构造析构

构造

- 用于数据成员的初始化
- 没有返回值，函数名与类名相同
- 构造函数可以被重载
- 当构造函数只有一个参数的时候，可以使用赋值运算符来初始化对象（隐式转换）
- 关键字`explicit`可以禁止隐式转换，只能显式调用
- 无参构造为默认构造，当没有重载构造，则编译器会提供一个默认构造，当重载了，则编译器不会提供默认构造
- `= default`指明使用编译器提供的默认构造
- `= delete`指明虽然提供了声明，但函数不可用

析构

- 函数名与类名相同，无参数，函数名前有`~`
- 析构不能被重载
- 析构不能使用`= delete`

构造析构的调用时机

定义（创建）的时候调用构造，销毁的时候调用析构——即生命周期的开始与结束

- 局部对象
  - 定义的时候调用构造，
  - 出函数作用域调用析构
- 全局对象
  - 构造执行早于`main`函数
  - 析构执行晚于`main`函数
- 堆对象
  - `new`的时候调用构造
  - `delete`的时候调用析构

手动调用构造和析构

不建议手动调用构造和析构

```
Mystring str;
str.Mystring::Mystring(...);    // 手动调构造
str.Mystring::~Mystring();      // 手动调析构
```

new与delete

new

1. 申请堆空间
2. 调用构造

delete

1. 调用析构
2. 释放堆空间

new[]与delete[]

```
int *arr = new int[5];
delete[] arr;
```

C++11特性

```
int *arr = new int[5] {1, 2, 3, 4, 5};
Mystring *str = new Mystring[3] {"Hello", "World", "!"};
```

使用`new`申请的对象可以使用`free`来释放，但不会调用析构。但是使用`new`申请的数组不能使用`free`释放