Студенты

Задача кажется наивной, но содержит подводные камни. Задача бегло проходится одновременно по (а) перегрузке операторов, (б) пониманию C++-ных строк, (в) пониманию потоков ввода-вывода.

Напишите класс, описывающий студента на контрольной. Прототип класса:

```
class Student
{
public:
    // Задать имя студента
    void setName(string name);

    // Указать количество баллов за контрольную
    void setScore(unsigned int score);

    // Получить имя студента
    string getName() const;

    // Получить количество баллов студента
    unsigned int getScore() const;
};
```

Реализуйте операторы >> (ввод данных студента) и << (вывод данных студента).

Оператор ввода >> должен считывать и заполнять только имя студента, а баллы выставлять в 0.
Внимание: в имени может быть более одного слова!

Оператор вывода << должен выводить данные студента **строго в следующем формате:** 'name': score. То есть имя в одиночных кавычках, после него двоеточие и пробел, после них баллы. Символ конца строки выводить не нужно.

Приблизительный код для тестирования реализованного класса:

```
    Student s;
    cin >> s;
    s.setScore(10);
    cout << s << endl;
```

Для входных данных:

```
Vasya Ivanov
```

Должно вывести:

```
'Vasya Ivanov': 10
```

Вектор 2D

В примерах к лекции был разобран класс вектора на плоскости. Давайте допишем его до конца, собрав по дороге некоторые шишки и нюансы. Взять за основу код с лекции и допиливать только недостающие моменты будет вполне честно и разумно.

Реализовать класс вектора на плоскости, перегрузив операторы для него. Прототип класса:

```
class Vector2D
{
public:
    // Конструкторы
    Vector2D();
    Vector2D(int x, int y);

    // Деструктор
    ~Vector2D();

    // Получение координат
    int getX() const;
    int getY() const;

    // Задание координат
    void setX(int x);
    void setY(int y);

    // Перегруженный оператор - сравнение двух векторов на равенство
    bool operator== (const Vector2D& v2) const;

    // Ещё один перегруженный оператор - неравенство векторов
    // Да, это отдельный оператор! Хинт - настоящие джедаи смогут для != использовать
уже написанное ==
    bool operator!= (const Vector2D& v2) const;

    // Сумма двух векторов, исходные вектора не меняются, возвращается новый вектор
    Vector2D operator+ (const Vector2D& v2) const;

    // Вычитание векторов, исходные вектора не меняются, возвращается новый вектор
    Vector2D operator- (const Vector2D& v2) const;

    // Оператор умножения вектора на скаляр, исходный вектор не меняется,
возвращается новый вектор
    Vector2D operator* (const int a) const;
};

// Оператор умножения скаляра на вектор, исходный вектор не меняется, возвращается
новый вектор
// Неожиданно, правда? Умножение скаляра на вектор - это отдельный оператор, причём
описанный *вне* класса.
Vector2D operator* (int a, const Vector2D& v);

// Вывод вектора, ожидается строго в формате (1; 1)
std::ostream& operator<<(std::ostream& os, const Vector2D& v);

// Чтение вектора, читает просто две координаты без хитростей
std::istream& operator>>(std::istream &is, Vector2D &v);
```

Ожидается, что класс позволит выполнить примерно такой код:

```
    Vector2D v1;
```

```
    cin >> v1;
    cout << v1 << endl;
    Vector2D v2;
    cin >> v2;
    cout << v2 << endl;
    cout << boolalpha << (v1 == v2) << endl;
    cout << boolalpha << (v1 != v2) << endl;
    Vector2D v3 = v1 - v2;
    cout << v3 << endl;
    cout << v3 * 42 << endl;
    cout << 42 * v3 << endl;
```

И на экране после этого будет:

```
1        <-- Это ввод с клавиатуры
2        <-- Ещё ввод
(1; 2)   <-- Это вывод считанного вектора v1 в cout
3        <-- Ввод
4        <-- Ввод
(3; 4)   <-- Это вывод считанного вектора v2 в cout
false
true
(-2; -2)
(-84; -84)
(-84; -84)
```

Вектор ND

Напишите класс N-мерного вектора. Прототип класса:

```
class VectorN
{
public:
    // Конструктор вектора размерности n
    VectorN(unsigned int n);

    // Деструктор
    ~VectorN();

    // Получение размерности вектора
    unsigned getSize() const;

    // Получение значения i-ой координаты вектора,
    // i находится в диапазоне от 0 до n-1
    int getValue(unsigned int i) const;

    // Задание значения i-ой координаты вектора равным value,
    // i находится в диапазоне от 0 до n-1
    void setValue(unsigned int i, int value);

    /*
     * Далее реализуйте перегруженные операторы
     */

    // Оператор == проверяет два вектора на равенство,
    // равны они, если строго равны всех их координаты

    // Оператор != проверяет два вектора на неравенство,
    // они не равны, если хотя бы одна координата отличается

    // Оператор + складывает два вектора покоординатно,
    // возвращает результат как новый экземпляр вектора

    // Оператор * умножает вектор на скаляр типа int покоординатно,
    // возвращает результат как новый экземпляр вектора.
    // Умножение должно работать при любом порядке операндов.
};
```

Важно: можно ориентироваться на то, что работа с классом будет заведомо корректной, складывать два вектора разной размерности не попросят, обращения к координате с номером меньше 0 или больше n-1 не будет.

Приблизительный код для тестирования базовой технической работоспособности реализованного класса:

```
    VectorN a(4);
    a.setValue(0, 0);
    a.setValue(1, 1);
    a.setValue(2, 2);
    a.setValue(3, 3);

    VectorN b(4);
    b.setValue(0, 0);
    b.setValue(1, -1);
    b.setValue(2, -2);
    b.setValue(3, -3);
```

```
    cout << (a == b) << endl;
    cout << (a != b) << endl;

    VectorN c = a + b;
    VectorN d = 5 * c;

    for(unsigned int i = 0; i < a.getSize(); ++i)
        cout << d.getValue(i) << endl;
```

Должно вывести:

```
0
1
0
0
0
0
```