浙江工业大学

面向对象程序设计

2022/2023(2)

实验二　字符串学习和类与对象初步

学生姓名　　陈王子

学生学号　　202103150503

学生班级　　大数据分析 1 班

任课教师　　毛国红

提交日期　　2023 年 4 月 5 日

理学院

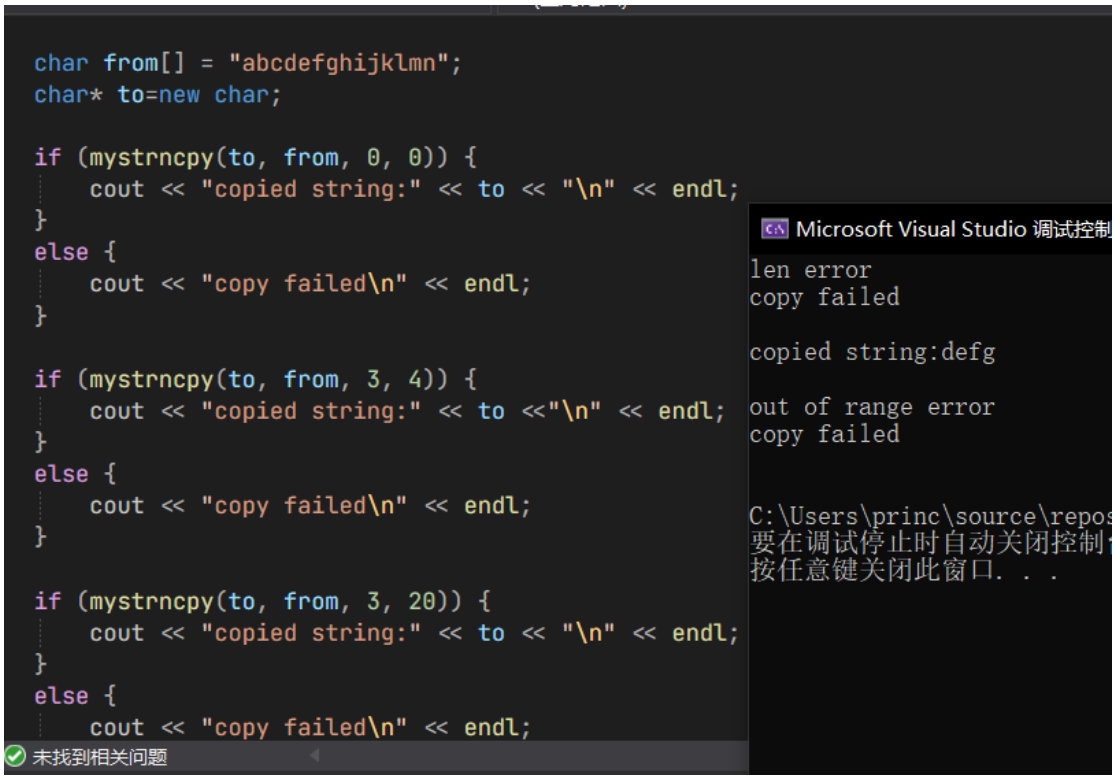
实验报告

第一题代码：拷贝字符串 n 个字符

```
#include <iostream>
#include <cstring>
using namespace std;

bool mystrncpy(char* to, char* from, unsigned startpos, unsigned
len) {
    if (startpos >= strlen(from)) {
        cerr << "pos error" << endl;//位置错误
        return false;
    }
    if (len <= 0) {
        cerr << "len error" << endl;//长度错误
        return false;
    }
    if (startpos + len > strlen(from)) {
        cerr << "out of range error" << endl;//过界错误
        return false;
    }
    //拷贝函数
    for (unsigned i = startpos; i < startpos + len; i++) {
        to[i - startpos] = from[i];
    }
    to[len] = '\0';
    return true;
}

int main()
{
    char from[] = "abcdefghijklmn";
    char* to=new char;

    if (mystrncpy(to, from, 0, 0)) {
        cout << "copied string:" << to << "\n" << endl;
    }
    else {
        cout << "copy failed\n" << endl;
    }

    if (mystrncpy(to, from, 3, 4)) {
```

```
        cout << "copied string:" << to <<"\n" << endl;
    }
    else {
        cout << "copy failed\n" << endl;
    }

    if (mystrncpy(to, from, 3, 20)) {
        cout << "copied string:" << to << "\n" << endl;
    }
    else {
        cout << "copy failed\n" << endl;
    }
}
```

