



C++与C的主要差异

– 动态内存分配和释放

在C++中，除了可以通过定义变量的方式来使用内存空间外，还可以使用new和delete两个关键字进行内存空间的动态分配和释放。使用动态方式分配内存时会在堆区分配内存空间来存储数据，因此动态内存分配也通常称为堆内存分配。相应地，动态内存释放也通常称为堆内存释放。

堆内存分配new的语法格式为：

```
new <数据类型>;
```

<数据类型>指定了分配的内存空间中存储的数据的类型，由于不同类型的数据需要不同尺寸的内存空间来保存，因此，<数据类型>不同，分配的内存空间大小也会不同。

堆内存分配成功后，会返回动态分配的内存空间的首地址。通常将该首地址保存在一个指针变量中，以便后面可以使用该指针变量操作内存空间中的数据。在分配内存的同时，还可以进行内存初始化工作：

```
new <数据类型>(<表达式>);
```

其中，<表达式>确定了分配的内存空间中初始存储的数据。

例如：

```
int *p;  
p=new int(3);
```

两条语句执行结束后，指针变量p就指向了通过new int动态分配的内存空间，如图所示。

P

内存地址	内存数据
------	------

0x001afe30	0xf0
------------	------

0x001afe31	0x4e
------------	------

0x001afe32	0x53
------------	------

0x001afe33	0x00
------------	------

内存地址	内存数据
------	------

0x00534ef0	0x03
------------	------

0x00534ef1	0x00
------------	------

0x00534ef2	0x00
------------	------

0x00534ef3	0x00
------------	------

也可以动态分配用于存储多个数据元素的内存空间，语法格式为：

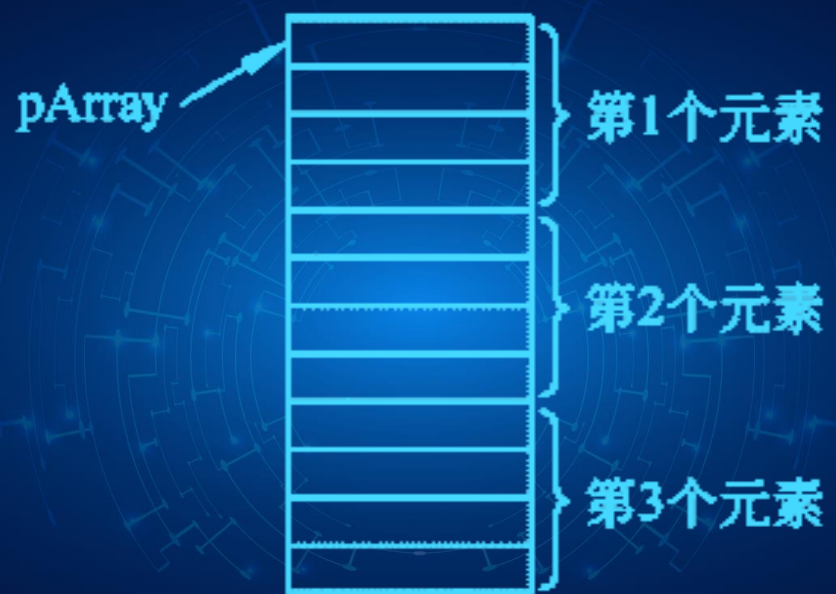
```
new <数据类型>[<表达式>];
```

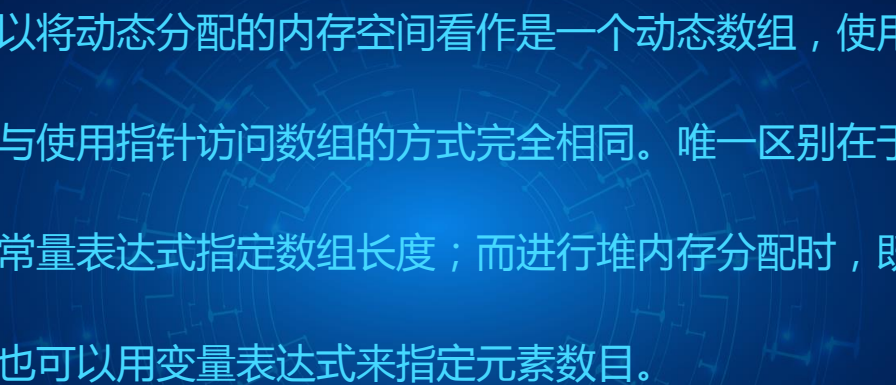
其中，<表达式>既可以是常量也可以是变量，但必须是整数，用于指定元素数目。例如：

```
int *pArray;
```

```
pArray=new int[3];
```

两条语句执行结束后，指针变量pArray就指向了通过new int[3]动态分配的内存空间，如图所示。





实际上，可以将动态分配的内存空间看作是一个动态数组，使用指针访问堆内存的方式与使用指针访问数组的方式完全相同。唯一区别在于：数组定义时，必须用常量表达式指定数组长度；而进行堆内存分配时，既可以使用常量表达式，也可以用变量表达式来指定元素数目。

使用new分配的内存必须使用delete释放，否则会造成内存泄露（即内存空间一直处于被占用状态，导致其他程序无法使用）。当系统出现大量内存泄露时，系统可用资源也会相应减少，从而导致计算机处理速度变慢，当系统资源枯竭时甚至会造成系统崩溃。堆内存释放delete的语法格式为：

```
delete [] <指针表达式>;
```

其中，<指针表达式>指向待释放的堆内存空间的首地址。


```
delete []p;
```

```
delete []pArray;
```

两条语句可以将前面使用new int(3)和new int[3]动态分配的内存空间释放。

如果<指针表达式>所指向的堆内存空间只包含一个元素，那么还可以将[]省掉，

即：

```
delete <指针表达式>;
```

例如：

```
delete p;
```

上面的语句可以将前面使用new int(3)动态分配的内存空间释放。由于pArray所指向的内存空间中包含3个元素，因此不能使用delete pArray来释放这些内存空间，而必须使用delete []pArray。

提示：在使用new分配堆内存时要区分()和[]。()中的表达式指定了内存的初值，而[]中的表达式指定了元素数目。例如：

```
p=new int(3);      //分配了1个int型元素大小的内存空间,且  
其初值为3
```

```
pArray=new int[3];//分配了3个int型元素大小的内存空间
```