Федеральное государственное бюджетное образовательное учреждение высшего образования Нижегородский государственный технический университет им. Р.Е. Алексеева

Лабораторная работа №2 «Строковой калькулятор»

> Выполнил: студент 2-го курса кафедры «Прикладная математика» Поляков Кирилл

## Оглавление

Введение	3
Документация	4
Файл main.cpp	4
Содержание main.cpp	5
Файл Stack.h	6
Содержание Stack.h	7
Шаблон класса Stack< T >	9
Структура Stack< T >::Node	13
Содержание Stack.cpp	14
Файл Value.h	16
Структура Data	17
Класс Value	18
Содержание Value.h	21
Содержание Value.cpp	22
Файл RPN.h	23
Класс RPN	24
Содержание RPN.h	27
Содержание RPN.cpp	28
Файл Operation.h	31
Пространство имен Operation	32
Содержание Operation.h	35
Содержание Operation.cpp	36
Тестирование	38

#### Введение

Цель работы - написать программу, анализирующую математическое выражение и вычисляющую его значение. Данная программа должна поддерживать базовые арифметические операции (+, -, \*, /), поддерживать числа с плавающей запятой, поддерживать переменные в выражениях, поддерживать унарный минус. Для выполнения вычисления нам нужно будет переводить строку, содержащую математическое выражение, в обратную польскую запись и только после этого выполнять само вычисление.

## Файл main.cpp

```
#include <iostream>
#include <fstream>
#include "RPN.h"
```

### Функции

```
int main (int argc, char *argv[])
```

### Функции

## main()

Входная точка программы

#### **Аргументы**

argc Количество аргументов, переданных в консольargv Аргументы, переданные в консоль

#### Возвращает

int

См. определение в файле main.cpp строка 14

#### main.cpp

```
#include <iostream>
 2
    #include <fstream>
    #include "RPN.h"
 3
 4
 5
    using namespace std;
 6
14
    int main(int argc, char* argv[])
15
          string expression = "";
16
          for (int i = 0; i < argc; i++)</pre>
17
18
         {
              if ((string("--expression").find(argv[i]) != string::npos) && (i + 1 < argc))
    expression = argv[i + 1];
else if ((string("--variable").find(argv[i]) != string::npos) && (i + 1 < argc))</pre>
19
20
21
22
23
                    ofstream fout("Data.txt");
24
                    int beg = i;
25
                   for (i = i + 1; i < argc; i++)
26
27
                    {
                         if (string(argv[i]).find("=") != string::npos)
28
29
                         {
30
                              string variable;
31
                             double value;
32
                             bool fl = false;
33
34
                             string tempValue;
35
36
                              for (int b = 0; b < string(argv[i]).length(); b++)</pre>
37
38
                                   if (argv[i][b] == '=')
39
                                        fl = true;
                                   if (!fl)
40
41
                                        variable += argv[i][b];
42
                                   else if (isdigit(argv[i][b]))
                                       tempValue += argv[i][b];
43
44
                              value = atof(tempValue.c_str());
45
46
                             fout << variable << "=" << tempValue << endl;
cout << "'" + variable + "' = " + tempValue << endl;</pre>
47
48
49
                         else
50
51
                         {
52
                              cout << "Expression number " << i - beg << " is written</pre>
     incorrectly!"
                       << endl;
53
54
                    }
55
56
          cout << expression << " = " << RPN::Calculation(expression) << endl;</pre>
57
58
59
    }
60
```

