第三章 数组与指针

数组

一维数组

```
数据类型  数组名[ 数组长度 ];
数据类型  数组名[ 数组长度 ] = { 值1，值2 ...};
数据类型  数组名[ ] = { 值1，值2 ...};
```

说明：

- **定义数组时[]里必须是常量表达式，不能是变量**
- 数组名的命名规范与变量名命名规范一致，不要和变量重名
- 数组中下标是从0开始索引

一维数组数组名

1.用途：

- 可以统计整个数组在内存中的长度(使用sizsof())
- 可以获取数组在内存中的首地址
  2.说明：
- 注意，数组名是常量，不可以赋值
- 直接打印数组名，可以查看数组所占内存的首地址
- 对数组名进行sizeof，可以获取整个数组占内存空间的大小

二维数组

1.定义：

```
数据类型  数组名[ 行数 ][ 列数 ];

数据类型  数组名[ 行数 ][ 列数 ] =
{ {数据1，数据2 } ，{数据3，数据4 } };

数据类型  数组名[ 行数 ][ 列数 ] =
{ 数据1，数据2，数据3，数据4};
```

```
数据类型  数组名[  ][ 列数 ] = { 数据1，数据2，数据3，数据4};
```

2.说明：

- 以上4种定义方式，利用 第二种 更加直观，提高代码的可读性
- 在定义二维数组时，如果初始化了数据，可以省略行数

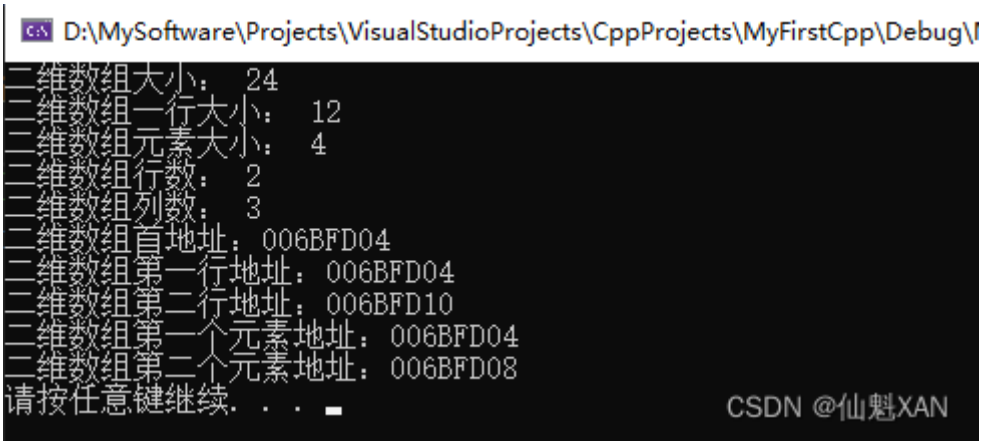
二维数组数组名

1.用途：

- 查看二维数组所占内存空间
- 获取二维数组首地址
  2.代码：

```
#include <iostream>
using namespace std;

int main() {

	//二维数组数组名
	int arr[2][3] =
	{
		{1,2,3},
		{4,5,6}
	};

	cout << "二维数组大小：  " << sizeof(arr) << endl;
	cout << "二维数组一行大小：  " << sizeof(arr[0]) << endl;
	cout << "二维数组元素大小：  " << sizeof(arr[0][0]) << endl;

	cout << "二维数组行数：  " << sizeof(arr) / sizeof(arr[0]) << endl;
	cout << "二维数组列数：  " << sizeof(arr[0]) / sizeof(arr[0][0]) <<
endl;

	//地址
	cout << "二维数组首地址：" << arr << endl;
	cout << "二维数组第一行地址：" << arr[0] << endl;
	cout << "二维数组第二行地址：" << arr[1] << endl;

	cout << "二维数组第一个元素地址：" << &arr[0][0] << endl;
	cout << "二维数组第二个元素地址：" << &arr[0][1] << endl;

	system("pause");
```

```
        return 0;
}
```



指针

指针变量的声明

指针是个变量，使用前必须声明

```
类型名 *标识符;
int *pi;
char *pc;
```

指针的运算

设a为变量，p、q为同一类型指针

```
&a                  //取变量地址
*p                  //取指针所指内容
p=q                 //指针间赋值
p++、p--或++p、--p      //指针增1或减1
p+n或p-n             //  指针加上或减去一个整数n
p-q                 //指针相减   注意！！指针不能相加，指针相减表示两指针间元素个数
p < = q             //指针关系运算
```

指针与数组

一维数组与指针

多维数组与指针

字符型指针与字符型数组

```
# include <iostream.h>
void main()
{
        char str1[]="character array";
        char *ps="character point";
        cout<<str1<<endl;
        cout<<ps<<endl;
}
```

运行结果为：
character array
character point

将一个字符串赋给一个字符型指针时，该指针指向字符串的首地址，可以通过移动该指针访问字符串的字符

字符型数组和字符型指针有以下区别：
ps是一个变量，可以改变ps使它指向不同的字符串，但是不能修改ps指向的字符串常量；
str1是数组名，可以修改数组中保存的内容

指针数组

1.定义：指针数组即数组中每个元素都是指针变量或称由指针构成的数组。
2.定义形式：

```
数据类型 *数组名[元素个数];
int a,b,c;
int *p[3]={&a,&b,&c};   //指针数组中的每个元素都是指针，初始化的值是变量的地址
```

字符型指针数组可以用以下形式初始化：

```
char s1[]="Basic";
char s2[]="Fortan";
char *ps[]={s1,s2};    //ps为指针数组
---------
//也可以直接给字符型指针数组赋字符串
char *ps1[]={"Basic","Fortran"};
```

堆内存分配

背景：设计程序时，有些数据空间的大小不能确定，只有在程序运行过程中才能确定，使用堆(heap)内存可以实现这一功能。堆是一种内存空间，它允许程序运行时根据需要申请内存空

间。由于这种空间的大小在编译和连接时不确定，而是随着程序运行可大可小，所以堆内存是动态的，又称动态内存分配。

new运算符

1.使用形式：

```
指针 = new 数据类型名
//作用是从内存的动态区域申请指定数据类型所需的存储单元
double *p;
p = new double;     //也可以对该内存单元初始化

-------------
//也可以申请一块连续的存储空间
指针 = new 数据类型[元素个数]
```

delete运算符