

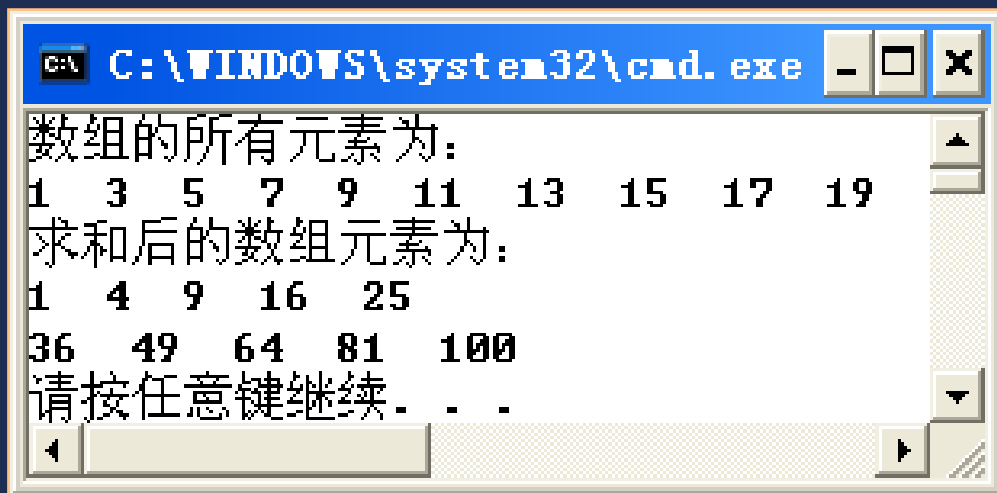
# 实验十三

1、求数组元素和值。

(1)主函数定义整型数组a[10]，数组元素值 $a[j]=2*j+1$ ，输出a数组；

(2)编一子函数，将主函数传递来的数组元素值改变为其前面所有数组元素的和值（包括该数组元素自身值），子函数头要求为sum(int a[],int n)，n用于传递数组的大小；

(3)主函数中输出改变后的a数组。



```
C:\WINDOWS\system32\cmd.exe
数组的所有元素为:
1 3 5 7 9 11 13 15 17 19
求和后的数组元素为:
1 4 9 16 25
36 49 64 81 100
请按任意键继续...
```

# 实验十三

## 2.求自然数对。

1)编写一子函数，判断两个自然数 $x,y$ 是否是自然数对(所谓自然数对是指两个自然数的和与差都是平方数，如： $17-8=9, 17+8=25$ )

2)调用子函数在 $0 < x \leq 50, 0 < y \leq 50$ 且 $x \neq y$ 范围内找出全部自然数对。

## 如何定义子函数？

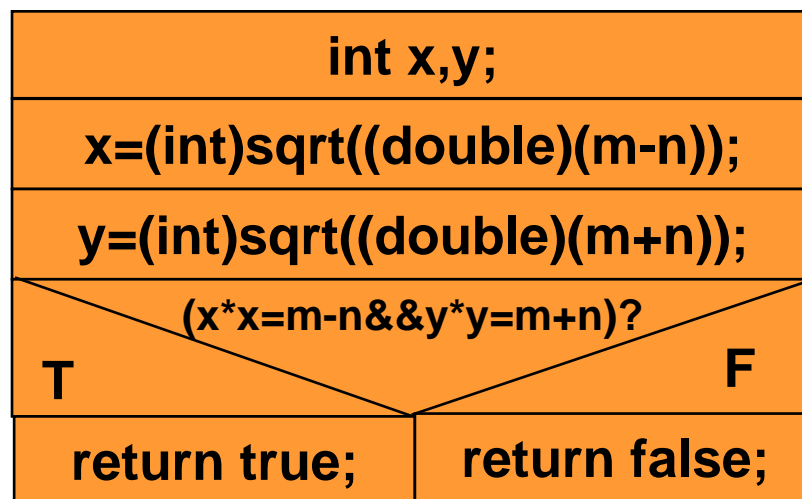
要判断 $x,y$ 是否为自然数对，则函数必须有2个参数 $m,n$ (形参名字不重要，甚至可以省略，但是形参的类型不能省略！)，类型为 **int**。返回类型为**bool**型(是否为自然数对)，即

```
bool pair_of_num(int m, int n) //函数定义
{
    //判断m, n是否为自然数对
    .....
}
```

```

#include<iostream>
#include<cmath>
using namespace std;
void main()
{
int i,j;
_____ ;    //函数定义
for(i=1;i<=50;i++)
    for(j=1;j<=50;j++)
        if( _____ )    //函数调用
            cout<<i<<"和"<<j<<"是自然数对"<<endl;
}