## Файл Stack.h

#include <iostream>

### Классы

class	Stack <t></t>	
-------	---------------	--

struct Stack< T >::Node

#### Stack.h

```
#pragma once
     #include <iostream>
  4
     template <class T>
  8
     class Stack
  9
         struct Node
 13
 14
              T element;
 15
 16
             Node* prev;
         }*top;
 17
 18
 19
     public:
 23
         Stack() :top(nullptr)
 24
         {}
 25
 29
         ~Stack()
 30
         {
 31
             while (top)
 32
              {
                  Node* prev = top;
 33
 34
                  top = top->prev;
 35
                  delete prev;
 36
              }
 37
         }
 38
 44
         int Count()
 45
 46
              if (IsEmpty())
 47
                  return 0;
 48
 49
              int count = 1;
 50
             Node* temp = top;
 51
             while (temp->prev)
 52
 53
                  count++;
 54
                  temp = temp->prev;
 55
 56
              return count;
 57
         }
 58
 64
         void Push(T element)
 65
              Node* newNode = new Node;
 66
 67
              newNode->element = element;
              newNode->prev = top;
 68
 69
              top = newNode;
 70
         }
 71
         T Pop()
 76
 77
 78
              //if (IsEmpty())
 79
             // return nullptr;
 80
              T temp = top->element;
 81
 82
             Node* del = top;
 83
             top = top->prev;
 84
             delete del;
 85
              return temp;
 86
         }
 87
 93
         T Peek()
 94
 95
              if (IsEmpty())
 96
                  return -1;
 97
              return top->element;
 98
         }
 99
         bool IsEmpty()
105
106
         {
107
              return top ? false : true;
         }
108
109
```

#### Шаблон класса Stack< T >

#include <Stack.h>

#### Классы

struct Node

### Открытые члены

```
Stack ()

~Stack ()

int Count ()

void Push (T element)

T Pop ()

T Peek ()

bool IsEmpty ()

void PrintStack ()
```

#### Закрытые данные

struct Stack::Node \* top

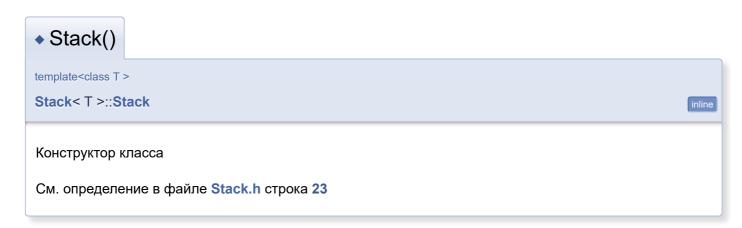
### Подробное описание

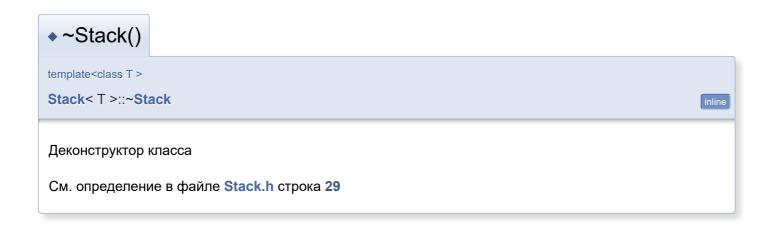
# template<class T> class Stack< T >

Класс, реализующий стек

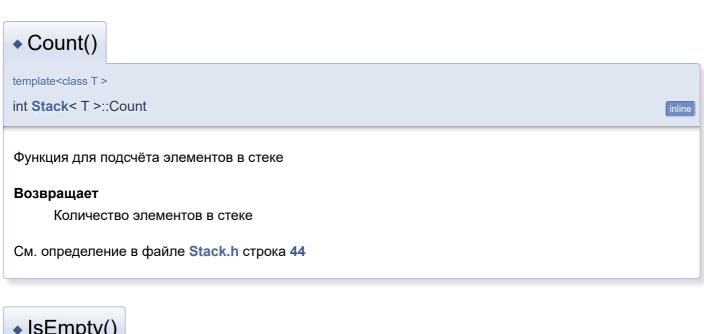
См. определение в файле Stack.h строка 8

