[数组的使用](#header-n2)  
 [1 数组的基本概念](#header-n3)  
 [2 数组定义语法](#header-n15)  
 [3 数组的使用](#header-n44)  
 [4 一维数组的初始化](#header-n60)  
 [4.1 方法一：全部初始化](#header-n64)  
 [4.2 部分元素赋初值](#header-n75)  
 [4.3 省略长度赋初值](#header-n84)  
 [5 一维数组的使用示例](#header-n93)  
 [5.1 求平均值、最大值、最小值](#header-n94)  
 [5.2 数组逆置](#header-n96)  
 [5.3 数组排序](#header-n98)  
 [5.4 最大值与数组末尾元素交换](#header-n111)  
 [5.5 模拟双色球](#header-n113)  
 [6 二维数组](#header-n115)  
 [6.1 二维数组的概念](#header-n116)  
 [6.2 二维数组的定义](#header-n118)  
 [6.3 二维数组的初始化](#header-n154)  
 [6.3.1 全部初始化](#header-n155)  
 [6.3.2 部分初始化](#header-n197)  
 [6.3.3 省略长度初始化](#header-n241)  
 [7 二维数组示例](#header-n305)  
 [7.1 二维数组遍历](#header-n306)  
 [7.2 求平均成绩](#header-n308)  
 [7.3 求元素对角线之和](#header-n310)  
 [7.4 求矩阵外圈元素之和](#header-n314)  
 [8 一维字符数组与字符串](#header-n316)  
 [8.1 字符串知识回顾](#header-n317)  
 [8.2 字符数组](#header-n330)  
 [8.3 字符串的保存](#header-n333)  
 [8.4 字符串的读取和打印](#header-n345)  
 [8.5 字符串基本操作的实现](#header-n368)

## 数组的使用

### 1 数组的基本概念

* 数组：类型相同的数据元素的集合，是C语言中的一种构造数据类型
  + 元素的类型可以是基本数据类型，也可以是构造数据类型
    - 构造数据类型：C语言中根据用户实际需求自定的类型，称之为构造数据类型，常见构造数据类型有：数组、结构体、枚举体、联合体。
* 这些元素会顺序的存储在内存的某段区域

### 2 数组定义语法

数据类型 数组名[数组大小];

* 数据类型：决定了数组中存放元素的类型
* 数组名：决定了数组的名称
  + 必须满足C语言变量名（合法标识符）的要求
    1. 由数字、字母、下划线组成
    2. 数字不能打头，不能作为第一个字符
    3. 不能与关键字冲突
    4. 在同一作用域不允许重名
* 数组大小：决定了数组中最大能够存放的元素数量
  + 数组大小不能是变量，只能是字面值或字面值表达式
    - C语言编译器要求在编译期间就需要确定数组的内存大小
  + 数组大小必须是大于0的正整数

### 3 数组的使用

int arr[6];

* 例如上面例子，定义了一个长度为6的整型数组，对于这个数组，可以认为它就是定义了6个变量。
* 这6个变量是什么？

arr[0], arr[1],arr[2], arr[3],arr[4], arr[5]

* 访问数组中的元素，都是通过数组名+下标的形式来完成的。
* 下标从0开始，最大下标为数组长度-1。
* 下标为什么从0开始？需要等到学完数组与指针的关系后再解答。

### 4 一维数组的初始化

* 初始化：定义变量的同时赋初值，称为初始化

#### 4.1 方法一：全部初始化

int arr[5] = {1, 2, 3, 4, 5};  
// arr[0]:1  
// arr[1]:2  
// arr[2]:3  
// arr[3]:4  
// arr[4]:5

* 初始化列表：用大括号括起来的，用于对数组进行初始化的一个值的列表，每个值之间通过逗号隔开
* 会根据初始化列表中元素顺序，依次对数组每一个元素初始化。
* 初始化列表只能在定义数组的时候使用，数组定义完成之后不能再使用初始化列表！
* 全部初始化要求初始化列表中元素个数必须与数组大小相同

