

Министерство образования и науки РФ
ФГБОУ ВО ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Кафедра «Информационная безопасность систем и технологий»

ОТЧЕТ
о лабораторной работе №4
КЛАССЫ В СИ++

Дисциплина: Языки программирования

Группа: 18ПИ1

Выполнил: Нестеров И.С.

Количество баллов:

Дата сдачи:

Принял: к.т.н., доцент Лупанов М.Ю.

Пенза 2018

1 Цель работы

1.1 Освоить создание классов в программах на Си++, работу с конструкторами и деструкторами классов, перегрузку операторов для классов.

2 Задание к лабораторной работе

2.1 Реализовать класс String с конструктором по умолчанию, конструктором копирования, деструктором и перегруженным оператором `operator<<` для вывода строки в поток.

2.2 Написать программу для работы с классом String, демонстрирующую его возможности.

2.3 Добавить к реализации класса конструктор инициализации Си-строкой. Модифицировать программу для демонстрации возможностей конструктора.

2.4 Добавить к реализации класса перегруженный `operator>>`, позволяющий вводить значения строки из потока ввода. Модифицировать программу для демонстрации возможностей оператора.

2.5 Добавить к реализации класса перегруженные операторы присваивания и унарного минуса. Модифицировать программу для демонстрации возможностей операторов.

2.6 Добавить к реализации класса еще один перегруженный бинарный `operator+` для выполнения операции конкатенации (сцепления) двух строк. Модифицировать программу для демонстрации возможностей оператора.

3 Результаты работы

3.1 Реализация класса String представлена в приложении А.

3.2 Реализация программы для работы с классом String представлена в приложении А. Работа программы представлена на Рисунке 1.

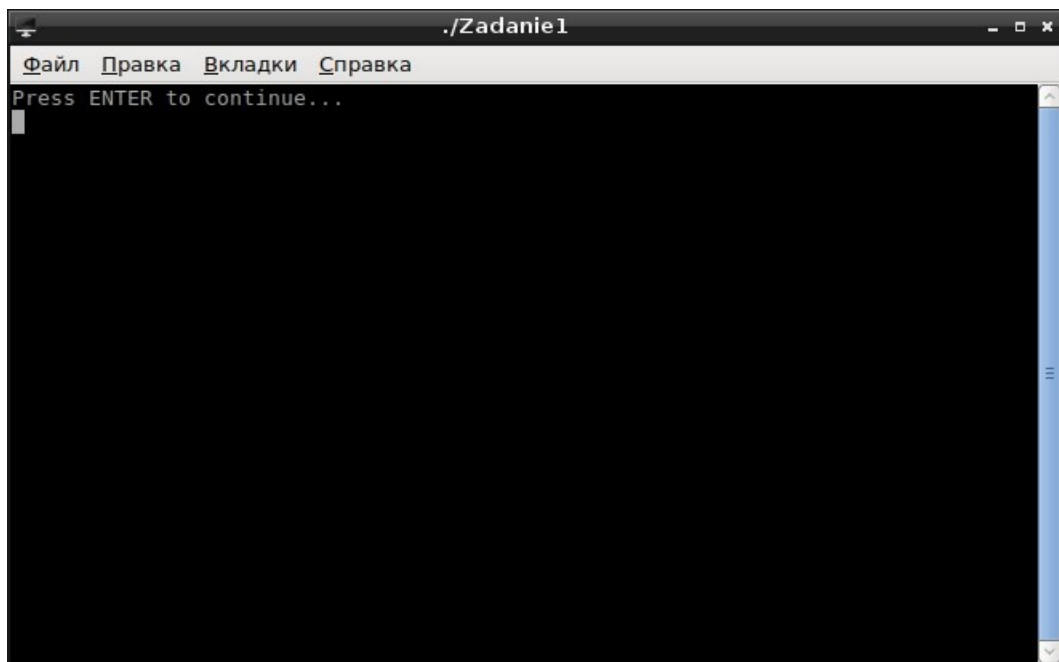


Рисунок 1 - Пример работы программы

3.3 Добавим к реализации класса конструктор инициализации Си-строкой и модифицируем программу для демонстрации возможностей этого конструктора. Для этого нам нужно создать конструктор с параметром `const char * s`. Затем в конструкторе с помощью цикла находится длина входной строки и наше значение `value` присваивается значению входного параметра `s`. Пример работы модифицированной программы представлен на Рисунке 2. Алгоритм работы конструктора представлен на Рисунке 3. Полный текст программы представлен в Приложении Б.

