }

    for(int i=1;i<n;i++)
    {
       for(int j=1;j<=n-i;j++)
          if(stu[j].score<stu[j+1].score)
             swap(stu[j],stu[j+1]);
    }
    for(int i=1;i<=n;i++)
    {
        printf("%d %s %.2lf",i, stu[i].name, stu[i].score);
     }
    return 0;
}
```

课堂作业列表

[1471] 结构体
[1952] NO.1
[2742]分数线划定
1472 打印一个学生的数据记录
1473 各门课的成绩

课前练习

include <iostream>
using namespace std;
int main()
{
    int sum=0,maxn,i;
	#include <iostream>
	using namespace std;
	int main()
	{
	    int i,j,n;

```
    cin>>maxn;
    for(i=1; i<=maxn; i++)
    {
        if(i%2!=0)
            sum=sum+i;
    }
    cout<<sum<<endl;
    return 0;
}
```
输入：200
输出：_____

```
    int b[11];
    n=2016;
    j=0;
    while(n>0)
    {
        j=j+1;
        b[j]=n%3;
        n=n/3;
    }
    for(i=j; i>=1; i--)
        cout<<b[i];
    cout<<endl;
    return 0;
}
```
输出：_____

习题

1 下面结构体说明正确的是(　　)

A)
```
struct  st
  {
      int x;
      double  y;
  }
```

B)
```
struct  st
  {
      int x;
      double  y;
  }  f1,f2;
```

C)
```
struct  st
  {
      int x;
      double  y;
  }
struct  st f1, f2;
```

D)
```
struct  st
  {
     int x;
     double  y;
  };
struct  f2, f2;
```

2. 有如下定义：
```
struct  date
{   int month;
      int day;
      int year;
};
struct worker
{    char name[20];
    char sex;
    struct date birthday;
}person1;
```
对变量 person1 的出生年份进行赋值时，下面正确的赋值语句是(　　)

A. year=1966;　　B. birthday.year=1966;

C. person1.year=1966;　　D. person1.birthday.year=1966;

3. 写出下列程序的运行结果(　　)

```
 #include<iostream>
struct abc { int a, b, c; };
    main()
    { struct abc s[2]={{1,2,3},{4,5,6}};
      int t;
      t=s[0].a+s[1].b;
      printf("%d \n",t);
    }
```

4. 写出下列程序的运行结果。

```
#include<bits/stdc++.h>
using namespace std;
struct stu{
    string name;
    string sex;
    int year,month;
};
const int MAXN = 110;
stu a[MAXN];
int main(){
    int n;
    cin >> n;
    for(int i = 1; i <= n; i++)
          cin >> a[i].name >> a[i].sex >> a[i].year >> a[i].month;
    for(int i = 1; i <= n; i++)
      for(int j = i+1; j <= n; j++)
      if(a[i].year < a[j].year || a[i].year == a[j].year && a[i].month < a[j].month)
            swap(a[i],a[j]);
    for(int i = 1; i <= n; i++){
          cout<< a[i].name << " " << a[i].sex <<  " " ;
          cout<< a[i].year <<  " " << a[i].month << endl;
    }
    return 0;
}
```

输入：　　输入出：_____

```
5
John male 1999 12
David female 1999 8
Jason male 1998 11
Jack female 1998 8
Kitty female 2000 7
```

5.输入学生的人数，然后再输入每位学生的分数和姓名，求获得最高分数的学生的姓名。

输入:第一行输入一个正整数 N（N ≤ 100），表示学生人数。接着输入 N 行，每行格式如下：

分数 姓名; 分数是一个非负整数，且小于等于 100；姓名为一个连续的字符串，中间没有空格，长度不超过 20。数据保证最高分只有一位同学。

输出：获得最高分数同学的姓名。

```
#include<iostream>
#include<string>
#include<algorithm>
using namespace std;
struct _1___________
{
    int x;
    ___2___________
}s[111];
int cmp(__3_____________)
{
   __4_________________;
}
int main()
{
    int n;
    while (cin>>n)
    {
        for (int i = 0; i < n; i++)
            cin >> s[i].x >> s[i].name;
        sort(s, s + n, cmp);
        cout << s[0].name << endl;
    }
    return 0;
}
```

样例输入

5

87 lilei

99 hanmeimei

97 lily

96 lucy

77 jim

样例输出

Hanmeimei

第十讲 结构体排序

10.1 结构体排序

结构体排序是指对结构体中的某一个成员的大小关系排序。例如对上述的 student 数组 stu 中的元素学号成员的大小关系从大到小的顺序(升序)排序。compare 函数可编写成：

```
Student Stu[100];
bool cmp2(Student a,Student b)
{
    return a.id>b.id;//按照学号降序排列
    //return a.id<b.id;//按照学号升序排列
}
sort(Stu,Stu+100,cmp2);
```

10.2 案例分析

1 [1473] 各门课的成绩

有 N 个学生，每个学生的数据包括学号、姓名、3 门课的成绩，从键盘输入 N 个学生的数据，要求打印出 3 门课的总平均成绩，以及最高分的学生的数据（包括学号、姓名、3 门课成绩）
输入:学生数量 N 占一行每个学生的学号、姓名、三科成绩占一行，空格分开。
输出:各门课的平均成绩 最高分的学生的数据（包括学号、姓名、3 门课成绩）
样例输入
2
1 blue 90 80 70
b clan 80 70 60
样例输出
85 75 65
1 blue 90 80 70

```
#include<bits/stdc++.h>
using namespace std;
struct student {
        char xh[20];
        char xm[20];
        int m1,m2,m3;
}stu[100];

int cmp( student s1,student s2){
```

```
        return s1.m1+s1.m2+s1.m3>s2.m1+s2.m2+s2.m3;
}
int main(){
        int n,i,j,s1=0,s2=0,s3=0;
        cin>>n;
        for(i=1;i<=n;i++)
                cin>>stu[i].xh>>stu[i].xm>>stu[i].m1>>stu[i].m2>>stu[i].m3;

        for(i=1;i<=n;i++)
        {
                s1+=stu[i].m1;
                s2+=stu[i].m2;
                s3+=stu[i].m3;

        }
        s1=s1/n;s2=s2/n;s3=s3/n;
        sort(stu+1,stu+n+1,cmp);
        cout<<s1<<" "<<s2<<" "<<s3<<endl;
         cout<<stu[1].xh<<"  "<<stu[1].xm<<"  "<<  stu[1].m1<<"  "<<stu[1].m2<<"
        "<<stu[1].m3<<endl;
        return 0;
}
```

2 [1952] NO.1

所谓NO.1,就是所有成绩都排在第一的同学，我们假设每个人只有理科，文科，体育这三门课。我们现在假设某门成绩并列第一，并列的人都是这门功课第一名，并且保证数据不会出现2个NO.1。
现给定n个人的信息，输出第一名的名字。
输入
多组数据，输入文件第一行为一个整数T，代表测试数据数。 (T<50)
接下来T个测试数据。
每个测试数据的的第一行为一个整数n(n<=100)，接下来有n行，
每行的格式如下：
名字 理科成绩 文科成绩 体育成绩 (数值越高代表成绩越好).
名字长度不超过20，3个成绩的为正整型。
输出
对于每个测试数据，输出NO.1的名字，如果不存在第一名，
就输出"NO NO.1".

```
3
2
lvhao 2 2 2
xiaoshua 1 1 1
2
lvhao 4 4 4
xiaoshua 4 4 3
3
lvhao 3 4 5
xiaoshua 1 3 1
pan 4 1 5
样例输出
lvhao
lvhao
NO NO.1
```

分析：
1. 定义一个学生结构体，包括学生的姓名，三门课。
2. 输入t,循环处理t组数据。
3. 输入每组数据的 理科成绩 文科成绩 体育成绩，求出单课

成绩的最大值。

4. 扫描每个学生的成绩，看看是否有每门课都等于最大成绩的学生，如果有做输入并记录状态。

```
#include<iostream>
using namespace std;
struct Stu {
        char name[20];
        int l,w,t;
}s[100];

int main(){
        int max1,max2,max3;
        int i,j;
        int t,n;
        cin>>t;
        for(i=1;i<=t;i++){//t 组数据
                __1__
                cin>>n;//每组数据的学生人数
                for(j=0;j<n;j++)//输入每个学生的分数和姓名
                __2__
                for(j=0;j<n;j++){//求解这组数据中的每门课的最大值
                        if(s[j].l>max1)max1=s[j].l;
                        if(s[j].w>max2)max2=s[j].w;
                        if(s[j].t>max3)max3=s[j].t;
                }
                __3__;//记录是否存在 NO1 的状态变量
                for(j=0;j<n;j++){
                        if( __4__){
                                cout<<s[j].name<<endl;
                                flag=0;
                                break;
                        }
                }
                if(__5__ )cout<<"NO NO.1"<<endl;
        }
        return 0;
}
```

3 [2742]分数线划定

题目描述

世博会志愿者的选拔工作正在 A 市如火如荼的进行。为了选拔最合适的人才，A 市对所有报名的选手进行了笔试，笔试分数达到面试分数线的选手方可进入面试。面试分数线根据计

划录取人数的150%划定，即如果计划录取m名志愿者，则面试分数线为排名第m*150%（向下取整)名的选手的分数,而最终进入面试的选手为笔试成绩不低于面试分数线的所有选手。现在就请你编写程序划定面试分数线，并输出所有进入面试的选手的报名号和笔试成绩。

输入

第一行，两个整数n，m（5 ≤ n ≤ 5000，3 ≤ m ≤ n），中间用一个空格隔开，其中n表示报名参加笔试的选手总数，m表示计划录取的志愿者人数。输入数据保证m*150%向下取整后小于等于n。

第二行到第 n+1 行,每行包括两个整数,中间用一个空格隔开,分别是选手的报名号k（1000 ≤ k ≤ 9999）和该选手的笔试成绩s（1 ≤ s ≤ 100）。数据保证选手的报名号各不相同。

输出

第一行，有两个整数，用一个空格隔开，第一个整数表示面试分数线；第二个整数为进入面试的选手的实际人数。

从第二行开始，每行包含两个整数，中间用一个空格隔开，分别表示进入面试的选手的报名号和笔试成绩，按照笔试成绩从高到低输出，如果成绩相同，则按报名号由小到大的顺序输出。

样例输入
6 3
1000 90
3239 88
2390 95
7231 84
1005 95
1001 88
样例输出
88 5
1005 95
2390 95
1000 90
1001 88
3239 88

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

```
int main(){
        int m,n,num,i,j;
        int line;
        cin>>n>>m;
        num=int(m*1.5);
        for(i=1;i<=n;i++){
                cin>>a[i].grade>>a[i].score;
        }
        for(i=1;i<n-1;i++){
                for(j=1;j<=n-i;j++){
                        if( 1                              )
                            swap(a[j],a[j+1]);
                        else if(a[j].score==a[j+1].score ){
                            if( 2                          )
                                swap(a[j],a[j+1]);
                        }

                }
        }
        line=a[num].score ;
        num=0;
```

```
        for(i=1;i<=n;i++){
                if( 3                        )
                num++;
        }
        cout<<line<<" "<<num<<endl;
        for(i=1;i<=num;i++){
                  cout<<a[i].grade<<" "<<a[i].score<<endl;
        }
return 0;
}
```

写法 2：

```
#include <iostream>
#include <string.h>
#include<iostream>#include <algorithm>
using namespace std;

int n,m;
struct node{
        int num;
        int score;
}a[5050];

bool cmp(node a,node b){
```

```
}

int main()
{
        while(scanf("%d%d",&n,&m)!=EOF){
                int men = m * 1.5;
                for(int i=0;i<n;i++) scanf("%d%d", &a[i].num, &a[i].score);
                 1
                int score = a[men-1].score;
                for(int i=0;i<n;i++){
                        if(a[i].score < score) break;
                        men = i+1;
                }
                printf("%d %d\n", a[men-1].score, men);
                for(int i=0;i<n;i++){
                        if(a[i].score < score) break;
```

```
                        printf("%d %d\n", a[i].num, a[i].score);
                }
        }
        return 0;
}
```

4 [1118] 奖学金

某校的惯例是在每学期的期末考试之后发放奖学金。发放的奖学金共有五种，获取的条件各自不同： 1) 院士奖学金，每人 8000 元，期末平均成绩高于 80 分（> 80），并且在本学期内发表 1 篇或 1 篇以上论文的学生均可获得； 2) 五四奖学金，每人 4000 元，期末平均成绩高于 85 分（> 85），并且班级评议成绩高于 80 分（> 80）的学生均可获得； 3) 成绩优秀奖，每人 2000 元，期末平均成绩高于 90 分（> 90）的学生均可获得； 4) 西部奖学金，每人 1000 元，期末平均成绩高于 85 分（> 85）的西部省份学生均可获得； 5) 班级贡献奖，每人 850 元，班级评议成绩高于 80 分（> 80）的学生干部均可获得； 只要符合条件就可以得奖，每项奖学金的获奖人数没有限制，每名学生也可以同时获得多项奖学金。例如姚林的期末平均成绩是 87 分，班级评议成绩 82 分，同时他还是一位学生干部，那么他可以同时获得五四奖学金和班级贡献奖，奖金总数是 4850 元。 现在给出若干学生的相关数据，请计算哪些同学获得的奖金总数最高（**假设总有同学能满足获得奖学金的条件**）。

输入

输入的第一行是一个整数 N（1 <= N <= 100），表示学生的总数。接下来的 N 行每行是一位学生的数据，从左向右依次是姓名，期末平均成绩，班级评议成绩，是否是学生干部，是否是西部省份学生，以及发表的论文数。姓名是由大小写英文字母组成的长度不超过 20 的字符串（不含空格）；期末平均成绩和班级评议成绩都是 0 到 100 之间的整数（包括 0 和 100）；是否是学生干部和是否是西部省份学生分别用一个字符表示，Y 表示是，N 表示不是；发表的论文数是 0 到 10 的整数（包括 0 和 10）。每两个相邻数据项之间用一个空格分隔。

输出

输出包括三行，第一行是获得最多奖金的学生的姓名，第二行是这名学生获得的奖金总数。如果有两位或两位以上的学生获得的奖金最多，输出他们之中在输入文件中出现最早的学生的姓名。第三行是这 N 个学生获得的奖学金的总数。

样例输入