```



子函数算法流程图

# 实验十三

## 3. 斐波那契数列。

1) 主函数求出斐波那契数列的前n项存入数组a[20]中，开始两项均为1，将子函数反序存放后的结果输出；

2) 编写一子函数，将主函数求出的斐波那契数列前n项反序存放，函数头要求为void fbnq(int\*p,int n), p指向数组a。

```
#include<iostream>
using namespace std;
void main()
{
    const int N=20;
    int a[20]={1,1},n,i;
    _____; //函数声明
    cout<<"请输入数组实际长度n: "<<endl;
    cin>>n;
    for(i=2;i<n;i++)          a[i]=a[i-1]+a[i-2];
    _____; //函数调用
    for(i=0;i<n;i++)
        cout<<a[i]<<" ";
    cout<<endl;
}
```

# 子函数算法分析

- 将N个整数按相反的顺序存放并输出。

- 思路分析：

分别取出数组最前面和最后面的元素，进行交换，即a[0]与a[N-1]交换；然后再分别取出a[1]与a[N-2]交换；直到交换完毕。

```
void fbnq(int*p,int n)
{
    //指针p指向数组首地址
}
```

```
int i=0, j=N-1;
for(i=0,j=N-1;i<j;i++,j--)
{
    temp=*(p+i);
    *(p+i)=*(p+j);
    *(p+j)=temp;
}
```

输出交换后数组的各元素

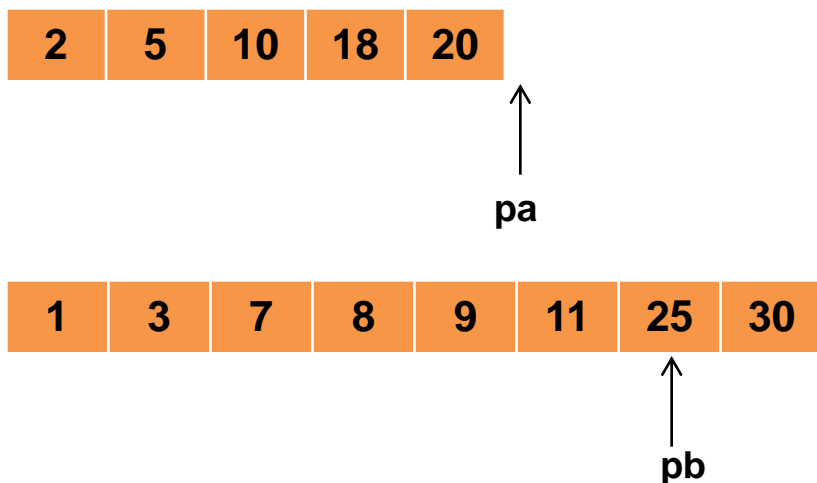
a[i], a[j]如何用指针变量表示  
\*(p+i)      \*(p+j)

# 实验十三

4、编程实现：将有序数组有序合并。

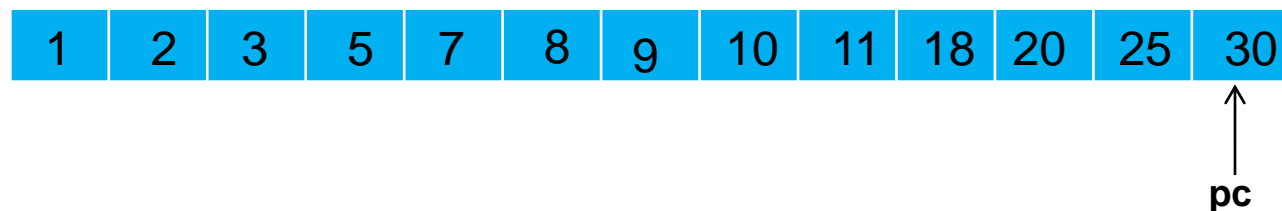
- 1) 有两个从小到大的整型有序数组a和b，编一子函数将这两个数组合并为从小到大的有序数组c，整个合并过程中c保持有序；
- 2) 子函数头要求为**void merge(int a[],int b[],int c[],int m,int n)**，m用来传递a数组元素个数，n传递b数组元素个数。

```
void main()
{
    const int N=20;  int m,n,i;  int a[N],b[N],c[2*N];
    _____; //函数声明
    cout<<"请输入数组a,b的实际长度: "<<endl;
    cin>>m>>n;
    cout<<"请输入数组a的所有元素: "<<endl;
    for(i=0;i<m;i++)  cin>>a[i];
    cout<<"请输入数组b的所有元素: "<<endl;
    for(i=0;i<n;i++)  cin>>b[i];
    _____; //函数调用
    for(int i=0;i<m+n;i++) //合并后数组c的长度为a、b长度之和
        cout<<c[i]<<" ";
    cout<<endl;
}
```