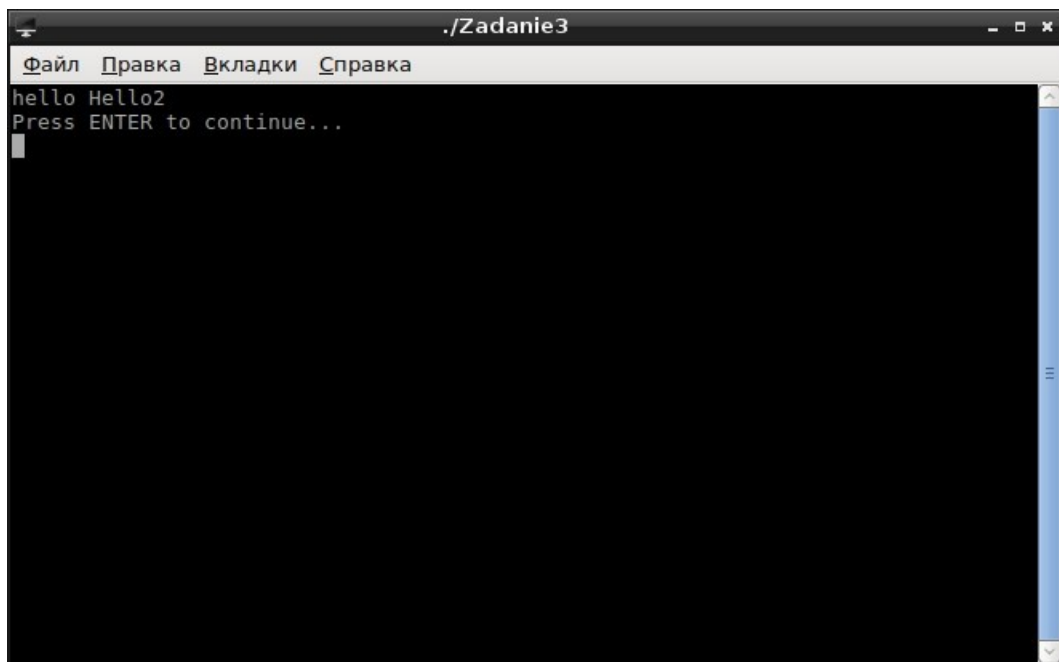


Рисунок 2 - Пример работы модифицированной программы

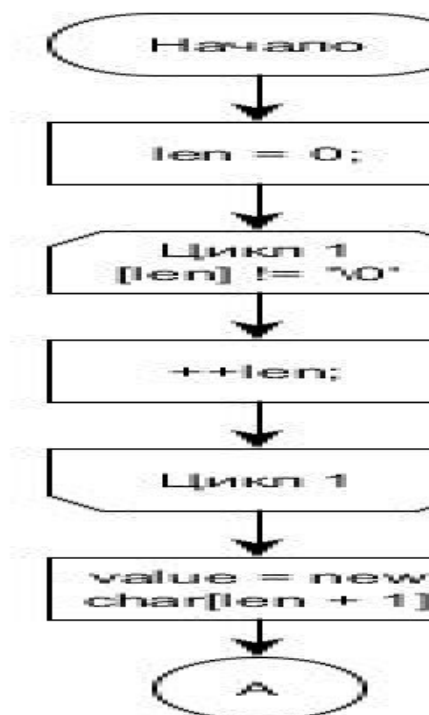
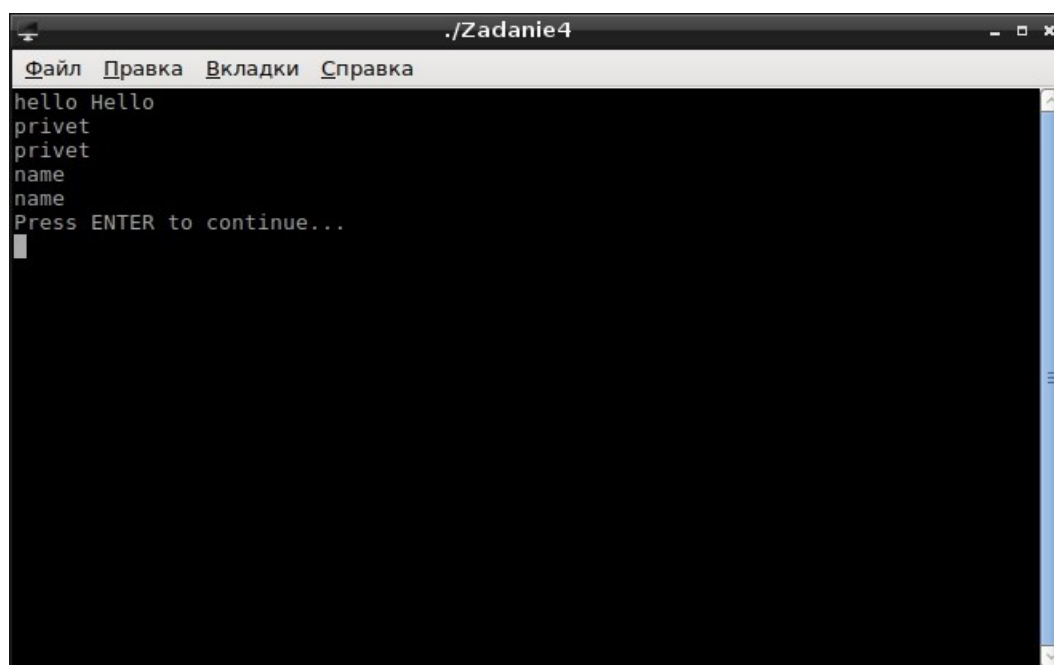


Рисунок 3 - Блок-схема 1

3.4 Добавим к реализации класса перегруженный `operator>>`, позволяющий вводить значения строки из потока ввода и модифицируем

программу для демонстрации возможностей оператора. Для этого мы добавляем в наш класс строку `friend istream &operator >> (istream &is, String &obj)` для реализации перегруженного оператора `>>`. В самой функции мы объявляем массив `v[2048]` типа `char`, и в него записываем буфер, который подавался нам на ввод, одновременно считая длину буфера. Затем мы выделяем необходимую память, предварительно