Федеральное государственное автономное образовательное учреждение

высшего образования

«Нижегородский государственный университет

им. Н.И. Лобачевского»

Институт информационных технологий математики и механики

Отчет по лабораторной работе

Арифметические операции с полиномами

Выполнил:

Студент института ИТММ

гр. 381703-1

Сизов К.Д

Проверил:

Ассистент кафедры МОСТ

Института ИТММ

Волокитин В.Д

Нижний Новгород

2019 г.

Оглавление

[Введение 3](#_Toc2745653)

[Постановка задачи 4](#_Toc2745654)

[Руководство пользователя 5](#_Toc2745655)

[Руководство программиста 6](#_Toc2745656)

# Введение

Во многих областях науки нередко приходится решать некую вычислительную задачу. Конкретная задача решается с использованием некоего математического аппарата. Как правило составляется некая математическая модель, которая способны ответить на интересующие исследователя вопросы. Внутри математической модели могут использоваться совершенно различные инструменты, походы и методы. Сталкиваясь с математикой так или иначе возникают те или иные математические объекты с которыми нужно уметь работать. Одним из фундаментальных и наиболее часто встречающемся математическим объектом являются полином. Полином это выражение вида:

Полиномы возникают во многих задачах математики, которые в свою очередь появляются в прикладных исследованиях. Например, полиномы возникают при разложении функции в ряд Тейлора, при нахождении собственных чисел, при решении дифференциальных уравнений и т.д. Именно в связи с таким широким спектром применения возникает потребность в разработке многофункционального, полноценного программного обеспечения, которое бы позволило работать с полиномами, которые даже содержат такое число слагаемых, которое не получится выписать на бумаге.

# Постановка задачи

Разработать программу, выполняющую арифметические операции с полиномами трех переменных (x, y и z): сложение, вычитание, умножение на константу, умножение двух полиномов. Считается, что полином составлен из мономов от трех переменных с ограничением на степень каждой переменой от 0 до 9 (Опционально можно расширить данное ограничение). Коэффициенты полинома - вещественные числа. Работоспособность программы необходимо проверить с помощью Google Test-ов. Кроме того, необходимо разработать пользовательское консольное приложение. При умножении и сложение (вычитание) необходимо следить, чтобы в итоговом полиноме были приведены подобные слагаемые и не хранилось мономов с нулевым коэффициентом. В качестве структуры хранения полинома использовать список мономов с ненулевыми коэффициентами (односвязный или двухсвязный).

# Руководство пользователя

Программа предоставляет возможность выполнять арифметические действия с полиномами от трех переменных (x, y, z). Для того, чтобы приступить к работе с полиномами, на первом этапе после запуска приложения (sample.exe), необходимо выбрать способ задания полиномов, программа предоставляет на выбор два способа ввода:

* Ручной ввод. Предполагается, что пользователь сначала задает количество мономов в полиноме. А затем поочередно введет значения мономов. Значения мономов задаются в формате: <коэффициент>, <свертка степени>. Под сверткой степени подразумевается целое трехзначное число, у которого первая цифра (кол-во сотен) указывает на степень переменной x в мономе, вторая цифра (кол-во десятков) на степень y, третья цифра (кол-во единиц) на степень z. Например для монома сверткой степени будет являться число 479. Вводить мономы упорядоченно (в порядке убывания степени) не обязательно.
* Чтение полиномов из файла. Для того, прочесть полином из файла. Нужно внести значение полинома в файл “polinoms.txt” в формате аналогичному формату для ручного ввода, т.е. сначала задается количество мономов, затем их значение. Сам файл располагается в папке проекта программы.

При задании полиномов не допускается ввод некорректных значение, в категорию «некорректные значения» входят следующие входные значения.

* Любое не числовое значения там, где требуется ввести числовое.
* Числовое значение свертки степени, которое меньше 0 или же больше 999.
* Не целое числовое значение для свертки степени.
* Любое числовое значение отличное от тех, которые предлагаются на выбор пользовательским приложением.