#### 4.2 部分元素赋初值

int arr[5] = {1, 2, 3};  
// arr[0]:1  
// arr[1]:2  
// arr[2]:3  
// arr[3]:0  
// arr[4]:0

* 首先会依次将初始化列表中的每一个元素值按照顺序对数组每一个元素赋初值
* 如果发现数组中仍有元素没有被初始化，则默认初始化为0
* 实际开发中，通常采用部分元素赋初值的方法对数组元素进行初始化，例如：int arr[100] = {0};

#### 4.3 省略长度赋初值

int a[] = {1, 2, 3, 4, 5};  
// 等同于 int a[5] = {1, 2, 3, 4, 5};

* 在定义数组时，如果后面跟有初始化列表，并且初始化列表元素值的个数就是预期的数组大小，那么可以省率中括号中的数组大小
* 定义数组时，如果后面没有进行初始化的初始化列表，那么数组长度不能省略
* 定义数组时，时刻铭记要通过某一个途径，准确的告诉编译器数组的大小才可以

### 5 一维数组的使用示例

#### 5.1 求平均值、最大值、最小值

#include <stdio.h>  
  
int main(void)   
{  
 int arr[10];  
 int i, sum, max, min;  
  
 // 命令行读取10个整数  
 for (i=0; i<10; i++)   
 {   
 scanf("%d", &arr[i]);   
 }  
  
 // 按顺序打印  
 for (i=0; i<10; i++)   
 {   
 printf("%d ", arr[i]);   
 }  
 printf("\n");  
  
 sum = 0;  
 max = arr[0];  
 min = arr[0];  
 // 求10个数之和、最大值  
 for (i=0; i<10; i++)  
 {  
 sum += arr[i];  
 if (max < arr[i])   
 { max = arr[i]; }   
 if (min > arr[i])  
 { min = arr[i]; }  
 }  
 printf("和为：%.2f\n", sum / 10.0);  
 printf("最大值为：%d\n", max);  
 printf("最小值为：%d\n", min);  
  
 // 倒序输出  
 for (i=9; i>=0; i--)   
 {  
 printf("%d ", arr[i]);  
 }  
 printf("\n");  
  
 return 0;  
}

#### 5.2 数组逆置

#include <stdio.h>  
  
int main(void)   
{  
 int arr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
 int i, temp;  
  
 for (i=0; i<10/2; i++)   
 {  
 temp = arr[i];  
 arr[i] = arr[10-1-i];  
 arr[10-1-i] = temp;  
 }  
  
 for (i=0; i<10; i++)   
 {   
 printf("%d ", arr[i]);   
 }  
 printf("\n");  
  
 return 0;  
}

#### 5.3 数组排序

* 冒泡排序

#include <stdio.h>  
  
int main(void)   
{  
 int arr[10] = {5, 7, 1, 4, 9, 2, 10, 3, 8, 6};  
 int i, j, temp;  
 int n = 10;  
  
 // 控制冒泡次数  
 for (i=0; i<n-1; i++)  
 {  
 // 控制两两比较次数  
 for (j=0; j<9; j++)  
 {  
 // 预期存放顺序从小到大，实际存储与预期不相符时交换  
 if (arr[j] > arr[j+1])   
 {  
 temp = arr[j];  
 arr[j] = arr[j+1];  
 arr[j+1] = temp;  
 }  
 }  
 }  
  
 for (i=0; i<10; i++)   
 {   
 printf("%d ", arr[i]);   
 }  
 printf("\n");  
  
 return 0;  
}

* 选择排序

#include <stdio.h>  
  
int main(void)   
{  
 int arr[10] = {5, 7, 1, 4, 9, 2, 10, 3, 8, 6};  
 int i, j, temp, min;  
 int n = 10;  
  
