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

ОТЧЁТ
о лабораторной работе №4
Классы в C++

Дисциплина: Языки программирования
Группа:18ПИ1
Выполнил: Асаян А.В.
Количество баллов:
Дата сдачи:
Проверил: к.т.н., доцент Лупанов М.Ю.

Пенза, 2019

1 Цель работы

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

2 Задания к практической 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 Результат выполнения работы

3.1 Был реализован класс String с конструктором по умолчанию, конструктором копирования, деструктором и перегруженным оператором « для вывода строки в поток.

Код класса:

```
#include <iostream>
#include <string>
using namespace std;
class String
```

```
    {
        char * value;
        int len;
    public:
        String():value(new char[1] {}),len(0) {};
        String(const String& s);
        ~String();
          friend ostream& operator<<(ostream& o, const
String & s);
    };

    ostream& operator<<(ostream& outputStream, const
String & s)
    {
        return outputStream << s.value;
    }
    String::~String()
    {
        delete[] value;
    }
    String::String(const String& s)
    {
        len=s.len;
        value = new char[len+1];
        for (int i=0; i<=len; i++)
            value[i]=s.value[i];
    }
```

3.2 Была написана программа, демонстрирующая возможности класса String, реализованного в пункте 3.1. На рисунке 1 представлен результат работы программы. Код программы :

```
    #include <iostream>
    #include <string>
    using namespace std;
    class String
    {
        char * value;
        int len;
    public:
        String():value(new char[1] {}),len(0) {};
        String(const String& s);
        ~String();
```

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

    ostream& operator<<(ostream& outputStream, const
String & s)
    {
        return outputStream << s.value;
    }
    String::~String()
    {
        delete[] value;
    }
    String::String(const String& s)
    {
        len=s.len;
        value = new char[len+1];
        for (int i=0; i<=len; i++)
            value[i]=s.value[i];
    }
    int main(int argc, char **argv)
    {   String s;
        String s1=s;
        cout<<s1;
        return 0;
    }
```

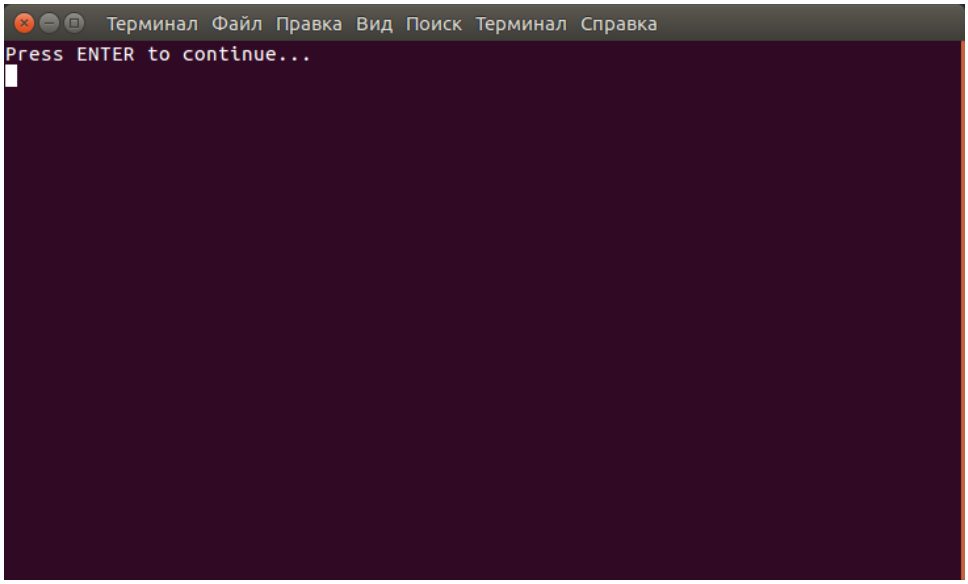

Рисунок 1 — Результат работы программы, демонстрирующие возможности класса String, реализованного в пункте 3.1.

3.3 К классу реализованном в пункте 3.1 был добавлен конструктор инициализации Си-строкой. Код конструктора:

