《C++程序设计实践》实验十三

班级： 学号： 2020212205113 姓名： 王俊琪 得分： 100

一、课内实验题（共5小题，100分）

题型得分 100

【描述】

矩阵是数学里的一种抽象对象，可以用C++提供的静态数组来表示矩阵，其大小在编译时就已经确定，在运行时无法修改，而且不检查下标是否越界。可以利用教材本章提供的向量类Vector，用向量的方式实现矩阵，用一个指针向量来表示矩阵，其中的每个指针又各指向一个向量，用它们来表示矩阵的行向量。矩阵的逻辑结构如下图所示。

组合向量类Vector，声明并实现一个Matrix类模板，表示矩阵。Matrix类模板包括：

Vector类型的私有数据成员rows，存放矩阵元素。

构造函数，将矩阵的行、列大小设置为给定的参数。

构造函数，将矩阵的行、列大小设置为给定的参数，给矩阵元素赋相同的初始值。

重载下标运算符[]。

访问器函数getRows，用于获取矩阵行数。

访问器函数getColumns，用于获取矩阵列数。

Matrix类模板如下：

template <typename T>

class Matrix {

public:

Matrix(int row, int column);

Matrix(int row, int column, const T &value);

Vector<T> &operator[](int index);

Vector<T> operator[](int index) const;

int getRows() const;

int getColumns() const;

private:

Vector<Vector<T>\*> rows; // 存放矩阵元素

};

以普通函数的形式重载\*运算符函数，实现矩阵乘法。

template <typename T>

Matrix<T> operator\*(const Matrix<T> &lhs, const Matrix<T> &rhs);

printMatrix函数输出矩阵的值。

template <typename T>

void printMatrix(const Matrix<T> &m);

【输入】

输入3×3矩阵a和矩阵b。

【输出】

矩阵a乘以矩阵b的结果。每个元素输出宽度为4。

【输入示例】

1 1 7

7 5 6

9 6 1

6 1 6

4 1 5

1 5 1

【输出示例】

17 37 18

68 42 73

79 20 85

【来源】

《程序设计基础——以C++为例》第9章实验1。

（20分）

我的答案：

template <typename T>

class Matrix {

public:

Matrix(int row, int column);

Matrix(int row, int column, const T& value);

Vector<T>& operator[](int index);

Vector<T> operator[](int index) const;

int getRows() const;

int getColumns() const;

private:

Vector<Vector<T>\*> rows; // 存放矩阵元素

};

template <typename T>

Matrix<T>::Matrix(int row, int column) : rows(row) {

for (int i = 0; i < row; i++) {

rows[i] = new Vector<T>(column);

}

}

template <typename T>

Matrix<T>::Matrix(int row, int column, const T& value) :rows(row) {

for (int i = 0; i < row; i++) {

rows[i] = new Vector<T>(column, value);

}

}

template <typename T>

Vector<T>& Matrix<T>::operator[](int index) {

return \*rows[index];

}

template <typename T>

Vector<T> Matrix<T>::operator[](int index) const {

return \*rows[index];

}

template<typename T>

int Matrix<T>::getRows() const{

return rows.getSize();

}

template<typename T>

int Matrix<T>::getColumns() const{

return rows[0]->getSize();

}

template <typename T>

Matrix<T> operator\*(const Matrix<T>& lhs, const Matrix<T>& rhs) {

Matrix<T> res(lhs.getRows(), rhs.getColumns(), 0);

for (int i = 0; i < lhs.getRows(); i++) {

for (int j = 0; j < rhs.getColumns(); j++) {

for (int k = 0; k < lhs.getColumns(); k++) {

res[i][j] += lhs[i][k] \* rhs[k][j];

}

}

}

return res;

}

template <typename T>

void printMatrix(const Matrix<T>& m) {

for (int i = 0; i < m.getRows(); i++) {

for (int j = 0; j < m.getColumns(); j++) {

cout << setw(4) << setfill(' ') << m[i][j];

}

}

}

题目得分 20

【描述】

给定一个链表，链表的每个节点只含有一个int型元素和Node\*指针，该链表共有十个节点，输出链表元素值为奇数的项。部分代码已经写好，请补全output函数即可。

【输入】

输入10个整数。

【输出】

输出奇数值，以空格隔开。

【输入示例】

1 3 4 5 6 7 8 10 11 15

【输出示例】

1 3 5 7 11 15

（20分）

我的答案：

Node \*tmpNode = head;

while(tmpNode){

if(tmpNode->value%2){

cout << tmpNode->value << " ";

}

tmpNode = tmpNode->next;

}

题目得分 20

【描述】

本章教材组合向量类Vector实现了一个Stack类模板。也可以用链表的方式实现栈。利用本章教材提供的链表类LinkedList，组合链表类LinkedList来实现一个新的Stack类模板。

template <typename T>

class Stack {

public:

Stack();

void clear();

bool isEmpty() const;

void push(const T &value);

T pop();

T peek() const;

private:

LinkedList<T> data;

};

【输入】

一系列正整数并入栈，输入-1表示结束，-1不是输入的数据的一部分。

【输出】

输出栈中所有的整数，每个整数后面跟一个空格以与后面的整数区分。

【输入示例】

1 3 5 2 -1

【输出示例】

2 5 3 1

【来源】

《程序设计基础——以C++为例》第9章实验4。

（20分）

我的答案：

template <typename T>

class Stack {

public:

bool isEmpty() const;

void push(const T& value);

T pop();

T peak() const;

private:

LinkedList<T> data;

};

template <typename T>

bool Stack<T>::isEmpty() const {

return data.isEmpty();

}

template <typename T>

void Stack<T>::push(const T& value) {

data.addLast(value);

}

template <typename T>

T Stack<T>::pop() {

T temp = data.getLast();

data.removeLast();

return temp;

}

template <typename T>

T Stack<T>::peak() const {

return data.getLast();

}

题目得分 20

【描述】

栈可以应用于算术表达式求值。这里对算术表达式做如下简化：运算符为+、-、\*、/、%；操作数为单数字（0～9）非负整数。

例如：

(2+5)\*3-8/3

上述形式的算术表达式也称中缀表达式，因为每个运算符出现在它的两个操作数之间。编译器在求算术表达式值时，往往将中缀表达式转换为后缀表达式。后缀表达式也称逆波兰表达式，由波兰数学家Jan Lukasiewicz发明，指运算符出现在操作数后面的不含括号的算术表达式。

中缀表达式：a+b\*c，其后缀表达式为：a b c \* +。因为\*的优先级比+高，所以先计算b\*c，即b c \*；+的操作数是a和b c \*，即a b c \* +。

中缀表达式：(a+b)\*c，其后缀表达式为：a b + c \*。先计算圆括号中a+b，即a b +；\*的操作数是a b +和c，即a b + c \*。

中缀表达式：(a\*b+c)/d+e，其后缀表达式为：a b \* c + d / e +。先计算圆括号中a\*b+c，即a b \* c +；/的操作数是a b \* c +和d，即a b \* c + d /；+的操作数是a b \* c + d /和e，即a b \* c + d / e +。

后缀表达式求值使用一个存放操作数的栈。求值过程顺序扫描后缀表达式，每次遇到操作数就将它入栈；遇到运算符时，就从栈中弹出两个操作数进行计算，并将结果入栈。到扫描结束，留在栈顶的操作数就是所求算术表达式的值。

例如：

4 2 7 \*+

如下图一所示，顺序读取操作数4、2、7，并入栈。如下图二所示，读取运算符\*，操作数7、2出栈，计算2\*7，结果为14并入栈。如下图三所示，读取运算符+，操作数14、4出栈，计算4+14，结果为18并入栈。扫描后缀表达式后，栈中惟一元素即为最终结果。

本章教材组合向量类Vector实现了一个Stack类模板。请组合栈类Stack，声明并实现一个PostfixEvaluation类，求后缀表达式的值。PostfixEvaluation类包括：

string类型的私有数据成员postfixExpression，存放后缀表达式。

Stack类型的私有数据成员operandStack，存放操作数（单数字非负整数）。

私有成员函数getOperands，连续从栈顶出栈两个操作数。

私有成员函数calculate，求出栈的两个操作数的计算结果，并将结果入栈。

私有成员函数isOperator，确定输入的运算符是有效运算符（+、-、\*、/、%）

有参构造函数。新建一个后缀表达式对象。

访问器函数getPostfixExpression，获取后缀表达式。

更改器函数setPostfixExpression，设置新的后缀表达式。

成员函数evaluate，计算并返回后缀表达式的值。

PostfixEvaluation类如下：

class PostfixEvaluation {

public:

PostfixEvaluation(const string&postfixExpression); // 构造函数

stringgetPostfixExpression() const; // 获取后缀表达式

void setPostfixExpression(const string&postfixExpression); // 设置后缀表达式

int evaluate(); // 计算并返回后缀表达式值

private:

string postfixExpression; // 存放要计算的后缀表达式

Stack<int> operandStack; // 存放操作数

void getOperands(int &left, int&right); // 操作数出栈

int calculate(int left, int right, char op) const; // 求操作数运算值

bool isOperator(char ch) const; // 是否是运算符

};

【输入】

输入一个后缀表达式。

【输出】

输出后缀表达式的值。

【输入示例1】

4 2 7 \* +

【输出示例1】

18

【输入示例2】

8 4 0 % +

【输出示例2】

Divisor cannot be zero!

【输入示例3】

2 3 ^ 4 +

【输出示例3】

Illegal input!

【输入示例4】

4 2 / \*

【输出示例4】

Too many operators!

【输入示例5】

1 2 3 +

【输出示例5】

Too many operands!

【来源】

《程序设计基础——以C++为例》第9章实验2。

（20分）

我的答案：

class PostfixEvaluation {

public:

PostfixEvaluation(const string& postfixExpression); // 构造函数

string getPostfixExpression() const; // 获取后缀表达式

void setPostfixExpression(const string& postfixExpression); // 设置后缀表达式

int evaluate(); // 计算并返回后缀表达式值

private:

string postfixExpression; // 存放要计算的后缀表达式

Stack<int> operandStack; // 存放操作数

void getOperands(int& left, int& right); // 操作数出栈

int calculate(int left, int right, char op) const; // 求操作数运算值

bool isOperator(char ch) const; // 是否是运算符

};

void PostfixEvaluation::getOperands(int& left, int& right) {

right = operandStack.pop();

if (operandStack.isEmpty()) {

throw runtime\_error("Too many operators!");

}

left = operandStack.pop();

}

int PostfixEvaluation::calculate(int left, int right, char op) const {

int value;

if (op == '+')

value = left + right;

else if (op == '-')

value = left - right;

else if (op == '\*')

value = left \* right;

else if (op == '/')

value = left / right;

else if (op == '%')

value = left % right;

else

value = INT\_MIN;

return value;

}

bool PostfixEvaluation::isOperator(char ch) const {

return ch == '+' || ch == '-' || ch == '\*' || ch == '/' || ch == '%';

}

PostfixEvaluation::PostfixEvaluation(const string& postfixExpression) {

this->postfixExpression = postfixExpression;

}

string PostfixEvaluation::getPostfixExpression() const {

return postfixExpression;

}

void PostfixEvaluation::setPostfixExpression(const string& postfixExpression) {

this->postfixExpression = postfixExpression;

}

int PostfixEvaluation::evaluate() {

char op;

int left, right, value;

int flag = 0;

for (int i = 0; i < postfixExpression.length(); i++) {

if (postfixExpression[i]=='^') {

throw runtime\_error("Illegal input!");

}

}

for (int i = 0; i < postfixExpression.length(); i++) {

if (!isOperator(postfixExpression[i]) && postfixExpression[i] != ' ') {

operandStack.push(postfixExpression[i] - '0');

}

else if (isOperator(postfixExpression[i])) {

op = postfixExpression[i];

getOperands(left, right);

if (right == 0 && (op == '%' || op == '/')) {

throw runtime\_error("Divisor cannot be zero!");

}

value = calculate(left, right, op);

operandStack.push(value);

}

}

operandStack.pop();

if (!operandStack.isEmpty()) {

throw runtime\_error("Too many operands!");

}

return value;

}

题目得分 20

【描述】

编写程序，输入若干个正整数，输入-1时输入结束，可以简化修改本章教材提供的链表类LinkedList，用单向链表组织输入的正整数。要求链表按照结点中整数值的大小从大到小排序，不包括最后标识结束的-1。输出单向链表。

【输入】

一系列正整数，输入-1表示结束，-1不是输入的数据的一部分。

【输出】

按照结点中整数值的大小从大到小输出所有的整数，每个整数后面跟一个空格以与后面的整数区分，最后的整数后面没有空格。

【输入示例】

1 3 5 2 -1

【输出示例】

5 3 2 1

【来源】

《程序设计基础——以C++为例》第9章实验3。

（20分）

我的答案：

template <typename T>

class Node {

public:

Node() {

next = NULL;

}

Node(const T& value) {

this->value = value;

next = NULL;

}

T value;

Node<T>\* next;

};

template <typename T>

class LinkedList {

public:

LinkedList();

void addFirst(const T& value);

void addLast(const T& value);

void add(const T& value);

void removeFirst();

void removeLast();

void removeAt(int index);

T getFirst() const;

T getLast() const;

void clear();

bool isEmpty() const;

int getSize() const;

void print() const;

private:

