面向对象的程序设计C++第 7 次实验

题目 1

定义一个复数（Complex）类，并重载运算符 + 、 - 、 * 、 / （加减乘除法）， 使得两个复数可以进行相应的运算。（注意：可能是复数加/减/乘/除上另一个复 数，也可能是复数加/减/乘/除上一个普通的实数）

代码思路

创建类Complex，内含两个double型变量real,imag分别代表复数的实部和虚部，重载四个操作符+-*/的方法，每个操作符有两种重载方式以支持其与复数和普通实数的运算。

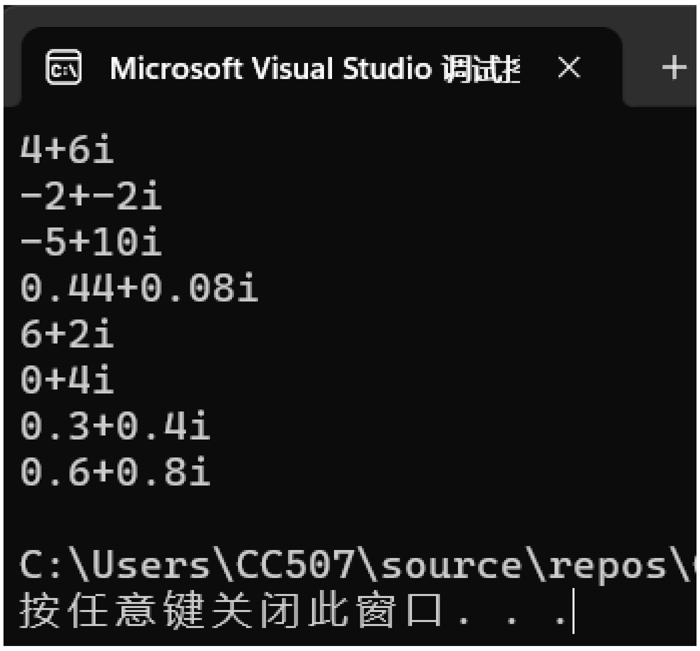
```cpp
class Complex {
private:
    double real = 0, imag = 0;
public:
    Complex(double real,double imag) {
        this->real = real;
        this->imag = imag;
    }
    friend ostream& operator<<(ostream& os, const Complex& c) {
        os << c.real << "+" << c.imag << "i";
        return os;
    }
    Complex operator+(Complex c) {
        return Complex(real + c.real, imag + c.imag);
    }
    Complex operator+(double d) {
        return Complex(real + d, imag);
    }
    Complex operator-(Complex c) {
        return Complex(real - c.real, imag - c.imag);
    }
    Complex operator-(double d) {
        return Complex(real - d, imag);
    }
    Complex operator*(Complex c) {
        return Complex(real * c.real - imag * c.imag, real * c.imag + imag * c.real);
    }
    Complex operator*(double d) {
        return Complex(real * d, imag * d);
    }
    Complex operator/(Complex c) {
        return Complex((real * c.real + imag * c.imag) / (c.real * c.real + c.imag * c.imag),
            (imag * c.real - real * c.imag) / (c.real * c.real + c.imag * c.imag));
    }
    Complex operator/(double d) {
        return Complex(real / d, imag / d);
    }
};
```

程序运行结果

main方法

```
int main() {
    Complex c1(1, 2), c2(3, 4);
    cout << c1 + c2 << endl;
    cout << c1 - c2 << endl;
    cout << c1 * c2 << endl;
    cout << c1 / c2 << endl;
    cout << c1 + 5 << endl;
    cout << c2 - 3 << endl;
    cout << c2 * 0.1 << endl;
    cout << c2 / 5 << endl;
}
```

运行结果



```
4+6i
-2+-2i
-5+10i
0.44+0.08i
6+2i
0+4i
0.3+0.4i
0.6+0.8i

C:\Users\CC507\source\repos\
按任意键关闭此窗口. . .
```