**算法思路：**设有有序（值从小到大）的两组数据，将这两组数据有序**合并**

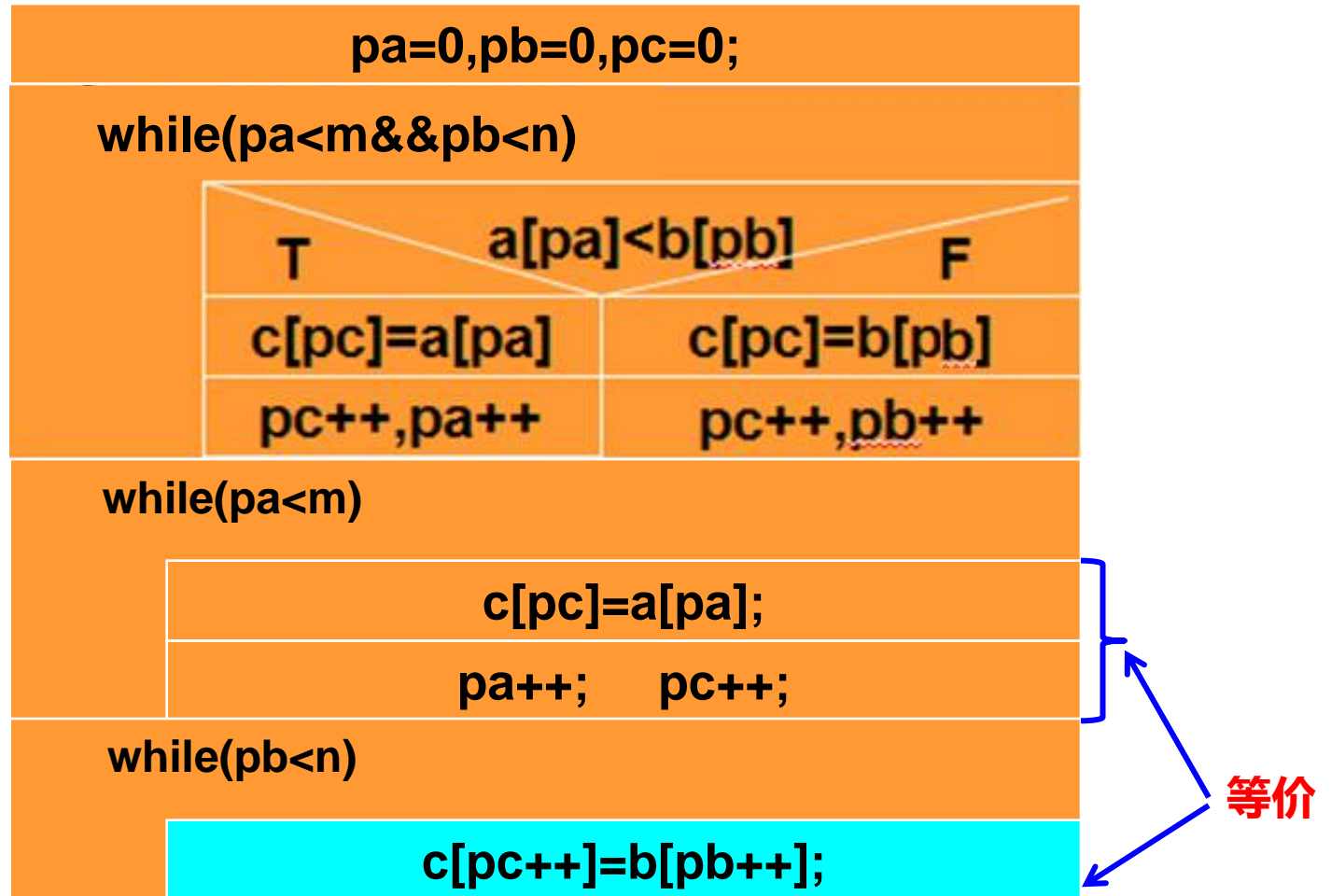
（并集：所有属于集合A或属于集合B的元素组成的集合）



比较 $a[pa]$ 和 $b[pb]$ 的大小，将小的数放到数组 $c[pc]$ 中。

$pa$ ,  $pb$ ,  $pc$ 为元素在数组中的“位置”（下标）。 $pa$ 从0到 $m-1$ ,  $pb$ 从0到 $n-1$ ,  $pc$ 从0到 $m+n-1$ 。

数组中的元素可表示为 $a[pa]$ ,  $b[pb]$ ,  $c[pc]$ 。 $pa++$ ,  $pb++$ ,  $pc++$ 即为位置下标的移动。移动下标即可遍历数组中的所有元素。



pa从0到m-1, pb从0到n-1, pc从0到m+n-1。



**问题拓展：**设有有序（**值从小到大**）的两组数据，将这两组数据有序合并且无重复元素

```
int pa=0,pb=0,pc=0;
while(pa<m&&pb<n)
{
```

```
    if(a[pa]==b[pb])
```

```
    {
```

```
        c[pc++]=a[pa++];
```

```
        pb++;
```

```
    } //如果a、b中的元素相同，则将a的元素写入c，同时移动a,b,c的下标
```

```
else
```

```
{
```

```
    if(a[pa]<b[pb])
```

```
        c[pc++]=a[pa++];
```

```
    else
```

```
        c[pc++]=b[pb++];
```

```
    } //按a、b中元素的大小顺序，将元素写入c中
```

