怎样学习英文原版教材?

- It depends on yourself, no special way .... ☺
- Anyway, need time, energy, interest ... and so forth ...
- Never give up … no wait, no quit !

第5章 数组（Arrays）

Definition

Array is an aggregation including a certain sequential and same type variables. These variables are called elements of the array which have the same data type.

Array belongs to tectonic type, and can represent each element exclusively with uniform array name and subscript.

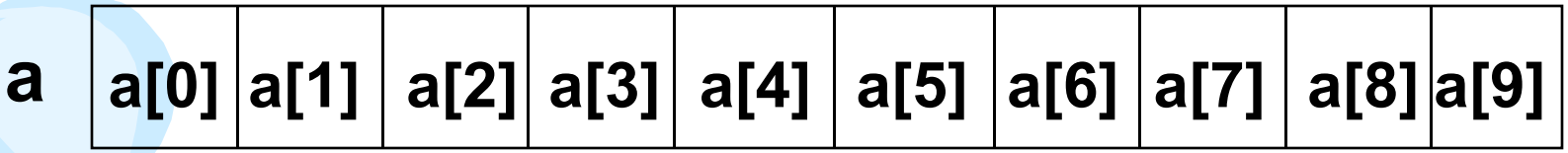
Declaration and Reference of 1-dimension array

- **Declaration**
- 类型标识符　数组名 [常量表达式];
- For example: int a[10];
  - "a" is an integer array with ten elements of a[0] to a[9]
- **Reference**
- 数组名[下标]
  - Declared firstly then used , and only can be reference one by one.

Storage order of 1-dimension array

Array elements are stored in succession, and their address is continuous.

For example：



Note: Array name is memory address of the first array element, so it is a constant and can't be valuated.

C + + does not allow the dynamic definition of the array size .

例5.1 数组元素的引用

```
#include <iostream>
using namespace std;
int main( )
{
    int i,a[10];
    for (i=0;i<=9;i++)
        a[i]=i;
    for (i=9;i>=0;i--)
        cout<<a[i]<<" ";
    cout<<endl;
    return 0;
}
```

•Initialization of 1-dimension array

- Evaluate the array elements with initial value while declaring the array.
  For example：static int a[10]={0,1,2,3,4,5,6,7,8,9};
- Evaluate some of the array elements with initial value.
  For example：static int a[10]={0,1,2,3,4};
- The length of the array may not be specified while evaluating all the elements of the array.
  For example：static int a[ ]={1,2,3,4,5};

Initialization of 1-dimension array

- Can not evaluate the array as a whole
- Global and static array's default value is 0

5.2.4 一维数组程序举例

例5.2 用数组来处理求例3.13 Fibonacci数列问题。

这是一个有趣的古典数学问题：有一对兔子，从出生后第3个月起每个月都生一对兔子。小兔子长到第3个月后每个月又生一对兔子。假设所有兔子都不死，问每个月的兔子总数为多少？

这个数列有如下特点：第1、2个数为1、1。从第3个数开始，每个数是其前面两个数之和。即

F1=1　　(n=1)

F2=1　　(n=2)

Fn=Fn-1+Fn-2　(n≥3)