题目2

定义一个交通工具（Vehicles）基类，包含 run、stop 成员函数，由此派生出 自行车(Bicycle)类、汽车(Car)类，从 Bicycle 和 Car 派生出摩托车(Motorcycle)类， 它们都有 run、stop 等成员函数，定义一个包含 7 个元素的 Vehicles 类型的数组， 在数组中存入 1 个 Vehicles 对象、1 个 Car 对象、3 个 Bicycle 对象、2 个 Motorcycle 对象，遍历此数组，观察虚函数的作用。

代码思路

将Vehicles的run,stop函数定义为虚函数，便于在子类中重写该方法并且子类实例化后优先调用子类的方法。

由于Motorcycle继承自Bicycle和Car，而Bicycle和Car都继承自同一个基类Vehicles会导致二义性问题，因此需要使用虚拟继承，将Vehicles定义为Bicycle和Car的虚基类。

```cpp
class Vehicles {
public:
    virtual void run() {
        cout << "Vehicle is running!" << endl;
    }
    virtual void stop() {
        cout << "Vehicle stopped!" << endl;
    }
};
class Bicycle :virtual public Vehicles {
    void run() {
        cout << "Bicycle is running!" << endl;
    }
    void stop() {
        cout << "Bicycle stopped!" << endl;
    }
};
class Car :virtual public Vehicles {
    void run() {
        cout << "Car is running!" << endl;
    }
    void stop() {
        cout << "Car stopped!" << endl;
    }
};
class Motorcycle :public Bicycle, public Car {
    void run() {
        cout << "Motorcycle is running!" << endl;
    }
    void stop() {
        cout << "Motorcycle stopped!" << endl;
    }
};
```

程序运行结果

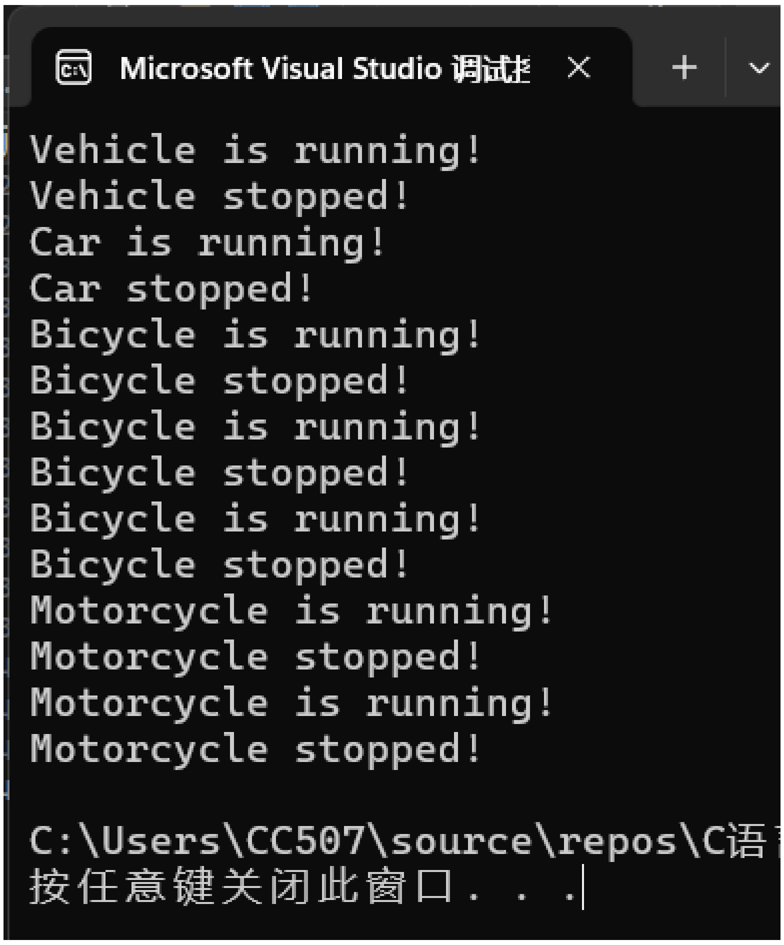
main()方法

将数组类型声明为指针类型才能动态绑定。

```
int main() {
    vector<Vehicles*> arr = { new Vehicles(),new Car(),new Bicycle(),new Bicycle(), new
Bicycle(), new Motorcycle(),new Motorcycle };
    for (int i = 0; i < arr.size(); i++) {
        arr[i]->run();
        arr[i]->stop();
    }
}
```