Если программе будет передано некорректное значение, то она предложит пользователю произвести ввод полинома заново.

После задания полиномов пользователю предлагается произвести над ними арифметические действия, на выбор предоставляется одно из трёх действий.

* Сложение
* Умножение
* Вычитание

После выбора арифметического действия на экран выводится его результат. В случае, если результат умножения полиномов содержит моном степени выше или равное 10 программа выдает ошибку.

# Руководство программиста

## Описание структуры программы

Программа состоит из следующих модулей:

* Exception. Данный модуль состоит из одного заголовочного файла “exception.h” и содержит объявление и реализацию одноименного класса, который служит для обработки ошибок программы с использованием конструкции try-catch. Данный модуль не зависит от других модулей программы. И с целью возможности работы со строками в модуле подключены библиотеки <iostream> и <string>.
* Monom. Данный модуль состоит из одного заголовочного файла “monom.h” и содержит объявление и реализацию одноименного класса Monom. В модуле присутствует макрос EPS, который служит для задания погрешности с которой число с плавающей запятой приравнивается к 0. В модуль подключена пользовательская библиотека “exception.h”, для обработки ошибок.
* TLink. Модуль состоит из одного файла “tlink.h” и содержит объявление и реализацию шаблонного класса Tlink<T>. В модуле отсутствует подключение других библиотек и содержится один макрос PTLink, который был создан с целью увеличения читаемости кода.
* TList. Модуль содержит один файл “tlist.h”, в прописано объявление и реализация шаблонного класса TList<T>. Подключаемые пользовательские библиотеки: “exception.h” для обработки ошибок, “tlink.h” для реализации класса TList<T>. Так же как в модуле Tlink с целью улучшения качества кода был создан макрос PTList.
* Polinom. Модуль состоит из одного файла “polinom.h” содержит реализацию основного класса Polinom, который является потомком класса TList<T>. В модуле подключатся все описанные выше модули.
* Sample. Данный модуль отвечает за пользовательское приложение и состоит из одного файла “sample.cpp”. В нем содержится функция main() и реализация демонстрации функционала всей программы по работе с полиномами.

## Описание структур данных

1. Exception. Класс Exception является классом исключений. Объектами данного класса являются некие исключения. Данные о произошедшей исключительной ситуации записываются в приватное поле name\_error в виде строке с сообщением об ошибке, запись этой строки производится на этапе создания объекта класса Exception. Класс помимо конструктора ещё также содержит метод GetNameError (), возвращает строку с сообщением об ошибке.
2. Monom. Класс Monom представляет собой моном от трёх переменных (x, y, z) в его математическом понятии. Моном имеет две основные характеристики: коэффициент и степень, в связи с этим класс содержит следующие поля в секции private:
   * Коэффициент монома, который в общем случае может принимать любые действительные значения, поэтому тип хранения коэффициента был выбран double.
   * Степень монома. Степень хранится в свернутом виде и принимает целочисленное значение типа int. На это значение вводится строгое ограничение (которое контролируется в конструкторе): оно не может быть отрицательным или больше 999.
   * Сумма степеней при различных переменных (x, y, z) или другими словами сумма цифр свертки степени. Значение этого поле никак не может быть проинициализировано вне класса и вычисляется единожды при создании объекта типа Monom. Данное поле является вспомогательным и используется при реализации арифметических действий над мономами.

В классе Monom реализованы все функции, необходимые для реализации функционала выполнения арифметических действий над полиномами, а именно:

* Set/Get функции позволяют задавать (кроме поля суммы) и получать значения полей монома.
* Перегрузка операторов сравнения. Для возможности сравнения и упорядочивания мономов.
* Перегрузка бинарных арифметических операторов. Конкретно плюс-равно, минус-равно, умножение на скаляр, сложение, вычитание, умножение на моном (аргументом в этих методах выступает моном, если не указано иного).
* Перегрузка операторов потокового ввода/выводы для возможности ввода/ вывода значения монома.