```
String::String(const char *s)
{
    len=0;
    while(s[len]!='\0') {
        len++;
    }
    value = new char[len];
    for(int i=0; i<len; i++) {
        value[i]=s[i];
    }
    value[len]=0;
}
```

Код программы, демонстрирующей возможности класса:

```
#include <iostream>
#include <string>
using namespace std;
class String
{
    char * value;
    int len;
public:
    String():value(new char[1] {}),len(0) {};
    String(const String& s);
    String(const char * s);
    ~String();
    friend ostream& operator<<(ostream& outputStream,
const String & s);
};

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

    }
    value[len]=0;
}
int main(int argc, char **argv)
{
    String s1("hello");
    String s2(s1);
    cout<<s2<<" "<<s1<<" "<<endl;
    return 0;
}
```

На рисунке 2 представлен результат работы программы.

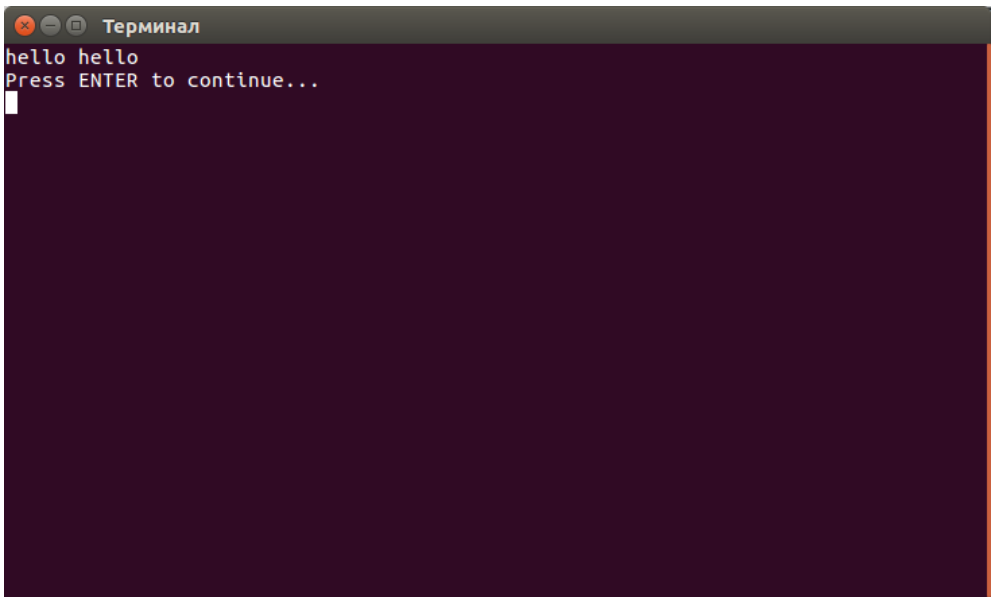

Рисунок 2 — Результат работы программы, с классом String, дополненным конструктором инициализации Си-строкой.

3.4 К реализации классы был добавлен перегруженный оператор >> , позволяющий вводить значения строки из потока ввода. Код конструктора:

```
istream& operator>>(istream& inputStream,  String & s)
{
    int dl=0;
    char temp[90];
    inputStream >> s.value;
    for(int i=0; s.value[i]!=0; i++) {
        temp[i]=s.value[i];
        dl=dl+1;
    }
    delete[] s.value;
    s.len=dl;
    s.value=new char[s.len+1];
    for(int i=0; i<s.len; i++) {
        s.value[i]=temp[i];
    }
    s.value[s.len]=0;
}
```

Код модифицированной программы, показывающей возможность класса, с перегруженным оператором >> :