 for (i=0; i<n-1; i++)  
 {  
 min = i;  
 // 求当前无序序列中最小值  
 for (j=i+1; j<n; j++)  
 {  
 if (arr[min] > arr[j])  
 {  
 min = j;  
 }  
 }  
 temp = arr[i];  
 arr[i] = arr[min];  
 arr[min] = temp;  
 }  
  
 for (i=0; i<10; i++)   
 {   
 printf("%d ", arr[i]);   
 }  
 printf("\n");  
  
 return 0;  
}

* 直接插入排序

#include <stdio.h>  
  
int main(void)   
{  
 int arr[10] = {5, 7, 1, 4, 9, 2, 10, 3, 8, 6};  
 int i, j, temp;  
 int n = 10;  
  
 // i代表着无序序列中第一个元素  
 for (i=1; i<n; i++)  
 {  
 // 通过temp保存无序序列第一个元素  
 temp = arr[i];  
 // 寻找要插入的位置  
 for (j=i-1; j>=0 && arr[j] > temp; --j)  
 {  
 arr[j+1] = arr[j];  
 }  
 arr[j+1] = temp;  
 }  
  
 for (i=0; i<10; i++)   
 {   
 printf("%d ", arr[i]);   
 }  
 printf("\n");  
  
 return 0;  
}

#### 5.4 最大值与数组末尾元素交换

#include <stdio.h>  
  
int main(void)   
{  
 /\*\*  
 \* 有个长度为10的数组，数组中的元素是无序排列的，  
 \* 要求将数组中最大元素与数组末尾元素进行交换。  
 \*/  
 int arr[10] = {7, 9, 1, 6, 2, 5, 3, 4, 8, 10};  
 int i, temp, max;   
 // max表示最大元素的下标, arr[max] 表示最大元素  
 max = 0;  
 for (i=1; i<10; i++) {  
 if (arr[i] > arr[max]) {  
 max = i;  
 }  
 }  
 temp = arr[max];  
 arr[max] = arr[9];  
 arr[9] = temp;  
 for (i=0; i<10; i++) {  
 printf("%d ", arr[i]);  
 }  
 printf("\n");  
  
 return 0;  
}

#### 5.5 模拟双色球

#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
  
/\*\*  
 \* 写一个模拟双色球摇号的程序，红色 1~33 选6，蓝色 1~16 选1  
 \*/  
int main(void)   
{  
 int redArr[6], blue, i, j;  
 srand(time(0));  
  
 for (i=0; i<6; i++) {  
 int flag = 0;  
 int red = rand() % 33 + 1;  
 for (j=0; j<i; j++) {  
 if (redArr[j] == red) {  
 flag = 1;  
 break;  
 }  
 }  
 if (flag == 1) {  
 i--;  
 }  
 else {  
 redArr[i] = red;  
 }  
 }  
 blue = rand()%16 + 1;  
   
 printf("红球为：");  
 for (i=0; i<6; i++) {  
 printf("%d ", redArr[i]);  
 }  
 printf("\n蓝球为：%d\n", blue);  
  
 return 0;  
}

### 6 二维数组

#### 6.1 二维数组的概念

二维数组就是存放一维数组的一维数组。本质上可以理解为二维数组就是一个一维数组，只不过这个一维数组里面的每一个元素都是一个一维数组。

#### 6.2 二维数组的定义

格式：

类型名 数组名[行数][列数];

例如：

int arr[3][4];

* 相当于定义了一个3行4列的二维数组；
* 就是一个长度为3的一维数组，这个一维数组里面每一个元素是：长度为4的整型一维数组。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| int arr[3][4] | 第0列 | 第1列 | 第2列 | 第3列 |
| 第0行 | 1 | 2 | 3 | 4 |
| 第1行 | 5 | 6 | 7 | 8 |
| 第2行 | 9 | 10 | 11 | 12 |