可以用20个元素代表数列中的20个数，从第3个数开始，可以直接用表达式

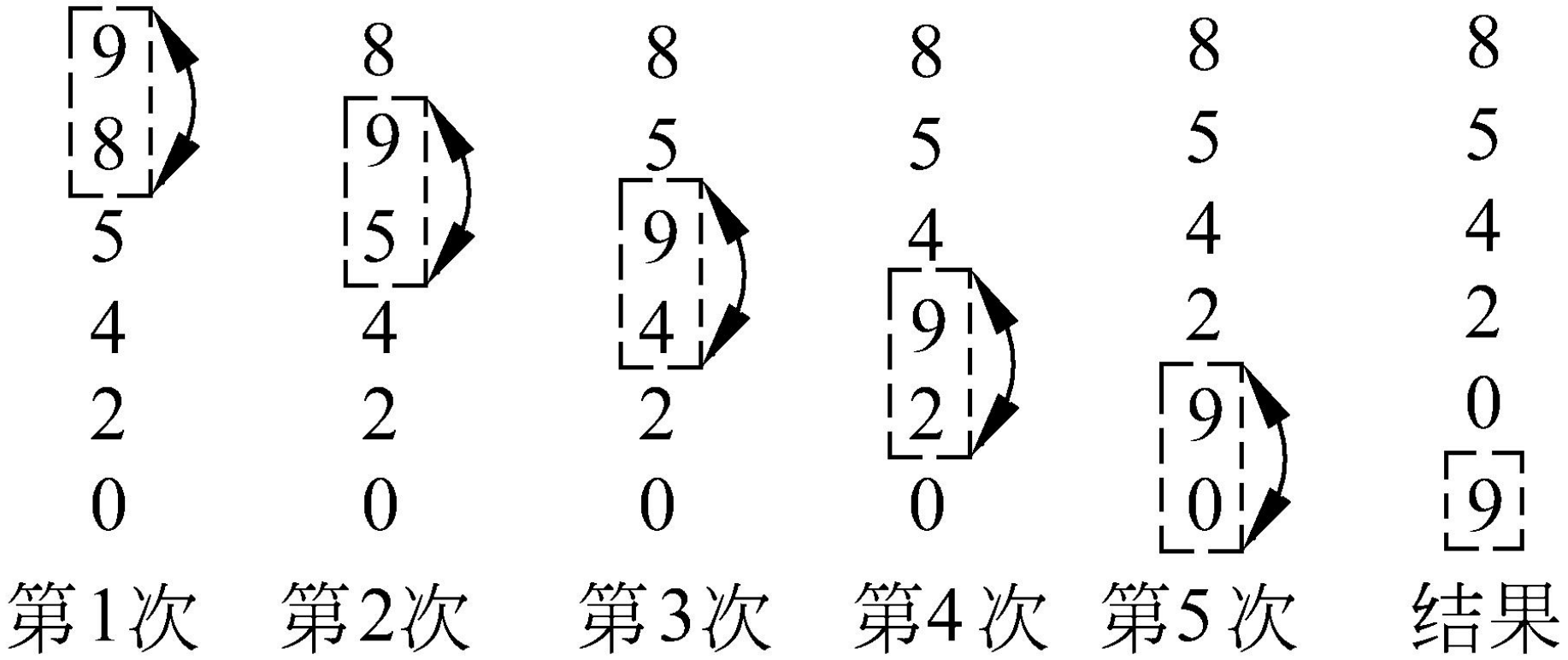
f[i]=f[i-2]+f[i-1]求出各数。

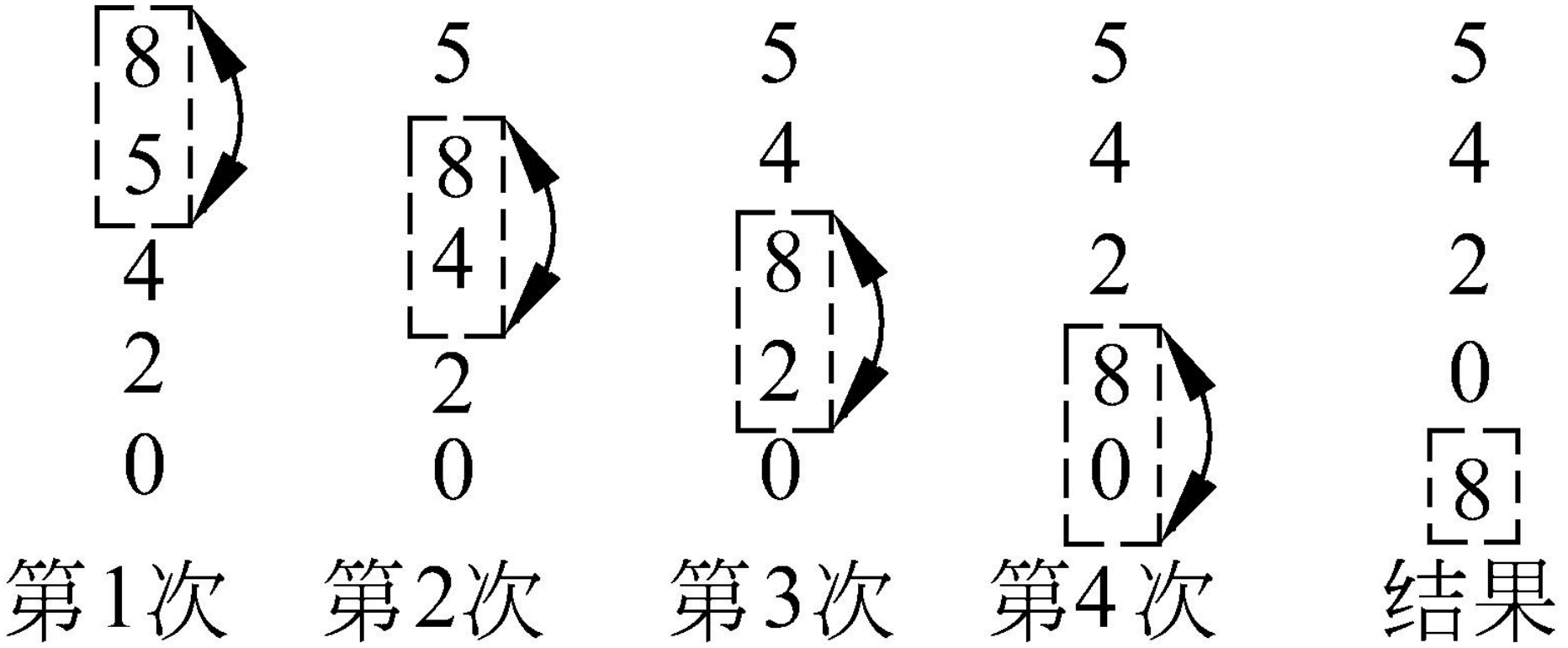
```
#include <iostream>
#include <iomanip>
using namespace std;
int main( )
{ long f1,f2;
   int i;
   f1=f2=1;
   for(i=1;i<=20;i++)
    { cout<<setw(12)<<f1<<setw(12)<<f2;
      if(i%2==0) cout<<endl;
      f1=f1+f2;
      f2=f2+f1;
    }
  return 0;
}
```

```
#include <iostream>
#include <iomanip>
using namespace std;
int main( )
{  int i，f[20]={1,1};
   for(i=2;i<20;i++)
      f[i]=f[i-2]+f[i-1];
   for(i=0;i<20;i++)
   { if(i%5==0) cout<<endl;
       cout<<setw(8)<<f[i];
   }
   cout<<endl; return 0;
}
```

例5.3 起泡法排序

起泡法的思路是：将相邻两个数比较，将小的调到前头。




8
5
4
2
0
5
8
4
2
0
5
4
8
2
0
5
4
2
8
0
5
4
2
0
8
第1次
第2次
第3次
第4次
结果

```
#include <iostream>
using namespace std;
int main( )
{
    int a[11];
    int i,j,t;
    cout<<"input 10 numbers ： "<<endl;
    for (i=1;i<11;i++)
        cin>>a[i];
```

```
cout<<endl;
for (j=1;j<=9;j++)
    for(i=1;i<=10-j;i++)
        if (a[i]>a[i+1])
        {t=a[i];a[i]=a[i+1];a[i+1]=t;}
cout<<"the sorted numbers ： \n";
for(i=1;i<11;i++)
    cout<<a[i]<<" ";
cout<<endl;
return 0;
}
```

习题4-7 验证哥德巴赫猜想

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    void godbaha(int);
    int n;
    cout<<"input n:";
    cin>>n;
    godbaha(n);
    return 0;
}
```

习题4-7 验证哥德巴赫猜想

```
void godbaha(int n)
{
    int prime(int);
    int a,b;
    for(a=3;a<=n/2;a=a+2)
    //为什么从3开始？为什么是a=a+2？为什么a<=n/2？
    {
        if(prime(a))
        {
            b=n-a;
            if (prime(b))
            cout<<n<<"="<<a<<"+"<<b<<endl;
        }
    }
}
```

习题4-7 验证哥德巴赫猜想

```
int prime(int m)
{
    int i,k=sqrt(m);
    for(i=2;i<=k;i++)
        if(m%i==0) break;
    if (i>k) return 1;
    else    return 0;
}
```

Declaration and Reference of 2-dimension array

- **Declaration**

  数据类型 标识符[常量表达式1][常量表达式2] …;

  For example: int a[3][4];

可以理解为：a

a[0]——a_{00} a_{01} a_{02} a_{03}

a[1]——a_{10} a_{11} a_{12} a_{13}

a[2]——a_{20} a_{21} a_{22} a_{23}

- **Storage Order**

  Storing data according to line sequence

a_{00} a_{01} a_{02} a_{03} a_{10} a_{11} a_{12} a_{13} a_{20} a_{21} a_{22} a_{23}

- **Reference**

  For example:

  b [1] [2] = a [1] [2] / 2

下标不要越界

Initialization of 2-dimension array

- Evaluate it according to line sequence.

  For example:

  int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};
- Including all sequential data in a “{ }”

  For example:

  int a[3][4]={{1,2,3,4}, {5,6,7,8}, {9,10,11,12}};
- Evaluate part of the array elements with initial value.

  For example:

  int a[3][4]={{1},{0,6},{0,0,11}};

  int a[3][4]={ };

5.3.4 二维数组程序举例

例5.4 将一个二维数组行和列元素互换，存到另一个二维数组中。例如

$$a=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}\qquad b=\begin{bmatrix}1 & 4\\ 2 & 5\\ 3 & 6\end{bmatrix}$$

