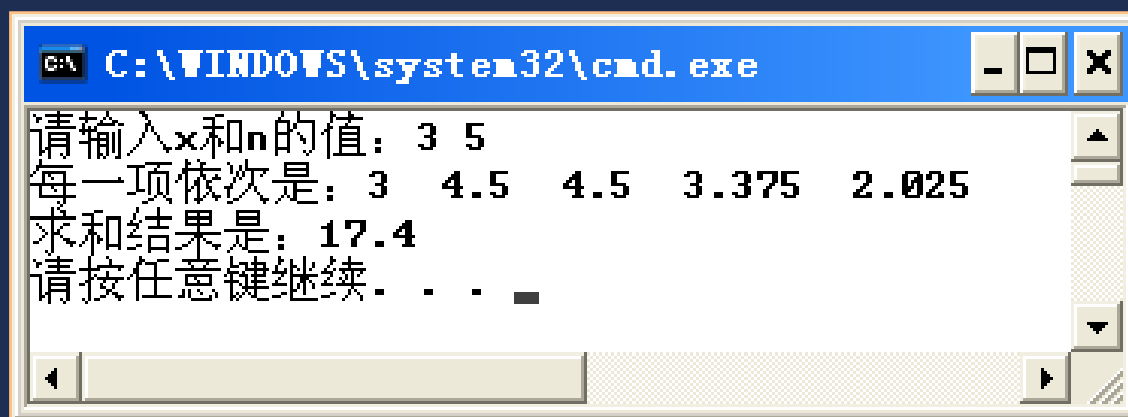


# 实验十二

1. 编程实现：键盘输入x和n的值，求下列公式的值，要求x,n,sum用指针。

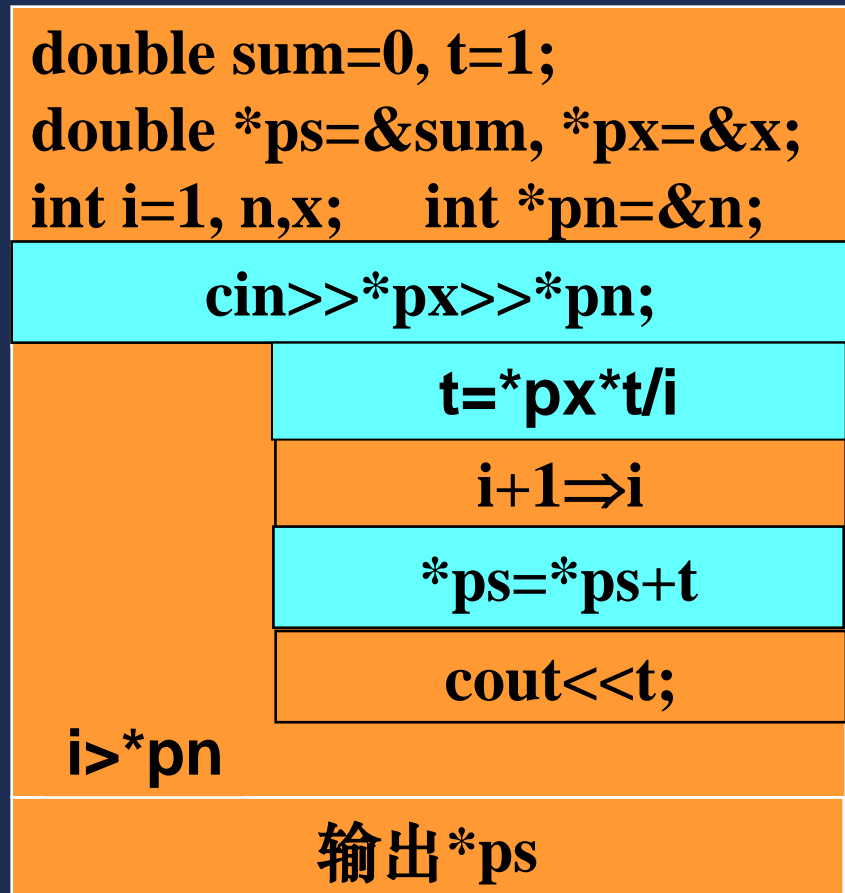
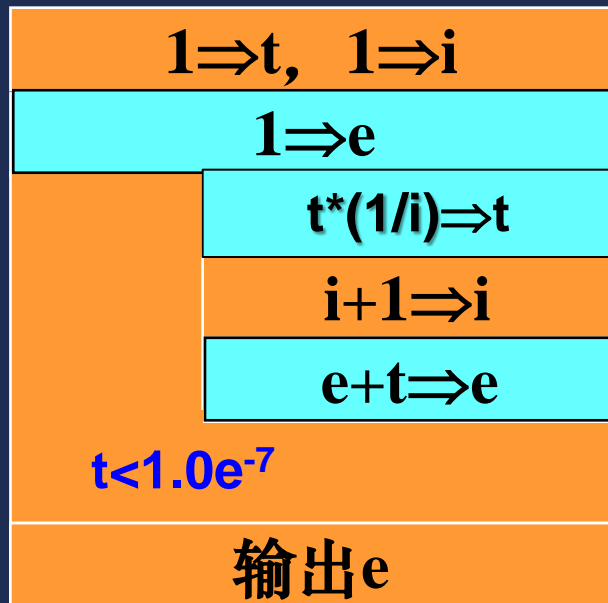
$$sum = \frac{x^1}{1!} + \frac{x^2}{2!} + \cdots \frac{x^n}{n!} + \cdots$$



```
C:\WINDOWS\system32\cmd.exe
请输入x和n的值: 3 5
每一项依次是: 3 4.5 4.5 3.375 2.025
求和结果是: 17.4
请按任意键继续. . .
```