```
4
YaoLin 87 82 Y N 0
ChenRuiyi 88 78 N Y 1
LiXin 92 88 N N 0
ZhangQin 83 87 Y N 1
```

样例输出

```
ChenRuiyi
9000
28700
```

```
#include<bits/stdc++.h>
using namespace std;
```

```
struct student{
    int qm,py,lw,jxj;
    char xm[25],gb,xb;
}stu[105];
int main()
{
    int i,n,sum=0,maxn=-100,s;
    cin>>n;
    for(i=1;i<=n;i++)
        cin>>stu[i].xm>>stu[i].qm>>stu[i].py>>stu[i].gb>>stu[i].xb>>stu[i].lw;
    for(i=1;i<=n;i++){
        if(stu[i].qm>80&&stu[i].lw>=1) stu[i].jxj+=8000;
        if(stu[i].qm>85&&stu[i].py>80) stu[i].jxj+=4000;
        if(stu[i].qm>90) stu[i].jxj+=2000;
        if(stu[i].qm>85&&stu[i].xb=='Y') stu[i].jxj+=1000;
        if(stu[i].py>80&&stu[i].gb=='Y') stu[i].jxj+=850;
    }
    for(i=1;i<=n;i++){
        sum=sum+stu[i].jxj;
        if(stu[i].jxj>maxn){
            maxn=stu[i].jxj;
            s=i;
        }
    }
    cout<<stu[s].xm<<endl;
    cout<<maxn<<endl;
    cout<<sum;
    return 0;
}
```

课堂作业列表

[1471] 结构体
[1952] NO.1
[2742]分数线划定
1472 打印一个学生的数据记录
1473 各门课的成绩

课前练习

1. 阅读程序

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

```
int main()
{
    int s, x;
    x = 0; s = 0;
    while (s < 55)
    {
        x = x + 1;
        s = s + x;
    }
    cout << x << endl;
    return 0;
}
```

输出：_____

2. 阅读程序

```
#include <iostream>
using namespace std;
int main()
{
    int i, j, s, h, v, n;
    int a[21][21];
    cin >> n >> h >> v;
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= n; j++)
        {
            cin >> a[i][j];
        }
    }
    s = 0;
    for (i = 1; i <= n; i++)
    {
        if (i == h)
        {
            for (j = 1; j <= n; j++)
                s = s + a[i][j];
        }
    }
    for (j = 1; j <= n; j++)
    {
        if (j == v)
        {
            for (i = 1; i <= n; i++)
                s = s + a[i][j];
```

```
        }
    }

    if (h <= v)
    {
        for (i = 1; i <= (n - (v - h)); i++)
            s = s + a[i][i + v - h];
    }
    else
    {
        for (j = 1; j <= (n - (h - v)); j++)
            s = s + a[j + h - v][j];
    }
    for (i = 1; i <= 2; i++)
        s = s - a[h][v];
    cout << s;
    return 0;
}
```

输入：

```
8 5 3
2 16 18 5 13 13 14 0
3 15 19 14 12 16 5 11
9 1 5 6 1 14 7 5
1 2 6 5 2 12 4 8
3 13 10 1 10 1 12 18
1 5 0 1 4 6 18 0
19 15 7 4 0 2 12 13
8 15 17 0 2 11 16 16
```

输出：_____

习题

1.已知表示学生信息的结构体如下，对结构体成员的赋值方式中正确的是(　　)

```
struct student
{     int num;
      char name[10];
      double english;
      double math;
      double computer;
}std;
```

A）student.name="Liu";　　B)std.name="Lee";

B）student.math=84;　　D)std.english=90;

2.有以下程序：

```
struct cargo
{     int num;
      char name[12];
      double price;
};
main( )
{    int i;
    struct                                                                cargo
cargo[4]={{1,"DVD",15.5},{2,"CD",12.0},{3,"VCD",13.8},{4,"BLANK",5.6}};
    for(i=0;i<=2;i=i+2)    printf("%s\n",cargo[i],name);
}
```

程序运行后的输出结果是(　　)

A)DVD　　B)CD　　C)DVD　　D)DVD
　CD　　VCD　　VCD　　BLANK

3.有以下程序:

```
struct{
    int a; int b;
 } a[3]={1,2,3,4,5,6};
main( )
{
    printf("%d\n",a[0].b*a[2].a/a[1].a*a[0].b);
}
```

程序运行后的输出结果是(　　)

A)6　　B)4　　C)1　　D2

第十一讲 模拟初步

现实中有些问题难以找到公式或规律来求解，只能按照一定的步骤不停的"模拟"下去，最后才能得到答案。这样的问题，用计算机来求解十分合适，只要能让计算机模拟人在解决此问题时的行为即可。这种求解问题的思想，可以称为"模拟"。

11.1 模拟的基本思路

例 1 **[2948]醉酒的狱卒**(The Drunk Jailer)

某个监狱有一排、共 n 间牢房，一间挨一间。每间牢房关着一名囚犯，每间牢房的门刚开始时都是关着的。有一天晚上，狱卒厌烦了看守工作，决定玩一个游戏。游戏的第 1 轮，他喝了一杯酒，然后沿着监狱，把所有牢房的门挨个挨个打开；游戏的第 2 轮，他又喝了一杯酒，然后沿着监狱，把编号为偶数的牢房的门关着；游戏的第 3 轮，他又喝了一杯酒，然后沿着监狱，对编号为 3 的倍数的牢房，如果牢房的门开着，则关上，否则打开；…，狱卒重复游戏 n 轮。游戏结束后，他喝下最后一杯酒，然后醉倒了。
这时，囚犯才意识到他们牢房的门可能是开着的，而且狱卒醉倒了，所以他们越狱了。
给定牢房的数目，求越狱囚犯的人数。

分析：

在本题中，n 轮游戏过后，哪些牢房的门是开着的，并无规律可循。但这个游戏的规则和过程都很简单：游戏有 n 轮，第 j 轮游戏是将编号为 j 的倍数的牢房状态变反，j = 1, 2, 3, …, n。这些规则和过程用程序能较容易地实现，所以适合采用"模拟"的思路求解。
具体实现时可以定义一个一维数组，表示每个牢房的状态，初始为 1，表示 Locked。如图 5.1 所示。模拟 n 轮游戏过程：第 j 轮时，改变牢房号为 j 的倍数的牢房的状态，为 0 改为 1，为 1 改为 0。最后统计状态为 0 的牢房数。

```
#include<iostream>
using namespace std;
int main(){
        int a[101],m,n,i,j,k,sum=0;
        cin>>m;//有 m 个监狱
        for(i=1;i<=m;i++){
                sum=0;
                cin>>n;//该监狱的牢房个数
                for(j=1;j<=n;j++)a[j]=1;//初始化牢房门关着
                for(j=1;j<=n;j++){//n 轮操作
                        for(k=1;k<=n;k++){//对 n 个牢房操作
                                if(k%j==0)a[k]=!a[k];
                        }
                }
                for(j=1;j<=n;j++)
                        if(a[j]==0)sum++;
                cout<<sum<<endl;
```

```
        }
        return 0;
}
```

11.2 案例讲解

1 [3204]模拟约瑟夫环

出列游戏：n 个人围成一圈，从第 1 个人开始报数，报数报到 m 的人出列；然后又从下一个人开始从 1 开始报数；重复 n-1 轮游戏，每轮游戏淘汰 1 个人，最后剩下的人就是胜利者。模拟该游戏，输出依次出列的位置及最后的胜利者。

开始报数位置：s=1
游戏人数：n=8
间隔：m=4
游戏规则：从起始位置开始，第 m 个位置上的数依次出列，循环直至只剩下一个数。
有 8 个人参加报数，每间隔 4 个位置淘汰 1 个出列。
有 7 轮，每轮淘汰一个位置出列，在算法中用 r 来代表第几轮；
用一维数组来存储 8 个位置上的 8 个号码，数组长度为 9，为符合人们的习惯，只使用 a[1]~a[8]。
用变量 i 来指向剩余的位置(要跳过已经出列的位置)
用变量 j 来累加间隔位置，达到 4 后，对应位置要出列，然后又从 1 开始计数。因此要对 4 进行取模运算。本来应该是 j=(j+1)%m，但是(j+1)%m 的范围是 0~3，我们希望 j 取到 1~4，所以改成 j=j%m+1。其中 m=4。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a[22],r,i,j,n,m;
    cin>>n>>m;
    for(i=1;i<=n;i++)a[i]=i;
    for(r=1,i=1,j=1; 1                          )
    {
        while(a[i]=  2          ){//跳过已经出列的位置
            i=  3          
        }
        if(  4                  )//判断是否数到 m 个数
        {
            cout<<a[i]<<" ";//输出当前的数
            5                   //更新变量的值
        }
    }
    return 0;
```

}

2 [6869]网络堵塞

你肯定经历过很多人同时使用网络，网络变得很慢很慢。为了彻底解决这个问题，Ulm 大学决定采取突发事件处理方案：在网络负荷高峰期，将公平地、系统地切断某些城市的网络连接。德国的城市被随机地标上 1~n 的序号。比如金华的序号为 1，杭州的序号为 2，温州的序号为 3 等等，这些序号顺序纯粹是随机的。然后随机地选择一个数 m。首先切断第 1 个城市的网络连接，然后间隔 m 个序号，切断对应的城市，如果超出范围，则取模，并且忽略已经被切断网络连接的城市。例如，如果 n=17，m=5，被切断网络连接的城市依次为：[1,6,11,16,5,12,2,9,17,10,4,15,14,3,8,13,7]。

本题的目的是，希望最后被切断网络连接的城市是杭州。对于给定的 n 值, m 的值必须很仔细地选择，使得 2 号城市(即杭州的序号)是最后被选中切断网络连接的城市。m 值应该如何选？

你的任务是编写程序，读入 n 的值，求 m 的值，使得 Ulm 最后被选择切断网络连接。

分析：

这道题也是模拟约瑟夫环问题，只不过不是求最后的胜利者，而是给定 n，要使得最后的胜利者为 2，求 m。与上题不同的是，这里的约瑟夫环问题首先淘汰的是编号为 1 的城市，然后是编号为 m+1 的城市。

本题的思路是借用上例的方法，定义函数 Joseph 实现 m 和 n 取任意值的约瑟夫环问题。在主函数中读入城市个数，从 1 开始枚举 m，直到某个 m 能使得该约瑟夫环问题的最后胜利者为 2 号城市，这个 m 值就是题目要求的结果。

```
#include<bits/stdc++.h>
using namespace std;
int joseph(int n,int m)
{
    int a[99999];
    for(int i=1;i<=n;i++)a[i]=i;
    a[1]=0;//第 1 个城市首先被淘汰
    for(int r=1,i=1,j=1;r<=n;i=i%n+1,j=j%n+1)
    {
        while(a[i]==0){i=i%n+1;}
        if(j%m==0)
        {
            a[i]=0, j=0,r++;
            if(i==2){//如果将被切断的城市是 2 号城市，提前结束
                if(r==n) return 1;
                return 0;//n 轮游戏后，2 号城市还没被切断
            }
        }
    }
}
int main(){
```

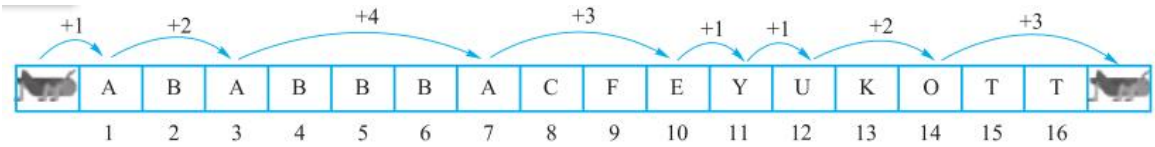
```
    int n,m;
    cin>>n;
    for(m=1;m<=n;m++)
        if(joseph(n,m)==1)break;
    cout<<m;
    return 0;
}
```

3 [6865]蚱蜢

有一天，一只蚱蜢像往常一样在草地上愉快地跳跃，它发现了一条写满了英文字母的纸带。

蚱蜢只能在元音字母(A、E、I、O、U、Y)间跳跃，一次跳跃所需的能力是两个位置的差。纸带所需的能力值为蚱蜢从纸带开头的前一个位置根据规则跳到纸带结尾的后一个位置的过程中能力的最大值。

蚱蜢想知道跳跃纸带所需的能力值(最小)是多少。如图 9.3-1 所示的纸带所需的能力值(最小)是 4。



[输入格式]一行一个字符串，字符串长不超过 100。

[输出格式]一行一个整数，代表(最小)能力值。

[输入样例]

