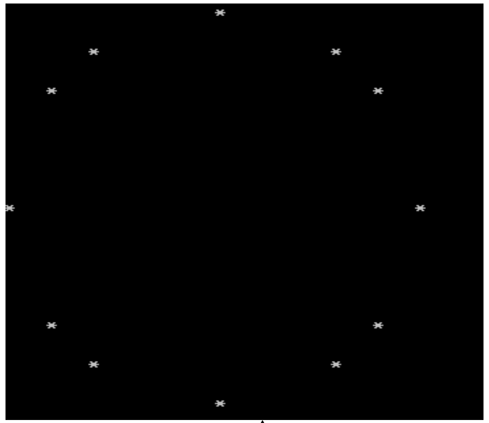
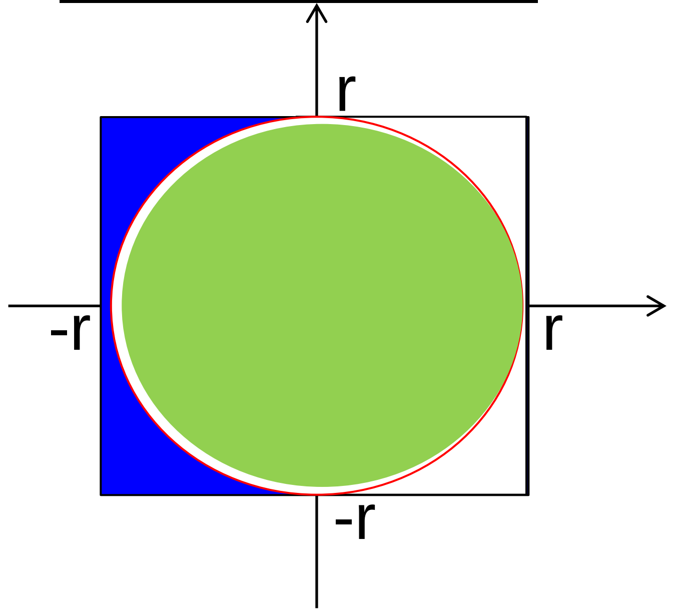
复习-1

在控制台绘制空心圆：流程控制

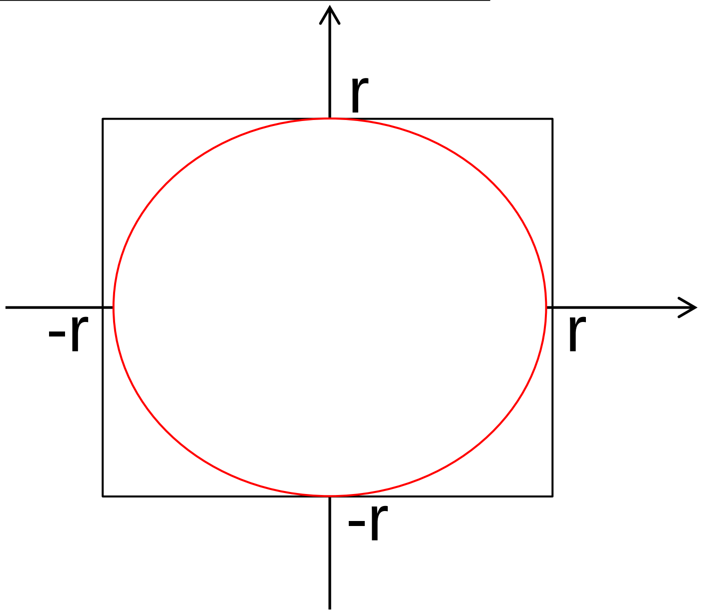
```
#include "cmath"
#define T 2.2

for(double y=r; y >= -r; y--)
{
     double i, x = T*sqrt(r*r - y*y);
     for(i = -r*T; i < -x; i++)
          cout << " ";    //蓝色部分
     cout << "*";         //左半圆

     for(; i < x; i++)
          cout << " ";    //绿色部分
     cout << "*" << endl;      //右半圆
}
```



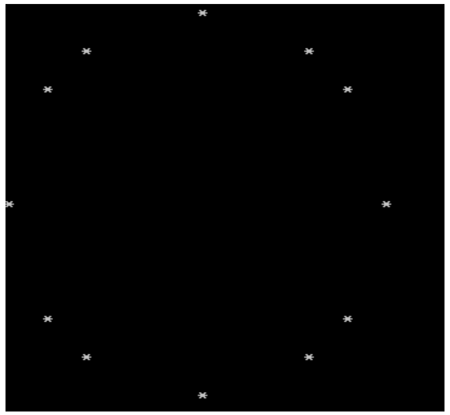


注意：
变量的作用域

```
for(double x=-r*T; x <= r*T; x++)
    if(x*x/T/T+y*y - r*r <= 0 && x*x/T/T+y*y - r*r > -10)
```



```
for(double y=r; y >= -r; y--)
{
     for(double x=-r; x <= r; x++)
          if(x*x+y*y == r*r)
               cout << "*";    //红色部分的圆
          else
               cout << " ";    //白色部分
     cout << endl;
}
```



注意：
除法操作符，数据类型
关系操作的边界问题

回文正整数：函数

- Was it a car or a cat I saw
- wasitacaroracatisaw

```
isPalindromeStr(a)
```

```
bool isPalindromeStr(const char str[])
{
    unsigned int length = strlen(str);
    for(int i=0, j=length-1; i < j; ++i, --j)
        if(str[i] != str[j])
            return false;
    return true;
}
```

回文正整数：递归

- Was it a car or a cat I saw
- wasitacaroracatisaw

```
isPalindromeStr(a, 0, strlen(a)-1)
```

