北 京 林 业 大 学

2019 学年—2020 学年第 1 学期 面向对象程序设计 B

实验指导书

专业名称： 地信 实验学时： 2 任课教师： 卢昊

实验题目：实验四 C++高级特性

实验环境：Visual Studio

实验目的

1. 掌握基本的运算符重载方法
2. 掌握函数模板和类模板方法
3. 了解文件流的使用
4. 掌握 C++异常处理方法
5. 了解基本的 STL 容器用法。

实验要求

1. 掌握 Visual Studio 开发工具的基本用法；
2. 按要求创建、命名和管理解决方案、工程目录；
3. 使用 C++程序写法，而非 C 语言写法；
4. 将程序测试运行通过后再整理提交，严格按照命名要求命名；
5. 实验完成后认真撰写实验报告。

实验内容和步骤

注意：本次实验只需创建 **1 个解决方案**，名称为“**你的学号_s4**”，如“180XXXXXX_s4”。实验的所有习题均**在该解决方案下创建独立的工程**。创建方法请参看实验一指导书，规范命名和存放目录，本指导书不再赘述。

重要：所有工程在创建时均设置为“空项目”，避免 VS 自动创建代码。

练习 1：运算符重载

我们把形如 z=a+bi（a,b 均为实数）的数称为复数，其中 a 称为实部，b 称为虚部，i 称为虚数单位。当虚部等于零时，这个复数可以视为实数；当 z 的虚部不等于零时，实部等于零时，常称 z 为纯虚数。

复数的**加法法则**：设 z1=a+bi，z2=c+di 是任意两个复数。两者和的实部是原来两个复数实部的和，它的虚部是原来两个虚部的和。两个复数的和依然是复数。**减法法则**只需将和换成差。由于复数不是 C++标准数据类型，可利用运算符重载特性实现复数的加法和减法。同时假定复数的自增和自减分别实部和虚部分别自增、自减，请以同样的方式实现。

步骤 1：在解决方案中创建工程，命名为“学号_Complex”，再添加**同名的 cpp** 文件。

步骤 2：在 cpp 文件中完成以下类的设计实现。

添加 **Complex** 类的定义，包括：

- double 类型的实部 r 和虚部 i，
- 带有默认参数（全 0）的构造函数，
- 复数相加的“+”运算符重载函数，
- 复数相减的“-”运算符重载函数，
- 复数自增的“++”运算符重载函数，
- 复数自减的“--”运算符重载函数。
- print 函数，用于输出复数的值。

在该 cpp 文件中写出 Complex 类的实现部分，提示：可用重载为友元函数

或类的成员函数的方式实现。

步骤 3：在该 cpp 文件中编写 main 函数，声明 Complex 类对象 c1(10+5i)和 c2(-15+i)测试运行，要求测试每一个成员函数和运算符函数。

思考：

- 本题中，运算符重载为友元函数和类的成员函数后，使用上有什么不同？你觉得哪种方式更好？
- 要实现本题要求，运算符重载和函数重载的区别有哪些？

练习 2：函数模板

swap 两个数的值是一种常见的变量操作，如果需要交换不同类型的值，一种方式是利用参数类型不同来重载 swap 函数，如 swap(int& a, int&b)和 swap(double& a, double& b)可以分别应对 a、b 分别为 int 和 double 类型的情况。但是 C++模板特性可以使我们只写一个 swap 模板，让编译器根据 swap 模板自动生成多个 swap 函数，提高代码的复用性，节约开发成本。

步骤 1：在解决方案中创建工程，命名为“学号_func_template”，再添加**同名的 cpp** 文件。

步骤 2：在该 cpp 文件中创建名称为“_你的学号”命名空间

步骤 3：在命名空间中设计 swap 函数模板。

步骤 4：在该 cpp 文件中编写 main 函数，测试代码如下：

```
int main()
{
    int n = 1, m = 2;
    cout << n << '\t' << m << endl;
    _你的学号::swap(n, m);
    cout << n << '\t' << m << endl;

    double f = 1.2, g = 2.3;
    cout << f << '\t' << g << endl;
    _你的学号::swap(f, g);
    cout << f << '\t' << g << endl;

    char c = 'A', d = 'a';
    cout << c << '\t' << d << endl;
    _你的学号::swap(c, d);
    cout << c << '\t' << d << endl;

    return 0;
}
```

思考：

- swap 函数交换 a 和 b 的值应注意如何声明类型？

尝试在程序中做出相应修改看你的答案是否正确。

练习 3：类模板

设计一个 Operator 模板类以支持基本数据类型的加减乘除运算。

步骤 1：在解决方案中创建工程，命名为“你的学号_class_template”，再添加**同名的 cpp** 文件。

步骤 2：在该 cpp 文件中完成 Operator 类模板的设计实现。包括以下函数：

add(T a, T b);

minus(T a, T b);

multiply(T a, T b);

divide(T a, T b);

步骤 3：在该 cpp 文件中编写主函数 main，测试代码如下：

```
int main()
{
    Operator<int> op1;
    cout << op1.add(1, 2) << endl;

    Operator<string> op2;
    cout << op2.add("D.T.", "Software") << endl;

    Operator<double> op3;
    cout << op3.minus(1.5, 2.8) << endl;

    Operator<float> op4;
    cout << op4.multiply(10, 2.5) << endl;

    Operator<float> op5;
    cout << op5.divide(100.0, 24) << endl;

    return 0;
}
```