第一题测试



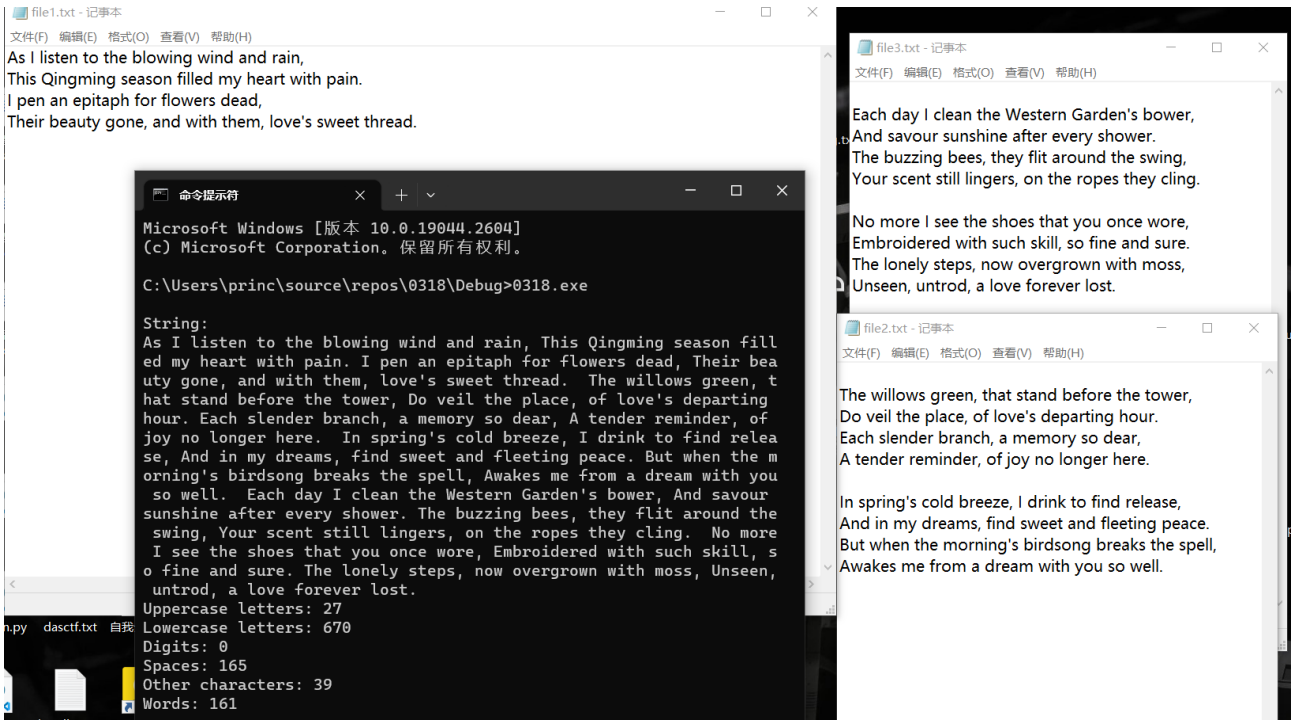
第二题代码：统计字符个数和单词数附加处理文章

```
#include <iostream>
#include <string>
#include <fstream>

using namespace std;

void count_chars(const string& s, int& upper, int& lower, int&
digit, int& space, int& other, int& words) {
    upper = lower = digit = space = other = words = 0;
    bool in_word = false;
    for (char c : s) {
        if (isupper(c)) {
            upper++;
        }
        else if (islower(c)) {
            lower++;
        }
        else if (isdigit(c)) {
            digit++;
        }
        else if (isspace(c)) {
            space++;
            in_word = false;
        }
        else {
            other++;
        }

        if (isspace(c)) {
            continue;
        }

        if (!in_word) {
            words++;
            in_word = true;
        }
    }
}

string readFilesToString(string text,const string& file)
{
    ifstream infile1(file);
```

```
    if (infile1) {
        string line;
        while (getline(infile1, line)) {
            text += line + " ";
        }
    }
    else {
        cerr << "Failed to open file " << file << endl;
        return "";
    }
    return text;
}

int main() {
    string article;
    /*
    单行读入版本
    cout << "Enter an article: "<<endl;
    getline(cin, article);
    */

    //多文件读入版本
    article = readFilesToString(article,
"C:\\Users\\princ\\source\\repos\\0318\\Debug\\file1.txt");
    article = readFilesToString(article,
"C:\\Users\\princ\\source\\repos\\0318\\Debug\\file2.txt");
    article = readFilesToString(article,
"C:\\Users\\princ\\source\\repos\\0318\\Debug\\file3.txt");

    cout << "\nString:\n" <<article<< endl;

    int upper, lower, digit, space, other, words;
    count_chars(article, upper, lower, digit, space, other,
words);

    cout << "Uppercase letters: " << upper << endl;
    cout << "Lowercase letters: " << lower << endl;
    cout << "Digits: " << digit << endl;
    cout << "Spaces: " << space << endl;
    cout << "Other characters: " << other << endl;
    cout << "Words: " << words << endl;
}
```