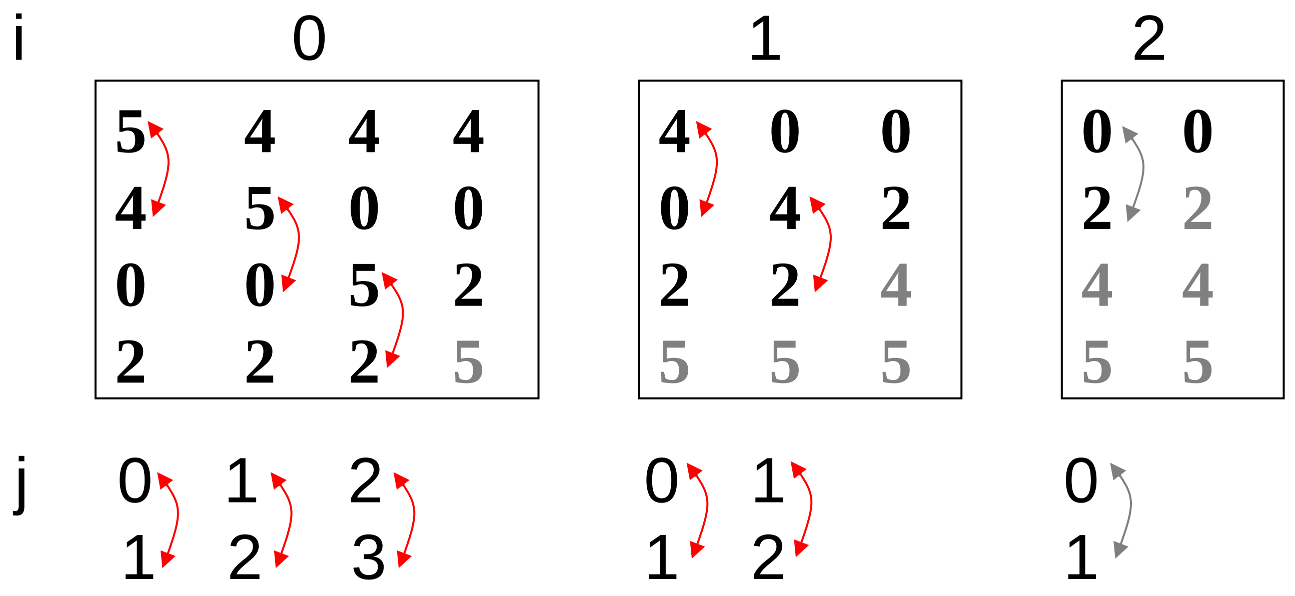
```
bool isPalindromeStr(const char str[], int i, int j)
{
     if(i >= j) return true;
     else
          if(str[i] != str[j])
               return false;
          else
               return isPalindromeStr(str, ++i, --j);
}
```

