МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Отчет по лабораторной работе**

**«Таблицы»**

**Выполнила:**

студент группы 3822Б1ФИ2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кошкин Н.А.

**Проверила:**

преподаватель каф. МОСТ ИИТММ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Арисова А.Н.

Нижний Новгород  
2024

**Содержание**

[**Содержание** 2](#_Toc104219358)

[Введение 3](#_Toc104219359)

[**1** Постановка задачи 4](#_Toc104219360)

[**2**. Руководство пользователя 5](#_Toc104219361)

[**3.** Руководство программиста 9](#_Toc104219362)

[**3.1.** Описание структуры программы 9](#_Toc104219363)

[**3.2.** Описание структур данных 10](#_Toc104219364)

[**3.3.** Описание алгоритмов 12](#_Toc104219365)

[3.3.1. Хэш-таблица 12](#_Toc104219366)

[3.3.2. Упорядоченная таблица 12](#_Toc104219367)

[3.3.3. Неупорядоченная таблица 13](#_Toc104219368)

[**4.** Проверка корректности 15](#_Toc104219369)

[Заключение 16](#_Toc104219370)

[Список литературы 17](#_Toc104219371)

# Введение

В лабораторной работе рассматривается вопрос разработки приложения для хранения полиномов в таблицах трёх типов (неупорядоченная таблица, упорядоченная таблица и хеш-таблица).

С целью реализации приложения в процессе выполнения данной лабораторной работы подробно описываются используемые структуры данных, а также рассматриваются и разбираются методы и алгоритмы решения основных задач (основанных на структуре данных таблиц (неупорядоченная, упорядоченная, хеш-таблица)), вытекающих из вопроса взаимодействия с хранящимися данными: добавление в таблицу нового полинома с уникальным ключом, удаление полинома из таблиц по ключу, поиск по ключу в таблицах всех трех типов.

# Постановка задачи

В данной лабораторной работе необходимо разработать программу, которая предоставляет возможность хранить данные в таблицах 3-х типов:

1)Неупорядоченная таблицы

2)Упорядоченная таблица

3)Хэш-таблица

**Особенности реализации:**

Работа происходит сразу с таблицами всех типов.

Пользователь может положить в таблицу свой полином, с заданием ему некоторого наименования (ключа).

Пользователь может запросить (найти в/извлечь из) таблицы полином с заданным наименованием.

В ходе выполнения программы у пользователя должна оставаться возможность производить операции с полиномами, а так же размещать полученный результат в таблицу.

Во время работы программы происходит логирования количества произведенных операций. Пример: Пользователь ввел свой полином и запросил положить его в таблица, программа в ответ выводит на экран или в файл, какое действие произошло и сколько операций потребовалось на нее у разных типов таблиц. Аналогично с поиском и извлечением.

В качестве данных используются полиномы из предыдущей лабораторной работы, её суть заключалась: разработке программы, выполняющей 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. Коэффициенты полинома - вещественные числа. Работоспособность программы необходимо было проверить с помощью Google Test-ов. Кроме того, необходимо разработать пользовательское консольное приложение.

**Особенности реализации:**

1. В качестве структуры хранения полинома использовать список мономов с ненулевыми коэффициентами (односвязный или двухсвязный). Односвязный список удобнее реализовать с фиктивной головой. Элементы списка хранить упорядоченными.
2. 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”.
3. Сложение полиномов осуществлять алгоритмом слияния упорядоченных массивов.
4. Вычитание полиномов допускается выполнять через сложение с умножением на константу (C = A - B = A + (-1)\*B)
5. При умножении и сложение (вычитание) необходимо следить, чтобы в итоговом полиноме были приведены подобные слагаемые и не хранилось мономов с нулевым коэффициентом.
6. Если при умножении полиномов полученные степени переменных больше 9, выводить сообщение об ошибке.
7. Считывание полинома у пользователя допускается в любом удобном для вас виде. Необходимо предоставить пользователю правила ввода данных
8. Следует учесть, что пользователь может вводить полином, не упорядочив в нем мономы.

# **2**. Руководство пользователя

При запуске консольного приложения по работе с полиномами, пользователю предлагается на выбор 8 пунктов меню:  
1. Создания полинома.

2. Напечатать полином по номеру.  
3. Сложить полиномы.

4. Вычесть полиномы.

5. Умножить полиномы.

6. Умножить полном на константу.