KMLPTGFHNBVCDRFGHNMBVXWSQFDCVBNHTJKLPMNFVCKMLPTGFHNBVCDRFGHNMBVXWSQFDCVBNHTJKLPMNFVC

[输出样例]

85

[问题分析]

从头到尾枚举纸带的每一个字母，按照规则模拟蚱蜢在元音字母之间跳跃的过程，打擂台记录能力值。

```
 #include<bits/stdc++.h>
using namespace std;
char str[101];
int main()
{
    scanf("%s",str+1);//下标从 1 开始
    int n=strlen(str+1);
    int ans=0,x=0;
    for(int i=1;i<=n;i++)
    {
        if(str[i]=='A'||str[i]=='E'||str[i]=='O'||str[i]=='U'||str[i]=='I'||str[i]=='Y')
        {
```

```
                ans=_____
                x=i;
            }
        }
        cout<<ans<<endl;
}
```

4 [6867]保龄球

打保龄球是用一个滚球去打击十个站立的柱，将柱击倒。一局分十轮，每轮可滚球一次或多次，以击倒的柱数为依据计分。一局得分为十轮得分之和，而每轮的得分不仅与本轮滚球情况有关，还可能与后续一两轮的滚球情况有关。即某轮某次滚球击倒的柱数不仅要计入本轮得分，还可能会计入前一两轮得分。具体的滚球击柱规则和计分方法如下：

(1) 若某一轮的第一次滚球就击倒全部十个柱，则本轮不再滚球(若是第十轮则还需另加两次滚球，不妨称其为第十一轮和第十二轮，并不是所有的情况都需要滚第十一轮和第十二轮球)。该轮得分为本次击倒柱数 10 与以后两次滚球所击倒柱数之和。

(2) 若某一轮的第一次滚球未击倒十个柱，则可对剩下未倒的柱再滚球一次。如果这两次滚球击倒全部十个柱，则本轮不再滚球(若是第十轮则还需另加一次滚球)，该轮得分为这两次共击倒柱数 10 与以后一次滚球所击倒柱数之和。

(3) 若某一轮两次滚球未击倒全部十个柱，则本轮不再继续滚球，该轮得分为这两次滚球击倒的柱数之和。总之，若某一轮中一次滚球或两次滚球击倒十个柱，则本轮得分是本轮首次滚球开始的连续三次滚球击倒柱数之和(其中有一次或两次不是本轮滚球)。若一轮内二次滚球击倒柱数不足十个，则本轮得分即为这两次击倒柱数之和。下面以实例说明如下：

轮	1	2	3	4	5	6	7	8	9	10	11	12
击球情况	/	/	/	72	9/	81	8/	/	9/	/	8/	
各轮得分	30	27	19	9	18	9	20	20	20	20		
累计总分	30	57	76	85	103	112	132	152	172	192		

现在请编写一个保龄球计分程序，用来计算并输出最后的总得分。

[输入格式]

输入一行，为前若干轮滚球的情况，每轮滚球用一到两个字符表示，每一个字符表示一次击球，字符"/"表示击倒当前球道上的全部的柱，否则用一个数字字符表示本次滚球击倒的当前球道上的柱的数目，两轮滚球之间用一个空格隔开。

[输出格式]

输出一行一个整数，代表最后的得分。

[输入样例 1]/ / / 72 9/ 81 8/ / 9/ / 8/

[输出样例 1]192

[输入样例 2]90 90 / 9/ 81 / / / 72 / /0

[输出样例 2]170

[输入样例 3]/ / / 72 9/ 81 8/ / 9/ 15

[输出样例 3]169

[问题分析]

本题的难点在于每一次、每一轮滚球的分数不能立刻算出，可能依赖于后面 1~2 轮，所以需要多次积分。

```
 #include <iostream>
#include <string>
using namespace std;
#define SIZE 210
string s[SIZE];
int hit[SIZE][2], score[SIZE];

int main(void)
{
	int n = 1, i, res = 0;
	 while (cin >> s[n])n++;//计算多少轮
	n--;
	for (i = 1; i <= n; ++i)
	{
		if (s[i][0] == '/')//第一次全部集中
			hit[i][0] = 10;
		else if (s[i][1] == '/')//第而次全部集中
		{
			hit[i][0] = s[i][0] - '0';
			hit[i][1] = 10 - hit[i][0];
		}
		else
		{
			hit[i][0] = s[i][0] - '0'; //  每轮的击中数
			hit[i][1] = s[i][1] - '0';
		}
	}
	for (i = 10; i; --i)
	{
		if (s[i][0] == '/') //  第一次就全部击中
			score[i] = hit[i+1][0] + ((s[i+1].size() < 2) ? hit[i+2][0] : hit[i+1][1]) + 10;
		else if (s[i][1] == '/') //  第二次全部击中
			score[i] = hit[i+1][0] + 10;
		else //  没有全部击中
			score[i] = hit[i][0] + hit[i][1];
		res += score[i];
	}
	printf("%d", res);
	return 0;
}
```

5 [3070]乒乓球

国际乒联立志推行一系列改革，以推动乒乓球运动在全球的普及。其中 11 分制改革引

起了很大的争议，有一部分球员因为无法适应新规则只能选择退役。华华就是其中一位，他退役之后走上了乒乓球研究工作，意图弄明白 11 分制和 21 分制对选手的不同影响。在开展研究之前，他首先需要对自己多年比赛的统计数据进行一些分析，所以需要一些帮忙。
华华通过以下方式进行分析，首先将比赛每个球的胜负列成一张表，然后分别计算在 11 分制和 21 分制下，双方的比赛结果(截至记录末尾)。
比如，现在有这么一份记录，(其中 W 表示华华获得一分，L 表示华华对手获得一分)：
WWWWWWWWWWWWWWWWWWWWWLW。在 11 分制下，此时比赛的结果是华华第一局 11 比 0 获胜，第二局 11 比 0 获胜，正在进行第三局，当前比分 1 比 1。而在 21 分制下，此时比赛结果是华华第一局 21 比 0 获胜，正在进行第二局，比分 2 比 1。如果一局比赛刚开始，则此时比分为 0 比 0。
本题就是要对于一系列比赛信息的输入(WL 形式)，输出正确的结果。
[输入格式]
输入文件包含若干行字符串(每行至多 20 个字母)，字符串由大写的 W、L 和 E 组成。其中 E 表示比赛信息结束，程序应该忽略 E 之后的所有内容。
[输出格式]
输出由两部分组成，每部分有若干行，每一行对应一局比赛的比分(按比赛信息输入顺序)。其中第一部分是 11 分制下的结果，第二部分是 21 分制下的结果，两部分之间由一个空行分隔。
[输入样例]
WWWWWWWWWWWWWWWWWWWW
WWLWE
[输出样例]
11：0
11：0
1：1
21：0
2：1
[问题分析]
读入字符串，分别按照 11 分制和 21 分制下的规则模拟比赛进行计分输出。

```
#include <bits/stdc++.h>
#include <cstring>
using namespace std;
int main()
{
    char a[100000];
    char c;
    int win=0,lost=0,total,i=0;

    scanf("%c",&c);   //E 是输入的结束标记
    while( c!='E')
    {
        a[i++]=c;   //把输入的字符存入数组 a
        scanf(" %c",&c);
    }
```

```
    total=i;//total 表示输入字符的个数
    for(i=0;i<total;i++)
    {
        if(a[i]=='W') win++;
        else lost++;
        if((win>=11|| lost>=11) && abs(win-lost)>=2)
        {   //比分差 2 分才算确定输赢
            printf("%d:%d\n",win,lost);
            win=0; //开始下一组
            lost=0;
        }

    }
    printf("%d:%d\n",win,lost);//输出最后一组数据
    printf("\n");
    win=0;   //开始 21 分制
    lost=0;
    for(i=0;i<total;i++)
    {
        if(a[i]=='W') win++;else lost++;
        if((win>=21|| lost>=21) && abs(win-lost)>=2)
        {
            printf("%d:%d\n",win,lost);
            win=0;
            lost=0;
        }
    }
    printf("%d:%d\n",win,lost);
    return 0;
}
```

课堂作业列表

[2948]醉酒的狱卒　　[3204]模拟约瑟夫环
[6869]网络堵塞　　[6865]蚱蜢
[6866]遭遇战　　[3070]乒乓球
[6867]保龄球枚举习题

课前练习

#include <iostream> using namespace std; int a(int m, int n)	#include <iostream> using namespace std; int i, a, b;

```
{
    int x;
    if (m == 0)
        x = n + 1;
    else if (n == 0)
        x = a(m - 1, 1);
    else
        x = a(m - 1, a(m, n - 1));
    return x;
}

int main()
{
    cout << a(1, 2) << endl;
    return 0;
}
```

输出：_____

```
int f(int i)
{
    if (i == 0 || i == 1)
        return 3;
    else
        return i - f(i - 2);
}
int main()
{
    i = 103;
    a = i / 10;
    b = i % 10;
    cout << f(a) << " " << f(b);
    return 0;
}
```

输出：_____

习题

1．[2004]

```
#include<iostream>
int main(){
    int i, j;
    char str1[] = "pig-is-stupid";
    char str2[] = "clever";
    str1[0] = 'd'; str1[1] = 'o';
    for (i = 7, j = 0; j < 6; i++, j++)
        str1[i] = str2[j];
    printf("%s\n", str1);
    return 0;
}
```

输出：_____

2．[2004]

```
#include<iostream>int main(){
    int u[4], a, b, c, x, y, z;
    scanf("%d %d %d %d",&(u[0]), &(u[1]), &(u[2]), &(u[3]));
    a = u[0] + u[1] + u[2] + u[3] - 5;
    b = u[0] * (u[1] - u[2] / u[3] + 8);
    c = u[0] * u[1] / u[2] * u[3];
    x = (a + b + 2) * 3 - u[(c + 3) % 4];
    y = (c * 100 - 13) / a / (u[b % 3] * 5);
    if ((x + y) % 2 == 0) z = (a + b + c + x + y) / 2;
```

```
        else z = (a + b + c -x - y) * 2;
        printf("%d\n", x + y - z);
        return 0;
}
```
输入：2 5 7 4

输出：_____

3．[2005]

```
#include<iostream>
int main()
{
char str[20] = "Today-is-terrible!";
int i;
for (i = 6; i <= 10; i++)
if (str[i] == '-') str[i - 1] = 'x';
for (i = 12; i >= 0; i--)
if (str[i] == 't') str[i + 1] = 'e';
printf("%s\n", str);
return 0;
}
```
输出：_____

4. 出列游戏：n 个人围成一圈，从第 1 个人开始报数，报数报到 m 的人出列；然后又从下一个人开始从 1 开始报数；重复 n-1 轮游戏，每轮游戏淘汰 1 个人，最后剩下的人就是胜利者。模拟该游戏，输出依次出列的位置及最后的胜利者

```
 #include <iostream>
using namespace std;
int main( ) {
      int n,m;
      cin>>n>>m;
      for( i=0; i<=n; i++ ) a[i] = i;
      int r = 1;  //第 r 轮
      int i,j;
      for( i=1, j=1; r<=n-1; i=__1_____  , j=__2_____    )
      {
            while( a[i]==-1 ) { i = i%n+1; }
            if(___3_____    )
            {
                  printf( "%d ",a[i] );
                  a[i] =-1; j = 0;
                  __4_____;
            }
      }
      for( i=0; i<n; i++ )
      {
```

```
            if( a[i+1]!=-1 )
            printf( "%d\n", a[i+1]);
    }
      return 0;
}
```

第十二讲 模拟应用

1 [7319] 听歌识曲

题目描述

洛洛有一份私人歌单，歌单里面塞满了他喜欢的歌曲，像夏恋、雨道、彩月、幻昼……整整有好几百首。洛洛每天都要把他的歌单听一遍，以致于他都能知道在什么时候放的是什么歌。

洛洛在向你推荐了他的歌单之后，决定考考你，从他的歌单开始播放起，第 t 秒正在播放的是第几首歌。

输入

第一行输入两个整数 n 和 t， 分别表示歌单的歌曲总数以及第 t 秒播放哪首歌。

第二行有 n 个整数， A1， A2， …， An， 分别表示歌单的第 i 首歌将会播放多长时间。

输出

输出一个整数，表示歌单按顺序播放后，第 t 秒播放的是第几首歌。

样例输入

3 5

1 3 5

样例输出

3

提示

歌单中总共有三首歌：

第一首歌播放 1 秒，占第 1 秒；

第二首歌播放 3 秒，占第 2-4 秒；

第三首歌播放 5 秒，占第 5-9 秒。

所以第 5 秒播放的是第三首歌曲。

对于 30%的数据，保证 1 ≤ n ≤ 3；

对于 60%的数据，保证 1 ≤ n ≤ 2000， 1 ≤ Ai ≤ 500；

对于 100%的数据，保证 1 ≤ n ≤ 100000， 1 ≤ Ai ≤ 1000， 1 ≤ t ≤ sum(Ai)

思路：

通过从前往后的累加，就可以知道每首歌结束的时候在第几秒，用一个数组来记录每首歌结束的时间（第 i 首歌结束的时间为 a[i]秒）。最后找到第一个结束时间晚于指定时间 t 的歌，输出它的下标即可。

```
#include<bits/stdc++.h>
using namespace std;
int a[100005];
int main()
{
    int n,t,m,i,j,sum=0;a[0]=0;
    scanf("%d %d",&n,&t);//一共有 n 首歌，求第 t 秒播放的歌
    for(i=1;i<=n;i++){
```

```
        scanf("%d",&m);
        sum=sum+m;//每次累加，即为每首歌结束的时间
        a[i]=sum;//将每次累加的值赋值给数组 a[]
    }
    for(i=1;i<=n;i++){//从头开始找
        if(a[i-1]<t&&a[i]>=t){
            printf("%d",i);break;//找到第一首结束的时间晚于指定时间的歌,输出
下标
        }
    }
    return 0;
}
```

2 [7340] 螺旋方阵

题目描述

输入一个正整数 N(1<=N<=20)后 ,可以得到一个 N*N 的数字螺旋方阵: 先分别求出该方阵中的主对角线与副对角线上的数字之和 S,P,然后输出 S、P 的积。

例如,N=5 时得到的数字螺旋方阵如下:

1 2 3 4 5

16 17 18 19 6

15 24 25 20 7

14 23 22 21 8

13 12 11 10 9

其中:主对角线从左上角到右下角,得到的数字之和 S=1+17+25+21+9=73,

副对角线从右上角到左下角,得到的数字之和 P=5+19+25+23+13=85。

最后 S*P=6205.