Node<T>\* head, \* tail;

int size;

};

template <typename T>

LinkedList<T>::LinkedList() {

head = tail = NULL;

size = 0;

}

template <typename T>

void LinkedList<T>::addFirst(const T& value) {

Node<T>\* temp = new Node<T>(value);

temp->next = head;

head = temp;

++size;

if (tail == NULL)

tail = head;

}

template <typename T>

void LinkedList<T>::addLast(const T& value) {

if (tail == NULL)

head = tail = new Node<T>(value);

else {

tail->next = new Node<T>(value);

tail = tail->next;

}

++size;

}

template <typename T>

void LinkedList<T>::removeFirst() {

if (head == NULL)

throw runtime\_error("空链表!");

else {

Node<T>\* temp = head;

head = head->next;

if (head == NULL)

tail = NULL;

delete temp;

--size;

}

}

template <typename T>

void LinkedList<T>::removeLast() {

if (tail == NULL)

throw runtime\_error("空链表!");

else if (head == tail) {

Node<T>\* temp = head;

head = tail = NULL;

size = 0;

delete temp;

}

else {

Node<T>\* current = head;

while (current->next != tail)

current = current->next;

Node<T>\* temp = tail;

tail = current;

tail->next = NULL;

delete temp;

--size;

}

}

template <typename T>

void LinkedList<T>::removeAt(int index) {

if (index < 0 || index >= size)

throw runtime\_error("下标越界!");

else if (index == 0)

return removeFirst();

else if (index == size - 1)

return removeLast();

else {

Node<T>\* previous = head;

for (int i = 1; i < index; ++i)

previous = previous->next;

Node<T>\* current = previous->next;

previous->next = current->next;

delete current;

--size;

}

}

template <typename T>

T LinkedList<T>::getFirst() const {

if (head == NULL)

throw runtime\_error("空链表!");

else

return head->value;

}

template <typename T>

T LinkedList<T>::getLast() const {

if (tail == NULL)

throw runtime\_error("空链表!");

else

return tail->value;

}

template <typename T>

void LinkedList<T>::clear() {

while (head != NULL) {

Node<T>\* temp = head;

head = head->next;

delete temp;

}

tail = NULL;

}

template <typename T>

bool LinkedList<T>::isEmpty() const {

return head == NULL;

}

template <typename T>

int LinkedList<T>::getSize() const {

return size;

}

template <typename T>

void LinkedList<T>::print() const {

Node<T>\* current = head;

while (current != NULL) {

cout << current->value << " ";

current = current->next;

}

cout << endl;

}

template <typename T>

void LinkedList<T>::add(const T& value) {

Node<T>\* temp = new Node<T>(value);

Node<T>\* current = head;

Node<T>\* previous = NULL;

while (current != NULL && current->value > value) {

previous = current;

current = current->next;

}

if (current == head) {

head = temp;

}

else {

previous->next = temp;

}

temp->next = current;

}

题目得分 20

《C++程序设计实践》实验十二

班级： 学号： 2020212205113 姓名： 王俊琪 得分： 50

一、课内实验题（共5小题，50分）

题型得分 50

【描述】

声明并实现一个Student类，表示学生信息。Student类包括：

int类型的私有数据成员num，表示学号。

string类型的私有数据成员name，表示姓名。

int类型的私有数据成员score，表示成绩

char类型的私有数据成员grade，表示等级。

无参（默认）构造函数。

有参构造函数，将学号、姓名、成绩和等级设置为给定的参数。

访问器函数getNum、getName、getScore、getGrade，分别用于访问学号、姓名、成绩和等级。

重载流提取运算符>>和流插入运算符<<。输入输出一个Student对象

输入若干个学生的信息（学号、姓名和成绩），学号为0时，输入结束，根据成绩计算出对应等级。假设90分以上的成绩属于A级；80～89分、70～79分、60～69分的成绩分别属于B、C、D级；60分以下属于E级。创建Student对象，将它们写入文本文件student.txt中。

【输入】

输入若干个学生的信息。

每行一个学生信息，学号、姓名和成绩之间以空格间隔。

学号为0时，输入结束。

【输出】

文件student.txt。

不需要在屏幕上显示信息。

【输入示例】

100101 ZhangSan 78

100102 LiSi 67

100103 WangWu 83

100104 LiuLiu 45

100105 QianQi 93

0

【输出示例】

生成文件student.txt，其中内容：

100101 ZhangSan 78 C

100102 LiSi 67 D

100103 WangWu 83 B

100104 LiuLiu 45 E

100105 QianQi 93 A

【来源】

《程序设计基础——以C++为例》第8章实验3。（10分）

我的答案：

#include<iostream>

#include<fstream>

using namespace std;

class Student {

public:

Student(){

};

Student(int number, string n, int s, char g) {

num = number;

name = n;

score = s;

grade = g;

};

int getNum() { return num; };

string getName() { return name; };

int getScore() { return score; };

char getGrade() { return grade; };

friend istream& operator>>(istream& cin, Student& stu);

friend ostream& operator<<(ostream& cout, Student& stu);

private:

int num;

string name;

int score;

char grade;

};

istream& operator>>(istream& cin, Student& stu) {

int num;

string name;

int score;

char grade;

cin >> num >> name >> score;

switch (score/10)

{

case 10:

case 9:

grade = 'A';

break;

case 8:

grade = 'B';

break;

case 7:

grade = 'C';

break;

case 6:

grade = 'D';

break;

default:

grade = 'E';

break;

}

Student s(num, name, score, grade);

stu = s;

return cin;

}

ostream& operator<<(ostream& cout, Student& stu) {

cout << stu.num << " " << stu.name << " " << stu.score << " " << stu.grade;

return cout;

}

int main() {

fstream f;

f.open("student.txt", ios::app);

Student stu;

while (cin >> stu) {

f << stu.getNum() << " " << stu.getName() << " " << stu.getScore() << " " << stu.getGrade() << endl;

}

f.close();

}

题目得分 10

【描述】

声明并实现了一个Rectangle类，表示矩形。Rectangle类包括：

double类型的私有数据成员width和height，表示矩形的宽和高。

带默认参数的构造函数，将矩形的宽和高设置为给定的参数。宽和高的默认参数值为1。

更改器函数setWidth和setHeight，分别用于修改矩形的宽和高。

访问器函数getWidth和getHeight，分别用于访问矩形的宽和高。

成员函数computeArea，返回矩形的面积。

成员函数computePerimeter，返回矩形的周长。

创建5个Rectangle对象（每个Rectangle对象的宽和高分别为1、2、3、4、5）并将它们写入二进制文件object.dat中。修改第3个对象的宽为10、高为3.5，再把修改后的该对象写回文件原位置。

【输入】

没有输入。

【输出】

生成文件object.dat

不需要在屏幕上显示信息。

【来源】

《程序设计基础——以C++为例》第8章实验4。（10分）

我的答案：

#include<iostream>

#include<fstream>

#include <iomanip>

using namespace std;

class Rectangle {

public:

Rectangle(double w = 1.0, double h = 1.0) {

width = w;

height = h;

};

void setWidth(double w) { width = w; };

void setHeight(double h) { height = h; };

double getWidth() { return width; };

double getHeight() { return height; };

double computeArea() { return width \* height; };

double computePerimeter() { return (width + height) \* 2; };

private:

double width;

double height;

};

int main() {

fstream f, f1;

f.open("object.dat", ios::out|ios::binary);

Rectangle R1(1, 1), R2(2, 2), R3(3, 3), R4(4, 4), R5(5, 5);

double buf[] = { 1, 1, 2, 2, 3, 3, 4, 4, 5, 5 };

f.write(reinterpret\_cast<char\*>(buf), sizeof(buf));

f.close();

f.open("object.dat", ios::out | ios::binary);

buf[4] = 10;

buf[5] = 3.5;

f.write(reinterpret\_cast<char\*>(buf), sizeof(buf));

f.close();

}

题目得分 10

【描述】

给定文件hours.txt，其中包含每个员工工作时间的记录。每一行表示一周的工作时间。每周有7天，所以每行最多有7个数字。规定每周从周一开始，文件中的每一行都是从周一的工作小时数开始，后面是周二，等等，周日的数字放在这一行的最后。每行中的数字可以少于7个，因为员工并不是每天都要工作。下面是文件hours.txt的内容：

8 8 8 8 8

8 4 8 4 8 4 4

8 4 8 4 8

3 0 0 8 6 4 4

8 8

0 0 8 8

8 8 4 8 4

编写一个程序从输入文件中读取数据，计算并报告每行和每列的总和。每行的总和表示该员工每周工作的小时数。每列的总和表示员工周一、周二等每天工作的累计小时数。最后输出总的小时数。针对上述文件hours.txt的输出结果见【输出示例】。

【输入】

文件hours.txt。（该文件已经存在，无需自己创建）

注意：本地调试程序时，则要求自己预先建立好hours.txt文件。在Windows下，可以使用记事本。

【输出】

员工每周工作的小时数。

员工周一、周二等每天工作的累计小时数。

最后输出总的小时数。

【输入示例】

文件hours.txt。

【输出示例】

Total hours = 40

Total hours = 40

Total hours = 32

Total hours = 25

Total hours = 16

Total hours = 16

Total hours = 32

Mon hours = 43

Tue hours = 32

Wed hours = 36

Thu hours = 40

Fri hours = 34

Sat hours = 8

Sun hours = 8

Total hours = 201

【提示】

文件hours.txt中数据组数是不定的。

（10分）

我的答案：

#include<iostream>

#include<fstream>

#include<sstream>

#include<string>

#include<vector>

using namespace std;

int main(){

string buf;

int time;

int total=0;

vector<int> vec(7,0);

vector< vector<int>> day(1,vector<int>(7,0));

string days[7]={"Mon","Tue","Wed","Thu","Fri","Sat","Sun"};

ifstream f;

f.open("hours.txt");

istringstream iss;

int i=0, j=0,k=0;

while (getline(f, buf)) {

j=0;

iss.clear();

iss.str(buf);

total=0;

while(iss>>time){

day[i][j]=time;

j++;

}

i++;

day.push\_back(vec);

}

for(k=0;k<i;k++){

total=0;

for(j=0;j<7;j++){

total+=day[k][j];

}

cout << "Total hours = " << total << endl;

}

cout << endl;

total=0;

int temp=0;

for(j=0;j<7;j++){

temp=0;

for(k=0;k<i;k++){

temp+=day[k][j];

}

total+=temp;

cout << days[j] << " hours = " << temp << endl;

}

cout << "Total hours = " << total << endl;

f.close();

return 0;

}

题目得分 10

【描述】

处理日志文件，日志文件的储存格式为“年/月/日 时:分:秒 用户名 操作”。

日志文件有多条记录：

2015/4/218:00:33 37c3b6b58c6ac3 LOGIN

2015/4/218:15:35 11734e186f24fe4c LOGIN

2015/4/218:34:57 9f3cf331d19a9f LOGIN

2015/4/219:00:29 389bcca2159f5de7 LOGIN

2015/4/219:08:29 c3bde693fdb3c3d LOGIN

……

可以下载日志文件：

鼠标右键另存为

【输入】

日志文件log.txt。（该文件已经存在，无需自己创建）

【输出】

日志文件中活跃用户的数量。

【输入示例】

无

【输出示例】

123

【提示】

活跃用户指的是在日志文件中有过操作的用户，记得把重复出现的用户去掉。

输出示例只是格式说明，并非正确答案。

（10分）

我的答案：

#include<iostream>

#include<fstream>

#include<string>

#include <iomanip>

#include<vector>

using namespace std;

int main() {

fstream f;

f.open("log.txt",ios::in);

int i,sum=0;

string s1,s2,s3,s4,s5;

vector<string> vec;

vec.push\_back("");

while(f>>s1>>s2>>s3>>s4){

for(i = 0; i < vec.size(); i++){

if (s3 == vec[i]) break;

}

if(i == vec.size()){

vec.push\_back(s3);

sum++;

}

}

cout << sum;

f.close();

}

题目得分 10

【描述】

将一个明文文件plaintext.txt中的内容，按照一定的方法，对每个字符加密后存放到另一个密文文件ciphertext.txt中。

【输入】

文件plaintext.txt。（该文件已经存在，无需自己创建）

注意：本地调试程序时，则要求自己预先建立好plaintext.txt文件。在Windows下，可以使用记事本。

【输出】

生成文件ciphertext.txt，里面存放加密后的信息。

不需要在屏幕上显示信息。

【输入示例】

文件plaintext.txt，假定其中内容如下：

Welcome to C++!

