C语言

第7讲：数组

赵岩

哈尔滨工业大学软件学院

August 20, 2011

目录

目录

目录

为什么使用数组-问题

问题

要读入某班全体50位同学C语言成绩，然后进行统计处理（求平均成绩、最高分、最低分.....）

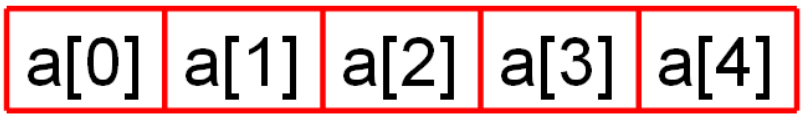
为什么使用数组

- 用简单变量，需50个不同变量名，要用很多个scanf命令
- 用数组，可共用一个scanf命令，并利用循环结构读取。

```
1 /* without array*/
2 int score1,score2,...,score50;
3 scanf("%d,%d,%d",&score1,&score2,&score3);
4 scanf("%d,%d,%d",&score4,&score5,&score6);
5 .........
6 /* use array */
7 int score[50],i; /*优雅*/
8 for (i=0;i<50;i++)
9     scanf("%d",&score[i]);
```

数组(array)定义

- 定义
  - 存储属性 数据类型 数组名 [整型常数1] [整型常数2] ..... [整型常数n];
- `int` a[5];
  - 定义一个有5个元素的数组，每个元素的类型均为`int`
  - 使用a[0]、a[1]、a[2]、...、a[4]这样的形式访问每个元素。它们与普通变量没有任何区别
  - 系统会在内存分配连续的5个`int`空间给此数组
  - 数组下标(index)可以是整型表达式
  - 直接对a的访问，就是访问此数组的首地址



数组的大小

- 数组大小必须是常量表达式(常量或符号常量)，其值必须为正，不能为变量。
- 可以用宏或枚举来定义来定义，以适应未来可能的变化
- 数组大小定义好后，将永远不变

```
1 int a[10]; /*方法0*/
2
3 #define SIZE 10 /*方法1*/
4 int a[SIZE];
5
6 enum {SIZE=10}; /*方法2*/
7 int a[SIZE];
```

数组的初始化

- 数组定义后的初值仍然是随机数，一般需要我们来初始化

```
1  int a[5] = { 12, 34, 56 ,78 ,9 };
2  int a[5] = { 1 }; /*后面的都是零*/
3  int a[] = { 12, 34, 56, 78, 9 };
```

数组的下标-index

- 数组的下标都是从0开始，既可以是常量，也可以是变量
  - 养成从0开始查数的“职业病”
- 下标越界是大忌！
  - 下标越界(编译程序不检查是否越界)，将访问数组以外的空间。那里是什么不受我们掌控，可能带来严重后果

数组下标越界的严重后果

```
1       int i=0;
2       int a[5]={1,2,3,4,5};
3       int end=4; /* 4,5,6 */
4       for(i=0;i<=end;i++)
5           a[i]=0;
6       printf("%d\n",a[4]);
```

数组的存储属性

- 根据数组的数据类型，为每一元素安排相同长度的存储单元根据数组的存储属性，将其安排在动态内存、静态存储区
- sizeof可以用来获得数组所占字节数
  - sizeof(a)
  - sizeof(a[0])

```
1  int a[]={1,2,3,4,5};
2  printf("%d\n",sizeof(a)/sizeof(a[0]));
```

一维数组的输入和输出

- 只能单独对数组元素操作（字符数组例外）

```
1  int a[10],i;
2  scanf("%d",&a[i]); /*input single */
3  printf("%d",a[i]); /*output single*/
4
5  for(i = 0;i<10;i++)
6  {
7      scanf("%d",&a[i]);
8      printf("%d",a[i]);
9  }
```