```
#include <iostream>
using namespace std;
int main( )
{
int a[2][3]={{1,2,3},{4,5,6}};
int b[3][2],i,j;
cout<<"array a：  "<<endl;
for (i=0;i<=1;i++)
{
        for (j=0;j<=2;j++)
        {
                cout<<a[i][j]<<" ";
                b[j][i]=a[i][j];
        }
        cout<<endl;
}
cout<<"array b：  "<<endl;
for (i=0;i<=2;i++)
{
        for(j=0;j<=1;j++)
                cout<<b[i][j]<<" ";
        cout<<endl;
}
return 0;
}
```

例5.5 有一个3×4的矩阵，要求编程序求出其中值最大的那个元素的值，以及其所在的行号和列号。

开始时把a[0][0]的值赋给变量max，然后让下一个元素与它比较，将二者中值大者保存在max中，然后再让下一个元素与新的max比，直到最后一个元素比完为止。max最后的值就是数组所有元素中的最大值。

```
#include <iostream>
using namespace std;
int main( )
{
    int i,j,row=0,colum=0,max;
    int a[3][4]={{5,12,23,56},{19,28,37,46},{-12,-34,6,8}};
    max=a[0][0];                    //使max开始时取a[0][0]的值
    for (i=0;i<=2;i++)              //从第0行~第2行
            for (j=0;j<=3;j++) //从第0列~第3列
                    if (a[i][j]>max)    //如果某元素大于max
                    {
                            max=a[i][j];          //max将取该元素的值
                            row=i;                //记下该元素的行号i
                            colum=j;              //记下该元素的列号j
                    }
    cout<<"max="<<max<<",row="<<row<<",colum="<<colum<<endl;
    return 0;
}
```

● Array elements as function parameters

- **Array elements act as practical parameters, just as the single variable.**

- **If we make the array name as parameters, the practical and format parameters must be both array name and their type must be the same. Thus when calling , the transfer is the address of the initial element.**

1. 用数组元素作函数实参

例5.6 用函数处理例5.5。

```
#include <iostream>
using namespace std;
int main( )
{  int max_value(int x,int max);                    //函数声明
   int i,j,row=0,colum=0,max
   int a[3][4]={{5,12,23,56},{19,28,37,46},{-12,-34,6,8}};
   max=a[0][0];
   for (i=0;i<=2;i++)
      for (j=0;j<=3;j++)
         { max=max_value(a[i][j],max);          //调用max_value
    函数
          if(max==a[i][j])                      //如果函数返回的是a[i][j]的
    值
             {row=i;                            //记下该元素行号i
```

```
                colum=j; //记下该元素列号j
            }
        }

cout<<"max="<<max<<",row="<<row<<",colum="<<colum<<endl;
return 0;
  }

int max_value(int x,int max)        //定义max_value函数
{if(x>max) return x;                 //如果x>max，函数返回值为x
else return max;                    //如果x≤max，函数返回值为max
  }
```

2. 用数组名作函数参数

```
#include <iostream>
using namespace std;
int  main(void)
{
    int Table[3][4] = {{1,2,3,4},{2,3,4,5},{3,4,5,6}};
    void RowSum(int A[ ][4], int nrow);
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 4; j++)
            cout << Table[i][j] << "   ";
        cout << endl;
    }
    RowSum(Table,3);  return 0;
}
```

```
void RowSum(int A[ ][4], int nrow)
{
    int sum;
    for (int i = 0; i < nrow; i++)
    {
        sum = 0;
        for(int j = 0; j < 4; j++)
            sum += A[i][j];
        cout << "Sum of row " <<i<< " is "
        << sum << endl;
    }
}
```

Reduce the dimension of 2-dimension array

- 由于二维数组在内存中是顺序线性排列的，传递一维数组和二维数组都是传送的地址，所以可以在被调用函数中用单重循环来遍历二维数组中的所有元素，因此只要传递数组名和元素的总个数就可以了。

```
#include <iostream>
using namespace std;
void RowSum(int Array[], int num)
{   int sum=0;

    for (int i = 0; i < num; i++)
    {
            sum += Array[i];
    }
    cout << "Sum is " << sum <<
    endl;
}
```

```
int main()
{   int Table[3][4] =
    {{1,2,3,4},{2,3,4,5},{3,4,5,6}};
    for (int i = 0; i < 3; i++)
    {     for (int j = 0; j < 4; j++)
              cout << Table[i][j] << "   ";
          cout << endl;
    }
    RowSum(&Table[0][0],3*4);//传递的
    必须是第一个元素的地址
    return 0;
}
```

例5.7 用选择法对数组排序。

所谓选择法就是先将10个数中最小的数与a[0]对换;再将a[1]到a[9]中最小的数与a[1]对换……每比较一轮,找出一个未经排序的数中最小的一个。共比较9轮。

```
#include <iostream>
using namespace std;
int main( )
{
    void select_sort(int array[],int n);          //函数声明
    int a[10],i;
    cout<<"enter the originl array：<<endl;
    for(i=0;i<10;i++)                             //输入10个数
        cin>>a[i];
    cout<<endl;
    select_sort(a,10);                            //函数调用，数组名作实参
    cout<<"the sorted array：<<endl;
    for(i=0;i<10;i++)                             //输出10个已排好序的数
        cout<<a[i]<<"  ";
    cout<<endl;
    return 0;
}
```

```
void select_sort(int array[],int n)     //形参array是数组名
{
    int i,j,k,t;
    for(i=0;i<n-1;i++)
    {
        k=i;
        for(j=i+1;j<n;j++)
            if(array[j]<array[k]) k=j;
        t=array[k];array[k]=array[i];array[i]=t;
    }
}
```

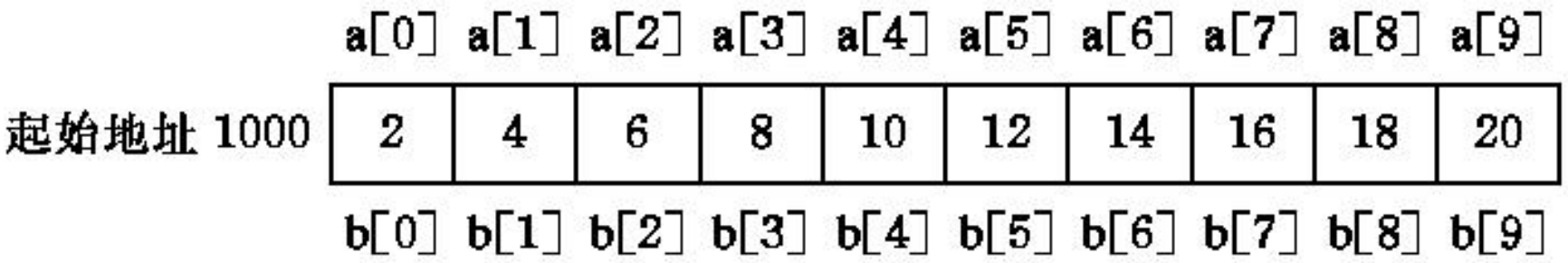
关于用数组名作函数参数有两点要说明：

(1) 如果函数实参是数组名，形参也应为数组名。实参数组与形参数组类型应一致。

(2) 数组名代表数组首元素的地址，并不代表数组中的全部元素。因此用数组名作函数实参时,不是把实参数组的值传递给形参，而只是将实参数组首元素的地址传递给形参。

形参可以是数组名，也可以是指针变量，它们用来接收实参传来的地址。

在调用函数时，将实参数组首元素的地址传递给形参数组名。这样，实参数组和形参数组就共占同一段内存单元。



在用变量作函数参数时，实参的值不因形参的值改变而改变。

而用数组名作函数实参时,改变形参数组元素的值将同时改变实参数组元素的值。

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，在调用函数时也不对它分配存储单元，只是用**array[]**这样的形式表示**array**是一维数组名，以接收实参传来的地址。

3. 用多维数组名作函数参数

如果用二维数组名作为实参和形参，在对形参数组声明时，必须指定第二维(即列)的大小，且应与实参的第二维的大小相同。

