**实验十五 组合与继承**

1. **实验目的和要求**

在实验14编程题第（3）题实现的LongLongInt类的基础上，创建一个带符号的任意长的整数类型。该类型支持输入输出、比较操作、加法操作、减法操作、++操作和--操作。用组合和继承两种方法实现。

**二、实验内容**

1. 实验准备
   1. 理论知识介绍
      1. 数据的输入：

要从用户依次读入int类型的变量a,n，可以使用如下语句:

cin >> a>>n;

* + 1. 数据的输出

要将变量的内容显示在显示器上，可以使用cout和流插入运算符。Cout还可以输出表达式的执行结果，比如:

cout << a << ‘+’ << b << ‘=’ << a+b << endl;

* + 1. 算术表达式

C++中，算术运算里有+（加法），-（减法），/（除法），%（取模），但是没有取平方算数运算。

* + 1. 数据类型 （整型）：

C++中一个整型变量可以储存一个整数，可以直接通过cin和cout输入和输出。

* + 1. 函数

函数定义的形式如下：

类型名 函数名（形式参数列表）

{

变量定义部分

语句部分

}

* + 1. 类的定义

定义一个类就是定义一组属性和一组堆属性进行操作的函数。属性称为类的数据成员，而函数称为类的成员函数。类定义的一般形式如下：

class 类名{

private:

私有数据成员和成员函数;

public:

公有数据成员和成员函数;

};

* + 1. 对象的使用

在程序中直接定义对象的方法与定义普通函数的方法一样，他的格式如下：

存储类别 类名 对象列表;

对象可以用如下的语句操作：

结构体变量名.成员名

* + 1. 对象的构造

有了构造函数之后，对象的定义的一般形式如下：

类名 对象名(实际参数表)

* + 1. 对象的析构

一般析构函数的定义是：

~类名() {

函数体语句

}

* + 1. 模块化开发

结构化程序设计的基本思想是在设计阶段采用“自顶向下，逐步求精” 的方法，将大问题划分成一系列小问题，这样做提高了程序的可靠性，并且保证了程序的质量。

* + 1. 运算符重载

要使C++知道如何对两个有理数的对象r1和r2执行r1+r2，我们需要“教会”C++如何完成这项任务。教会C++如何对类类型的对象执行内置运算符的操作称为运算符重载。

* + 1. 组合和继承

面向对象的重要特征之一是代码重用。C++中有两种方法可以完成这个任务。第一种方法是用已有的类的对象作为新定义类的数据成员。因为新的类是是已有类的对象组合而成，所以这种方法被成为组合。第二种方法是在一个已存在类的基础上，对它进行扩展，形成一个心累。这种方法称为继承。

1. 实验项目
   1. 分析
      1. 由于需要用到实验14(3)题目中的LongLongInt类，所以要先将13(4)中的LongLongInt类进行运算符的重载。
      2. 组合和继承两种方法所实现的功能差不多，代码也基本上相同，所以主要就完成一种代码的重用就可以。
   2. 方案
      1. 主要要先构造一种LongLongInt的基类，分为long\_long\_int.h和long\_long\_int.cpp两种文件
      2. 利用组合的方法的时候，构造一个SLLIC类，分为signed\_lli\_combine.h 和signed\_lli\_combine.cpp 两个文件，这两个文件都要include long\_long\_int.h。
      3. 利用继承的方法构造一个SLLII类，方法和构造SLLIC类类似。
   3. 测试数据

SLLIC sllic1, sllic2, sllic3;

cin >> sllic1 >> sllic2 >> sllic3;

cout << (sllic1 > sllic2) << endl;

cout << (sllic1 < sllic3) << endl;

cout << sllic1 + sllic2;

cout << sllic1 - sllic3;

sllic1++;

cout << sllic1;

--sllic1;

cout << sllic1;

SLLII sllii1, sllii2, sllii3;

cin >> sllii1 >> sllii2 >> sllii3;

cout << (sllii1 > sllii2) << endl;

cout << (sllii1 < sllii3) << endl;

cout << sllii1 + sllii2;

cout << sllii1 - sllii3;

sllii1++;

cout << sllii1;

--sllii1;

cout << sllii1;

return 0;

* 1. 运行结果

-1234

123

-12

0

0

-1111

-1222

-1233

-1234

-1234

123

-12

0

0

-1111

-1222

-1233

-1234

**三、实验小结**

1. 完成情况

完成了题目要求的所有功能，但是代码不够整洁。

1. 重点

运算符重载，类的组合和继承。

1. 难点

在类的组合和继承过程中对类的成员的权限的控制以及构造函数和析构函数中对数组空间的正确分配。

1. 解决方法

在继承类的构造函数中包含基类的构造函数，并防止动态数组被delete多次。

1. 有待改进之处

程序中的分支过多，某些分支可以简化，代码重用率不高。

1. 收获、体会

在构造复杂的类的对象的时候，一定不能忽视构造函数的重要性，基类的构造函数可以用来简化派生类的构造函数。动态数组的空间分配很容易出错。