```
#include <iostream>
#include <string>
using namespace std;
class String
{
    char * value;
    int len;
public:
    String():value(new char[1] {}),len(0) {};
    String(const String& s);
    String(const char * s);
    ~String();
      friend ostream& operator<<(ostream& o, const String & s);
     friend istream& operator>>(istream& o, String & s);
};
```

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

        }

    }
    istream& operator>>(istream& inputStream,   String &
s)
    {
        int dl=0;
        char temp[90];
        inputStream >> s.value;
        for(int i=0; s.value[i]!=0; i++) {
            temp[i]=s.value[i];
            dl=dl+1;
        }
        delete[] s.value;
        s.len=dl;
        s.value=new char[s.len+1];
```

```
    for(int i=0; i<s.len; i++) {
        s.value[i]=temp[i];
    }
    s.value[s.len]=0;
}

int main(int argc, char **argv)
{

    String s;
    cin>>s;
    String s1("hello");
    String s2(s1);
    cout<<s2<<" "<<s1<<" "<<s<<endl;
    return 0;
}
```

На рисунке 3 представлен результат работы этой программы.

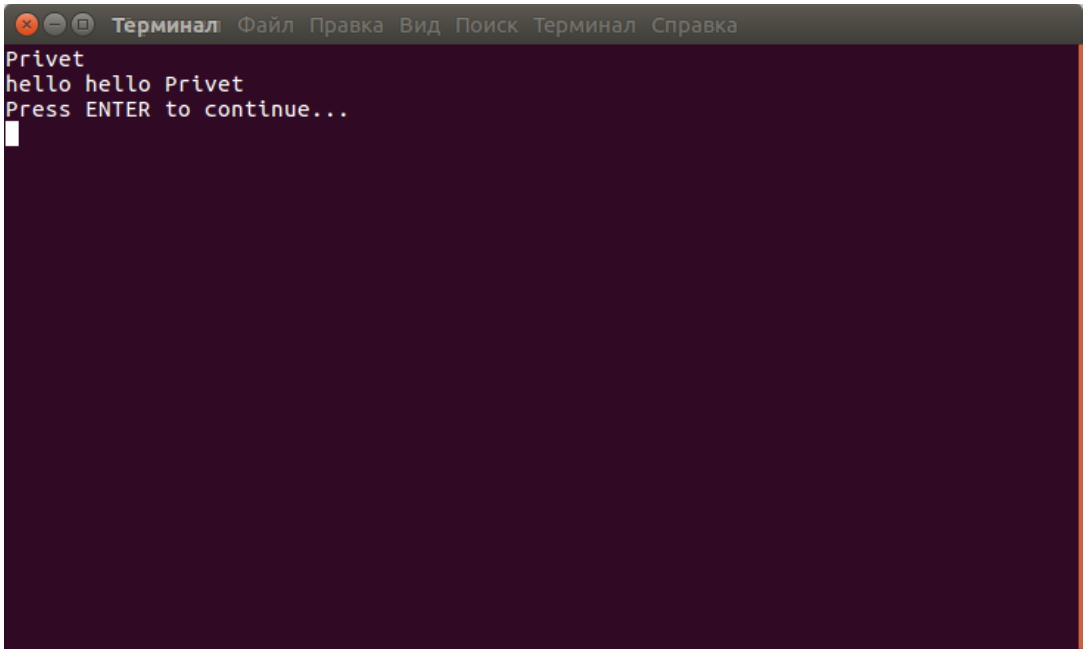

Рисунок 3 — результат работы программы с классом String с перегруженным оператором >>.

3.5 К реализации класса были присвоены перегруженные оператор унарного минуса и присваивания. Код программы, был модифицирован для демонстрации работы перегруженных операторов. На рисунке 4 изображен результат работы программы.Код перегрузки унарного минуса:

```
String String::operator-() const
```

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

Код перегрузки оператора присваивания:

```
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;
}
```