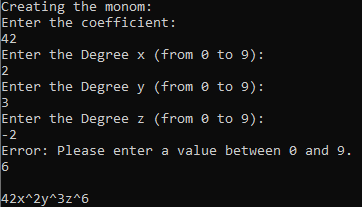
7. Очистить выбранный полином по номеру.

0. Выход из программы.

Примечание: консоль реализована с использованием вектора хранения полиномов, каждый новый добавленный полином или операция с полиномами хранится в векторе для удобства.

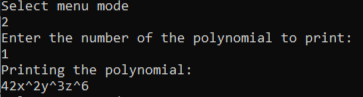
1. Create a new polynomial

Здесь пользователю предлагается ввести коэффициент, следом степень числа “X”, “Y” и “Z”. В случае неправильного ввода, пользователя попросят ввести сначала.



1. Print a polynomial

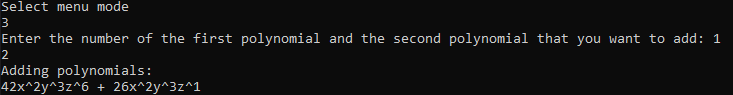
Среди уже добавленных полиномов – предлагается ввести номер полинома для печати.



1. Add two polynomials

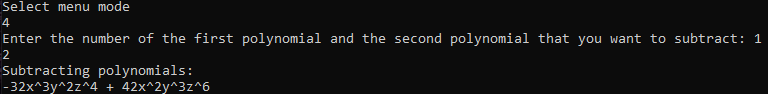
Ввод номеров полинома, которые необходимо сложить.

Имеется проверка на наличие, хотя бы двух добавленных полиномов.

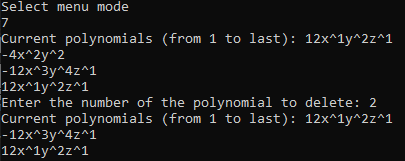


1. Subtract two polynomials

Аналогично сложению, необходимо выбрать номер полинома, из которого хотим вычесть, следом номер вычитаемого.

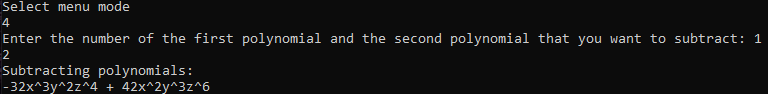


Точно также присутствует проверка на наличие хотя бы двух полиномов. Во всех последующих пунктах меню присутствует данная проверка.



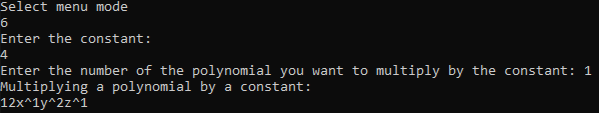
1. Multiply two polynomials

Аналогично двум другим функциям – ввести первый номер полинома и второй, которые хотим перемножить. В случае получения степени, выше 10 у “X”, “Y”, “Z” как и вместе, программа завершит свою работу досрочно и выведет ошибку.



1. Multiply a polynomial by a constant

Ввод полинома, затем константы целочисленного или вещественного значения.

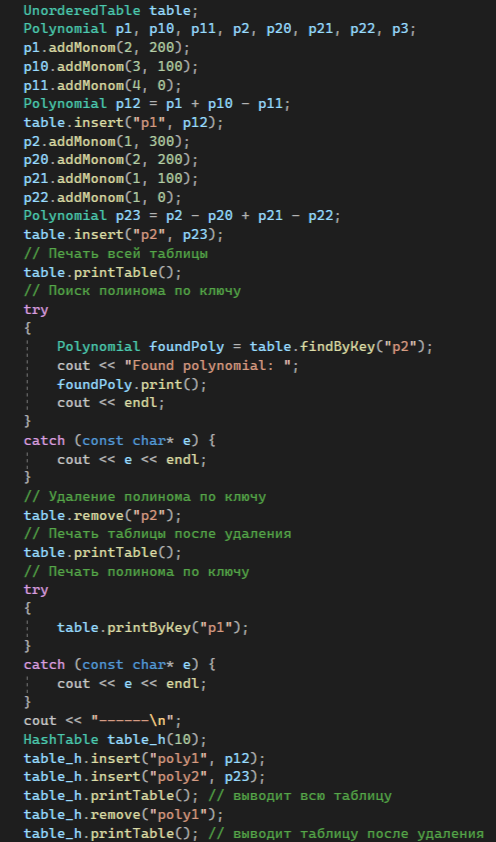


Выводится текущий список всех созданных полиномов, затем предлагается удалить номер полинома из списка. После чего выводится обновленный список.