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \dots \frac{1}{n!} + \dots$$

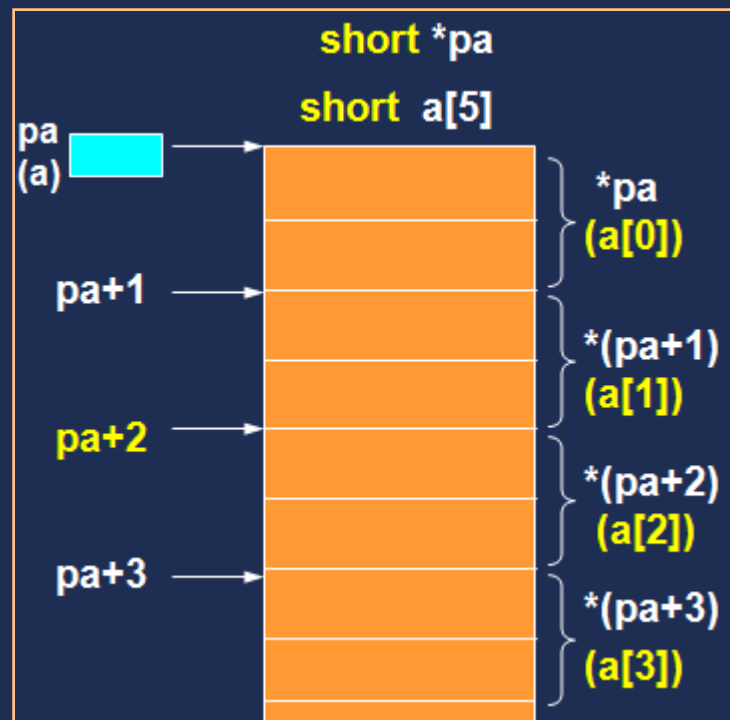
$$sum = \frac{x^1}{1!} + \frac{x^2}{2!} + \dots \frac{x^n}{n!} + \dots$$



# 通过指针引用数组元素

对数组中第 $i+1$ 个元素可表示成以下四种：

- $a[i]$ ,  $*(pa+i)$ ,  $*(a+i)$ ,  $pa[i]$  都是等效的。这四个等价关系使得数组与指针相互转换非常灵活。
- $*pa$  就是  $a[0]$ 。
- 不能写  $a++$ , 因为  $a$  是数组首地址是常量。



与上面相对应的有下面四个地址等价关系：

- $\&a[i]$ ,  $pa+i$ ,  $a+i$ ,  $\&pa[i]$  均表示第 $i$ 个数组元素在内存中的地址。

# 应用举例

在C++中，有了指针和地址的概念，在操作数组时，就可以用如下的四种方法来操作数组。

- 使用数组名和下标 (**a[i]**)
- 使用指针变量的下标表示法 (**pa[i]**)
- 使用数组名和指针运算 (**\*(a+i)**)
- 使用指针变量 (**\*(pa+i)**)

```
#include <iostream>
using namespace std;
void main( )
{ int a[10];
  int *p, i;
```

设有一个int型数组a，有10个元素。用上述四种方法访问数组的各个元素。

前4种方法有元素的下标信息，如果需要处理位置的问题可选用其中任何一种，第5种方法没有下标信息，通过移动指针操作所有数组元素。在进行下一次处理时，需要回溯指针到数组开始位置，即p=a;

```
for(i=0; i<10; i++)
    cin>>a[i]; //①数组名和
```

```
p=a;
for(i=0; i<10; i++)
    cout<<p[i]; //②指针变量访问数组元素
```

```
for(i=0; i<10; i++)
    cout<<*(a+i); //③数组名和指针运算访问数组
```