汉诺塔问题：个别答疑

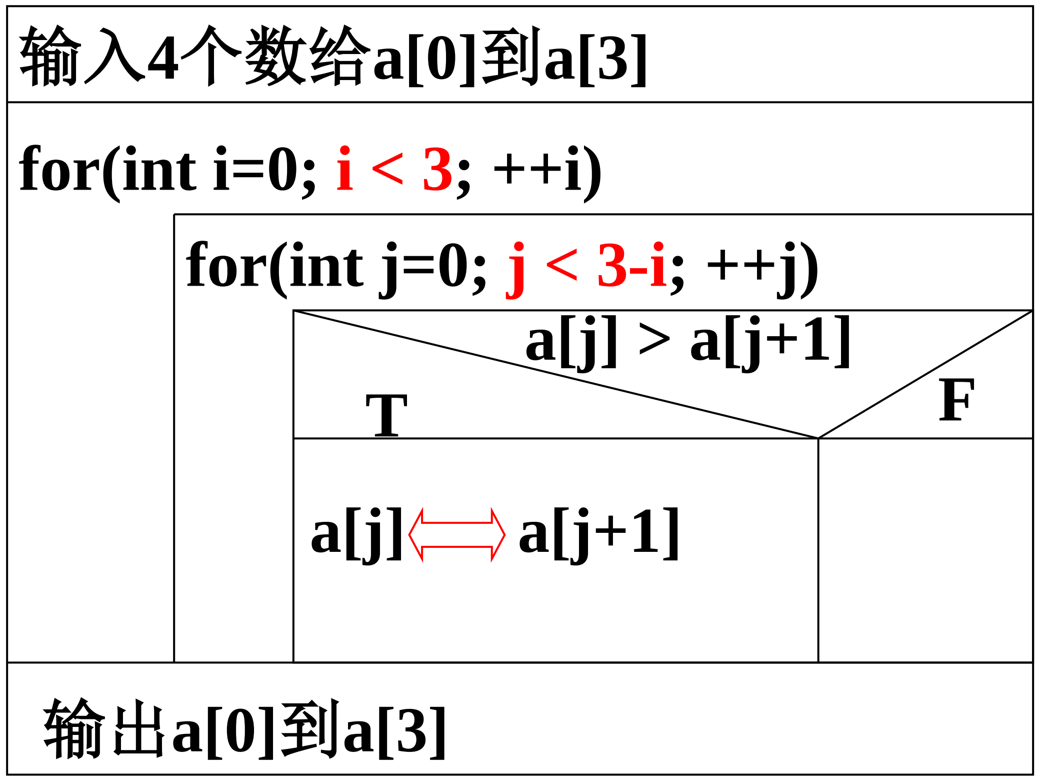
- 数组
  - **存储一组数**
  - 存储一组数对应的状态

冒泡法排序（升序）

- 比较相邻两个数，小的调到前头
- **4个数排3趟，每趟内比较的次数随趟数递减。**



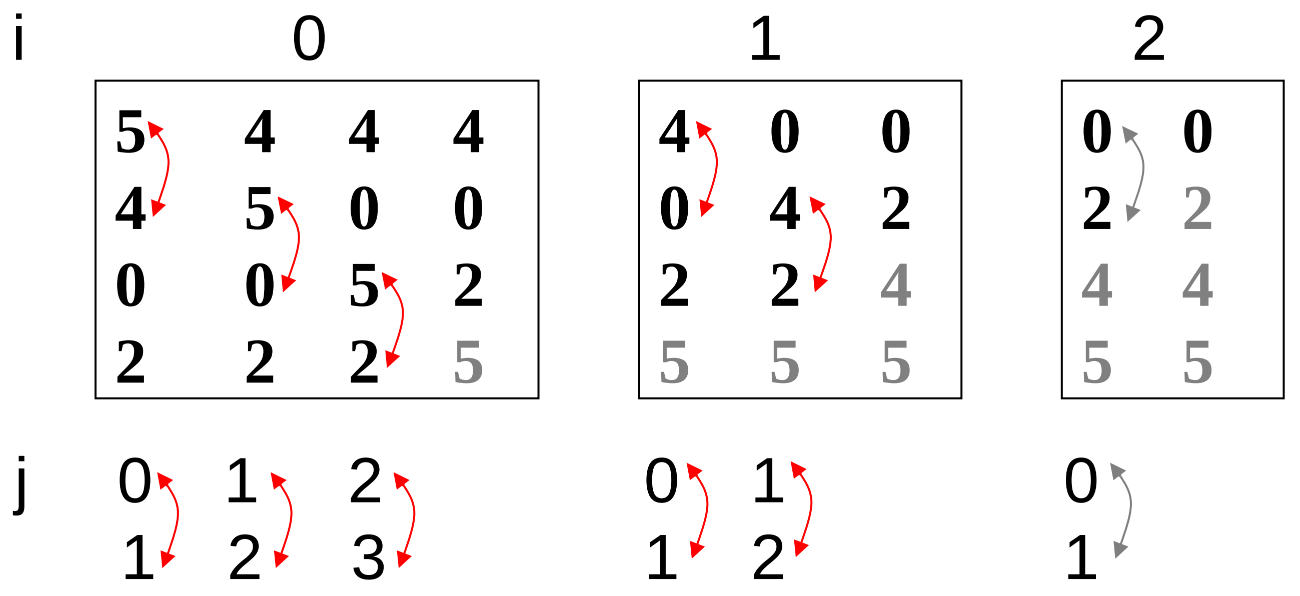
int a[4]



```
输入4个数给a[0]到a[3]
for(int i=0; i < 3; ++i)
    for(int j=0; j < 3-i; ++j)
        a[j] > a[j+1]
        T: a[j]⟺a[j+1]
        F:
输出a[0]到a[3]
```

冒泡法排序（升序）

- 比较相邻两个数，小的调到前头
- **4个数排3趟，每趟内比较的次数随趟数递减。**



int a[4]

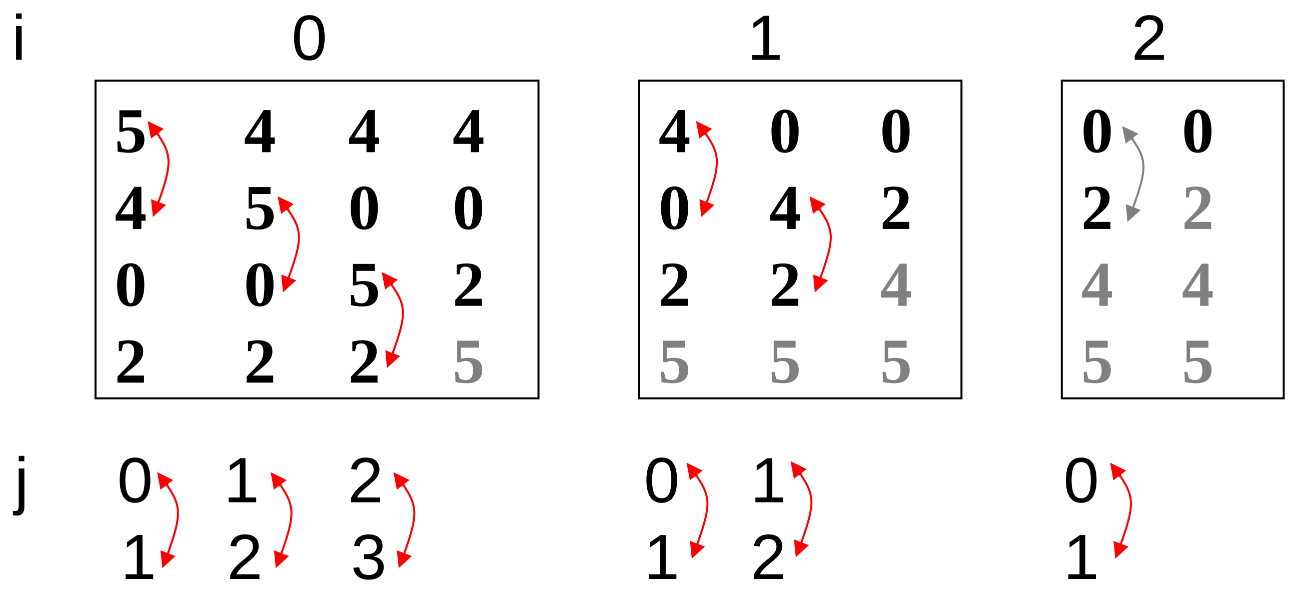
输入4个数给a[0]到a[3]

```
for(int i=0; i < 3; ++i)
        for(int j=0; j < 3-i; ++j)
              if(a[j] > a[j+1])

                    a[j]⟺a[j+1]
```

输出a[0]到a[3]

冒泡法排序（升序）

- 比较相邻两个数，小的调到前头
- N个数排N-1趟，每趟内比较的次数随趟数递减。



```
int a[N]

输入N个数给a[0]到a[N-1]

for(int i=0; i < N-1; ++i)
        for(int j=0; j < N-1-i; ++j)
            if(a[j] > a[j+1])

                a[j]⟺a[j+1]

输出a[0]到a[N-1]
```

```
#define N 4
int main( )
{
  int a[N];
  for(int i=0; i < N; ++i)
     cin >> a[i];
```

```
for(int i=0; i < N-1; ++i)
    for(int j=0; j < N-1-i; ++j)
        if(a[j] > a[j+1])
        {
          int temp = a[j];
          a[j] = a[j+1];
          a[j+1] = temp;
        }//交换
```

```
  for(int i=0; i < N; ++i)
     cout << a[i] << '\t';
  return 0;
}
```

涉及的程序设计要素：
- 数组
- 循环、分支流程控制
- 赋值、比较、算术操作

i 、j 的作用域

```
#define N 4
int main( )
{
  int a[N], i, j;
  for(i=0; i < N; ++i)
     cin >> a[i];



  for(i=0; i < N; ++i)
     cout << a[i] << '\t';
  return 0;
}
```

```
for(i=0; i < N-1; ++i)
    for(j=0; j < N-1-i; ++j)
        if(a[j] > a[j+1])
        {
          int temp = a[j];
          a[j] = a[j+1];
          a[j+1] = temp;
        }//交换
```