第二题测试（附加读入文本和处理多篇文章）

```
C:\Users\princ\source\repos\0318\Debug>0318.exe
Enter an article:
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch o
f incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before
us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way-in short, the period was so. far like the present p
eriod, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.114514
Uppercase letters: 4
Lowercase letters: 471
Digits: 6
Spaces: 117
Other characters: 20
Words: 118
```



第三题大整数算术（以及随机数附加题）

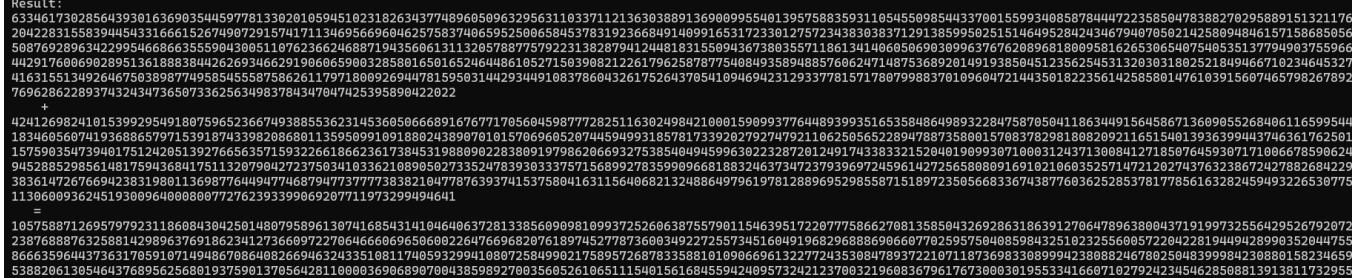
```cpp
#include <iostream>
#include <string>
#include <algorithm>
#include <ctime>
#include <cstdlib>
#include <random>
using namespace std;

// 加法运算
string add(string num1, string num2) {
    int len1 = num1.size(), len2 = num2.size();
    int carry = 0, sum;
    string res;
    while (len1 || len2) {
        int x = len1 ? num1[--len1] - '0' : 0;
        int y = len2 ? num2[--len2] - '0' : 0;
        sum = x + y + carry;
        carry = sum / 10;
        res += (sum % 10) + '0';
    }
    if (carry) res += carry + '0';
    reverse(res.begin(), res.end());
    return res;
}

// 减法运算
string sub(string num1, string num2) {
    int len1 = num1.size(), len2 = num2.size();
    int borrow = 0, diff;
    string res;
    while (len1 || len2) {
        int x = len1 ? num1[--len1] - '0' : 0;
        int y = len2 ? num2[--len2] - '0' : 0;
        diff = x - y - borrow;
        if (diff < 0) {
            diff += 10;
            borrow = 1;
        }
        else {
            borrow = 0;
        }
        res += diff + '0';
```

```
    }
    reverse(res.begin(), res.end());
    while (res.size() > 1 && res[0] == '0') res.erase(0, 1);
    return res;
}

// 乘法运算
string mul(string num1, string num2) {
    int len1 = num1.size(), len2 = num2.size();
    string res(len1 + len2, '0');
    for (int i = len1 - 1; i >= 0; i--) {
        int carry = 0;
        for (int j = len2 - 1; j >= 0; j--) {
            int tmp = (num1[i] - '0') * (num2[j] - '0') + carry
+ (res[i + j + 1] - '0');
            res[i + j + 1] = tmp % 10 + '0';
            carry = tmp / 10;
        }
        res[i] += carry;
    }
    while (res.size() > 1 && res[0] == '0') res.erase(0, 1);
    return res;
}

// 处理输入的字符串，得到两个数和运算符
void parseInput(string input, string& num1, string& num2, char&
op) {
    int pos = input.find(',');
    num1 = input.substr(0, pos);
    input = input.substr(pos + 1);
    pos = input.find(',');
    num2 = input.substr(0, pos);
    op = input[pos + 1];
}

// 生成一个指定位数的随机大数
// 生成指定位数的随机大数
string genRandNum(int numDigits) {
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<> dis(0, 9);
    string randNum;
    randNum.reserve(numDigits);
    for (int i = 0; i < numDigits; i++) {
```

```
        randNum += std::to_string(dis(gen));
