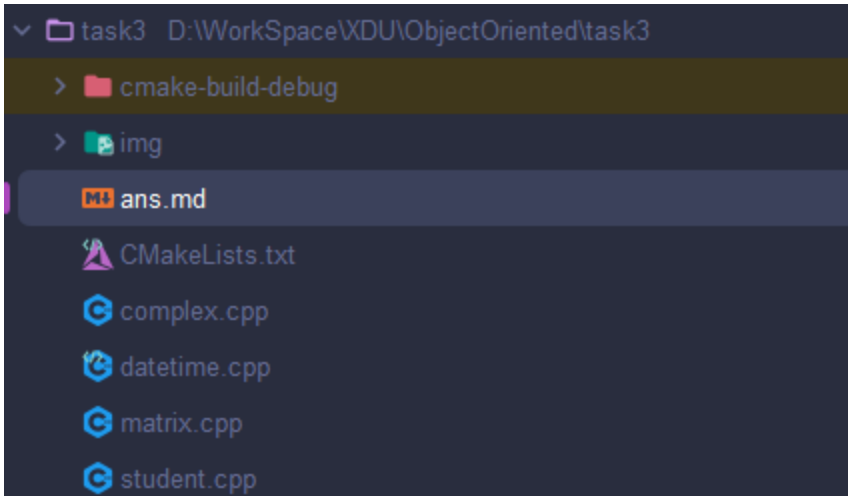
# OOP作业3

## 文件结构


task3 D:\WorkSpace\XDU\ObjectOriented\task3
cmake-build-debug
img
ans.md
CMakeLists.txt
complex.cpp
datetime.cpp
matrix.cpp
student.cpp

# 具体实现

## 1. 对象指针

### 源码

CMakeLists.txt × student.cpp ×

D:\WorkSpace\XDU\ObjectOriented\task3\CMakeLists.txt

```cpp
#include <iostream>

class Student {
public:
    int id;
    float score;

    Student(int id, float score) : id(id), score(score) {}
};

// 函数声明
void printStudents(Student* students);
void max(Student* students);

int main() {
    // 创建5个学生的数组
    Student students[5] = {
        Student( id: 1,  score: 85.0),
        Student( id: 2,  score: 90.5),
        Student( id: 3,  score: 78.0),
        Student( id: 4,  score: 88.5),
        Student( id: 5,  score: 95.0)
    };

    // 用指针指向数组首元素，输出第1，3，5个学生的数据
    printStudents(students);

    // 找出成绩最高的学生并输出其学号
    max(students);

    return 0;
}

// 输出第1，3，5个学生的数据
void printStudents(Student* students) {
    std::cout << "Student 1: ID = " << students[0].id << ", Score = " << students[0].score << std::endl;
    std::cout << "Student 3: ID = " << students[2].id << ", Score = " << students[2].score << std::endl;
    std::cout << "Student 5: ID = " << students[4].id << ", Score = " << students[4].score << std::endl;
}
```

```cpp
// 输出第1，3，5个学生的数据
void printStudents(Student* students) {
    std::cout << "Student 1: ID = " << students[0].id << ", Score = " << students[0].score << std::endl;
    std::cout << "Student 3: ID = " << students[2].id << ", Score = " << students[2].score << std::endl;
    std::cout << "Student 5: ID = " << students[4].id << ", Score = " << students[4].score << std::endl;
}

// 找出成绩最高的学生并输出其学号
void max(Student* students) {
    Student* maxStudent = students;

    for (int i = 1; i < 5; ++i) {
        if (students[i].score > maxStudent->score) {
            maxStudent = &students[i];
        }
    }

    std::cout << "The student with the highest score is: ID = " << maxStudent->id << ", Score = " << maxStudent->score << std::endl;
}

```

## 运行结果

```
D:\WorkSpace\XDU\ObjectOriented\task3\cmake-build-debug\task3.exe
Student 1: ID = 1, Score = 85
Student 3: ID = 3, Score = 78
Student 5: ID = 5, Score = 95
The student with the highest score is: ID = 5, Score = 95

进程已结束，退出代码为 0
```