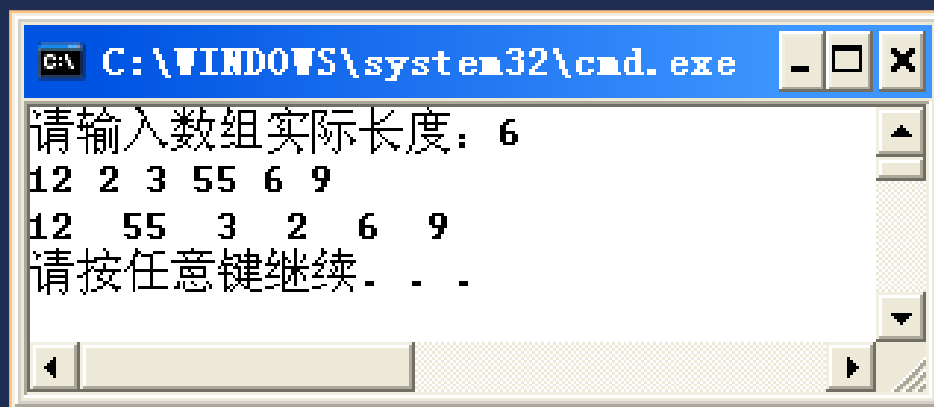
```
p=a;
for(i=0; i<10; i++)
    cout<<*(p+i); //④数组名和指针运算访问数组
```

```
for(p=a; p<a+10; p++)
    cout<<*p; //⑤使用指针变量访问数组 (重点掌握)
p=a; //将指针回溯到数组开始位置，以便下一次处理。
```

```
}
```

# 指针操作数组示例

**编程实现：**用指针技术将一维整数（数组元素值互不相同）数组中的最大元素和最小元素位置对调。



```
C:\WINDOWS\system32\cmd.exe
请输入数组实际长度: 6
12 2 3 55 6 9
12 55 3 2 6 9
请按任意键继续. . .
```

**本实验编码要求：**先用数组方法实现算法，把代码保存好，再新建一个项目，在新建项目的源文件中复制刚才编写的代码，然后利用指针法修改访问数组元素的部分。

# 算法(数组)

定义数组，定义变量

从键盘接收数组元素的值

max=a[0],min=a[0], m=0,n=0;

i=1

i<n

T a[i]>max F

max=a[i]

j=i;

T a[i]<min F

min=a[i]

m=i;

i=i+1

交换a[j] ↔ a[m]

# 程序

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

```
{
```

```
    const int N=10;
```

```
    int a[N];           //声明数组
```

```
    int n,i,min,max,j,m;
```

```
    cout<<"请输入数组实际长度：";
```

```
    cin>>n;
```

```
    for(i=0;i<n;i++) cin>>a[i];
```

```
    max=a[0]; min=a[0];
```

```
    m=0; j=0;
```

```
    for(i=1;i<n;i++) //从第二个元素开始比较
```

```
    {
```

```
        if(a[i]>max)
```

```
        { max=a[i]; //记录最大值
```

```
          j=i; //记录最大元素的下标 }
```

```
        if(a[i]<min)
```

```
        { min=a[i]; //记录最大值
```

```
          m=i; //记录最大元素的下标 }
```

```
    }
```

```
    交换a[j] ↔ a[m]
```

```
}
```

先参照算法给出数组实现方法，  
再用指针表示算法中的数组元素

## 程序(数组)

```
#include <iostream>
using namespace std;
void main( )
{
    const int N=10;
    int a[N];
    int n,i,min,max,j,m,t;
    cout<<"请输入数组实际长度: ";
    cin>>n;
    for(i=0;i<n;i++) cin>>a[i];
    max=a[0]; min=a[0];
    m=0; j=0;
    for(i=1;i<n;i++)
    {
        if(a[i]>max)
        { max=a[i];
          j=i; }
        if(a[i]<min)
        { min=a[i];
          m=i; }
    }
    t=a[j]; a[j]=a[m]; a[m]=t;
}
```