    }
    return randNum;
}


// 检查输入是否合法
bool isValidInput(const string& input) {
    // 检查输入长度
    if (input.size() < 3) {
        return false;
    }
    // 检查逗号个数
    int commaCount = count(input.begin(), input.end(), ',');
    if (commaCount != 2) {
        return false;
    }
    // 检查运算符合法性
    char op = input[input.size() - 1];
    if (op != '+' && op != '-' && op != '*') {
        return false;
    }
    // 检查数字合法性
    for (int i = 0; i < input.size() - 2; i++) {
        if (!isdigit(input[i]) && input[i] != ',') {
            return false;
        }
    }
    return true;
}

int main() {
    srand(time(NULL)); // 随机数种子

    while (cout << "\nPlease enter your equation:" << endl) {
        string input;
        string num1, num2;
        char op = { 0 };
        getline(cin,input); // 从用户输入中解析出两个操作数和运算符
        if (input == "exit") break; // 用户输入 exit 时退出程序

        if (isValidInput(input)) { // 判断输入是否合法
            parseInput(input, num1, num2, op); // 将用户输入解析
为数字和运算符
```

```
            if (op == '+') {
                string res = add(num1, num2); // 调用加法函数
                cout << num1 << " + " << num2 << " = " << res <<
endl;
            }
            else if (op == '-') {
                string res = sub(num1, num2); // 调用减法函数
                cout << num1 << " - " << num2 << " = " << res <<
endl;
            }
            else if (op == '*') {
                string res = mul(num1, num2); // 调用乘法函数
                cout << num1 << " * " << num2 << " = " << res <<
endl;
            }
        }
        else {
            cout << "Invalid input. Please try again." << endl;
        }
    }

    // 生成随机大数并计算
    int digits = 1000;
    string num1 = genRandNum(digits);
    string num2 = genRandNum(digits);
    string res = add(num1, num2); // 计算 num1 * num2
    cout << "Random number digits:"<< digits <<"\nResult: \n"
<< num1 << "\n    +    \n" << num2 << "\n   =   \n" << res << endl;
}
```

第三题测试（上百位整数和简单计算）



第三题思考题：

1. 基本存储类型的选择：选择了 string，因为内置函数比较多，方便进行选择切片等处理。

2. 理论极限：处理数字的范围应该是 0 到无穷大（由用户内存决定）

第四题：游泳池和停车场

1. 游泳池

```
//pool.hpp
#include <iostream>
using namespace std;
class pool {
public:
    void build();
    double rail_length();
    double rail_area();
private:
    double radius;
    double c;
};
//pool.cpp
#define pi 3.14
//#include "pool.hpp"
void pool::build()
{
    cin >> radius >> c;
}
double pool::rail_area()
{
    return pi * ((radius + c) * (radius + c)- radius * radius);
}
double pool::rail_length()
{
    return 2 * pi * radius;
}
//pooltest.cpp
int main()
{
    const double zddj = 167.5;//走道单价
    const double wldj = 36.4;//围栏单价
    double r;//池直径
    double c;//走道宽
    pool apool;
    for (int I = 1; I <= 5; I++)
    {
        cout << "建造第"<< I << "号泳池:";
        apool.build();
        cout << "第"<< I << "号泳池造价为"
```

