第七章

1.合成的默认构造函数

当我们在没有定义任何构造函数时,编译器会默认给我们生成,一个,他基本上时什么都不做的,一般使用在有继承关系的类中,初始化父类.

合成的默认构造函数执行初始化遵循的规则

1. 当类内有类内初始值时(c11新标准),时执行类内初始值进行初始会
2. 执行默认初始化,对于内置类型就是,不初始化,也就是未定义,对于自定义的类型,执行该类型默认构造函数.

注意;

1.只有当我们没有定义任何构造函数时,编译器才会帮我们生成,另外,如果我们自定义了构造函数,就不会有这个默认构造函数了.

这是我们可以通过=default告诉编译器保留默认构造函数,详细用法见: 

2.使用合成的构造函数可能是危险的,只有在你当全部给定类内初始且,自定义类型有默认构造函数才可以

2.类型成员

在class中由类定义的类型成员同样有,访问限制,是由using,或者typdef

1. 对于类内成员函数,即使不加inline默认也是inline,能不能实现,有编译器决定;
2. mutable

在变量的声明时加上mutable,即使与const和在,const成员函数中,也可以改变他的值

1. 友元之令成员函数作为友元

比如win class中的clear()函数似乎,screen的友元可以调用,screen的成员

则这两个类在设计时应该遵循以下的顺序

首先先给出win class的clear()的声明;

class win {

public:

void clear ();

};

接着是友元的声明,和screen的声明和定义

class screen {

friend void win::clear ();

public:

void get()

{

cout<<i<<endl;

}

private:

int i=0;

};

void win::clear ()

{

screen \*p=new screen();

p->get();

delete p;

}

1. 友元的声明

友元的声明仅仅指定了访问权限,而非一个通常意义上的声明,我们为了,用户可以调用这个函数,必须在友元声明之外,再真正意义的声明一次.(许多编译器并不强制检查这个错误)

1. 类型名的特殊处理

如果再外层作用域中定义了,一个类型的别名,再类中使用了该别名后,就不能,重新将其他类型定义为该别名,即使类型一样,也不可以(许多编译器并不会检查该错误)

8.