высвободив её, для нашей строки `obj.value`, и записываем в `obj.value` значение `v`. Пример работы программы представлен на Рисунке 4. Алгоритм работы конструктора представлен на Рисунке 5. Полный текст программы представлен в Приложении В.



```
./Zadanie4
Файл  Правка  Вкладки  Справка
hello Hello
privet
privet
name
name
Press ENTER to continue...
```

Рисунок 4 - Пример работы программы перегруженного оператора ввода

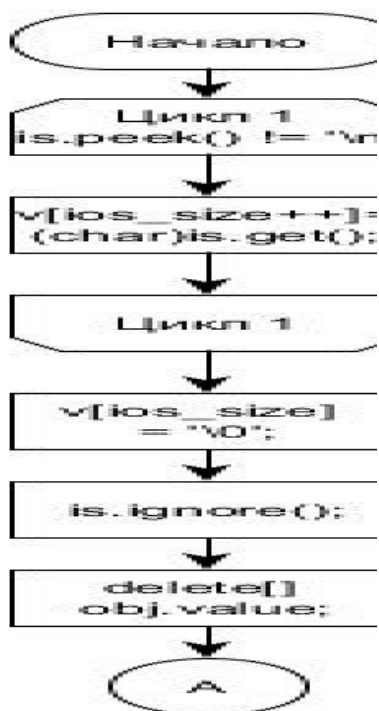


Рисунок 5 - Блок-схема 2

3.5 Реализация, перегруженных операторов присваивания и унарного минуса, и модификация программы для демонстрации возможностей операторов представлена в Приложении Г. Пример работы программы представлен на Рисунке 6.