【输出示例】

生成文件ciphertext.txt，加密后的内容如下：

Ygneqog"vq"E--#

【提示】

这里采用一种简单的加密方法，将每个字符的编码加2。

【来源】

《程序设计基础——以C++为例》第8章实验2。（10分）

我的答案：

#include<iostream>

#include<fstream>

#include<string>

#include <iomanip>

using namespace std;

int main() {

fstream fin, fout;

fin.open("plaintext.txt", ios::in);

fin.seekg(0, ios::beg);

string buf;

while (getline(fin, buf)) {

cout << buf << endl;

int size = buf.length();

for (int i = 0; i < size; i++) {

int ascii = (int)buf[i];

buf[i] = (char)(ascii + 2);

}

fout.open("ciphertext.txt", ios::out|ios::app);

fout << buf;

fout.close();

}

fin.close();

}

题目得分 10

《C++程序设计实践》实验十一

班级： 学号： 2020212205113 姓名： 王俊琪 得分： 100

一、课内实验题（共10小题，100分）

题型得分 100

【描述】

①声明并实现一个名为Arc的类，在二维空间中以某一个点为中心，绘制圆弧。Arc类包含私有数据成员p（圆心，Point类型），radius（圆弧半径，double类型）；有参构造函数，将圆心、圆弧半径设置为给定的参数；成员函数draw，输出圆心和半径。

②声明并实现一个名为Circle的类，在二维空间中以某一个点为中心，绘制圆。Circle类包含私有数据成员p（圆心，Point类型），radius（圆半径，double类型）；有参构造函数，将圆心、圆半径设置为给定的参数；成员函数draw，输出圆心和半径。

③声明并实现一个名为Ellipse的类，在二维空间中以某一个点为中心，绘制椭圆。Ellipse类包含私有数据成员p（圆心，Point类型），xRadius、yRadius（椭圆轴，double类型）；有参构造函数，将圆心、椭圆轴设置为给定的参数；成员函数draw，输出圆心和轴。

④声明并实现一个名为Rectangle的类，在二维空间中以某一个点为左上角，绘制矩形。Rectangle类包含私有数据成员p（左上角坐标，Point类型），width、height（矩形宽度和高度，double类型）；有参构造函数，将左上角坐标、矩形宽度和高度设置为给定的参数；成员函数draw，输出左上角坐标和宽度、高度。

⑤声明并实现一个名为Mix的类，在二维空间中以某一个点为中心，绘制圆弧、圆、椭圆，以某一个点为左上角，绘制矩形。Mix类包含有参构造函数，将点坐标、圆弧半径、圆半径、椭圆轴、矩形宽度和高度设置为给定的参数；成员函数draw绘制圆弧、圆、椭圆和矩形，调用Arc类、Circle类、Ellipse类、Rectangle类的draw函数，输出相关信息。

Mix类是Arc类、Circle类、Ellipse类、Rectangle类的多继承派生类。

【输入】

没有输入。

【输出】

Drawing an arc: Center(320, 240), radius(100)

Drawing a circle: Center(320, 240), radius(70)

Drawing an ellipse: Center(320, 240), x-axis(100), y-axis(70)

Drawing a rectangle: Upper left corner coordinates(320, 240), width(100), height(70)

【提示】

需要自定义Point类。

给定如下main函数：

int main() {

Point p(320, 240);

Mix mix(p, 100, 70);

mix.draw();

return 0;

}

【来源】

《程序设计基础——以C++为例》第7章实验5。（10分）

我的答案：

class Point{

public:

Point(double X=0, double Y=0){

x = X;

y = Y;

};

double getX(){

return x;

}

double getY(){

return y;

}

private:

double x;

double y;

};

class Arc

{

public:

Arc(){};

Arc(Point point, double r){

p = point;

radius = r;

};

void draw();

private:

Point p;

double radius;

};

void Arc::draw()

{

cout << "Drawing an arc: " << "Center(" << p.getX() << ", " << p.getY() << "), ";

cout << "radius(" << radius << ")" << endl;

}

class Circle

{

public:

Circle(){};

Circle(Point point, double r){

p = point;

radius = r;

};

void draw();

private:

Point p;

double radius;

};

void Circle::draw()

{

cout << "Drawing an circle: " << "Center(" << p.getX() << ", " << p.getY() << "), ";

cout << "radius(" << radius << ")" << endl;

}

class Ellipse

{

public:

Ellipse(){};

Ellipse(Point point, double a, double b){

p = point;

xRadius = a;

yRadius = b;

};

void draw();

private:

Point p;

double xRadius;

double yRadius;

};

void Ellipse::draw()

{

cout << "Drawing an circle: " << "Center(" << p.getX() << ", " << p.getY() << "), ";

cout << "x-axis(" << xRadius << "), " << "y-axis(" << yRadius << ")" << endl;

}

class Rectangle

{

public:

Rectangle(){};

Rectangle(Point point, double w, double h){

p = point;

width = w;

height = h;

};

void draw();

private:

Point p;

double width;

double height;

};

void Rectangle::draw()

{

cout << "Drawing an rectangle: " << "Upper left corner coordinates(" << p.getX() << ", " << p.getY() << "), ";

cout << "width(" << width << "), " << "height(" << height << ")" << endl;

}

class Mix : public Arc, public Circle, public Ellipse, public Rectangle

{

public:

Mix(Point point, double a, double b){

A = Arc(point, a);

C = Circle(point, b);

E = Ellipse(point, a, b);

R = Rectangle(point, a, b);

};

void draw();

private:

Arc A;

Circle C;

Ellipse E;

Rectangle R;

};

void Mix::draw()

{

A.draw();

C.draw();

E.draw();

R.draw();

}

题目得分 10

【描述】

下面是不完整的类定义:

class A {

public:

virtual void print(){

cout << "print come form class A" << endl;

}

};

class B : public A {

private:

char \*buf;

public:

void print() {

cout << "print come from class B" << endl;

}

};

void fun(A \*a) {

delete a;

}

试完成其定义（需要增加必要的构造函数、析构函数）。

类A析构函数输出：A::~A() called

类B析构函数输出：B::~B() called

给定如下main函数：

int main() {

A \*a = new B(10);

a->print();

fun(a);

B \*b = new B(20);

fun(b);

return 0;

}

【输入】

没有输入。

【输出】

主函数的输出已经写好。（10分）

我的答案：

class A {

public:

A(){};

virtual ~A(){cout << "A::~A() called" << endl;}

virtual void print(){

cout << "print come form class A" << endl;

}

};

class B : public A {

private:

char \*buf;

public:

B(int a){

buf = (char\*)a;

};

~B(){cout << "B::~B() called" << endl;}

void print() {

cout << "print come from class B" << endl;

}

};

题目得分 10

【描述】

①声明并实现一个名为Person的基类，Person类有保护数据成员name（姓名，string类型）、sex（性别，char类型，'M'表示男，'F'表示女）。以及有参构造函数，将姓名、性别设置为给定的参数；成员函数print，输出姓名和性别。

②从Person类派生出Student类，Student类有私有数据成员status（状态，枚举类型），表示年级（FRESHMAN、SOPHOMORE、JUNIOR、SENIOR），表示大一、大二、大三、大四学生。以及有参构造函数，将姓名、性别、状态设置为给定的参数；成员函数print，print函数输出姓名、性别和状态。

③定义MyDate类，它包含私有数据成员year、month和day以及带默认参数的有参构造函数，年、月、日的默认参数值分别为1900、1、1；成员函数print，输出年、月、日。

④从Person类派生出Employee类，Employee类有保护数据成员salary（薪水，int类型）、dateHired（雇佣日期），dataHired的类型是MyDate。以及有参构造函数，将姓名、性别、薪水和雇佣日期设置为给定的参数；成员函数print，输出姓名、性别、薪水和雇佣日期。

⑤从Employee类派生出Faculty类，Faculty类有私有数据成员rank（级别，枚举类型），有（PROFESSOR、ASSOCIATE\_PROFESSOR、LECTURER），表示教授、副教授、讲师。以及有参构造函数，将姓名、性别、薪水、雇佣日期和级别设置为给定的参数；成员函数print，输出姓名、性别、薪水、雇佣日期和级别。

⑥从Employee类派生出Staff类，Staff类有私有数据成员headship（职务，枚举类型），有（PRESIDENT、DEAN、DEPARTMENT\_CHAIRMAN），表示校长、院长、系主任。以及有参构造函数，将姓名、性别、薪水、雇佣日期和职务设置为给定的参数；成员函数print，输出姓名、性别、薪水、雇佣日期和职务。

【输入】

没有输入。

【输出】

Name:ZhangSan, Sex:M

Name:LiSi, Sex:F

Status:Freshman

Name:WangWu, Sex:M

Salary:5000, Hire date:2012-3-1

Name:LiuLiu, Sex:M

Salary:10000, Hire date:2012-3-1

Rank:Professor

Name:QianQi, Sex:M

Salary:8000, Hire date:2012-3-1

Headship:Department chairman

【来源】

《程序设计基础——以C++为例》第7章实验4。

（10分）

我的答案：

class Person

{

public:

Person(){};

Person(string personName, char gender)

{

name = personName;

sex = gender;

}

string getName();

char getSex();

void print();

void setName(string personName);

void setSex(char gender);

protected:

string name;

char sex;

};

string Person::getName()

{

return name;

}

char Person::getSex()

{

return sex;

}

void Person::print()

{

cout << "Name:" << name << ", Sex:" << sex << endl;

}

void Person::setName( string personName )

{

name = personName;

}

void Person::setSex( char gender )

{

sex = gender;

}

enum Grade{FRESHMAN,SOPHOMORE,JUNIOR,SENIOR};

class Student : public Person

{

public:

Student(string personName, char gender, Grade grade){

status = grade;

setName(personName);

setSex(gender);

};

void print();

private:

Grade status;

};

void Student::print()

{

cout << "Name:" << getName() << ", Sex:" << getSex() << endl;

switch (status)

{

case FRESHMAN:

cout << "Status:Freshman" << endl;

break;

case SOPHOMORE:

cout << "Status:Sophomore" << endl;

break;

case JUNIOR:

cout << "Status:Junior" << endl;

break;

case SENIOR:

cout << "Status:Senior" << endl;

break;

default:

break;

}

}

class MyDate

{

public:

MyDate(int y = 1900, int m = 1, int d = 1){

year = y;

month = m;

day = d;

};

void print();

private:

int year;

int month;

int day;

};

void MyDate::print()

{

cout << year << "-" << month << "-" << day << endl;

}

class Employee : public Person

{

public:

Employee(){};

Employee(string personName, char gender, int money, MyDate date){

setName(personName);

setSex(gender);

salary = money;

dateHired = date;

};

void print();

protected:

int salary;

MyDate dateHired;

};

void Employee::print()

{

cout << "Name:" << getName() << ", Sex:" << getSex() << endl;

cout << "Salary:" << salary << ", Hire date:";

dateHired.print();

}

enum Level{PROFESSOR,ASSOCIATE\_PROFESSOR,LECTURER};

class Faculty : public Employee

{

public:

Faculty(string personName, char gender, int money, MyDate date, Level level){

setName(personName);

setSex(gender);

salary = money;

dateHired = date;

rank = level;

};

void print();

private:

Level rank;

};

void Faculty::print()

{

cout << "Name:" << getName() << ", Sex:" << getSex() << endl;

cout << "Salary:" << salary << ", Hire date:";

dateHired.print();

switch (rank)

{

case PROFESSOR:

cout << "Rank:Professor" << endl;

break;

case ASSOCIATE\_PROFESSOR:

cout << "Rank:Associate professor" << endl;

break;

case LECTURER:

cout << "Rank:Lecturer" << endl;

break;

default:

break;

}

}

enum Job{PRESIDENT,DEAN,DEPARTMENT\_CHAIRMAN};

class Staff : public Employee

{

public:

Staff(string personName, char gender, int money, MyDate date, Job post){

setName(personName);

setSex(gender);

salary = money;

dateHired = date;

headship = post;

};

void print();

private:

Job headship;

};

void Staff::print()

{

cout << "Name:" << getName() << ", Sex:" << getSex() << endl;

cout << "Salary:" << salary << ", Hire date:";

dateHired.print();

switch (headship)

{

case PRESIDENT:

cout << "Headship:President" << endl;

break;

case DEAN:

cout << "Headship:Dean" << endl;

break;

case DEPARTMENT\_CHAIRMAN:

cout << "Headship:Department chairman" << endl;

break;

default:

break;

}

}

题目得分 10

【描述】

①Employee类是抽象类。Employee类的派生类有Boss类、CommissionWorker类、PieceWorker类和HourlyWorker类。

②Employee类包括私有数据成员name（姓名，string类型）；有参构造函数，将name设置为给定的参数；访问器函数getName；还有纯虚函数show和earnings。show和earning函数将在每个派生类中实现，因为每个派生类显示的信息不同、计算工资的方法不同。

③Boss类有固定的周工资且不计工作时间。Boss类包括私有数据成员weeklySalary（周工资，double类型）；有参构造函数，将name、weeklySalary设置为给定的参数；更改器函数setWeeklySalary；show函数和earnings函数。

④CommissionWorker类有工资加上销售提成。CommissionWorker类包括私有数据成员salary（工资，double类型）、commission（佣金，double类型）和quantity（销售数量，int类型）；有参构造函数，将name、salary、commission、quantity设置为给定的参数；更改器函数setSalary、setCommission和setQuantity；show函数和earnings函数。

⑤PieceWorker类的工资根据其生产的产品数量而定。PieceWorker类包括私有数据成员wagePerPiece（每件产品工资，double类型）、quantity（生产数量，int类型）；有参构造函数，将name、wagePerPiece、quantity设置为给定的参数；更改器函数setWage、setQuantity；show函数和earnings函数。

⑥HourlyWorker类的工资根据小时计算并有加班工资。HourlyWorker类包括私有数据成员wage（小时工资，double类型）、hours（工作时数，double类型）；有参构造函数，将name、wage、hours设置为给定的参数；更改器函数setWage、setHours；show函数和earnings函数。

