Министерство науки и высшего образования РФ

ФГАОУ ВО ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИУ)

Институт естественных и точных наук

Факультет математики, механики и компьютерных технологий

Кафедра прикладной математики и программирования

«Библиотека классов для работы с полиномами»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ   
по дисциплине «Объектно-ориентированное программирование»

ЮУрГУ–01.03.02.2021.313.ПЗ КР

|  |  |  |
| --- | --- | --- |
|  |  | *Руководитель,*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Демидов А.К.*  *«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021г.* |
|  |  | *Автор работы:*  *Студентка группы: ЕТ – 213*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Абзалимов К.В.*  *«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021г.* |
|  |  | *Работа защищена с оценкой*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г.* |

Челябинск – 2021

Федеральное государственное автономное образовательное учреждение высшего образования

«Южно-Уральский государственный университет   
(национальный исследовательский университет)»

Институт естественных и точных наук

Кафедра «Прикладная математика и программирование»

Направление \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

УТВЕРЖДАЮ

Заведующий кафедрой ПМиП

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А.Замышляева

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

**ЗАДАНИЕ**

**на курсовую работу студента**

\_\_\_\_\_\_\_\_*Абзалимова К.В.*\_\_\_\_\_\_\_\_\_

Группа \_\_*ЕТ-213*\_\_\_\_

1. Дисциплина  *Объектно-ориентированное программирование*

2. Тема работы \_\_\_\_*Библиотека классов для работы с полиномами*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Срок сдачи студентом законченной работы *21 декабря 2021 г.*

4. Перечень вопросов, подлежащих разработке

1. разработка иерархии и интерфейса классов;
2. реализация программы (библиотеки классов) на языке С++
3. оформление программной документации (описание программы (библиотеки классов), руководство пользователя, листинг кода) и отчета по курсовой работе
4. презентация проектных решений для защиты КР (иерархия и интерфейсы классов, особенности реализации)

5. Календарный план

|  |  |  |
| --- | --- | --- |
| **Наименование разделов**  **(этапов) курсовой работы** | **Срок выполнения**  **разделов (этапов) работы** | **Отметка**  **о выполнении**  **руководителя** |
| анализ предметной области | 01.09.2021-10.10.2021 |  |
| разработка иерархии и интерфейса классов | 20.09.2021-07.11.2021 |  |
| реализация основных классов, функций | 01.10.2021-20.11.2021 |  |
| тестирование программы и/или классов, улучшение и исправление ошибок | 20.10.2021-11.12.2021 |  |
| оформление программной документации и отчета по курсовой работе | 30.10.2021-20.12.2021 |  |
| защита курсовой работы | 20.12.2021-28.12.2021 |  |

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (расшифровка)

АННОТАЦИЯ

Абзалимов К.В. Библиотека классов для работы с полиномами. – Челябинск: ЮУрГУ, ЕТ-213, 2021. – 21с., библиографический список – 3 наим., 1 прил.

В курсовой работе описывается разработка библиотеки классов для работы с полиномами с помощью объектно-ориентированного подхода. Работа содержит результаты объектно-ориентированного анализа и проектирования, инструкции по установке и использованию библиотеки.

В результате работы была разработана библиотека классов для работы с полиномами, код которой приводится в приложении.

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 5](#__RefHeading___Toc2047_846668840)

[1 Постановка задачи 6](#__RefHeading___Toc5595_846668840)

[2 Описание программы 6](#__RefHeading___Toc5597_846668840)

[3 Инструкция по установке и требования к системе 8](#__RefHeading___Toc5599_846668840)

[4 Руководство пользователя 9](#__RefHeading___Toc2363_3782873837)

[ЗАКЛЮЧЕНИЕ 11](#__RefHeading___Toc2049_846668840)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 12](#__RefHeading___Toc2051_846668840)

[ПРИЛОЖЕНИЕ А 13](#__RefHeading___Toc5610_846668840)

# ВВЕДЕНИЕ

**Актуальность темы**. Объектно-ориентированный подход является наиболее прогрессивной технологией разработки программных систем, позволяет разрабатывать более сложные системы.

**Цель работы** – разработать библиотеку классов для работы с полиномами.

**Задачи работы**:

– изучить приемы объектно-ориентированного анализа;

– научиться разрабатывать программы в объектно-ориентированном стиле;

– овладеть технологиями объектно-ориентированного анализа и проектирования [1];

– изучить концепции объектно-ориентированного программирования; изучить особенности объектной модели языка программирования C++ [2]

– научиться самостоятельно и творчески использовать знания и полученные практические навыки;

– овладеть навыками самостоятельного получения новых знаний по теории и практике объектного подхода в программировании.