- **数组**
  - 存储一组数
  - **存储一组数对应的状态**

约瑟夫斯（Josephus）问题

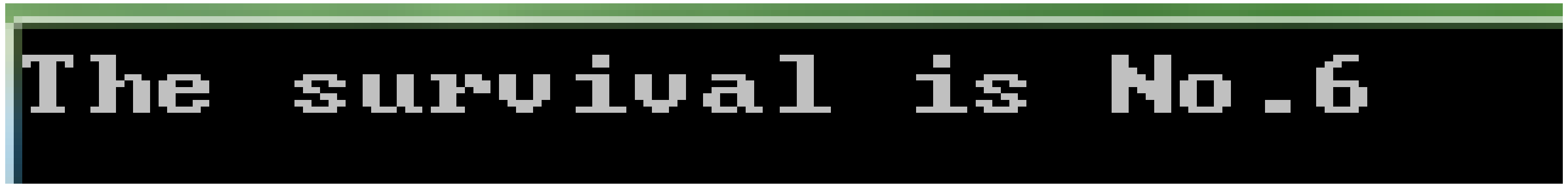
```
…
#define N 20
#define K 5
int Josephus(int n, int k);
int main( )
{
  cout << "The survival is No." << Josephus(N, K) << endl;
  return 0;
}
```

- 分析：
  - bool in_circle[n];
  - in_circle[index]为true表示编号为index的囚犯在圈子里
  - 剩下的人数 numRemained: n → 1
  - 从index为0的囚犯开始报数，圈子中index的下一个位置为(index+1)%n

```
int Josephus(int n, int k)
{
  bool in_circle[n];
  int index;

  for(index = 0; index < n; ++index)
     in_circle[index] = true;        //初始化数组in_circle
```

```
int numRemained = n;
index = ~~0~~; n-1
while(numRemained > 1)
{
   int count = 0;
   while(count < k)
   {
      index = (index+1)%n;
      if(in_circle[index])
         count++;
~~      index = (index+1)%n;~~
   }
   in_circle[index] = false;  //囚犯离开圈子
   numRemained--; //圈中人数减1
}
```

```
  //找最后一个囚犯
  for (index = 0; index < n; index++)
     if (in_circle[index])
          break;
  //cout << "The survival is No." << index << endl;
  return index;
}
```



```
The survival is No.6
```

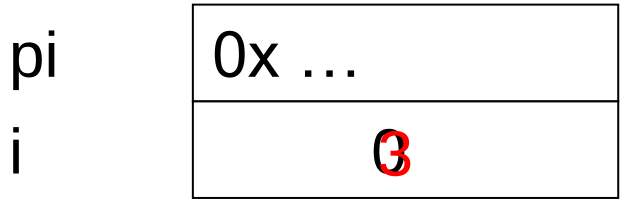
- **指针**
  - 地址常量
  - 指针数据类型
  - 指针类型的变量

原理

类型关键字

- 定义一个指针变量，并初始化

```
int i = 0;
int  *pi = &i;
*pi = 3;
```



```
int a[10] = {0};
int  *pa = a;

int  *pv = 0;
```

典型用法一：作为函数的参数

```
struct Stu
{     int no;
      char name[20];
      int age;
};

void f(Stu t)
{
      cout << t.no;
      cout << t.name;
      cout << t.age;
}
```

```
int main()
{
      Stu s;
      cin >> s.no >> s.name >> s.age;
      f(&s);
      return 0;
}
```

```
Stu *p = &s;
```

```
void f(Stu *p)
{
      cout << p -> no;   //(*p).no
      cout << p -> name;//(*p).name
      cout << p -> age; //(*p).age
}
```

副作用及其避免

```
struct Stu
{       int no;
        char name[20];
        int age;
};

void f(Stu t)
{
        cout << t.no;
        cout << t.name;
        cout << t.age;
        t.no++;
}
```

```
int main()
{
        Stu s;
        cin >> s.no >> s.name >> s.age;
        f(&s);
        return 0;
}
```

```
void f(const Stu *p)
```

```
void f(Stu *p)
{
        cout << p -> no;   //(*p).no
        cout << p -> name;//(*p).name
        cout << p -> age; //(*p).age
```
~~p -> no++;~~
```
}
```

const的限制作用

必须初始化

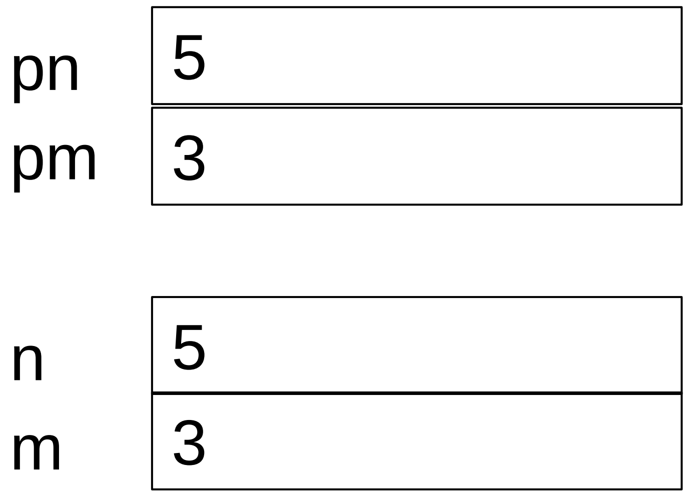
```
int n;
 const int M = 0;      //m 是常量
 const int *p1;  //*p1 是常量，  不过 p1 = &n 也可以


 int * const P2 = &n; // P2是常量

 const int * const P3 = &M;     //*P3，  P3是常量，  不过const
 int * const P3 = &n;    也可以
```

副作用及其利用

无法交换

```
int m = 3;
int n = 5;
MySwap(m, n);
```

```
void MySwap(int pm, int pn)
{
      int temp = pm;
      pm = pn;
      pn = temp;
}
```

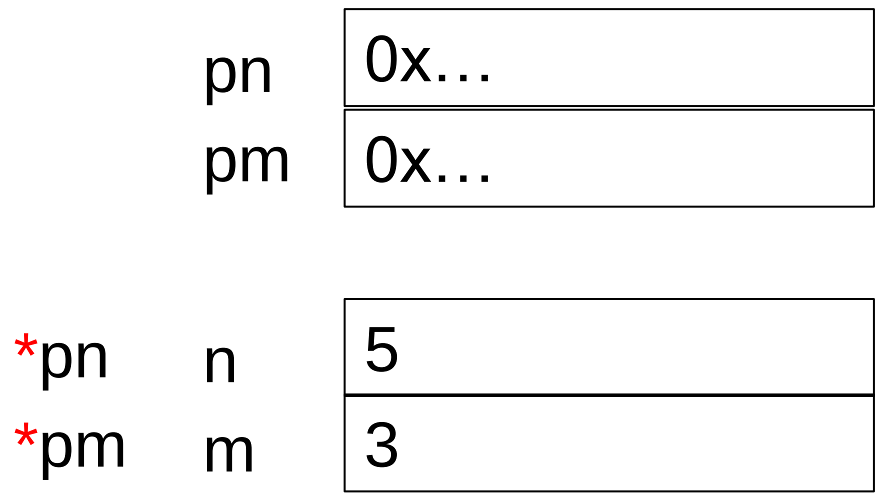


副作用及其利用

可以交换

```
int m = 3;
int n = 5;
MySwap(&m, &n);
```

```
void MySwap(int *pm, int *pn)
{
      int temp = *pm;
      *pm = *pn;
      *pn = temp;
}
```