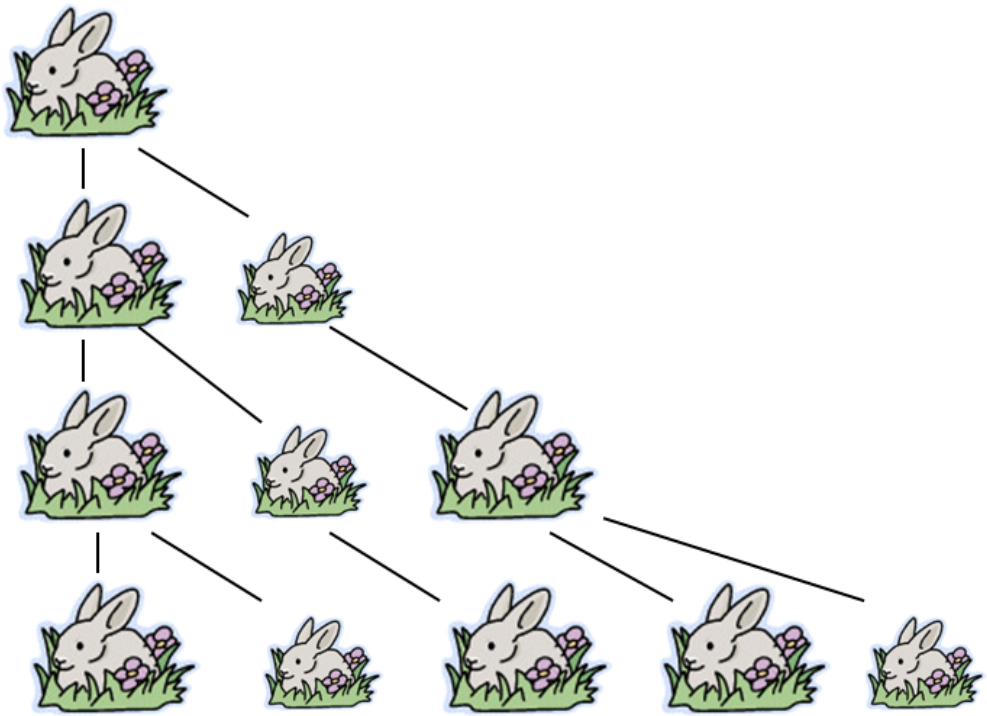
数组使用特点总结

- 数组名表示数组的首地址,其值不可改变!
- 快速地随机访问
- 一旦定义，不能改变大小（小心越界）

```
1  main()
2  {
3      int a[4]={1,2,3,4}, b[4];
4      b=a; /*- error*/
5
6      int i;
7      for (i=0;i<4;i++)
8          b[i]=a[i];
9  }
```

Fibonacci数列

- $f_1 = 1,\ f_2 = 2,\ f_n = f_{n-1} + f_{n-2}$
- $1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55...$



利用数组计算Fibonacci数列

```
1  #define YEAR_MONTH   12
2
3  int   f[YEAR_MONTH+1] = {0,1,2};
4  int   month;
5  for (month=3; month<=YEAR_MONTH; month++)
6  {
7      f[month] = f[month-1] + f[month-2];
8  }
9  for (month=1; month<=YEAR_MONTH; month++)
10 {
11     printf("%d\t", f[month]);
12 }
13 printf("\nsum = %d\n", f[YEAR_MONTH]);
```

查找最高分

- 键盘输入学生人数n;
- 键盘输入所有学生的学号和成绩
- 分别存入数组num和score
- 对所有学生成绩进行比较
- 打印最高分maxScore及其学号maxNum;

```
1  for (i=0; i<n; i++)
2  {
3      若score[i] > maxScore, 则
4      修改maxScore值为score[i],
5      并记录其学号maxNum = num[i];
6  }
```

打印最大值源码

```
1  #define   ARR_SIZE   40
2  void main(){
3       maxScore = score[0];
4       maxNum = num[0];
5       for (i=1; i<n; i++){
6            if (score[i] > maxScore){
7                 maxScore = score[i];
8                 maxNum = num[i];
9            }
10      }
11  printf("maxScore = %.0f, maxNum = %ld\n",
12                              maxScore, maxNum);
13 }
```

从类型的角度理解数组

- int a[10];
  - 定义一个有10个元素的数组，每一个元素都是**int类型**
- int a[10][10];
  - 定义一个有10个元素的数组，每一个元素都是**包含10个int的数组的类型**
  - a[0], a[1],...,a[9]的类型是int[10]，所以a[0][0]...,a[9][9]的类型是int
- int a[30][20][10];
  - 这个呢？
- 此原理决定了数组在内存的分布规律，也解释了数组的很多语法现象

