МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

Высшего образования

**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**Отчет по учебной практике**

**«Название лабораторной»**

**Выполнил:**

студент группы 381706­1

Карин Тимофей Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Проверил:**

ассистент каф. МОСТ ИИТММ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc536219868)

[3. Руководство пользователя 5](#_Toc536219869)

[4. Руководство программиста 6](#_Toc536219870)

[4.1. Описание структуры программы 6](#_Toc536219871)

[4.2 Описание структур данных 6](#_Toc536219872)

[4.3 Описание алгоритмов 6](#_Toc536219873)

[5. Заключение 7](#_Toc536219874)

[6. Список литературы 8](#_Toc536219875)

1. Введение

Полиномом (или многочленом) от n переменных называется сумма мономов. Полином можно представить в виде: , где С – коэффициент, – переменные, – степени переменных. Слагаемых в полиноме может быть столько угодно, главное, чтобы не было двух слагаемых с одинаковыми степенями всех переменных. Сами слагаемые являются мономами.

Как представить полином в памяти компьютера? Прежде всего надо представить в памяти моном. Для этого отдельно будем хранить коэффициент C, количество переменных n, массив, хранящий степени переменных, и указатель на следующий моном. Таким образом, можно сделать вывод, что полином будет представлять из себя список, элементами которого являются мономы.

Как определяется очерёдность мономов в полиноме? Известно, что от перестановки мест слагаемых, сумма не меняется. Но для работы с полиномом, добавления в него новых мономов, да и просто для наглядности всё же важна очерёдность расположения мономов. В связи с этим мономы располагаются в лексикографическом порядке, то есть ближе к началу стоит тот моном, у которого степень первой переменной больше. Если же степени первых переменных у нескольких мономов равны, то поочерёдно проверяются следующие переменные, и ближе к началу ставится тот, у которого первая различающаяся степень больше.

В данной лабораторной работе будут рассмотрены полиномы с неотрицательными степенями переменных.

2. Постановка задачи

Целью данной лабораторной работы является создание библиотеки для работы с мономами и полиномами. Для этого необходимо:

* Создать класс для работы с мономами;
* Создать класс для работы с полиномами;
* Проверить основные методы классов;
* Написать программу, демонстрирующую основные функции созданной библиотеки.

3. Руководство пользователя

Библиотека не предназначена для работы пользователей. В связи с этим, была написана небольшая программа, в которой пользователь может добавлять в полином мономы и выводить этот полином на экран.

В начале работы пользователю необходимо выбрать наибольшее количество переменных, которые будут присутствовать в полиноме. Затем будет создан пустой полином, в который пользователь может добавлять мономы. Чтобы создать отдельный моном, пользователю необходимо сначала ввести коэффициент, стоящий в начале монома. Затем, поочерёдно ввести степени всех переменных в мономе, начиная с x0. Если переменная отсутствует в мономе, то её степень равна нулю.

После ввода монома программа выведет его на экран, а также выведет и полином с уже добавленным мономом. Затем пользователю нужно выбрать один из двух вариантов: продолжить ввод мономов или выйти из программы. Для выбора нужно ввести соответствующее значение.

4. Руководство программиста

4.1. Описание структуры программы

Программа состоит из 3 основных модулей:

* *PolynomLib –* представляет собой библиотеку для работы с мономами и полиномами;
* *Polynom –* пример использования программы;
* *PolynomTest –* тесты для проверки работоспособности библиотеки с использованием фреймворка Google Test.

Кроме основных модулей в программе также используется модуль с исключениями *Exception.*

4.2 Описание структур данных

Рассмотрим поля и методы класса *TMonom* модуля *PolynomLib*:

Со спецификатором доступа protected:

* *TMonom\* next* –указатель на следующий моном;
* *unsigned\* power –* степени;
* *double c* – коэффициент;
* *const int n* – кол-во переменных;

Со спецификатором доступа public:

* *TMonom(int \_n, unsigned\* \_power, double c)* – конструктор;
* *virtual ~TMonom()* – деструктор;
* *TMonom(const TMonom& monom)* – конструктор копирования;
* *TMonom\* GetNext()* – возвращает указатель на следующий моном;
* *unsigned\* GetPower() –* возвращает степени переменных;
* *double GetC()* – возвращает коэффициент;
* *const int GetN() –* возвращает кол-во переменных;
* *void SetNext(TMonom\* \_next)* – присваивает полю *next* значение *\_next*;
* *void SetPower(unsigned\* \_power)* – присваивает степеням переменных значения из массива *\_power;*
* *void SetC(double \_c)* – устанавливает коэффициент;
* *TMonom& operator =(const TMonom& monom)* – оператор присваивания;
* *TMonom operator +(TMonom& monom)* – оператор сложения двух мономов;
* *TMonom& operator+=(TMonom& monom);*
* *TMonom operator -(TMonom& monom);*
* *TMonom operator\*(TMonom& monom)const;*
* *TMonom operator\*(int a)* – оператор умножения монома на число;
* TMonom& operator \*=(TMonom& monom);
* *bool operator ==(TMonom& monom)* – проверка на равенство степеней;
* bool operator >(TMonom& monom);
* bool operator <(TMonom& monom);
* friend std::istream& operator >> (std::istream& \_s, TMonom& Tm) – функция ввода;
* friend std::ostream& operator << (std::ostream& \_s, TMonom& Tm) – функция вывода;

Рассмотрим поля и методы класс *TPolynom*:

Со спецификатором доступа protected:

* *TMonom \*start –* указатель на первый моном в полиноме;
* *int n -* количество переменных в полиноме и в каждом из его мономов.