## Конструктор(ы)



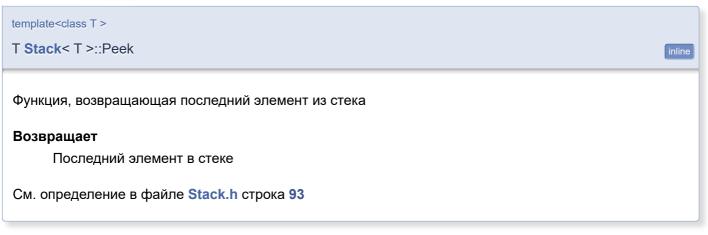


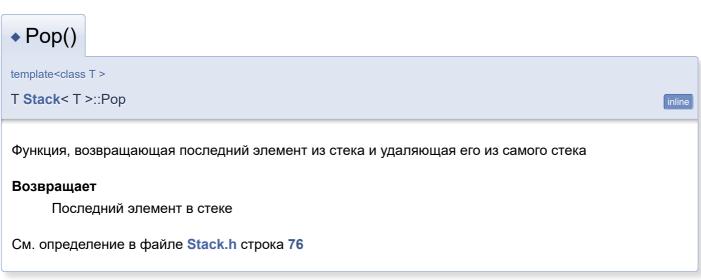
### Методы

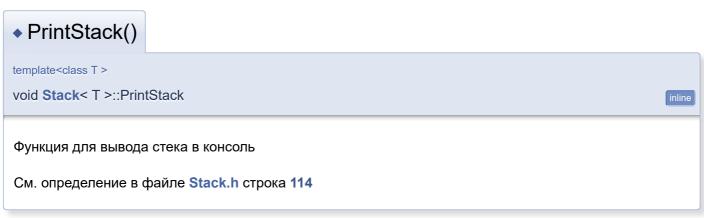














## Данные класса



Объявления и описания членов классов находятся в файлах:

- Stack.h
- Stack.cpp

## Структура Stack< T >::Node

### Открытые атрибуты

T element

Node \* prev

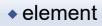
### Подробное описание

# template<class T> struct Stack< T >::Node

Структкра, реализующая элементы стека

См. определение в файле Stack.h строка 13

### Данные класса



template<class T >

T Stack < T >:: Node:: element

См. определение в файле Stack.h строка 15



template<class T >

Node\* Stack< T >::Node::prev

См. определение в файле Stack.h строка 16

Объявления и описания членов структуры находятся в файле:

· Stack.h

### Stack.cpp

```
#include "Stack.h"
 2
    #include <iostream>
 4
    using namespace std;
 5
    template <class T>
 6
 7
    Stack<T>::Stack() :top(nullptr)
 8
 9
10
11
12
    template <class T>
    Stack<T>::~Stack()
13
14
15
        while (top)
16
        {
            Node* prev = top;
17
18
            top = top->prev;
19
            delete prev;
20
        }
    }
21
22
23
    template <class T>
24
    int Stack<T>::Count()
25
26
        if (IsEmpty())
27
            return 0;
28
29
        int count = 1;
30
        Node *temp = top;
31
        while (temp->prev)
32
        {
33
            count++;
34
            temp = temp->prev;
35
36
        return count;
37
    }
38
39
    template <class T>
40
    void Stack<T>::Push(T element)
41
    {
42
        Node* newNode = new Node;
43
        newNode->element = element;
44
        newNode->prev = top;
45
        top = newNode;
46
    }
47
48
    template <class T>
49
    bool Stack<T>::IsEmpty()
50
    {
51
        return top ? false : true;
52
53
54
    template <class T>
55
    T Stack<T>::Pop()
56
57
        if (IsEmpty())
58
            return -1;
59
60
        T temp = top->element;
61
        Node* del = top;
        top = top->prev;
62
63
        delete del;
64
        return temp;
65
    }
66
67
    template <class T>
68
    T Stack<T>::Peek()
69
70
        if (IsEmpty())
71
            return -1;
72
        return top->element;
73
```

## Файл Value.h

#include <string>

См. исходные тексты.

### Классы

struct	Data		
class	Value		

## Структура Data

#include <Value.h>

### Открытые атрибуты

```
double DoubleValue = NULL
string StringValue = ""
```

## Подробное описание

Структура, содержащая данные о типе данных числа

См. определение в файле Value.h строка 9

### Данные класса

### DoubleValue

double Data::DoubleValue = NULL

См. определение в файле Value.h строка 11

## StringValue

string Data::StringValue = ""

См. определение в файле Value.h строка 12

Объявления и описания членов структуры находятся в файле:

· Value.h

#### Класс Value

#include <Value.h>

#### Открытые члены

Value (string value)

bool IsString ()

bool IsDouble ()

int TypeInfo ()

### Открытые атрибуты

Data data

## Подробное описание

Класс, реализующий входное значение

См. определение в файле Value.h строка 19

## Конструктор(ы)

Value()

Value::Value ( string value )

Конструктор класса

#### **Аргументы**

value Переданное значение

См. определение в файле Value.cpp строка 3

### Методы

IsDouble()

#### bool Value::IsDouble ( )

Функция для проверки числа на принадлежность к типу double

#### Возвращает

true, если тип значения является double и false в любом другом случае

См. определение в файле Value.cpp строка 16

## IsString()

bool Value::IsString ( )