二维数组

- 二维数组
  - 用两个下标来确定各元素在数组中的顺序。可用排列成i行，j列的元素组来表示。
  - 如int b[2][3];
- n维数组
  - 用n个下标来确定各元素在数组中的顺序。
  - 如int c[3][2][4];
  - n≥3时，维数组无法在平面上表示其各元素的位置。

b[0][0]	b[0][1]	b[0][2]
b[1][0]	b[1][1]	b[1][2]

二维数组的输入输出

Demo 1: array example

```
1  int b[2][3],i,j;
2  for(i = 0;i<2;i++)
3      for (j = 0;j<3;j++)
4      {
5          scanf("%d",&b[i][j]);
6          printf("%d",b[i][j]);
7      }
```

二维数组的内存分布

- 物理结构上是线性存放的
- 存放顺序：按行存放，先顺序存放第一行的元素，再存放第二行的元素

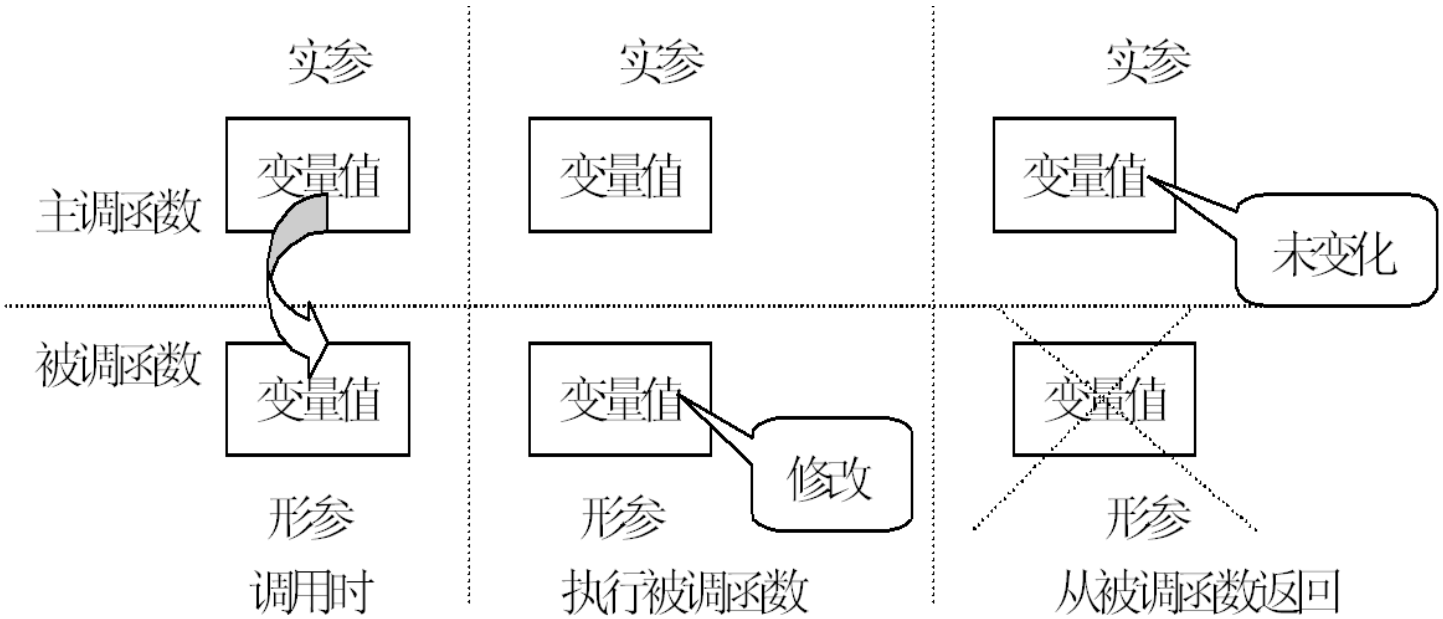
- 如定义数组：int a[M][N];
- 那么元素a[m][n]在数组a中的位置是：
- m * N + n