动态绑定show和earnings函数显示和计算各类人员的信息和工资。

【输入】

输入Boss的姓名和周工资。

输入CommissonWorker的姓名、工资、佣金和销售数量。

输入PieceWorker的姓名、每件产品工资和生产数量。

输入HourlyWorker的姓名、小时工资、工作时数。

【输出】

见【输出示例】

【输入示例】

ZhangSan 800.0

LiSi 400.0 3.0 150

WangWu 2.5 200

LiuLiu 13.75 40

【输出示例】

Boss: ZhangSan

Earned: $800

Commission Worker: LiSi

Earned: $850

Piece Worker: WangWu

Earned: $500

Hourly Worker: LiuLiu

Earned: $550

【来源】

《程序设计基础——以C++为例》第7章实验7。

（10分）

我的答案：

class Employee

{

public:

Employee(){

};

Employee(string n){

name = n;

};

string getName();

void setName(string n);

virtual void show(){};

virtual int earnings(){return 0;};

private:

string name;

};

string Employee::getName()

{

return name;

}

void Employee::setName(string n)

{

name = n;

}

class Boss : public Employee

{

public:

Boss(string n, double s){

setName(n);

weeklySalary = s;

};

void show();

int earnings();

private:

double weeklySalary;

};

void Boss::show(){

cout << "Boss: " << getName() << endl;

}

int Boss::earnings(){

return (int)weeklySalary;

}

class CommissionWorker : public Employee

{

public:

CommissionWorker(string n, double s, double c, int q){

setName(n);

salary = s;

commission = c;

quantity = q;

};

void show();

int earnings();

void setSalary(double s);

void setCommission(double c);

void setQuantity(int q);

private:

double salary;

double commission;

int quantity;

};

void CommissionWorker::show()

{

cout << "Commission Worker: " << getName() << endl;

}

int CommissionWorker::earnings()

{

return (int)(salary + commission \* quantity);

}

void CommissionWorker::setSalary( double s )

{

salary = s;

}

void CommissionWorker::setCommission( double c )

{

commission = c;

}

void CommissionWorker::setQuantity( int q )

{

quantity = q;

}

class PieceWorker : public Employee

{

public:

PieceWorker(string n, double wage, int q){

setName(n);

wagePerPiece = wage;

quantity = q;

};

void show();

int earnings();

void setWage(double wage);

void setQuantity(int q);

private:

double wagePerPiece;

int quantity;

};

void PieceWorker::show()

{

cout << "Piece Worker: " << getName() << endl;

}

int PieceWorker::earnings()

{

return (int)(wagePerPiece \* quantity);

}

void PieceWorker::setWage( double wage )

{

wagePerPiece = wage;

}

void PieceWorker::setQuantity( int q )

{

quantity = q;

}

class HourlyWorker : public Employee

{

public:

HourlyWorker(string n, double w, double h){

setName(n);

wage = w;

hours = h;

};

void show();

int earnings();

void setWage(double w);

void setHours(double h);

private:

double wage;

double hours;

};

void HourlyWorker::show()

{

cout << "Hourly Worker: " << getName() << endl;

}

int HourlyWorker::earnings()

{

return (int)(wage \* hours);

}

void HourlyWorker::setWage( double w )

{

wage = w;

}

void HourlyWorker::setHours( double h )

{

hours = h;

}

题目得分 10

【描述】

有5个类A、B、C、D、E。它们的形式均为：

class T {

public:

T() {

cout << "T()" << endl;

}

~T() {

cout << "~T()" << endl;

}

};

这里T代表类名A、B、C、D、E。

给定如下main函数：

int main() {

E e;

return 0;

}

要求根据输出结果，声明并实现类A、B、C、D、E，确定类A、B、C、D、E之间的继承关系。

【输入】

没有输入。

【输出】

C()

B()

C()

A()

D()

E()

~E()

~D()

~A()

~C()

~B()

~C()

（10分）

我的答案：

class C {

public:

C() {

cout << "C()" << endl;

};

~C() {

cout << "~C()" << endl;

};

};

class A : public C{

public:

A() {

cout << "A()" << endl;

};

~A() {

cout << "~A()" << endl;

};

};

class B : public C{

public:

B() {

cout << "B()" << endl;

};

~B() {

cout << "~B()" << endl;

};

};

class D : public A{

public:

D() {

cout << "D()" << endl;

};

~D() {

cout << "~D()" << endl;

};

};

class E : public B, public D{

public:

E() {

cout << "E()" << endl;

};

~E() {

cout << "~E()" << endl;

};

};

题目得分 10

【描述】

①Shape类是抽象类，包括了纯虚函数getArea（求面积）、getPerimeter（求周长）、show（输出对象信息）以及成员函数getClassName（返回类名“Shape”）。

②Circle类继承了Shape类，包括了double类型的私有数据成员radius，表示圆半径；带默认参数的有参构造函数，radius的默认参数值为1；访问器函数getRadius和更改器函数setRadius；重定义了getArea（求圆面积）、getPerimeter（求圆周长）、show（输出半径）、getClassName（返回类名“Circle”）函数。

③Rectangle继承了Shape类，包括了double类型的私有数据成员width、height，分别表示矩形的宽度和高度；带默认参数的有参构造函数，width和height的默认参数值分别为1、1；访问器函数getWidth、getHeight和更改器函数setWidth、setHeight；重定义了getArea（求矩形面积）、getPerimeter（求矩形周长）、show（输出宽度和高度）、getClassName（返回类名“Rectangle”）函数。

④Triangle类继承了Shape类，包括了double类型的私有数据成员side1、side2和side3，表示三角形三条边；有参构造函数，将三角形三条边设置为给定的参数；重定义了getArea（求三角形面积）、getPerimeter（求三角形周长）、show（输出三条边）、getClassName（返回类名“Triangle”）函数。

假设PI为3.14159。

【输入】

输入圆的半径、矩形的宽度和高度以及三角形的三条边。

【输出】

见【输出示例】

【输入示例】

3.5

5.8 11.8

1 1.5 1

【输出示例】

Circle:

Radius:3.5

Area:38.4845, Perimeter:21.9911

Rectangle:

Width:5.8, Height:11.8

Area:68.44, Perimeter:35.2

Triangle:

Side:1, 1.5, 1

Area:0.496078, Perimeter:3.5

【来源】

《程序设计基础——以C++为例》第7章实验6。

（10分）

我的答案：

class Shape

{

public:

Shape(){};

virtual double getArea(){return 0;};

virtual double getPerimeter(){return 0;};

virtual void show(){};

virtual string getClassName(){return "Shape";};

};

class Circle : public Shape

{

public:

Circle(double r = 1.0){

radius = r;

};

double getRadius();

void setRadius(double r);

double getArea();

double getPerimeter();

void show();

string getClassName();

private:

double radius;

};

double Circle::getRadius()

{

return radius;

}

void Circle::setRadius( double r )

{

radius = r;

}

double Circle::getArea()

{

return PI \* radius \* radius;

}

double Circle::getPerimeter()

{

return 2 \* PI \* radius;

}

void Circle::show()

{

cout << "Radius:" << radius << endl;

}

string Circle::getClassName(){

return "Circle";

}

class Rectangle : public Shape

{

public:

Rectangle(double w = 1.0, double h = 1.0){

width = w;

height = h;

};

double getWidth();

double getHeight();

void setWidth(double w);

void setHeight(double h);

double getArea();

double getPerimeter();

void show();

string getClassName();

private:

double width;

double height;

};

double Rectangle::getWidth()

{

return width;

}

double Rectangle::getHeight()

{

return height;

}

void Rectangle::setWidth( double w )

{

width = w;

}

void Rectangle::setHeight( double h )

{

height = h;

}

double Rectangle::getArea()

{

return width \* height;

}

double Rectangle::getPerimeter()

{

return 2 \* (width + height);

}

void Rectangle::show()

{

cout << "Width:" << width << ", " << "Height:" << height << endl;

}

string Rectangle::getClassName()

{

return "Rectangle";

}

class Triangle : public Shape

{

public:

Triangle(double s1, double s2, double s3){

side1 = s1;

side2 = s2;

side3 = s3;

};

double getArea();

double getPerimeter();

void show();

string getClassName();

private:

double side1;

double side2;

double side3;

};

double Triangle::getArea()

{

double p = (side1 + side2 + side3) / 2;

return sqrt(p \* (p - side1) \* (p - side2) \* (p - side3));

}

double Triangle::getPerimeter()

{

return side1 + side2 + side3;

}

void Triangle::show()

{

cout << "Side:" << side1 << ", " << side2 << ", " << side3 << endl;

}

string Triangle::getClassName()

{

return "Triangle";

}

题目得分 10

【描述】

将输入数据按特定要求原样输出。

【输入】

第一行是整数t，表明一共t组数据。

对每组数据：

第一行是整数n，表示下面一共有n行，0<n<100。

下面的每行，以一个字母开头，然后跟着一个整数，两者用空格分隔。字母只会是'A'或'B'。整数范围0到100。

【输出】

对每组输入数据，将其原样输出，每组数据的最后输出一行"\*\*\*\*"。

【输入示例】

2

4

A 3

B 4

A 5

B 6

3

A 4

A 3

A 2

【输出示例】

4

A 3

B 4

A 5

B 6

\*\*\*\*

3

A 4

A 3

A 2

\*\*\*\*

（10分）

我的答案：

class Base {

public:

Base(){};

Base(int n) {

val = n;

type = 'T';

};

int getVal(){return val;}

char getType(){return type;}

void setVal(int n){val = n;}

void setType(char t){type = t;}

private:

int val;

char type;

};

class A : public Base{

public:

A(){};

A(int n) {

setVal(n);

setType('A');

};

};

class B : public Base{

public:

B(){};

B(int n) {

setVal(n);

setType('B');

};

};

Base \*a[100];

void PrintInfo(Base \*x){

cout << x->getType() << " " << x->getVal() << endl;

}

题目得分 10

【描述】

按要求计算数值。

【输入】

第一行是整数n，表示第二行有n个整数。

第二行：n个整数。

所有整数都在0和100之间。

【输出】

先输出：

1

100

101

101

对于输入中第二行的每个整数x，输出两行：

第一行：k=x；

第二行：x的平方。

【输入示例】

3

3 4 5

【输出示例】

1

100

101

101

k=3

9

k=4

16

k=5

25

（10分）

我的答案：

A(int num){

n = num;

}

friend ostream& operator<<(ostream& cout, A& a){

cout << a.n;

return cout;

};

operator int(){

return n;

}

A& operator ++ (){

n = n + 1;

return \*this;

};

A operator ++ (int){

A a(\*this);

n = n + 1;

return a;

};

题目得分 10

【描述】

下面是类A的定义，需要补充或增加完整的无参构造函数、虚析构函数以及fun()虚函数。

class A {

public:

// ...

void g() {

fun();

}

};

构造函数输出：A constructor，并调用g()函数

析构函数输出：A destructor

fun()函数输出：Call class A's fun

下面是类B的定义，继承类A，需要补充或增加完整的无参构造函数、析构函数。

class B {

public:

// ...

};

无参构造函数输出：B constructor

析构函数输出：B destructor

下面是类C的定义，继承类B，需要补充或增加完整的无参构造函数、析构函数以及fun()函数。

class C {

public:

// ...

};

无参构造函数输出：C constructor

析构函数输出：C destructor

fun()函数输出：Call class C's fun

给定如下main函数：

int main() {

A \*a = new C;

a->g();

delete a;

return 0;

}

【输入】

没有输入。

【输出】

主函数的输出已经写好。

（10分）

我的答案：

class A {

public:

A(){

cout << "A constructor" << endl;

g();

};

virtual ~A(){

cout << "A destructor" << endl;

};

virtual void fun(){

cout << "Call class A's fun" << endl;

};

void g() {

fun();

};

};

class B : public A{

public:

B(){

cout << "B constructor" << endl;

};

virtual ~B(){

cout << "B destructor" << endl;

}

};

class C : public B{

public:

C(){

cout << "C constructor" << endl;

}

~C(){

cout << "C destructor" << endl;

}

void fun(){

cout << "Call class C's fun" << endl;

}

};

题目得分 10

【描述】

①声明并实现一个名为Vehicle的基类，表示汽车。Vehicle类包括：

int类型的私有数据成员numberOfDoors，表示车门数量。

int类型的私有数据成员numberOfCylinders，表示气缸数量。

string类型的私有数据成员color，表示汽车颜色。

double类型的私有数据成员fuelLevel，表示油位，即燃油数量。

int类型的私有数据成员transmissionType，表示变速箱类型，即0表示手动、1表示自动。

string类型的私有数据成员className，表示汽车类别。

有参构造函数，将车门数量、气缸数量、汽车颜色、燃油数量、变速箱类型设置为给定的参数。

访问器函数getNumberOfDoors、getNumberOfCylinders、getColor、getFuelLevel、getTransmissionType、getClassNme，分别用于访问车门数量、气缸数量、汽车颜色、燃油数量、变速箱类型、汽车类别。

