Задачи на оценку «отлично»

1. Создать класс

```
class rectangle
{
    private:
        double x1, y1, x2, y2;
        int res;
        // прочие скрытые переменные и функции
    public:
        rectangle();
        rectangle(double, double, double, double);
        void set(double, double, double, double);
        void show();
        void save(ofstream fout);
        rectangle operator*(rectangle);
        rectangle operator*(double k);
};
```

Класс `rectangle` должен описывать прямоугольник со сторонами, параллельными осям координат. Координаты одного из углов хранятся в переменных `x1, y1`, а противоположного ему в — `x2, y2`. Конструктор без параметров заполняет координаты значениями (0; 0) и (1; 1), конструктор с параметрами — набором из четырех чисел, задаваемых пользователем, функция `set` также задает координаты набором из четырех чисел. Эти три функции также записывают в переменную `res` единицу. Функция `show` выводит на экран значения координат левого нижнего и правого верхнего углов, если `res` равна единице, и “Rectangle does not exist” в противном случае. Функция `save` сохраняет координаты вершин прямоугольника в файл, соответствующий поточной переменной `fout`, если таковой прямоугольник существует. Перегруженный оператор умножения `operator*(rectangle)` определяет новый четырехугольник, соответствующий пересечению исходных. Если такой четырехугольник существует (см. рис. 1.1), то в переменные `x1, y1, x2, y2` записываются координаты его двух противоположных вершин, а в переменную `res` — единица. В противном случае (см. рис. 1.2) в переменную `res` записывается ноль. Другой перегруженный оператор умножения `operator*(double k)` увеличивает координаты `x1, y1, x2, y2` в `k` раз.

Необходимо выполнить следующие задания:

1. Продемонстрировать работоспособность оператора умножения `operator*(rectangle)`. Сохранить последовательность координат вершин двух исходных прямоугольников и результата их умножения в текстовом файле, нарисовать

прямоугольники с помощью программы — графопостроителя (Origin, MathLab, Scilab, gnuplot, Mathematica, etc).

2. Продемонстрировать работоспособность оператора умножения `operator*(double k)`. Сохранить последовательность координат вершин исходного прямоугольника и результата умножения в текстовом файле, нарисовать прямоугольники с помощью программы — графопостроителя (Origin, MathLab, Scilab, gnuplot, Mathematica, etc).

3. Реализовать перегруженный оператор умножения, позволяющий выполнять умножение числа типа `double` на прямоугольник `rectangle`. Продемонстрировать его работоспособность аналогично заданию 2.

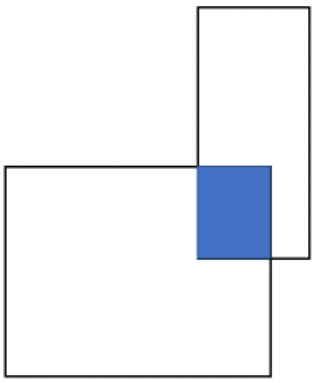

Рис. 1.1

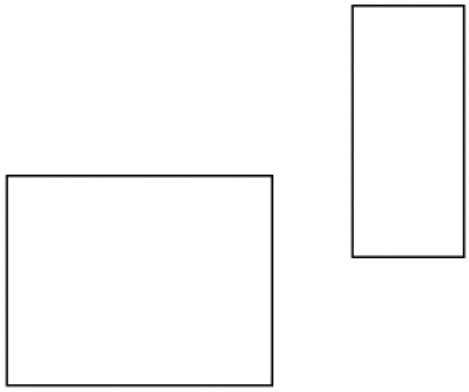

Рис. 1.2

2. Создать класс

```
class ratio
{
    private:
        int num, den;
        // прочие скрытые переменные и функции
    public:
        ratio();
        ratio(int n, int d);
        void set(int, int);
        void show();
        ratio operator*(ratio);
        ratio operator+(ratio);
        ratio operator*(int);
};
```

Класс `ratio` должен хранить рациональную дробь (`num` — числитель, `den > 0` — знаменатель) и обеспечивать операции с ней. Дробь хранится в сокращённом виде. Конструктор без параметров записывает в `num` и `den` единицы, конструктор с параметрами записывает в `num` значение `n`, а в `den` — значение `d`, функция `set` работает аналогично. Функция `show` выводит на экран значение дроби в виде

```
109
###
23
```

Если значение дроби отрицательно, то перед символами `###` должен стоять минус. Перегруженный оператор умножения `operator*(ratio)` вычисляет произведение дробей, перегруженный оператор сложения `operator+(ratio)` — их сумму, а оператор `operator*(int)` — произведение дроби на целое число.

Необходимо выполнить следующие задания:

1. Продемонстрировать работоспособность операторов умножения и сложения.

2. Реализовать перегруженный оператор умножения, позволяющий выполнять умножение числа типа `int` на дробь `ratio`. Продемонстрировать его работоспособность.

Задачи на оценку «хорошо»

3. Создать класс

```
class ratio
{
     private:
          int num, den;
          // прочие скрытые переменные и функции
     public:
          ratio();
          ratio(int n, int d);
          void set(int, int);
          void show();
          ratio operator*(ratio);
          ratio operator/(ratio);
          ratio operator+(ratio);
          ratio operator-(ratio);
          ratio operator*(int);
};
```

Класс `ratio` должен хранить рациональную дробь (`num` — числитель, `den > 0` — знаменатель) и обеспечивать операции с ней. Конструктор без параметров записывает в `num` и `den` единицы, конструктор с параметрами записывает в `num` значение `n`, а в `den` — значение `d`, функция `set` работает аналогично. Функция `show` выводит на экран значение дроби в виде

```
109/23
```

Перегруженный оператор умножения `operator*(ratio)` вычисляет произведение дробей, `operator/(ratio)` — их частное, `operator+(ratio)` — их сумму, `operator-(ratio)` — их разность, а оператор `operator*(int)` — произведение дроби на целое число.

Необходимо выполнить следующие задания:

1. Продемонстрировать работоспособность всех операторов.

2. Реализовать перегруженный оператор умножения, позволяющий выполнять умножение числа типа `int` на дробь `ratio`. Продемонстрировать его работоспособность.

4. Создать класс

```
class mycomplex
{
    private:
        int Re, Im;
        // прочие скрытые переменные и функции
    public:
        mycomplex();
        mycomplex(double R, double I);
        void set(double, double);
        void show();
        mycomplex operator*(mycomplex);
        mycomplex operator/(mycomplex);
        mycomplex operator+(mycomplex);
        mycomplex operator-(mycomplex);
        mycomplex operator*(double);
};
```

Класс `mycomplex` должен хранить комплексное число (`Re` — действительная часть, `Im` — мнимая часть) и обеспечивать операции с ним. Конструктор без параметров записывает в `Re` и `Im` нули, конструктор с параметрами записывает в `Re` значение `R`, а в `Im` — значение `I`, функция `set` работает аналогично. Функция `show` выводит на экран значение комплексного числа в виде

```
109 + 23i
```

в случае положительной мнимой части и

```
109 - 23i
```

в случае отрицательной. Перегруженный оператор умножения `operator*(mycomplex)` вычисляет произведение комплексных чисел, `operator/(mycomplex)` — их частное, `operator+(mycomplex)` — их сумму, `operator-(mycomplex)` — их разность, а оператор `operator*(double)` — произведение комплексного числа на число типа `double`.

Необходимо выполнить следующие задания:

1. Продемонстрировать работоспособность всех операторов.

2. Реализовать перегруженный оператор умножения, позволяющий выполнять умножение числа типа `double` на комплексное число `mycomplex`. Продемонстрировать его работоспособность.