Со спецификатором доступа public:

* *TPolynom(int \_n) –* конструктор;
* *TPolynom(TPolynom &p) –* конструктор копирования;
* *virtual ~TPolynom() –* деструктор;
* *TMonom\* GetStart()* – возвращает указатель на первый моном;
* *TPolynom& operator+=(TMonom &m) –* оператор сложения полинома и монома *m*;
* *TPolynom& operator=(const TPolynom &p) –* оператор присваивания;
* *TPolynom operator+(TPolynom &p) –* операторсложения полиномов;
* *bool operator ==(const TPolynom& p) –* проверка на равенство двух полиномов (без учёта коэффициентов)
* *friend ostream& operator << (ostream& ostr, TPolynom& p) -* функция вывода полинома.

4.3 Описание алгоритмов

Сложение полинома и монома

Прежде всего проверяем количество переменных в полиноме и мономе. Если оно не совпадает, то выбрасываем исключение. Также проверяем коэффициент монома, если он нулевой, то получается, что добавлять ничего не надо и можно просто вернуть полином без изменений. Ещё нужно проверить полином на пустоту, т.к. если он пустой, то можно просто создать в нём добавляемый моном и на этом всё.

Если же коэффициент не нулевой, да и моном к тому же не пустой, то дела обстоят немного хуже. В этом случае нужно отдельно рассмотреть вариант, когда моном больше первого члена полинома (то есть тот случай, когда моном нужно добавить в начало). Просто создаём копию монома, в качестве указателя на следующий записываем указатель на начало полинома. После этого в указатель на начало записываем адрес созданной копии монома. Ещё нужно рассмотреть похожий случай, когда моном равен первому члену полинома. Если такое произошло, то нужно просто сложить эти два монома.

Если и этот случай не подходит, то всё намного хуже. Если предыдущие варианты не подошли, то моном нужно либо добавить к какому-то члену полинома, либо вставить между двумя членами, либо добавить в конец. Для этого поочерёдно будем рассматривать последовательные пары членов полинома. Если моном равен второму члену из пары, то складываем их. Если моном больше чем второй член, то получается, что моном меньше первого члена (иначе он был бы добавлен на предыдущей итерации) и больше второго. То есть, моном нужно вставить между ними. Для этого копируем моном, в указатель на следующий элемент записываем адрес второго члена пары, а адрес копии кладём в указатель первого члена пары. И наконец, если после просмотра всех пар места для монома так и не нашлось, то его нужно добавить в конец полинома, а его адрес записать в указатель последнего монома.

В любом случае сразу после добавления монома полином необходимо сразу же вернуть из функции.

Сложение двух полиномов

Проверяем количество переменных в полиномах. Если они не равны, ты выбрасываем исключение. Создадим полином с таким же количеством переменных, как и в слагаемых. 3 указателя: один на текущий элемент первого полинома, на текущий элемент второго полинома и на текущий элемент полинома суммы. В самом начале текущим является первый элемент.

Теперь нужно найти моном, который будет добавлен в полином суммы. Создаём пустой моном. Смотрим, что хранится по указателям слагаемых. Если мономы первого и второго полиномов равны, то складываем их, результат записываем в созданный пустой моном. Оба указателя передвигаем на следующий моном. Если же какой-то из мономов больше, то его и добавляем в созданный пустой моном. В этом случае указатель нужно сдвигать только у того полинома, из которого взяли значение.

Моном найден. Теперь нужно найти ему место в полиноме суммы. Если полином пустой, то адрес монома записываем в указатель на начало. Если полином не пустой, то адрес добавляемого монома помещаем в указатель того монома, адрес которого хранится в созданном в самом начале указателя (то есть в указатель последнего добавленного элемента). В обоих случаях указатель передвигаем на добавленный элемент.

Добавление происходит до тех пор, пока не закончится один из полиномов. После того, как это произойдёт, нужно просто переписать остаток ещё не закончившегося полинома в полином суммы и вернуть полученный полином.

5. Заключение

В ходе проделанной работы были получены следующие результаты:

* Созданы классы для работы с мономами и полиномами, которые объединены в одну библиотеку;
* Создан пример использования программы, позволяющий пользователю создавать полином и управлять им;
* Обработаны основные исключительные ситуации;
* Проверено большинство методов с помощью фреймворка Google Test.

6. Список литературы

* Ахо Альфред В, Хопкрофт Джон Э и Ульман Джеффри Д Структуры данных и алгоритмы [Книга]. - [б.м.] : Вильямс, 2003.
* Лафоре Роберт Структуры данных и алгоритмы в Java [Книга]. - СПб : Питер, 2013. - 2 : стр. 704.
* Многочлен [В Интернете] // Wikipedia. - Wikimedia Foundation, Inc, 29 декабрь 2018 г.. - 16 февраль 2019 г.. - https://ru.wikipedia.org/wiki/Многочлен.
* Павловская Т. А. C/C++ Программирование на языке высокого уровня [Книга]. - СПб : Питер, 2003.
* Страуструп Бьерн Язык программирования C++ [Книга]. - [б.м.] : Бином, 2004.