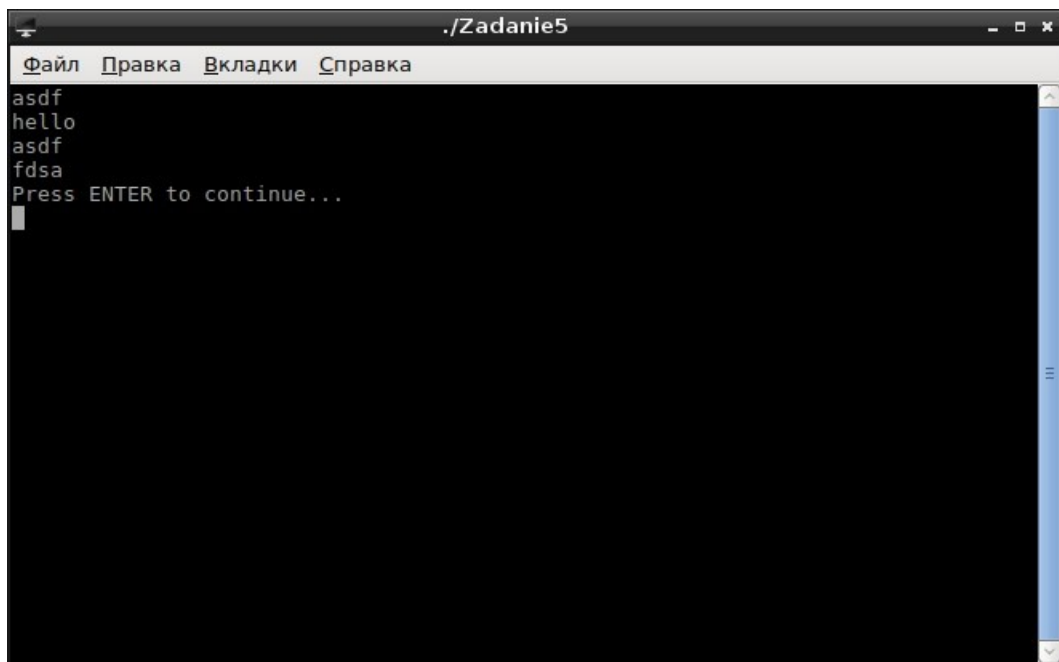


Рисунок 6 - Пример работы программы перегруженных операторов

3.6 Добавим к реализации класса перегруженный бинарный оператор+, позволяющий сцепить две строки, и модифицируем программу для демонстрации возможностей оператора. Для этого мы добавляем в наш класс строку `friend String operator+(const String&, const String&)` для реализации перегруженного оператора. В самой функции мы объявляем массив `ts` типа `String`, и в него записываем сначала значение строки левого аргумента `lhs.value`, а затем правого `rhs.value`. Пример работы программы представлен на Рисунке 7. Алгоритм работы конструктора представлен на Рисунке 8. Полный текст программы представлен в Приложении Д.

