10_C++新成员

动态内存分配

- C++ 中通过 `new` 关键字进行动态内存申请
- `new` 开辟的空间存**储在堆上**，而我们定义的变量存储在栈上
- C++ 中的动态内存申请是**基于类型进行**的
- `delete` 关键字用于内存释放，且**仅仅是作用于** `指针`

```
// 变量申请
Type* pointer = new Type;

// 单个变量内存释放
delete pointer;

// 数组申请
Type* pointer = new Type[N];


// 数组内存释放
deleter[] pointer;

```

`lesson_10-1.cpp`

```
#include <stdio.h>

int main(void)
{
    int *p = new int;
    *p = 5;
    *p = *p + 10;

    printf("p = %p\n", p);
    printf("*p = %d\n", *p);

    delete p;
    p = new int[10];

    for (int i = 0; i < 10; i++)
    {
        *(p + i) = i + 1;
        printf("p[%d] = %d\n", i, *(p + i));
    }

    delete[] p;

```

```
    return 0;
}
```

> C/C++中 动态内存的申请字节数计算，如：
> `int * p = malloc(40)` 和 `int * p = new int[10]`
> 问：以上申请的字节数是多少字节数?
> 答：**至少**40个字节，该malloc库函数与new关键字动态申请的内存空间不一定是 `40` 个字节，可能多点，只是系统会保证你至少能够使用40个字节

注意以下写法：

```
int* pi = new int(5);  // 申请 1个int大小的空间，然后将这块空间初始化为 5，最后将这块空间的首地址赋值给 pi
int* pi = new int[5];  // 申请 连续5个int类型大小的空间，将这块空间的首地址赋值给 pi
```

`new` 关键字与 `malloc` 函数的区别

- new 关键字是C++的一部分
- malloc是由C库提供的函数
- new 以**具体类型为单位**进行内存分配
- malloc 以**字节为单位**进行内存分配
- new 在申请单个类型变量时可进行初始化
- malloc 不具备内存初始化的特性

new 关键字的初始化

```
int * pi = new int(1); // 申请 以int大小的内存空间，然后将这块空间初始化为 1，最后将这块空间首地址赋值给 pi
float * pf = new float(2.0f);  // 申请 以float大小的内存空间，然后将这块空间初始化为 2.0，最后将这块空间首地址赋值给 pf
char * pc = new char('c');  // 申请 以char大小的内存空间，然后将这块空间初始化为 'c'，最后将这块空间首地址赋值给 pc
```

C++中的命名空间

- 在C语言中**只有一个全局作用域**
  - C语言中所有的全局标识符共享同一个作用域
  - 标识符之间可能发生冲突
- C++中提出了命名空间的概念

- 命名空间将**全局作用域分成不同的部分**
- 不同命名空间中的标识符可以同名而不会发生冲突
- 命名空间可以相互嵌套
- 全局作用域也叫 **默认命名空间**

C++命名空间的定义

```
namespace Name
{
    namespace Internal
    {
        /* ... */

    }

    /* ... */
}
```

> 命名空间无论怎么划分，本质还是全局作用域，在命名空间定义的变量及函数，都是全局变量，全局函数

C++命名空间的使用

- 使用整个命名空间：`using namespace name;`
- 使用命名空间中的变量：`using name::variable;`
- 使用默认命名空间中的变量：`::variable`

实例分析

`lesson_10-3.cpp`

```
#include <stdio.h>

namespace First
{
    int i = 0;
}

namespace Second
{
    int i = 1;

    namespace Internal
    {
        struct P
        {
            int x;
```

```
            int y;
        };
    }
}

int main(void)
{

    using namespace First;
    using Second::Internal::P;

    printf("First::i = %d\n", i);
    printf("Second::i = %d\n", Second::i);

    P p = {2, 3};
    printf("p.x = %d\n", p.x);
    printf("p.y = %d\n", p.y);


    return 0;
}
```

小结

- C++ 中内置了动态内存分配的专用关键字
- C++ 中的动态内存分配可以**同时进行初始化**
- C++ 中的动态内存分配是**基于类型进行的**
- C++ 中的命名空间概念用于**解决名称空间冲突的问题**