Федеральное государственное автономное образовательное учреждение

высшего образования

«Нижегородский государственный университет

им. Н.И. Лобачевского»

Институт информационных технологий, математики и механики

Отчёт по лабораторной работе

Арифметические операции с полиномами

Выполнил:

студент ИИТММ ПМИ гр. 381703-1

Боганов С. В.

Проверил:

Ассистент кафедры МОСТ ИИТММ

Волокитин В. Д.

Нижний Новгород

2018 г.

**Содержание**

[Введение](#Введение) 3

[Постановка задачи](#Постановка_задачи) 4

[Руководство пользователя](#Руководство_пользователя) 5

[Руководство программиста](#Руководство_программиста) 6

[Описание структур данных](#Описание_структур_данных) 6

[Описание алгоритмов](#Описание_алгоритмов) 7

[Описание структуры программы](#Описание_структуры_программы) 9

[Заключение](#Заключение) 11

[Литература](#Литература) 12

[Приложения](#Приложения) 13

**Введение**

Полиномы играют ключевую роль в математике. Техническая простота вычислений, связанных с полиномами, по сравнению с более сложными классами функций, способствовала развитию методов разложения в ряды в математическом анализе. Также они представляют собой довольно простые функции, поэтому их дифференцирование и интегрирование не составляют большого труда. Поэтому любую непрерывную функцию на заданном отрезке можно приблизить полиномом, что дает возможность анализировать поведение и характер функции, находящейся вблизи заданной точки. Кроме того, полиномы имеют большое значение в алгебраической геометрии.

**Постановка задачи**

Необходимо разработать и реализовать программу, выполняющую арифметические операции с полиномами трех переменных (x, y и z): сложение, вычитание, умножение на константу, умножение двух полиномов.

**Требования к программе:**

* Программа должна быть написана на языке C++.
* Полином должен быть составлен из мономов от трех переменных с ограничением на степень каждой переменой от 0 до 9.
* Коэффициентами полинома должны быть вещественные числа.
* В качестве структуры хранения полинома использовать список мономов с ненулевыми коэффициентами (односвязный или двухсвязный).
* Элементы списка хранить упорядоченными.
* Степень полинома хранить в "свернутом" виде, т. е. степень должна быть представлена как трехзначное число, где число сотен – это степень при переменной “x”, число десятков - степень при переменной “y”, число единиц - степень при переменной “z”.
* Сложение полиномов осуществлять алгоритмом слияния упорядоченных массивов.
* Вычитание полиномов допускается выполнять через сложение с умножением на константу (C = A - B = A + (-1)\*B)
* При умножении и сложении (вычитании) необходимо следить, чтобы в итоговом полиноме были приведены подобные слагаемые, и не хранилось мономов с нулевым коэффициентом.
* Если при умножении полиномов полученные степени переменных больше 9, выводить сообщение об ошибке.
* Считывание полинома у пользователя допускается в любом удобном виде. Необходимо предоставить пользователю правила ввода данных.
* Следует учесть, что пользователь может вводить полином, не упорядочив в нем мономы.
* Все функции должны тестироваться с помощью Google tests.
* Создание консольного интерфейса пользователя.

**Руководство пользователя**

После запуска программы, пользователь должен ввести 2 полинома, согласно стандартным правилам ввода. Вещественные числа в коэффициентах монома следует разделять точкой. Степени монома неотрицательные, не превышают 9. После чего пользователю предлагается на выбор одна из следующих операций: сложение, вычитание, умножение.

На выходе пользователь получает либо результат операции, либо сообщение об ошибке, связанной с переполнением степени при умножении.

**Руководство программиста**

**Описание структур данных**

**Polynomial.h**

**class Monom** содержит 4 поля:

* **double** coef - содержит коэффициент монома.
* **int** exp - степень монома в "свернутом" виде.
* **Monom** \*next - указатель на следующий моном.
* **Monom** \*prev - указатель на предыдущий моном.

**class Polynom** представляет собой список мономов исодержит 1 поле:

* **Monom** \*f\_head - указатель на фиктивную голову.