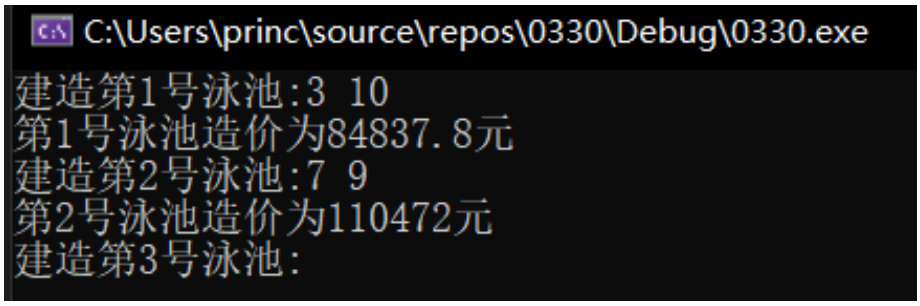
```
            << (apool.rail_length() * wldj + apool.rail_area()
* zddj) << "元" << endl;
    }
}
```

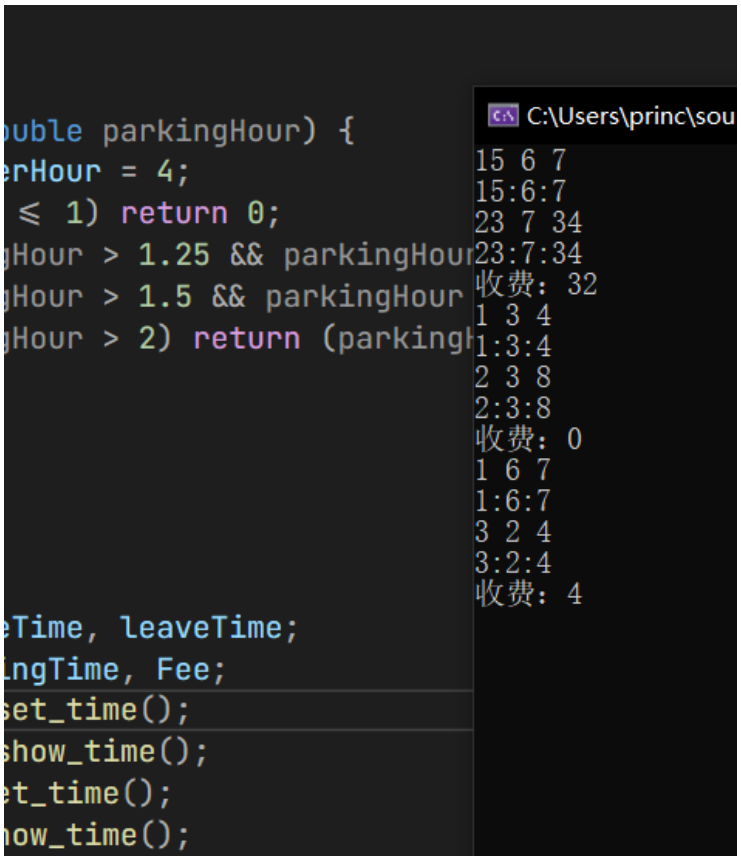
2. 停车场

```
#include <iostream>
using namespace std;

//clock.hpp
class Clock {
public:
    void show_time();
    void set_time();
    double diff(Clock& T);
private:
    long normalize();
    int hour;
    int minute;
    int second;
};
//clock.cpp
void Clock::show_time()
{
    cout << hour << ":" << minute << ":"
        << second;
    cout << endl;
}

void Clock::set_time()
{
    do {
        cin >> hour >> minute >> second;
    } while (hour < 0 || hour>24 ||
        minute < 0 || minute>59 ||
        second < 0 || second>59);
}

double Clock::diff(Clock& T)
{
    long d = this->normalize() - T.normalize();
```

```
    if (d < 0) return 0;
    int 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 Clock::normalize()
{
    return (this->hour * 60 + this->minute);
}

//parkingcost.cpp
double parkingFee(double parkingHour) {
    int parkingFeePerHour = 4;
    if (parkingHour <= 1) return 0;
    else if (parkingHour > 1.25 && parkingHour <= 1.5) return
0.5 * parkingFeePerHour;
    else if (parkingHour > 1.5 && parkingHour <= 2) return
parkingFeePerHour;
    else if (parkingHour > 2) return (parkingHour) *
parkingFeePerHour;
}


int main()
{
    while (1) {
        Clock arriveTime, leaveTime;
        double parkingTime, Fee;
        arriveTime.set_time();
        arriveTime.show_time();
        leaveTime.set_time();
        leaveTime.show_time();
        parkingTime = leaveTime.diff(arriveTime);
        Fee = parkingFee(parkingTime);
        cout << "收费: " << Fee << endl;
    }
}
```