思考：

- 该程序中，若声明 Operator<string> op6 并调用 minus 函数会怎样？为了避免这种情况，可以如何处理？（提示：运算符重载）

尝试在程序中做出相应修改看你的答案是否正确。

练习 4：文件流

C 语言中可使用 FILE 结构体读写二进制文件，C++则可以使用文件流。请将**实验一练习 4** 程序中 FILE 结构体读取文件的部分改写成 C++文件流读取。

步骤 1：在解决方案中添加新的工程，命名为“你的学号_fstream”，再添加**同名的 cpp** 文件。

步骤 2：在该 cpp 文件中完成程序实现，要求结果一致。提示：参考课件二进制文件读写（代码 6-11），使用 ifstream 或 fstream 实现文件读入。

练习 5：异常处理

练习 4 的程序中，文件流打开文件需要传入正确的文件路径名称，若文件路径名称不对，程序运行会如何？在 C 语言版程序中，我们使用了 if 语句判断 FILE 结构体是否有效，实际上也可以使用异常处理机制进行错误的处理。

步骤 1：在解决方案中添加新的工程，命名为“你的学号_exception”，再添加同名的 cpp 文件。

步骤 2：将练习 4 代码拷贝到该 cpp 文件。将文件流打开文件的语句部分放在 try 语句块中，在恰当位置 throw 异常，并在恰当位置 catch 异常和处理。

步骤 3：测试运行。将文件路径名称改成不存在的文件名称，测试异常处理是否有效。

思考：

- 异常发生后，程序后面应该怎么做？退出 or 继续执行？若退出，应当做什么工作？（提示：清理资源，如若有动态申请的内存）

练习 6：STL

vector 是一种简单实用的 STL 容器，在很多情况下可以用作动态数组。设计一个类，类包含一个 vector<float>类型成员变量用于存储用户随机输入的任意个数的数字。

步骤 1：在解决方案中添加新的工程，命名为“你的学号_stl”，再添加同名的 cpp 文件。

步骤 2：在该 cpp 文件中设计类 DynamicArray，该类包含：

- vector<float>类型的成员变量 vec，
- 成员函数 input();
- 成员函数 output();

要求：在 input 函数中利用循环输入数字，并使用 Ctrl+Z 结束输入（这里的 Ctrl+Z 相当于 EOF，参考：百科，CSDN）。然后结合**迭代器 iterator** 的使用输出全部数字的值，最后显示当前 vector 对象的 size 和 capacity。**提示**：可参考课件（代码 8-1、8-4）

步骤 3：在该 cpp 文件中编写 main 函数测试。

思考：

- 尝试多次运行输入不同个数的数字，查看 size 和 capacity 的情况，是否有规律可循？（提示：可在循环中实时输出它们的值）

附加题：自定义 myString 类

实现自己的字符串类，要求使用 char 数组实现 myString 类，并且通过重载运算符的方法实现字符串的连接、赋值、下标访问，以及重载输入输出符号“>>”、“<<”。

步骤 1：在解决方案中添加新的工程，命名为“你的学号_mystring”，再添加同名的 cpp 文件。

步骤 2：#include 适当的头文件，using namespace std;

步骤 3：将以下类声明复制到你的代码中：

```
class myString
{
private:
    char *_elem;
public:
    myString() { _elem = NULL; }
    myString(char* s);
    myString(const char* s);
    myString(const myString &str);   // 为什么要写拷贝构造函数
    ~myString();
    myString operator+(const myString &str);
```

```
    myString operator=(const myString &str);
    char operator[](int i);
    void print() { cout << _elem << endl; }

    // 以下部分为选做内容
    friend ostream &operator <<(ostream &os, const myString &str);
    friend istream &operator >>(istream &is, myString &str);
};
```

步骤 4：实现以上声明中未实现的部分

步骤 5：用以下 main 函数测试你的实现

```
int main()
{
    myString s1("qwer");
    s1.print();
    char c[] = "asdf";
    myString s2(c);
    s2.print();
    myString s3;
    s3 = s1 + s2;
    s3.print();
    cout << s3[0] << endl;
    // 如果你没有做选做内容，请把下面三行注释掉
    myString s4;
    cin >> s4;
    cout << s4;
    return 0;
}
```

思考：

- const char*和 char*做为函数参数的区别？哪一个是更为安全的类型？
- 为什么需要拷贝构造函数？（回答这个问题的前提是你要写出一个正确的析构函数，千万不要仅仅为了得到运行结果而写一个空的析构函数！）

实验考核

1、实验完毕后上交实验报告，实验报告模板从 FTP 中下载，实验报告的内容包括实验目的、实验内容和结果分析，实验报告一律写成 Word 格式文档（.doc/.docx）。

2、将调试好的程序源代码和程序的输入输出数据的情况附到实验报告中，并对程序的数据结果进行分析，说明基本运行机理。

3、实验报告重点总结你的心得体会，遇到了什么问题，怎么解决的，对以后有什么帮助。

4、上传电子版实验报告和源程序（删除 Debug、Release 目录和其他无用中间文件，仅保留解决方案、工程等文件和程序代码文件）到 FTP 服务器 homework 目录下对应课程、班级目录中。多个文件请压缩打包，请注意文件命名格式。

5、文件打包命名为“你的学号_姓名_s4”。

6、鼓励提问、交流、讨论，严禁相互复制、抄袭，违者按 0 分处理。