```
int array[3][10];      //形参数组的两个维都指定
int array[  ][10];     //第一维大小省略
int array[8][10];      //第一维大小与实参不同
```

例5.8 求矩阵中所有元素中的最大值

```
#include <iostream>
using namespace std;
int main( )
{  int max_value(int array[][4]);
   int a[3][4]={  {11,32,45,67},
                  {22,44,66,88},
                  {15,72,43,37}};
   cout<<"max value is "<<max_value(a)<<endl;
   return 0;
}
```

```
int max_value(int array[][4])
{
    int i,j,max;
    max=array[0][0];
    for( i=0;i<3;i++)
        for(j=0;j<4;j++)
            if(array[i][j]>max)   max=array[i][j];
      return max;
}
```

典型例题

- 1979年,诺贝尔奖获得者李政道教授到中国科技大学讲学,他给少年班的同学出了这样一道算术题:有5只猴子在海边发现一堆桃子,决定第二天来平分.第二天清晨,第一只猴子最早来到,它左分右分分不开,就朝海里扔了一只,恰好可以分成5份,它拿上自己的一份走了.第2,3,4,5只猴子也遇到同样的问题,采用了同样的方法,都是扔掉一只后,恰好可以分成5份.问这堆桃子至少有多少只.

```
#include <iostream>
using namespace std;
int main()
{
    int i,j,k,count;
    for(i=4;i<10000;i+=4)  //最后剩下四份桃子的总数
    {
        count=0;
        j=i;
        for(k=5;k>=1;k--)
        {
            j=j/4*5+1; //猴子面前的桃子数，也被上一个猴子分为四份
            if(j%4==0) count++;
            else break;
        }
        if(count==4)
        {cout<<j<<'\n'; break;}
    }
    return 0;
}
```

- 其通解：设有n个猴子，每次要扔掉m个桃子，m,n>0,求桃子的最小数

- 应该是：Total=n^n-(m*n-m)

典型例题

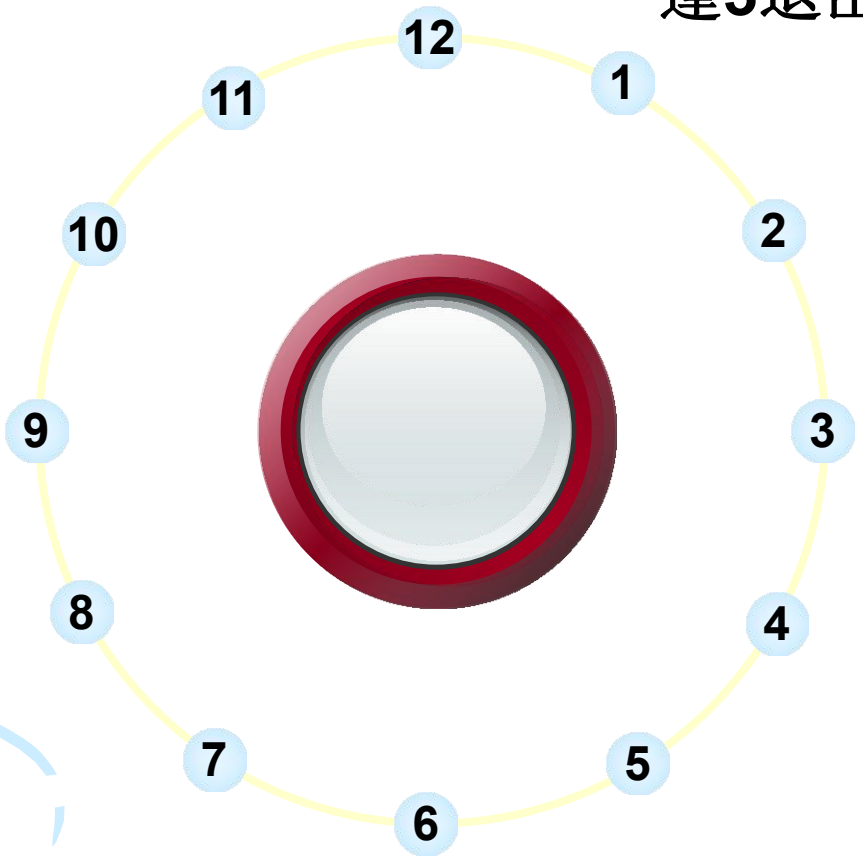
- 题目：有M个敢死队员要炸掉敌人的一碉堡，谁都不想去，排长决定用轮回数数的办法来决定哪个士兵去执行任务。如果前一个士兵没完成任务，则要再派一个士兵上去。现给每个士兵编一个号，大家围坐成一圈，随便从某一个士兵开始计数，当数到5时，对应的士兵就去执行任务，且此士兵不再参加下一轮计数。如果此士兵没完成任务，再从下一个士兵开始数数，被数到第5时，此士兵接着去执行任务。以此类推，直到任务完成为止。
- 排长是不愿意去的，假设排长为1号，请你设计一程序，求出从第几号士兵开始计数才能让排长最后一个留下来而不去执行任务。

典型例题

- 题目：有n个人围成一圈，顺序排号。从第一个人开始报数（从1到3报数），凡报到3的人退出圈子，问最后留下的是原来第几号的那位。

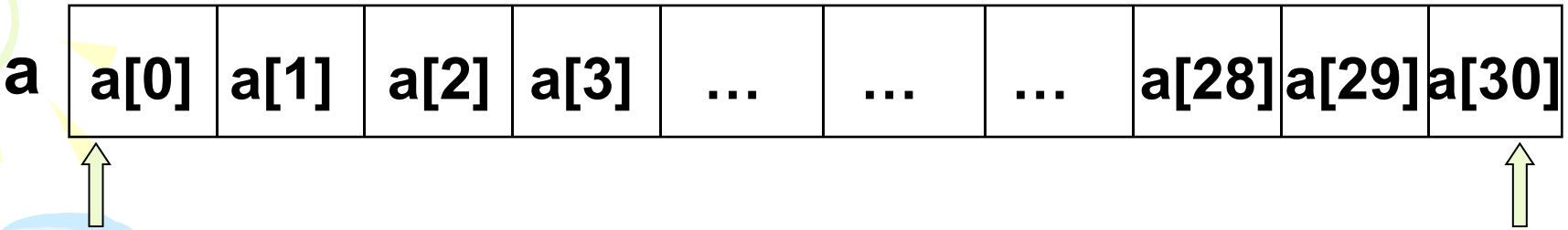
示意图

逢3退出



从1号开始计数，先计算最后一个人的编号。

算法分析



- 初始化数组，利用循环变量 i引用数组元素
- 报数，计数器k=5时，相应的数组元素记0
- 到数组末尾时，循环变量i 置 0
- 检查 a[i] !=0
- 退出条件m，m=n-1时只剩一人

数组解法