#### 6.3 二维数组的初始化

##### 6.3.1 全部初始化

* 按行全部赋初值

int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};

|  |  |  |  |
| --- | --- | --- | --- |
| int arr[2][3] | 第0列 | 第1列 | 第2列 |
| 第0行 | 1 | 2 | 3 |
| 第1行 | 4 | 5 | 6 |

* 顺序全部赋初值

int arr[2][3] = {1, 2, 3, 4, 5, 6};

|  |  |  |  |
| --- | --- | --- | --- |
| int arr[2][3] | 第0列 | 第1列 | 第2列 |
| 第0行 | 1 | 2 | 3 |
| 第1行 | 4 | 5 | 6 |

##### 6.3.2 部分初始化

* 按行部分赋值

int arr[2][3] = {{1, 2}, {4}};

|  |  |  |  |
| --- | --- | --- | --- |
| int arr[2][3] | 第0列 | 第1列 | 第2列 |
| 第0行 | 1 | 2 | 0 |
| 第1行 | 4 | 0 | 0 |

* 顺序部分赋值

int arr[2][3] = {1, 2, 3, 4};

|  |  |  |  |
| --- | --- | --- | --- |
| int arr[2][3] | 第0列 | 第1列 | 第2列 |
| 第0行 | 1 | 2 | 3 |
| 第1行 | 4 | 0 | 0 |

注意：按行和顺序不能同时使用。例如 int arr[2][3] = {{1,2}, 3, 4};

##### 6.3.3 省略长度初始化

* 可以省率行数，不能省率列数
* 按行

int arr[][3] = {{1,2,3}, {4,5,6}, {7,8,9}}; // int arr[3][3]  
int brr[][3] = {{1,2}, {3, 4}}; // int brr[2][3]

|  |  |  |  |
| --- | --- | --- | --- |
| int arr[][3] | 第0列 | 第1列 | 第2列 |
| 第0行 | 1 | 2 | 3 |
| 第1行 | 4 | 5 | 6 |
| 第2行 | 7 | 8 | 9 |
| int brr[][3] | 第0列 | 第1列 | 第2列 |
| 第0行 | 1 | 2 | 0 |
| 第1行 | 3 | 4 | 0 |

* 按顺序

int arr[][3] = {1, 2, 3, 4, 5}; // int arr[2][3]

|  |  |  |  |
| --- | --- | --- | --- |
| int arr[][3] | 第0列 | 第1列 | 第2列 |
| 第0行 | 1 | 2 | 3 |
| 第1行 | 4 | 5 | 0 |

### 7 二维数组示例

#### 7.1 二维数组遍历

#include <stdio.h>  
  
int main(void)   
{  
 /\*  
 1 2 3 4  
 5 6 7 8  
 9 10 11 12  
 \*/  
 int arr[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};  
 int i, j;  
  
 for (i=0; i<3; i++) {  
 for (j=0; j<4; j++) {  
 printf("%3d ", arr[i][j]);   
 }  
 printf("\n");  
 }  
  
 return 0;  
}

#### 7.2 求平均成绩

#include <stdio.h>  
  
int main(void)   
{  
 /\*  
 张三 李四 王五 赵六  
 高数 64 55 12 78  
 英语 75 90 14 68  
 C语言 89 85 99 84  
  
 求每个人的平均成绩和每门课的平均成绩  
 \*/  
   
 int arr[3][4] = {0};  
 double ave1[3], ave2[4];   
 // ave1 每一科的平均成绩 ave2 每个人的平均成绩  
 int i, j;  
 for (i=0; i<3; i++) {  
 printf("请输入");  
 switch (i) {  
 case 0: printf("高数"); break;  
 case 1: printf("英语"); break;  
 case 2: printf("C语言"); break;  
 }  
 printf("四个人的成绩，用空格隔开：");  
 for (j=0; j<4; j++) {  
 scanf("%d", &arr[i][j]);  
 }  
 }  
 // 求每一科的平均成绩  
 for (i=0; i<3; i++) {  
 int sum = 0;  
 for (j=0; j<4; j++) {  
 sum += arr[i][j];  
 }  
 ave1[i] = sum / 4.0;  
 }  
 // 求每个人的平均成绩  
 for (i=0; i<4; i++) {  
 int sum = 0;  
 for (j=0; j<3; j++) {  
 sum += arr[j][i];  
 }  
 ave2[i] = sum / 3.0;  
 }  
  
