**Отчет**

**по Лабораторной работе № 8**

**«РЕШЕНИЕ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ КЛАССОВ»**

Специальность 09.02.03«Программирование в компьютерных системах»

ПМ.01 «Разработка программных модулей и программного обеспечения для компьютерных систем»

МДК.01.02 «Прикладное программирование»

**Министерство образования и науки Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,**

**МЕХАНИКИ И ОПТИКИ”**

**ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

Выполнил:

студент группы 2337

Коряков С.А.

Санкт-Петербург

2018/2019

Преподаватель:

ФИО Сорокин Д. Ю.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_201\_\_г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Цель работы

продемонстрировать работу с основными механизмами объектно-ориентированного программирования.

Практическое задание

Реализовать класс согласно варианту. Объект описать минимально возможным количеством полей, остальные характеристики вычислять динамически с помощью методов. Класс должен содержать минимум 2 конструктора: конструктор по умолчанию и параметризованный. Перегрузить операторы записи и чтения потока, для обеспечения потокового ввода-вывода. Перегрузить минимум 2 стандартных оператора(унарный и бинарный).

Написать программу, выполняющую:

1. создание 2-х экземпляров класса, с динамическим выделением памяти, первый с параметрами по умолчанию, второй с параметрами заданными пользователем;

2. вывод всех характеристик заданного экземпляра класса на экран;

3. демонстрацию работы перегруженных операторов;

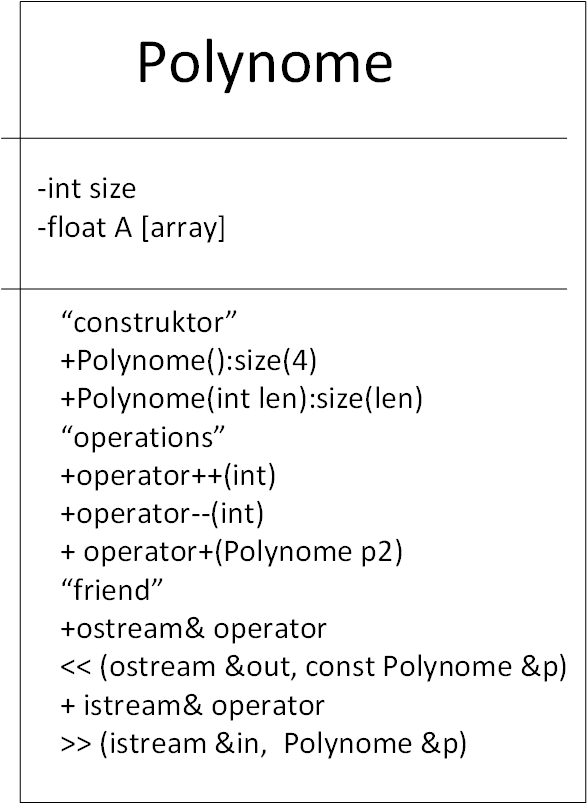
4. освобождение выделенной памяти.

Индивидуальное задание

8. Полином

Диаграмма классов программы

Рисунок 1, Диаграмма классов программы



Листинг

#include <iostream>

using namespace std;

class Polynome //создание класса

{

protected://создание защищенных переменных

int size;

float \*A = new float[size];

public://создание публичных переменных и функций

Polynome() : size(3) {}//создание прототипа класса для вызова без аргументов

Polynome(int len) : size(len) {}{}//создание прототипа класса для вызова с аргументами

~Polynome() {}//деструктор

friend ostream& operator<< (ostream &out, const Polynome &p);//указание на дружественные функции перегрузок потоковых операторов

friend istream& operator>> (istream &in, Polynome &p);

Polynome operator++(int) //перегрузка унарного оператора

{

for (int i = 0; i < size; i++) {

A[i]++;

}

return 0;

}

Polynome operator--(int) {

for (int i = 0; i < size; i++) {

A[i]--;

}

return 0;

}

Polynome operator+(Polynome p2) //перегрузка бинарного оператора

{

int temp\_size = size;

if (temp\_size == p2.size) {

for (int i = 0; i < temp\_size; i++) {

p2.A[i] += A[i];

}

}

temp\_size = p2.size;

return p2;

}

};

ostream& operator<< (ostream &out, const Polynome &p)//функция перегрузки потокового оператора

{

for (int i = p.size; i > 0; i--) {

out << p.A[i-1] << "X^" << i;

if (i > 1) {

out << " + ";

}

}

return out;

}

istream& operator>> (istream &in, Polynome &p) {

for (int i = p.size; i > 0; i--) {

in >> p.A[i-1];

}

return in;

}

int main(int argc, char const \*argv[]) {

Polynome p1;

int size;

cout << "Enter the first polynome:\n";

cin >> p1;

cout << p1 << endl;

p1--;

cout << p1 << endl;

cout << "Enter the second polynome size:\n";

cin >> size;

Polynome p2(size);

cout << "Enter the second polynome:\n";

cin >> p2;

cout << p2 << endl;

p2 = p1 + p2;

cout << p2 << endl;

return 0;

}

Протокол выполнения



Рисунок 2, протокол выполнения программы