输入只有一行,一个正整数 N

输出只 有一行,一个正整数 (表示主对角线与副对角线上的数字之和的积)。

样例输入 5

样例输出 6205

思路:先对一个二维数组进行赋值，使之符合螺旋矩阵的要求，再求主对角线和副对角线的数字之和的积

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int N=25;
int a[N][N];
int main()
{
    //先赋值，后计算 行列均为 1-n
    int n,k=0,t; //k：计数 赋值
    cin>>n; //输入方阵大小 n*n
```

```
    t=n*n;   //方阵数字的个数
    int x=1,y=0; //（x，y）表示当前在方阵的哪个位置
    while(k<t)   //当 k>=t 时，表示赋值完成
    {
        while(!a[x][y+1]&&y<=n-1) a[x][++y]=++k;
         //从左向右赋值，x（行）不变，y（列）递增，能够赋值的条件：当下一个格子还没有被赋值并且 y 不能超出方阵的范围
        while(!a[y][x+1]&&x<=n-1) a[++x][y]=++k;
        //从上向下赋值，y（列）不变，x（行）递增，能够赋值的条件：当下一个格子还没有被赋值并且 x 不能超出方阵的范围
        while(!a[x][y-1]&&y>=2) a[x][--y]=++k;
        //从右向左赋值，x（行）不变，y（列）递减，能够赋值的条件：当下一个格子还没有被赋值并且 y 不能超出方阵的范围
        while(!a[x-1][y]&&x>=2) a[--x][y]=++k;
        //从下向上赋值，y（列）不变，x（行）递减，能够赋值的条件：当下一个格子还没有被赋值并且 x 不能超出方阵的范围
    }
    int sum1=0,sum2=0;
    for(int i=1;i<=n;i++)
    {
        sum1+=a[i][i]; //主对角线数字之和 (主对角线上（x，y）x=y)
        sum2+=a[i][n-i+1]; //副对角线数字之和 （副对角线上（x，y）x+y=n+1）
    }
    int sum=sum1*sum2;
    cout<<sum;
    return 0;
}
```

3 [7343] 号码分类

小明有来自 A、B、C 三城市的 n 个朋友，现在要将他们的电话号码按 A、B、C 的顺序分类输出，但相同地区的号码则仍按原序输出。已知各城市电话号码的第一位是不同的：A 城为 8，B 城为 5，C 城为 2。

输入

共二行，第一行有一个正整数 n（n<=100），表示朋友的数目。 第二行是 n 个八位电话号码（号码间以空格相隔）。

输出

共三行。格式如下：

A：A 城朋友的电话号码（以空格相隔，如没有 A 城的，则空着）

B：B 城朋友的电话号码（以空格相隔，如没有 B 城的，则空着）

C：C 城朋友的电话号码（以空格相隔，如没有 C 城的，则空着）

样例输入

3

85552088 22826558 82222205

样例输出
A: 85552088 82222205
B:
C: 22826558

思路:取出号码的第一位数，根据条件分别存入 A.B.C.数组。再依次输出。

```
#include<bits/stdc++.h>
using namespace std;
int A[110],B[110],C[110];
int a,b,c;
int main(){
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
        int x;
        scanf("%d",&x);
        int f=x/10000000;//取得号码的第一位（八位数/10000000）
        //根据 f 是多少，将号码存入不同的数组
        if(f==8) A[++a]=x;
        else if(f==5) B[++b]=x;
        else if(f==2) C[++c]=x;
    }
    //A.B.C 依次输出。注意：A:和号码之间有一个空格(A: xxxxxxxx)
    printf("A:");
    for(int i=1;i<=a;i++)
        printf(" %d",A[i]);
    printf("\n");
    printf("B:");
    for(int i=1;i<=b;i++)
        printf(" %d",B[i]);
    printf("\n");
    printf("C:");
    for(int i=1;i<=c;i++)
        printf(" %d",C[i]);
    printf("\n");
    return 0;
}
```

4 [7344] 排队

程程是一个喜欢跳舞的女孩，还报了一个专门学习跳舞的班呢。

在入学的时候，老师让大家根据自己的身高排了一下队，个子小的同学排前面，个子高的同学排在后面，高度相等的先后顺序随意.。

如果给你这些同学的身高数据，你能计算一下程程最前可以排第几、最后可以排第几吗？

输入共三行。第一行是一个整数 n(1<=n<=30)，表示跳舞班所有同学的人数。第二行是 n 个整数，表示所有同学的身高，以厘米为单位。这 n 个同学的数据，包括程程本人的。 第三行是一个整数，表示程程的身高。

输出文件中只有两个整数，用空格分开。分别表示：从前面数，程程可能排的最前的位置和最后的位置。

样例输入

8

133 134 132 133 131 130 138 133

133

样例输出

4 6

思路:根据所用人的身高进行排序。找到第一个和程程身高一样的位置，和这之后位置中第一个和程程身高不一样的位置。第一个找到的是最前位置，第二个找到的位置的前一位置是最后位置

```
#include<bits/stdc++.h>
using namespace std;
int a[35];
int main(){
    int n,m;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    scanf("%d",&a[i]);
    sort(a+1,a+1+n);//身高按照从小到大排序
    scanf("%d",&m);
    int i,j;
    for(i=1;a[i]<m&&i<=n;i++);
    //找到顺序上第一个和程程一样高的人，这所在位置是她可以站的最前位置
    for(j=i+1;j<=n;j++)
    if(a[j]!=a[i]) break;//身高相等的人随意站，
    //找到从 i 往后第一个不同的，这个的前一个位置就是最后位置
    printf("%d %d",i,j-1);
    return 0;
}
```

5 [7347] 立方和

现给出一个三位数，先对这个三位数的各位数字的立方求和，然后再对求出的和中的各位数字的立方求和，如此一直继续下去，判断最后能否得到一个不再变化的固定值。如能得到一个固定值，就求出这个固定值；如果不能，则输出提示信息“error”。另外请注意，在求解过程中，若某一次求和过程中得到的值超过三位数，则取该数的低三位继续往下运算……

例如，对于三位数 111，则第一次计算应是 1×1×1+1×1×1+1×1×1=3，第二次计算应是

0×0×0+0×0×0+3×3×3=27，第三次计算应是 0×0×0+2×2×2+7×7×7=351，第四次计算应是 3×3×3+5×5×5+1×1×1=153，第五次计算应是 1×1×1+5×5×5+3×3×3=153，与第四次计算的结果相同，这时可不再计算，输出固定值 153。

亲爱的同学，请你也来计算一下。

输入只有一行，是一个三位数

输出也只有一行，如能得到一个固定值，则输出这个固定值；如不能，则输出一个提示信息“error”

样例输入 111

样例输出 153

思路:纯模拟，定义三个变量储存各位数字，开一个 vis 数组记录这个三位数有没有出现过。然后我们用永真循环，如果这个立方和数出现过直接输出 error，然后 return 0，如果这个立方和数和原来的相同则输出这个数。

代码

```
#include <bits/stdc++.h>
using namespace std;
typedef unsigned long long ll;
const int N = 1e3 + 5;
int vis[N];
int main(){
    int n;
    cin >> n;
    while (1) {
        int a = n % 10;
        int b = (n / 10) % 10;
        int c = (n / 100) % 10;
        int t = a * a * a + b * b * b + c * c * c;
        t %= 1000;
        if (t == n) {
            cout << t;
            return 0;
        }
        if (vis[t]) {
            cout << "error";
            return 0;
        }
        vis[t]++;
        n = t;
    }
    return 0;
}
```

课前练习

#include <iostream>	#include <iostream>

```
using namespace std;
int x,y,z;
void silly(int x,int y)
{
    x=7;
    y=17;
    z=18;
    cout<<x<<" "<<y<<" "<<z<<endl;
}
int main()
{
    x=1;
    y=2;
    z=3;
    silly(x, y);
    cout<<x<<" "<<y<<" "<<z<<endl;
    return 0;
}
```

输出：

7 17 18

1 2 3

```
using namespace std;
int n;
void change(int n)
{
    int i,j;
    if(n==0)
        return;
    i=n%8;
    j=n/8;
    change(j);
    cout<<i;
    return;
}
int main()
{
    cin>>n;
    change(n);
    return 0;
}
```

输入：

2017

输出： 3741

习题

1.
```
#include<iostream>
using namespace std;
int main()//题意：从 n~m 相加
{
    int i,n,m,ans;
    cin>>n>>m;
    i=n;
    ans=0;
    while(i<=m){
        ans+=i;
        i++;
    }
    cout<<ans<<endl;
    return 0;
}
```

输入：10 20

输出：_____

2. #include<iostream>
#include<cstring>
using namespace std;
const int SIZE = 100;
int main()
{
    int n,i,sum,x,a[SIZE];
    cin>>n;
    memset(a,0,sizeof(a));
    for(i=1;i<=n;i++){
        cin>>x;
        a[x]++;//记录数 x 出现的次数
    }
    i=0;
    sum=0;
    while(sum<(n/2+1)){//统计在 1~i 数出现的次数大于 n/2+1
        i++;
        sum+=a[i];
    }
    cout<<i<<endl;
    return 0;
}
输入：
11
4 5 6 6 4 3 3 2 3 2 1
输出：_____

3. #include <iostream>
using namespace std;
int n,i,ans;
int main()
{
    cin>>n;
    ans=0;
    for(i=1;i<=n;i++)
        if(n%i==0) ans++;//查找 1~n 中有多少 n 的因数
    cout<<ans<<endl;
    return 0;
}

输入：18
输出：_____

第十三讲　枚举初步

枚举又称为穷举，是一种很朴素的解题思想。计算机的特点之一就是运算速度快，善于重复做一件事。"穷举"正是基于这一特点的最古老的算法。它一般是在一时找不出解决问题的更好途径，即从数学上找不到求解的公式或规则时，根据问题中的"约束条件"，将解的所有可能情况一一列举出来，然后再逐个验证是否符合整个问题的求解要求，从而求得问题的可行解或者最优解。

例如，在判断 m=199 是否为素数时，从素数的定义出发，试图找出 2~198 范围内能整除 m 的自然数，如果不能找到，则 m 是素数。根据分析，我们将范围缩小到 2~14。依次判断 2 能否整除 m，判断 3 能否整除 m，…，一直判断到 14 都还没有找到能整除 m 的自然数，因此得出结论：m=199 是素数。这个过程实际上就是一个枚举的过程，枚举 2, 3, 4, 5, … ,14 能否整除 m。

13.1　枚举的基本思路

例 1[3309]　求 $x^2 + y^2 = 2000$ 的正整数解。

分析：x 和 y 都是正整数，因此 x 和 y 的取值范围只能是：1、2、…、44，其中 44 是小于等于 sqrt(2000)的最大正整数。对于在这个范围内的所有(x, y)组合，都去判断一下。也就是枚举所有的(x, y)组合，判断是否满足 x2+y2=2000，如果满足，则是一组解。即当 x 取 1 时，考虑 y 取 1、2、…、44；然后当 x 取 2 时，又考虑 y 取 1、2、…、44；…；最后当 x 取 44 时，又考虑 y 取 1、2、…、44。整个过程如图 4.1 所示。在实现时要用到 2 重循环，从算法思想的角度看，这个过程就是枚举，即枚举所有的(x, y)组合。

```
 #include<iostream>#include <math.h>
int main( )
{
    int x, y;
    int m = sqrt(2000);  //循环变量 x 和 y 的终值
    for( x=1; x<=m; x++ )      //x 从 1 枚举到 m
    {
        for( y=1; y<=m; y++ )      //y 也从 1 枚举到 m
        {
            if( x*x + y*y == 2000 )    //判断相等必须用"=="
                printf( "2000=%d*%d+%d*%d\n", x, x, y, y );
        }
    }
    return 0;
}
```

思考：从运行结果可以看出，(8, 44)这一组解和(44, 8)这一组解实际上只是交换了 x 和 y。如果认为这是同一组解，那么方程就只有两组解：(4, 44)和(20, 40)，该怎样修改程序呢？