 for (i=0; i<3; i++) {  
 switch (i) {  
 case 0: printf("高数"); break;  
 case 1: printf("英语"); break;  
 case 2: printf("C语言"); break;  
 }  
 printf("的平均成绩为：%.2f\n", ave1[i]);  
 }  
 printf("------------------------\n");  
 for (i=0; i<4; i++) {  
 switch (i) {  
 case 0: printf("张三"); break;  
 case 1: printf("李四"); break;  
 case 2: printf("王五"); break;  
 case 3: printf("赵六"); break;  
 }  
 printf("的平均成绩为：%.2f\n", ave2[i]);  
 }  
  
 return 0;  
}

#### 7.3 求元素对角线之和

#include <stdio.h>  
  
int main(void)   
{  
 /\*  
 用户输入一个5\*5的整数矩阵，求矩阵的对角线元素之和  
 \*/  
   
 int arr[5][5] = {0};  
 int i, j, sum = 0;  
 for (i=0; i<5; i++) {  
 for (j=0; j<5; j++) {  
 scanf("%d", &arr[i][j]);  
 }  
 }  
  
 for (i=0; i<5; i++) {  
 for (j=0; j<5; j++) {  
 if (i==j || i+j==4) {  
 sum += arr[i][j];  
 }  
 }  
 }  
 printf("%d\n", sum);  
  
 return 0;  
}

简化方法

#include <stdio.h>  
  
int main(void)   
{  
 int i, j, num, sum = 0;  
 for (i=0; i<5; i++) {  
 for (j=0; j<5; j++) {  
 scanf("%d", &num);  
 if (i==j || i+j==4) {  
 sum += num;  
 }  
 }  
 }  
 printf("%d\n", sum);  
  
 return 0;  
}

#### 7.4 求矩阵外圈元素之和

#include <stdio.h>  
  
int main(void)   
{  
 /\*  
 用户输入一个5\*5的整数矩阵，求矩阵最外圈元素之和  
 \*/  
   
 int i, j, num, sum = 0;  
 for (i=0; i<5; i++) {  
 for (j=0; j<5; j++) {  
 scanf("%d", &num);  
 if (i==0 || j==0 || i==4 || j==4)  
 sum += num;  
 }  
 }  
 printf("%d\n", sum);  
  
 return 0;  
}

### 8 一维字符数组与字符串

#### 8.1 字符串知识回顾

* 字符串长度：第一个'\0'之前字符的个数
* 字符串字符个数：所有字符的数量
* "abcde"：字符串长度：5 字符个数：6
* "ab\\\012\0cd"：字符串长度：4 字符个数：8
  + 'a' 'b' '\\' '\012' '\0' 'c' 'd' '\0'

#### 8.2 字符数组

保存字符的数组，数组中每个元素都是一个字符。

char ch[5] = {'a', 'b', 'c', 'd', 'e'};

#### 8.3 字符串的保存

**C语言中没有专门存放字符串的类型，如果期望对字符串进行保存，只能通过字符数组来保存。**

char s1[6] = {'a', 'b', 'c', 'd', 'e', '\0'};  
char s2[6] = "abcde";  
// s1 与 s2 初始化的效果是相同的

* C语言中允许通过字符串字面值初始化字符数组。
* **字符串可以理解为是一种特殊的字符数组**，如果字符数组中存在'\0'，那么它就可以当成是一个字符串了，如果没有'\0'，那就是一个普通的字符数组。
* '\0' 的 ASCII码值是 0，0表示假，表示停止循环 输入0相当于输入’\0’