Код программы:

```
#include <iostream>
#include <string>
using namespace std;
class String
{
    char * value;
    int len;
public:
    String():value(new char[1] {}),len(0) {};
    String(const String& s);
    String(const char * s);
    ~String();
          String(int  size):value(new  char[size+1]),
len(size){};
    String operator-() const;
    String& operator=(const String& other);
      friend ostream& operator<<(ostream& o, const
String & s);
     friend istream& operator>>(istream& o,  String &
s);
};
```

```
String String::operator-() const
{
String ret(len);
for (int i=0; i<len; i++) {
ret.value[i] = value[len-i-1];
}
ret.value[len] = 0;
return ret;
}
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;
}
ostream&   operator<<(ostream&   outputStream,   const
String & s)
{
    return outputStream << s.value;
}
String::~String()
{
    delete[] value;
}
String::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::String(const char *s)
{
    int i=0;
    len=0;
    while(s[i]!='\0') {
        len++;
        i++;
```

```
        }
        value = new char[len];
        for(int i=0; i<len; i++) {
            value[i]=s[i];
        }
        value[len]=0;
    }
    istream& operator>>(istream& inputStream,   String &
s)
    {
        int dl=0;
        char temp[90];
        inputStream >> s.value;
        for(int i=0; s.value[i]!=0; i++) {
            temp[i]=s.value[i];
            dl=dl+1;
        }
        delete[] s.value;
        s.len=dl;
        s.value=new char[s.len+1];
        for(int i=0; i<s.len; i++) {
            s.value[i]=temp[i];
        }
        s.value[s.len]=0;
    }

    int main(int argc, char **argv)
    {
        String s;
        cin>>s;
        String s1("hello");
        String s2(s1);
        String s3=-s;
        cout<<s2<<" "<<-s1<<" "<<s<<" "<<s3<<endl;
        return 0;
    }
```

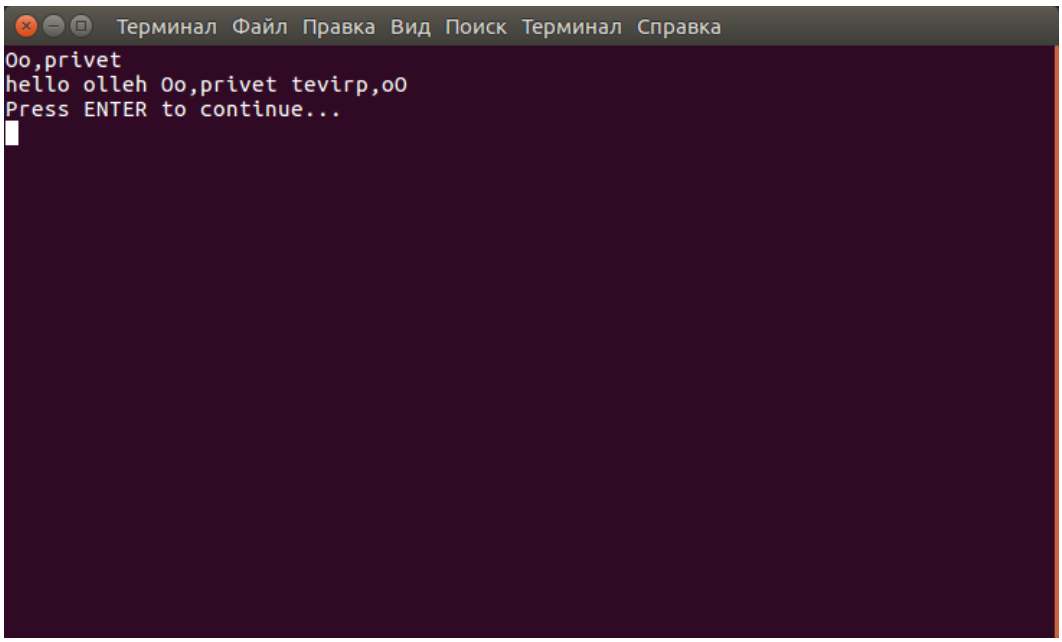

Рисунок 4 — Результат работы программы с перегруженными операторами унарного минуса и присваивания.

3.6 К реализации класса был добавлен перегруженный бинарный оператор +. Код перегрузки оператора:

```
String String::operator+(const String& other)
{
    String ret;
    delete [] ret.value;
    ret.len=len+other.len;
    ret.value=new char[ret.len+1];
    for(int i=0;i<len;i++)
        ret.value[i]=value[i];
    for(int i=0;i+len<ret.len;i++){
        ret.value[i+len]=other.value[i];
    }
        return ret;

}
```