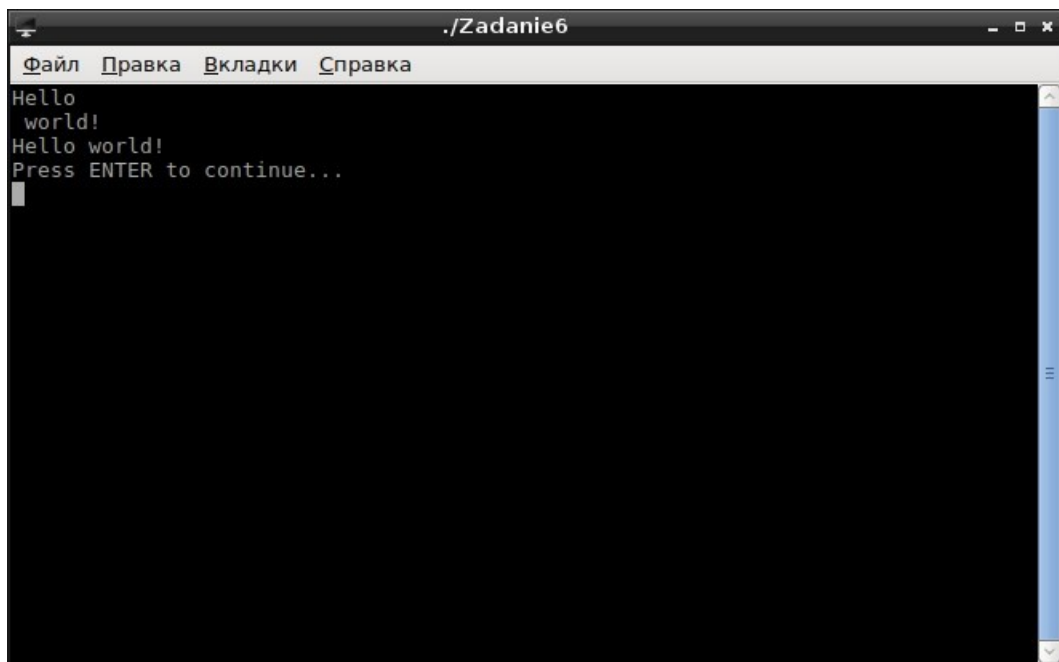


Рисунок 7 - Пример работы программы бинарного оператора +

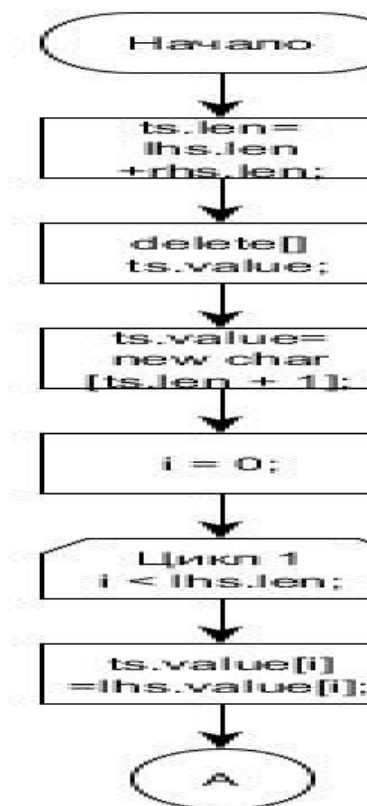


Рисунок 8 - Блок-схема 4

4 Вывод

В результате выполнения работы были изучены классы языка c++, освоены и был применен собственный класс типа String, а также были написаны стандартные функции для работы с этим классом, и получены практические навыки в написании классов на c++.

Приложение А

Текст программы класса String

```
#include <iostream>
using namespace std;

class String
{
    char * value;
    int len;
public:
    String():value(new char[1]{}),len(0){};
    String(const String& s)
    {
        len=s.len;
        value = new char[len+1];
        for (int i=0; i<=len; i++)
            value[i]=s.value[i];
    };
    ~String()
    {
        delete[] value;
    };

    friend ostream& operator<<(ostream& outputStream, const String &
s);
};

ostream& operator<<(ostream& outputStream, const String & s)
{
    return outputStream << s.value;
}
```

```
int main(int argc, char **argv) // Программа для работы с классом
String
{
    String s;
    String s1{};
    String s2 = s;
    return 0;
}
```