```
#include <iostream>
using namespace std;
const int nmax= 500;
int main()
{
    int i, k, m, n, num[nmax];
    cout<<"please input the total of numbers:";
    cin>>n;                          //总人数
    for(i=0;i<n;i++)                 //数组初始化，从1-30编号
          num[i]=i+1;
    i=0;                             //循环变量
    k=0;                             //计数器，从1-5计数
    m=0;                             //已出局人数
```

数组解法

```
while(m<n-1)                    //剩一人退出
{
    if(num[i]!=0) k++;          //没有退出的报数
    if(k==5)                    //报数到5
    {
        num[i]=0;               //数到5的退出
        k=0;                    //重新开始报数
        m++;                    //退出人数加1
    }
    i++;                        //访问数组下一个元素
    if(i==n) i=0;               //回到数组第一个位置
}
```

数组解法

```
    i=0;
    while(num[i]==0) i++;              //找到不为0的元素
    cout<<num[i]<<“   is left\n”;   //输出
    ////cout<<n-i<<endl;////
    return 0;
}
```

数组解法

```
#include <iostream>
using namespace std;
const nmax= 50;
int main()
{
        int i,k,m,n,num[nmax];
        cout<<"please input the total of numbers:";
        cin>>n;
        for(i=0;i<n;i++)
                num[i]=i+1;
        i=0;
        k=0;
        m=0;
        while(m<n-1)
        {
                if(num[i]!=0) k++;
                if(k==5)
                {
                        num[i]=0;
                        k=0;
                        m++;
                }
                i++;
                if(i==n) i=0;
        }
        i=0;
        while(num[i]==0) i++;
        cout<<num[i]<<"   is left\n";
        return 0;
}
```

指针解法

```
#include <iostream>
using namespace std;
#define nmax 50
int main()
{
        int i,k,m,n,num[nmax],*p;
        printf("please input the total of numbers:");
        scanf("%d",&n);
        p=num;
        for(i=0;i<n;i++)
                *(p+i)=i+1;
        i=0;
        k=0;
        m=0;
        while(m<n-1)
        {
                if(*(p+i)!=0) k++;
                if(k==3)
                { *(p+i)=0;
                k=0;
                m++;
                }
                i++;
                if(i==n) i=0;
        }
        while(*p==0) p++;
        printf("%d is left\n",*p);
        return 0;
}
```

习题4-9 汉诺塔问题

- 将n个盘子从A座移到C座上，可分为3个步骤：
- （1）将A上n-1个盘子借助C座先移到B座上；
- （2）把A座上剩下的一个盘子B座上；
- （3）将n-1个盘子从B座借助A座移到C座上。
- 上面三个步骤可分为两类操作：
- （1）将n-1个盘子从一个座移到另一个座上（当n>1时），这是一个递归的过程；
- （2）将1个盘子从一个座上移到另一个座上。

```
#include <iostream>
using namespace std;
int main()
{
    void hanoi(int n,char one,char two,char three);
    int m;
    cout<<"input the number of diskes:";
    cin>>m;
    cout<<"The steps of moving"<<m<<“disks:"<<endl;
    hanoi(m,'A','B','C');
    return 0;
}
```

```
void hanoi(int n,char one,char two,char three)
//将n个盘从one座借助two座，移到three座
{
    void move(char x,char y);
    if(n==1) move(one,three);
    else
    {
        hanoi(n-1,one,three,two);
        move(one,three);
        hanoi(n-1,two,one,three);
    }
}
void move(char x,char y)
{cout<<x<<"-->"<<y<<endl;}
```

5.5 character array

Declaration and Initialization of character array

A character array can be initialized with either a list of comma-separated character literals enclosed in

braces or a string literal.

Note, however, that the two forms are not equivalent.

The string constant contains the additional terminating null character('\0'). For example:

- char ca1[] = { 'C', '+', '+' };
- char ca2[] = { "C++" } ;
- char ca3[] = "C++";
- ca1 is of dimension 3; ca2,ca3 is of dimension 4. It is equivalent to
- char ca4[]={'C', '+', '+' ,'\0'};
- The following declaration will be flagged as an error:
- const char ch3[6] = "Daniel";
- // error: array bounds overflow

Assignment and references of character array

- An array cannot be initialized with another array, nor can one array be assigned to another.
- char c[5];
- c={‘C’,‘h’,‘i’,‘n’,‘a’};
- // error
- int ia2[] = ia;
- // error: cannot initialize one array with another

- Additionally, it is not permitted to declare an array of references.

- int *iap [] = { &ix, &jx, &kx };
- // ok: array of pointers of type

- int &iar[] = { ix, jx, kx };
- // error: array of references not allowed

例5.9 设计和输出一个钻石图形

```
#include <iostream>
using namespace std;
int main( )
{
    char diamond[][5]={{' ',' ','*'},{' ','*',' ','*'},{'*',' ',' ',' ','*'},{' ','*',' ','*'},{' ',' ','*'}};
    //char diamond[][5]={{"  *"},{" * * "},{"*   *"},{" * * "},{"  *"}};
    int i,j;
    for (i=0;i<5;i++)
    {
            // cout<<diamond[i];
            for (j=0;j<5;j++)          cout<<diamond[i][j];
            cout<<endl;
    }
    return 0;
}
```

例5.9 设计和输出一个钻石图形

