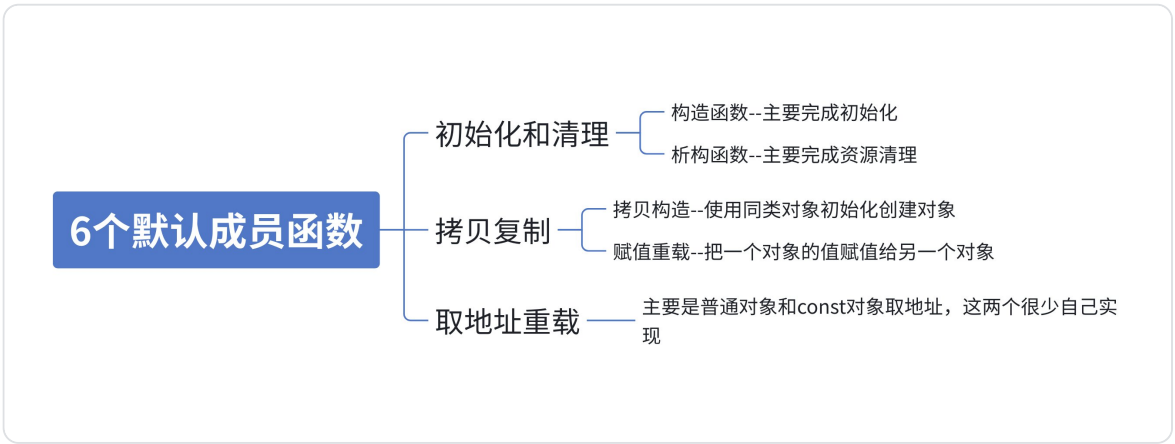
类和对象复习(中)

1. 类的默认成员函数

默认成员函数就是用户没有显示实现，编译器自动生成的函数。一个类的默认函数共有6个，但重要的只有4个，那两个取地址重载不重要。在C++11中还会增加两个默认成员函数，移动构造和移动赋值，在对C++11的复习中会提及。



2. 构造函数

构造函数不是开空间创建对象，而是对象实例化时初始化对象。

构造函数的特点：

a. 函数名与类名相同

b. 无返回值

c. 对象实例化时系统自动调用对应的构造函数

d. 构造函数可以重载

e. 如果类内没有显示实现一个构造函数，那么编译器会自动实现一个无参的默认构造函数。

f. 无参构造函数、全缺省构造函数、我们不写构造时编译器默认生成的构函数，都叫做默认构造函数。但是这三个函数有且只有一个存在，不能同时存在。无参构造函数和全缺省构造函数虽然构成函数重载，但是调用时会有歧义。实际上无参构造函数、全缺省构造函数也是默认构造，也就是说**不穿实参就可以调用的构造就叫默认构造。**

g. 不显示实现时，编译器默认生成的构造，对内置类型的成员变量的初始化是随机的，由编译器决定。对于自定义变量，如果它没有默认构造来初始化，那么编译器就会报错。

说明：C++把类型分成内置类型(基本类型)和自定义类型。内置类型就是语言提供的原生数据类型，

如：int/char/double/指针等，自定义类型就是我们使用class/struct等关键字自己定义的类型。

3. 析构函数

析构函数与构造函数功能相反，析构函数不是完成对对象本身的销毁，比如局部对象是存在栈帧的，函数结束栈帧销毁，他就释放了，不需要我们管，C++规定对象在销毁时会自动调用析构函数，完成对象中资源的清理释放工作。