运行结果

各个成员能够正确调用各自的重写方法。



```
Vehicle is running!
Vehicle stopped!
Car is running!
Car stopped!
Bicycle is running!
Bicycle stopped!
Bicycle is running!
Bicycle stopped!
Bicycle is running!
Bicycle stopped!
Motorcycle is running!
Motorcycle stopped!
Motorcycle is running!
Motorcycle stopped!

C:\Users\CC507\source\repos\C语
按任意键关闭此窗口. . .
```

题目3

利用习题6.8的第14题中的时间类Time，定义一个带时区的时间类ExtTime。除了构造函数和时间调整函数外，ExtTime的其他功能与Time类似。

代码思路

使新类ExtTime继承自Time类，构造函数也继承自Time，可以省去大量代码的编写。

```
class Time
{
```

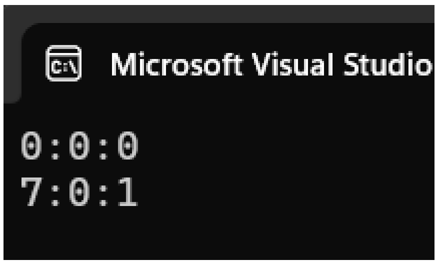
```
private:
    int hour, minute, second;

public:
    Time(int h, int m, int s) {
        hour = h;
        minute = m;
        second = s;
    }
    void set(int h, int m, int s) {
        hour = h;
        minute = m;
        second = s;
    }
    void increment() {
        int s = hour * 3600 + minute * 60 + second + 1;
        hour = s / 3600 % 24;
        minute = s % 3600 / 60;
        second = s % 60;
    }
    void display() {
        cout << hour << ":" << minute << ":" << second << endl;
    }
    bool equal(Time& other_time) {
        return (hour == other_time.hour) && (minute == other_time.minute) && (second ==
other_time.second);
    }
    bool less_than(Time& other_time) {
        return (hour * 3600 + minute * 60 + second + 1 < other_time.hour * 3600 +
other_time.minute * 60 + other_time.second);
    }
};
class ExtTime :public Time {
private:
    string timezone;
public:
    ExtTime(string tz, int h, int m, int s) :Time(h, m, s) {
        timezone = tz;
    }
};
```

程序运行结果

main()方法

```
int main() {
    Time t1(23, 59, 59);
    ExtTime t2("GMT+5", 7, 0, 0);
    t1.increment();
    t1.display();
    t2.increment();
    t2.display();
}
```

运行结果:



```
0:0:0
7:0:1
```

题目4

利用习题6.8的第18题中的LinearList类定义一个栈类

代码思路

对比线性表，栈只支持两种操作：入栈（push）和出栈（pop），因此只需继承线性表并实现这两种方法即可。

将线性表的表头作为栈顶，入栈则是在表头插入元素，出栈即删除表头元素。

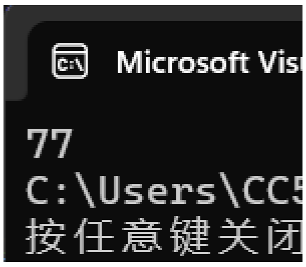
```
class LinearList {
private:
    struct Node
    {
        int value;
        Node* next;
    };
    int count;
    Node* head;
public:
    LinearList() { count = 0; head = NULL; }
    ~LinearList() {
        while (head != NULL) {
            Node* p = head;
            head = head->next;
            delete p;
        }
        count = 0;
    }
    bool insert(int x, int pos) {
        if (pos > count || pos < 0) {
            return false;
        }
        Node* q = new Node;
        q->value = x;
        q->next = NULL;
        if (pos == 0) {
            q->next = head;
            head = q;
        }
        else {
            Node* p = head;
            for (int i = 1; i < pos; i++) p = p->next;
            q->next = p->next;
            p->next = q;
        }
        count++;
        return true;
    }
    bool remove(int& x, int pos) {
        if (pos > count || pos <= 0)return false;
        Node* p = head;
        if (pos == 1) {
            head = head->next;
            x = p->value;
            delete p;
        }
        else {
            for (int i = 2; i < pos; i++)p = p->next;
            Node* temp = p->next;
            p->next = temp->next;
            x = temp->value;
            delete temp;
        }
        count--;
        return true;
    }
    int element(int pos)const {
        if (pos > 0 && pos <= count) {
            Node* p = head;
            for (int i = 1; i < pos; i++)p = p->next;
            return p->value;
        }
        return -1;
    }
    int search(int x)const {
        int pos = 0;
        Node* p = head;
        while (p != NULL) {
            pos++;
            if (p->value == x)return pos;
            p = p->next;
        }
        return 0;
    }
    int length()const {
        return count;
    }
};
class Stack :public LinearList {
public:
    bool push(int x) {
        return insert(x, 0);
    }
    bool pop(int& x) {
        return remove(x, 1);
    }
    LinearList::length;
};
```