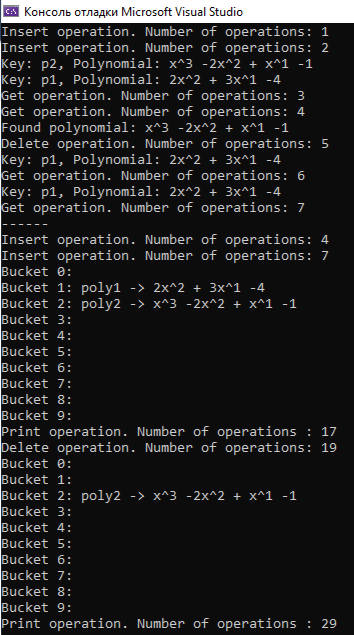
1. Exit

Выход из приложения.

Работа с таблицами трёх видов представлена в сыром виде:



Создается по каждому виду таблицы и добавляется полином в эту таблицу, следом производятся действия: 1) печать всей таблицы, поиску по ключу, удаление полинома по ключу, печать после удаления, печать полинома по ключу – с неупорядоченной таблицей., ключ задается пользователем; 2) аналогично с упорядоченной таблицей; 3) создание хеш -таблицы, вставка полиномов в хеш-таблицу, вывод всей таблицы, удаление полинома по ключу, вывод всей таблицы после удаления.



# Руководство программиста

## Описание структуры программы

С учетом сформулированных выше тезисов к реализации целесообразной представляется следующая структура программы:

**Проект “polys”:**

“list.h”:

Это реализация двусвязного списка. В нем используются два указателя, head и tail, для отслеживания начала и конца списка соответственно. Класс List содержит несколько методов для работы со списком, таких как:

IsEmpty() - проверка списка на пустоту

GetHead() и GetTail() - получение указателей на начало и конец списка

push\_back(T data) - добавление элемента в конец списка

print\_list() - печать содержимого списка

clear() - очистка списка

sort() - сортировка списка

pop\_list(Node<T>\* node) - удаление указанного узла из списка

Структура Node представляет узел списка и содержит указатели на предыдущий и следующий узлы, а также данные, хранящиеся в узле.

“monom.h”:

Это реализация класса многочлена Polynomial, который состоит из списка мономов List<Monom> monoms. Каждый моном представляет собой коэффициент и степень переменной в виде трёхзначного числа. В классе Polynomial определены следующие методы:

isEqual(const Polynomial& other) - проверка равенства двух многочленов

deleteZeroMonoms() - удаление нулевых мономов из списка

bringSimilarMonoms() - группировка подобных мономов

addMonom(double coeff, int degree) - добавление монома в многочлен

print() - печать многочлена

sort() - сортировка мономов в многочлене

operator<< - перегрузка оператора вывода в поток

operator+ - сложение многочленов

operator- - вычитание многочленов

negate() - изменение знака коэффициентов мономов

operator\* const - умножение многочлена на число

operator\*= const - умножение многочлена на число с изменением самого многочлена

operator\*= - умножение многочленов

operator\* - умножение многочленов с возвратом результата

Класс Monom определяет моном как структуру, содержащую коэффициент и степень переменной. В классе Monom определены следующие операторы:

operator< - перегрузка оператора сравнения на меньше

operator> - перегрузка оператора сравнения на больше

operator<< - перегрузка оператора вывода в поток

operator>> - перегрузка оператора ввода из потока

**Проект “poly\_main”** хранит реализацию консольного приложения для работы с полиномами и тестовую реализацию для обработки исключительных случаев, для тестирования без GoogleTests.

**Проект “gtest”** – соответственно реализации встроенный тестов для тестирования реализаций своих программ от компании Google.

**Проект “tables”:**

“table.h”: Структура для хранения пар ключ-значение.  
“ hash.h”:

Хэш-таблица реализована с помощью массива указателей на заголовки связанных списков (bucket). Размер массива устанавливается при создании объекта хэш-таблицы. Каждый элемент массива (bucket) содержит указатель на первый элемент связанного списка, который хранит пару «ключ-значение» в поле key и poly соответственно.