```
 #include<iostream>#include <math.h>
int main( )
{
    int x, y;
```

```
    int m = sqrt(2000);  //循环变量 x 和 y 的终值
    for( x=1; x<=m; x++ )      //x 从 1 枚举到 m
    {
        for( y=x; y<=m; y++ )      //y 也从 1 枚举到 m
        {
            if( x*x + y*y == 2000 )    //判断相等必须用"=="
                printf( "2000=%d*%d+%d*%d\n", x, x, y, y );
        }
    }
    return 0;
}
```

13.2 案例讲解

1 [2019] 开关灯

一条线上有 N 盏灯，灯有开有关，1 表示开，0 表示关（ON/OFF），您的任务是确定至少需要切换多少次灯（从 ON 到 OFF，或者从 OFF 到 ON），以使灯交替打开和关闭。

如 1110111 -》 1010101 操作两次

1110111 -》 0101010 操作五次

输入

每个测试数据一行。第一个数为整数 n 表示灯的数量（1＜n<＝10000），接着是表示灯的状态的 n 个整数（ON 为 1），OFF 为“0”。

输出

对于每个测试数据，输出至少需要操作多少次开关。

样例输入

3 1 1 1

3 1 0 1

样例输出

1

0

分析：要使得 N 盏灯开和关交替出现，实际上只有两种可能：奇数位置上的灯是开着的，和偶数位置上的灯上是开着的。只要分别计算从 N 盏灯的初始状态出发，切换到这两种状态所需要切换灯的数目，取较小者即可。

稍加分析就可以得到结论：如果将 N 盏灯调整成奇数位置上的灯是开着的，需要调整灯的数目为 numo，则将这 N 盏灯调整为偶数位置上的灯是开着的，需要调整灯的数目 nume = N － numo。因此只要判断 numo 是否小于 N/2，如果是则取 numo，否则取 N － numo。

```
#include<iostream>
using namespace std;
int main( )
{
    int N, i;          //N 表示灯的数目
    int numo;//调整成奇数位置上的灯是开着的需要调整灯的数目
```

```
    int lights[10001];
    while( scanf( "%d", &N )!=EOF )
    {
        numo = 0;
        for( i=1; i<=N; i++ )   scanf( "%d", &lights[i] ); //读入初始状态
         //将 N 盏灯的转换调整为奇数位置上为 1，偶数位置上为 0
        for( i=1; i<=N; i++ )
        {
            if(    1                    ) //奇数位置为 0，需要调整
                numo++;
            if(   2                     ) //偶数位置为 1，需要调整
                numo++;
        }
        printf( "%d\n", numo<=N/2 ? numo : N-numo );//"="不能去掉
    }
    return 0;
}
```

2 [1311] 歌德巴赫猜想

1742 年，德国数学家哥德巴赫(Goldbach)提出了著名的哥德巴赫猜想(Goldbach Conjecture)：任何一个不小于 4 的偶数可以表示为两个素数之和。这个猜想至今都没有完全被证明是正确的。但是，对于一个大于或等于 5 的奇数，有的可以表示成两个素数之和，有的则不能。**给定一个大于或等于 5 的奇数，判断是否能分解成两个素数之和。**

输入输入文件包含多个测试数据，每个测试数据占一行，为一个正整数 m，m 为奇数，且不小于 5，不大于 32767。测试数据一直到文件尾。

输出对每个测试数据，如果 m 能分解成两个素数之和，输出 yes，否则输出 no。

样例输入	样例输出
21	yes
75	yes
99	yes
113	no

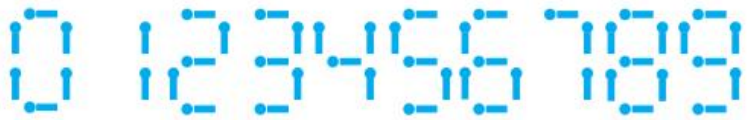
分析：写一个判断素数的函数，然后枚举每一个数 i 与 n-i 是否都是素数即可。这里可以优化 i 的取值范围。

```
#include<iostream>
#include<cmath>
using namespace std;
int prime(int x){
    for(int i=2;i<=sqrt(x);i++)
        if(  4                    );
    return 1;
```

```
}

int main(){
    int n,flag=1;
    while(cin>>n){
        ____1_____;
        for(int j=2;j<n;j++){
            if(____2_____)){
                cout<<"yes"<<endl;
                flag=0;
                break;
            }
        }
        if(____3_____)cout<<"no"<<endl;
    }
```

3 [6860 2975]火柴棒等式

[问题描述]
给出 n 根火柴棒，可以拼出多少个形如"A+B=C"的等式？
等式中的 A、B、C 是用火柴棒拼出的整数（若该数非零，则最高位不能是 0）。用火柴棒拼数字 0~9 的拼法如图所示。



需要注意以下几点：
（1） 加号与等号各自需要两根火柴棒。
（2） 如果 A ≠ B，则 A+B=C 与 B+A=C 视为不同的等式（A、B、C 均大于或等于 0）。
（3） n 根火柴棒必须全部用上（n≤24）。
[输入样例]14
[输出样例]2
[样例说明]两个等式分别为：0+1=1 和 1+0=1。

问题分析

首先，预处理每个数字（0~9）需要用几根火柴棒，存储在数组 f 中。然后，穷举 a 和 b，算出它们的和 c，再判断下列约束条件是否成立：f （a）+f （b）+f （c）=n-4。现在的问题是：a 和 b 的范围有多大？可以发现尽量用数字 1 拼成的数比较大，分析可知最多不会超过 1111。程序实现时，分别用三个循环语句预处理好所有两位数、三位数、四位数构成所需要的火柴棒数量。

```
#include <iostream>
using namespace std;
int fun(int x)      //用来计算一个数所需要的火柴棍总数
{
    int num=0;   //用来计数变量
```

```
        int f[10]={6,2,5,5,4,5,6,3,7,6};   //用一个数组记录 0~9 数字所需的火柴棍数
        while(  1          )        // x 除以 10 不等于 0 的话，说明该数至少有两位
        {
            num+=  2                ;     //加上该位火柴棍数
            x=x/10;
        }
        num+=f[x];        //加上最高位的火柴棍数
        return  3                ;
    }
    int main()
    {
         int a,b,c,m,sum=0;
         cin>>m;              //火柴棍总个数
         for(a=0;  4                          )   //开始枚举
         {
              for(b=0;  5                          )
              {
                   c=a+b;
                   if(  6                          )   //去掉+和=
                        sum++;
              }
         }
         cout<<sum<<endl;
        return 0;
}
```

4 [2660 6864]金币

国王将金币作为工资，发放给忠诚的骑士。第一天，骑士收到一枚金币；之后两天（第二天和第三天）里，每天收到两枚金币；之后三天（第四、五、六天）里，每天收到三枚金币；之后四天（第七、八、九、十天）里，每天收到四枚金币……这种工资发放模式会一直这样延续下去。当连续 n 天每天收到 n 枚金币后，骑士会在之后的连续 n+1 天里，每天收到 n+1 枚金币（n 为正整数）。

请编程确定从第一天开始的给定天数内，骑士一共获得了多少金币。

[输入格式]

输入包含至少一行，但不多于 1000 。

除最后一行外，输入的每行是一组输入数据，包含一个正整数 n，表示天数。

输入的最后一行为 0，表示输入结束。

[输出格式]

对每个数据输出一行一个整数，表示该数据对应总金币数。

[输入样例]	[输出样例]
10 6 7	30 14 18

11	35
15	55
16	61
100	945
10000	942820
1000	29820
21	91
22	98
0	

[数据规模]

对于 60% 的数据满足：$n \leqslant 10^3$ ；

对于 80% 的数据满足：$n \leqslant 10^6$ ；

对于 100% 的数据满足：$n \leqslant 10^{12}$ 。

[问题分析]

每次穷举每天获得的金币数，或者将答案保存在数组中直接输出，可以获得部分分。下面利用数学推导进行优化。

方法一：找规律，第一个为第一组，后两个为第二组，在后两个为第三组……以此类推。思路就是找到 n 在第几组里，如果 n 在这一组的末尾，则直接求平方和；若在这一组的中间，则还需求出它在这一组的第几个，将数据分为从第一组到前一组和这一组来算。判断它在第几组的方法是：发现每一组的最后一个数是所有组数的和，因此将组数升序加起来，一旦当天数 n 刚不大于组数，就说明 n 在该组里。

代码：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int n,a=0,b,sum=0,result=0;
    scanf("%d",&n);//输入天数
    for(int i=1;;i++){ //分为两类讨论
        sum+=i;//循环，直到从第一组开始的每组数据的个数加起来不小于天数
        if(sum>n){//讨论数据个数大于天数的情况，此时 n 在第 i 组里
            for(int j=1;j<=i-1;j++)
                a+=j;//前（i-1）组求和得最后一天的天数
            b=n-a;//b 为超出（i-1）组的，在第 i 组里的天数
            for(int k=1;k<=i-1;k++)
                result+=pow(k,2);
            result+=b*i;//两部分加起来得结果
            break;
        }
        else if(sum==n){//讨论数据个数小于天数的情况
            for(int k=1;k<=i;k++){
                result+=pow(k,2);
            }//天数正好在第 i 组的末尾，直接求
```

```
            break;
        }

    }
    printf("%d",result);//输出结果
    return 0;
}
```

方法二：

```
#include<iostream>
using namespace std;
int main()
{
   int n;//1 2 2 3 3 3 4 4 4 4
   cin >> n; //1+2*2+3*3+4*4+....
   int res = 0;
   int cmp = 1;
   while (n >= cmp)
   {
      n -= cmp;
      res += cmp*cmp;
      cmp++;
   }
   res += n*cmp;
   cout << res << endl;
   return 0;
}
```

方法三：

```
#include<iostream>
using namespace std;
int main()
{
   int K,N,coin=0;
   scanf("%d",&K);
   for(N=1;K-N>=0;K-=N++)
      coin+=N*N;
   printf("%d\n",coin+K*N);
   return 0;
}
```

课堂作业列表

2019 切换状态(Switch)
1311 歌德巴赫猜想
6860 2975 火柴棒等式
2660 6864 金币
3344 比例简化
6861 奶牛碑文

课前练习

```
#include <iostream>
using namespace std;
int n;
int s(int n, int t)
{
    if(n==0)return(1);
    else if (t==0)return s(n-1, t+1);
    else return s(n-1, t+1)+s(n, t-1);
}
int main()
{
    cin>>n;
    cout<<s(n, 0);
}
```

输入：
4
输出：_____

```
#include <iostream>
using namespace std;
int n,sum;
void f(int n,int a, int c,int b) {
    if(n==0)return;
    f(n-1,a,b,c);
    f(n-1,b,c,a);
    sum=sum+1;
}
int main() {
    cin>>n;
    sum=0;
    f(n,1,0,1);
    cout<<sum<<endl;
    return 0;
}
```

输入：
6
输出：_____

习题

```
1.#include<stdio.h>
int main()
{
  int x,i,n,minn=100,maxn=0,sum;
  scanf("%d",&n);
  sum=0;
  for(i=1;i<=n;i++){
    scanf("%d",&x);
    sum=sum+x;
```

```
    if(x>maxn)
    maxn=x;
    if(x<minn)
    minn=x;
  }
printf("%.2lf",1.0*(sum-minn-maxn)/(n-2);
  return 0;
}
```

输入： 5 98 90 95 88 85

输出_____

2.

```
#include<stdio.h>
int main()
{
    int k=2,n=0;
    double Sn=0;
    while(Sn<=k){
        n=n+1;
        Sn=Sn+1.0/n;
    }
    printf("%d",n);
    return 0;
}
```

输出_____

3. 题目描述

假设有 N 盏灯(N 为不大于 5000 的正整数)，从 1 到 N 按顺序依次编号，初始时全部处于开启状态；有 M 个人(M 为不大于 N 的正整数)也从 1 到 M 依次编号。第一个人(1 号)将灯全部关闭，第二个人(2 号)将编号为 2 的倍数的灯打开，第三个人(3 号)将编号为 3 的倍数的灯做相反处理（即将打开的灯关闭，将关闭的灯打开）。依照编号递增顺序，以后的人都和 3 号一样，将凡是自己编号倍数的灯做相反处理。请问：当第 M 个人操作之后，哪几盏灯是关闭的，按从小到大输出其编号，其间用逗号间隔。

10 10

样例输出

1,4,9

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
  bool a[5001]=____1_____;
  int n, m;
  cin>>n>>m;
  for(int i=1;i<=m;i++)
```

```
        for(int j=_2_____________________)
            a[j]=!a[j];
    for(int i=1,j=0;i<=n;i++)
        if(a[i]==true)
        {
            j++;
            If_3_____________cout<<i;
            else cout<<','<<i;
        }
    return 0;
}
```

第十四讲　枚举应用

1[7320] 多面骰子

洛洛现在手上有三颗多面骰子，多面骰子不是常见的六面骰子，而是 33 面骰子、100 面骰子……一般来说， i 面骰子每个面上的点数分别是 1， 2， 3，……i。
洛洛手上的三颗骰子的面数可能并不相同，他想知道掷出三颗骰子的所有情况中，三颗骰子的点数之和出现最多次数是几点。
如果存在多个点数之和出现次数相同的情况，则按点数之和从小到大顺序输出。
输入
第一行输入三个整数 n1， n2， n3，分别表示三颗骰子各自的面数。
输出
输出一行含任意个整数，分别表示次数最多的点数之和，用空格隔开。
样例输入
1 2 3
样例输出
4 5
提示
样例 1，骰子投出来有以下六种情况：
1+1+1=3 1+1+2=4 1+1+3=5
1+2+1=4 1+2+2=5 1+2+3=6
可以看出 4 和 5 的出现次数最多且都是 2 次。
对于 50%的数据，保证 1 ≤ n1， n2， n3 ≤ 5；
对于 80%的数据，保证 1 ≤ n1， n2， n3 ≤ 10。
对于 100%的数据，保证 1 ≤ n1， n2， n3 ≤ 100。
题目类型：枚举
思路：
因为这道题目的数据范围很小，我们可以直接枚举出所有的情况是可以的。那就直接用三层循环，枚举出所有可能的点数和，用一个数组来记录每种结果出现的次数。最后输出出现次数最多的点数。