## 程序(指针)

```
#include <iostream>
using namespace std;
void main( )
{
    const int N=10;
    int a[N];
    int n,i,min,max,j,m,t;
    for( ;p<a+n;p++)
        cin>>*p;
    p=a; //注意回溯指针
    int *p; //定义指针变量
    p=a; //指针变量指向数组a在内存中的首地址
    cout<<"请输入数组实际长度: ";
    cin>>n;
    for(i=0;i<n;i++) cin>>*(p+i); //可以方法④或⑤
    max=*p; min=*p;
    m=0; j=0;
    for(i=1;i<n;i++)
    {
        if(*(p+i)>max)
        { max=*(p+i); j=i; }
        if(*(p+i)<min)
        { min=*(p+i); m=i; }
    }
    t=*(p+j); *(p+j)=*(p+m); *(p+m)=t;
}
```



# 实验十二

3. 学生成绩统计：随机产生30名学生的成绩（0~100之间的整数），存放于一维数组中；输出学生成绩，每行10个数。然后统计并输出90-100、80-89、70-79、60-69、小于60这五个分数段的学生人数。

要求：用指针方式访问数组。程序运行结果如下图：

```
C:\Windows\system32\cmd.exe
学生成绩为：
97  8  60  47  42  84  6  47  90  29
 2  43  60  99  57  0  9  0  49  97
 4  42  78  89  57  87  74  15  3  80
学生成绩统计如下：
分数段  90-100  80-89  70-79  60-69  60以下
人数      4      4      2      2     18
请按任意键继续. . .
```

## 算法(数组)

```
int a[N],c[5]={0}; int i,
```

随机产生数组的元素

输出数组元素，每行10个元素

**i=0**

**i<n**

```
switch(a[i]/10)
```

```
{
```

```
case 10:
```

```
case 9: c[0]++; break;
```

```
case 8: c[1]++;break;
```

```
.....
```

```
default:
```

```
}
```

输出成绩统计结果c[i];

## 程序 (数组)

```
#include <iostream>
```

```
#include <ctime>
```

```
using namespace std;
```

```
void main( )
```

```
{
```

```
const int N=30;
```

```
int a[N], c[5]={0}; //数组c统计各分数段人数
```

```
int i;
```

```
cout<<"请输入数组实际长度: ";
```

```
cin>>n;
```

```
srand(time(NULL));
```

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

```
{ a[i]=rand()%(100-1);
```

```
//每行输出 10个元素
```

```
cout<<a[i]<<" ";
```

```
}
```

```
for(i=0;i<n;i++) //处理每个元素
```

```
{
```

```
//switch语句
```

```
}
```

```
输出数组c的所有元素
```

```
}
```

先参照算法给出数组实现方法，  
再用指针表示算法中的数组元素

## 实验十二

4. 编程将一维数组中保存的10个整数循环右移或循环右移m位，要求数组用指针方式控制

```
C:\WINDOWS\system32\cmd.exe
请输入数组的实际长度:
10
13 38 11 39 21 47 19 24 23 21
请输入右移的位数:
5
47 19 24 23 21 13 38 11 39 21
请按任意键继续. . .
```

**思路分析：** 假设原数组序列为9876501234，要求变换成的数组序列为0123498765，即循环右移了5位。仔细观察循环右移的特点，不难发现：每个元素右移N位后都会回到自己的位置上。因此，如果 $m > N$ ，右移 $m-N$ 之后的数组序列跟右移K位的结果是一样的。进而可得出一条通用的规律：**右移m位之后的情形，跟右移 $m' = m \% N$ 位之后的情形一样。**

# 循环右移m位示意图

$m=15 \longrightarrow k=m\%N=15\%10=5$  移动k次  
for(; k>0;k--)

9	8	7	6	5	0	1	2	3	4
---	---	---	---	---	---	---	---	---	---

每次移动处理：

$t=a[N-1]=4$	9	8	7	6	5	0	1	2	3	4
--------------	---	---	---	---	---	---	---	---	---	---

$t=a[N-1]=3$	4	9	8	7	6	5	0	1	2	3
--------------	---	---	---	---	---	---	---	---	---	---

右移1位的结果

$t=a[N-1]=2$	3	4	9	8	7	6	5	0	1	2
--------------	---	---	---	---	---	---	---	---	---	---

右移2位的结果

$t=a[N-1]=1$	2	3	4	9	8	7	6	5	0	1
--------------	---	---	---	---	---	---	---	---	---	---

右移3位的结果

$t=a[N-1]=0$	1	2	3	4	9	8	7	6	5	0
--------------	---	---	---	---	---	---	---	---	---	---

右移4位的结果

0	1	2	3	4	9	8	7	6	5
---	---	---	---	---	---	---	---	---	---

右移5位的结果

```
const int N=10;  
int a[N],i,j,k,m,t;
```

输入数据a[0]至a[N-1]