注意：在通过字符串字面值对字符数组进行初始化的时候，一定要保证字符数组的长度足够保存下字符串字面值最后的'\0'。否则视为数组访问越界，字符数组也无法完成字符串的保存。

#### 8.4 字符串的读取和打印

char str[100];  
scanf("%s", str);   
printf("%s\n", str);

* 注意：scanf读取字符串使用%s，另外**数组名前面不需要加 &** 。
* 字符串在打印的时候，实际上是**打印的第一个'\0'之前所有的字符,直到看见‘、0’**。

**gets**

char str[100];  
//输入 ni hao a  
scanf("%s", str);   
//打印 ni  
printf("%s\n", str);

* scanf在读取字符串时，**遇到空格会任务字符串输入截止**，不会读取空格及空格之后的内容。
* scanf不会读取空格。
* **如果想要完整的读取一行内容（包含空格），可以使用gets()**

**puts**

char str[100];  
gets(str);   
puts(str);

* puts()在完成字符串打印之后，**自动打印一个换行**。
* 如果只是原原本本的输出某个字符串，使用puts()
* 区分getchar putchar这两个只能输入输出一个字符

#### 8.5 字符串基本操作的实现

#include <stdio.h>  
  
int main(void)  
{  
 // 求字符串长度  
 char str[] = "lizhemeipingshichengji";  
 int i;  
 for (i=0; str[i]; i++);  
 printf("%d\n", i);  
  
 // 字符串拷贝  
 char str1[] = "yangshize";  
 char str2[100];  
 int i;  
 for (i=0; str1[i]; i++)   
 str2[i] = str1[i];  
 str2[i] = 0;  
 puts(str2);  
  
 // 字符串拼接  
 char str1[100] = "chen";  
 char str2[] = "ziyue";  
 int i, j;  
 for (i=0; str1[i]; i++);  
 for (j=0; str2[j]; j++)  
 str1[i+j] = str2[j];  
 puts(str1);  
  
 // 字符串比较大小 相等打印0， 大于打印1， 小于打印-1  
 // "abcd" 相等 "abcd"   
 // "aaaa" 小于 "abcd"  
 // "cd" 大于 "abcd"  
 // "abcd" 大于 "abc"  
 char str1[] = "abcd";  
 char str2[] = "abcd";  
 int i, rtn;  
 for (i=0; str1[i]==str2[i] && str1[i]!='\0'; i++)  
 ;  
 rtn = str1[i] - str2[i];  
 if (rtn > 0)  
 rtn = 1;  
 else if(rtn < 0)  
 rtn = -1;  
 printf("%d\n", rtn);  
  
 return 0;  
}

8.6 字符串操作函数

必须引入头文件 **<string.h>**

* strlen() ：求字符串长度
* strcpy() ：字符串拷贝
* strcat() ： 字符串拼接
* strcmp() ： 字符串比较

#include <stdio.h>  
#include <string.h> // C语言关于字符串操作的库  
  
int main(void)  
{  
 // 求字符串长度 strlen()  
 char str1[] = "zhangzhuofan";  
 printf("%d\n", strlen(str1));  
  
 // 字符出拷贝 strcpy()  
 char str1[] = "xujieru";  
 char str2[100];  
 strcpy(str2, str1); // 将str1的内容拷贝到str2中  
 puts(str2);  
 strcpy(str2, "hechaokang");  
 puts(str2);  
  
 // 字符串拼接 strcat()  
 char str1[100] = "wang";  
 char str2[] = "miao";  
 strcat(str1, str2); // 把str2拼接到str1后面  
 puts(str1);  
  
 // 字符串比较 strcmp();  
 // 相等返回0 参数1大于参数2返回1 参数1小于参数2返回-1  
 char str1[] = "lizhe";  
 char str2[] = "wangyuheng";  
 printf("%d\n", strcmp(str1, str2));  
  
 return 0;  
}