**Lab Exercises**

Class: 软件1801 Name: 田豪举 StuID:\_\_201816040116\_\_\_

Date: 11.11 Score:

**Lab Exercise 1 — Modifying Class Account**

**I Lab Objectives**

In this lab, you will practice:

1. Creating member functions.
2. Invoking functions and receiving return values from functions.
3. Testing a condition using an if statement.
4. Outputting variables with stream insertion and the cout object.

**II Description of the Problem**

Modify class Account to provide a member function called debit that withdraws money from an Account. Ensure that the debit amount does not exceed the Account’s balance. If it does, the balance should be left unchanged and the function should print a message indicating "Debit amount exceeded account balance." Modify class AccountTest to test member function debit.

**III Sample Output**



**IV Your Solution**

Account::Account( int initialBalance )

{

balance = 0; //所操作账户余额

if ( initialBalance > 0 ) //检测输入账户余额合理性

balance = initialBalance; //若符合则赋予账户余额数值

if ( initialBalance < 0 )

cout << "Error: Initial balance cannot be negative.\n" << endl; //不符合显示操作错误

}

void Account::credit( int amount ) //对帐户进行存钱操作

{

balance = balance + amount;

}

void Account::debit(int num) //对帐户进行取钱操作

{

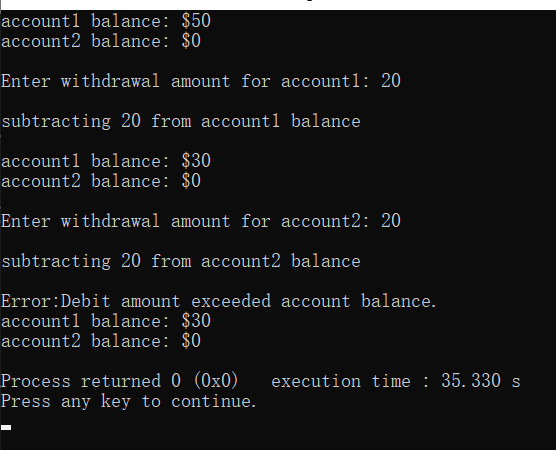
if(num<=balance)

balance=balance-num; //比较账户余额和取钱额度

else

cout<<"Error:Debit amount exceeded account balance.\n";

}



**Lab Exercise 2 — Modifying class GradeBook**

**I Lab Objectives**

In this lab, you will practice:

1. Declaring a data member.
2. Providing *set* and *get* functions to manipulate a data member’s value.
3. Declaring member functions with parameters.

**II Description of the Problem**

Modify class GradeBook. Include a second string data member that represents the name of the course’s instructor. Provide a *set* function to change the instructor’s name and a *get* function to retrieve it. Modify the constructor to specify *two* parameters—one for the course name and one for the instructor’s name. Modify member function displayMessage such that it first outputs the welcome message and course name, then outputs "This course is presented by: " followed by the instructor’s name. Modify the test application to demonstrate the class’s new capabilities.

**III Sample Output**



**IV Your Solution**

GradeBook::GradeBook( string course, string instructor ) //初始化类的信息

{

courseName=course; //初始化课程名

setInstructorName( instructor ); //读取课程教师名字

}

string GradeBook::getCourseName() //获取课程名

{

return courseName;

}

void GradeBook::setInstructorName(string instructor) //重载课程教师名

{

instructorName=instructor;

}

string GradeBook::getInstructorName() //获取课程教师名

{

return instructorName;

}

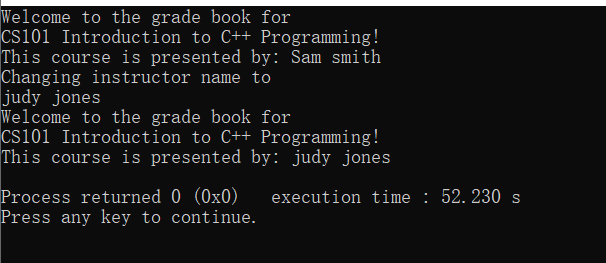
void GradeBook::displayMessage()

{

cout<<"Welcome to the grade book for\n"<<getCourseName()<<"!"<<endl; //显示课程信息

cout<<"This course is presented by: "<<instructorName<<endl; //显示课程的教师信息

}



**Lab Exercise 3 — Creating an Employee Class**

**I Lab Objectives**

In this lab, you will practice:

1. Creating a class definition.
2. Declaring data members.
3. Defining a constructor.
4. Defining set and get functions.
5. Writing a test application to demonstrate the capabilities of another class.