更改器函数setColor、setFuelLevel、setClassNme，分别用于更改汽车颜色、燃油数量、汽车类别。

重载流插入运算符<<，输出相关信息。

②从Vehicle类派生出Taxi类，表示出租车。Taxi类有数据成员customers（是否载客，bool类型）以及有参构造函数，将车门数量、气缸数量、汽车颜色、燃油数量、变速箱类型、是否载客设置为给定的参数；访问器/更改器函数hasCustomers和setCustomers；重载流插入运算符<<，输出相关信息。

③从Vehicle类派生出Truck类，表示卡车。Truck类有数据成员cargo（是否载货，bool类型）以及有参构造函数，将车门数量、气缸数量、汽车颜色、燃油数量、变速箱类型、是否载货设置为给定的参数；访问器/更改器函数hasCargo和setCargo；重载流插入运算符<<，输出相关信息。

【输入】

输入车门数量、气缸数量，颜色、燃油数量、变速箱类型。

【输出】

见【输出示例】

【输入示例】

2 6 red 50 1

4 6 yellow 60 0

2 16 black 100 0

【输出示例】

Vehicle

Number of doors:2

Number of cylinders:6

Transmission type:Automatic

Color:red

Fuel level:50

Taxi

Number of doors:4

Number of cylinders:6

Transmission type:Manual

Color:yellow

Fuel level:60

Has no passengers

Truck

Number of doors:2

Number of cylinders:16

Transmission type:Manual

Color:black

Fuel level:100

Is carrying cargo

【来源】

《程序设计基础——以C++为例》第7章实验3。

（10分）

我的答案：

#include<iostream>

#include <string>

using namespace std;

class Vehicle

{

friend ostream & operator << (ostream &cout, Vehicle vehicle);

public:

Vehicle(){};

Vehicle(int doors, int cylinders, string carColor, double fuel, int transmission){

numberOfDoors = doors;

numberOfCylinders = cylinders;

color = carColor;

fuelLevel = fuel;

transmissionType = transmission;

className = "Vehicle";

};

int getNumberOfDoors();

int getNumberOfCylinders();

string getColor();

double getFuelLevel();

int getTransmissionType();

string getClassName();

void setDoors(int doors);

void setCylinders(int cylinders);

void setColor(string col);

void setFuelLevel(double fue);

void setTransmissionType(int transmission);

void setClassName(string type);

private:

int numberOfDoors;

int numberOfCylinders;

string color;

double fuelLevel;

int transmissionType;

string className;

};

int Vehicle::getNumberOfDoors()

{

return numberOfDoors;

}

int Vehicle::getNumberOfCylinders()

{

return numberOfCylinders;

}

std::string Vehicle::getColor()

{

return color;

}

double Vehicle::getFuelLevel()

{

return fuelLevel;

}

int Vehicle::getTransmissionType()

{

return transmissionType;

}

std::string Vehicle::getClassName()

{

return className;

}

void Vehicle::setDoors( int doors )

{

numberOfDoors = doors;

}

void Vehicle::setCylinders(int cylinders)

{

numberOfCylinders = cylinders;

}

void Vehicle::setColor( string col )

{

color = col;

}

void Vehicle::setFuelLevel( double fue )

{

fuelLevel = fue;

}

void Vehicle::setTransmissionType(int transmission)

{

transmissionType = transmission;

}

void Vehicle::setClassName( string type )

{

className = type;

}

ostream & operator << (ostream &cout, Vehicle vehicle)

{

cout << vehicle.getClassName() << endl;

cout << "Number of doors:" << vehicle.getNumberOfDoors() << endl;

cout << "Number of cylinders:" << vehicle.getNumberOfCylinders() << endl;

cout << "Transmission type:" << vehicle.getTransmissionType() << endl;

cout << "Color:" << vehicle.getColor() << endl;

cout << "Fuel level:" << vehicle.getFuelLevel() << endl;

return cout;

}

class Taxi : public Vehicle

{

friend ostream & operator << (ostream &cout, Taxi taxi);

public:

Taxi(int doors, int cylinders, string carColor, double fuel, int transmission, bool guest){

customers = guest;

setDoors(doors);

setCylinders(cylinders);

setColor(carColor);

setFuelLevel(fuel);

setTransmissionType(transmission);

setClassName("taxi");

};

bool hasCustomers();

void setCustomers(bool guest);

private:

bool customers;

};

bool Taxi::hasCustomers()

{

return customers;

}

void Taxi::setCustomers( bool guest )

{

customers = guest;

}

ostream & operator << (ostream &cout, Taxi taxi)

{

cout << taxi.getClassName() << endl;

cout << "Number of doors:" << taxi.getNumberOfDoors() << endl;

cout << "Number of cylinders:" << taxi.getNumberOfCylinders() << endl;

cout << "Transmission type:" << taxi.getTransmissionType() << endl;

cout << "Color:" << taxi.getColor() << endl;

cout << "Fuel level:" << taxi.getFuelLevel() << endl;

if(taxi.hasCustomers()) cout << "Has passengers" << endl;

else cout << "Has no passengers" << endl;

return cout;

}

class Truck : public Vehicle

{

friend ostream & operator << (ostream &cout, Truck truck);

public:

Truck(int doors, int cylinders, string carColor, double fuel, int transmission, bool load){

cargo = load;

setDoors(doors);

setCylinders(cylinders);

setColor(carColor);

setFuelLevel(fuel);

setTransmissionType(transmission);

setClassName("truck");

};

bool hasCargo();

void setCargo(bool load);

private:

bool cargo;

};

bool Truck::hasCargo()

{

return cargo;

}

void Truck::setCargo( bool load )

{

cargo = load;

}

ostream & operator << (ostream &cout, Truck truck)

{

cout << truck.getClassName() << endl;

cout << "Number of doors:" << truck.getNumberOfDoors() << endl;

cout << "Number of cylinders:" << truck.getNumberOfCylinders() << endl;

cout << "Transmission type:" << truck.getTransmissionType() << endl;

cout << "Color:" << truck.getColor() << endl;

cout << "Fuel level:" << truck.getFuelLevel() << endl;

if(truck.hasCargo()) cout << "Is carrying cargo" << endl;

else cout << "Is no carrying cargo" << endl;

return cout;

}

题目得分 10

《C++程序设计实践》实验十[期中]

班级： 学号： 2020212205113 姓名： 王俊琪 得分： 100

一、课内实验题（共10小题，100分）

题型得分 100

【描述】

声明并实现一个Line类，表示线段。Line类包括：

Point类的私有对象数据成员start和end，表示线段的两个端点。

有参构造函数，将线段端点设置为给定的参数。

成员函数slope，计算线段的斜率。

【输入】

输入4个数，分别表示点1坐标(x1, y1)和点2坐标(x2, y2)。

【输出】

输出直线的斜率。

【输入示例】

10 20 30 70

【输出示例】

2.5

【提示】

Point类可以参阅《程序基础基础——以C++为例》第5章实验1。

【来源】

《程序设计基础——以C为例》第6章实验1。

（10分）

我的答案：

class Point{

public:

Point(double X=0, double Y=0){

x = X;

y = Y;

}

double getX(){

return x;

}

double getY(){

return y;

}

private:

double x;

double y;

};

class Line : public Point{

public:

Line(Point s, Point e){

start = s;

end = e;

}

double slope();

private:

Point start;

Point end;

};

double Line::slope(){

return (end.getY() - start.getY()) / (end.getX() - start.getX());

}

题目得分 10

【描述】

声明并实现一个类模板Pair，表示数对。类模板Pair包括：

私有数据成员first和second，first的类型为类参数T1，second的类型为类参数T2。

有参构造函数，将first和second设置为给定的参数。

访问器函数，分别返回first和second的值。

成员函数print，输出一个数对。、

成员函数swap，交换两个数对的值。

重载关系运算符<、<=、>、>=、==、!=，判断两个数对之间的关系。如果满足关系，返回true，否则返回false。

【输入】

输入两行。第一行和第二行都是一个整数、一个浮点数，表示一个数对，整数和浮点数之间以空格间隔。

【输出】

见【输出示例】

【输入示例】

1 2.5

3 4.5

【输出示例】

p1:[1,2.5]

p2:[3,4.5]

p1:[3,4.5]

p2:[1,2.5]

p1 > p2

【提示】

约定数对p1小于p2，等价于p1.first小于p2.first或者p1.first大于等于p2.first且p1.second小于p2.second。

【来源】

《程序设计基础——以C++为例》第6章实验7。

（10分）

我的答案：

template <class T1, class T2>

class Pair{

public:

T1 first;

T2 second;

Pair(T1 f, T2 s) : first(f), second(s) { };

void print();

void swap(Pair<T1, T2>& p);

bool operator < (const Pair<T1, T2>& p) const;

bool operator == (const Pair<T1, T2>& p) const;

bool operator > (const Pair<T1, T2>& p) const;

};

template<class T1, class T2>

void Pair<T1, T2>::print(){

cout << "[" << first << "," << second << "]" << endl;

}

template<class T1, class T2>

void Pair<T1, T2>::swap(Pair<T1, T2>& p){

T1 fir = first;

T2 sec = second;

first = p.first;

second = p.second;

p.first = fir;

p.second = sec;

}

template<class T1, class T2>

bool Pair<T1, T2>::operator < (const Pair<T1, T2>& p) const{

return (first < p.first || (first >= p.first && second < p.second));

}

template<class T1, class T2>

bool Pair<T1, T2>::operator == (const Pair<T1, T2>& p) const{

return (first == p.first && second == p.second);

}

template<class T1, class T2>

bool Pair<T1, T2>::operator > (const Pair<T1, T2>& p) const{

return (first > p.first || (first <= p.first && second > p.second));

}

题目得分 10

【描述】

以成员函数的形式重载关系运算符（<、<=、==、!=、>、>=），使用半径比较Circle对象大小。

【输入】

输入一行，两个圆半径，圆半径以空格间隔。

【输出】

分行输出圆半径对应的圆对象的比较结果。

【输入示例】

5 6

【输出示例】

true

true

false

true

false

false

【提示】

Circle类可以参阅《程序设计基础——以C++为例》第5章【例5.1】。

【来源】

《程序设计基础——以C++为例》第6章实验3。

（10分）

我的答案：

class Circle{

public:

Circle(double r){

radius = r;

}

bool operator<(const Circle& c){

if(this->radius < c.radius)

return true;

else

return false;

}

bool operator<=(const Circle& c){

if(this->radius <= c.radius)

return true;

else

return false;

}

bool operator==(const Circle& c){

if(this->radius == c.radius)

return true;

else

return false;

}

bool operator!=(const Circle& c){

if(this->radius != c.radius)

return true;

else

return false;

}

bool operator>(const Circle& c){

if(this->radius > c.radius)

return true;

else

return false;

}

bool operator>=(const Circle& c){

if(this->radius >= c.radius)

return true;

else

return false;

}

private:

double radius;

};

题目得分 10

【描述】

声明并实现一个Time类，表示时间。Time类包括：

int类型的私有数据成员hour、minute、second，表示时、分、秒。

带默认参数的构造函数，将时、分、秒设置为给定的参数。时、分、秒的默认参数值为0。需要检查时、分、秒的有效性。如果超出允许范围，则抛出invalid\_argument("Invalid argument!")异常。

访问器函数getHour、getMinute和getSecond，分别用于访问时、分、秒。

成员函数setTime，用于同时修改时、分、秒。需要检查时、分、秒的有效性。如果超出允许范围，则抛出invalid\_argument("Invalid argument!")异常。

重载自增运算符++，将时间递增1秒。要考虑增加1秒后，时间增加到下一分钟、下一小时、下一天的情况。要考虑前置++和后置++。

重载关系运算符==，判断两个时间是否相等。

使用类型转换函数，将时间转换为秒数。

重载流插入运算符<<，以24小时制格式输出时间，格式为时:分:秒。

【输入】

输入一行，有3个数据，分别表示时、分、秒，以空格间隔。

【输出】

见【输出示例】

【输入示例】

23 59 57

【输出示例】

t1:23:59:57

t2:00:00:00

t1:23:59:58

t2:23:59:57

t1 != t2

86398

【来源】

《程序设计基础——以C++为例》第6章实验5。

（10分）

我的答案：

class Time {

friend ostream& operator<<(ostream& cout, Time& t);

public:

Time(int h = 0, int m = 0, int s = 0){

setHour(h);

setMinute(m);

setSecond(s);

}

int getHour();

int getMinute();

int getSecond();

void setHour(int h);

void setMinute(int m);

void setSecond(int s);

bool operator == (const Time& t) const;

bool operator != (const Time& t) const;

operator int();

Time& operator ++ (); //前置

Time operator ++ (int); //后置

private:

int hour;

int minute;

int second;

};

int Time::getHour() {

return hour;

}

int Time::getMinute() {

return minute;

}

int Time::getSecond() {

return second;

}

void Time::setHour(int h) {

if (h < 0 || h >= 24)

throw invalid\_argument("Invalid argument!");

hour = h;

}

void Time::setMinute(int m) {

if (m < 0 || m >= 60)

throw invalid\_argument("Invalid argument!");

minute = m;

}

void Time::setSecond(int s) {

if (s < 0 || s >= 60)

throw invalid\_argument("Invalid argument!");

second = s;

}

bool Time::operator==(const Time& t) const{

return (hour == t.hour && minute == t.minute && second == t.second);

}

bool Time::operator!=(const Time& t) const

{

return (hour != t.hour || minute != t.minute || second != t.second);

}

Time::operator int(){

return (hour \* 60 + minute) \* 60 + second;

