对象传参和返回

参数对象

- 小对象
  - 直接`push`
- 大对象
  - 在往栈顶拷贝

```
mov edi, esp    ; 识别标志
rep movsd
```

拷贝构造

```
mov ecx, esp    ; 识别标志
...
call xxx
```

`ecx`为this指针且指向栈顶，函数内部直接使用`ecx`

行为特征：针对于参数，函数外构造（拷贝），函数内析构（退出前）

返回对象

```
CMyString GetStrinig()
{
    CMyString obj;
    ...
    return obj;
}

CMyString str1 = GetString();   // 此处属于拷贝构造，不属于赋值运算，不产生临时对象
/*
* 此时，隐含参数str1的地址，并且返回
* GetString()的行为：
*   1. 构造局部对象
*   2. 针对参数str1进行拷贝构造，如果没有拷贝构造，则使用默认的拷贝构造（浅拷贝，以参数为目标的memcpy）
*   3. 析构局部对象
*   4. str1的析构发生是GetString()外
*/
```

临时对象

```
CMyString GetStrinig()
{
    CMyString obj;
    ...
    return obj;
}

CMyString str1;
str1 = GetString();      // 属于赋值运算，如果定义了运算符重载则调用重载函数，否正产生临时对象，并浅拷贝
                         // 该临时对象的生命期：GetString内构造开始 --- 本条语句结束析构
/*
* 此时，隐含参数为临时对象的指针，并且返回
*/
```

无名对象

```
CMyString GetStrinig()
{
    CMyString obj;
    ...
    return obj;
}

CMyString &str1 = GetString();  // 此处产生无名对象，使用指针保留并维护这个返回对象的地址
                                // 无名对象的作用域和其引用的作用域定义一致，故不会析构这个返回对象

// 如果是下例子，则产生的是临时对象，因为无名对象必须是引用维护对象
CMyString *str1 = &GetString();
```

识别拷贝构造

1. 该函数为构造函数
2. 该函数的参数为本类对象的指针，有且仅有这一个参数
3. 在对象传参的时候，调用了此函数
4. 在对象作为返回值的时候，调用了此函数

即拷贝构造的识别可以和**参数对象**以及**返回对象**相互举证