**Объект работы** – полиномы

**Предмет работы** – применение объектно-ориентированного подхода для разработки библиотеки.

**Результаты работы** можно использовать в процессе последующего обучения в соответствии с учебным планом подготовки бакалавров по направлению «Прикладная математика и информатика»

## 1 Постановка задачи

Необходимо разработать библиотеку классов для работы с полиномами [3] на языке C++, используя объектно-ориентированное программирование. В качестве элементов полинома используются латинские буквы и числа. Пример:

0.1\*a\*b^3+x\*y^4

Библиотека должна включать в себя следующие возможности:

* Операции +, -, \*, = для операндов полиномов;
* Операции +, -, \*, / , где 1 или 2 операнд – число double (для деления только второй)
* Вычисление полинома при указанных значениях для всех неизвестных;
* Ввод(>>), вывод (<<) в виде одной строки;
* Получить набор неизвестных;
* конструкторы-преобразователи double->полином, string->полином

Анализ предметной области выявляет, что объекты должны хранить коэффициенты и пары (имя переменной, степень) для каждого члена полинома в структуре или в динамическом массиве (в векторе). Но так же должна иметься возможность эффективных вычислений новых коэффициентов, сокращений и изменениями видов основного полинома.

## 2 Описание программы

2.1 Для разработки программы были использованы:

– компилятор MinGW GNU C/C++ 7.2

2.2 Библиотеки состоит из 1 модуля:

Модуль **polynome** (интерфейсная часть в файле .hpp, реализация в файле interface.cpp) содержит следующие классы:

class Monome {

private:

double coef; // Коэффициент монома

std::map<char, int> pows; // Переменны и их степени в формате map

public:

// Конструкторы

Monome():coef(1) {}

Monome(std::string); // Конструктор-преобразователь string -> Polynome

Monome(double num):coef(num) {} // double -> Monome

// Определение операций

Monome operator \*(const Monome &obj) const ;

Monome operator \*(double number) const;

Monome operator /(double number) const;

friend bool operator==(const Monome &obj1, const Monome &obj2);

// Ввод и вывод в поток

friend std::ostream &operator <<(std::ostream &out, const Monome &obj);

friend std::istream &operator >>(std::istream &in, Monome &obj);

// Метод для сравнения переменных + степеней

bool similar(const Monome &obj) const {return pows == obj.pows;}

friend class Polynome;

};

inline bool operator!=(const Monome &obj1, const Monome &obj2) {

return !(obj1 == obj2);

}

class Polynome {

private:

// Вектор содержащий мономы, на которые раскладывается полином

std::vector<Monome>terms;

public:

// Конструкторы вектор

Polynome() {}

// Конструктор преобразователь string -> Polynome

Polynome(std::string);

Polynome(double num);// double -> Polynome

Polynome(Monome num);// Monome -> Polynome

// Определение операций для работы с полиномами

Polynome operator +(const Polynome &obj) const;

Polynome operator -(const Polynome &obj) const;

Polynome operator \*(const Polynome &obj) const;

// Определение операции для деления на число Конструктор прео

Polynome operator /(double number) const;

// Ввод и вывод в поток

friend std::ostream &operator<<(std::ostream &out, const Polynome &obj);

friend std::istream &operator>>(std::istream &in, Polynome &obj);

// Получение переменных

std::vector<char> print\_variables();

// Вычисление полинома для значений

double count\_for(std::map<char, double> &mp);

};

2.3 Пояснения по алгоритму и особенностям реализации

В классе Polynome дополнительно определён конструктор-преобразователь Monome -> Polynome. Все операции полиномов с мономами реализованы через конструктор-преобразователь.

Для double чисел таким же образом реализованы +, -, \* . Деление на число реализовано перегрузкой оператора / .

Для ввода

## 3 Инструкция по установке и требования к системе

Для сборки библиотеки необходим компилятор, поддерживающий  
С++14.

Установка и настройки не требуются. Для работы необходимо добавить файлы polynome.hpp, polynome.cpp в проект, использующий формат JSON.

## 4 Руководство пользователя

При создании полинома необходимо передать в конструктор строку с полиномом. В данную строку входят коэффициенты перед каждым одночленом, знак умножения после коэффициента, знак степени и показатель степени, знак умножения после каждой степени. Для полиномов определены операции сложения, вычитания, умножения, умножения на число, деления на число, также перегружены операции потокового ввода и вывода.