第九次作业-用指针实现

不重要，一般不必用指针操纵数组

```
#define N 4
int main( )
{     int a[N];
      for (int i=0; i<N; ++i)
           cin >> a[i];

      BubbleSort(a, N);

      …
}
```

```
void BubbleSort(int *pa, int count)
 {
     for(int i = 0; i < count-1; ++i)
          for(int j = 0; j < count-1-i; ++j)
               if(pa[j] > pa[j+1])
               {
                    int temp = pa[j];
                    pa[j] = pa[j+1];
                    pa[j+1] = temp;
               }
}
```

第九次作业-用指针实现

不重要，一般不必用指针操纵数组

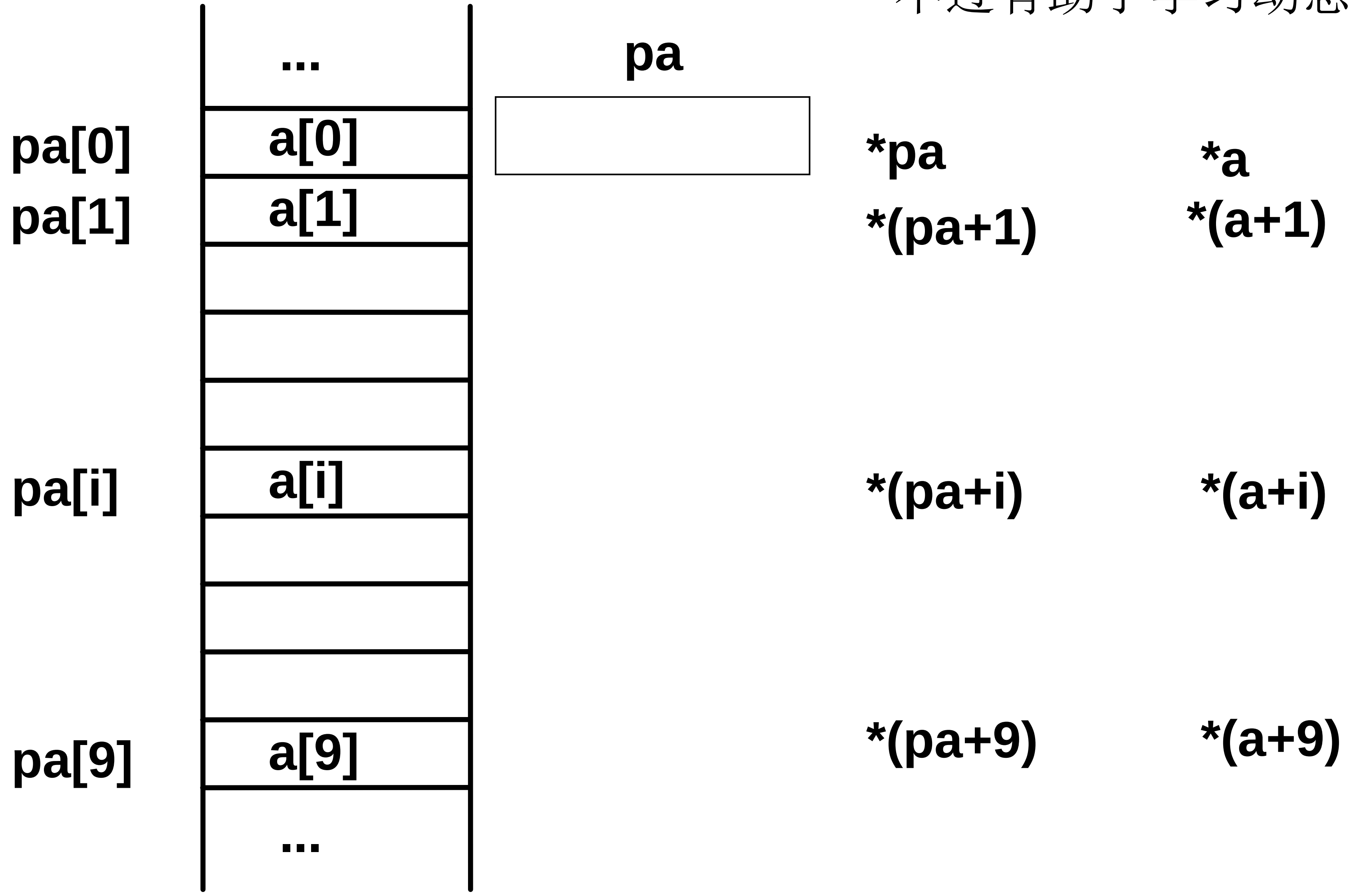
```
#define N 4
int main( )
{     int a[N];
      for (int i=0; i<N; ++i)
           cin >> a[i];

      BubbleSort(a, N);

      …
}
```

```
int *pa = a; //int *pa = &a[0];
```

```
void BubbleSort(int *pa, int count)
 {
     for(int i = 0; i < count-1; ++i)
          for(int j = 0; j < count-1-i; ++j)
               if(*(pa+j) > *(pa+j+1))
               {
                    int temp = *(pa+j);
                    *(pa+j) = *(pa+j+1);
                    *(pa+j+1) = temp;
               }
}
```

