



知识点总结



- 在C语言中，**数组**用于将**相同类型**的元素按一定**顺序**排列在一起。
- 一维数组**：用来表示一组同类型的有顺序关系的数据。
- 定义一个一维数组的语法格式：**数据类型 数组名[数组长度]**;
例如：`int arr[5]`；定义了一个名叫arr，包含5个整数的一维数组。
- 定义并初始化一维数组的方式有多种：
 - 1) **给全部元素赋值**。
例：`int arr[5] = {1, 2, 3, 4, 5};`



2) 只给部分元素赋值，未赋初始值的元素会被自动附为数组类型对应的零值。
比如，在整数数组里，它们会被赋为整数0。
在浮点数数组里，它们会被赋为浮点数0.0。
在字符数组里，它们会被赋为空字符'\0'，空字符是一个ASCII值为0的字符。

例：

```
int arr_1[5] = {1, 2};    实际上是{1, 2, 0, 0, 0}
```

```
float arr_2[5] = {1.0, 2.0};  实际上是{1.0, 2.0, 0.0, 0.0, 0.0}
```

```
char arr_3[5] = {'a', 'b'};   实际上是{'a', 'b', '\0', '\0', '\0'}
```



3) 不指定数组长度，程序会根据花括号里元素的个数，来确定数组的大小

例：`int arr[] = {6, 7, 8, 9, 10};` 该数组大小为5

·可以通过索引（即下标）来访问和操作一维数组元素，索引从0开始。

语法格式：数组名[索引值]

例：

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

// 把数组arr里索引为2的元素（即数组arr里第3个元素）的值修改为10

```
arr[2] = 10;
```

// 打印出数组arr里索引为2的元素值

```
printf("%d", arr[2]);
```



- 可以用一个for循环遍历一维数组：

例：用for循环遍历数组numbers，打印出所有元素值。

```
#include <stdio.h>
```

```
int main() {  
    int numbers[5] = {1, 2, 3, 4, 5};  
  
    for (int i = 0; i < 5; i++) {  
        printf("%d ", numbers[i]);  
    }  
    printf("\n");  
}
```



· **二维数组**：可以看作是由多个一维数组组成的数组，像一张有行有列的表格。

定义一个二维数组的语法格式：**数据类型 数组名[行数][列数]**；

例：`int arr[3][4]`； 定义了一个名叫arr，包含3行4列的二维整数数组。

· 定义并初始化二维数组的方式有多种：

1) **给全部元素赋值**。

例：`int arr[2][3] = {{1, 2, 3}, {4, 5, 6}}`； // 一个内层花括号表示一行
等价于`int arr[2][3] = {1, 2, 3, 4, 5, 6}`； // 程序会按行自动存储为{{1, 2, 3}, {4, 5, 6}}

2) **给部分元素赋值**，未赋初始值的元素会被自动附为数组类型对应的零值。

例：`int arr[2][3] = {{1, 2}, {4}}`； 实际上是{{1, 2, 0}, {4, 0, 0}}

3) **不指定**二维数组的行数，程序会根据花括号里的元素个数和二维数组的列数来确定行数。

例：`int arr[][3] = {1, 2, 3, 4, 5, 6, 7, 8, 9}`；

这个花括号里有9个元素，列数是3，所以这个二维数组的行数是 $9 \div 3 = 3$ 。

注意：**可以不指定行数**，但**一定不能不指定列数**。



- 可以通过两个索引来获取或修改二维数组元素值，即行索引和列索引，都是从0开始。

语法格式：数组名[行索引][列索引]

例：arr[1][2]表示第2行第3列的元素。

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

// 把数组arr里行索引为1，列索引为2的元素（即数组arr里第2行第3列的元素）的值修改为10

```
arr[1][2] = 10;
```

// 打印出数组arr里行索引为1，列索引为2的元素值

```
printf("%d", arr[2]);
```



- 可以用一个嵌套的for循环遍历二维数组：

例：用嵌套的for循环遍历数组numbers，并打印出所有元素值。

```
#include <stdio.h>
```

```
int main() {  
    int numbers[2][3] = {{1, 2, 3}, {4, 5, 6}};  
    // 外层for循环控制行  
    for (int i = 0; i < 2; i++) {  
        // 内层for循环控制列  
        for (int j = 0; j < 3; j++) {  
            printf("%d ", numbers[i][j]);  
        }  
        printf("\n");  
    }  
}
```



·注意：

- 1、定义数组时，数组长度或行数、列数必须是**常量表达式**，不能是变量。
- 2、初始化数组时，提供的初始值数量不能超过数组的大小。
- 3、数组**在内存中是连续存储的**。另外，**二维数组在内存中是按行存储的**，也就是说先存储第一行的元素，然后再存储第二行，以此类推。
- 4、数组名本身代表数组的**首地址**（即第一个元素的地址）。