思考

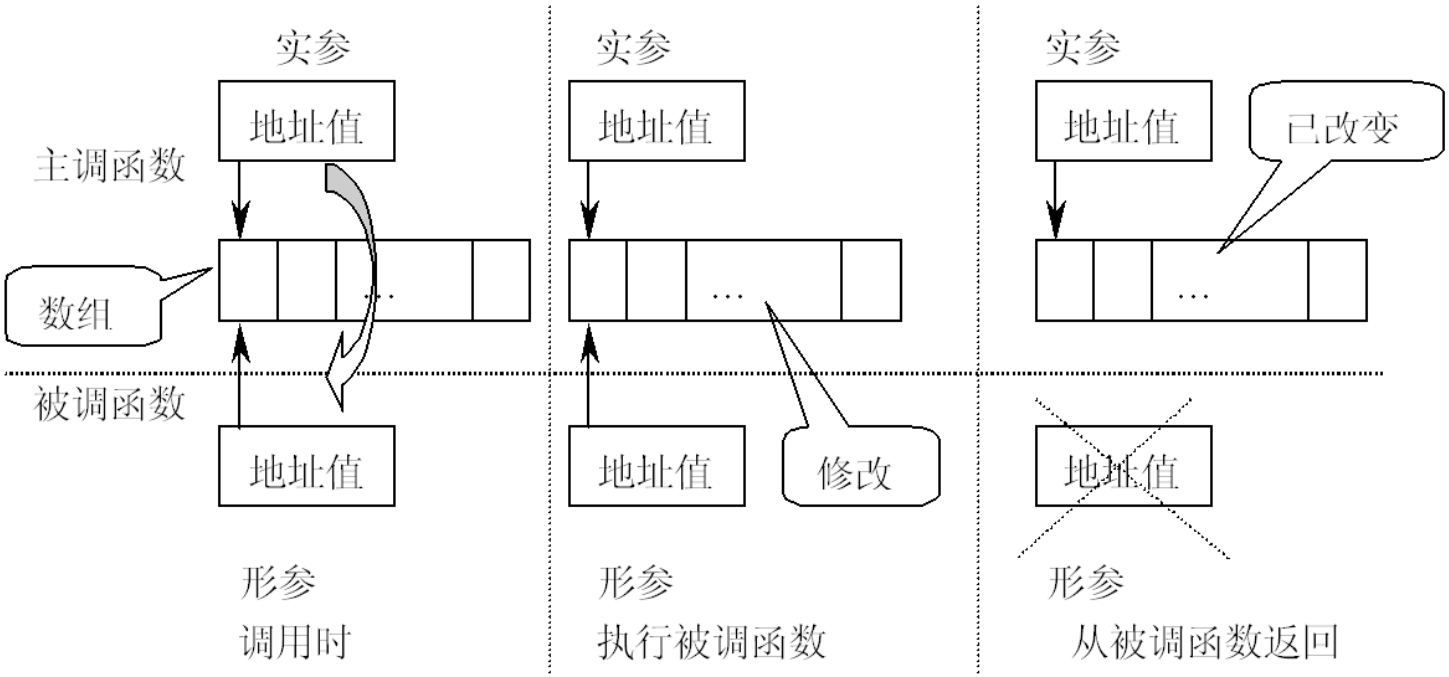
Demo 2: array init

```
1  char ch = (char)0xAA;
2  char a[6][4]={(char)0xFF};
3
4  a[0][0] = 0x00;
5  a[0][4] = 0x04;/*what happen?*/
6  a[1][1] = 0x11;
7  a[5][4] = 0x54;/*what happen then?*/
```

简单变量做函数参数


实参
实参
实参
主调函数
变量值
变量值
变量值
未变化
被调函数
变量值
变量值
变量值
修改
形参
形参
形参
调用时
执行被调函数
从被调函数返回

数组变量做函数参数


实参
地址值
主调函数
数组
被调函数
地址值
形参
调用时
实参
地址值
地址值
修改
形参
执行被调函数
实参
地址值
已改变
地址值
形参
从被调函数返回

用数组名做函数参数

- 用数组名作参数，就是将数组的首地址传递给函数
- 实参数组与形参数组占用同一段内存
- 在函数中可对形参数组元素修改的结果，会影响主调函数中的实参数组结果

问题

为什么要进行地址传递而不进行单向值传递

实例

- 一维数组

```
1 sort(int array[],int n); /*函数定义*/
2
3 int score[5] = {89,67,98,66,56};
4 sort(score, 5); /*函数调用*/
```