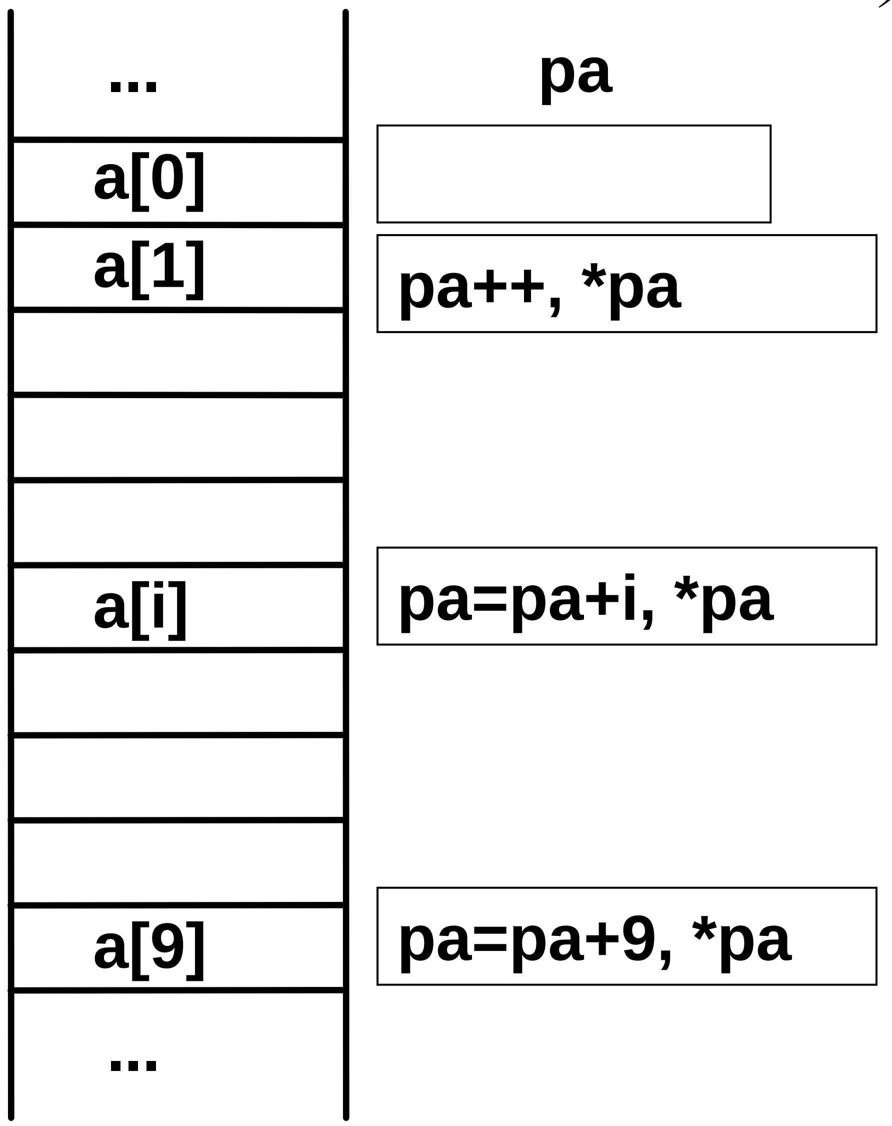
a

不过有助于学习动态数组



a

不过有助于学习动态数组



二维数组的指针*

不过有助于学习二维动态数组

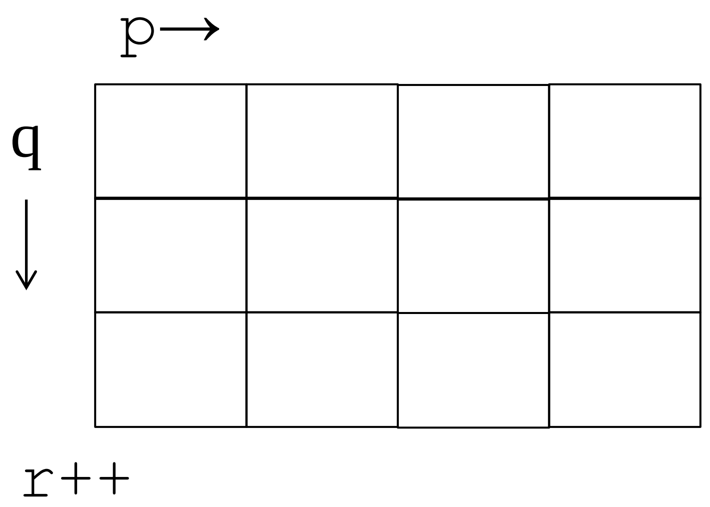
```
int b[5][10];
int *p;
p = &b[0][0];//或“p = b[0];”
```

$\boxed{\text{第一行某个元素 p[j]}}$

```
int (*q)[10];
q = &b[0];   //或“q = b;”
```

$\boxed{\text{某个元素 q[i][j]}}$

```
int (*r)[5][10];
r = &b;
```



典型用法二：操纵动态变量或动态数组

```
int *pd = new int;
*pd = 3;
cout << endl << *pd << endl;

int *pda = new int[5];
for(int i=0; i < 5; ++i, ++pda)
     cin >> *pda;

pda -= 5;
for(int i=0; i < 5; ++i, ++pda)
     cout << *pda << ", ";
```

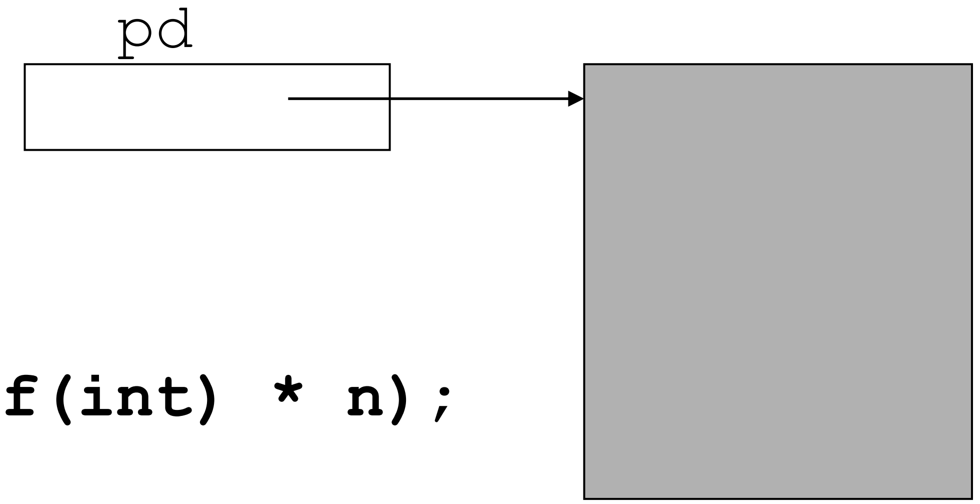
```
for(int i=0; i < 5; ++i)
    printf("%d, ", *pda++);
```

```
int (*pdaa)[10] = new int[n][10];
```