Приложение Б

Текст программы с конструктором инициализации си-строкой

```
#include <iostream>
using namespace std;

class String
{
    char * value;
    int len;
public:
    String() :value(new char[1]{}), len(0) {};
    String(const String& s)
    {
        len = s.len;
        value = new char[len + 1];
        for (int i = 0; i <= len; i++)
            value[i] = s.value[i];
    };
    String(const char * s)
    {
        len = 0;
        while (s[len] != '\0') ++len;
        value = new char[len + 1];
        for (int i = 0; i <= len; i++)
            value[i] = s[i];
    };
    ~String()
    {
        delete[] value;
    };
};
```

```

        friend ostream& operator<<(ostream& outputStream, const
String & s);
};

ostream& operator<<(ostream& outputStream, const String & s)
{
    return outputStream << s.value;
}

int main(int argc, char **argv) // Программа для работы с классом
String
{
    String s;
    String s1{};
    String s2 = s;
    String s3("hello");
    String s4 = "Hello2";
    cout << s2;
    cout << s3 << " " << s4 << endl;
    return 0;
}

```

Приложение В

Текст программы для перегруженного оператора ввода

```
#include <iostream>
using namespace std;

class String
{
    char * value;
    int len;
public:
    String() :value(new char[1]{}), len(0) {};
    String(const String& s)
    {
        len = s.len;
        value = new char[len + 1];
        for (int i = 0; i <= len; i++)
            value[i] = s.value[i];
    };
    String(const char * s)
    {const char * s
        len = 0;
        while (s[len] != '\0') ++len;
        value = new char[len + 1];
        for (int i = 0; i <= len; i++)
            value[i] = s[i];
    };
    ~String()
    {
        delete[] value;
    };
};
```

```

        friend ostream& operator<<(ostream& outputStream, const
String & s);
        friend istream &operator >> (istream &is, String &obj)
        {
            int ios_size = 0, i = 0;
            char v[2048];
            while (is.peek() != '\n')
                v[ios_size++] = (char)is.get();
            v[ios_size] = '\0';
            is.ignore();
            delete[] obj.value;
            obj.value = new char[ios_size + 1];
            for (i = 0; i <= ios_size; i++)
                obj.value[i] = v[i];
            obj.len = ios_size;
            return is;
        }
};

```

```

ostream& operator<<(ostream& outputStream, const String & s)
{
    return outputStream << s.value;
}

```

```

int main(int argc, char **argv) // Программа для работы с классом
String
{
    String s;
    String s1{};
    String s2 = s;
    String s3("hello");
    String s4 = "Hello";
    String s5;
}

```

```
    cout << s2;
    cout << s3 << " " << s4 << endl;
    cin >> s5;
    cout << s5 << endl;
    cin >> s4;
    cout << s4 << endl;
    return 0;
}
```


Приложение Г

Текст программы для перегруженных операторов

```
#include <iostream>
using namespace std;

class String
{
    char * value;
    int len;
public:
    String() :value(new char[1]{}), len(0) {};
    String(const String& s)
    {
        len = s.len;
        value = new char[len + 1];
        for (int i = 0; i <= len; i++)
            value[i] = s.value[i];
    };
    String(const char * s)
    {
        len = 0;
        while (s[len] != '\0') ++len;
        value = new char[len + 1];
        for (int i = 0; i <= len; i++)
            value[i] = s[i];
    };
    ~String()
    {
        delete[] value;
    };
};
```

```

        friend ostream& operator<<(ostream& outputStream, const
String & s);
        friend istream &operator >> (istream &is, String &obj)
        {
            int ios_size = 0, i = 0;
            char v[2048];
            while (is.peek() != '\n')
                v[ios_size++] = (char)is.get();
            v[ios_size] = '\0';
            is.ignore();
            delete[] obj.value;
            obj.value = new char[ios_size + 1];
            for (i = 0; i <= ios_size; i++)
                obj.value[i] = v[i];
            obj.len = ios_size;
            return is;
        }
        String& operator=(const String& other);
        String operator-() const;
};

ostream& operator<<(ostream& outputStream, const String & s)
{
    return outputStream << s.value;
}

String& String::operator=(const String& other)
{
    if (this != &other) {
        delete[] value;
        len = other.len;
        value = new char[len + 1];
        for (int i = 0; i <= len; i++)