```
int main( )
{
    char diamond[][5]=
{{' ',' ','*'},{' ','*',' ','*'},{'*',' ',' ',' ','*'},{' ','*',' ','*'},{' ',' ','*'}};
    //未被赋值的元素自动定为空字符'\0'
    int i,j;
    for (i=0;i<5;i++)
            cout<<diamond[i]<<endl;
    //从数组第一个元素开始逐个输出字符，直到'\0'为止
    return 0;
}
```

输出 *　***的原因在于diamond[2]刚好5个元素，结尾没有'\0',遇到的'\0'是在下一行diamond[3]结尾处。

5.5.4 字符数组的输入输出

字符数组的输入输出可以有两种方法：

(1) 逐个字符输入输出。

(2) 将整个字符串一次输入或输出。如

```
char str[20];
cin>>str;                    //用字符数组名输入字符串
cout<<str;                   //用字符数组名输出字符串
```

字符数组名str代表字符数组第一个元素的地址，执行“cout<<str;”的过程是从str所指向的数组第一个元素开始逐个输出字符，直到遇到'\0'为止。

Input/output of character string

- Method:
- Input/output character gradually.
- Input/output the whole character string once.
  For example：　　char c[ ]="China";
  　　cout<<c;
- Note:
- The output character don’t include '\0‘.
- 输出字符串时，输出项是字符数组名，输出时遇到'\0'结束。

Input character string of a line

- cin输入多个字符串时，以空格分隔；输入单个字符串时其中不能有空格。
- cin.getline(字符数组名St, 字符个数N, 结束符);
- 功能：一次连续读入多个字符（可以包括空格），直到读满N个，或遇到指定的结束符（缺省为‘\n’）。读取但不存储结束符。
- cin.get(字符数组名St, 字符个数N, 结束符);
- 功能：一次连续读入多个字符（可以包括空格），直到读满N个，或遇到指定的结束符（缺省为'\n'）。既不读取也不存储结束符。

习题 3.16

```
#include <iostream>
using namespace std;
int main ()
{
        char c;
        int letters=0,space=0,digit=0,other=0;
        cout<<"enter one line::"<<endl;
        while((c=getchar())!='\n')
        {
                if (c>='a' && c<='z'||c>='A' && c<='Z')   letters++;
                else if (c==' ')    space++;
                else if (c>='0' && c<='9')          digit++;
                else      other++;
        }
        cout<<"letter:"<<letters<<", space:"<<space<<",
digit:"<<digit<<", other:"<<other<<endl;
        return 0;
}
```

Example of input character string of a line

```
#include <iostream>
using namespace std;
int main ()
{
    char city[80], state[80];
    for (int i = 0; i < 2; i++)
    {
        cin.getline(city,80,',');
        cin.getline(state,80,'\n');
        cout<<"City:"<<city<<"State:"<<state<<endl;
    }
    return 0;
}
```

注意

用cin从键盘向计算机输入一个字符串时，从键盘输入的字符串应短于已定义的字符数组的长度，否则会出现问题。

```
char city[8];
cin>>city;
```

5.5.5 字符串处理函数

C和C++提供了一些字符串函数，它们放在函数库中，在string和string.h头文件中定义。

如果程序中使用这些字符串函数，应包含头文件string或string.h 。

```
#include <string.h >
```

或

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

5.5.5 字符串处理函数

- #include <string>　　//应包含头文件string
- string catenate
  strcat(char[],const char[]);
- string copy
  strcpy(char[],const char[]);
- string compare
  strcmp(const char[],const char[]);
- string length
  strlen(const char[]);

1. 字符串连接函数 strcat

其函数原型为

strcat(char[],const char[]);

strcat（string catenate）函数有两个字符数组的参数，函数的作用是：将第二个字符数组中的字符串连接到前面字符数组的字符串的后面。

```
char str1[30]="People's Republic of ";
char str2[]="China";
cout<<strcat(str1，str2)); //调用strcat函数
```

输出：

People's Republic of China

连接前后的状况如所示。

tr1:	P	e	o	p	l	e	'	s	␣	R	e	p	u	b	l	i	c	␣	o	f	␣	\0	\0	\0	\0	\0	\0
tr2:	C	h	i	n	a	\0																					
tr3:	P	e	o	p	l	e	'	s	␣	R	e	p	u	b	l	i	c	␣	o	f	␣	C	h	i	n	a	\0

2. 字符串复制函数strcpy

其函数原型为

strcpy(char[],const char[]);

strcpy（string copy）的作用是将第二个字符数组中的字符串复制到第一个字符数组中去，将第一个字符数组中的相应字符覆盖。如

char str1[10]，str2[]="China";

strcpy(str1，str2);

3. 字符串比较函数strcmp

其函数原型为

strcmp(const char[],const char[]);

strcmp（string compare）的作用是比较两个字符串。两个字符数组只参加比较而不应改变其内容，因此两个参数都加上**const**声明。

strcmp(str1，str2);

strcmp("China"，"Korea");

strcmp(str1，"Beijing");

比较的结果由函数值带回。

(1) 如果字符串1=字符串2，函数值为0。

(2) 如果字符串1>字符串2，函数值为一正整数。

(3) 如果字符串1<字符串2，函数值为一负整数。

字符串比较即对两个字符串自左至右逐个字符相比(按ASCII码值大小比较)，直到出现不同的字符或遇到'\0'为止。如全部字符相同，则认为相等；若出现不相同的字符，则以第一个不相同的字符的比较结果为准。

```
if(str1>str2) cout<<"yes";
if(strcmp(str1，str2)>0) cout<<"yes";
```

4. 字符串长度函数strlen

函数原型为

strlen(const char[]);

strlen（string length）测试字符串的实际长度，不包括'\0'在内。如

```
char str[10]="China";
cout<<strlen(str);
```

结果是5

5.5.6 字符数组应用举例

例5.10 有3个字符串,要求找出其中最大者。

```
#include <iostream>
#include <string>
using namespace std;
int main( )
{
    void max_string(char str[][30],int i);      //函数声明
    int i;
    char country_name[3][30];
    for(i=0;i<3;i++)
        cin>>country_name[i];                   //输入3个国家名
    max_string(country_name,3);                 //调用max_string函
数
    return 0;
}
```

```
void max_string(char str[][30],int n)
{
    int i;
    char string[30];
    strcpy(string,str[0]);
    for(i=0;i<n;i++)
        if(strcmp(str[i],string)>0)
            strcpy(string,str[i]);
    cout<<endl<<"the largest string is：  "<<string<<endl;
}
```

5.6 字符串类与字符串变量

C++提供了一种新的数据类型——字符串类型(string类型)， string是在C++标准库中声明的一个字符串类。

在使用方法上，string和char、int类型一样，可以用来定义变量，每一个字符串变量都是string类的一个对象。

要使用string类的功能时，必须包含的头文件。

```
#include <string>
```

5.6.1 字符串变量的定义和引用

1. 定义字符串变量

定义字符串变量要用类名string。如

```
string string1;              //定义string1为字符串变量

string string2="China";      //定义string2同时对其初始化
```

2. 对字符串变量的赋值

- 可以对字符串变量赋予一个字符串常量，如

string1="Canada";

- 可用字符串变量给另一字符串变量赋值。如

string2=string1;

- 可以对字符串变量中某一字符进行操作

3. 字符串变量的输入输出

可以在输入输出语句中用字符串变量名，输入输出字符串，如