1. TLink. Шаблонный класс звена списка. Класс содержит два поля в секции private:
   * Данные, тип которых определяется параметром шаблона.
   * Указатель на объект того же класса TLink<T>

Класс содержит все необходимые конструкторы (по умолчанию, с параметром, копирования), а так же для обеспечения возможности в доступу к приватным полям данного класса он является дружественным классом по отношению к шаблонному классу TList<T> .

1. TList. Шаблонный класс, реализующий односвязный список с фиктивной головой. Класс содержит одно поле в секции private – это указатель на объект класса TLink<T>, который несет в себе смысл фиктивной головы для односвязного текста. В классе определены и реализованы все необходимые методы:
   * Конструкторы. Для класса TList<T> был реализован конструктор по умолчанию, и конструктор копирования. В конструкторе по умолчанию осуществляется выделение памяти под фиктивную голову. В конструкторе же копирование копируется все звенья из одного списка и вставляются в другой с сохранением порядка следования, при этом каждый из списков имеет свою собственную память.
   * Метод проверки на пустоту. Функция возвращающая переменную типа bool. Проверка на пустоту необходима для реализации других методов класса.
   * Вставка в список. Метод вставки в список принимает в качестве аргумента переменную типа T, который определяется как параметр шаблона. Звено типа TLink<T> с полем значения равным значению аргумента вставляется в список по убыванию. Сравнение элементов списка при этом производится по значению типа T.
   * Удаление звена из списка. Метод удаления в качестве аргумента принимает указатель на удаляемое звено, если данное звено в списке присутствует, то производится его удаление, если такого звена в списке нет, то никаких действий не производится.
   * Слияние списков. Метод слияния списков реализует алгоритм слияния двух упорядоченных списков.
2. Polinom. Класс Polinom наследуется от класса TList<Monom> со спецификатор наследования public. В классе реализованы все методы, которые необходимы для выполнения все арифметических операций с полиномами. Класс реализован таким образом, что мономы в полиноме всегда хранятся в упорядоченном виде и также отсутствуют нулевые мономы. Конкретно были реализованы следующие методы:
   * Конструктор инициализации. Конструктор инициализации принимает два параметра: массив мономов, количество мономов в полиноме. Реализован же он с помощью конструктора по умолчанию и метода Insert, принадлежащих классу TList<Monom>.
   * Метод приведения подобных слагаемых. Метод складывает (с помощью реализованного метода перегрузки оператора сложения) мономы полинома с одинаковыми степенями и убирает из полинома мономы нулевыми коэффициентами.
   * Перегрузка оператора сравнения полиномов.
   * Перегрузка бинарных операторов. Конкретно плюс-равно, умножение-равно, плюс, минус, умножение на скаляр, умножение на моном, умножение на полином (аргументом в этих методах является полином, если не указано иного). Именно перегрузка этих операторов и реализует полый функционал для выполнения арифметических действий над полиномами.
   * Перегрузка операторов потокового ввода/выводы для возможности ввода/ вывода значения полинома.