Код программы :

```
#include <iostream>
#include <string>
using namespace std;
class String
{
    char * value;
    int len;
```

```
    public:
        String():value(new char[1] {}),len(0) {};
        String(const String& s);
        String(const char * s);
        ~String();
             String(int  size):value(new  char[size+1]),
len(size) {}
        String operator-() const;
        String operator+(const String& other);
        String& operator=(const String& other);
          friend ostream& operator<<(ostream& o, const
String & s);
         friend istream& operator>>(istream& o,  String &
s);
    };
    String String::operator+(const String& other)
    {
        String ret;
        delete [] ret.value;
        ret.len=len+other.len;
        ret.value=new char[ret.len+1];
        for(int i=0;i<len;i++)
            ret.value[i]=value[i];
        for(int i=0;i+len<ret.len;i++){
            ret.value[i+len]=other.value[i];
        }
            return ret;

    }
    String String::operator-() const
    {
        String ret(len);
        for (int i=0; i<len; i++) {
            ret.value[i] = value[len-i-1];
        }
        ret.value[len] = 0;
        return ret;
    }
    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;
    }
    ostream&   operator<<(ostream&   outputStream,   const
String & s)
    {
        return outputStream << s.value;
    }
    String::~String()
    {
        delete[] value;
    }
    String::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::String(const char *s)
    {
        int i=0;
        len=0;
        while(s[i]!='\0') {
            len++;
            i++;
        }
        value = new char[len];
        for(int i=0; i<len; i++) {
            value[i]=s[i];

        }
        value[len]=0;

    }
    istream& operator>>(istream& inputStream,   String &
s)
    {
        int dl=0;

        char temp[90];
```

```
        inputStream >> s.value;
        for(int i=0; s.value[i]!=0; i++) {
            temp[i]=s.value[i];
            dl=dl+1;
        }
        delete[] s.value;
        s.len=dl;
        s.value=new char[s.len+1];
        for(int i=0; i<s.len; i++) {
            s.value[i]=temp[i];
        }
        s.value[s.len]=0;
    }

    int main(int argc, char **argv)
    {
        String s;
        cin>>s;
        String s1("hello");
        String s2(s1);
        String s3=-s;
        String s4(", Artem");
        String s5=s1+-s+s4;
              cout<<s2<<"   "<<-s1<<"   "<<s<<"   "<<s3<<"
"<<s5<<endl;
        return 0;
    }
```

На рисунке 5 изображен результат работы данной программы.

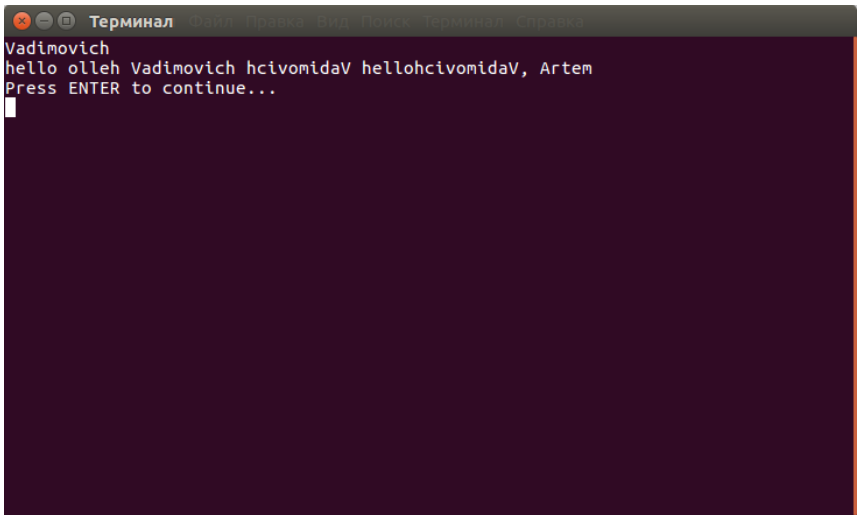

Рисунок 5 — Результат работы программы с оператором + для сцепления строк.

4 Вывод

В результате выполнения лабораторной работы были изучены возможности создания классов в C++. Были освоена работа с классами, получены практические навыки по созданию деструкторов, конструкторов и методов класса. Были получены практические навыки по перегрузке операторов для классов.