**实验3：类与对象初步**

**姓名\_\_徐佳辉\_\_班级\_\_计科1902\_\_学号\_\_201906080621\_\_**

* **请阅读此说明：实验3满分110分；其中10分为附加，可选做；做完实验后请按要求将代码和截图贴入该文档。然后将此文档、源代码文件（.hpp, .cpp）打包上传到学习通。**

1. **停车场计费系统的升级思考：（60分+10分）**

**以课堂材料给出的停车场系统为基础考虑(见附件)，如果 停车收费规则改为：**

* **1小时内免费；**
* **1小时以上：**

**超过15分钟不到30分钟按半小时收费；**

**超过30分钟不到1小时按1小时收费；**

**超过1小时按每小时4元收费；**

* **超过1天，每天按30元收费；停车超过1个月的，停车费打9折；超过1年的停车费打8折。**
* **实验要求：**

**❶考虑修改Time的设计，改为DateTime，补充数据成员年、月、日，补充或者调整类内成员函数的设计，并实现它。采用你实现的新的DateTime进行停车收费的测试。并将完整的源代码和测试截图黏贴在下面。（30分）**

**源代码只需要黏贴类定义部分即可（DateTime的声明和实现）**

**❷考虑另一种程序调整模型，补充Date类，数据成员部分为：年、月、日；成员函数部分可以仿照原来Time类的功能设计。采用补充的新类Date和原来的Time类一起工作完成停车收费程序。（30分）**

**\*并思考：在新的收费规则下，两种程序调整的策略差异导致调整的工作量不同，你更倾向哪一种？给出你的理由（附加10分）**

**❶**

* **源代码粘贴处：**

**DateTime.hpp**

//DateTime.hpp

class DateTime{

public:

void showTime();

void setTime();

double diff(const DateTime& T);

private:

long normalize() const;

long countdays() const;

int year;

int mon;

int day;

int hour;

int minute;

int second;

};

**DateTime.cpp**

//DateTime.cpp

#include "Time.hpp"

#include <iostream>

using namespace std;

void DateTime::showTime()

{ cout<<year<<":"<<mon<<":"<<day<<":"<<hour<<":"<<minute<<":"<<second;

cout<<endl;

}

void DateTime::setTime()

{ do{

cin>>year>>mon>>day>>hour>>minute>>second;

}while(mon<1||mon>12||day<1||day>31||hour<0||hour>24||minute<0||minute>59||second<0||second>59);

}

double DateTime::diff(const DateTime& T)

{ long d=normalize()-T.normalize()-60;

if(d<0) return 0;

double h=d/60,m=d%60;

if(m<15) return h;

if(m>=15&&m<30) return h+0.5;

if(m>=30&&m<60) return h+1;

}

long DateTime::normalize() const{ //天数转分钟

int d = countdays();

return (hour + d \* 24)\*60+minute;

}

long DateTime::countdays() const{ //统计天数

long sum = 0;

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

for (int i = 2000; i < year; i++){ //年份转天数，从2000年开始计数

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

sum += 366;

}

else{

sum += 365;

}

}

for (int i = 1; i < mon; i++){ //月份转天数

if (i == 2 && ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0)){

sum += 29;

}

else{

sum += days[i-1];

}

}

sum += day;

return sum;

}

**parking.cpp**

#include "Time.hpp"

#include <iostream>

using namespace std;

int main()

{

DateTime arriveTime,leaveTime;

double parkingTime,Fee;

arriveTime.setTime();

arriveTime.showTime();

leaveTime.setTime();

leaveTime.showTime();

parkingTime=leaveTime.diff(arriveTime);

cout << "parkingtime:" << parkingTime << "(h)" << endl;

getchar();

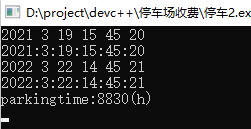
getchar();

return 0;

}

* **程序测试截图：**





**❷**

* **源代码粘贴处：**

**Date.hpp**

class Date{

public:

void showDate();

void setDate();

long normalize(); //年份月份转天数

private:

int year;

int mon;

int day;

};

**Date.cpp**

#include "Date.hpp"

#include <iostream>

using namespace std;

void Date::showDate()

{ cout<<year<<":"<<mon<<":"<<day<<":";

}

void Date::setDate()

{ do{

cin>>year>>mon>>day;

}while(day<1||day>31||mon<1||mon>12);

}

long Date::normalize() { //确定天数

long sum = 0;

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

for (int i = 2000; i < year; i++){ //年份转天数，从2000年开始计数

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

sum += 366;

}

else{

sum += 365;

}

}

for (int i = 1; i < mon; i++){ //月份转天数

if (i == 2 && ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0)){

sum += 29;

}

else{

sum += days[i-1];

}

}

sum += day;

return sum;

}

**Time.hpp**

class Date;

class Time{

public:

void showTime();

void setTime();

double diff(Time& T);

~Time(){

delete m\_date;

}

private:

long normalize();

class Date \*m\_date;

int hour;

int minute;

int second;

};

**Time.cpp**

#include "Date.hpp"

#include "Time.hpp"

#include <iostream>

using namespace std;

void Time::showTime()

{ m\_date->showDate();

cout<<hour<<":"<<minute<<":"<<second;

cout<<endl;

}

void Time::setTime()

{ do{

m\_date = new Date;

m\_date->setDate();

cin>>hour>>minute>>second;