```

```

        {
            value[i] = other.value[i];
        }
    }
    return *this;
}

String String::operator-() const
{
    String ret;
    ret.len = len;
    delete[] ret.value; // в примере не правильно, память
выделили, а перед этим не освободили
    ret.value = new char[len+1];
    for (int i=0; i<len; i++) {
        ret.value[i] = value[len-i-1];
    }
    ret.value[len] = 0;
    return ret;
}

int main(int argc, char **argv) // Программа для работы с классом
String
{
    String s = "hello";
    String s1;
    String s2;
    String s3;
    String s4;
    s1 = s;
    cin >> s2;
    s3 = s2;
    cout << s1 << endl;
}

```

```
    cout << s3 << endl;  
    s4 = -s3;  
    cout << s4 << endl;  
    return 0;  
}
```

Приложение Д

Текст программы для перегруженного бинарного оператора+

```
#include <iostream>
using namespace std;

class String
{
    char * value;
    int len;
public:
    String() :value(new char[1]{}), len(0) {};
    String(const String& s)
    {
        len = s.len;
        value = new char[len + 1];
        for (int i = 0; i <= len; i++)
            value[i] = s.value[i];
    };
    String(const char * s)
    {
        len = 0;
        while (s[len] != '\0') ++len;
        value = new char[len + 1];
        for (int i = 0; i <= len; i++)
            value[i] = s[i];
    };
    ~String()
    {
        delete[] value;
    };
};
```

```

        friend ostream& operator<<(ostream& outputStream, const
String & s);
        friend istream &operator >> (istream &is, String &obj)
        {
            int ios_size = 0, i = 0;
            char v[2048];
            while (is.peek() != '\n')
                v[ios_size++] = (char)is.get();
            v[ios_size] = '\0';
            is.ignore();
            delete[] obj.value;
            obj.value = new char[ios_size + 1];
            for (i = 0; i <= ios_size; i++)
                obj.value[i] = v[i];
            obj.len = ios_size;
            return is;
        }
        String& operator=(const String& other);
        String operator-() const;
        friend String operator+(const String&, const String&);
};

ostream& operator<<(ostream& outputStream, const String & s)
{
    return outputStream << s.value;
}

String& String::operator=(const String& other)
{
    if (this != &other) {
        delete[] value;
        len = other.len;
        value = new char[len + 1];
    }
}

```

```

        for (int i = 0; i <= len; i++)
        {
            value[i] = other.value[i];
        }
    }
    return *this;
}

```

String String::operator-() const

```

{
    String ret;
    ret.len = len;
    delete[] ret.value;
    ret.value = new char[len + 1];
    for (int i = 0; i < len; i++) {
        ret.value[i] = value[len - i - 1];
    }
    ret.value[len] = 0;
    return ret;
}

```

String operator+(const String& lhs, const String& rhs)

```

{
    int i = 0, j = 0;
    String ts;
    ts.len = lhs.len + rhs.len;
    delete[] ts.value;
    ts.value = new char[ts.len + 1];
    for (i = 0; i < lhs.len; i++)
        ts.value[i] = lhs.value[i];
    for (j = 0; i <= ts.len; i++, j++)
        ts.value[i] = rhs.value[j];
    return ts;
}

```

```
}
```

```
int main(int argc, char **argv) // Программа для работы с классом  
String
```

```
{
```

```
    String s;
```

```
    String s1;
```

```
    String s2;
```

```
    cin >> s;
```

```
    cin >> s1;
```

```
    s2 = s + s1;
```

```
    cout << s2 << endl;
```

```
    return 0;
```

```
}
```