**Polynom** a - временная переменная.

**Main.cpp**

**Polynom** a, b - переменные для хранения полиномов a и b соответственно.

**int** input - переменная для выбора одной из операций.

**Polynomial.cpp**

**Monom** \*temp, \*a, \*b, \*f, \*s - временные указатели.

**Polynom** a - временная переменная.

**int** x, y, z - временные переменные для хранения степеней x, y и z соответственно.

**double** coef - временная переменная для хранения коэффициента.

**vector<pair<double, int>>** v - вектор для хранения пар коэффициентов и степеней.

**pair<double, int>** temp - пара для хранения коэффициента и степени.

**Описание алгоритмов**

Код функций см. в приложении.

**Polynomial.cpp**

Функция упрощения полинома

1. Создадим указатель, который будет указывать на первый элемент после фиктивный головы.
2. Пока созданный указатель не будет указывать на ноль или указатель на следующий моном не будет указывать на ноль, выполним следующие действия:
   1. Смотрим на моном, на который указывает созданный указатель: если его степень оказалась равна степени следующего за ним монома, то сложим их коэффициенты.
      1. Если в результате сложения получился нулевой коэффициент, то удалим оба монома.
      2. Если в результате сложения получился ненулевой коэффициент, то удалим второй моном, а первому присвоим этот коэффициент.
   2. Переместим указатель на следующий моном.

Сложение двух полиномов

1. Создадим полином для хранения ответа, и 2 указателя: первый указатель должен указывать на начало первого элемента после фиктивной головы первого полинома, второй указатель - второго полинома.
2. Смотрим на моном, на который указывает первый указатель: если его степень оказалась больше степени монома, на который указывает второй указатель, добавим этот моном в ответ и переместим первый указатель на следующий элемент. Если же степень оказалась меньше, добавим второй моном в ответ и переместим второй указатель.
3. Если в ходе перемещения указателей случилась такая ситуация, что один из указателей стал указывать на ноль, то добавим к ответу все мономы, начиная с того, на который указывает другой указатель.
4. Вызовем функцию упрощения полинома.
5. Вернем получившийся полином.

Вычитание двух полиномов

1. Прибавим к первому полиному второй, умноженный на “-1”, и вернем результат.

Умножение полинома на константу

1. Создадим пустой полином.
2. Если константа равна нулю, вернем этот полином.
3. Если константа не равна нулю, создадим указатель, который будет указывать на начало первого элемента после фиктивной головы.
4. Умножим коэффициент монома, на который указывает этот указатель, на константу, и переместим его на следующий элемент.
5. Вернем получившийся полином.

Умножение двух полиномов

1. Создадим полином для хранения ответа, и 2 указателя: первый указатель должен указывать на начало первого элемента после фиктивной головы первого полинома, второй указатель - второго полинома.
2. Пока первый указатель не будет указывать на ноль, выполним следующие действия:
   1. Переместим второй указатель на первый элемент после фиктивной головы второго полинома
   2. Пока второй указатель не будет указывать на ноль, выполним следующие действия:
      1. Создадим временные переменные для хранения сумм степеней x, y, z.
      2. Сложим степени x, y, z мономов, на которые указывают два указателя.
      3. Если одна из полученных степеней получилась больше 9, выдадим сообщение об ошибке.
      4. Создадим временную переменную для хранения нового коэффициента.
      5. Перемножим коэффициенты мономов, на которые указывают два указателя.
      6. Соберём степень монома в “свернутый вид”.
      7. Добавим в ответ полученный моном.
      8. Переместим второй указатель на следующий элемент.
   3. Переместим первый указатель на следующий элемент.
3. Вызовем функцию упрощения полинома.
4. Вернем получившийся полином.

**Описание структуры программы**

Программа состоит из 4-х проектов: **Example**, **gtest**, **Polynomial** и **tests**.

Проект **gtest** включает в себя 2 файла gtest.h и gtest-all.cc, которые содержат реализацию функционала google tests.

Проект **tests** включает в себя 2 файла: test\_main.cpp, test\_polynom.cpp.

* Файл test\_main.cpp запускает все google tests.
* Файл test\_arithmetic.cpp содержит тесты для методов класса Polynom.

Проект **Example** включает в себя файл Main.cpp, который содержит реализацию пользовательского приложения, предназначенного для выполнения арифметических операций с полиномами трех переменных.

Проект **Polynomial** включает в себя 2 файла: Polynomial.h и Polynomial.cpp.

* Файл Polynomial.h содержит 2 класса: Monom и Polynom.

Класс Monom не содержит методов.

