



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ \_\_\_\_\_ «Информатика и системы управления»

КАФЕДРА \_\_\_\_\_ «Теоретическая информатика и компьютерные технологии»

**Лабораторная работа № 9**  
**по курсу «Языки и методы программирования»**  
**«Перегрузка операций»**

Студент группы ИУ9-22Б Тараканов В. Д.

Преподаватель Посевин Д. П.

*Москва 2024*

# 1 Задание

`Program<Statement, Env>` – последовательность «команд», представленных объектами некоторого класса `Statement`, которые можно выполнять в окружении, заданном некоторым классом `Env`. Подразумевается, что окружение содержит данные, необходимые для работы «команд». Требование к классу `Statement`: наличие операции «`()`», которая принимает в качестве параметра ссылку на объект класса `Env`, выполняет «команду» и возвращает номер команды, на которую должно быть передано управление, или `-1`, если данная команда завершает закодированную последовательностью программу. Операции, которые должны быть перегружены для `Program<Statement, Env>`: 1. «`<<`» и «`>>`» – добавление новой команды в конец или в начало последовательности, соответственно (эти операции возвращают ссылку на текущую последовательность); 2. «`( )`» – выполнение последовательности команд до тех пор, пока некоторая команда не возвратит `-1` (принимает в качестве параметра ссылку на объект класса `Env`); 3. «`+`» – конкатенация двух последовательностей; 4. «`==`», «`!=`». Класс `Program<Statement, Env>` должен иметь конструктор без параметров, который создаёт пустую последовательность.

## 2 Результаты

Листинг 1 — Вспомогательный класс `Env`

```
1      class Env {
2          private:
3              int data;
4          public:
5              Env(int initialData = 0) : data(initialData) {}
6              int getData() const {
7                  return data;
8              }
9              void setData(int newData) {
10                 data = newData;
11             }
12         };
13
14
```

## Листинг 2 — Класс Statement

```
1
2     #include "Env.h"
3     #include <iostream>
4
5     class Statement {
6     private:
7         std::string id;
8         std::string type;
9     public:
10        Statement(const std::string& identifier, const std::string&
type) : id(identifier), type(type) {}
11        int operator()(Env& env) {
12            int newData = env.getData();
13            if (type == "plus") {
14                newData++;
15            } else if (type == "minus") {
16                newData--;
17            } else if (type == "divide") {
18                newData/=2;
19            } else if (type == "product") {
20                newData*=2;
21            } else {
22                return -1;
23            }
24            env.setData(newData);
25            if (newData > 100) {
26                return -1;
27            }
28            std::cout << "Data: " << newData << std::endl;
29            return 0;
30        }
31        bool operator==(const Statement& other) const {
32            return id == other.id;
33        }
34    };
35
36
37
```

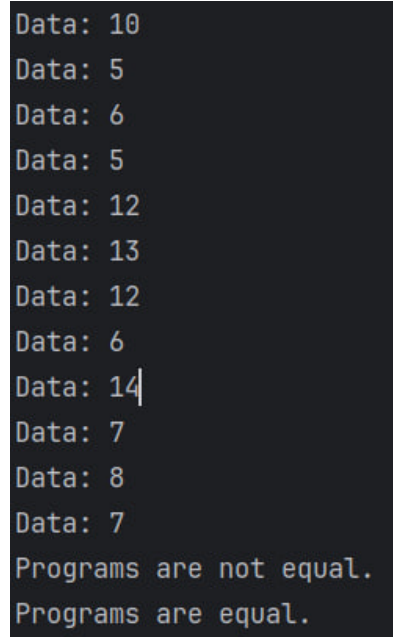
### Листинг 3 — Класс Program

```
1
2  #include <vector>
3  #include "Statement.h"
4
5  template<class Statement, class Env>
6  class Program {
7      private:
8          std::vector<Statement> statements;
9
10     public:
11
12     Program() {}
13
14     Program& operator<<(const Statement& statement) {
15         statements.push_back(statement);
16         return *this;
17     }
18
19
20     Program& operator>>(const Statement& statement) {
21         statements.insert(statements.begin(), statement);
22         return *this;
23     }
24
25     void operator()(Env& env) {
26         for (auto& statement : statements) {
27             int nextCommand = statement(env);
28             if (nextCommand == -1) {
29                 break;
30             }
31         }
32     }
33
34     Program operator+(const Program& other) const {
35         Program result = *this;
36         for (const auto& statement : other.statements) {
37             result << statement;
38         }
39         return result;
40     }
41
42     bool operator==(const Program& other) const {
43         return statements == other.statements;
44     }
45
46     bool operator!=(const Program& other) const {
47         return !(*this == other);
48     }
49 };
50
51
52
```

## Листинг 4 — Функция main для проверки работы класса Program

```
1      #include <iostream>
2      #include "Program.h"
3
4      int main() {
5          Env env(5);
6          Env env2(6);
7          Env env3(7);
8          Statement statement1("1", "plus"), statement2("2", "minus"),
statement3("3", "divide"), statement4("4", "product");
9          Program<Statement, Env> program;
10         Program<Statement, Env> program2;
11         Program<Statement, Env> program3;
12         program2 << statement1 << statement2 << statement3 >> statement4
;
13         program3 << statement1 << statement2 >> statement3 >> statement4
;
14         program << statement1 << statement2 >> statement3 >> statement4;
15         program(env);
16         program2(env2);
17         program3(env3);
18         if (program==program2) {
19             std::cout << "Programs are equal." << std::endl;
20         } else {
21             std::cout << "Programs are not equal." << std::endl;
22         }
23         if (program != program3) {
24             std::cout << "Programs are not equal." << std::endl;
25         } else {
26             std::cout << "Programs are equal." << std::endl;
27         }
28         return 0;
29     }
30
31
```

Результат запуска представлен на рисунках ??.



```
Data: 10
Data: 5
Data: 6
Data: 5
Data: 12
Data: 13
Data: 12
Data: 6
Data: 14
Data: 7
Data: 8
Data: 7
Programs are not equal.
Programs are equal.
```

Рис. 1 — Результат