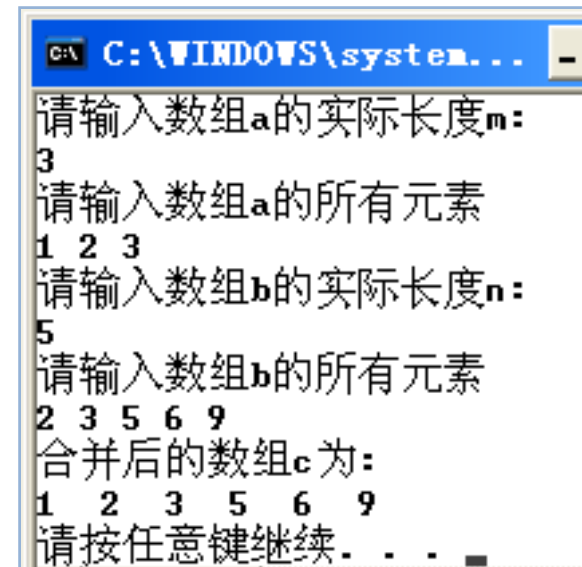
```
}
```

```
while(pa<m)
```

```
    c[pc++]=a[pa++]; //处理a中剩余元素，将其写入c中
```

```
while(pb<n)
```

```
    c[pc++]=b[pb++]; //处理b中剩余元素，将其写入c中
```



## 问题拓展：将两组无序数据进行合并

**分析：**在集合a的基础上实现两个集合的求并，即找出b中不在a中出现的元素，然后写入a中即可。

```
int pa=0,pb=0;
for(pb=0;pb<n;pb++) //处理b中的每一个元素，判断其是否与a有相同元素
{
    for(pa=0;pa<m;pa++)
        if(a[pa]==b[pb])
            break; //如b的当前元素已在集合a中，则处理下一个元素
    if(pa>=m) //此时a中所有元素都和b的当前元素不同，故将其写入c中
        a[m++]=b[pb];
}
cout<<"并集的各个元素依次为:"<<endl;
for(i=0;i<m;i++)    cout<<setw(4)<<a[i];
```

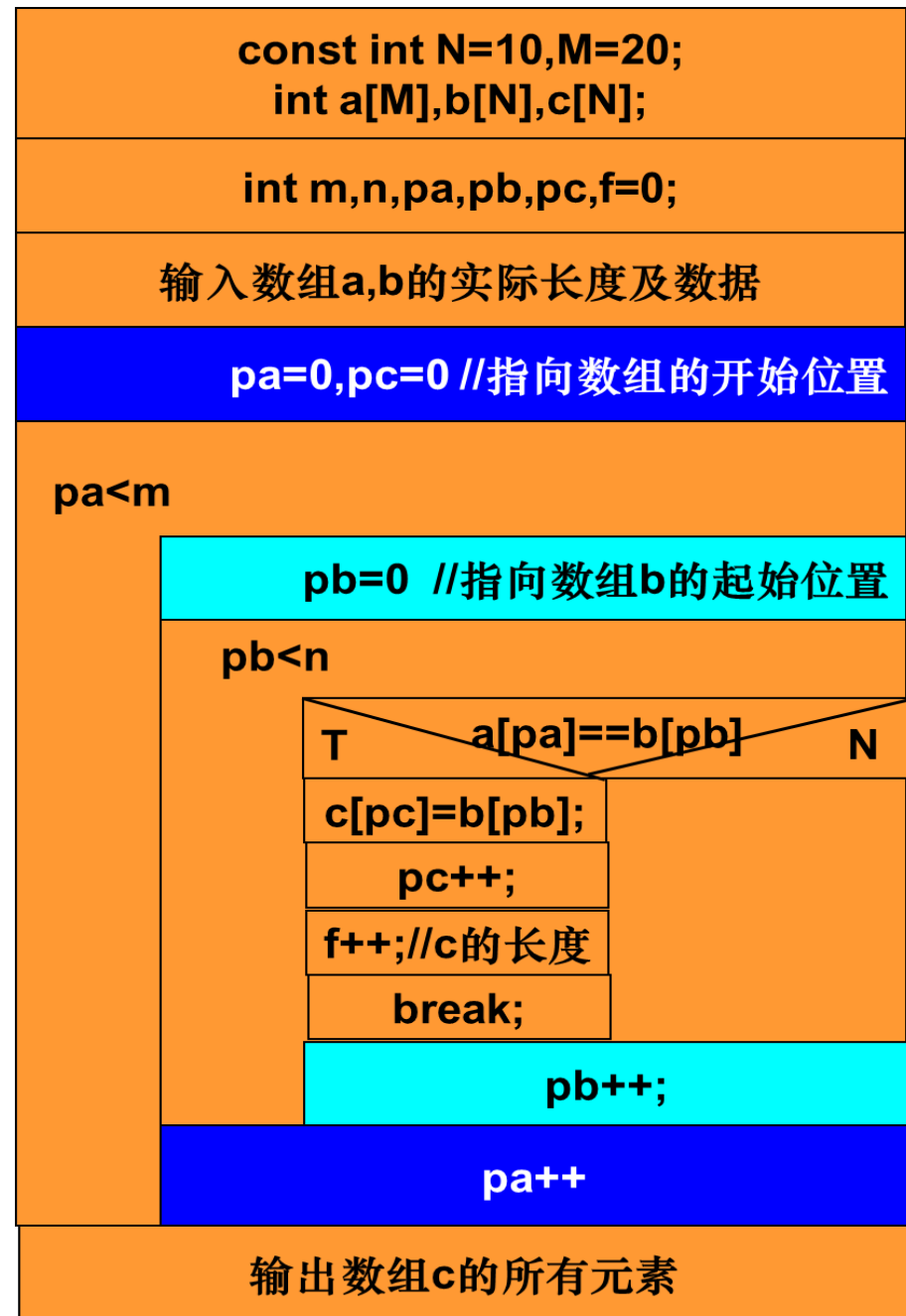
```
C:\WINDOWS\system32\...
请输入数组a的实际长度m:
5
请输入数组a的所有元素
2 1 4 3 5
请输入数组b的实际长度n:
6
请输入数组b的所有元素
3 2 9 7 6 8
合并后的数组c为:
2 1 4 3 5 9 7 6 8
请按任意键继续. . .
```