- 二维数组

```
1 sort(int array[][3],int m,int n); /*函数定义*/
2
3 int bi[2][3] = {{89,67,55},{98,66,56}};
4 sort(bi,2,3); /*函数调用*/
```

交换法降序

```
1  ExchangeSort(int score[], int n)
2  {
3      for (i=0; i<n-1; i++)
4      {
5          for (j=i+1; j<n; j++)
6          {
7              if (score[j] > score[i])
8                  交换成绩score[j]和score[i];
9          }
10     }
11 }
```

冒泡法降序

```
1  BubbleSort(int score[], int n)
2  {
3      for (i=0; i<n-1; i++)
4      {
5          for (j=n-1; j>i; j--)
6          {
7              if (score[j-1] < score[j])
8                  交换成绩score[j-1]和score[j];
9          }
10     }
11 }
```

选择法排序

```
1  SelectionSort(int score[], int n)
2  {
3      for (i=0; i<n-1; i++){
4          k = i;
5          for (j=i+1; j<n; j++){
6              if (score[j] > score[k]){
7                  记录此轮比较中最高分的元素下标
8                  k = j;
9              }
10             若k中记录的最大数不在位置i,则
11             交换成绩score[j]和score[i],
12     }
13 }
```

排序算法总结

算法名称	稳定性	算法复杂度
交换法排序	no	n*n
冒泡排序	yes	n*n
快速排序	depends	n*log(n)
选择法排序	no	n*n
插入排序	yes	n*n

- 动画演示
  - http://maven.smith.edu/ thiebaut/java/sort/demo.html

查找算法

- 顺序查找
- 二分查找

顺序查找

```
1  int   Search(long a[], int n, long x)
2  {
3      int i;
4      for (i=0; i<n; i++)
5      {
6          if (a[i] == x)
7          {
8              return (i);
9          }
10     }
11     return (-1);
12 }
```

二分查找

```
1  int  BinSearch(long a[], int n, long x){
2  int  low=0, high=n-1, mid;
3  while (low <= high){
4          mid = (high + low) / 2;
5          if (x > a[mid]){
6              low = mid + 1;
7          }
8          else  if (x < a[mid]){
9              high = mid - 1;
10         }
11         else{ return (mid); }
12 }
13 return(-1);
14 }
```

字符串与字符数组

- 字符串
  - 一串以‘\0’结尾的字符序列在C语言中被看作字符串
  - 用双引号括起的一串字符是字符串常量，C语言自动为其添加’\0’终结符
  - C语言并没有为字符串提供任何专门的表示法，完全使用字符数组和字符指针来处理
- 字符数组
  - 每个元素都是字符类型的数组
  - char string[80];

字符数组的初始化

- 用字符型数据对数组进行初始化
  - char str[6] = {'C','h','i','n','a','\0'};
- 用字符串常量直接对数组初始化（更简单的方法）
  - char str[6] = {"China"};
  - char str[6] = "China";
  - char str[5] = "China"; 正确吗？
  - char str[] = "China";
- 二维字符数组
  - char week[][10]={"Saturday","Sunday"};

scanf函数

```
1       char str[10];
2       scanf("%s", str);
```

- scanf不能读入带空格的字符串，gets()可以。
- gets和scanf用法都不安全。
- 当用户的输入多于10个（含10个），str数组将越界
- scanf被公认为最易遭到黑客攻击的函数之一

fgets函数

- 对输入字符串长度有限制的函数调用
- fgets(char* s, int n, stdin);
- 最多读入n-1个字符
- 当读到换行回车符、文件末尾或读满n-1个字符后返回
- 返回后，在字符串的末尾加'\0'标记

字符串的输出

```
1       char str[80] = "Hello World";
2       printf("%s \n", str);
3       puts(str); /* more simple */
```

字符串处理函数

- 在<string.h>中定义了若干专门的字符串处理函数
- 字符串拷贝
  - char *strcpy(char *dest, const char *src);
- 字符串长度
  - size_t strlen(const char *s);
  - 返回字符串的实际长度，不包括'\0'
- 字符串粘结
  - char *strcat(char *dest, const char *src);
- 字符串比较
  - int strcmp(const char *s1, const char *s2);
  - 当出现第一对不相等的字符时，就由这两个字符决定所在字符串的大小

