第二章 数组

1.1 数组的概念

数组是若干个相同类型的变量在内存中有序存储的集合。
int a[10];//定义了一个整型的数组 a，a 是数组的名字，数组中有 10 个元素，每个元素的类型都是 int 类型，而且在内存中连续存储。
这十个元素分别是 a[0] a[1] …. a[9]
a[0]~a[9]在内存中连续的顺序存储

1.2 数组的分类

1.2.1 按元素的类型分类

1）字符数组
即若干个字符变量的集合，数组中的每个元素都是字符型的变量
char s[10]; s[0],s[1]....s[9];
2）短整型的数组
short int a[10]; a[0] ,a[9]; a[0]=4;a[9]=8;
3）整型的数组
int a[10]; a[0] a[9]; a[0]=3;a[0]=6;
4) 长整型的数组
lont int a[5];
5）浮点型的数组（单、双）
float a[6]; a[4]=3.14f;
double a[8]; a[7]=3.115926;
6）指针数组
char *a[10]
int *a[10];
7）结构体数组
struct stu boy[10];

1.2.2 按维数分类

一维数组
int a[30];
类似于一排平房
二维数组
int a[2][30];
可以看成一栋楼房 有多层，每层有多个房间，也类似于数学中的矩阵
二维数组可以看成由多个一维数组构成的。
有行，有列，

多维数组

int a[4][2][10];

三维数组是由多个相同的二维数组构成的

int a[5][4][2][10];

1.3 数组的定义

定义一个数组，在内存里分配空间

1.3.1 一维数组的定义

格式：

数据类型　数组名　[数组元素个数];

int　　a　　[10];

char　　b　　[5];定义了 5 个 char 类型变量的数组 b

5 个变量分别为　b[0] ,b[1],b[2]，b[3],b[4];

在数组定义的时候可以不给出数组元素的个数，根据初始化的个数来定数组的大小

例 1 :

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    int a[]={1,2,3,4,5};
    printf("%d\n",sizeof(a));
    return 0;
}
```

1.3.2 二维数组的定义

格式:

数据类型　　数组名　[行的个数][列的个数];

int　　a　　[4][5];

定义了 20 个 int 类型的变量 分别是

a[0][0] ,a[0][1],a[0][2] ,a[0][3] ,a[0][4];

a[1][0] ,a[1][1],a[1][2] ,a[1][3] ,a[1][4];

a[2][0] ,a[2][1],a[2][2] ,a[2][3] ,a[2][4];

a[3][0] ,a[3][1],a[3][2] ,a[3][3] ,a[3][4];

多维数组定义：

int　a[3][4][5]

int a[8][3][4][5];

扩展：

二维数组在定义的时候，可以不给出行数，但必须给出列数，二维数组的大小根据初始化的行数来定

例 2：

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    int a[][3]={
        {1,2,3},
        {4,5,6},
        {7,8,9},
        {10,11,12}
    };
    printf("%d\n",sizeof(a));
    return 0;
}
```

1.4 定义并初始化

开辟空间的同时并且给变量赋值

1.4.1 一维数组的初始化

a、全部初始化

int a[5]={2,4,7,8,5};

代表的意思： a[0]=2; a[1]=4;a[2]=7;a[3] = 8;a[4]=5;

b、部分初始化

int a[5]={2,4,3};初始化赋值不够后面补 0

a[0] = 2; a[1]= 4;a[2]=3;a[3]=0;a[4]=0;

注意：只能省略后面元素，可以不初始化，不能中间的不初始化

例 3：

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    int a[5]={2,3,5};
    int i;
    for(i=0;i<5;i++)
    {
        printf("a[%d]=%d\n",i,a[i]);
    }
    return 0;
}
```

1.4.2 二维数组的初始化

按行初始化：

a、全部初始化

```
int a[2][2]={{1,2},{4,5}};
a[0][0] =1; a[0][1] = 2; a[1][0] = 4,a[1][1]=5;
```

b、部分初始化

```
int a[3][3]={{1,2},{1}};
a[0][0] = 1;a[0][2] =0;
```

逐个初始化：

全部初始化：

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

部分初始化：

```
int a[2][3]={3,5,6,8};
```

1.5 数组元素的引用方法

一维数组元素的引用方法

数组名[下标]；//下标代表数组元素在数组中的位置，注意从 0 开始

```
int a[10];
a[2];
```

二维数组元素的引用方法

数组名[行下标][列下标];

```
int a[3][4];
a[1][2]
```

例 4：

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    int a[3][4]={{1,2,3,4},{5,6},{5}};
    int b[3][4]={11,12,13,14,15,16,17,18,19};
    int i,j;
    for(i=0;i<3;i++)//遍历所有行
    {
        for(j=0;j<4;j++)//遍历一行的所有列
        {
            printf("a[%d][%d]=%d     ",i,j,a[i][j]);
        }
        printf("\n");
```

```
        }

        for(i=0;i<3;i++)//遍历所有行
        {
            for(j=0;j<4;j++)//遍历一行的所有列
            {
                printf("b[%d][%d]=%d     ",i,j,b[i][j]);
            }
            printf("\n");
        }
        return 0;
    }
```

1.5.1 字符数组的定义

```
char   c1[] ={ 'c' ,'  ' ,' p' ,' r' ,' o' ,' g' };
char c2[] =  "c prog" ;
char a[][5] = {
                { 'B' ,' A' ,' S' ,' I' ,' C' },
                { 'd' ,' B' ,' A' ,' S' ,' E' }
              };
 char a[][6] = {"hello" ,"world" };
```

- **字符数组的引用**

1.用字符串方式赋值比用字符逐个赋值要多占 1 个字节,用于存放字符串结束标志'\0';

2.上面的数组 c2 在内存中的实际存放情况为:

'c'	' '	'p'	'r'	'o'	'g'	'\0'

注:'\0'是由 C 编译系统自动加上的

3.由于采用了'\0'标志,字符数组的输入输出将变得简单方便.

例 5:

```
int main( )
{
    char str[15];
    printf("input string:\n");
    scanf("%s",str);//hello
    printf("output:%s\n",str);
    return 0;
```

```
}
```