“ ordered.h” и “unordered.h”:

В данной реализации классов используется связанный список для хранения пар «ключ-значение». Каждый узел списка (Node) содержит поле data типа Entry, которое, в свою очередь, содержит поля key и poly для хранения ключа и значения соответственно. Кроме того, у каждого узла есть указатель next на следующий узел в списке.

Также присутствует проект “tests” по тестирования каждого написанного класса.

## Описание структур данных

**Таблица** - это набор записей, состоящих из полей, одно из которых является ключом, остальные образуют тело записи.

**Таблица** – динамическая структура данных, предоставляющая три основные операции:

– поиск записи по ключу

– вставка новой записи

– удаление записи

**Неупорядоченная таблица** - это таблица, порядок расположения записей в которой произволен по отношению к значениям ключей.

**Неупорядоченная таблица** - это таблица, поиск в которой возможен только полным перебором записей.

**Упорядоченная таблица** - это таблица, записи в которой

расположены в порядке «возрастания/убывания» ключей.

**Хеш-таблица** — таблицы с вычислимыми адресами.

**Функция**, отображающая значение ключа в индекс строки, называется

функцией хеширования (расстановки, перемешивания).

**Таблицы**, в которых данные размещены на основе хеш-функции, называются

хеш-таблицами (таблицами с вычислимыми адресами). Качество хеш-функций оценивают по вероятности коллизий.

**Основные требования к хеш-функциям**:

– быстрое вычисление

– равномерное распределение индексов

**Коллизии**

Ситуация, когда для двух разных значений ключей 𝑘1 и 𝑘2 значения хеш-функции равны, т.е. записи с ключами 𝑘1 и 𝑘2 должны быть размещены в одной и той же строке таблицы, называется **коллизией**.

**Метод цепочек**

В методе цепочек хеш-таблица представляет собой массив из 𝑀 списков. В каждом списке находятся записи с одинаковым значением хеш-функции.

Метод повторного перемешивания

Используем повторное перемешивание хеш-функции.

## Описание алгоритмов

3.3.1 Хэш-таблица.

**Конструктор и деструктор**

Конструктор инициализирует массив bucket нулевыми указателями, а также устанавливает начальное значение счетчика операций operationsCount и счетчика количества элементов count. Деструктор очищает память, выделенную под bucket и массив bucket.

**Хэширование**

Функция хэширования hash() вычисляет индекс для bucket в хэш-таблице на основе ключа. Для этого суммируется код каждого символа ключа, а затем полученная сумма берётся по модулю размера массива bucket.

**Вставка элемента**

Вставка элемента в хэш-таблицу осуществляется методом insert(). Вначале вычисляется индекс bucket с помощью функции хэширования. Затем происходит обход связанного списка bucket до тех пор, пока не будет найден конец списка или ключ, равный искомому. Если ключ уже существует, то значение обновляется. В противном случае создается новый bucket и вставляется в начало связанного списка bucket.

**Удаление элемента**

Удаление элемента из хэш-таблицы осуществляется методом remove(). Вначале вычисляется индекс bucket с помощью функции хэширования. Затем происходит обход связанного списка bucket до тех пор, пока не будет найден конец списка или ключ, равный искомому. Если ключ найден, то он удаляется из связанного списка bucket.

**Поиск элемента**

Поиск элемента в хэш-таблице осуществляется методом printByKey(). Вначале вычисляется индекс bucket с помощью функции хэширования. Затем происходит обход связанного списка bucket до тех пор, пока не будет найден конец списка или ключ, равный искомому. Если ключ найден, то возвращается значение, соответствующее этому ключу.

**Проверка наличия ключа**

Проверка наличия ключа в хэш-таблице осуществляется методом containsKey(). Вначале вычисляется индекс bucket с помощью функции хэширования. Затем происходит обход связанного списка bucket до тех пор, пока не будет найден конец списка или ключ, равный искомому. Если ключ найден, то метод возвращает true, в противном случае false.

**Печать хэш-таблицы**

Печать хэш-таблицы осуществляется методом printTable(). Для каждого bucket в массиве выводится связанный список bucket с ключами и значениями.

3.3.2 Упорядоченная таблица.

**Конструктор и деструктор**

Конструктор инициализирует указатель table нулевым указателем и устанавливает начальное значение счетчика операций operationsCount. Деструктор удаляет все узлы связанного списка table, освобождая память.

