第6章 C++的其他特性

In [1]:
```
#include <iostream>
using namespace std;
```

In [2]:
```
// 禁用对特定类型转换的警告
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wliteral-conversion"
```

函数模板

In [3]:
```
template <class T>
void mySwap(T &t1, T &t2 ){
    T temp = t1;
    t1 = t2;
    t2 = temp;
}
```

In [4]:
```
int a = 2;
int b = 3;
mySwap(a,b);
cout << a << " " << b;
```

```
3 2
```

In [5]:
```
float c = 1.2;
float d = 99;
mySwap(c,d);
cout << c << " " << d;
```

```
99 1.2
```

1.上面编译器在遇到调用函数的语句时，会现将函数模板转换为对应的函数，然后进行编译
2.上面对函数模板转换时，会自动进行类型匹配，如果匹配失败，编译器会报错
3.也可以显示进行函数模板使用，如下

In [6]:
```
mySwap<float>(c,d);
cout << c << " " << d;
```

```
1.2 99
```

类模板

In [7]:
```
template <class T>
class myVector{
    public:

    // 定义成员变量
        T *data;
        int size;

    //构造函数和析构函数
        myVector(int s){
            size = s;
            data = new T[size];
        }
        ~myVector(){delete [] data;}
};
```

使用类模板时，必须指定模板类型

In [8]:
```
myVector<int> v1 = myVector<int>(3);
```

In [9]:
```
v1.data[0] = 3.14;
cout << v1.data[0];
```

```
3
```

In [10]:
```
myVector<float> v2 = myVector<float>(3);
```

In [11]:
```
v2.data[0] = 3.14;
cout << v2.data[0];
```

```
3.14
```

引用

引用简单来说就是变量的别名，类似指针，但是通过引用访问变量以及传参比使用指针更加简洁

In [12]:
```
int n=100;
int &a=n;
a=200;
cout << n;
```

```
200
```

const 关键字

const就是constant的缩写，使用const定义变量可以使变量变成常量，不可修改，保护其值；
这里主要容易混淆的是指针常量和常量指针，下面着重讲这两个

指针常量
用const修饰指针指向的那个量,保护指针所指向的值，使其不可修改

In [13]:
```
const int *p1 = &n;
```

In [14]:
```
*p1 = 300;  // 这里通过指针修改其指向的变量的值，会报错
```

```
input_line_21:2:6: error: read-only variable is not assignable
 *p1 = 300;  // 这里通过指针修改其指向的变量的值，会报错
 ~~~ ^
Interpreter Error:
```

常量指针
就是常指针，这个指针本身不可修改，但是其指向的值可以修改

In [15]:
```
int * const p2 = &n;
```

In [16]:
```
*p2 = 300;
cout << n;
```

```
300
```

动态内存分配

在c++中，使用new关键字分配动态空间，不在函数栈当中，
用完后要用delete释放空间

In [17]:
```
int *p3 = new  int[20];
```

In [18]:
```
delete [] p3;
```

In [19]:
```
class Student{
    public:
        int a;
}
```

In [20]:
```
Student *p4 = new Student;
```

In [21]:
```
delete p4;
```

使用new创建的对象，返回的是这个对象的地址，要用相应的指针接收

auto关键字

无论初始化一个什么类型的变量，都可以用auto定义，c++自动判断类型
typeid().name()返回的字符串意义：
i 表示int，b=bool c=char s=short
l=long f=float, d=double x=long long

In [22]:
```
auto a = 4;
typeid(a).name()
```

Out[22]:
```
"i"
```

In [23]:
```
auto b = 3.14;
typeid(b).name()
```

Out[23]:
```
"d"
```

In [24]:
```
char c[]= "hello";
typeid(c).name()
```

Out[24]:
```
"A6_c"
```

"A6_c" 这个字符串表示的含义如下：

- "A": 表示这是一个数组类型。
- "6": 表示数组的长度为 6（包括字符串末尾的空字符 '\0'）。
- "_c": 表示数组元素的类型是字符类型（char）。

In [25]:
```
template <class T>
class Complex{
    public:
        T real,imag;
        Complex(T r=0, T i=0){real = r; imag = i;}
        Complex operator+(Complex &c2){
            return Complex(real+c2.real,imag+c2.imag);
        }
};
```

In [26]:
```
auto x1 = Complex<int>(3,4);
```

In [27]:
```
typeid(x1).name()
```

Out[27]:
```
"N12__cling_N5277ComplexIiEE"
```

全局变量与局部变量

全局变量就是在所有函数外面声明的变量，不是保存在函数栈中，而是存储在data区，
如果在函数或者类中声明一个全局变量，需要加上关键字 static ，也就是通过
static声明的变量，是一个静态变量，也就是全局变量，即便在对象里面声明，它也
不属于对象，不需要通过对象点的访问方式，可以直接访问。
并且全局变量只在第一次声明有效，后面重复声明会直接跳过。

In [29]:
```
void func(){
    static int zzz = 1;
    zzz++;
    cout << zzz << endl;
}
```

In [30]:
```
func();
func();
func();
```

```
2
3
4
```

In [ ]: