**Lab Exercises**

Class: 软件1801 Name: 邓海燕 StuID:\_\_\_\_\_\_201816040130\_\_\_\_\_\_

Date: 2019.11.11 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**

// Lab 1: Account.h

// Definition of Account class.

class Account

{

public:

Account( int ); // constructor initializes balance

void credit( int ); // add an amount to the account balance

//declare member function debit.

void debit(double);

int getBalance(); // return the account balance

private:

int balance; // data member that stores the balance

}; // end class Account

// Lab 1: AccountTest.cpp

// Create and manipulate Account objects.

#include <iostream>

using namespace std;

// include definition of class Account from Account.h

#include "Account.h"

// function main begins program execution

int main()

{

Account account1( 50 ); // create Account object

Account account2( 0 ); // create Account object

// display initial balance of each object

cout << "account1 balance: $" << account1.getBalance() << endl;

cout << "account2 balance: $" << account2.getBalance() << endl;

int withdrawalAmount; // stores withdrawal amount read from user

cout << "\nEnter withdrawal amount for account1: ";

cin >> withdrawalAmount; // obtain user input

cout << "\nsubtracting " << withdrawalAmount

<< " from account1 balance\n\n";

//withdraw money from account1

account1.debit(withdrawalAmount);

// display balances

cout << "account1 balance: $" << account1.getBalance() << endl;

cout << "account2 balance: $" << account2.getBalance() << endl;

cout << "\nEnter withdrawal amount for account2: ";

cin >> withdrawalAmount; // obtain user input

cout << "\nsubtracting " << withdrawalAmount

<< " from account2 balance\n\n";

//withdraw money from account2

account2.debit(withdrawalAmount);

// display balances

cout << "account1 balance: $" << account1.getBalance() << endl;

cout << "account2 balance: $" << account2.getBalance() << endl;

}

// Lab 1: Account.cpp

#include <iostream>

using namespace std;

#include "Account.h"

// initializes data member balance

Account::Account( int initialBalance )

{

balance = 0;

// if initialBalance is greater than 0, set this value as the

// balance of the Account; otherwise, balance remains 0

if ( initialBalance > 0 )

balance = initialBalance;

// if initialBalance is negative, print error message

if ( initialBalance < 0 )

cout << "Error: Initial balance cannot be negative.\n" << endl;

}

// credit (add) an amount to the account balance

void Account::credit( int amount )

{

balance = balance + amount; // add amount to balance

}

void Account::debit(double withdrawalAmount)

{

if(withdrawalAmount <= balance)

balance = balance - withdrawalAmount;

else

cout << "Debit amount exceeded account balance." << endl;

}

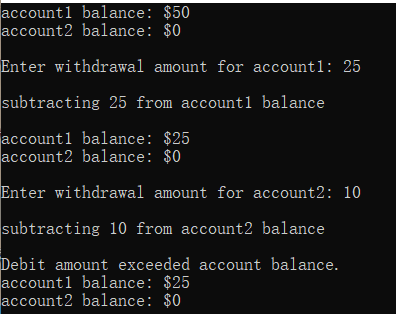
// return the account balance

int Account::getBalance()

{

return balance; // gives the value of balance to the calling function

}

**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**

// Lab 2: GradeBook.h

// Definition of GradeBook class that stores an instructor's name.

#include <string> // program uses C++ standard string class

using namespace std;

// GradeBook class definition

class GradeBook

{

public:

// constructor initializes course name and instructor name

GradeBook( string, string );

void setCourseName( string ); // function to set the course name

string getCourseName(); // function to retrieve the course name

//declare function for the instructor's name

string getInstructorName();

//declare function for the instructor's name

void setInstructorName(string);

void displayMessage(); // display welcome message and instructor name

private:

string courseName; // course name for this GradeBook

string instructorName; // instructor name for this GradeBook

};

// Lab 2: GradeBookTest.cpp

// Test program for modified GradeBook class.

#include <iostream>

using namespace std;

// include definition of class GradeBook from GradeBook.h

#include "GradeBook.h"

// function main begins program execution

int main()

{

// create a GradeBook object; pass a course name and instructor name

GradeBook gradeBook(

"CS101 Introduction to C++ Programming","Sam Smith");

// display welcome message and instructor's name

gradeBook.displayMessage();

//change instructor's name

cout << "Changing instructor name to Judy Jones\n " << endl;

GradeBook gradeBook1("CS101 Introduction to C++ Programming","Judy Jones");

gradeBook1.displayMessage();

}

// Lab 2: GradeBook.cpp

// Member-function definitions for class GradeBook.

#include <iostream>

using namespace std;

// include definition of class GradeBook from GradeBook.h

#include "GradeBook.h"

// constructor initializes courseName and instructorName

// with strings supplied as arguments

GradeBook::GradeBook( string course, string instructor )

{

setCourseName( course ); // initializes courseName

setInstructorName( instructor ); // initialiZes instructorName

}

// function to set the course name

void GradeBook::setCourseName( string name )

{

courseName = name;

}

// function to retrieve the course name

string GradeBook::getCourseName()

{

return courseName;

}

//define function for the instructor's name

string GradeBook::getInstructorName()

{

return instructorName;

}

//define function for the instructor's name

void GradeBook::setInstructorName(string name)

{

instructorName = name;

}

// display a welcome message and the instructor's name

void GradeBook::displayMessage()

{

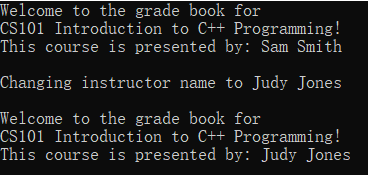
// display a welcome message containing the course name

cout << "Welcome to the grade book for\n" << getCourseName() << "!"

<< endl;

cout << "This course is presented by: " << getInstructorName() << "\n" << endl;

}



**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**

// Lab 3: Employee.h

// Employee class definition.

#include <string> // program uses C++ standard string class

using namespace std;

// Employee class definition

class Employee

{

public:

/\* Declare a constructor that has one parameter for each data member \*/

Employee(string,string,int);

//Declare methods

void setEmployeeFirstName(string);

string getEmployeeFirstName();

void setEmployeeLastName(string);

string getEmployeeLastName();

void setEmployeeMonthlySalary(int);

int getEmployeeMonthlySalary();

private:

// Declare data member

string employeeFirstName;

string employeeLastName;

int employeeMonthlySalary;

};

// Lab 3: Employee.cpp

// Employee class member-function definitions.

#include <iostream>

using namespace std;

#include "Employee.h" // Employee class definition

// Define the constructor.

Employee::Employee(string firstName,string lastName,int salary)

{

setEmployeeFirstName(firstName);

setEmployeeLastName(lastName);

setEmployeeMonthlySalary(salary);

}

// Define function for the first name data member.

void Employee::setEmployeeFirstName(string firstName)

{

employeeFirstName = firstName;

}

//Define function for the first name data member.

string Employee::getEmployeeFirstName()

{

return employeeFirstName;

}

// Define function for the last name data member.

void Employee::setEmployeeLastName(string lastName)

{

employeeLastName = lastName;

}

// Define function for the last name data member.

string Employee::getEmployeeLastName()

{

return employeeLastName;

}

// Define function for the monthly salary data member.

void Employee::setEmployeeMonthlySalary(int salary)

{

employeeMonthlySalary = salary;

if(salary < 0)

salary = 0;

}

// Define function for the monthly salary data member.

int Employee::getEmployeeMonthlySalary()

{

return employeeMonthlySalary \* 12;

}

// Lab 3: EmployeeTest.cpp

// Create and manipulate two Employee objects.

#include <iostream>

using namespace std;

#include "Employee.h" // include definition of class Employee

// function main begins program execution

int main()

{

//Create two Employee objects and assign them to Employee variables.

Employee Employee1("Bob","Jones",2875);

Employee Employee2("Baker","Baker",3150);

cout << "Employee1: " << Employee1.getEmployeeFirstName() << " " << Employee1.getEmployeeLastName() << " Yearly Salary: " << Employee1.getEmployeeMonthlySalary() << endl;

cout << "Employee2: " << Employee2.getEmployeeFirstName() << " " << Employee2.getEmployeeLastName() << " Yearly Salary: " << Employee2.getEmployeeMonthlySalary() << "\n" << endl;

/\* Give each Employee a 10% raise. \*/

cout << "Increasing employee salaries by 10%\n" << endl;

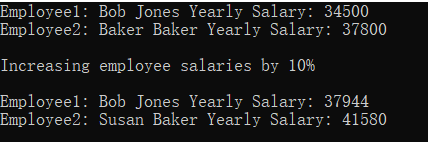
Employee Employee3("Bob","Jones",2875\*1.1);

Employee Employee4("Susan","Baker",3150\*1.1);

cout << "Employee1: " << Employee3.getEmployeeFirstName() << " " << Employee3.getEmployeeLastName() << " Yearly Salary: " << Employee3.getEmployeeMonthlySalary() << endl;

cout << "Employee2: " << Employee4.getEmployeeFirstName() << " " << Employee4.getEmployeeLastName() << " Yearly Salary: " << Employee4.getEmployeeMonthlySalary() << endl;

}

**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**

#ifndef COMPLEX\_H

#define COMPLEX\_H

/\* class definition for Complex \*/

class Complex

{

public:

Complex(double real,double imaginary);

void setComplexNumber(double,double);

void printComplex();

Complex add( const Complex &right );

Complex subtract( const Complex &right );

private:

double realPart;

double imaginaryPart;

};

#endif

#include <iostream>

using namespace std;

#include "Complex.h"

Complex::Complex( double real, double imaginary )

{

setComplexNumber( real, imaginary );

} // end Complex constructor

Complex Complex::add( const Complex &right ) /\* define function add \*/

{

Complex m(realPart = realPart + right.realPart,imaginaryPart = imaginaryPart + right.imaginaryPart);

return m;

}

Complex Complex::subtract( const Complex &right ) /\* define function subtract \*/

{

Complex n(realPart = realPart - right.realPart,imaginaryPart = imaginaryPart - right.imaginaryPart);

return n;

}

void Complex::printComplex() /\* define function printComplex \*/

{

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

}

void Complex::setComplexNumber( double rp, double ip ) /\* define function setComplexNumber \*/

{

realPart = rp;

imaginaryPart = ip;

}

// Lab 4: ComplexTest.cpp

#include <iostream>

using namespace std;

#include "Complex.h"

int main()

{

Complex a( 9, 7 ), b( 1, 2 ), c(0,0); // create three Complex objects

a.printComplex(); // output object a

cout << " + ";

b.printComplex(); // output object b

cout << " = ";

c = a.add( b ); // invoke add function and assign to object c

c.printComplex(); // output object c

cout << '\n';

a.setComplexNumber( 10, 1 ); // reset realPart and

b.setComplexNumber( 11, 5 ); // and imaginaryPart

a.printComplex(); // output object a

cout << " - ";

b.printComplex(); // output object b

cout << " = ";

c = a.subtract( b ); // invoke add function and assign to object c

c.printComplex(); // output object c

cout << endl;

} // end main



**V Follow-Up Questions and Activities**

1. Why do you think const was used in the parameter list of add and subtract?

为了不改变形参的别名

1. 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.

可以用值传递替换引用传递，如果是值传递，将耗费一定的空间和时间，比引用传递效率低

class Complex

{

public:

Complex(double real,double imaginary);

void setComplexNumber(double,double);

void printComplex();

Complex add( Complex b);

Complex subtract(Complex b);

private:

double realPart;

double imaginaryPart;

};

1. Declare a Complex number, as follows, without passing any arguments to the constructor. What happens?Does the default constructor get called?

Complex a;

构造函数会被调用，该类对象被创建时，自动调用该构造函数完成初始化工作**Lab Exercise 5 — Dates**

**I Lab Objectives**

In this lab, you will practice:

1. Using access functions and utility functions so that it is not necessary for non-member functions to be able to access a class’ data members.

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

1. Overloading constructors and using default arguments with constructors.
2. Defining a destructor.

**II Description of the Problem**

Modify the Date class to provide a member function nextDay to increment the day by one. The Date object should always remain in a consistent state. Write a program that tests function nextDay in a loop that prints the date during each iteration to illustrate that the nextDay function works correctly. Be sure to test the following cases:

1. Incrementing into the next month.
2. Incrementing into the next year.

**III Sample Output**



**IV Your Solution**

// Lab 5: Date.h

#ifndef DATE\_H

#define DATE\_H

class Date

{

public:

Date( int = 1, int = 1, int = 2000 ); // default constructor

void print(); // print function

void setDate( int, int, int ); // set month, day, year

void setMonth( int ); // set month

void setDay( int ); // set day

void setYear( int ); // set year

int getMonth(); // get month

int getDay(); // get day

int getYear(); // get year

void nextDay();

private:

int month; // 1-12

int day; // 1-31 (except February(leap year), April, June, Sept, Nov)

int year; // 1900+

bool leapYear(); // leap year

int monthDays(); // days in month

}; // end class Date

#endif

// Lab 5: Date.h

#ifndef DATE\_H

#define DATE\_H

class Date

{

public:

Date( int = 1, int = 1, int = 2000 ); // default constructor

void print(); // print function

void setDate( int, int, int ); // set month, day, year

void setMonth( int ); // set month

void setDay( int ); // set day

void setYear( int ); // set year

int getMonth(); // get month

int getDay(); // get day

int getYear(); // get year

void nextDay();

private:

int month; // 1-12

int day; // 1-31 (except February(leap year), April, June, Sept, Nov)

int year; // 1900+

bool leapYear(); // leap year

int monthDays(); // days in month

}; // end class Date

#endif

// Lab 5: DateTest.cpp

#include <iostream>

using namespace std;

#include "Date.h" // include definitions of class Date

int main()

{

const int MAXDAYS = 16;

Date d( 12, 24, 2004 ); // instantiate object d of class Date

// output Date object d's value

for ( int loop = 1; loop <= MAXDAYS; ++loop )

{

d.print(); // invokes function print

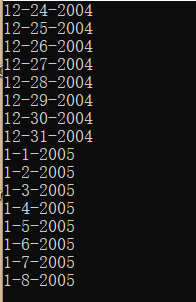
/\* call to nextDay \*/

d.nextDay();

} // end for

cout << endl;

} // end main

**V Follow-Up Questions and Activities**

1. The Date class has only one constructor. Is it possible to have more than one constructor?

可以

1. What happens when a member function that takes no arguments is called without the parentheses (i.e.,dateObject.nextDay)?

系统会提供一个默认值

1. Write a destructor for the Date class. The destructor should print text indicating that the destructor for the Date class was called successfully.

Date::~Date()

{

Cout << “destructor”<<endl;

};

1. In main, try to change d’s year to 2003 using an assignment statement. Do not call function setYear. What happens? Are you able to change the value?

不能

**Lab Exercise 6 — Simple Calculator**

**I Lab Objectives**

In this lab, you will practice:

1. Using classes to create a data type Simple Calculator capable of performing arithmetic operations.
2. Creating const member functions to enforce the principle of least privilege.

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

1. Using constructors to specify initial values for data members of a programmer-defined class.

**II Description of the Problem**

Write a SimpleCalculator class that has public methods for adding, subtracting, multiplying and dividing two doubles. A sample call is as follows:

double answer = sc.add( a, b );

Object sc is of type SimpleCalculator. Member function add returns the result of adding its two arguments.

**III Sample Output**



**IV Your Solution**

// Lab Exercise 6: SimpleCalculator.h

// class SimpleCalculator definition

class SimpleCalculator

{

public:

double add(double,double)const;

double subtract( double, double ) const;

double multiply( double, double ) const;

double divide(double,double)const;

}; // end class SimpleCalculator

// Lab Exercise 6: CalcTest.cpp

#include <iostream>

using namespace std;

#include "SimpleCalculator.h"

int main()

{

double a = 10.0;

double b = 20.0;

/\* Instantiate an object of type Simplecalculator \*/

SimpleCalculator sc;

cout << "The value of a is: " << a << "\n";

cout << "The value of b is: " << b << "\n\n";

double addition = sc.add(a,b);

cout << "Adding a and b yields " << addition << "\n";

double subtraction = sc.subtract( a, b );

cout << "Subtracting b from a yields " << subtraction << "\n";

double multiplication = sc.multiply( a, b );

cout << "Multiplying a by b yields " << multiplication << "\n";

double division = sc.divide(a,b);

cout << "Dividing a by b yields " << division << endl;

}

// Lab Exercise 6: SimpleCalculator.cpp

#include "SimpleCalculator.h"

/\* definition for add member function \*/

double SimpleCalculator::add( double a, double b ) const

{

return a + b;

}

// function subtract definition

double SimpleCalculator::subtract( double a, double b ) const

{

return a - b;

} // end function subtract

// function multiply definition

double SimpleCalculator::multiply( double a, double b ) const

{

return a \* b;

} // end function multiply

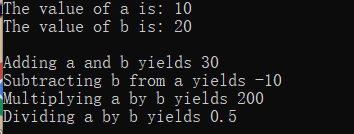
/\* define divide member function \*/

double SimpleCalculator::divide( double a, double b ) const

{

return a / b;

}



**V Follow-Up Questions and Activities**

1. Why doesn’t the SimpleCalculator class have a constructor?

编译器会自动添加默认的构造函数

1. Why are no private data members needed for class SimpleCalculator?

因为计算加减乘除运算时，直接将对象的数据传给了函数，所以不需要私有成员

1. Modify your class so that SimpleCalculator has a private data member called answer. After performing an operation, assign the result to answer. Add a member function named getAnswer to retrieve the result of the last arithmetic operation performed by the object. Also, add a constructor for class SimpleCalculator that initializes the value of answer to 0.

**class** SimpleCalculator

{

public:

SimpleCalculator(answer = 0);

double getAnswer();

private:

double answer;

};

1. Modify the program so that the SimpleCalculator class has an input member function that allows the user to input two doubles. The function should then store the values that were input in private data members. Use these two values for each of the arithmetic calculations. Create two constructors for this class, one that takes no arguments and initializes a and b to 0 and another that takes two doubles and initializes a and b to those values. Finally, create a member function printValues that displays the values of a and b. A segment of the driver program might now look like this:



class SimpleCalculator

{

public:

SimpleCalculator(double an=0.0)

:answer(an)

{

}

/\* Write prototype for add member function \*/

double add( double, double ) const;

double subtract( double, double ) const;

double multiply( double, double ) const;

double divide( double, double ) const;

/\* Write prototype for divide member function \*/

double getAnswer( double) const;

private:

double answer;

}; // end class SimpleCalculator

void SimpleCalculator::printValue(double a,double b)

{

cout <<a<<b;

}

class SimpleCalculator

{

public:

SimpleCalculator(double a,double b)

:a(0),b(0){

}

SimpleCalculator(double ,double);

void printValue(double a,double b);

/\* Write prototype for add member function \*/

double add( double, double ) const;

double subtract( double, double ) const;

double multiply( double, double ) const;

double divide( double, double ) const;

/\* Write prototype for divide member function \*/

private:

double a;

double b;

}; // end class SimpleCalculator

**\*Lab Exercise 7 — Integer Set**

**I Lab Objectives**

In this lab, you will practice:

1. Using classes to create a data type, IntegerSet, capable of storing a set of integers
2. Using dynamic memory allocation with the new and delete operators

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

1. Using destructors to deallocate memory that was dynamically allocated.

**II Description of the Problem**

Create class IntegerSet for which each object can hold integers in the range 0 through 100. A set is represented internally as an array of ones and zeros. Array element a[ i ] is 1 if integer *i* is in the set. Array element a[ j ] is 0 if integer *j* is not in the set. The default constructor initializes a set to the so-called “empty-set,” i.e., a set whose array representation contains all zeros.

Provide member functions for the common set operations. For example, aunionOfSets member function (already provided) creates a third set that is the set-theoretic union of two existing sets (i.e., an element of the third array’s is set to 1 if that element is 1 in either or both of the existing sets, and an element of the third set’s array is set to 0 if that element is 0 in each of the existing sets).

Provide an intersectionOfSetsmember function which creates a third set which is the set-theoretic intersection of two existing sets (i.e., an element of the third set’s array is set to 0 if that element is 0 in either or both of the existing sets, and an element of the third set’s array is set to 1 if that element is 1 in each of the existing sets).

An insertElement member function (already provided) inserts a new integer k into a set (by setting a[ k ] to 1 ). Provide a deleteElement member function that deletes integer m (by setting a[ m ] to 0 ).

A printSet member function (already provided) prints a set as a list of numbers separated by spaces. Print only those elements which are present in the set (i.e., their position in the array has a value of 1 ). Print --- for an empty set.

Provide anisEqualTo member function that determines whether two sets are equal.

Provide an additional constructor that receives an array of integers and the size of that array and uses the array to initialize a set object.

Now write a driver program to test your IntegerSet class. Instantiate several IntegerSet objects. Test that all your member functions work properly.

**III Sample Output**



**IV Your Solution**

**V Follow-Up Questions and Activities**

1. Why might it be advantageous for the set array to be allocated dynamically using new [], if the IntegerSet class were to be used for more general sets?