二维动态数组：个别答疑

动态变量的撤销

```
int *pd = new int;
delete pd;

int *pd = new int[n];
delete []pd;



int *pd = (int *)malloc(sizeof(int) * n);
free(pd);
```



内存泄露与悬浮指针*

```
int *pda;
int m;
pda = new int[n];
……
                    //应该在使用之后释放动态空间
pda = &m;
```

pda所指向的动态空间没有释放，但无法访问，泄漏了

pda所指向的动态空间释放了，不知道会分配给谁，
但pda 里存储的还是该动态空间的首地址

```
int *pda;
pda = new int[n];
……
delete []pda;
                  //应该清理 pda 里的地址，以免乱指
```

指针类型返回值：一般用来返回一组数据 链表

复习-2

```
Node *InsCreate( )
{
      Node *head = NULL;

      for(int i = 0; i < N; i++)
      {     Node *p = new Node;
            p -> data = i;
            p -> next = head;
            head = p;
      }
      return head;
}
```

```
int main( )
{
      Node *h = InsCreate( );
      PrintList(h);
      ……
```

```
int main( )
{
      Node *h = NULL;
      h = InsCreate( );
      PrintList(h);
      ……
```

不要返回局部变量的地址：个别答疑

函数指针：个别答疑

指针类型返回值：一般用来返回一组数据 字符串

```
char *strCpy(char *dst, const char *src)
{
     int i;
     for (i = 0; src[i] != '\0'; ++i)
           dst[i] = src[i];
      dst[i] = '\0';
     return dst;
}
```

```
strCpy(str, "NJU");
cout << str;

cout << strCpy(str, "NJU");
```

常用字符串库函数

```
unsigned int strlen(const char *s);
// int len = strlen(str);

char *strncpy(char *s1, const char *s2, int n);
// char *str = strncpy(str, "nju", 2);

char *strcat(char * s1, const char * s2);
char *strncat(char * s1, const char * s2, int n);
int strcmp(const char *s1, const char *s2);

int strncmp(const char *s1, const char *s2, int n);
//if( strncmp(str, "nju", 2) == 0 )说明 str 前两个字符为 nj
```

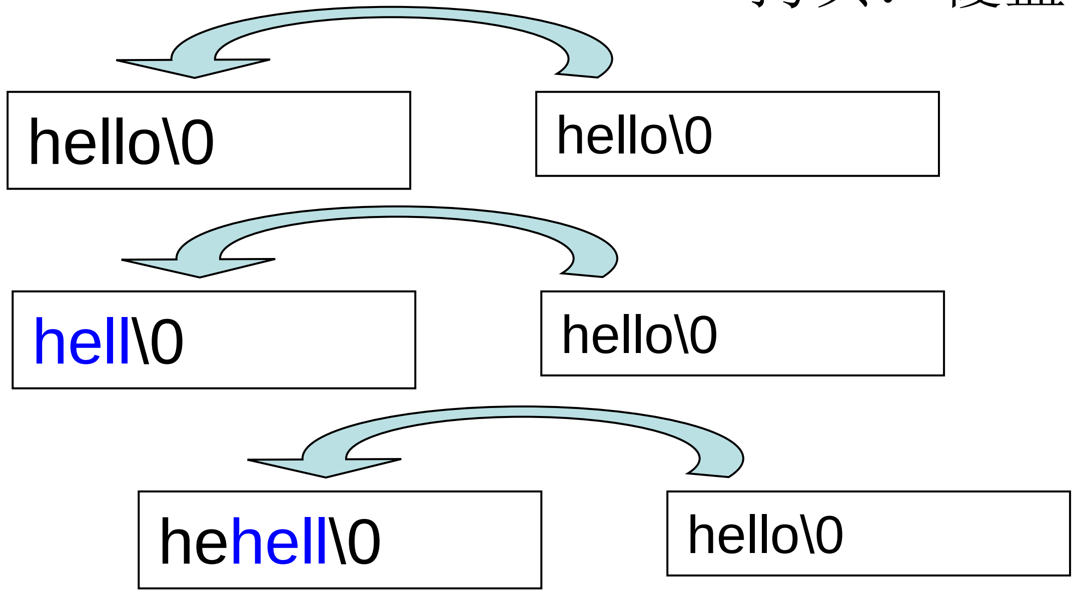
sqrt
fabs
pow
rand
srand

新标准下的常用字符串库函数

结果字节数（含'\0'）

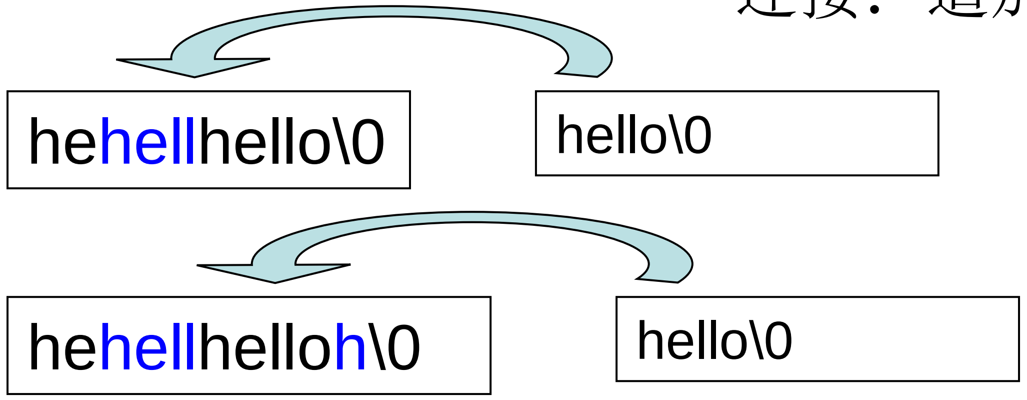
```
strcpy_s(dstr, 6,  "hello")

strncpy_s(dstr, 5,  "hello", 4)

strncpy_s(dstr+2, 7,  "hello", 4)
```

拷贝：覆盖



```
strcat_s(dstr, 12,  "hello")

strncat_s(dstr, 8,  "hello", 1)
```

连接：追加



字符的输入

输入单个字符

```
    char ch;
    cin >> ch;
    ch = getchar( );
    scanf("%c", &ch);
    scanf_s("%c", &ch, 1);
```