## 问题拓展：将两组无序数据进行合并

```
for(pb=0;pb<n;pb++) //处理b中的每一个元素, 判断其是否与a有相同元素
{
    flag=0; //设置标志, 如果b中有与a相同的元素, 则flag=1;
    for(pa=0;pa<m;pa++)
    if(a[pa]==b[pb])
    {
        flag=1; //b中若有与a相同的元素, 结束本次处理, 继续判断b的下一个元素
        break;
    }
    if(flag==0) //如果b的元素与a中所有元素均不相同, 则写入a数组的末端
        a[f++]=b[pb];
}
```

**问题拓展：**对两组无序  
数据求交集（既属于A也  
属于B的元素的集合）。

```
for( pa=0,pc=0;pa<m;pa++)  
    for( pb=0;pb<n;pb++)  
        if(a[pa]==b[pb])  
        {  
            c[pc++]=a[pa]; //将a、 b  
            中相同元素写入c,同时下标pc后移  
            f++; //用变量f记录交集数  
            组c的实际长度  
            break; //找到a、 b中相同元  
            素即退出本轮比较  
        }
```



## 问题拓展：对两组有序数据求交集。

```
pa=0,pb=0,pc=0;  
while(pa<m&&pb<n)
```

```
{
```

```
    if(a[pa]==b[pb])
```

```
    {
```

```
        c[pc++]=a[pa++];
```

```
        pb++;
```

```
    } //如果a、b中的元素相同，则将a的元素写入c，同时移动a,b,c的下标
```

```
else
```

```
{
```

```
    if(a[pa]<b[pb])                pa++;
```

```
    else                            pb++;
```

```
} //因a、b中的元素有序，如果不等，则移动对应集合的下标
```

```
}
```

```
cout<<"交集c的各个元素依次为:"<<endl;    //pc记录交集的实际长度
```

```
for (i=0;i<pc;i++)
```

```
    cout<<setw(3)<<c[i];
```

```
}
```

输入数组a中元素的个数:

3

输入数组a的元素:

1 2 3

输入数组b中元素的个数:

5

输入数组b的元素:

2 3 5 6 9

交集c的各个元素依次为:

2 3 请按任意键继续. . .