Функция для проверки числа на принадлежность к строке

#### Возвращает

true, если тип значения является строкой и false в любом другом случае

См. определение в файле Value.cpp строка 26

## TypeInfo()

int Value::TypeInfo()

Функция, возвращающая тип значения

#### Возвращает

1-double, 2-в другом случае

См. определение в файле Value.cpp строка 36

### Данные класса

### data

Data Value::data

См. определение в файле Value.h строка 22

Объявления и описания членов классов находятся в файлах:

Value.h

• Value.cpp

### Value.h

```
#pragma once
#include <string>
      using namespace std;
5
9
      struct Data
10 {
11
12
             double DoubleValue = NULL;
string StringValue = "";
13 };
14
15
19 class Value
20 {
21 public:
22 Data
            Data data;
Value(string value);
bool IsString();
bool IsDouble();
int TypeInfo();
28
34
40
46
47 };
```

### Value.cpp

```
#include "Value.h"
 2
 3
    Value::Value(string value)
 4
 5
        char* end;
        double res = strtod(value.c_str(), &end);
 6
 7
        if (end == value.c_str() || *end != '\0')
 8
            data.StringValue = value;
 9
10
        //data->StringValue = value;
11
12
            data.DoubleValue = res;
13
            //data->DoubleValue = res;
14 }
15
16
    bool Value::IsDouble()
17
18
        /*if (sizeof(data) == sizeof(double))
19
            return true;
        return false;*/
if (data.DoubleValue == NULL)
20
21
22
            return false;
23
        return true;
24
    }
25
    bool Value::IsString()
26
27
28
        //if (sizeof(data) == sizeof(string))
29
        // return true;
30
        //return false;
        if (data.StringValue == "")
31
32
            return false;
33
        return true;
    }
34
35
36
    int Value::TypeInfo()
37
38
        if (IsDouble())
            return 1;
39
40
        return 2;
41
    }
42
```

## Файл RPN.h

#include <string>

### Классы

class RPN

#### Класс RPN

#include <RPN.h>

### Открытые статические члены

static string Calculation (string input)

### Закрытые статические члены

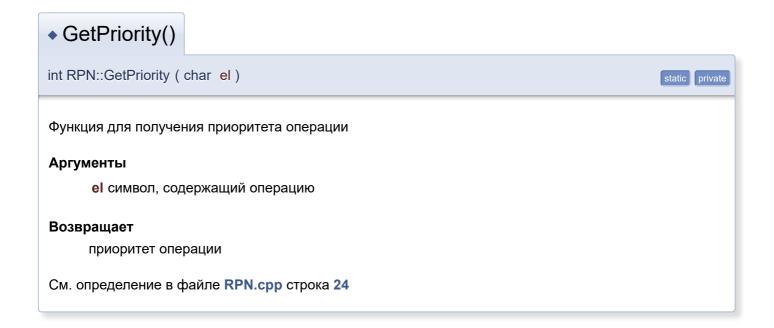
static bool IsDelimeter (char el)
static int GetPriority (char el)
static bool IsOperator (char el)
static string Reverse (string input)

## Подробное описание

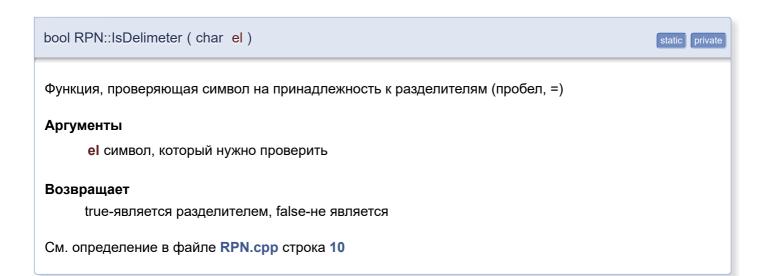
Класс для выполнения основных операция для вычисления значения