**Вставка элемента**

Вставка элемента в таблицу осуществляется методом insert(). Вначале создается новый узел newNode и инициализируются его поля data. key и data. poly. Затем осуществляется вставка нового узла в нужную позицию связанного списка table в соответствии с порядком сортировки по ключам. Если список пуст или ключ меньше ключа первого узла, то новый узел вставляется в начало списка. В противном случае происходит обход списка до тех пор, пока не будет найдена позиция для вставки нового узла.

**Поиск элемента**

Поиск элемента в таблице осуществляется методом printByKey(). Вначале происходит обход связанного списка table до тех пор, пока не будет найден узел с заданным ключом или не закончится список. Если ключ найден, то выводится значение, соответствующее этому ключу.

**Печать таблицы**

Печать таблицы осуществляется методом printTable(). Для этого происходит обход связанного списка table и для каждого узла выводится ключ и значение.

**Удаление элемента**

Удаление элемента из таблицы осуществляется методом remove(). Вначале происходит обход связанного списка table до тех пор, пока не будет найден узел с заданным ключом или не закончится список. Если ключ найден, то узел удаляется из списка и освобождается память.

**Поиск минимального элемента**

Поиск минимального элемента осуществляется методом reorderTable. Вначале ищется узел с минимальным ключом в текущем отрезке связанного списка table. Затем минимальный узел удаляется из списка и вставляется в начало отрезка. Далее рекурсивно вызывается метод reorderTable для следующего отрезка списка.

**Поиск значения по ключу**

Поиск значения по ключу осуществляется методом findByKey. Вначале происходит обход связанного списка table до тех пор, пока не будет найден узел с заданным ключом или не закончится список. Если ключ найден, то возвращается значение, соответствующее этому ключу.

3.3.2 Неупорядоченная таблица.

**Конструктор и деструктор**

Конструктор инициализирует указатель table нулевым указателем и устанавливает начальное значение счетчика операций operationsCount. Деструктор удаляет все узлы связанного списка table, освобождая память.

**Вставка элемента**

Вставка элемента в таблицу осуществляется методом insert(). Вначале создается новый узел newNode и инициализируются его поля data.key и data.poly. Затем новый узел вставляется в начало связанного списка table.

**Поиск элемента**

Поиск элемента в таблице осуществляется методом printByKey(). Вначале происходит обход связанного списка table до тех пор, пока не будет найден узел с заданным ключом или не закончится список. Если ключ найден, то выводится значение, соответствующее этому ключу.

**Печать таблицы**

Печать таблицы осуществляется методом printTable(). Для этого происходит обход связанного списка table и для каждого узла выводится ключ и значение.

**Удаление элемента**

Удаление элемента из таблицы осуществляется методом remove(). Вначале происходит обход связанного списка table до тех пор, пока не будет найден узел с заданным ключом или не закончится список. Если ключ найден, то узел удаляется из списка и освобождается память.

**Поиск значения по ключу**

Поиск значения по ключу осуществляется методом findByKey(). Вначале происходит обход связанного списка table до тех пор, пока не будет найден узел с заданным ключом или не закончится список. Если ключ найден, то возвращается значение, соответствующее этому ключу.