**问题拓展：**对两组有序数据求差集（属于A而不属于B的元素集合，即A中去掉在B中出现的元素，得到的即为差集）。

for(pa=0;pa<m;pa++) //处理a中的每一个元素，若b中有和a的当前元素相同的，则删掉a的当前元素，最后的差集结果就保存在a中

{

for(pb=0;pb<n;pb++) //考察b中所有元素  
if(a[pa]==b[pb]) //若与a的当前元素相同

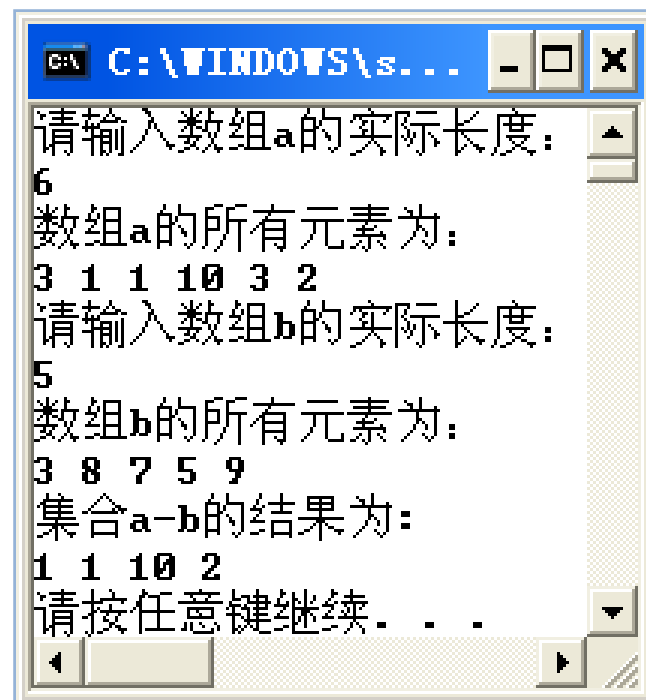
{

for(i=pa; i<m; i++)  
a[i]=a[i+1]; //将a的当前元素删除  
m--; //a的实际长度减1  
pa--; //回溯a的下标

}

}

cout<<endl<<"集合a-b的结果为:"<<endl;  
for(i=0;i<m;i++) cout<<a[i]<<" ";  
cout<<endl;



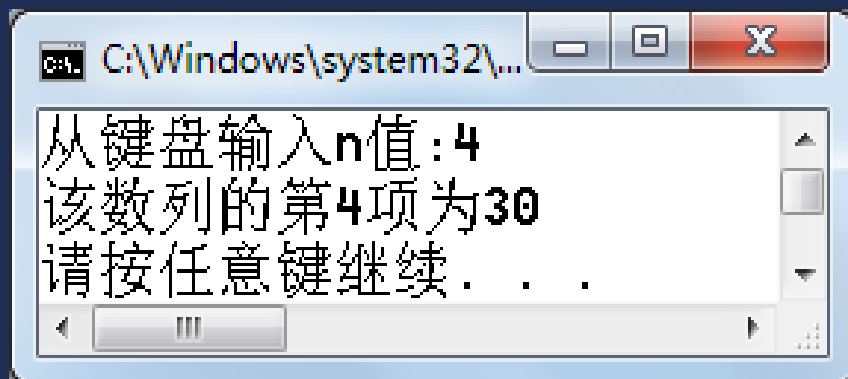
# 实验十三

2、实现功能：已知数列2, 6, 14, 30, ....., 计算该数列的第n项。其中的递推关系为：
$$f(n) = \begin{cases} 2 & (n = 1) \\ f(n-1) + 2^n & (n > 1) \end{cases}$$

编程要求：

1) 主函数功能：从键盘输入n值，通过调用子函数计算数列的第n项并输出。

2) 子函数功能：使用递归法，计算数列2, 6, 14, 30, ....., 的第n项。



```

#include <iostream>
#include<cmath>
using namespace std;
void main()
{
    _____; // 函数声明
    int n;
    long result;
    cout<<"请输入整数n的值:";
    cin>>n;
    _____; // 函数调用
    cout<<"数列的第"<<n<<"项为: " <<result<<endl;
}

```

```

long fun(int n)
{

```

```

    long f;

```

```

    .....

```

```

    //函数递归调用

```

```

    return(f);

```

```

}

```

long f;	
T	F
n==1	
f=2	f=fun(n-1)+pow(2.0,n);
return f;	



# 补充练习

求最大公约数和最小公倍数。

1) 编写一子函数求两个自然数m和n的最大公约数，子函数头要求为：`void fun()`;

2) 主函数输出两个自然数m和n的最大公约数及最小公倍数。

●分析：子函数声明 `void fun()`；没有返回值，而又需要通过子函数得到最大公约数和最小公倍数，故可以设置全局变量(P<sub>224</sub>)

```
#include<iostream>
using namespace std;
int m,n,k; //全局变量
void main()
{
    _____; //函数声明
    cout<<"输入两个自然数"<<endl;
    cin>>m>>n;
    _____; //函数调用
    cout<<m<<"和"<<n<<"的最大公约数是: "<<k<<endl;
    cout<<m<<"和"<<n<<"的最小公倍数是: "<<m*n/k<<endl;
    //分析两个输出语句, 可发现变量k保存最大公约数
}
```

```
void fun()
{
    //辗转法实现求最大公约数
}
```

# 补充练习

统计奇偶个数。

- 1) 编写一子函数分别求出一维整型数组a[n]中所有值为奇数和偶数的元素个数，要求奇偶个数通过函数调用后返回值（return只能返回一个值！故不能通过return返回2个参数的值！）。
- 2) 结果在主函数中输出。