См. определение в файле RPN.h строка 9

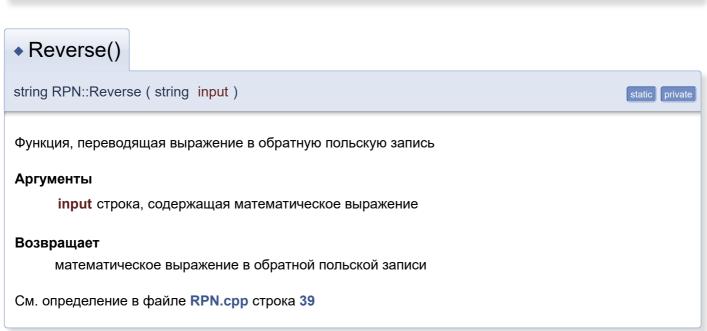
### Методы



IsDelimeter()







## Calculation()

#### string RPN::Calculation (string input)



Функция, вычисляющая значение математического выражения в обратной польской записи

#### **Аргументы**

input математическое выражение в обратной польской записи

#### Возвращает

вычисленное значение

См. определение в файле RPN.cpp строка 154

Объявления и описания членов классов находятся в файлах:

- RPN.h
- RPN.cpp

### RPN.h

```
#pragma once
     #include <string>
 4
     using namespace std;
5
9
    static class RPN
10 {
          static bool IsDelimeter(char el);
static int GetPriority(char el);
static bool IsOperator(char el);
17
24
31
          static string Reverse(string input);
38
39
46
47
     public:
          static string Calculation(string input);
48
```

#### RPN.cpp

```
#include "RPN.h"
    #include "Stack.h"
 2
    #include "Operation.h"
    #include <iostream>
    #include <fstream>
#include "Value.h"
 6
 7
 8
    using namespace Operation;
 9
10
    bool RPN::IsDelimeter(char el)
11
    {
        if (string(" =").find(el) != string::npos)
    return
return;
12
13
14
         return false;
15
    }
16
    bool RPN::IsOperator(char el)
17
18
         if (string("+-/*^()").find(el) != string::npos)
19
20
             return true;
21
         return false;
22
    }
23
24
    int RPN::GetPriority(char el)
25
26
         switch (el)
27
         {
        case '(': return 0;
case ')': return 1;
28
29
        case '+': return 2;
case '-': return 3;
case '*': return 4;
30
31
32
        case '/': return 4;
33
         case '^': return 5;
34
         default: return 6;
35
36
37
    }
38
39
    string RPN::Reverse(string input)
40
41
         if (input.empty())
42
             return "Error. The string is empty!";
43
44
         string output;
45
         Stack <char> operStack;
46
47
48
         string temp = "";
         for (int i = 0; i < input.length(); i++)</pre>
49
50
              if (input[i] !=
51
                  temp += input[i];
52
53
         input = temp;
54
55
         for (int i = 0; i < input.length(); i++)</pre>
56
57
             if (IsDelimeter(input[i]))
58
                  continue;
59
60
             if (isdigit(input[i]))
61
                  while (!IsDelimeter(input[i]) && !IsOperator(input[i]) && i < input.length())</pre>
62
63
64
                       output += input[i];
65
                       i++;
66
                  output += " ";
67
68
69
             else if (isalpha(input[i]))
70
71
72
73
                  while (!IsDelimeter(input[i]) && !IsOperator(input[i]) && i < input.length())</pre>
```

```
74
                  {
 75
                      temp += input[i];
 76
 77
 78
                  ifstream fin("Data.txt");
 79
                  if (fin.is_open())
 80
 81
                      string str;
 82
                      while (getline(fin, str))
 83
 84
                           string tempStr;
 85
                           tempStr = str.substr(0, str.find_first_of('='));
                           if (temp == tempStr)
 86
 87
 88
                               output += str.substr(str.find_first_of('=')+1, str.length()-1) +
 89
                               temp = "";
 90
                               break;
 91
                           }
 92
                      }
 93
 94
                      output += temp;
 95
                  }
 96
                  else
                       cout << "Failed to open the file" << endl;</pre>
 97
 98
 99
                  fin.close();
100
              }
101
102
              if (IsOperator(input[i]))
103
104
                  if (input[i] == '(')
105
                  {
106
                      operStack.Push(input[i]);
107
108
                  else if (input[i] == ')')
109
110
                       char s = operStack.Pop();
111
                      try
112
                       {
113
                           while (s != '(')
114
                           {
115
                               output += s;
                               s = operStack.Pop();
116
117
118
119
                       catch (exception ex)
120
121
                           cout << "Perhaps there is no '('" << endl;</pre>
122
123
                  }
                  else
124
125
126
                      if (!operStack.IsEmpty())
127
                      {
128
                           if (GetPriority(input[i]) <= GetPriority(operStack.Peek()))</pre>
129
130
                               output += operStack.Pop();
131
                               operStack.Push(input[i]);
132
133
134
                               operStack.Push(input[i]);
135
                      else
136
137
                           operStack.Push(input[i]);
138
139
                  }
140
              }
141
142
         while (!operStack.IsEmpty())
143
144
              char el = operStack.Pop();
145
              if (el ==
                          (')
"Missing ')'" << endl;
146
                  cout <<
147
              else
148
                  output += el;
149
         }
150
151
         return output;
152 }
```

```
153
     string RPN::Calculation(string input)
154
155
156
         string result;
157
         Stack<string> temp;
158
         string RevInput = Reverse(input);
159
         for (int i = 0; i < RevInput.length(); i++)</pre>
160
161
162
              if (isdigit(RevInput[i]))
163
164
                  string str;
165
                  while (!IsDelimeter(RevInput[i]) && !IsOperator(RevInput[i]))
166
167
                      str += RevInput[i];
168
169
                      i++;
170
                      if (i == RevInput.length())
171
                           break;
172
                  temp.Push(str);
173
                  i--;
174
175
              else if (IsOperator(RevInput[i]))
176
177
178
                  if (temp.Count() < 2)</pre>
179
                      return temp.Pop();
180
                  Value first(temp.Pop());
Value second(temp.Pop());
181
182
183
                  switch (RevInput[i])
184
185
186
                  case '+':
187
188
                      result = Addition(first, second);
189
                      break;
190
191
                      result = Subtraction(first, second);
192
193
                  case '*':
194
                      result = Multiply(first, second);
195
                      break;
196
197
                      result = Divide(first, second);
198
                      break;
                  case '^':
199
200
                      result = Pow(first, second);
201
                      break;
202
203
                  temp.Push(result);
204
              }
205
         }
206
         string str;
         while (!temp.IsEmpty())
207
             str += temp.Pop()'+ " ";
208
209
         return str;
210 }
```