**Замечание:** Если перед каким-либо слагаемым полинома нет коэффициента, то нужно написать коээфициент равный единице.

Примеры работы с объектами класса Polynome:

Polynome p("2\*a^2+3\*b");

Polynome k("-3\*b + 2\*x");

Операции с числами:

cout << p+5.0; // 2\*a^2+3\*b+5

cout << p-5.0; // 2\*a^2+3\*b-5

cout << p\*5.0; // 10\*a^2+15\*b

cout << p/5.0; // 0.2\*a^2+0.6\*b

Операции с полиномами:

cout << p+k; // 2\*a^2+2\*x

cout << p-k; // 2\*a^2+6\*b-2\*x

cout << p\*k; // -6\*a^2\*b-9\*b^2+4\*a\*x+6\*b\*x

Пример вычисления значения полинома методами print\_variables и count\_for:

// Метод print\_variables возвращает вектор уникальных переменных

for (char c:p.print\_variables()) cout << c<< " "; // a b

//Методу count\_for необходимо передать map с именами переменных и их значениями

std::map<char, double> mp = { {'a' , 5.0}, {'b' , 6.5} };

cout << p.count\_for(mp); //30.5

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были выявлены объекты предметной области и определена система классов для них. После объектно-ориентированного проектирования классы были реализованы на языке С++. Разработанный код был проверен на контрольных тестах и в код были внесены необходимые исправления. Для библиотеки была разработана документация, описывающая её установку и использование. Таким образом, цель работы была достигнута, задачи – решены.

Результаты работы можно использовать в процессе последующего обучения в форме навыков практического применения объектно-ориентированного подхода для разработки сложных программных систем, понимания порядка этапов разработки программного обеспечения и достигаемых на каждом этапе результатов.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Гамма, Э. Приемы объектно ориентированного проектирования. Паттерны проектирования. [Электронный ресурс] / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. — Электрон. дан. — М. : ДМК Пресс, 2007. — 368 с. — Режим доступа: http://e.lanbook.com/book/1220

2 Липман, С. Язык программирования С++. Полное руководство. [Электронный ресурс] / С. Липман, Ж. Лажойе. — Электрон. дан. — М. : ДМК Пресс, 2006. — 1105 с. — Режим доступа: http://e.lanbook.com/book/1216

3 . Полиномы. – URL: https://ru.wikipedia.org/wiki/Многочлен (дата обращения: 01.12.2021).

# ПРИЛОЖЕНИЕ А

### А.1 Файл polynome.hpp

#ifndef \_POLYNOME\_H\_

#define \_POLYNOME\_H\_

#include <iostream>

#include <string>

#include <vector>

#include <map>

#include <math.h>

// Класс мономов из которых состоит полином

class Monome {

private:

double coef; // Коэффициент монома

std::map<char, int> pows; // Переменные и их степени в формате map

public:

// Конструкторы

Monome():coef(1) {}

// Конструктор-преобразователь string -> Monome

Monome(std::string);

Monome(double num):coef(num) {} // double -> Monome

// Определение операций

Monome operator \*(const Monome &obj) const ;

Monome operator \*(double number) const;

Monome operator /(double number) const;

friend bool operator==(const Monome &obj1, const Monome &obj2);

// Ввод и вывод в поток

friend std::ostream &operator <<(std::ostream &out, const Monome &obj);

friend std::istream &operator >>(std::istream &in, Monome &obj);

// Метод для сравнения переменных + степеней

bool similar(const Monome &obj) const {return pows == obj.pows;}

friend class Polynome;

};

inline bool operator!=(const Monome &obj1, const Monome &obj2) {

return !(obj1 == obj2);

}

class Polynome {

private:

// Вектор содержащий мономы, на которые раскладывается полином

std::vector<Monome>terms;

public:

// Конструкторы вектор

Polynome() {}

// Конструктор преобразователь string -> Polynome

Polynome(std::string);

Polynome(double num);// double -> Polynome

Polynome(Monome num);// Monome -> Polynome

// Определение операций для работы с полиномами

Polynome operator +(const Polynome &obj) const;

Polynome operator -(const Polynome &obj) const;

Polynome operator \*(const Polynome &obj) const;

// Определение операции для деления на число Конструктор прео

Polynome operator /(double number) const;

// Ввод и вывод в поток

friend std::ostream &operator<<(std::ostream &out, const Polynome &obj);

friend std::istream &operator>>(std::istream &in, Polynome &obj);

