



C++

Разбор домашней работы



Перегрузки операций

Теория



20 минут



Разберем пример: класс вектор (из математики)



Вектор

```
class Vector {  
    int x, y;  
public:  
    Vector(int x = 0, int y = 0): x(x), y(y) {}  
  
    void read() { cin >> x >> y; }  
    void print() { cout << x << " " << y << endl; }  
};
```

**Какие действия можно делать
с векторами?**

**Как реализовать, например,
? сложение векторов?**

Сложение

```
class Vector {  
    int x, y;  
public:  
    Vector(int x = 0, int y = 0): x(x), y(y) {}  
  
    void read() { cin >> x >> y; }  
    void print() { cout << x << " " << y << endl; }  
  
    Vector sum(Vector v) {  
        return Vector(x + v.x, y + v.y);  
    }  
};  
//Где-то на просторах программы:  
Vector a, b;  
a.read(); b.read();  
Vector c = a.sum(b);  
c.print();
```

**Как бы эти же действия
записывались в математике?**

**Можно ли также сделать в
C++?**

Операции в C++

С точки зрения программы все операции в C++ являются **функциями**.

А значит, как и любую другую функцию, операции можно перегрузить для другого набора параметров, в нашем случае - для объектов нашего класса.

Имя функции, являющейся перегрузкой операции, состоит из ключевого слова **operator** и значка той операции, которую необходимо перегрузить:

тип результата **operator** знак операции (параметры) { }

Сложение

```
class Vector {  
    int x, y;  
public:  
    Vector(int x = 0, int y = 0): x(x), y(y) {}  
  
    void read() { cin >> x >> y; }  
    void print() { cout << x << " " << y << endl; }  
  
    Vector operator+(Vector v) {  
        return Vector(x + v.x, y + v.y);  
    }  
};  
//Где-то на просторах программы:  
Vector a, b;  
a.read(); b.read();  
Vector c = a + b;  
c.print();
```

Сложение

```
Vector operator+(Vector v) {  
    return Vector(x + v.x, y + v.y);  
}
```

Vector c = a + b;



**Какие ещё операции можно
производить с векторами?**

**Разберем скалярное
произведение.**

**Что является результатом
скалярного произведения
векторов?**

Произведение

```
class Vector {  
    int x, y;  
public:  
    Vector(int x = 0, int y = 0): x(x), y(y) {}  
  
    void read() { cin >> x >> y; }  
    void print() { cout << x << " " << y << endl; }  
  
    int operator*(Vector v) {  
        return x * v.x + y * v.y;  
    }  
};  
//Где-то на просторах программы:  
Vector a, b;  
a.read(); b.read();  
int c = a * b;  
cout << c;
```

**А можно ли вектор умножить
на число?**

Произведение

```
class Vector {  
    int x, y;  
public:  
    Vector(int x = 0, int y = 0): x(x), y(y) {}  
    void read() { cin >> x >> y; }  
    void print() { cout << x << " " << y << endl; }  
    int operator*(Vector v) {  
        return x * v.x + y * v.y;  
    }  
    Vector operator*(int k) {  
        return Vector(x * k, y * k);  
    }  
};  
  
//Где-то на просторах программы:  
  
Vector a;  
a.read();  
Vector c = a * 3;  
c.print();
```

Унарные и бинарные операции

Все операции, рассмотренные до этого, являются **бинарными**, т.е. применяемыми к **двум** значениям (операндам).

Но помимо бинарных операций, есть и **унарные** - операции, применяемые к **одному** значению (операнду). Пример такой операции - унарный минус.

Перегрузка унарной операции отличается от бинарной тем, что не принимает никаких параметров.

Унарный минус

```
class Vector {  
    int x, y;  
public:  
    Vector(int x = 0, int y = 0): x(x), y(y) {}  
    void read() { cin >> x >> y; }  
    void print() { cout << x << " " << y << endl; }  
    Vector operator-() {  
        return Vector(-x, -y);  
    }  
};  
  
//Где-то на просторах программы:  
Vector a;  
a.read();  
Vector c = -a;  
c.print();
```

Практика



15 минут



Считывание дробей

Для считывания дробей не нужно считывать строку, после чего разбивать её на отдельные части. Достаточно между двумя числами считать / в отдельную символьную переменную:

```
void read() {  
    char c;  
    cin >> x >> c >> y;  
}
```

Также обратите внимание на то, что дроби в задачах сокращать **не обязательно**.

Теория



10 минут



Сравнение

**Что является результатом
сравнения двух значений?**

Операции сравнений

Какие операции сравнений есть: $>$, $>=$, $<$, $<=$, $==$, $!=$.

Результатом каждого из этих действий является `true` или `false`, а значит типом возвращаемого значения сравнения должен быть **bool**.

Равенство

```
class Vector {  
    int x, y;  
public:  
    Vector(int x = 0, int y = 0): x(x), y(y) {}  
    void read() { cin >> x >> y; }  
    void print() { cout << x << " " << y << endl; }  
    bool operator==(Vector v) {  
        return x == v.x && y == v.y;  
    }  
};  
  
//Где-то на просторах программы:  
Vector a, b;  
a.read(); b.read();  
if (a == b)  
    cout << "YES";  
else  
    cout << "NO";
```

Практика



25 минут



Итоги урока

- 1) Что такое операции с точки зрения языка
- 2) Как перегружать арифметические операции для своего класса
- 3) Как перегружать операции сравнения
- 4) Отличие бинарных от унарных операций