## Файл Operation.h

#include <string>
#include "Value.h"

## Пространства имен

#### Operation

## Функции

string	Operation::Addition (Value one, Value two)
string	Operation::Subtraction (Value one, Value two)
string	Operation::Multiply (Value one, Value two)
string	Operation::Divide (Value one, Value two)
string	Operation::Pow (Value one, Value two)

### Пространство имен Operation

### Функции

```
string Addition (Value one, Value two)
string Subtraction (Value one, Value two)
string Multiply (Value one, Value two)
string Divide (Value one, Value two)
string Pow (Value one, Value two)
```

## Подробное описание

Пространство имён, содержащее базовые математические операции

### Функции

## Addition()

Функция, реализующая операцию сложения

#### **Аргументы**

**one** Аргумент, содержащий первое значение для сложения **two** Аргумент, содержащий второе значение для сложения

#### Возвращает

Строка, содержащее значение после вычисления

См. определение в файле Operation.cpp строка 7

Divide()

Функция, реализующая операцию деления

#### **Аргументы**

**one** Аргумент, содержащий первое значение для деления **two** Аргумент, содержащий второе значение для деления

#### Возвращает

Строка, содержащее значение после вычисления

См. определение в файле Operation.cpp строка 61

## Multiply()

Функция, реализующая операцию умножения

#### **Аргументы**

**one** Аргумент, содержащий первое значение для умножения **two** Аргумент, содержащий второе значение для умножения

#### Возвращает

Строка, содержащее значение после вычисления

См. определение в файле Operation.cpp строка 43

◆ Pow()

Функция, реализующая операцию возведения в степень

#### **Аргументы**

**one** Аргумент, содержащий первое значение для возведения в степень **two** Аргумент, содержащий второе значение для возведения в степень

#### Возвращает

Строка, содержащее значение после вычисления

См. определение в файле Operation.cpp строка 79

## Subtraction()

Функция, реализующая операцию вычитания

#### **Аргументы**

one Аргумент, содержащий первое значение для вычитания two Аргумент, содержащий второе значение для вычитания