```
cin>> string1;          //从键盘输入一个字符串给字符串变量string1

cout<< string2;         //将字符串string2输出
```

5.6.2 字符串变量的运算

- (1) 字符串复制用赋值号
  string1=string2;

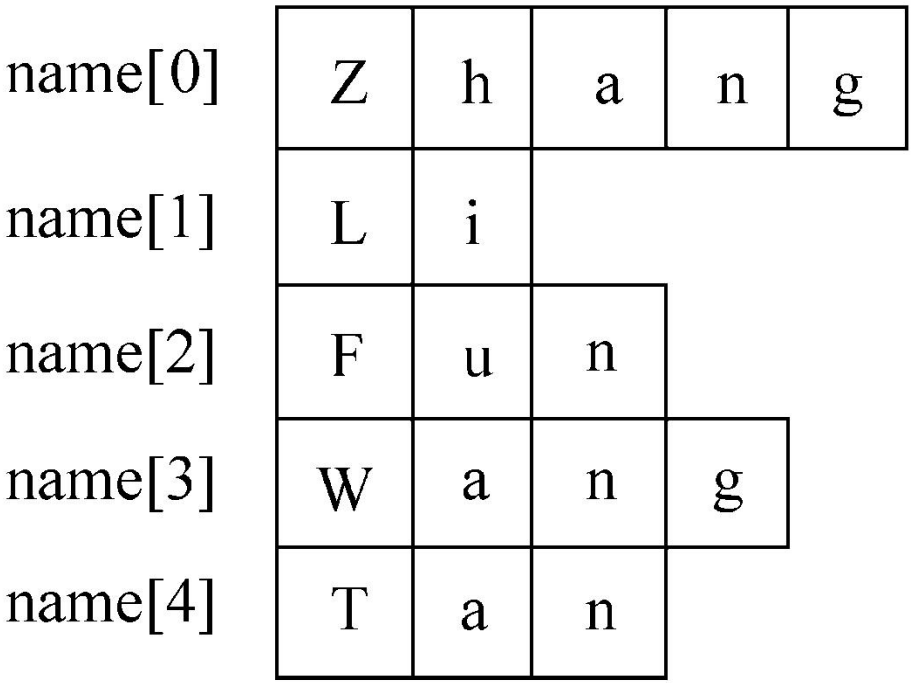
- (2) 字符串连接用加号
  string1=string1 + string2;

- (3) 字符串比较直接用关系运算符
  ==、>、<、!=、>=、<=等进行字符串的比较。

5.6.3 字符串数组

可以用string定义字符串数组。如

```
string
name[5]={"Zhang","Li","Fun","Wang","Tan"};
//定义一个字符串数组并初始化
```



(1) 在字符串数组中，每个元素相当于一个字符串变量。

(2) 不要求每个字符串元素具有相同的长度。

(3) 在字符串数组的每一个元素中存放一个字符串，而不是一个字符。

(4) 每一个字符串元素中只包含字符串本身的字符而不包括'\0'。

在定义字符串数组时怎样给数组分配存储空间呢？

实际上，编译系统为每一个字符串变量分配4个字节，在这个存储单元中，存放字符串的地址。

在字符串变量中存放的是字符串的指针(字符串的地址)。

例5.11 按顺序输出字符串

```
#include <iostream>
#include <string>
using namespace std;
int main( )
{
    string string1,string2,string3,temp;
    cout<<"please input three strings：";
    cin>>string1>>string2>>string3;
    if(string2>string3) //使串2≤串3
    {temp=string2;string2=string3;string3=temp;}
```

```
if(string1<=string2)
    cout<<string1<<" "<<string2<<" "<<string3<<endl;
    //如果串1≤串2，则串1≤串2≤串3
else if(string1<=string3)
    cout<<string2<<" "<<string1<<" "<<string3<<endl;
    //如果串1＞串2，且串1≤串3，则串2＜串1≤串3
else
    cout<<string2<<" "<<string3<<" "<<string1<<endl;
    //如果串1＞串2，且串1＞串3，则串2＜串3＜串1
return 0;
}
```

例5.12 一个班有n个学生，需要把每个学生的简单材料(姓名和学号)输入计算机保存。然后可以通过输入某一学生的姓名查找其有关资料。当输入一个姓名后，程序就查找该班中有无此学生，如果有，则输出他的姓名和学号，如果查不到，则输出“本班无此人”。

```
#include <iostream>
#include <string>
using namespace std;
string name[50],num[50];
int n;
int main( )
{
   void input_data( );
   void search(string find_name);
   string find_name;
   cout<<"please input number of this class：";
   cin>>n;
   input_data( );
   cout<<"please input name you want find：";
   cin>>find_name;
   search(find_name);
   return 0;
}
```

```
void input_data( )
{
    int i;
    for (i=0;i<n;i++)
    {
        cout<<"input name and NO. of student "<<i+1<<":
";
        cin>>name[i]>>num[i];
    }
}
```

```
void search(string find_name)
{
    int i;
    bool flag=false;
    for(i=0;i<n;i++)
       if(name[i]==find_name)
       {
           cout<<name[i]<<" has been found, his number is
" <<num[i]<<endl;
           flag=true;
           break;
       }
    if(flag==false) cout<<"can't find this name";
}
```

思考

(1) 程序第5行定义全局变量时，数组的大小不指定为50，而用变量n，即string name[n],num[n];n在运行时输入，行不行？为什么？

(2) search函数for循环中最后有一个break语句，它起什么作用？不要行不行？

(3) 如果不使用全局变量，把变量n和数组name，num都作为局部变量，通过虚实结合的方法在函数间传递数据，这样行不行？

字符数组与string变量

```
#include <iostream>
#include <string>
using namespace std;
int main( )
{
        char str[]="abcd";
        string temp="abcd";
        for (int i=0;i<6;i++)
                temp[i]=str[i];
        cout<<temp;
        putchar(temp[4]+65);
        cout<<temp.size();
        char str1[4]="";
        for (i=0;i<6;i++)
                putchar(str1[i]);

        return 0;
}
```

作业

- **P155：4，5，6，8，10，13**

5-4

```
#include <iostream>
using namespace std;
int main()
 {
        int a[11]={1,4,6,9,13,16,19,28,40,100};
        int num,i,j;
        cout<<"array a:"<<endl;
        for (i=0;i<10;i++)
                cout<<a[i]<<" ";
        cout<<endl;
        cout<<"insert data:";
        cin>>num;
        if (num>a[9])
                a[10]=num;
        else
    {
                for (i=0;i<10;i++)
                {
                        if (a[i]>num)
                        {
                                for (j=9;j>=i;j--)
                                        a[j+1]=a[j];
                                a[i]=num;
                                break;
                        }
                }
        }
        cout<<"Now, array a:"<<endl;
        for (i=0;i<11;i++)
                cout<<a[i]<<" ";
        cout<<endl;
        return 0;
 }
```

习题5-6 杨辉三角