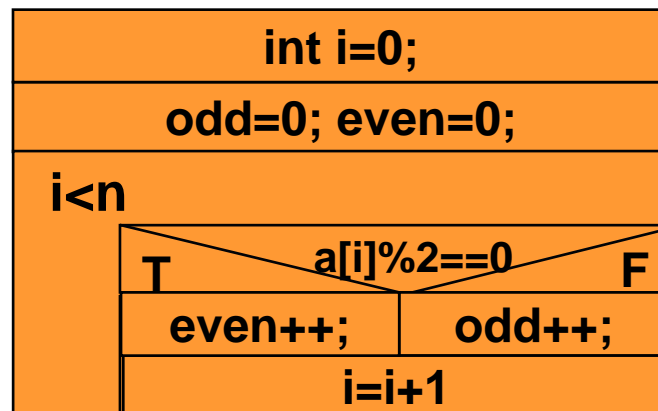
●分析：如果希望在执行被调用函数时，能够改变主调用函数中的某个变量值时，可在对应的被调用函数的形参变量名前加上引用符号&。

```
void fun(int a[],int n,int &odd,int &even); //函数声明
```

```

#include<iostream>
using namespace std;
void main()
{
    const int N=20;
    int a[N],odd_num,even_num,n;
    _____; //函数声明
    cout<<"请输入数组 实际长度n: "<<endl;
    cin>>n;
    for(int i=0;i<n;i++) cin>>a[i];
    _____; //函数调用
    cout<<"值为奇数的元素个数:"<<odd_num;
    cout<<"值为偶数的元素个数:"<<even_num<<endl;
}

```



子函数算法流程图

# 补充练习

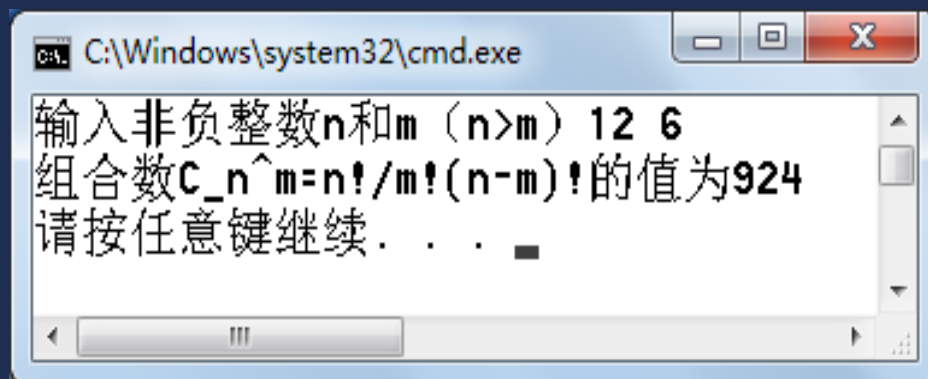
● 实现功能：输入非负整数 $n$ 和 $m$  ( $n > m$ )，计算组合数 $C_n^m = \frac{n!}{m!(n-m)!}$ 的结果值。

编程要求：

● 主函数功能：通过调用子函数实现计算组合数

$C_n^m = \frac{n!}{m!(n-m)!}$ 的结果值。

● 子函数功能：使用递归法实现计算 $n!$ 。



```
C:\Windows\system32\cmd.exe
输入非负整数n和m (n>m) 12 6
组合数C_n^m=n!/m!(n-m)!的值为924
请按任意键继续. . .
```

# 递归求n!

P<sub>235</sub>

分析：计算n!的公式如下：

$$n! = \begin{cases} 1 & (n = 0) \quad \text{递归结束条件} \\ n(n-1)! & (n > 0) \quad \text{规律} \end{cases}$$

这是一个递归形式的公式，应该用递归函数实现。

- 子函数的定义：

函数接收一个整型参数，返回一个整数值long int，即

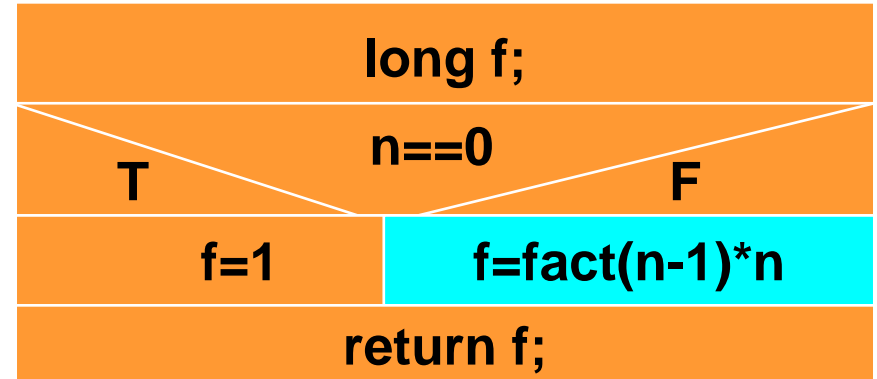
```
long fact(int n) //函数定义
{
.....
}
```

long f;	
T	F
n==0	n==0
f=1	f=fact(n-1)*n
return f;	