#### Возвращает

Строка, содержащее значение после вычисления

См. определение в файле Operation.cpp строка 25

## Operation.h

```
#pragma once
#include <string>
#include "Value.h"

using namespace std;

namespace Operation
{
    string Addition(Value one, Value two);
    string Subtraction(Value one, Value two);
    string Multiply(Value one, Value two);
    string Divide(Value one, Value two);
    string Divide(Value one, Value two);
    string Pow(Value one, Value two);
};

};
```

### Operation.cpp

```
#include "Operation.h"
 2
 3
    #include <iostream>
 4
 5
    using namespace std;
 6
 7
    string Operation::Addition(Value one, Value two)
 8
 9
        if (one.TypeInfo() == two.TypeInfo())
10
11
            if (one.IsDouble())
12
            {
                string temp = to_string(two.data.DoubleValue + one.data.DoubleValue);
13
                return temp.substr(0, temp.find_last_not_of('0') + 1);
14
15
            if (one.IsString())
16
                return two.data.StringValue + " + " + one.data.StringValue;
17
18
19
        if (one.IsDouble() && two.IsString())
    return two.data.StringValue + " + " + to_string(one.data.DoubleValue).substr(0,
20
    to_string(one.data.DoubleValue).find_last_not_of('0') + 1);
21
        return to_string(two.data.DoubleValue).substr(0,
    22
23
24
    string Operation::Subtraction(Value one, Value two)
25
26
27
        if (one.TypeInfo() == two.TypeInfo())
28
29
            if (one.IsDouble())
30
            {
31
                string temp = to_string(two.data.DoubleValue - one.data.DoubleValue);
32
                return temp.substr(0, temp.find_last_not_of('0') + 1);
33
            if (one.IsString())
34
35
                return two.data.StringValue + " - " + one.data.StringValue;
36
        if (one.IsDouble() && two.IsString())
    return two.data.StringValue + " - " + to_string(one.data.DoubleValue).substr(0,
37
38
    to_string(one.data.DoubleValue).find_last_not_of('0') + 1);
39
        return to_string(two.data.DoubleValue).substr(0;
    to_string(two.data.DoubleValue).find_last_not_of('0') + 1)
                - " + one.data.StringValue;
40
41
    }
42
43
    string Operation::Multiply(Value one, Value two)
44
45
        if (one.TypeInfo() == two.TypeInfo())
46
47
            if (one.IsDouble())
48
                string temp = to_string(two.data.DoubleValue * one.data.DoubleValue);
49
50
                return temp.substr(0, temp.find_last_not_of('0') + 1);
51
            if (one.IsString())
52
                return two.data.StringValue + " * " + one.data.StringValue;
53
54
        if (one.IsDouble() && two.IsString())
    return two.data.StringValue + " * " + to_string(one.data.DoubleValue).substr(0,
55
56
    to_string(one.data.DoubleValue).find_last_not_of('0') + 1);
        return to_string(two.data.DoubleValue).substr(0;
57
    58
59
60
    string Operation::Divide(Value one, Value two)
61
62
63
        if (one.TypeInfo() == two.TypeInfo())
64
        {
65
            if (one.IsDouble())
66
            {
67
                string temp = to string(two.data.DoubleValue / one.data.DoubleValue);
```

```
68
             return temp.substr(0, temp.find_last_not_of('0') + 1);
69
          }
if (one.IsString())
70
             return two.data.StringValue + " / " + one.data.StringValue;
71
72
73
      to_string(one.data.DoubleValue).find_last_not_of('0') + 1);
75
      return to_string(two.data.DoubleValue).substr(0,
   76
77
   }
78
79
   string Operation::Pow(Value one, Value two)
80
      if (one.TypeInfo() == two.TypeInfo())
81
82
83
          if (one.IsDouble())
84
          {
85
             string temp = to_string(pow(two.data.DoubleValue, one.data.DoubleValue));
             return temp.substr(0, temp.find_last_not_of('0') + 1);
86
87
          if (one.IsString())
88
89
             return two.data.StringValue + "^" + one.data.StringValue;
90
91
      if (one.IsDouble() && two.IsString())
    return two.data.StringValue + "^" + to_string(one.data.DoubleValue).substr(0,
92
   to_string(one.data.DoubleValue).find_last_not_of('0') + 1);
93
      return to_string(two.data.DoubleValue).substr(0,
94
   95
96
```

#### Тестирование

```
'a' = 4
'b' = 226
10^a+b/2 = 10113.
```