字符串的输入

输入字符串

```
char str[10];
cin >> str;     //空格等空白符之后的字符忽略
gets(str);
gets_s(str, 5) //最多可输入4个字符
cin.getline(str,9);
cin.get(str, 9);
scanf("%s", str)
scanf_s("%s", str, 5) //最多可输入4个字符
```

字符与字符串的输出

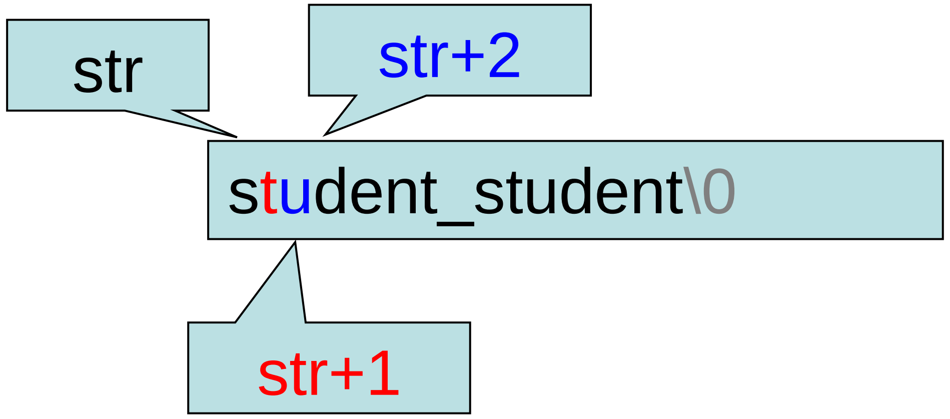
```
printf("%c \n", ch);
printf("%s \n", str);

printf("%x. \n", str); //输出地址
```

```
cout << ch << endl;
cout << str << endl;
cout << *str << endl;

cout << (void *)pstr;
```

```
ABCD
A
0x22ff50
```

字符型地址，默认输出字符串

```
char str[] = "student_student";
cout << str << endl;  //输出整个数组的字符，直到'\0'
cout << str+1 << endl;
cout << str+2 << endl;
```

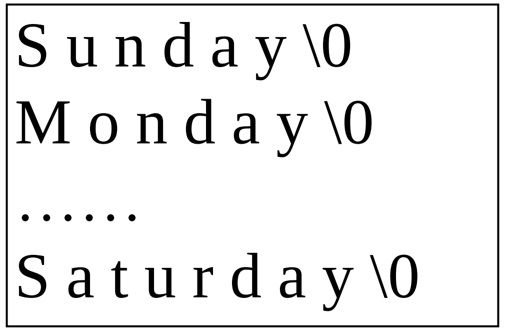


```
char ch = 'c';
char *pc = &ch;
cout << pc << endl; //输出以c开头的乱码字符串
```

指针数组

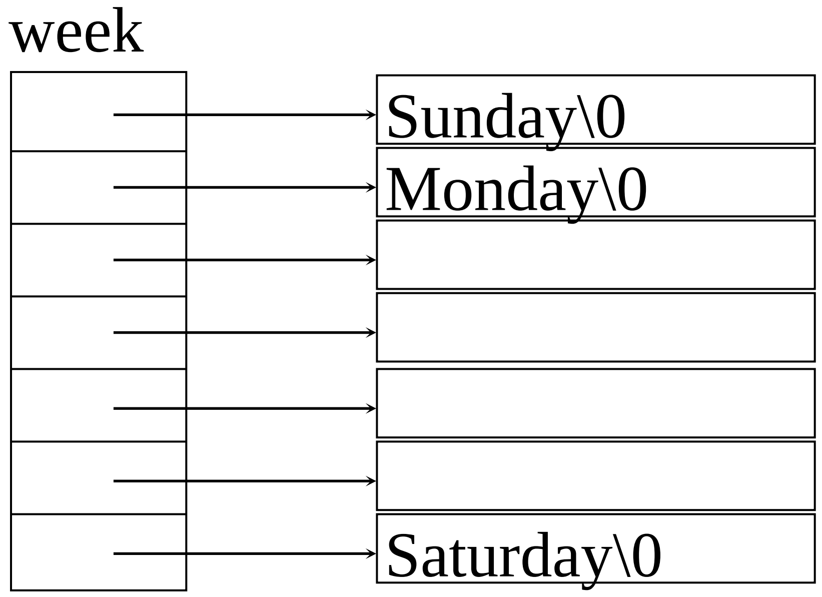
- 二维字符型数组

```
char weekday[7][10] = {"Sunday", "Monday", "…", "Saturday" };
```



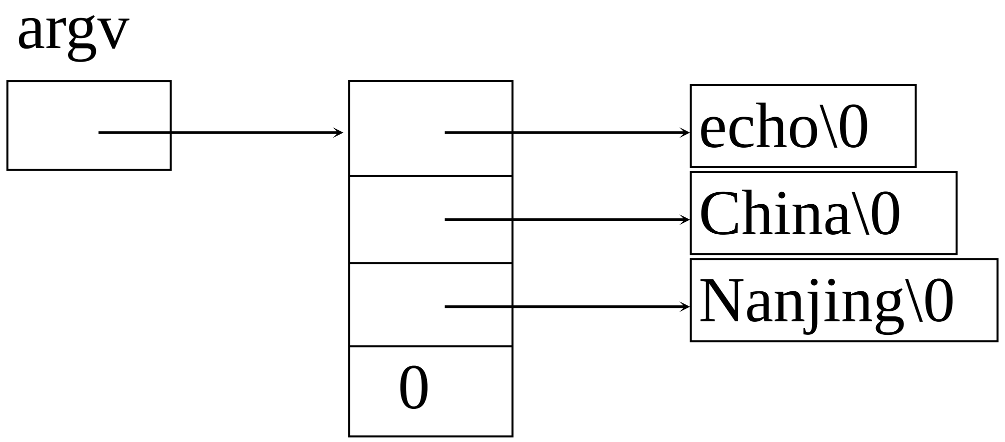
- 字符指针数组

```
char *week[7] = {"Sunday", "Monday", "…", "Saturday"};
```



带形参的 main 函数*

```
#include <stdio.h>
int main(int argc, char *argv[ ])
{
  while(argc > 1)
  {
     ++argv;
     printf("%s \n", *argv);
     --argc;
  }
  return 0;
}
```



Thanks !