例：`int arr[2] = {0, 1};` 数组名arr代表的是数组里的**第一个元素的地址&arr[0]**。



5、在C语言中，**不能直接将一个数组名赋给另一个数组**。如果要将一个数组的内容复制到另一个数组，**必须逐个元素地进行赋值操作**。

例：将数组arr_1里的元素复制到数组arr_2里。

```
#include <stdio.h>
```

```
int main() {  
    int arr_1[] = {1, 2, 3, 4, 5};  
    int arr_2[5];  
    // 逐个元素复制  
    for (int i = 0; i < 5; i++) {  
        arr_2[i] = arr_1[i];  
    }  
}
```



! 定义时没指定一维数组的大小，程序会根据初始化的元素个数来确定大小

题目✓1、以下定义数组的语句中错误的是（ C ）

A. `int num[] = {1,2,3,4,5,6};`

B. `int num[][3] = {{1,2},3,4,5,6};`

C. `int num[2][4] = {{1,2},{3,4},{5,6}};`

D. `int num[][4] = {1,2,3,4,5,6};`



✓ 题目2、若有定义语句：`int m[] = {5, 4, 3, 2, 1}, i = 4;`，则下面对m数组元素的引用中错误的是 (C)

A. `m[--i]`

B. `m[2 * 2]`

C. `m[m[0]]`

D. `m[m[i]]`



题目3、有以下程序：

```
#include <stdio.h>
```

```
main()
```

```
{
```

```
    int a[]={2, 3, 5, 4}, i;
```

```
    for(i = 0; i < 4; i++)
```

```
        switch(i % 2)
```

```
        {
```

```
            case 0:
```

```
                switch(a[i] % 2)
```

```
                {
```

```
                    case 0: a[i]++; break;
```

```
                    case 1: a[i]--;
```

```
                }
```

```
                break;
```

```
            case 1: a[i]=0;
```

```
        }
```

```
    for(i = 0; i < 4; i++) printf("%d", a[i]);
```

```
    printf("\n");
```

```
}
```

程序运行后的输出结果是 (C)

A. 3344

B. 2050

C. 3040

D. 0304

索引是偶数：进入判断元素

元素是偶数 -> 元素自增1

元素是奇数 -> 元素自减1

索引是奇数：索引位置的值修改为0



题目4、有如下程序：

```
#include <stdio.h>
```

```
main()
```

```
{
```

```
    int i, k;
```

```
    int array[4][2] = {{1, 2}, {4, 9}, {6}};
```

```
    for(i = 0; i < 2; i++)
```

```
        for(k = 0; k < 4; k++)
```

```
        {
```

```
            printf("%d,", array[k][i]);
```

```
        }
```

```
    printf("\n");
```

```
}
```

- 外层for循环控制的是列,
- 内层for循环控制的是行

程序运行后的输出结果是 (B)

A. 1,2,4,9,6,

B. 1,4,6,0,2,9,0,0,

C. 2,9,0,0,1,4,6,0,

D. 2,9,6,1,4,



题目5、有以下程序

```
#include <stdio.h>
```

```
main()
```

```
{
```

```
    int b[3][3] = {0, 1, 2, 0, 1, 2, 0, 1, 2}, i, j, t = 1;
```

```
    for (i = 0; i < 3; i++)
```

```
        for (j = i; j <= i; j++) t += b[i][b[j][i]];
```

```
    printf("%d\n", t);
```

```
}
```

$$\begin{aligned} t &= t + b[0][b[0][0]] + b[1][b[1][1]] + b[2][b[2][2]] \\ &= t + b[0][0] + b[1][1] + b[2][2] \\ &= 1 + 0 + 1 + 2 = 4 \end{aligned}$$

程序运行后的输出结果是 (A)

A. 4

B. 3

C. 1

D. 9

· 对角线



题目6、下面程序的划线处有语法或逻辑错误，请找出并改正，使其得到符合题意的执行结果。

求用户输入一个数组，求该数组中最大值及其下标。

```
int main( ) {  
    int max, j, m;  
    int a[5]; j < 5 或 j <= 4  
    for(j = 1; j <= 5; j++) {  
        scanf("%d", a); &a[j]  
    }  
    max = a[0]; j < 5 或 j <= 4  
    for(j = 1; j <= 5; j++) {  
        if(max > a[j]) { max < a[j]  
            max = a[j];  
            m = j;  
        }  
    }  
    printf("下标: %d\n 最大值:%d", j, max);  
    return 0; m  
}
```