## Описание алгоритмов

1. Вставка в список.
   1. Вход: список, значение, которое вставляем в список.
   2. Создаем звено с заданными значением, которое будем вставлять.
   3. Если список пустой, вставляем звено сразу после фиктивной головы. Конец.
   4. Иначе ищем место для вставки нового звена, сравнивая звено по значению со всеми звеньями списка, по порядку убывания.
   5. Вставляем звено на найденное место. Конец.
   6. Выход: список со вставленным новым звеном.
2. Удаление из списка.
   1. Вход: список, указатель на удаляемое звено списка.
   2. Ищем звено по заданному указателю на него.
   3. Если звено было найдено, то удаляем его из списка. Конец.
   4. Выход: список с удаленным звеном.
3. Слияние упорядоченных списков.
   1. Вход: список1, список2.
   2. Создаем указатель1 и указатель2, которые указывают на первое звено списка1 и списка2 соответственно.
   3. Пока не закончился список1 или список2.
      1. Сравниваем значение звеньев на которые указывают указатель1 и указатель2.
      2. Большее из звеньев добавляем в результирующий список.
      3. Переставляем вперед (на следующие звено) указатель, который указывал на большие звено.
   4. Если указателей1 (или указатель 2) указывает на конец списка1 (списка2), до пока указатель2 (указатель1) не указывает на конец списка2 (списка1), добавляем соответствующие элементы списка2 (списка1) в результирующий список.
   5. Выход: упорядоченный список содержащий все элементы списка1 и списка2.
4. Умножение мономов.
   1. Вход: моном1, моном2.
   2. Создаем результирующий моном.
   3. Складываем степени моном1 и моном2 и записываем в степень результирующего монома.
   4. Если сумма цифр свертки степени результирующего монома не равна сумме сумм цифр сверток степени монома1 и монома2, тогда выводим сообщение об ошибке (так как программа работает только для мономов степени не выше 9). Конец.
   5. Иначе перемножаем коэффициенты моном1 и моном2, результат записываем в результирующий моном.
   6. Выход: результирующий моном.
5. Приведение подобных слагаемых полинома.
   1. Вход: полином1.
   2. Сравниваем все соседние мономы полинома1.
   3. Цикл по всем мономам полинома1.
      1. Если степень монома совпадает со степенью следующего за ним монома, то складываем их, и если коэффициент их суммы не равен нулю, то вставляем в результирующий полином.
      2. Или если коэффициент монома равен 0, то удаляем этот моном.
      3. Иначе вставляем моном в результирующий.
   4. Выход: результирующий полином который равен полиному1 в приведённом виде.
6. Сложение полиномов.
   1. Вход: полином1, полином2.
   2. Создаем результирующий полином0.
   3. Сливаем полином1 и полином2 в полином0 с помощью функции слияния упорядоченных списков.
   4. Приводим подобные слагаемые в полином0.
   5. Выход: полином0, который равен сумме полином1 и полином2.
7. Умножение полинома на моном.
   1. Вход: полином1, моном1.
   2. Создаем пустой результирующий полином0.
   3. Цикл по всем мономам полинома1.
      1. Умножаем моном полинома1 на моном1 и результат добавляем в полином0.
   4. Приводи подобные слагаемые в полином0.
   5. Выход: полином0, который равен произведению полинома1 на моном1.
8. Умножение полиномов
   1. Вход: полином1, полином2.
   2. Создаем пустой результирующий полином0.
   3. Цикл по всем мономам полинома1.
      1. Умножаем моном полинома1 на полином2, результат добавляем в полином0.
   4. Приводим подобные слагаемые в полинм0.
   5. Выход: полином0, который равен произведению полинома1 и полинома2.

# Заключение

По результату выполнения лабораторной работы была написана программа, которая позволяет выполнять арифметические операции над полиномами от трех переменных, мономы которых не выше степени 9. Программа удовлетворяет постановке задачи и работает с соблюдением всех критериев. Были реализованы все необходимые алгоритмы включая алгоритм слияния упорядоченных список, алгоритм вставки в список, умножение полиномов и т.д. Был реализован шаблонный класс TList<T>, который реализует структуру хранения данных в виде односвязного списка. Весь функционал программы был протестирован с помощью библиотеки для модульного тестирования Google C++ Testing Framework. Так же было разработано демонстрационное приложение показывающие доступный функционал программы.

# Список литературы

1. Гергель В.П. Рабочие материалы преподавателя по общему курсу “Методы программирования” – Нижний Новгород – 2002. – 100 с.
2. Бьёрн Страуструп. Язык программирования C++ — 3-е изд. — СПб.; М.: Невский диалект — Бином, 1999. — 991 с.