```
#include <iostream>
using namespace std;
int main()
{
        int a[10][10];
        int i;
        for(i=0;i<10;i++)
        {
                a[i][0]=1;
                a[i][i]=1;
        }
        for(i=2;i<10;i++)
        {
                for(int j=1;j<i;j++)
                        a[i][j]=a[i-1][j-1]+a[i-1][j];
        }
        for(i=0;i<10;i++)
        {
                for(int j=0;j<=i;j++)
                        cout<<a[i][j]<<" \t";
                cout<<endl;
        }
        return 0;
}
```

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
        int m, n, c = 1;
        for (m = 0; m <=13; m++)
        {
                cout<<setw(40-3*m)<<c;
                for (n = 1; n<=m; n++)
                {
                        c = c*(m-n+1)/n;
                        cout<<setw(6)<<c;
                }
                cout << "\n";
        }
        return 0;
}
```

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
   int i,j,n=10;
       int c(int x,int y);
   for(i=0;i<=n;i++)      /*控制输出N行*/
   {
      for(j=0;j<24-2*i;j++) cout<<" ";  /*控制输出第i行前面的空格*/
      for(j=1;j<i+2;j++) cout<<setw(4)<<c(i,j);              /*输出第i行的第j个值*/
      cout<<"\n";
   }
       return 0;
}
int c(int x,int y)    /*求杨辉三角形中第x行第y列的值*/
{
   int z;
   if((y==1)||(y==x+1))  return 1;  /*若为x行的第1或第x+1列，则输出1*/
   z=c(x-1,y-1)+c(x-1,y);        /*否则，其值为前一行中第y-1列与第y列值之和*/
   return z;
}
```

5-8

```
#include <iostream>
using namespace std;
int main()
{
        const int n=7;
        int i,number,top,bott,mid,loca;
        bool flag=true,sign;
        char c;
        int a[n]={11,23,35,47,59,61,75};
        for (i=0;i<n;i++)
                cout<<a[i]<<" ";
        cout<<endl;
        while(flag)
    {
                cout<<"input number to look for:";
                cin>>number;
                sign=false;
                top=0;          //top是查找区间的起始位置
                bott=n-1;       //bott是查找区间的最末位置
                if ((number<a[0])||(number>a[n-1]))  //要查的数不在查找区间内
                        loca=-1;       // 表示找不到
                while ((!sign) && (top<=bott))
                {
                        mid=(bott+top)/2;
                        if (number==a[mid])
                        {
                                loca=mid;
                                cout<<"Find "<<number<<", its position is "<<loca+1<<endl;
                                sign=true;
                        }
                        else if (number<a[mid])
                                bott=mid-1;
                        else
                                top=mid+1;
                }
                if(!sign||loca==-1)
                        cout<<number<<" has not found."<<endl;
                cout<<"continu or not(Y/N)?";
                cin>>c;
                if (c=='N'||c=='n')
                        flag=false;
    }
        return 0;
}
```

十进制转二进制

```
/**********************************
 *  10进制转换为2进制输出  *
 **********************************/
#include <iostream>
using namespace std;
void convert(int);
int main(void)
{
    int data;
    cout << "请输入你要转换的数：  ";
    cin >> data;
    convert(data);
    return 0;
}
/**函数*/
void convert(int fdata)
{
    int c[12] = {0}, count;
    count = 0;
    for(int i = 0; fdata > 0; i++)
{
        c[i] = fdata%2;
        fdata = fdata/2;
        count++;
    }
    for(int j = count; j > 0; j--)
{
        cout << c[j-1];
    }
    cout << endl;
}
```

```
/*
输入一个正整数(十进制)，输出此整数对应的二进制数(用循环实现，但不用数组)

分析：十进制转换成二进制的方式：
    用辗转相除依次取余，直到商为0，获取逆序的余数序列就是对应的二进制数
            用循环不用数组，因此，将按先得到的余数序列组成一个整数值，
            再逆序输出该整数值（从低位到高位输出），即获得二进制代码
*/

#include <iostream>
using namespace std;
int main()
{
            int n; //待转换的十进制正整数
            int m; //存每次转换得到的余数
            int sum=0;          //进制转换逆序值
            int count=0; //记录二进制位数
            int i;
            cout<<"请输入待转换的十进制正整数:";
            cin>>n;
   while(n<0)
            {
              cout<<"请重新输入待转换的十进制正整数:";
              cin>>n;
            }
            cout<<endl;
   cout<<"十进制"<<n<<"的二进制形式为:";
            if(n==0)
            {cout<<n<<endl;
             return 0;
            }
            while(n!=0)   //辗转相除取余到商为0
            {
              m=n%2;   //获取对应此次的余数
    count++;  //二进制位数增1
              sum=sum*10+m;   //余数按先得到顺序组成一个整数，最后反序就是2进制数
              n=n/2;
            }
   for(i=count;i>0;i--)  //循环从低位到高位逆序输出sum各个位上的数
            { cout<<sum%10;
              sum=sum/10;
            }
            cout<<endl;
            return 0;
}
```

```
#include <iostream>
using namespace std;
void convert(int x);
int main()
{
    int x;
    cin>>x;
    convert(x);
    return 0;
}
void convert(int x)
{
    if(x/2==0)      cout<<1;
    else {  convert(x/2);  cout<<x%2;   }
}
```

C++初学者必看的50个建议

- 38.不要漏掉书中任何一个练习题——请全部做完并记录下解题思路；
- 39.C++语言和C++的集成开发环境要同时学习和掌握；
- 40.既然决定了学C++,就请坚持学下去，因为学习程序设计语言的目的是掌握程序设计技术，而程序设计技术是跨语言的；
- 41.就让C++语言的各种平台和开发环境去激烈的竞争吧，我们要以学习C++语言本身为主；

C++初学者必看的50个建议

- 42.当你写C++程序写到一半却发现自己用的方法很拙劣时，请不要马上停手；请尽快将余下的部分粗略的完成以保证这个设计的完整性，然后分析自己的错误并重新设计和编写；

- 44.决不要因为程序“很小”就不遵循某些你不熟练的规则——好习惯是培养出来的，而不是一次记住的；

- 45.每学到一个C++难点的时候，尝试着对别人讲解这个知识点并让他理解——你能讲清楚才说明你真的理解了；

C++初学者必看的50个建议

- 46.记录下在和别人交流时发现的自己忽视或不理解的知识点；
- 47.请不断的对自己写的程序提出更高的要求,哪怕你的程序版本号会变成Version 100.XX；
- 48.保存好你写过的所有的程序——那是你最好的积累之一；
- 49.请不要做浮躁的人；
- 50.请热爱C++!