第四题测试：





第五题：停车场改造 Date 类和 DateTime 类

1. DateTime 类

```
class DateTime {
public:
    void show_time();
    void set_time();
    int days_in_month(int month, int year);
```

```
    bool is_leap_year(int year);
    double diff(DateTime& T);
private:
    long normalize();
    int year;
    int month;
    int day;
    int hour;
    int minute;
    int second;
};

void DateTime::show_time() {
    cout << year << "-" << month << "-" << day << " " << hour
<< ":" << minute << ":" << second << endl;
}

void DateTime::set_time() {
    cout << "Enter year: ";
    cin >> year;
    cout << "Enter month: ";
    cin >> month;
    cout << "Enter day: ";
    cin >> day;
    cout << "Enter hour: ";
    cin >> hour;
    cout << "Enter minute: ";
    cin >> minute;
    cout << "Enter second: ";
    cin >> second;
}

double DateTime::diff(DateTime& T) {
    long seconds1 = normalize();
    long seconds2 = T.normalize();
    return (double)abs((seconds2 - seconds1) / (24 * 3600));
}

long DateTime::normalize() {
    // 计算从公元 0 年 0 月 0 日起至当前日期的天数
    long days = 0;
    for (int i = 0; i < year; i++) {
        days += is_leap_year(i) ? 366 : 365;
    }
```

```
    for (int i = 1; i < month; i++) {
        days += days_in_month(i, year);
    }
    days += day - 1;

    // 计算总秒数
    long seconds = days * 24 * 3600;
    seconds += hour * 3600;
    seconds += minute * 60;
    seconds += second;
    return seconds;
}

int DateTime::days_in_month(int month, int year) {
    if (month == 2) {
        if (is_leap_year(year)) {
            return 29;
        }
        else {
            return 28;
        }
    }
    else if (month == 4 || month == 6 || month == 9 || month ==
11) {
        return 30;
    }
    else {
        return 31;
    }
}

bool DateTime::is_leap_year(int year) {
    if (year % 4 == 0 && year % 100 != 0 || year % 400 == 0) {
        return true;
    }
    else {
        return false;
    }
}



//parkingcost.cpp
```

```
double parkingFee(double parkingHour) {
    int parkingFeePerHour = 4;
    int parkingFeePerDay = 30;
    double parkingFee = 0;

    if (parkingHour <= 1) {
        parkingFee = 0;
    }
    else if (parkingHour > 1.25 && parkingHour <= 1.5) {
        parkingFee = 0.5 * parkingFeePerHour;
    }
    else if (parkingHour > 1.5 && parkingHour <= 2) {
        parkingFee = parkingFeePerHour;
    }
    else if (parkingHour > 2 && parkingHour <= 24) {
        parkingFee = parkingHour * parkingFeePerHour;
    }
    else if (parkingHour <= 720) {  // 超过1个月，按30元/天计费，
打9折
        parkingFee = (int)parkingHour / 24 * parkingFeePerDay
* 0.9;
        if ((int)parkingHour % 24 > 2) {
            parkingFee += parkingFeePerDay * 0.9;
        }
        else if ((int)parkingHour % 24 > 0) {
            parkingFee += parkingFeePerDay * 0.5;
        }
    }
    else if (parkingHour <= 8760) {  // 超过1年，按30元/天计费，
打8折
        parkingFee = (int)parkingHour / 24 * parkingFeePerDay
* 0.8;
        if ((int)parkingHour % 24 > 2) {
            parkingFee += parkingFeePerDay * 0.8;
        }
        else if ((int)parkingHour % 24 > 0) {
            parkingFee += parkingFeePerDay * 0.5;
        }
    }
    else {  // 超过1年按30元/天计费，打8折
        parkingFee = (int)(parkingHour / 24 / 365 *
parkingFeePerDay * 365 * 0.8);
        int days = (int)parkingHour / 24 % 365;
        if (days > 30) {
```

```
            parkingFee += parkingFeePerDay * 0.8 * 30;
        }
        else {
            parkingFee += parkingFeePerDay * 0.8 * days;
        }
    }

    return parkingFee;
}
```