输入循环右移的位数m

k=m%N

while(k--)

t=a[N-1];

for(i=N-1; i>0; i--)

a[i]=a[i-1] /\*第N-1个(倒数第2  
个)~第1个元素依次往后移动1位\*/

a[0]=t;

输出数组a的值

# 循环右移

```
const int N=10;  
int a[N],i,j,k,m,t;
```

输入数据a[0]至a[N-1]

输入循环右移的位数m

```
k=m%N
```

```
while(k--)
```

```
t=a[0];
```

```
for(i=0; i<N-1; i++)
```

```
a[i]=a[i+1] /*从第2个元素到最  
最后一个元素依次往前移动1位*/
```

```
a[N-1]=t;
```

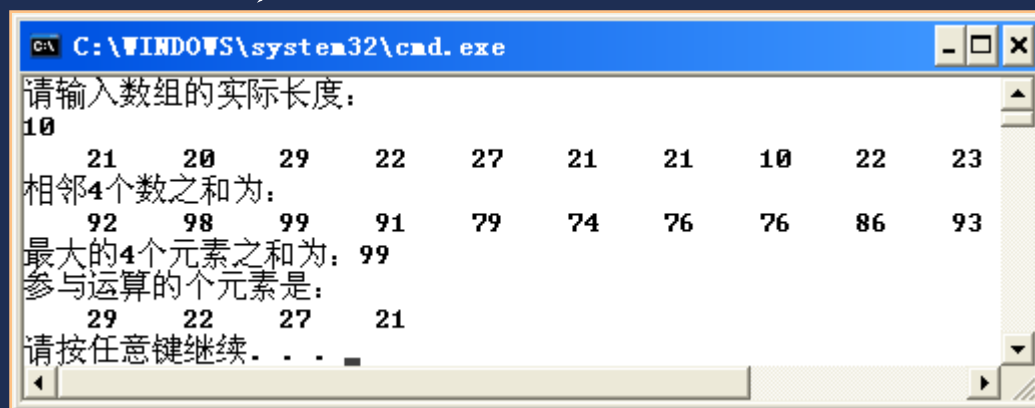
输出数组a的值

# 循环左移

## 实验十二

5.5. 有一组整数，所有元素随机产生。把数组的最后一个元素和第一个元素视为首尾相连，构成一个环形。编程求出相邻的任意四个数之和，并输出。然后找出相加之和最大的四个数，输出求和的结果以及这四个数。。

要求：用指针方式访问数组。程序运行结果类似于下图，实验报告中的运行结果截图，不能与下图中的数据一样。



```
C:\WINDOWS\system32\cmd.exe
请输入数组的实际长度:
10
    21    20    29    22    27    21    21    10    22    23
相邻4个数之和为:
    92    98    99    91    79    74    76    76    86    93
最大的4个元素之和为: 99
参与运算的个元素是:
    29    22    27    21
请按任意键继续. . .
```

如何理解：首尾相连，构成一个环形？

设数组有10个元素，则下标表示为 $a[i\%10]$ ，例如，最后一个元素，其下标为9，其再下一个元素就是 $a[10\%10]=a[0]$ ，即第1个元素。

```
int a[N]; int i,j,n,max,s;
```

输入数组实际长度

随机产生数组的元素并输出数组元素

```
s=a[0]+a[1]+a[2]+a[3]; max=s;
```

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

```
    s=a[i%10]+a[(i+1)%10]  
    +a[(i+2)%10]+a[(i+3)%10];
```

```
    cout<<setw(6)<<s;
```

```
    T          s>max          F
```

```
        max=s;
```

```
        j=i;
```

```
    cout<<"最大的4个数之和:"<<max;
```

```
cout<<"参与运算的4个元素是: "<<endl;  
cout<<setw(6)<<a[j%10]<<setw(6)<<  
a[(j+1)%10]<<setw(6)<<a[(j+2)%10]  
<<setw(6)<<a[(j+3)%10];
```

只用一个数组，计算结果不保存

```
int a[N],b[N]={0}; int i,j,m=4,n,max;
```

输入数组实际长度

随机产生数组的元素并输出数组元素

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

```
    for(j=0;j<m;j++)
```

```
        b[i]=b[i]+a[(i+j)%n];
```

输出数组b中各个元素;

```
    max=b[0];
```

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

```
    T          b[i]>max          F
```

```
        max=b[i];
```

```
        j=i;
```

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

```
    cout<<a[(i+j)%n];
```

使用两个数组，计算结果保存在b中

先参照算法给出数组实现方法，再用指针表示算法中的数组元素