# 2. 重载计算运算符

## 源码

CMakeLists.txt | student.cpp | complex.cpp ×

```cpp
#include <iostream>
#include <cmath>

class Complex {
private:
    double real;
    double imag;

public:
    Complex(double r = 0, double i = 0) : real(r), imag(i) {}

    // 加法运算符重载
    Complex operator+(const Complex& other) const {
        return Complex(real + other.real, imag + other.imag);
    }

    // 减法运算符重载
    Complex operator-(const Complex& other) const {
        return Complex(real - other.real, imag - other.imag);
    }

    // 乘法运算符重载
    Complex operator*(const Complex& other) const {
        return Complex(real * other.real - imag * other.imag, real * other.imag + imag * other.real);
    }

    // 除法运算符重载
    Complex operator/(const Complex& other) const {
        double denominator = other.real * other.real + other.imag * other.imag;
        return Complex((real * other.real + imag * other.imag) / denominator,
                       (imag * other.real - real * other.imag) / denominator);
    }

    // 输出复数
    void display() const {
        std::cout << real << (imag >= 0 ? " + " : " - ") << std::abs(imag) << "i" << std::endl;
    }
};
```

```cpp
int main() {
    Complex c1(r: 3, i: 4);
    Complex c2(r: 1, i: 2);

    // 计算复数的和、差、积和商
    Complex sum = c1 + c2;
    Complex diff = c1 - c2;
    Complex product = c1 * c2;
    Complex quotient = c1 / c2;

    // 输出结果
    std::cout << "Sum: ";
    sum.display();

    std::cout << "Difference: ";
    diff.display();

    std::cout << "Product: ";
    product.display();

    std::cout << "Quotient: ";
    quotient.display();

    return 0;
}
```

## 运行结果

```
D:\WorkSpace\XDU\ObjectOriented\task3\cmake-build-debug\task3.exe
Sum: 4 + 6i
Difference: 2 + 2i
Product: -5 + 10i
Quotient: 2.2 - 0.4i

进程已结束，退出代码为 0
```

# 3. 重载输入输出流运算符

## 源码

CMakeLists.txt | student.cpp | complex.cpp | matrix.cpp

```cpp
#include <iostream>

class Matrix {
private:
    int data[2][3];

public:
    // 默认构造函数
    Matrix() {
        for (int i = 0; i < 2; ++i) {
            for (int j = 0; j < 3; ++j) {
                data[i][j] = 0;
            }
        }
    }

    // 友元函数，重载流提取运算符 >>
    friend std::istream& operator>>(std::istream& is, Matrix& matrix);

    // 友元函数，重载流插入运算符 <<
    friend std::ostream& operator<<(std::ostream& os, const Matrix& matrix);
};

// 重载流提取运算符 >>
std::istream& operator>>(std::istream& is, Matrix& matrix) {
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 3; ++j) {
            is >> matrix.data[i][j];
        }
    }
    return is;
}
```

```cpp
// 重载流插入运算符 <<
std::ostream& operator<<(std::ostream& os, const Matrix& matrix) {
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 3; ++j) {
            os << matrix.data[i][j] << " ";
        }
        os << std::endl;
    }
    return os;
}

int main() {
    Matrix mat;

    std::cout << "Enter the elements of a 2x3 matrix (6 integers):" << std::endl;
    std::cin >> mat;

    std::cout << "The matrix you entered is:" << std::endl;
    std::cout << mat;

    return 0;
}

```

## 运行结果

```
D:\WorkSpace\XDU\ObjectOriented\task3\cmake-build-debug\task3.exe
Enter the elements of a 2x3 matrix (6 integers):
2 4 6
9 6 3
The matrix you entered is:
2 4 6
9 6 3

进程已结束，退出代码为 0
```

# 4. 友元类

## 源码

CMakeLists.txt | datetime.cpp ×

```cpp
#include <iostream>

class Date;  // 前向声明

class Time {
private:
    int hour;
    int minute;
    int second;

public:
    Time(int h = 0, int m = 0, int s = 0) : hour(h), minute(m), second(s) {}

    // 友元函数，用于显示时间和日期
    void display(const Date& date);
};

class Date {
private:
    int year;
    int month;
    int day;

    // 声明Time类为友元类
    friend class Time;

public:
    Date(int y = 2000, int m = 1, int d = 1) : year(y), month(m), day(d) {}
};

// Time类的display函数实现
void Time::display(const Date& date) {
    std::cout << "Date: " << date.year << "-" << date.month << "-" << date.day << std::endl;
    std::cout << "Time: " << hour << ":" << minute << ":" << second << std::endl;
}

int main() {
    // 创建Date和Time对象
    Date date( y: 2024,  m: 11,  d: 18);
    Time time( h: 22,  m: 30,  s: 45);

    // 使用Time类的display函数输出日期和时间
    time.display(date);

    return 0;
}
```

## 运行结果

```
D:\WorkSpace\XDU\ObjectOriented\task3\cmake-build-debug\task3.exe
Date: 2024-11-18
Time: 22:30:45
```