程序运行结果

main()方法

```
int main() {
    Stack s;
    s.push(77);
    int x = 0;
    s.pop(x);
    cout << x;
}
```

运行结果



```
Microsoft Vis
77
C:\Users\CC5
按任意键关闭
```

题目5

利用习题6.8的第14、15题中的时间类Time和日期类Date，定义一个带日期的时间类TimeWithDate。对该类对象能进行比较、增加(增加值为秒数)、相减(结果为秒数)等操作。

代码思路：

(1)比较less_than

调用Date中的less_than先对日期进行比较，如果小于则返回，再调用Date中的equal判断日期是否相同，如果是则调用Time中的less_than进行比较。

```
bool less_than(const TimeWithDate& td2)const {
    if (Date::less_than(td2))return true;
    else if (Date::equal(td2) && Time::less_than(td2)) {
        return true;
    }
    else return false;
}
```

(2)增加increment

调用Time中的increment增加秒数，判断如果增加后的时间是0:0:0说明发生了日期的进位，则调用Date中的increment。

```
void increment() {
    Time::increment();
    if (((Time*)this)->equal(Time(0, 0, 0)))Date::increment();
}
```

(3)相减disfference

调用less_than判断时间较低的日期，循环从低日期增加秒数，直到达到高日期为止并计数返回。

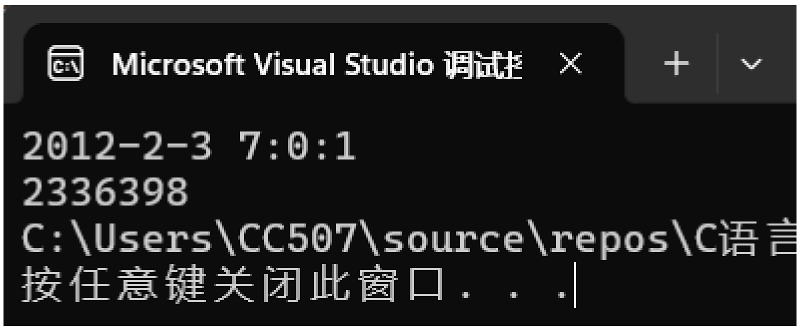
```
int difference(const TimeWithDate& td2)const {
    int days = 0;
    if (less_than(td2))
        for (TimeWithDate td = *this; td.less_than(td2); td.increment())days--;
    else
        for (TimeWithDate td = td2; td.less_than(*this); td.increment())days++;
    return days;
}
```

程序运行结果

main()方法

```
int main() {
    TimeWithDate td1(2012, 2, 3, 7, 0, 0), td2(2012, 3, 1, 8, 0, 0);
    td1.increment();
    td1.display(); cout << endl;
    cout << td2.difference(td1);
}
```

运行结果：



```
2012-2-3 7:0:1
2336398
C:\Users\CC507\source\repos\C语言
按任意键关闭此窗口. . .
```

题目6

回答

Square类不能以public继承自Rectangle类，如果以public方式继承，那么Square对象就可以访问所有的Rectangle方法，如果调用set方法改变长和宽就不再是正方形。

解决方法：改为private继承就行

```
class Square: private Rectangle{
    ...
    ...
}
```