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

```
long fact(int n)
```

```
{
    long f;
    if (n==0)
        f=1;
    else
        f=fact(n-1)*n; //函数递归调用
    return(f);
}
```



```
void main()
```

```
{
    int m,n;
    long Cnm;
    cout<<"请输入正整数m和n的值(n>m):";
    cin>>m>>n;
```

```
_____ ; // 函数调用
```

```
cout<<"组合数Cnm=n!/m!(n-m)!的值为: "<<Cnm<<endl;
}
```

# 补充练习

实现功能：在屏幕上输出表达式

$m^n + (m+1)^n + (m+2)^n + (m+3)^n + (m+4)^n$ 的结果值，其中m和n为1到3之间的随机数。

编程要求：

- 1) 主函数功能：通过调用子函数实现在屏幕上输出表达式  $m^n + (m+1)^n + (m+2)^n + (m+3)^n + (m+4)^n$  的结果值。
- 2) 子函数功能：使用递归法实现计算  $m^n$  的结果。

## 如何定义子函数？

要计算  $m^n$ ，则函数必须有2个参数m,n(形参名字不重要，甚至可以省略，但是形参的类型不能省略！)，类型为 int。返回类型为 long int，即

```
long power(int m, int n) //函数定义
{
    ..... //求  $m^n$ 
}
```



```

#include <iostream>
#include <ctime>
using namespace std;
void main()
{
    _____; // 函数声明
    int m,n,i;
    long result;
    srand(time(NULL));
    m=_____ ;
    n=_____ ; // 产生两个指定范围内的随机数
    for(i=0;i<=4; i++)
        result=result+_____ ; // 函数调用
    cout<<"表达式的值为："<<result<<endl;
}

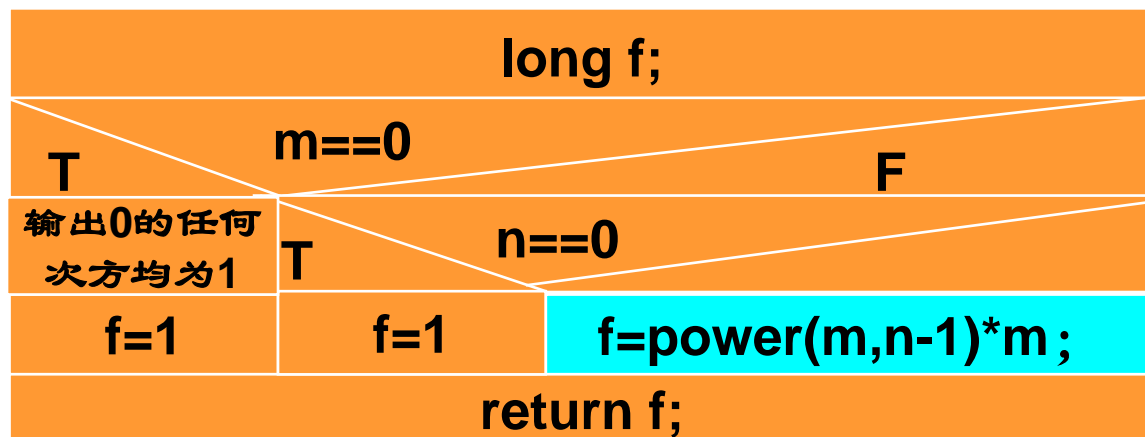
```

**long power(int n)**

```

{
    long f;
    .....
    //函数递归调用
    return(f);
}

```



# 补充练习

实现功能：输出斐波那契数列的前n项中奇数项的和值。

编程要求：

1) 主函数功能：调用子函数计算斐波那契数列的第n项，输出斐波那契数列的前n项中奇数项的和值。

2) 子函数功能：使用递归法计算斐波那契数列的第n项。

# Fibonacci数列 P<sub>253</sub>

- 问题：用递归方法计算Fibonacci数列第n项的值。
- 分析：该问题可以用递归形式来表示，即有递归公式：

$$\text{fibo} = \text{fibonacci}(n-1) + \text{fibonacci}(n-2)$$

递归结束条件为：当 $n=1$ 或 $n=2$ ， $\text{fibo}=1$

可以构建一个递归子函数`int fibonacci(int n)`来实现求第n项的值，主函数中调用该递归子函数。

```

int fibonacci(int n)
{
    int fibo;
    .....

    //函数递归调用
    return fibo;
}

```

int fibo;		
T	n==1    n==2	F
fibo=1		fibo=fibonacci(n-1) +fibonacci(n-2);
return fibo;		

```

#include <iostream>
using namespace std;
void main( )
{
    _____;    //函数声明

    int i,n,fibon=0;
    cout<<"请输入菲波拉契数列的项数:"<<endl;
    cin>>n;
    for(i=1;i<=n; i=i+2)    //求菲波拉契数列奇数项的和
        _____;    //函数调用
    cout<<"前"<<n<<"项菲波拉契数列的奇数项之和为:"<<fibon<<endl;
}

```