}

ostream& operator<<(ostream& cout, Time& t){

cout << setfill('0') << setw(2) << t.hour << ":";

cout << setfill('0') << setw(2) << t.minute << ":";

cout << setfill('0') << setw(2) << t.second << endl;

return cout;

}

Time& Time::operator++(){

if (second < 59) {

second += 1;

}

else {

if (minute < 59) {

minute += 1;

}

else {

if (hour < 23) hour += 1;

else hour = 0;

minute = 0;

second = 0;

}

}

return \*this;

}

Time Time::operator++(int){

Time temp(\*this);

if (second < 59) {

second += 1;

}

else {

if (minute < 59) {

minute += 1;

}

else {

if (hour < 23) hour += 1;

else hour = 0;

minute = 0;

second = 0;

}

}

return temp;

}

题目得分 10

【描述】

我们都知道，对于一个表达式可以这么写：1+2-3+4

那我们是否也可以这样想，在对象的层次上能不能也有这样类似的级联（链式）调用来完成这样的功能呢？

例如上边的表达式可以变成如下格式：

Number op(1);

op.add(2).sub(3).add(4)

op.print(); // 在一行上输出最终结果：4

注意：本题所涉及的类型均为int型，无需考虑其他类型。

现在你被安排完成这样一个级联调用的功能，给定类Number，请你完成其中的add方法和sub方法，这两个方法分别接受一个int型的参数，此外Number还有一个有参构造函数，接受的参数也是int型，最后你需要实现一个print的方法，用于打印结果。

【输入】

输入在一行中给出三个整数a、b、c，以空格间隔。

【输出】

输出a-b+c的结果。

【输入示例】

1 3 5

【输出示例】

3

（10分）

我的答案：

class Number{

public:

Number(int n){

num = n;

}

Number& add(int n);

Number& sub(int n);

void print();

private:

int num;

};

Number& Number::add(int n){

num += n;

return \*this;

}

Number& Number::sub(int n){

num -= n;

return \*this;

}

void Number::print(){

cout << num << endl;

}

题目得分 10

【描述】

声明和实现一个向量类MyVector，包括一个点的坐标位置x、y和z，实现其构造函数和三个友元函数，完成两个向量的加法、减法、点乘与叉乘运算。

【输入】

输入一个点的坐标位置x、y和z。

【输出】

见【输出示例】

【输入示例】

3 4 5

【输出示例】

(1,0,0)

(-1,-3,0)

(15,-10,-1)

18

（10分）

我的答案：

MyVector::MyVector(int a, int b, int c){

x = a;

y = b;

z = c;

}

MyVector::MyVector(const MyVector &v){

x = v.x;

y = v.y;

z = v.z;

}

MyVector add(MyVector &v1, MyVector &v2){

MyVector v;

v.x = v1.x + v2.x;

v.y = v1.y + v2.y;

v.z = v1.z + v2.z;

return v;

}

MyVector sub(MyVector &v1, MyVector &v2){

MyVector v;

v.x = v1.x - v2.x;

v.y = v1.y - v2.y;

v.z = v1.z - v2.z;

return v;

}

int dot(MyVector &v1, MyVector &v2){

return v1.x\*v2.x + v1.y\*v2.y + v1.z\*v2.z;

}

MyVector cross(MyVector &v1, MyVector &v2){

MyVector v;

v.x = v1.y\*v2.z - v1.z\*v2.y;

v.y = v2.x\*v1.z - v1.x\*v2.z;

v.z = v1.x\*v2.y - v1.y\*v2.x;

return v;

}

void MyVector::display(){

cout << "(" << x << "," << y << "," << z << ")" << endl;

}

题目得分 10

【描述】

请根据main函数中对该类的操作，补充类实现部分，完成代码。

该类有个私有静态变量count记录该类的所有对象数，main将会在不同语句之后输出对象数，只有正确地实现该类，保证count正确记录该类的对象数，才能输出正确的结果。

【输入】

没有输入。

【输出】

主函数的输出已经写好。

（10分）

我的答案：

Student(int ID = 0) {

id = ID;

}

friend void PrintCount();

friend void Print(Student s);

static void InitCount() {}

题目得分 10

【描述】

声明并实现一个带下标越界检查的TwoArray类，表示二维数组（矩阵），存放整数。TwoArray类包括：

int类型的指针pArray，指向存放数组元素的动态分配内存空间。

int类型的私有数据成员rowSize，存放数组行长度。

int类型的私有数据成员columnSize，存放数组列长度。

有参构造函数，将数组行、列长度设置为给定的参数，并动态分配对应的内存空间。

拷贝构造函数。

析构函数。

重载赋值运算符。

重载函数调用运算符。

两个访问器函数getRowSize和getColumnSize，用于获取数组行、列长度。

函数调用运算符()必须以成员函数形式进行重载。假设matrix是TwoArray类的对象，重载函数调用运算符的意图是使用matrix(i, j)这样的形式来访问二维数组元素，而不是原来的matrix[i][j]的形式。

重载函数调用运算符的语法如下：

函数类型operator()(参数表);

和重载下标运算符一样，在twoArray类中，重载函数调用运算符有两种形式：

int &operator()(int row, int column);

以及：

int operator()(int row, int column)const;

定义普通函数：void printTwoArray(const TwoArray &array)

按行输出二维数组，每个元素宽度为3。

【输入】

输入一个3×4矩阵和一个4×4矩阵。

【输出】

见【输出示例】

【输入示例】

0 1 2 3

2 3 4 5

4 5 6 7

0 1 2 3

3 4 5 6

6 7 8 9

9 10 11 12

【输出示例】

array1:

0 1 2 3

2 3 4 5

4 5 6 7

array2:

0 1 2 3

3 4 5 6

6 7 8 9

9 10 11 12

array3:

0 1 2 3

2 3 4 5

4 5 6 7

Row index out of range!

array1:

0 1 2 3

3 4 5 6

6 7 8 9

9 10 11 12

【来源】

《程序设计基础——以C++为例》第6章实验4。

（10分）

我的答案：

class MyException : public out\_of\_range {

public:

MyException(string str) : out\_of\_range(str.c\_str()) {}

};

class TwoArray {

public:

TwoArray(int row, int col) {

rowSize = row;

colSize = col;

pArray = new int\*[row];

for (int i = 0; i < row; i++) {

pArray[i] = new int[col];

}

}

~TwoArray(){}

int& operator()(int row, int col) {

if (row >= rowSize)

throw MyException("Row index out of range!");

if (col >= colSize)

throw MyException("Column index out of range!");

return pArray[row][col];

}

friend void printTwoArray(const TwoArray& array);

int getRowSize();

int getColumnSize();

private:

int\*\* pArray;

int rowSize;

int colSize;

};

void printTwoArray(const TwoArray& array) {

for (int i = 0; i < array.rowSize; i++) {

for (int j = 0; j < array.colSize; j++) {

cout << setw(3) << array.pArray[i][j];

}

cout << endl;

}

}

int TwoArray::getRowSize() {

return rowSize;

}

int TwoArray::getColumnSize() {

return colSize;

}

题目得分 10

【描述】

声明并实现一个Rational类，表示有理数。有理数的＋、－、×和÷运算要求通过重载运算符来实现。Rational类包括：

int类型的私有数据成员numerator、denominator，表示分子、分母。

私有成员函数gcd，用于求分子、分母的最大公约数。

私有成员函数compare，用于判断两个有理数之间的大小关系。一个有理数与另一个有理数相比是大于、等于还是小于，分别返回1、0和-1。

带默认参数值的构造函数，将分子、分母设置为给定的参数。分子、分母的默认参数值分别为0、1。

访问器函数getNumerator、getDenominator，分别用于获取分子、分母。

重载算术运算符+、-、\*、/，实现有理数算术运算。

重载自增运算符++和自减运算符--。要同时考虑前置和后置。

重载复合算术赋值运算符+=、-=、\*=、/=。

重载关系运算符<、<=、>、>=、==、!=，判断两个有理数之间的关系。如果满足关系，返回true，否则返回false。

使用类型转换函数，将有理数转换为浮点数。

重载流提取运算符>>和流插入运算符<<。输入输出一个有理数。

【输入】

输入两行。第一行为第一个有理数的分子、分母，以空格间隔。第二行为第二个有理数的分子、分母，以空格间隔。

【输出】

见【输出示例】。

【输入示例】

7 3

1 3

【输出示例】

a=7/3

b=1/3

7/3+1/3=8/3

7/3-1/3=2

7/3\*1/3=7/9

7/3/1/3=7

7/3<1/3=false

7/3>1/3=true

7/3==1/3=false

7/3!=1/3=true

c=1/2

2.5+1/2=3

3/2

3/2

3/2

5/2

17/6

【来源】

《程序设计基础——以C++为例》第6章实验6。

（10分）

我的答案：

class Rational {

friend istream& operator>>(istream& cin, Rational& r);

friend ostream& operator<<(ostream& cout, const Rational& r);

friend Rational operator +(const double& n, const Rational& ra);

friend Rational operator -(const double& n, const Rational& ra);

friend Rational operator \*(const double& n, const Rational& ra);

friend Rational operator /(const double& n, const Rational& ra);

public:

Rational(int a = 0, int b = 1) {

int n = gcd(a, b);

numerator = a / n;

denominator = b / n;

}

Rational(const double num) {

double n = num, d = 1;

while ((int)n != n) {

n \*= 10;

d \*= 10;

}

numerator = n;

denominator = d;

numerator /= gcd(numerator, denominator);

denominator /= gcd(numerator, denominator);

}

Rational(const Rational& r) {

numerator = r.numerator;

denominator = r.denominator;

numerator /= gcd(numerator, denominator);

denominator /= gcd(numerator, denominator);

}

int getNumerator();

int getDenominator();

bool operator < (const Rational& r) const;

bool operator <= (const Rational& r) const;

bool operator > (const Rational& r) const;

bool operator >= (const Rational& r) const;

bool operator == (const Rational& r) const;

bool operator != (const Rational& r) const;

operator double();

Rational& operator ++ (); //前置

Rational operator ++ (int); //后置

Rational operator +(const Rational& ra);

Rational operator -(const Rational& ra);

Rational operator \*(const Rational& ra);

Rational operator /(const Rational& ra);

Rational operator +(const double& n);

Rational operator -(const double& n);

Rational operator \*(const double& n);

Rational operator /(const double& n);

Rational& operator +=(const Rational& ra);

Rational& operator -=(const Rational& ra);

Rational& operator \*=(const Rational& ra);

Rational& operator /=(const Rational& ra);

Rational& operator +=(const double& n);

Rational& operator -=(const double& n);

Rational& operator \*=(const double& n);

Rational& operator /=(const double& n);

private:

int numerator; //分子

int denominator; //分母

int gcd(int a, int b);

int compare(Rational& r);

};

int Rational::gcd(int a, int b) {

if (b == 0)

return 1;

if (a % b == 0)

return b;

else

return gcd(b, a % b);

}

int Rational::compare(Rational& r) {

if (\*this > r)

return 1;

else if (\*this < r)

return -1;

else

return 0;

}

int Rational::getNumerator() {

return numerator;

}

int Rational::getDenominator()

{

return denominator;

}

bool Rational::operator<(const Rational& r) const

{

return numerator \* r.denominator < r.numerator \* denominator;

}

bool Rational::operator<=(const Rational& r) const

{

return numerator \* r.denominator <= r.numerator \* denominator;

}

bool Rational::operator>(const Rational& r) const

{

return numerator \* r.denominator > r.numerator \* denominator;

}

bool Rational::operator>=(const Rational& r) const

{

return numerator \* r.denominator >= r.numerator \* denominator;

}

bool Rational::operator==(const Rational& r) const

{

return numerator \* r.denominator == r.numerator \* denominator;

}

bool Rational::operator!=(const Rational& r) const

{

return numerator \* r.denominator != r.numerator \* denominator;

}

istream& operator>>(istream& cin, Rational& r)

{

int n,d;

cin >> n >> d;

Rational ra(n, d);

r = ra;

return cin;

}

ostream& operator<<(ostream& cout, const Rational& r)

{

if (r.denominator < 0) {

cout << -r.numerator << "/" << -r.denominator;

}

else if (r.denominator == 1) {

cout << r.numerator;

}

else {

cout << r.numerator << "/" << r.denominator;

}

return cout;

}

Rational::operator double() {

return numerator \* 1.0 / denominator;

}

Rational Rational::operator +(const double& n)

{

Rational tmp(n);

return \*this + tmp;

}

Rational Rational::operator -(const double& n)

{

Rational tmp(n);

return \*this - tmp;

}

Rational Rational::operator \*(const double& n)

{

Rational tmp(n);

return \*this \* tmp;

}

Rational Rational::operator /(const double& n)

{

Rational tmp(n);

return \*this / tmp;

}

Rational operator +(const double& n, const Rational& ra)

{

Rational tmp(n);

return tmp + ra;

}

Rational operator -(const double& n, const Rational& ra)

{

Rational tmp(n);

return tmp - ra;

}

Rational operator \*(const double& n, const Rational& ra)

{

Rational tmp(n);

return tmp \* ra;

}

Rational operator /(const double& n, const Rational& ra)

{

Rational tmp(n);

return tmp / ra;

}

Rational& Rational::operator +=(const Rational& ra)