2. Date 类

```
#include <iostream>
using namespace std;

class Date {
public:
    void show_date();
    void set_date();
    int days_in_month();
    void add_day();
    double diff(Date& T);
private:
    int year;
    int month;
    int day;
};

class Clock {
public:
    void show_time();
    void set_time();
    double diff(Clock& T);
private:
    long normalize();
    int hour;
    int minute;
    int second;
};

void Date::show_date() {
    cout << year << "-" << month << "-" << day << endl;
}
```

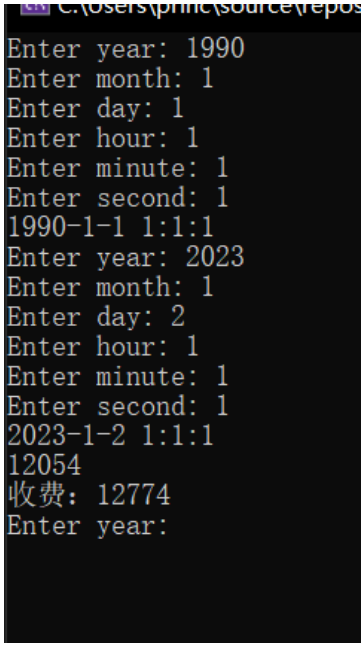
```
void Date::set_date() {
    cout << "Enter year: ";
    cin >> year;
    cout << "Enter month: ";
    cin >> month;
    cout << "Enter day: ";
    cin >> day;
}

int Date::days_in_month() {
    if (month == 2) {
        if ((year % 4 == 0 && year % 100 != 0) || year % 400 ==
0) {
            return 29;
        }
        else {
            return 28;
        }
    }
    else if (month == 4 || month == 6 || month == 9 || month ==
11) {
        return 30;
    }
    else {
        return 31;
    }
}

void Date::add_day() {
    day++;
    if (day > days_in_month()) {
        day = 1;
        month++;
        if (month > 12) {
            month = 1;
            year++;
        }
    }
}

double Date::diff(Date& T) {
    int days1 = 0;
    for (int i = 1; i < month; i++) {
```

```
        Date d = { year, i, 1 };
        days1 += d.days_in_month();
    }
    days1 += day - 1;
    int days2 = 0;
    for (int i = 1; i < T.month; i++) {
        Date d = { T.year, i, 1 };
        days2 += d.days_in_month();
    }
    days2 += T.day - 1;
    int days = 0;
    if (year == T.year) {
        days = days2 - days1;
    }
    else {
        for (int i = year; i < T.year; i++) {
            Date d = { i, 1, 1 };
            days += d.days_in_month();
        }
        days -= days1;
        days += days2;
    }
    return (double)days;
}

void Clock::show_time() {
    cout << hour << ":" << minute << ":" << second << endl;
}

void Clock::set_time() {
    cout << "Enter hour: ";
    cin >> hour;
    cout << "Enter minute: ";
    cin >> minute;
    cout << "Enter second: ";
    cin >> second;
}

double Clock::diff(Clock& T) {
    long seconds1 = normalize();
    long seconds2 = T.normalize();
    return (double)(seconds2 - seconds1) / (24 * 3600);
}
```

```
long Clock::normalize() {
    long seconds = hour * 3600;
    seconds += minute * 60;
    seconds += second;
    return seconds;
}

double parkingFee(Date& d1, Clock& t1, Date& d2, Clock& t2) {
    int parkingFeePerHour = 4;
    int parkingFeePerDay = 30;