**II Description of the Problem**

Create a class called Employee that includes three pieces of information as data members—a first name (type string), a last name (type string) and a monthly salary (type int). Your class should have a constructor that initializes the three data members. Provide a *set* and a *get* function for each data member. If the monthly salary is not positive, set it to 0. Write a test program that demonstrates class Employee’s capabilities. Create two Employee objects and display each object’s yearly salary. Then give each Employee a 10 percent raise and display each Employee’s yearly salary again.

**III Sample Output**



**IV Your Solution**

Employee::Employee(string name1,string surn1,int salary1)

{

name=name1;

surn=surn1;

if(salary1<0) //Judge an employee's monthly salary. If it is negative, it is zero

salary=0;

else

salary=salary1\*12; //Calculation of annual salary

}

void Employee::setname(string name1)

{

name=name1;

}

string Employee::getname()

{

return surn;

}

void Employee::setsurn(string surn1)

{

surn=surn1;

}

string Employee::getsurn()

{

return surn;

}

void Employee::setsalary()

{

salary=salary\*1.1; //Calculate the employee's annual salary after a 10% raise

}

int Employee::getsalary()

{

return salary; //Obtain the data of the employee's annual salary after a 10% salary increase

}

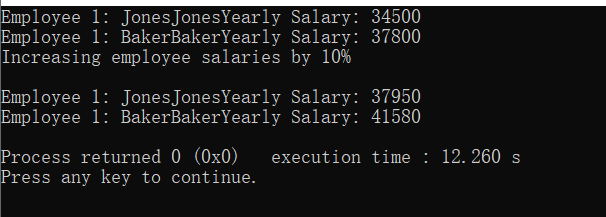
void Employee::displayMessage()

{

cout<<"Employee 1: "<<getname()<<getsurn()<<"Yearly Salary: "<<getsalary()<<endl;

//Output first name, surnname, annual salary

}



**Lab Exercise 4 — Complex Numbers**

**I Lab Objectives**

In this lab, you will practice:

1. Creating new data types by writing class definitions.
2. Defining member functions of programmer-defined classes.
3. Instantiating objects from programmer-defined classes.
4. Calling member functions of programmer-defined classes.

The follow-up questions and activities will also give you practice:

1. Initializing programmer-defined class data members with class constructors.

**II Description of the Problem**

Create a class called Complex for performing arithmetic with complex numbers. Write a program to test your class.

Complex numbers have the form：

realPart + imaginaryPart \* *i*

where *i* is

Use double variables to represent the private data of the class. Provide a constructor that enables an object of this class to be initialized when it is declared. The constructor should contain default values in case no initializers are provided. Provide public member functions that perform the following tasks:

1) Adding two Complex numbers: The real parts are added together and the imaginary parts are added together.

2) Subtracting two Complex numbers: The real part of the right operand is subtracted from the real part of the left operand and the imaginary part of the right operand is subtracted from the imaginary part of the left operand.

3) Printing Complex numbers in the form (a, b) where a is the real part and b is the imaginary part.

**III Sample Output**



**IV Your Solution**

Complex::Complex( double real, double imaginary ) //Initialize the class

{

setComplexNumber(real,imaginary);

}

void Complex::setComplexNumber(double rp,double ip) //function set to Complex

{

realPart=rp;

imaginaryPart=ip;

} // end function setComplexNumber

Complex Complex::add( const Complex &right ) //Pass in the corresponding data for the class

{ //function add the data of the class

realPart=realPart+right.realPart;

imaginaryPart=imaginaryPart+right.imaginaryPart;

return{realPart,imaginaryPart};

} // end function add

Complex Complex::subtract( const Complex &right ) //Pass in the corresponding data for the class

{ //function subtract the data of the class

realPart=realPart-right.realPart;

imaginaryPart=imaginaryPart-right.imaginaryPart;

return{realPart,imaginaryPart};

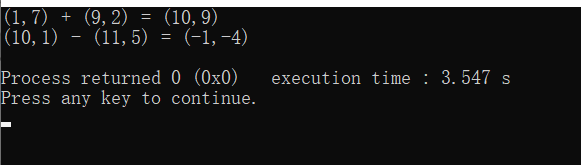
} // end function subtract

void Complex::printComplex() //The output data

{

cout<<'('<<realPart<<","<<imaginaryPart<<')';

} // end function printComplex



**V Follow-Up Questions and Activities**

1. Why do you think const was used in the parameter list of add and subtract?
2. Can add and subtract’s parameters be passed by value instead of by reference? How might this affect the design of class Complex? Write a new class definition that illustrates how the parameters would be passed by value.
3. Declare a Complex number, as follows, without passing any arguments to the constructor. What happens?Does the default constructor get called?

Complex a;

1. 保证参数在加减运算时被加减参数不会改变。
2. 12
3. 结果会将参数默认为0进行运算。调用了默认构造函数。