{

Rational tmp = \*this + ra;

\*this = tmp;

return \*this;

}

Rational& Rational::operator -=(const Rational& ra)

{

Rational tmp = \*this - ra;

\*this = tmp;

return \*this;

}

Rational& Rational::operator \*=(const Rational& ra)

{

Rational tmp = \*this \* ra;

\*this = tmp;

return \*this;

}

Rational& Rational::operator /=(const Rational& ra)

{

Rational tmp = \*this / ra;

\*this = tmp;

return \*this;

}

Rational& Rational::operator +=(const double& ra)

{

Rational tmp(ra);

\*this += tmp;

return \*this;

}

Rational& Rational::operator -=(const double& ra)

{

Rational tmp(ra);

\*this -= tmp;

return \*this;

}

Rational& Rational::operator \*=(const double& ra)

{

Rational tmp(ra);

\*this \*= tmp;

return \*this;

}

Rational& Rational::operator /=(const double& ra)

{

Rational tmp(ra);

\*this /= tmp;

return \*this;

}

Rational& Rational::operator++()

{

numerator += denominator;

return \*this;

}

Rational Rational::operator++(int)

{

Rational temp(\*this);

numerator = numerator + denominator;

return temp;

}

Rational Rational::operator+(const Rational& r)

{

return Rational(numerator \* r.denominator + denominator \* r.numerator, denominator \* r.denominator);

}

Rational Rational::operator-(const Rational& r)

{

return Rational(numerator \* r.denominator - denominator \* r.numerator, denominator \* r.denominator);

}

Rational Rational::operator\*(const Rational& r)

{

return Rational(numerator \* r.numerator, denominator \* r.denominator);

}

Rational Rational::operator/(const Rational& r)

{

return Rational(numerator \* r.denominator, denominator \* r.numerator);

}

题目得分 10

【描述】

第6题中的TwoArray类，表示二维数组（矩阵），只能存放整数。修改TwoArray类，使其成为一个类模板。并新增重载关系运算符==和!=，判断两个矩阵是否相等。

【输入】

输入一个3×4矩阵和一个4×4矩阵。

【输出】

见【输出示例】

【输入示例】

0 1 2 3

2 3 4 5

4 5 6 7

0 1 2 3

3 4 5 6

6 7 8 9

9 10 11 12

【输出示例】

array1:

0 1 2 3

2 3 4 5

4 5 6 7

array2:

0 1 2 3

3 4 5 6

6 7 8 9

9 10 11 12

array3:

Row index out ofrange!

array1:

0 1 2 3

3 4 5 6

6 7 8 9

9 10 11 12

array1 == array2

【提示】

设置输出宽度为3。

【来源】

《程序设计基础——以C++为例》第6章实验8

（10分）

我的答案：

class MyException : public out\_of\_range {

public:

MyException(string str) : out\_of\_range(str.c\_str()) {}

};

template<typename Type>

class TwoArray {

friend void printTwoArray(const TwoArray<Type>& array) {

for (int i = 0; i < array.rowSize; i++) {

for (int j = 0; j < array.colSize; j++) {

cout << setw(3) << array.pArray[i][j];

}

cout << endl;

}

}

public:

TwoArray(Type row, Type col) {

rowSize = row;

colSize = col;

pArray = new int\* [row];

for (int i = 0; i < row; i++) {

pArray[i] = new int[col];

}

}

int& operator()(Type row, Type col) const;

bool operator == (const TwoArray<Type>& r) const;

bool operator != (const TwoArray<Type>& r) const;

int getRowSize();

int getColumnSize();

private:

Type\*\* pArray;

Type rowSize;

Type colSize;

};

template<typename Type>

int TwoArray<Type>::getRowSize() {

return rowSize;

}

template<typename Type>

int TwoArray<Type>::getColumnSize() {

return colSize;

}

template<typename Type>

int& TwoArray<Type>::operator()(Type row, Type col) const

{

if (row >= rowSize)

throw MyException("Row index out of range!");

if (col >= colSize)

throw MyException("Column index out of range!");

return pArray[row][col];

}

template<typename Type>

bool TwoArray<Type>::operator == (const TwoArray<Type>& r) const {

bool flag = true;

if (rowSize != r.rowSize || colSize != r.colSize)

flag = false;

for (int i = 0; i < rowSize; i++) {

for (int j = 0; j < colSize; j++) {

if (pArray[i][j] != r.pArray[i][j])

flag = false;

}

}

return flag;

}

template<typename Type>

bool TwoArray<Type>::operator != (const TwoArray<Type>& r) const {

bool flag = false;

if (rowSize != r.rowSize || colSize != r.colSize)

flag = true;

for (int i = 0; i < rowSize; i++) {

for (int j = 0; j < colSize; j++) {

if (pArray[i][j] != r.pArray[i][j])

flag = true;

}

}

return flag;

}

题目得分 10

《C++程序设计实践》实验9[期中]

班级： 学号： 2020212205113 姓名： 王俊琪 得分： 100

一、课内实验题（共10小题，100分）

题型得分 100

【描述】

声明并实现一个Time类，表示时间。Time类包括：

int类型的私有数据成员hour、minute、second，表示时、分、秒。

带默认参数的构造函数，将时、分、秒设置为给定的参数。时、分、秒的默认参数值为0。需要检查时、分、秒的有效性。若超出允许范围，则抛出invalid\_argument标准异常。

更改器函数setHour、setMinute和setSecond，分别用于修改时、分、秒。每个更改器函数都要检查对应的时、分、秒的有效性。若超出允许范围，则抛出invalid\_argument标准异常。

访问器函数getHour、getMinute和getSecond，分别用于访问时、分、秒。

成员函数setTime，用于同时修改时、分、秒。需要检查时、分、秒的有效性。若超出允许范围，则抛出invalid\_argument标准异常。

成员函数printTime24，以24小时制格式输出时间，格式为时:分:秒。

成员函数printTime12，以12小时制格式输出时间，格式为上午或下午时:分:秒。

成员函数tick，将时间递增1秒。要考虑增加1秒后，时间增加到下一分钟、下一小时、下一天的情况。

【输入】

没有输入。

【输出】

00:00:00

AM12:00:00

02:00:00

AM2:00:00

21:34:00

PM9:34:00

12:25:42

PM12:25:42

Invalid argument!

00:00:00

AM12:00:00

【来源】

《程序设计基础——以C++为例》第5章实验6。

（10分）

我的答案：

class Time{

public:

Time(int hour = 0, int minute = 0, int second = 0){

setHour(hour);

setMinute(minute);

setSecond(second);

}

void setHour(int hour);

void setMinute(int minute);

void setSecond(int second);

void setTime(int hour, int minute, int second);

int getHour();

int getMinute();

int getSecond();

void printTime24();

void printTime12();

void tick();

private:

int hours;

int minutes;

int seconds;

};

void Time::setHour(int hour){

if(hour < 0 || hour >= 24){

throw invalid\_argument("Invalid argument!");

}

hours = hour;

}

void Time::setMinute(int minute){

if(minute < 0 || minute >= 60){

throw invalid\_argument("Invalid argument!");

}

minutes = minute;

}

void Time::setSecond(int second){

if(second < 0 || second >= 60){

throw invalid\_argument("Invalid argument!");

}

seconds = second;

}

int Time::getHour(){

return hours;

}

int Time::getMinute(){

return minutes;

}

int Time::getSecond(){

return seconds;

}

void Time::printTime24(){

cout << setfill('0') << setw(2) << hours << ":";

cout << setfill('0') << setw(2) << minutes << ":";

cout << setfill('0') << setw(2) << seconds << endl;

}

void Time::printTime12(){

cout << ((hours < 12) ? "AM" : "PM");

hours = (hours == 0 || hours == 12) ? 12 : hours;

cout << setfill('0') << setw(2) << hours << ":" << setw(2) << minutes << ":" << setw(2) << seconds << endl;

}

void Time::tick(){

if(seconds < 59){

seconds += 1;

}

else{

if(minutes < 59){

minutes += 1;

}

else{

if(hours < 23){

hours += 1;

}

else{

hours = 0;

minutes = 0;

seconds = 0;

}

}

}

}

题目得分 10

【描述】

声明并实现一个Account类，表示银行账户。Account类包括：

string类型的私有数据成员id，表示账号；string类型的私有数据成员name，表示客户名；double类型的私有数据成员balance，表示账户余额；double类型的私有数据成员annualInterestRate，表示年利率。

有参构造函数，将账号、客户名、账户余额、年利率设置为给定的参数。

更改器函数setId、setName、setBalance和setAnnualInterestRate，分别用于修改账号、客户名、账户余额、年利率。

访问器函数getId、getName、getBalance和getAnnualInterestRate，分别用于访问账号、客户名、账户余额、年利率。

成员函数withdraw，从账户中取款。

成员函数deposit，往账户中存款。

成员函数computeMonthlyInterestRate，返回月利率。

成员函数print，输出账户信息。

【输入】

输入账号、客户名、账户余额和年利率。

【输出】

输出账号、客户名、账户余额和月利率。

【输入示例】

112233 ZhangSan 20000 4.5

【输出示例】

112233

ZhangSan

20500

0.375%

【提示】

假设年利率为4.5%，则以4.5作为输入值。

月利率=年利率/12

【来源】

《程序设计基础——以C++为例》第5章实验4。

（10分）

我的答案：

class Account{

public:

Account(string account, string user, double money, double air){

id = account;

name = user;

balance = money;

annualInterestRate = air;

}

void setId(double account);

void setBalance(double money);

void setAnnualInterestRate(double air);

string getId();

string getName();

double getBalance();

double getAnnualInterestRate();

void withdraw(double money);

void deposit(double money);

double computeMonthlyInterestRate();

void print();

private:

string id;

string name;

double balance;

double annualInterestRate;

};

void Account::setId(double account){

id = account;

}

void Account::setBalance(double money){

balance = money;

}

void Account::setAnnualInterestRate(double air){

annualInterestRate = air;

}

string Account::getId(){

return id;

}

string Account::getName(){

return name;

}

double Account::getBalance(){

return balance;

}

double Account::getAnnualInterestRate(){

return annualInterestRate;

}

void Account::withdraw(double money){

balance -= money;

}

void Account::deposit(double money){

balance += money;

}

double Account::computeMonthlyInterestRate(){

return annualInterestRate / 12;

}

void Account::print(){

cout << id << endl;

cout << name << endl;

cout << balance << endl;

cout << computeMonthlyInterestRate() << "%";

}

题目得分 10

【描述】

自定义异常类NegativeNumberException，表示对负数执行操作时出现的异常，如计算负数的平方根。该类有一个string类型的私有数据成员message，用来存放异常信息；一个无参（默认）构造函数和一个有参构造函数，用来设置异常信息；成员函数what，用来显示异常信息。在main函数中，让用户输入某个数，并调用squareRoot函数，计算该数的平方根。如果输入的是负数，squareRoot函数将抛出NegativeNumberException异常，否则返回该数的平方根。

【输入】

输入一个数。

【输出】

输出该数的平方根或者输出错误信息“Invalid argument!”。

【输入示例】

-8

【输出示例】

Invalid argument!

【来源】

《程序设计基础——以C++为例》第5章实验7。

（10分）

我的答案：

class NegativeNumberException{

public:

NegativeNumberException(){

message = "Invalid argument!";

}

NegativeNumberException(string msg){

message = msg;

}

string what(){

return message;

}

private:

string message;

};

double squareRoot(double value){

if(value < 0){

throw NegativeNumberException();

}

return sqrt(value);

}

题目得分 10

【描述】

声明并实现一个Cylinder类，表示圆柱体。Cylinder类包括：

double类型的私有数据成员radius和height，分别表示圆柱体底半径和高。

带默认参数的构造函数，将圆柱体底半径和高设置为给定的参数。半径和高的默认参数值为1。

访问器函数，分别用于访问圆柱体底半径和高。

成员函数computeVolume，返回圆柱体体积。

成员函数computeSurfaceArea，返回圆柱体表面积。

假设PI为3.14159。

【输入】

输入圆柱体的底半径和高。

【输出】

输出圆柱体的体积和表面积。

【输入示例】

4 8

【输出示例】

402.124

301.593

【来源】

《程序设计基础——以C++为例》第5章实验3。

（10分）

我的答案：

class Cylinder{

public:

Cylinder(double r = 1, double h = 1){

radius = r;

height = h;

}

double getRaidus();

double getHeight();

void setRaidus(double r);

void setHeight(double h);

double computeVolume();

double computeSurfaceArea();

private:

double radius;

double height;

};

double Cylinder::getRaidus(){

return radius;

}

double Cylinder::getHeight(){

return height;

}

void Cylinder::setRaidus(double r){

radius = r;

}

void Cylinder::setHeight(double h){

height = h;

}

double Cylinder::computeVolume(){

return PI \* radius \* radius \* height;

}

double Cylinder::computeSurfaceArea(){

return 2 \* PI \* radius \* radius + 2 \* PI \* radius \* height;

}

题目得分 10

【描述】

声明并实现一个Point类，表示直角坐标系中的一个点。Point类包括：

double类型的私有数据成员x和y，表示坐标。

无参（默认）构造函数，将坐标设置为原点。

有参构造函数，将坐标设置为给定的参数。

访问器函数getX和getY，分别用于访问点的x坐标和y坐标。

【输入】

0,0 4,5

【输出】

(0,0)