    double parkingFee = 0;

    double parkingTime = d1.diff(d2) + t1.diff(t2) / 3600.0;

    if (parkingTime <= 1) {
        parkingFee = 0;
    }
    else if (parkingTime <= 1.25) {
        parkingFee = 0.5 * parkingFeePerHour;
    }
    else if (parkingTime <= 2) {
        parkingFee = parkingFeePerHour;
    }
    else if (parkingTime <= 24) {
        parkingFee = parkingTime * parkingFeePerHour;
    }
    else if (parkingTime <= 720) {  // 超过1个月，按30元/天计费，
打9折
        parkingFee = d1.diff(d2) * parkingFeePerDay * 0.9;
        d1.add_day();
        while (d1.diff(d2) > 0) {
            parkingFee += parkingFeePerDay * 0.9;
            d1.add_day();
        }
        if (t2.diff(t1) / 3600.0 > 2) {
            parkingFee += parkingFeePerDay * 0.9;
        }
        else if (t2.diff(t1) / 3600.0 > 0) {
            parkingFee += parkingFeePerDay * 0.5;
        }
    }
    else if (parkingTime <= 8760) {  // 超过1年，按30元/天计费，
打8折
        parkingFee = d1.diff(d2) * parkingFeePerDay * 0.8;
```

```
        d1.add_day();
        while (d1.diff(d2) > 0) {
            parkingFee += parkingFeePerDay * 0.8;
            d1.add_day();
        }
        if (t2.diff(t1) / 3600.0 > 2) {
            parkingFee += parkingFeePerDay * 0.8;
        }
        else if (t2.diff(t1) / 3600.0 > 0) {
            parkingFee += parkingFeePerDay * 0.5;
        }
    }
    else {  // 超过 1 年按 30 元/天计费，打 8 折
        parkingFee = d1.diff(d2) * parkingFeePerDay * 365 * 0.8;
        d1.set_date();
        while (d1.diff(d2) > 0) {
            if (d1.diff(d2) < 30) {
                parkingFee += d1.diff(d2) * parkingFeePerDay *
0.8;
                break;
            }
            else {
                parkingFee += parkingFeePerDay * 0.8 * 30;
                d1.add_day();
            }
        }
        if (t2.diff(t1) / 3600.0 > 2) {
            parkingFee += parkingFeePerDay * 0.8;
        }
        else if (t2.diff(t1) / 3600.0 > 0) {
            parkingFee += parkingFeePerDay * 0.5;
        }
    }

    return parkingFee;
}
```

第五题测试结果：



第五题倾向分析：

我更倾向于方案一，即修改 Time 的设计成为 DateTime，并增加年、月、日等数据成员和相应的类内成员函数。原因如下：

1. 代码可读性更高：将年、月、日等属性与时间相关的信息放在一个 DateTime 类中，在代码中调用时更加直观明了。
2. 代码复用率更高：从程序结构的角度来看，DateTime 类是 Time 类的一种扩展，这样两个类之间的部分代码可以共享，使得代码复用率更高，且能够减少代码的冗余。
3. 系统可拓展性更好：如果未来需求再次变更，需要增加支持星期、季节等更多时间相关信息的时候，基于 DateTime 类设计的系统代码更容易进行拓展。

在采用补充的新类 Date 和原来的 Time 类一起工作完成停车收费程序的情况下，方案一的工作量比方案二少，因为方案一只需要在原本的 Time 类的基础上增加一些属性和方法即可，而方案二需要新建一个全新的类，并独立开发所有必要的属性和方法，这就需要额外的工作量和时间成本。同时，如果未来需求进一步变化，现有的 Date 类可能还需要进行重构，这个过程会更加繁琐。因此，综合考虑程序的可读性、复用性和拓展性等多个因素，我更倾向于方案一，即采用 DateTime 类，增加年、月、日等数据成员和相应的类内成员函数。