}while(hour<0||hour>24||minute<0||minute>59||second<0||second>59);

}

double Time::diff(Time& T)

{

long d=normalize()-T.normalize()-60;

if(d<0) return 0;

double h=d/60,m=d%60;

if(m<15) return h;

if(m>=15&&m<30) return h+0.5;

if(m>=30&&m<60) return h+1;

}

long Time::normalize() //天数转分钟

{

return (hour + m\_date->normalize() \* 24)\*60+minute;

}

**Parking.cpp**

#include "Date.hpp"

#include "Time.hpp"

#include <iostream>

using namespace std;

int main()

{

Time arriveTime,leaveTime;

double parkingTime;

arriveTime.setTime();

arriveTime.showTime();

leaveTime.setTime();

leaveTime.showTime();

parkingTime = leaveTime.diff(arriveTime);

cout << "parkingtime:" << parkingTime << "(h)" << endl;

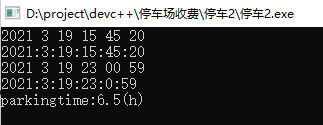
getchar();

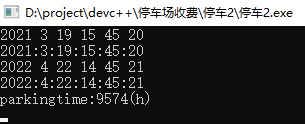
getchar();

return 0;

}

* **程序测试截图：**





**\*附加思考：**

**第二种**

**2、位置类position：（40分）**

**设计并实现一个平面坐标系内的位置类position。包含的基本数据成员有：横坐标，纵坐标；包含的基本成员函数有：设置位置；读取位置；判断第几象限；计算到源点的距离；计算到其他点的距离；计算经过源点到这个位置的直线的斜率；计算经过这个位置到其他点的直线的斜率；按坐标轴平移位置。其他成员函数功能可以自行补充。**

* **实验要求：**

**❶ 按照描述完成position类的基本的设计和实现。将数据成员设计为私有(private)成员；将成员函数设计为公有(public)成员。并通过以下测试程序。**

#include “position.hpp”

#include <iostream>

using namespace std;

int main() {

position a, b, c,d,e;

a.set(5, 15);

a.show();

b.set(-4.5, 6.7);

b.show();

c.set(-10, -100);

c.show();

d.set(20.5, 5.5);

e.set();//默认为原点

e.show();

cout<<distance(a, b)<<endl;

cout<<distance(c)<<endl;//默认求与原点的距离

cout<<a.slope()<<endl;//与原点构成直线的斜率

cout<<a.slope(d)<<endl; //与d构成直线的斜率

a.move(3);//沿x轴平移

a.show();

b.move(-4, 5);

b.show();

c.move(0, 6);//沿y轴平移

c.show();

return 0;

}

* **实验提交：**

**将完整的源代码和测试截图 粘贴在下面。**

* **源代码粘贴处：**

**Position.hpp**

class position{

public:

void set(double x = 0, double y = 0);

void show();

void move(double x, double y);

void move(double x);

double slope(position& p);

double slope();

friend double Distance(const position &p1, const position &p2);

friend double Distance(const position &p);

private:

double m\_x;

double m\_y;

};

**Position.cpp**

#include "position.hpp"

#include <iostream>

#include <math.h>

using namespace std;

void position::set(double x = 0, double y = 0){

m\_x = x;

m\_y = y;

}

void position::show(){

cout << m\_x << ' ' << m\_y << endl;

}

void position::move(double x, double y){

m\_x += x;

m\_y += y;

}

void position::move(double x){

m\_x += x;

}

double position::slope(position& p){

double slope = 0;

return slope = (m\_y - p.m\_y) / (m\_x - p.m\_x);

}

double position::slope(){

double slope = 0;

return slope = m\_y / m\_x;

}

**Main.cpp**

#include "position.hpp"

#include <iostream>

#include <math.h>

using namespace std;

double Distance(const position &p1, const position &p2){

return sqrt(pow(p1.m\_x - p2.m\_x, 2) + pow(p1.m\_y - p2.m\_y, 2));

}

double Distance(const position &p){

return sqrt(pow(p.m\_x, 2) + pow(p.m\_y, 2));

}

int main() {

position a, b, c, d, e;

a.set(5, 15);

a.show();

b.set(-4.5, 6.7);

b.show();

c.set(-10, -100);

c.show();

d.set(20.5, 5.5);

e.set();//默认为原点

e.show();

cout<<Distance(a, b)<<endl;

cout<<Distance(c)<<endl;//默认求与原点的距离

cout<<a.slope()<<endl;//与原点构成直线的斜率

cout<<a.slope(d)<<endl; //与d构成直线的斜率

a.move(3);//沿x轴平移

a.show();

b.move(-4, 5);

b.show();

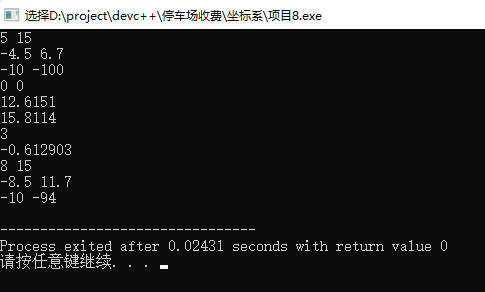
c.move(0, 6);//沿y轴平移

c.show();

return 0;

}

* **程序测试截图：**



* **拓展源代码粘贴处（可选）：**