实例

- 从键盘任意输入5个学生的姓名，编程找出并输出按字典顺序排在最前面的学生姓名
- 等价于求最小字符串

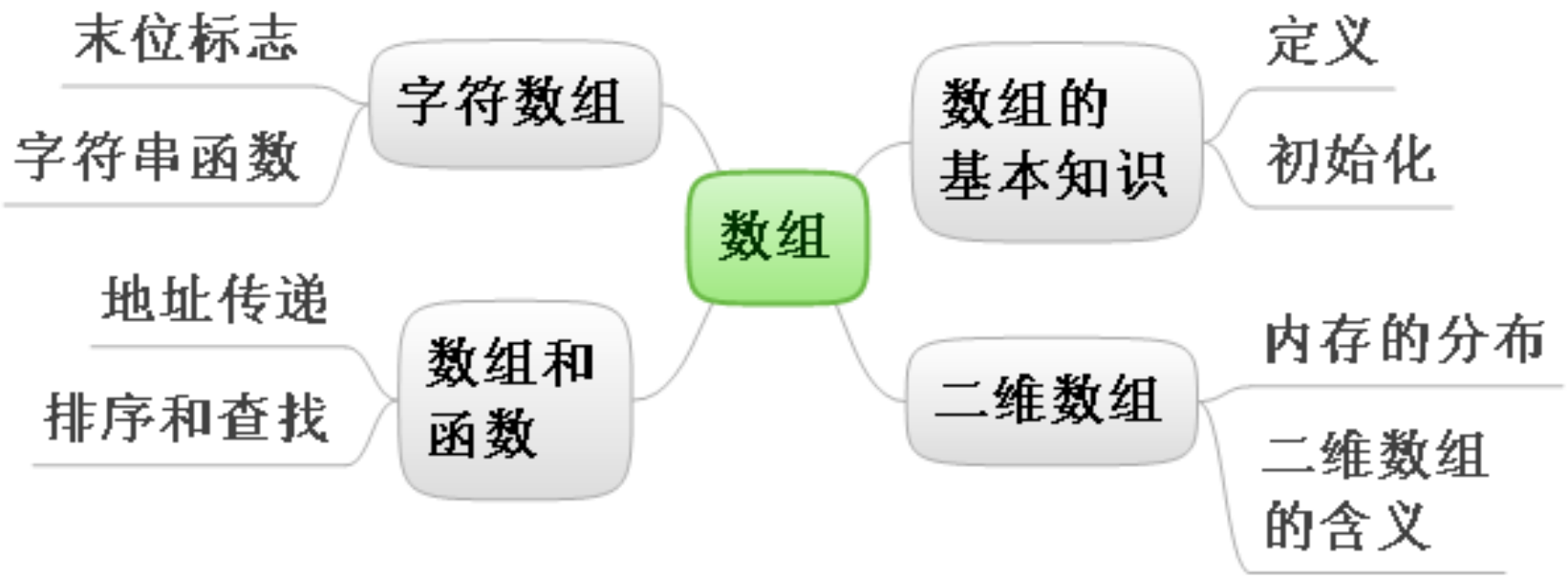
实例源码

```
1  #include  <string.h>
2  #define ARRA_SIZE 80
3       int     n, num;
4       char   str[ARRA_SIZE], min[ARRA_SIZE];
5       printf("Please enter five names:\n");
6       fgets(str,ARRA_SIZE,stdin);
7       strcpy(min, str);
8       for (n=1; n<5; n++){
9            fgets(str,ARRA_SIZE,stdin);
10           if (strcmp(str, min) < 0){
11                strcpy(min, str);
12           }
13      }
14      printf("The min is %s",min);
```

n族字符串处理函数

- 假若交给这些字符串处理函数的字符串没有'\0'会如何？
  - '\0'很关键，如果没有，那么这些处理函数会一直进行处理直到遇到一个'\0'为止。此时可能已经把内存弄得乱七八糟
- ANSI C定义了一些“n族”字符处理函数
- 包括strncpy strncat strncmp等
- 通过增加一个参数来限制处理的最大长度
  - char *strncpy(char *dest, const char *src, unsigned int count);

总结


数组
字符数组
末位标志
字符串函数
数组和函数
地址传递
排序和查找
数组的基本知识
定义
初始化
二维数组
内存的分布
二维数组的含义

谢谢大家，欢迎提问！