// Получение переменных

std::vector<char> print\_variables();

// Вычисление полинома для значений

double count\_for(std::map<char, double> &mp);

};

#endif

### А.2 Файл polynome.cpp

#include "polynome.h"

#include <sstream>

#include <algorithm>

using namespace std;

Monome::Monome(string str) {

stringstream ss(str);

ss >> coef;

if (coef == 0) coef = 1;

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

if (isalpha(str[i])) {

if (str[i + 1] == '^') {

string number = "";

int k = i + 1;

while (isdigit(str[k + 1])) {

k++;

number += str[k];

}

pows[str[i]] += stoi(number);

}

else

pows[str[i]] = 1;

}

}

}

Monome Monome::operator\*(const Monome &obj) const {

Monome result = \*this;

result.coef = result.coef \* obj.coef;

for (auto it : obj.pows) {

if (result.pows.contains(it.first)) {

result.pows[it.first] =result.pows[it.first] + it.second;

}

else {

result.pows[it.first] = it.second;

}

}

return result;

}

Monome Monome::operator\*(double number) const {

Monome result = \*this;

result.coef = result.coef \* number;

return result;

}

Monome Monome::operator/(double number) const {

if (number == 0) throw exception();

Monome result = \*this;

result.coef = result.coef / number;

return result;

}

bool operator==(const Monome &obj1, const Monome &obj2) {

return obj1.coef == obj2.coef && obj1.pows == obj2.pows;

}

ostream &operator <<(ostream &out, const Monome &obj) {

out << obj.coef;

if (obj.pows.size() == 0) return out;

out << "\*";

int counter = 0;

for (auto it = obj.pows.begin(); it != obj.pows.end(); ++it) {

if (it->second == 1)

out << it->first;

else

out << it->first << "^" << it->second;

counter++;

if (counter != obj.pows.size())

out << "\*";

}

return out;

}

istream &operator >>(istream &in, Monome &obj) {

in >> obj.coef;

string str;

in >> str;

obj.pows.clear();

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

if (isalpha(str[i])) {

if (str[i + 1] == '^') {

string number = "";

int k = i + 1;

while (isdigit(str[k + 1])) {

k++;

number += str[k];

}

obj.pows[str[i]] += stoi(number);

}

else

obj.pows[str[i]] = 1;

}

}

return in;

}

Polynome::Polynome(string str) {

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

if (isdigit(str[i]) || str[i]=='-') {

string buf = "";

int k = i;

while (str[k] != '+' && k < str.size())

buf += str[k++];

Monome m(buf);

terms.push\_back(m);

i = k;

}

}

}

Polynome::Polynome(double num) {

terms.push\_back(num);

}

Polynome::Polynome(Monome obj) {

terms.push\_back(obj);

}

Polynome Polynome::operator/(double number) const {

if (number == 0) throw exception();

Polynome res = \*this;

for (auto &term : res.terms)

term.coef /= number;

return res;

}

Polynome Polynome::operator +(const Polynome &obj) const {

Polynome res = \*this;

bool flg;

for (auto term : obj.terms) {

flg=false;

for (auto &term2 : res.terms) {

if (term.similar(term2)) {

term2.coef += term.coef;

flg=true;

break;

}

}

if(!flg) res.terms.push\_back(term);

}

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

if (res.terms[i].coef == 0){

res.terms.erase(res.terms.begin()+i);

}

}

return res;

}

Polynome Polynome::operator -(const Polynome &obj) const {

Polynome res = \*this;

return res + obj\*(-1.);

}

Polynome Polynome::operator\*(const Polynome &obj) const {

Polynome result;

for (auto it1 : this->terms) {

for (auto it2 : obj.terms) {

result = result + (it1 \* it2);

}

}

return result;

}

ostream &operator<<(std::ostream &out, const Polynome &obj) {

int end = 0;

for (auto term : obj.terms) {

out << term;

end++;

if (end != obj.terms.size())

out << "+";

}

return out;

}

std::vector<char> Polynome::print\_variables(){

std::vector<char> v = {' '};

for (auto term: this->terms){

for (auto it : term.pows) {

if (\*std::find(v.begin(), v.end(), it.first) != it.first){

v.push\_back(it.first);

}

}

}

return v;

}

double Polynome::count\_for(std::map<char, double> &mp){

int allsum = 0;

for (auto term: this->terms){

int msum = term.coef;

for (auto it : term.pows) {

msum \*= std::pow(mp[it.first], it.second);

}

allsum += msum;

}

return allsum;

}