```
#include<bits/stdc++.h>
using namespace std;
int a[1000];
int main(){
    int n1,n2,n3,i,j,k,sum=0,max=0;
    scanf("%d %d %d",&n1,&n2,&n3);//输入三个骰子的面数
    for(i=1;i<=n1;i++){
        for(j=1;j<=n2;j++){
            for(k=1;k<=n3;k++){
                sum=i+j+k;//sum 即为对应的点数和
                a[sum]++;//对应的点数和出现次数加一
                if(a[sum]>max)max=a[sum];//用 max 来记录出现的最多次数
```

```
            }
        }
    }
    for(i=1;i<305;i++){//题目给出最多有 100 面，那么点数最大为 300
        if(a[i]==max)printf("%d ",i);//输出出现次数为 max 的所有点数
    }
    return 0;
}
```

2 [7509] 排名

宁波市的小学生们在镇海中学完成程序设计比赛后，老师们批出了所有学生的成绩，成绩按分数从高到低排名，成绩相同按年级从低到高排（注：纯属虚构，请勿对号入座）。现在主办单位想知道每一个排名的学生前，有几位学生的年级低于他（她）。

输入

第 1 行只有一个正整数 n(1<=n<=200),表示参赛的学生人数。

第 2 行至第 n+1 行共 n 行，每行有两个正整数 s(0<=s<=400),g(1<=g<=6)。其中第 i+1 行的第一个数 s 表示第 i 个学生的成绩，第 i+1 行的第二个

输出

每行只有一个正整数，其中第 i 行的数 k 表示排名第 i 名的学生前面有 k 个学生排名比他（她）高，且年级比他（她）低。

样例输入	样例输出
5 300 5 200 6 350 4 400 6 250 5	0 0 1 1 3

提示：

50%的数据，每个学生的成绩互不相同。

思路

用结构体排序后开两个循环求。

代码

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e2 + 5;
int n;
struct node {
    int score;int grade;
}f[N];//结构体
bool cmp(node a, node b) {//cmp 函数
```

```
    if (a.score == b.score)
        return a.grade < b.grade;
    return a.score > b.score;
}
int main() {
    scanf("%d", &n);
    for (int i = 1;i <= n;i++) scanf("%d%d", &f[i].score, &f[i].grade);
    sort(f + 1, f + n + 1, cmp);
    for (int i = 1;i <= n;i++) {
        int k = 0;
        for (int j = 1;j < i;j++)
            if (f[j].score >= f[i].score && f[j].grade < f[i].grade)//找出符合条件的人
                k++;
        printf("%d\n", k);
    }
    return 0;
}
```

3 [3339] 三连击 2

将 1，2，…，9 共 9 个数分成三组，分别组成三个三位数，且使这三个三位数的比例是 A:B:C，试求出所有满足条件的三个三位数，若无解，输出“No!!!”。

输入

三个数，A B C。

保证 A<B<C

输出

若干行，每行 3 个数字。按照每行第一个数字升序排列。

样例输入

1 2 3

样例输出

192 384 576

219 438 657

273 546 819

327 654 981

```
#include<iostream>
using namespace std;
int a[10],b1,b2,b3,l,k1,k2,k3,ans;
int main ()
{
  cin >>k1>>k2>>k3;
  for (int b=1;b<=1000/k3;++b)
  {
    b1=b*k1;//求出三个数
```

```
    b2=b*k2;
    b3=b*k3;
    if (b2>999||b3>999)break;
    for (int c=1;c<=3;++c)//将三个数进行分解，判断是否重复
    {
      a[b1%10]++;
      b1/=10;
    }
    for (int c=1;c<=3;++c)
    {
      a[b2%10]++;
      b2/=10;
    }
    for (int c=1;c<=3;++c)
    {
      a[b3%10]++;
      b3/=10;
    }
    for (int c=1;c<=9;++c)if (a[c]!=1){l=1;break;}
    for (int c=1;c<=9;++c)a[c]=0;
    if (!l){cout <<b*k1 <<" " <<b*k2 <<" " <<b*k3 <<endl;ans++;}//将解输出，并做标记
    else l=0;
  }
  if (!ans)cout <<"No!!!";//判断无解情况
  return 0;
}
```

4 [3328] 珠心算测验

珠心算是一种通过在脑中模拟算盘变化来完成快速运算的一种计算技术。珠心算训练，既能够开发智力，又能够为日常生活带来很多便利，因而在很多学校得到普及。

某学校的珠心算老师采用一种快速考察珠心算加法能力的测验方法。他随机生成一个正整数集合，集合中的数各不相同，然后要求学生回答：其中有多少个数，恰好等于集合中另外两个（不同的）数之和？

最近老师出了一些测验题，请你帮忙求出答案。

输入

共两行，第一行包含一个整数 n ，表示测试题中给出的正整数个数。

第二行有 n 个正整数，每两个正整数之间用一个空格隔开，表示测试题中给出的正整数。

输出

一个整数，表示测验题答案。

15

100 12 88 44 46 2 56 99 23 32 11 47 56 9991 9993

8

样例输入

4
1 2 3 4
样例输出
2

```
#include<iostream>
#include<cstdio>
using namespace std;
int t[200005],g[200005];//t 是桶，t[i]表示值为 i 的数在集合中两两相加出现了几次，g[i]表示值为 i 的数是否在集合中，1 为在，0 为不在
int n,a[105],ans;
int main(){
  cin>>n;
  for (int i=1;i<=n;i++){
    cin>>a[i];//读入
    g[a[i]]=1;//在集合中赋值为 1
  }
  for (int i=1;i<n;i++){//枚举
    for (int j=i+1;j<=n;j++){
      t[a[i]+a[j]]++;//被加出来了
    }
  }
  for (int i=1;i<=200002;i++){
    if (t[i]>0&&g[i]) ans++;//判断是否满足，满足 ans++
  }
  cout<<ans<<endl;
  return 0;
}
```

5 [3344] 比例简化

在社交媒体上，经常会看到针对某一个观点同意与否的民意调查以及结果。例如，对某一观点表示支持的有 1498 人，反对的有 902 人，那么赞同与反对的比例可以简单的记为 1498:902 。
不过，如果把调查结果就以这种方式呈现出来，大多数人肯定不会满意。因为这个比例的数值太大，难以一眼看出它们的关系。对于上面这个例子，如果把比例记为 5:3 ，虽然与真实结果有一定的误差，但依然能够较为准确地反映调查结果，同时也显得比较直观。

在社交媒体上，经常会看到针对某一个观点同意与否的民意调查以及结果。例如，对某一观点表示支持的有 1498 人，反对的有 902 人，那么赞同与反对的比例可以简单的记为 $1498:902$ 。

不过，如果把调查结果就以这种方式呈现出来，大多数人肯定不会满意。因为这个比例的数值太大，难以一眼看出它们的关系。对于上面这个例子，如果把比例记为 $5:3$ ，虽然与真实结果有一定的误差，但依然能够较为准确地反映调查结果，同时也显得比较直观。

现给出支持人数A，反对人数 B ，以及一个上限 L ，请你将 A 比 B 化简为 A' 比 B' ，要求在 A' 和 B' 均不大于 L 且 A' 和 B' 互质（两个整数的最大公约数是 1 ）的前提下， $A'/B' \geq A/B$ 且 $A'/B'-A/B$ 的值尽可能小。

输入共一行，包含三个整数 A,B,L，每两个整数之间用一个空格隔开，分别表示支持人数、反对人数以及上限。
输出共一行，包含两个整数 A， B，中间用一个空格隔开，表示化简后的比例。
样例输入 1498 902 10
样例输出 5 3

```
#include<iostream>
using namespace std;
int main(){
  int a,b,c,d,n;//a,b 表示输入进来的，c,d 表示题目中的 a',b'，也就是上文中的 i,j，n 就是上限啦
  cin>>a>>b>>n;//输入
  int ans1=n,ans2=1;//将 ans 赋值为不超过上限的最大值
  c=d=1;//初值为 1
  while(c<=n&&d<=n){//不能超过上限
    if(a*d<=b*c){//如果大于等于（用上面推出的公式）
      if(c*ans2<d*ans1){//看是否能更新答案（还是用上面推出的公式）
        ans1=c;//更新
        ans2=d;
      }
      d++;//分母加 1
    }else{//如果小于
      c++;//分子加 1
    }
  }
  cout<<ans1<<' '<<ans2<<endl;//输出答案
  return 0;
}
```

课前练习

#include <iostream> using namespace std; int main() {	#include <iostream> #include <string> #include <cmath> using namespace std;

```
    int i, j, max;
    int a[9], s[9];
    max = -32765;
    for (i = 0; i <= 8; i++)
        s[i] = 0;
    for (i = 1; i <= 8; i++)
    {
        cin >> a[i];
        s[i] = s[i - 1] + a[i];
    }
    for (i = 0; i <= 7; i++)
        for (j = i + 1; j <= 8; j++)
        {
            if ((s[j] - s[i]) > max)
                max = s[j] - s[i];
        }
    cout << max << " ";
    for (i = 1; i <= 7; i++)
        cout << s[i] << " ";
    return 0;
}
```

输入：

46 36 46 11 28 28 21 18

输出：_____

```
int main()
{
    string n;
    int i, j, d;
    double s, t;
    char m;
    cin >> n;
    cin >> m;
    j = 1;
    s = 0;
    for (i = n.length() - 1; i >= 0; i--)
    {
        switch (n[i])
        {
        case '2':
            d = 2;
            break;
        case '1':
            d = 1;
            break;
        case '0':
            d = 0;
            break;
        }
        s = s + d * j;
        j = j * 3;
    }
    t = m - 'A';
    s = s - t;
    s = sqrt(s);
    cout << s << endl;
    return 0;
}
```

输入：

1201001 S

输出：_____

习题

1.

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

```
    int a, b, u, i, num;
    cin>>a>>b>>u; num = 0;//统计在 a~b 中有多少可以整除 u 的数
    for (i = a; i <= b; i++) if ((i % u) == 0)
        num++;
    cout<<num<<endl; return 0;
}
```
输入： 1 100 15
输出： _____

2.
```
#include <iostream>
using namespace std;
int main()//题意：查找数 f 在数组 a 中的位置
{
    const int SIZE = 100;
    int n, f, i, left, right, middle, a[SIZE];
    cin>>n>>f;
    for (i = 1; i <= n; i++)
        cin>>a[i]; left = 1;
    right = n;
    do {
        middle = (left + right) / 2;
        if (f <= a[middle])
            right = middle;
        else
            left = middle + 1;
            } while (left < right);
    cout<<left<<endl;
    return 0;
}
```
输入：
12 17
2 4 6 9 11 15 17 18 19 20 21 25
输出： _____

3.
```
#include <iostream>
using namespace std;
int fun(int n)
{
  if(n == 1)
    return 1;
  if(n == 2)
    return 2;
```

```
    return fun(n -2) - fun(n - 1);
    }
int main()
{
    int n;
    cin >> n;
    cout << fun(n) << endl;
     return 0;
    }
```

输入：7

输出：_____

第十五讲 贪心初步

15.1 理解贪心

所谓贪心算法是指，在对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，他所做出的仅是在某种意义上的局部最优解。

[例 1]在 N 行 M 列的正整数矩阵中，要求从每行中选出 1 个数，使得选出的总共 N 个数的和最大。

算法分析]

要使总和最大，则每个数要尽可能大，自然应该选每行中最大的那个数。因此，我们设计出如下算法:

$$\begin{bmatrix} 5 & 6 & 19 & 23 & 56 \\ 32 & 123 & 5 & 1 & 23 \\ 66 & 1 & 23 & 567 & 2 \\ 100 & 23 & 4 & 231 & 3 \end{bmatrix}$$

```
读入 N, M,矩阵数据；
total = 0;
for (int i = 1;    i <= N; i++ )
{                                        //对 N 行进行选择
    选择第 I 行最大的数,记为 K；
    Total += K；
}
```

贪心算法没有固定的算法框架，算法设计的关键是贪心策略的选择，贪心策略使用的前提是局部最优能导致全局最优。必须注意的是，贪心算法不是对所有问题都能得到整体最优解，选择的贪心策略必须具备无后效性，即某个状态以后的过程不会影响以前的状态，只与当前状态有关。所以对所采用的贪心策略一定要仔细分析其是否满足无后效性。

利用贪心算法求解的问题往往具有两个重要的特性：贪心选择性质和最优子结构性质。如果满足这两个性质就可以使用贪心算法了。

（1）贪心选择

贪心选择是指原问题的整体最优解可以通过一系列局部最优的选择得到。应用同一规则，将原问题变为一个相似的但规模更小的子问题，而后的每一步都是当前最佳的选择。这种选择依赖于已做出的选择，但不依赖于未做出的选择。运用贪心策略解决的问题在程序的运行过程中无回溯过程。