析构函数的特点：

a. 析构函数名是在类名前加上字符 ~。

b. 参数无返回值。

c. 一个类只能有一个析构函数，若未显示实现，系统会自动生成默认的析构函数。

d. 对象生命周期结束时会自动调用析构函数。

e. 不显示实现时，编译器对内置类型不做处理，自定义类型会调用它的析构函数。

f. 即是我们显示实现析构函数时，对于自定义类型它也会调用它自己的析构函数，也就是说自定义类型在任何时候都会调用它自己的析构函数。

g. 如果类中没有申请资源时，析构函数可以不写，让编译器生成默认析构函数。

h. 一个局部域多个对象，C++规定后定义的西安西析构。

4. 拷贝构造函数

如果一个构造函数的第一个参数是自身类类型的引用，且任何额外的参数都有默认值，则此构造函数 也叫做拷贝构造函数，也就是说拷贝构造是一个特殊的构造函数。

拷贝构造的特点：

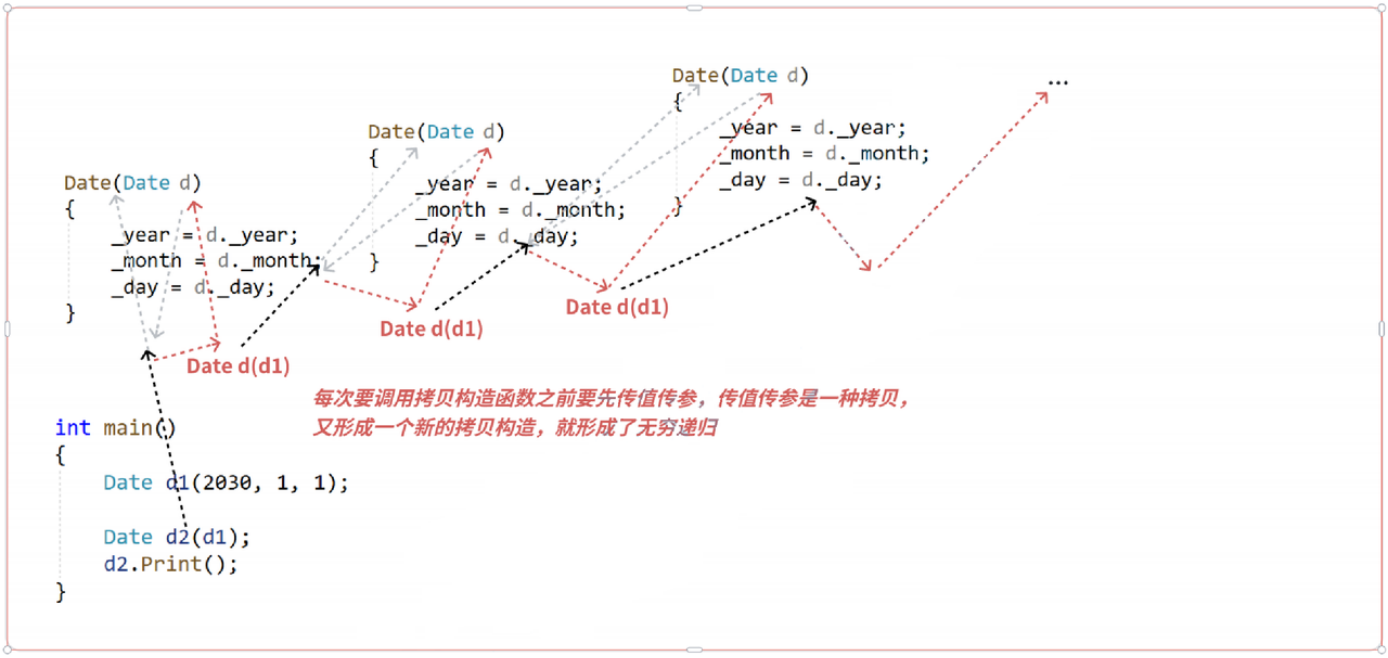
a. 拷贝构造函数是构造函数的重载

b. 拷贝构造函数的第一个参数类型必须是类类型对象的引用，否则会导致无穷递归，导致报错。拷贝构造函数也可以多个参数，但是第一个参数必须是类类型对象的引用，后面的参数必须有缺省值。

c. C++规定自定义类型对象进行拷贝行为必须调用拷贝构造，所以这里自定义类型传值传参和传值返回都会调用拷贝构造完成。

d. 编译器生成的默认拷贝构造对内置类型是浅拷贝，对自定义类型会调用它的拷贝构造。

e. 传值返回会产生一个临时对象调用拷贝构造，传值引用返回，返回的是返回对象的别名(引用)，没 有产生拷贝。但是如果返回对象是一个当前函数局部域的局部对象，函数结束就销毁了，那么使用引用返回是有问题的，这时的引用相当于一个野引用，类似一个野指针一样。传引用返回可以减少拷贝，但是一定要确保返回对象，在当前函数结束后还在，才能用引用返回。



5. 赋值运算符重载

5.1 运算符重载

i. 运算符重载是具有特殊名字的函数，他的名字是由operator和后面要定义的运算符共同构成。和其他函数一样，它也具有其返回类型和参数列表以及函数体。

ii. 重载运算符函数的参数个数和该运算符作用的运算对象数量一样多。一元运算符有一个参数，二元运算符有两个参数，二元运算符的左侧运算对象传给第一个参数，右侧运算对象传给第二个参数。

iii. 如果一个重载运算符函数是成员函数，则它的第一个运算对象默认传给隐式的this指针，因此运算符重载作为成员函数时，参数比运算对象少一个。

iv. 不能通过连接语法中没有的符号来创建新的操作符：比如operator@

v. **.* :: sizeof ?: .** 这五个运算符不能重载。(选择题常考)

vi. 重载操作符至少有一个类类型参数，不能通过运算符重载改变内置类 型对象的含义，如：`int operator+(int x, int y)`

vii. 重载<<和>>时，需要重载为全局函数，因为重载为成员函数，this指针默认抢占了第一个形参位置，第一个形参位置是左侧运算对象，调用时就变成了 对象<<cout，不符合使用习惯和可读性。重载为全局函数把ostream/istream放到第一个形参位置就可以了，第二个形参位置当类类型对象。

5.2 赋值运算符重载

赋值运算符重载是一个默认成员函数，用于完成两个已经存在的对象直接的拷贝赋值，这里要注意跟拷贝构造区分，拷贝构造用于一个对象拷贝初始化给另一个要创建的对象。

赋值重载的特点：

i. 赋值运算符重载是一个运算符重载，规定必须重载为成员函数。赋值运算重载的参数建议写成const 当前类类型引用，否则会传值传参会有拷贝。

ii. 有返回值，且建议写成当前类类型引用，引用返回可以提高效率，有返回值目的是为了支持连续赋值场景。

iii. 默认生成的赋值重载对内置类型是浅拷贝，对自定义类型则会调用它的赋值重载。

代码块

```
Date& operator=(const Date& d)
{
    // 不要检查自己给自己赋值的情况
    if (this != &d)
    {
        _year = d._year;
        _month = d._month;
        _day = d._day;
    }
    // d1 = d2表达式的返回对象应该为d1，也就是*this
    return *this;
}
```

6. 取地址运算符重载

6.1 const成员函数

a. const放在成员函数参数列表后，这样的成员函数被称为const成员函数

b. const实际修饰成员函数隐含的this指针，表面该成员函数不能对类内的任何成员函数进行修改。例如：Date* const this -> const Date* const this

6.2 取地址运算符重载

只有在一些特殊场景中才会使用，比如不想让别人取到当前对象的地址。

代码块

```
class Date
{
public:
    Date * operator&()
    {
        return this;
        // return nullptr;
    }

```

```
10      const Date * operator&()const
11      {
12          return this;
13          // return nullptr;
14      }
15  private:
16       int _year; // 年
17       int _month; // 月
18       int _day; // 日
19   };
```