(4,5)

【来源】

《程序设计基础——以C++为例》第5章实验1。

（10分）

我的答案：

class Point{

public:

Point(double \_x = 0, double \_y = 0){

x = \_x;

y = \_y;

}

double getX();

double getY();

private:

double x;

double y;

};

double Point::getX(){

return x;

}

double Point::getY(){

return y;

}

题目得分 10

【描述】

声明并实现一个Rectangle类，表示矩形。Rectangle类包括：

double类型的私有数据成员width和height，表示矩形的宽和高。

带默认参数的构造函数，将矩形的宽和高设置为给定的参数。宽和高的默认参数值为1。

更改器函数setWidth和setHeight，分别用于修改矩形的宽和高。

访问器函数getWidth和getHeight，分别用于访问矩形的宽和高。

成员函数computeArea，返回矩形的面积。

成员函数computePerimeter，返回矩形的周长。

【输入】

5 40

10 3.5

【输出】

200 90

35 27

【来源】

《程序设计基础——以C++为例》第5章实验2。

（10分）

我的答案：

class Rectangle{

public:

Rectangle(double w = 1, double h = 1){

width = w;

height = h;

}

void setWidth(double w);

void setHeight(double h);

double getWidh();

double getHeight();

double computeArea();

double computePerimeter();

private:

double width;

double height;

};

void Rectangle::setWidth(double w){

width = w;

}

void Rectangle::setHeight(double h){

height = h;

}

double Rectangle::getWidh(){

return width;

}

double Rectangle::getHeight(){

return height;

}

double Rectangle::computeArea(){

return width \* height;

}

double Rectangle::computePerimeter(){

return 2 \* (width + height);

}

题目得分 10

【描述】

有理数是由分子和分母组成的a/b形式的数，a是分子，b是分母。例如，1/3、3/4和10/4等都是有理数。有理数不能以0为分母，但可以以0为分子。整数a等价于有理数a/1。

有理数用于包含分数的精确运算。例如，1/3=0.333333……，这个数是不能用浮点数精确表示，为了得到精确的结果，必须使用有理数。

一个有理数可能有很多与其值相等的其他有理数，例如，1/3=2/6=3/9=4/12。为简单起见，用1/3表示所有值等于1/3的有理数。因此，需要对有理数进行优化，使分子和分母之间没有公约数（1除外）。求分子和分母绝对值的最大公约数，然后将分子和分母都除以此最大公约数，得到有理数的优化表示形式。

声明并实现一个Rational类，表示有理数。Rational类包括：

int类型的私有数据成员numerator、denominator，表示分子、分母。

私有成员函数gcd，用于求分子、分母的最大公约数。

带默认参数的构造函数，将分子、分母设置为给定的参数。分子、分母的默认参数值分别为0、1。

访问器函数getNumerator、getDenominator，分别用于访问分子、分母。

成员函数add、subtract、multiply、divide，实现有理数的＋、－、×和÷运算。

成员函数equals，判断一个有理数与另一个有理数是否相等。如果相等，返回true，否则返回false。

成员函数doubleValue，将有理数转换为浮点数。

成员函数print，输出一个有理数。若分母为0，则输出“Inf”。

【输入】

4/2 2/3

【输出】

2

2/3

8/3

4/3

4/3

3

a!=b

0.666667

【来源】

《程序设计基础——以C++为例》第5章实验9。

（10分）

我的答案：

class Rational {

public:

Rational(int a = 0, int b = 1) {

numerator = a;

denominator = b;

int n = gcd(numerator, denominator);

numerator /= n;

denominator /= n;

}

int getNumerator();

int getDenominator();

Rational add(Rational r);

Rational subtract(Rational r);

Rational multiply(Rational r);

Rational divide(Rational r);

void print();

bool equals(Rational r);

double doubleValue();

private:

int numerator; //分子

int denominator; //分母

int gcd(int a, int b);

};

int Rational::gcd(int a, int b) {

if (b == 0)

return 1;

if (a % b == 0)

return b;

else

return gcd(b, a % b);

}

int Rational::getNumerator() {

return numerator;

}

int Rational::getDenominator() {

return denominator;

}

Rational Rational::add(Rational r) {

Rational temp;

temp.numerator = numerator \* r.getDenominator() + denominator \* r.getNumerator();

temp.denominator = denominator \* r.getDenominator();

int num = gcd(temp.numerator, temp.denominator);

temp.numerator /= num;

temp.denominator /= num;

return temp;

}

Rational Rational::subtract(Rational r) {

r.numerator = -r.numerator;

return add(r);

}

Rational Rational::multiply(Rational r) {

Rational temp;

temp.numerator = numerator \* r.getNumerator();

temp.denominator = denominator \* r.getDenominator();

int num = gcd(temp.numerator, temp.denominator);

temp.numerator /= num;

temp.denominator /= num;

return temp;

}

Rational Rational::divide(Rational r) {

Rational temp;

temp.numerator = numerator \* r.getDenominator();

temp.denominator = denominator \* r.getNumerator();

int num = gcd(temp.numerator, temp.denominator);

temp.numerator /= num;

temp.denominator /= num;

return temp;

}

void Rational::print() {

if (denominator == 1) {

cout << numerator << endl;

}

else if (denominator == 0) {

cout << "Inf" << endl;

}

else if(denominator < 0){

cout << "-" << numerator << "/" << -denominator << endl;

}

else {

cout << numerator << "/" << denominator << endl;

}

}

bool Rational::equals(Rational r) {

if (numerator \* r.denominator == r.numerator \* denominator) {

return true;

}

else {

return false;

}

}

double Rational::doubleValue() {

return numerator \* 1.0 / denominator;

}

题目得分 10

【描述】

小明是一个熊孩子，他很想知道某一个日期是星期几。他知道你正在学习C++，所以想请你写一个程序，实现一个Date类。Date类的定义如下：

class Date {

public:

Date(int year, int month, int day);

int getWeekday() const;

private:

int year;

int month;

int day;

};

成员变量year、month、day分别代表年月日。现在你要实现Date函数和getWeekday函数，并编写main函数。getWeekday函数，如果日期合法，返回1～7中某个数值，表示星期一到星期天中某一天（其中1为星期一），如果日期不合法，则返回-1。

【输入】

输入包含多组数据。每行包括三个整数year（2000≤y≤9999)、month、day。输入数据以0 0 0结束。

【输出】

每组数据输出一行，每行包括一个整数。1表示星期一，7表示星期日，等等，若数据非法，则输出-1。

0 0 0不需输出

【输入示例】

2013 2 26

2013 2 29

0 0 0

【输出示例】

2

-1

【提示】2000年1月1日是星期六。

求星期几：（6 + 2000年1月1日至输入日期的天数 - 1）% 7 + 1

（10分）

我的答案：

class Date {

public:

Date(int y, int m, int d){

year = y;

month = m;

day = d;

};

int getWeekday() const;

private:

int year;

int month;

int day;

};

int Date::getWeekday() const {

int flag = 1;

int daysOfMonth[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

if (year % 400 == 0 || (year % 4 == 0 && year % 100 != 0)) {

daysOfMonth[1] = 29;

}

if (year < 2000 || year > 9999) {

flag = 0;

}

else if (month > 12 || month < 1) {

flag = 0;

}

else if (day > daysOfMonth[month - 1] || day < 1) {

flag = 0;

}

int days = 0;

for (int i = 2000; i < year; i++) {

if ((i % 4 == 0 && i % 100 != 0) || (i % 400 == 0))

days += 366;

else

days += 365;

}

if (flag) {

for (int i = 0; i < month - 1; i++) {

days += daysOfMonth[i];

}

days += day;

return ((5 + days) % 7) ? (5 + days) % 7 : 7;

}

else {

return -1;

}

}

题目得分 10

【描述】

自定义异常类TriangleException。该类有一个string类型的私有数据成员message，用来存放异常信息；一个无参（默认）构造函数（默认值为“Exception”）和一个有参构造函数，用来设置异常信息；成员函数what，用来显示异常信息。

声明并实现一个Triangle类，表示三角形。Triangle类包括：

double类型的私有数据成员side1、side2、side3，表示三角形三条边。

私有成员函数isValid，判断三条边能否构成三角形。如果能构成三角形，返回true，否则返回false。

私有成员函数check，判断边是否大于0。如果边大于0，返回true，否则返回false。

带默认参数的构造函数，将三角形三条边设置为给定的参数。三条边的默认参数值为1。需要调用check函数检查边是否大于0，如果边小于等于0，则抛出TriangleException异常；以及调用isValid函数判断更改边后能否构成三角形，如果不能构成三角形，则抛出TriangleException异常。

更改器函数setSide1、setSide2和setSide3，分别用于修改三角形三条边。和构造函数一样，每个更改器函数需要调用check函数和isValid函数检查边的有效性。

访问器函数getSide1、getSide2和getSide3，分别用于访问三角形三条边。

成员函数computeArea，返回三角形面积。

成员函数computePerimeter，返回三角形周长。

【输入】

输入三角形的三条边，边长以空格间隔。

【输出】

若三条边有负数，输出“Negative side”。

若三条边能构成三角形，输出三角形面积和周长，以空格间隔。

若不能构成三角形，输出“Don't make a triangle”。

【来源】

《程序设计基础——以C++为例》第5章实验8。

（10分）

我的答案：

class TriangleException {

public:

TriangleException() {

message = "Exception";

}

TriangleException(string msg) {

message = msg;

}

string what() {

return message;

}

private:

string message;

};

class Triangle {

public:

Triangle(double s1 = 1, double s2 = 1, double s3 = 1) {

side1 = s1;

side2 = s2;

side3 = s3;

if (!check()) {

throw TriangleException("Negative side");

}

else if (!isValid()) {

throw TriangleException("Don't make a triangle");

}

}

void setSide1(double s1);

void setSide2(double s2);

void setSide3(double s3);

double getSide1();

double getSide2();

double getSide3();

double computeArea();

double computePerimeter();

private:

double side1;

double side2;

double side3;

bool isValid();

bool check();

};

bool Triangle::isValid() {

if (side1 + side2 > side3 && side2 + side3 > side1 && side1 + side3 > side2)

return true;

else

return false;

}

bool Triangle::check() {

if (side1 > 0 && side2 > 0 && side3 > 0)

return true;

else

return false;

}

void Triangle::setSide1(double s1) {

side1 = s1;

}

void Triangle::setSide2(double s2) {

side2 = s2;

}

void Triangle::setSide3(double s3) {

side3 = s3;

}

double Triangle::getSide1(){

return side1;

}

double Triangle::getSide2(){

return side2;

}

double Triangle::getSide3(){

return side3;

}

double Triangle::computeArea() {

double p;

p = 0.5 \* (side1 + side2 + side3);

return sqrt(p \* (p - side1) \* (p - side2) \* (p - side3));

}

double Triangle::computePerimeter() {

return side1 + side2 + side3;

}

题目得分 10

【描述】

声明并实现一个Loan类，表示贷款。Loan类包括：

double类型的私有数据成员loanAmount，表示贷款额。double类型的私有数据成员annualInterestRate，表示贷款年利率。int类型的私有数据成员numberOfYears，表示贷款年限。

有参构造函数，将贷款额、贷款年利率、贷款年限设置为给定的参数。

更改器函数setLoanAmount、setAnnualInterestRate和setNumberOfYears，分别用于修改贷款额、贷款年利率、贷款年限。

访问器函数getLoanAmount、getAnnualInterestRate和getNumberOfYears，分别用于访问贷款额、贷款年利率、贷款年限。

成员函数getMonthlyPayment，返回月还款额。

成员函数getTotalPayment，返回总还款额。

【输入】

输入贷款额、贷款年利率、贷款年限。

【输出】

月还款额和总还款额。

【输入示例】

60000

6.25

15

【输出示例】

514.454

92601.7

【提示】

假设年利率为6.25%，则以6.25作为输入值。

计算月还款额的公式如下：

【来源】

《程序设计基础——以C++为例》第5章实验5。

（10分）

我的答案：

class Loan {

public:

Loan(double loan, double rate, int years) {

loanAmount = loan;

annualInterestRate = rate;

numberOfYears = years;

}

void setLoanAmount(double loan);

void setAnnualInterestRate(double rate);

void setNumberOfYears(int years);

double getLoanAmount();

double getAnnualInterestRate();

int getNumberOfYears();

double getMonthlyPayment();

double getTotalPayment();

private:

double loanAmount;

double annualInterestRate;

int numberOfYears;

};

void Loan::setLoanAmount(double loan) {

loanAmount = loan;

}

void Loan::setAnnualInterestRate(double rate) {

annualInterestRate = rate;

}

void Loan::setNumberOfYears(int years) {

numberOfYears = years;

}

double Loan::getLoanAmount() {

return loanAmount;

}

double Loan::getAnnualInterestRate() {

return annualInterestRate;

}

int Loan::getNumberOfYears() {

return numberOfYears;

}

double Loan::getMonthlyPayment() {

return (loanAmount \* annualInterestRate / 1200) / (1 - 1 / pow(1 + annualInterestRate / 1200, numberOfYears \* 12));

}

double Loan::getTotalPayment() {

return getMonthlyPayment() \* numberOfYears \* 12;

}

题目得分 10