例如，挑选苹果，如果你认为个大的是最好的，那你每次都从苹果堆中拿一个最大的，作为局部最优解，贪心策略就是选择当前最大的苹果；如果你认为最红的苹果是最好的，那你每次都从苹果堆中拿一个最红的，贪心策略就是选择当前最红的苹果。因此根据求解目标不同，贪心策略也会不同。

（2）最优子结构

当一个问题的最优解包含其子问题的最优解时，称此问题具有最优子结构性质。问题的最优子结构性质是该问题是否可用贪心算法求解的关键。例如原问题

S={a1，a2，… ，ai，… ，an}，通过贪心选择选出一个当前最优解{ai}之后，转化为求解子问题 S−{ai}，如果原问题的最优解包含子问题的最优解，则说明该问题满足最优子结构性质。

15.2 案例分析

1 [3867]最优装载问题

在北美洲东南部，有一片神秘的海域，那里碧海蓝天、阳光明媚，这正是传说中海盗最活跃的加勒比海（Caribbean Sea）。17 世纪时，这里更是欧洲大陆的商旅舰队到达美洲的必经之地，所以当时的海盗活动非常猖獗，海盗不仅攻击过往商人，甚至攻击英国皇家舰……

有一天，海盗们截获了一艘装满各种各样古董的货船，每一件古董都价值连城，一旦打碎就失去了它的价值。虽然海盗船足够大，但载重量为 C，每件古董的重量为 wi，海盗们该如何把尽可能多数量的宝贝装上海盗船呢？

输入第一行是一个整型数 m(m<100)表示共有 m 组测试数据。每组测试数据的第一行是两个整数 c,n(1<c,n<10000)表示该测试数据载重量 c 及古董的个数 n。

输出对于每一组输入，输出能装入的古董最大数量。每组的输出占一行

样例输入

2

30 8

4 10 7 11 3 5 14 2

45 10

5 12 7 3 20 9 15 11 8 32

样例输出

5

6

1 问题分析

根据问题描述可知这是一个可以用贪心算法求解的最优装载问题，要求装载的物品的数量尽可能多，而船的容量是固定的，那么**优先把重量小的物品放进去**，在容量固定的情况下，装的物品最多。**采用重量最轻者先装的贪心选择策略，从局部最优达到全局最优**，从而产生最优装载问题的最优解。

（1）当载重量为定值 c 时，wi 越小时，可装载的古董数量 n 越大。只要依次选择最小重量古董，直到不能再装为止。

（2）把 n 个古董的重量从小到大（非递减）排序，然后根据贪心策略尽可能多地选出前 i 个古董，直到不能继续装为止，此时达到最优。

2 图解说明

表 1 古董重量清单

重量	w[i]	4	10	7	11	3	5	14	2

每个古董的重量如表 1 所示，海盗船的载重量 c 为 30，那么在不能打碎古董又不超过载重的情况下，怎么装入最多的古董？

（1）因为贪心策略是每次选择重量最小的古董装入海盗船，因此可以按照古董重量非递减排序，排序后如表 2 所示。

表 2 按重量排序后古董清单

重量	w[i]	2	3	4	5	7	10	11	14

（2）按照贪心策略，每次选择重量最小的古董放入（tmp 代表古董的重量，ans 代表已

装载的古董个数）。
i=0，选择排序后的第 1 个，装入重量 tmp=2，不超过载重量 30，ans =1。
i=1，选择排序后的第 2 个，装入重量 tmp=2+3=5，不超过载重量 30，ans =2。
i=2，选择排序后的第 3 个，装入重量 tmp=5+4=9，不超过载重量 30，ans =3。
i=3，选择排序后的第 4 个，装入重量 tmp=9+5=14，不超过载重量 30，ans =4。
i=4，选择排序后的第 5 个，装入重量 tmp=14+7=21，不超过载重量 30，ans =5。
i=5，选择排序后的第 6 个，装入重量 tmp=21+10=31，超过载重量 30，算法结束。
即放入古董的个数为 ans=5 个

3 代码详解

（1）数据结构定义

根据算法设计描述，我们用一维数组存储古董的重量：

double w[N]; //一维数组存储古董的重量

（2）按重量排序

可以利用 C++中的排序函数 sort，对古董的重量进行从小到大（非递减）排序。要使用此函数需引入头文件：

#include <algorithm>

（3）按照贪心策略找最优解

首先用变量 ans 记录已经装载的古董个数，tmp 代表装载到船上的古董的重量，两个变量都初始化为 0。然后按照重量从小到大排序，依次检查每个古董，tmp 加上该古董的重量，如果小于等于载重量 c，则令 ans ++；否则，退出。

```
int tmp = 0,ans = 0; //tmp 代表装载到船上的古董的重量，ans 记录已经装载的古董个数
for(int i=0;i<n;i++) {
   tmp += w[i];
   if(tmp<=c) ans ++;
   else
   break;
}
```

```
#include <iostream>
#include <algorithm>
const int N=1000005;
using namespace std;
double w[N]; //古董的重量数组
int main()
{
      double c;
      int n;
      cin>>c>>n;//请输入载重量 c 及古董个数 n
      for(int i=0;i<n;i++)
         cin>>w[i]; //输入输入每个古董的重量
      1                          //按古董重量升序排序
```

```
    double tmp=0.0;
    int ans=0; // tmp 为已装载到船上的古董重量，ans 为已装载的古董个数
    for(int i=0;i<n;i++)
    {
       tmp+=w[i];
       if(__2_____)
          ans++;
       else
          __3_____
    }
    cout<<ans<<endl;//能装入的古董最大数量 Ans
    return 0;
}
```

2 [3868]阿里巴巴与四十大盗

有一天，阿里巴巴赶着一头毛驴上山砍柴。砍好柴准备下山时，远处突然出现一股烟尘，弥漫着直向上空飞扬，朝他这儿卷过来，而且越来越近。靠近以后，他才看清原来是一支马队，他们共有四十人，一个个年轻力壮、行动敏捷。一个首领模样的人背负沉重的鞍袋，从丛林中一直来到那个大石头跟前，喃喃地说道："芝麻，开门吧！"随着那个头目的喊声，大石头前突然出现一道宽阔的门路，于是强盗们鱼贯而入。阿里巴巴待在树上观察他们，直到他们走得无影无踪之后，才从树上下来。他大声喊道："芝麻，开门吧！"他的喊声刚落，洞门立刻打开了。他小心翼翼地走了进去，一下子惊呆了，洞中堆满了财物，还有多得无法计数的金银珠宝，有的散堆在地上，有的盛在皮袋中。突然看见这么多的金银财富，阿里巴巴深信这肯定是一个强盗们数代经营、掠夺所积累起来的宝窟。为了让乡亲们开开眼界，见识一下这些宝物，他想一种宝物只拿一个，如果太重就用锤子凿开，但毛驴的运载能力是有限的，怎么才能用驴子运走最大价值的财宝分给穷人呢？

阿里巴巴陷入沉思中……

输入每组测试数据的第一行是两个整数 n,c (1<n,c<10000)表示该测试数据宝物数量及驴子的承载重量。随后的 n 行，每行有两个正整数 wi, vi 分别表示第 i 个宝物的重量和价值(1<wi,vi<100)。

输出对于每一组输入，输出毛驴运走宝物的最大价值。每组的输出占一行,结果保留一位小数

样例输入

```
6 19
2 8
6 1
7 9
4 3
10 2
3 4
```

样例输出

```
24.6
```

1 问题分析

假设山洞中有 n 种宝物，每种宝物有一定重量 w 和相应的价值 v，毛驴运载能力有限，只能运走 m 重量的宝物，一种宝物只能拿一样，宝物可以分割。那么怎么才能使毛驴运走宝物的价值最大呢？

我们可以尝试贪心策略：

（1）每次挑选价值最大的宝物装入背包，得到的结果是否最优？

（2）每次挑选重量最小的宝物装入，能否得到最优解？

（3）每次选取单位重量价值最大的宝物，能否使价值最高？

思考一下，如果选价值最大的宝物，但重量非常大，也是不行的，因为运载能力是有限的，所以第 1 种策略舍弃；如果选重量最小的物品装入，那么其价值不一定高，所以不能在总重限制的情况下保证价值最大，第 2 种策略舍弃；而**第 3 种是每次选取单位重量价值最大的宝物，也就是说每次选择性价比（价值/重量）最高的宝物，如果可以达到运载重量 m 那么一定能得到价值最大。**因此采用第 3 种贪心策略，每次从剩下的宝物中选择性价比最高的宝物。

（1）数据结构及初始化。将 n 种宝物的重量和价值存储在结构体 three（包含重量、价值、性价比 3 个成员）中，同时求出每种宝物的性价比也存储在对应的结构体 three 中，将其按照性价比从高到低排序。采用 sum 来存储毛驴能够运走的最大价值，初始化为 0。

（2）根据贪心策略，按照性价比从大到小选取宝物，直到达到毛驴的运载能力。每次选择性价比高的物品，判断是否小于 m（毛驴运载能力），如果小于 m，则放入，sum（已放入物品的价值）加上当前宝物的价值，m 减去放入宝物的重量；如果不小于 m，则取该宝物的一部分 m * p[i]，m=0，程序结束。m 减少到 0，则 sum 得到最大值。

2 图解说明

假设现在有一批宝物，价值和重量如表 3 所示，毛驴运载能力 m=30，那么怎么装入最大价值的物品？

表 3 宝物清单

宝物	i	1	2	3	4	5	6	7	8	9	10
重量	w[i]	4	2	9	5	5	8	5	4	5	5
价值	v[i]	3	8	18	6	8	20	5	6	7	15

（1）因为贪心策略是每次选择性价比（价值/重量）高的宝物，可以按照性价比降序排序，排序后如表 4 所示。

表 4 排序后宝物清单

宝物	i	2	10	6	3	5	8	9	4	7	1
重量	w[i]	2	5	8	9	5	4	5	5	5	4
价值	v[i]	8	15	20	18	8	6	7	6	5	3
性价比	p[i]	4	3	2.5	2	1.6	1.5	1.4	1.2	1	0.75

（2）按照贪心策略，每次选择性价比高的宝物放入：

第 1 次选择宝物 2，剩余容量 30−2=28，目前装入最大价值为 8。

第 2 次选择宝物 10，剩余容量 28−5=23，目前装入最大价值为 8+15=23。

第 3 次选择宝物 6，剩余容量 23−8=15，目前装入最大价值为 23+20=43。
第 4 次选择宝物 3，剩余容量 15−9=6，目前装入最大价值为 43+18=61
第 5 次选择宝物 5，剩余容量 6−5=1，目前装入最大价值为 61+8=69。
第 6 次选择宝物 8，发现上次处理完时剩余容量为 1，而 8 号宝物重量为 4，无法全部放入，那么可以采用部分装入的形式，装入 1 个重量单位，因为 8 号宝物的单位重量价值为 1.5，因此放入价值 1×1.5=1.5，你也可以认为装入了 8 号宝物的 1/4，目前装入最大价值为 69+1.5=70.5，剩余容量为 0。
（3）构造最优解

把这些放入的宝物序号组合在一起，就得到了最优解（2，10，6，3，5，8），其中最后一个宝物为部分装入（装了 8 号财宝的 1/4），能够装入宝物的最大价值为 70.5。

3 代码详解

（1）数据结构定义

根据算法设计中的数据结构，我们首先定义一个结构体 three：

```
struct three{
    double w; //每种宝物的重量 double v; //每种宝物的价值
    double p; //每种宝物的性价比（价值/重量）
};
```

（2）性价比排序

我们可以利用 C++中的排序函数 sort，对宝物的性价比从大到小（非递增）排序。

```
bool cmp(three a,three b)//比较函数按照宝物性价比降序排列 {
    return a.p > b.p; //指明按照宝物性价比降序排列
 }
sort(s, s+n, cmp);
```

（3）贪心算法求解

在性价比排序的基础上，进行贪心算法运算。如果剩余容量比当前宝物的重量大，则可以放入，剩余容量减去当前宝物的重量，已放入物品的价值加上当前宝物的价值。如果剩余容量比当前宝物的重量小，表示不可以全部放入，可以切割下来一部分（正好是剩余容量），然后令剩余容量乘以当前物品的单位重量价值，已放入物品的价值加上该价值，即为能放入宝物的最大价值。

```
for(int i = 0;i < n;i++)//按照排好的顺序，执行贪心策略
{
  if( m > s[i].w )//如果宝物的重量小于毛驴剩下的运载能力，即剩余容量
  {
    m -= s[i].w;
    sum += s[i].v;
  }
  else //如果宝物的重量大于毛驴剩下的承载能力
  {
    sum += m*s[i].p; //进行宝物切割，切割一部分(m 重量)，正好达到驴子承重
    break;
  }
```

```
}

#include<bits/stdc++.h>
using namespace std;
struct bao_wu{
        int w;//每个宝物的重量
        int v;//每个宝物的价值
        double p;//性价比
}bw[10001];
int cmp(__1__){
        __2__//根据性价比从大到小排序
}
int main(){
        int n,c;
        double sum=0;
        cin>>n>>c;
        for(int i=0;i<n;i++){
                cin>>bw[i].w>>bw[i].v;//输入
                __3__//计算性价比
        } //输入数据，初始化
        __4__ //对决策参数（性价比）作排序操作
        for(int i=0;i<n;i++){
                if(bw[i].w<=c){
                        __5__ //获得价值
                        __6__ //装入背包，重量减小
                }
                else{//没法把当前物品整个装入，进行分割
                        __7__
                          break;
                }
        }
        printf("%.1lf",sum);
        return 0;
}
```