Класс Polynom содержит 21 метод:

* + 1. Конструктор по умолчанию
    2. Конструктор инициализации
    3. Конструктора копирования
    4. Деструктор
    5. Метод для сложения двух полиномов
    6. Метод для вычитания двух полиномов
    7. Метод для умножения полинома на константу
    8. Метод для умножения двух полиномов
    9. Метод для сравнения двух полиномов на равенство
    10. Метод для сравнения двух полиномов на неравенство
    11. Метод для проверки полинома на пустоту
    12. Метод для присваивания одного полинома в другой
    13. Приватный метод для упрощения полинома (приведение подобных слагаемых)
    14. Приватный метод вставки монома после последнего
    15. Метод вставки монома с упорядочением
    16. Метод поиска монома с его возвратом
    17. Метод поиска монома на наличие
    18. Метод удаления монома
    19. Метод очистки полинома (удаление всех мономов)
    20. Ввод полинома
    21. Вывод полинома
* Файл Polynomial.cpp содержит реализацию методов класса Polynom.

**Заключение**

Достигнутые результаты:

* Разработана и реализована программа, выполняющая арифметические операции с полиномами трех переменных (x, y и z): сложение, вычитание, умножение на константу, умножение двух полиномов.
* Программа написана на языке C++.
* Полином составлен из мономов от трех переменных с ограничением на степень каждой переменной от 0 до 9.
* Коэффициенты полинома ­- вещественные числа.
* В качестве структуры хранения полинома использован двухсвязный список мономов с ненулевыми коэффициентами.
* Элементы списка хранятся в упорядоченном виде.
* Степень полинома хранится в "свернутом" виде.
* Сложение полиномов осуществляется при помощи алгоритма слияния упорядоченных массивов.
* Вычитание полиномов реализовано через сложение с умножением на константу (C = A - B = A + (-1)\*B)
* При умножении и сложении (вычитании) в итоговом полиноме приводятся подобные слагаемые, и не хранятся мономы с нулевым коэффициентом.
* Если при умножении полиномов полученные степени переменных больше 9, выводится сообщение об ошибке.
* Пользователю предоставлено правило ввода данных.
* Пользователь может вводить полином, не упорядочив в нем мономы.
* Все функции тестируются с помощью Google tests.
* Создан консольный интерфейс пользователя.

**Литература**

1. Многочлен

[https://ru.wikipedia.org/wiki/Многочлен]

**Приложения**

Сложение двух полиномов

Polynom Polynom::operator+(const Polynom &p) const

{

Polynom a;

Monom \*f = f\_head->next, \*s = p.f\_head->next;

while (true)

{

if (f == NULL && s != NULL)

{

while (s != NULL)

{

a.ins\_last(s->coef, s->exp);

s = s->next;

}

break;

}

else if (f != NULL && s == NULL)

{

while (f != NULL)

{

a.ins\_last(f->coef, f->exp);

f = f->next;

}

break;

}

else

{

if (f->exp >= s->exp)

{

a.ins\_last(f->coef, f->exp);

f = f->next;

}

else

{

a.ins\_last(s->coef, s->exp);

s = s->next;

}

}

}

simplification(a);

return a;

}

Вычитание двух полиномов

Polynom Polynom::operator-(const Polynom &p) const

{

return \*this + p \* (-1.0);

}

Умножение полинома на константу

Polynom Polynom::operator\*(double val) const

{

Polynom a;

if (!(abs(val) < EPS))

{

a = \*this;

Monom \*temp = a.f\_head->next;

while (temp != NULL)

{

temp->coef = temp->coef \* val;

temp = temp->next;

}

}

return a;

}

Умножение двух полиномов

Polynom Polynom::operator\*(const Polynom &p) const

{

Polynom a;

Monom \*f = f\_head->next, \*s = p.f\_head->next;

while (f != NULL)

{

s = p.f\_head->next;

while (s != NULL)

{

int x, y, z;

x = s->exp / 100 + f->exp / 100;

y = s->exp / 10 % 10 + f->exp / 10 % 10;

z = s->exp % 10 + f->exp % 10;

if ((x > 9) || (y > 9) || (z > 9))

{

throw (string)"Overflow";

}

double coef = s->coef \* f->coef;

int exp = x \* 100 + y \* 10 + z;

a.ins(coef, exp);

s = s->next;

}

f = f->next;

}

simplification(a);

return a;

}