МИНИСТЕРСТВО ТРАНСПОРТА И КОММУНИКАЦИЙ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Белорусский государственный университет транспорта»

Кафедра «Информационно-управляющие системы и технологии»

Отчет  
по лабораторным работам

по дисциплине «Объектно-ориентированное программирование»

Выполнил Проверил

студент группы ГИ-21 ст. преп. каф. «ИУСиТ»

Зайцев И. С. Голдобина Т. А.

Гомель, 2020

Оглавление

[3 Лабораторная работа № 3 Тема: «Дружественные функции и классы» 3](#_Toc53741405)

[1.1 Задания 3](#_Toc53741406)

[1.2 Контрольные вопросы 5](#_Toc53741407)

1. Лабораторная работа № 3  
   Тема: «Дружественные функции и классы»

### **Цель**

Понять назначение дружественных функций и классов, изучить принципы перегрузки бинарных и унарных операций.

## Задания

### Задание 1.1

### Условие

Реализовать класс String для работы со строками символов. Перегрузить операторы «=», «+=» так, чтобы производилось сложение строки и объекта. Предоставить конструктор копирования. Определить friend-функции для операторов ввода/вывода в поток.

Программный код

#include <iostream>

#include <clocale>

using namespace std;

class String {

int size;

char\* mas;

public:

String(string str) {

this->size = str.size();

mas = new char[size];

for (int i = 0; i < size; i++) { mas[i] = str[i]; }

}

String(char\* mas, int size) {

this->size = size>0 ? size : 0;

this->mas = mas;

}

String operator = (string s) {

int newStrSize = this->size + s.size();

char newStr[newStrSize];

int i;

for (i = 0; i < this->size; i++) { newStr[i] = this->mas[i];}

for (int j = 0; i < newStrSize; i++, j++) {newStr[i] = s[j];}

return String(newStr, newStrSize);

}

String operator += (string s) {

int newStrSize = this->size + s.size();

char newStr[newStrSize];

int i;

for (i = 0; i < this->size; i++) { newStr[i] = this->mas[i];}

for (int j = 0; i < newStrSize; i++, j++) {newStr[i] = s[j];}

return String(newStr, newStrSize);

}

void print() {

for(int i = 0; i < size; i++) { cout << mas[i]; }

cout << endl;

}

friend std::ostream& operator<< (std::ostream &out, const String &string) {

for(int i = 0; i < string.size; i++) { out << string.mas[i]; }

out << endl;

}

friend std::istream& operator>> (std::istream &in, String &strin) {

in >> strin.size;

for (int i = 0; i < strin.size; i++) { in >> strin.mas[i];}

return in;

}

};

int main() {

setlocale(LC\_ALL, "");

char a[3] = {'a', 'b', 'c'};

String first(a, 3);

cout << first;

String second("hjhjbjb");

cin >> second;

cout << second;

(first = "dab").print();

cout << (second += "cad");

return 0;

}

### Результат

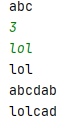


Рисунок 1 – Результат задания 1

## Контрольные вопросы

1. Почему может потребоваться перегрузка оператора присваивания?

Перегрузка оператора присваивания занимает важное место при реализации классов, использующих динамическую память. Она необходима, если нам присвоить полям экземпляра класса, поля другого экземпляра, либо инициировать их после объявления. Присваивание, так же, как и конструктор копии, по умолчанию реализуется побайтным копированием, которое подразумевает только копирование значений полей объекта. Поэтому при использовании динамической памяти в классе произойдет копирование ссылки на нее, а сама область динамической памяти станет разделяемой между двумя объектами. Такое разделение доступа к памяти в C++ при работе с указателями не является корректным.

1. Можно ли изменить приоритет перегруженного оператора?

Перегрузка операций подчиняется правилу, что сохраняются приоритеты операций и правила ассоциации (справа налево или слева направо) по сравнению с использованием в стандартных типах данных.

1. Когда следует переопределять операторы с помощью дружественных функций, а когда с помощью функций элементов класса?

Их нужно предопределять, когда мы хотим, чтобы при использовании с объектами конкретного класса они выполняли заданные функции, что дает возможность использовать собственные типы данных точно так же, как стандартные.

1. Назовите особенности дружественных функций.

Особенностей у дружеских функций достаточно много, но главные из них: дружеские функции не являются компонентами-функциями класса; имеют доступ ко всем компонентами этого класса (private, public, protected); дружеские функции не могут использовать указатель this; в общем случае, дружеские функции являются глобальными независимо от сектора, в котором объявлена (если не объявлена ни в одном другом классе без спецификатора friend); дружественные функции могут рассматриваться как часть интерфейса класса с внешней среды; дружественные функции не наследуются; отношение friend не является ни симметрическим, ни транзитивным, также друзьями класса можно определить перегружение функции.

1. Опишите особенности перегрузки постфиксных и префиксных операторов «++» и «--»

Особенности перегрузки этих операторов состоит в том, что нужно перегружать как префиксную, так и постфиксную форму этих операторов, дружественная к классу функция не получает указателя this класса, параметр в эту функцию может передаваться по ссылке, а не по значению, т. е. перед именем параметра указывается символ ссылки &. Вообще у префиксной формы определенных особенностей нет, в то время как постфиксная должна возвращать значение, которое было до его изменения.

### Вывод

В результате лабораторной работы научились перегружать операторы, изучили дружественные функции и классы.