4 进一步分析

想一想，**如果宝物不可分割，贪心算法是否能得到最优解？**下面我们看一个简单的例子。

假定物品的重量和价值已知，如表 5 所示，最大运载能力为 10。采用贪心算法会得到怎样的结果？

表 5 物品清单

物品	i	1	2	3	4	5
重量	w[i]	3	4	6	10	7
价值	v[i]	15	16	18	25	14

性价比	p[i]	5	4	3	2.5	2

如果我们采用贪心算法，先装性价比高的物品，且物品不能分割，剩余容量如果无法再装入剩余的物品，不管还有没有运载能力，算法都会结束。那么我们选择的物品为 1 和 2，总价值为 31，而实际上还有 3 个剩余容量，但不足以装下剩余其他物品，因此得到的最大值为 31。但实际上我们如果选择物品 2 和 3，正好达到运载能力，得到的最大价值为 34。也就是说，在物品不可分割、没法装满的情况下，贪心算法并不能得到最优解，仅仅是最优解的近似解。

想一想，为什么会这样呢？

物品可分割的装载问题我们称为背包问题，物品不可分割的装载问题我们称之为 0-1 背包问题。在物品不可分割的情况下，即 0-1 背包问题，已经不具有贪心选择性质，原问题的整体最优解无法通过一系列局部最优的选择得到，因此这类问题得到的是近似解。如果一个问题不要求得到最优解，而只需要一个最优解的近似解，则不管该问题有没有贪心选择性质都可以使用贪心算法。

3 [3407] 看电视-会议安排

暑假到了，小明终于可以开心的看电视了。但是小明喜欢的节目太多了，他希望尽量多的看到完整的节目。

现在他把他喜欢的电视节目的转播时间表给你，你能帮他合理安排吗？

输入包含多组测试数据。每组输入的第一行是一个整数 n（n<=100），表示小明喜欢的节目的总数。

接下来 n 行，每行输入两个整数 si 和 ei（1<=i<=n），表示第 i 个节目的开始和结束时间，为了简化问题，每个时间都用一个正整数表示。

当 n=0 时，输入结束。

输出

对于每组输入，输出能完整看到的电视节目的个数。

样例输入

```
12
1 3
3 4
0 7
3 8
15 19
15 20
10 15
8 18
6 12
5 10
4 14
2 9
0
```

样例输出

```
5
```

1 问题分析

这是一个典型的会议安排问题，**会议安排的目的是能在有限的时间内召开更多的会议（任何两个会议不能同时进行）**。在会议安排中，每个会议 i 都有起始时间 bi 和结束时间 ei，且 bi<ei，即一个会议进行的时间为半开区间[bi，ei)。如果[bi，ei)与[bj，ej)均在"有限的时间内"，且不相交，则称会议 i 与会议 j 相容的。也就是说，当 bi≥ej 或 bj≥ei 时，会议 i 与会议 j 相容。会议安排问题要求在所给的会议集合中选出最大的相容活动子集，即尽可能在有限的时间内召开更多的会议。

在这个问题中，"有限的时间内（这段时间应该是连续的）"是其中的一个限制条件，也应该是有一个起始时间和一个结束时间（简单化，起始时间可以是会议最早开始的时间，结束时间可以是会议最晚结束的时间），任务就是实现召开更多的满足在这个"有限的时间内"等待安排的会议，会议时间表如表 6 所示。

表 6 会议时间表

会议	i	1	2	3	4	5	6	7	8	9	10
开始时间	bi	8	9	10	11	13	14	15	17	18	16
结束时间	ei	10	11	15	14	16	17	17	18	20	19

会议安排的时间段如图所示。

从表 6 中可以看出，{会议 1，会议 4，会议 6，会议 8，会议 9}，{会议 2，会议 4，会议 7，会议 8，会议 9}都是能安排最多的会议集合。

要让会议数最多，我们需要选择最多的不相交时间段。我们可以**尝试贪心策略**：

（1）每次从剩下未安排的会议中选择会议具有最早开始时间且与已安排的会议相容的会议安排，以增大时间资源的利用率。

（2）每次从剩下未安排的会议中选择持续时间最短且与已安排的会议相容的会议安排，这样可以安排更多一些的会议。

（3）每次从剩下未安排的会议中选择具有最早结束时间且与已安排的会议相容的会议安排，这样可以尽快安排下一个会议。

思考一下，如果选择最早开始时间，则如果会议持续时间很长，例如 8 点开始，却要持续 12 个小时，这样一天就只能安排一个会议；如果选择持续时间最短，则可能开始时间很晚，例如 19 点开始，20 点结束，这样也只能安排一个会议，所以我们最好选择那些开始时间要早，而且持续时间短的会议，即最早开始时间+持续时间最短，就是最早结束时间。

因此采用第（3）种贪心策略，每次从剩下的会议中选择具有最早结束时间且与已安排的会议相容的会议安排。

2 算法设计

（1）初始化：将 n 个会议的开始时间、结束时间存放在结构体数组中，如果需要知道选中了哪些会议，还需要在结构体中增加会议编号，然后按结束时间从小到大排序（非递减），结束时间相等时，按开始时间从大到小排序（非递增）；

（2）根据贪心策略就是选择第一个具有最早结束时间的会议，用 last 记录刚选中会议的结束时间；

（3）选择第一个会议之后，依次从剩下未安排的会议中选择，如果会议 i 开始时间大于等于最后一个选中的会议的结束时间 last，那么会议 i 与已选中的会议相容，可以安排，更新 last 为刚选中会议的结束时间；否则，舍弃会议 i，检查下一个会议是否可以安排。

(1)原始的会议时间表（见表 7）：

表 7 原始会议时间表

会议	num	1	2	3	4	5	6	7	8	9	10
开始时间	beg	3	1	5	2	5	3	8	6	8	12
结束时间	end	6	4	7	5	9	8	11	10	12	14

(1) 排序后的会议时间表（见表 8）：

表 8 排序后的会议时间表

会议	num	2	4	1	3	6	5	8	7	9	10
开始时间	beg	1	2	3	5	3	5	6	8	8	12
结束时间	end	4	5	6	7	8	9	10	11	12	14

(2) 贪心选择过程

（1）首先选择排序后的第一个会议即最早结束的会议（编号为 2），用 last 记录最后一个被选中会议的结束时间，last=4。

（2）检查余下的会议，找到第一个开始时间大于等于 last（last=4）的会议，子问题转化为从该会议开始，余下的所有会议。如表 8 所示。

表 9 会议时间表

会议	num	2	4	1	3	6	5	8	7	9	10
开始时间	beg	1	2	3	5	3	5	6	8	8	12
结束时间	end	4	5	6	7	8	9	10	11	12	14

从子问题中，选择第一个会议即最早结束的会议（编号为 3），更新 last 为刚选中会议的结束时间 last=7。

（3）检查余下的会议，找到第一个开始时间大于等于 last（last=7）的会议，子问题转化为从该会议开始，余下的所有会议。如表 9 所示。

表 10 会议时间表

会议	num	2	4	1	3	6	5	8	7	9	10
开始时间	beg	1	2	3	5	3	5	6	8	8	12
结束时间	end	4	5	6	7	8	9	10	11	12	14

从子问题中，选择第一个会议即最早结束的会议（编号为 7），更新 last 为刚选中会议的结束时间 last=10。

（4）检查余下的会议，找到第一个开始时间大于等于 last（last=11）的会议，子问题转化为从该会议开始，余下的所有会议。如表 11 所示。

表 11 会议时间表

会议	num	2	4	1	3	6	5	8	7	9	10
开始时间	beg	1	2	3	5	3	5	6	8	8	12
结束时间	end	4	5	6	7	8	9	10	11	12	14

从子问题中，选择第一个会议即最早结束的会议（编号为 10），更新 last 为刚选中会议的结束时间 last=14；所有会议检查完毕，算法结束。如表 11 所示。

4．构造最优解

从贪心选择的结果，可以看出，被选中的会议编号为{2，3，7，10}，可以安排的会议数量最多为 4，如表 12 所示。

4 代码详解

（1）数据结构定义

以下 C++程序代码中，结构体 meet 中定义了 beg 表示会议的开始时间，end 表示会议的结束时间，会议 meet 的数据结构：

```
struct Meet {
    int beg; //会议的开始时间
    int end; //会议的结束时间
} meet[1000];
```

（2）对会议按照结束时间非递减排序

我们采用 C++中自带的 sort 函数，自定义比较函数的办法，实现会议排序，按结束时间从小到大排序（非递减），结束时间相等时，按开始时间从大到小排序（非递增）：

```
bool cmp(Meet x,Meet y) {
  if(x.end==y.end) //结束时间相等时
  return x.beg>y.beg; //按开始时间从大到小排序
return x.end<y.end; //按结束时间从小到大排序
}
sort(meet,meet+n,cmp);
```

（3）会议安排问题的贪心算法求解

在会议按结束时间非递减排序的基础上，首先选中第一个会议，用 last 变量记录刚刚被选中会议的结束时间。下一个会议的开始时间与 last 比较，如果大于等于 last，则 选中。每次选中一个会议，更新 last 为最后一个被选中会议的结束时间，被选中的会议数 ans 加 1；如果会议的开始时间不大于等于 last，继续考查下一个会议，直到所有会议考查完毕。

```
int ans=1; //用来记录可以安排会议的个数，初始时选中了第一个会议
int last = meet[0].end; //last 记录第一个会议的结束时间
for( i = 1;i < n; i++) //依次检查每个会议 {
   if(meet[i].beg > =last)
   { //如果会议 i 开始时间大于等于最后一个选中的会议的结束时间
      ans++;
      last = meet[i].end; //更新 last 为最后一个选中会议的结束时间
   }
}
return ans; //返回可以安排的会议最大数
```

上面介绍的程序中，只是返回了可以安排的会议最大数，而不知道安排了哪些会议，这显然是不满足需要的。我们可以改进一下，在会议结构体 meet 中添加会议编号 num 变量，选中会议时，显示选中了第几个会议。

```
#include<bits/stdc++.h>
```

```
using namespace std;
struct meeting{
   int s;
   int e;
}meet[101];

int cmp(meeting a, meeting b){
   return a.e<b.e;
}

int main(){
   int last,n;
   int sum;
   while(cin>>n){
      if(n==0)break;
      for(int i=0;i<n;i++){
         cin>>meet[i].s>>meet[i].e;
      }//输入数据
      sort(meet,meet+n,cmp);
      sum=1;//选择第一个会议
      last=meet[0].e;//当前会议的结束时间
      for(int j=1;j<n;j++){//扫描除第一个会议外的所有会议
         if(meet[j].s>=last){
            sum++;
            last=meet[j].e;//更新 last
         }
      }
      cout<<sum<<endl;
   }

   return 0;
}
```

课堂作业列表

3867　最优装载问题
3868　阿里巴巴与四十大盗
3407　会议安排
1912　硕鼠
6990　购买贺年卡(card)
2349　贪心的小猫咪

课前练习

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string map= "2223334445556667778889999";
    string tel;
    int i;
    cin>>tel;
    for(i=0;i<tel.length();i++)
       if((tel[i]>='0') && (tel[i]<='9') )
           cout<<tel[i];
       else if( (tel[i]>='A') && (tel[i]<='Z'))
           cout<<map[tel[i]-'A'];
    cout<<endl;
    return 0;
}
```

输入：CCF-NOIP-2011

输出：_____

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string a, b;
    int i;
    a = "AABBCCDKKRRSSXX";
    cin >> b;
    for (i = 0; i < b.length(); ++i)
    {
        if ((b[i] >= '0') && (b[i] <= '9'))
            cout << b[i];
        else
        {
            if ((b[i] >= 'A') && (b[i] <= 'Z'))
                cout << a[b[i] - 'A' - 1];
        }
    }
    return 0;
}
```

输入：NOIP-2012

输出：_____

贪心习题

1. 执行以下程序段后，k 的值是____

```
int a[7],b[4],i,k=0;
for( i=0;i<7;i++ )  a[i]=i*i;
for( i=0;i<4;i++ )
{   b[i]=a[i*(i-1)];
        k=k+b[i];
}
```

2.[2016]

```
#include <iostream>
using namespace std;
int main() {
    int i = 100, x = 0, y = 0;
    while (i > 0) {
        i--;
        x = i % 8;
     if (x == 1) y++;
    }
    cout << y << endl;
    return 0;
}
```

输出：_____

3.[2015]

```
#include <iostream>
using namespace std;
int main() {
    int a,b,c;
    a=1; b=2; c=3;
    if (a>c){
        if(a>c)
        cout << a <<" ";
        else
        cout << b <<" ";
    }
    cout << c << endl;
    return 0;
}
```

输出：_____

4. 阿里巴巴填空

```
#include<iostream>
#include<algorithm>
using namespace std;
struct bao_wu{
        1
}bw[10001];
int cmp(bao_wu a, bao_wu b){
    return a.p>b.p;
}
int main(){
    int n,c;
    double sum=0;
    cin>>n>>c;
    for(int i=0;i<n;i++){
        cin>>bw[i].w>>bw[i].v;
        2                                ;
    } //输入数据，初始化
    3                                ;
        for(int i=0;i<n;i++){
        if(    4              ){
            sum=sum+bw[i].v; //获得价值
            5                                //装入背包，重量减小
        }
        else{//没法把当前物品整个装入，进行分割
            6
            7
        }
    }
    printf("%.1lf",sum);



    return 0;
}
```