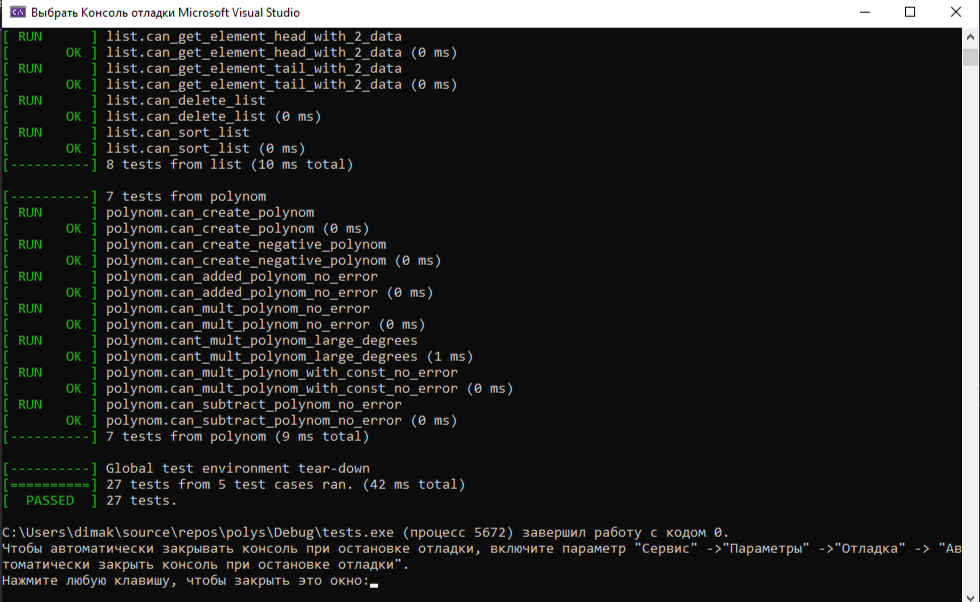
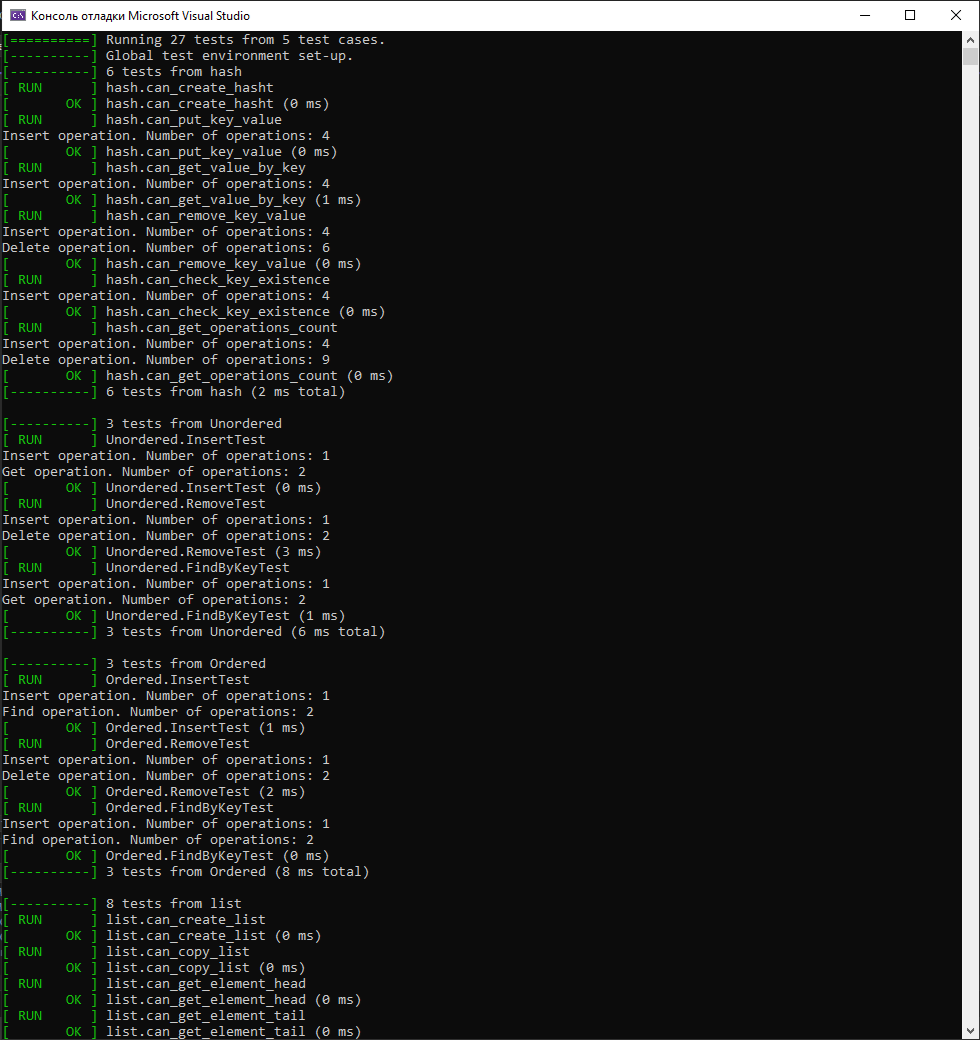
**Вывод количества операций**

Вывод количества операций осуществляется методом logOperations. Он выводит текущее значение счетчика операций operationsCount.

# Проверка корректности

При помощи встроенной библиотеки Google тестов, произведена проверка базовых функций классов “list”, “polynom”, “HashTable”, “UnorderedTable” , “OrderedTable”.

Результаты прохождения тестов:



# Заключение

В ходе данной лабораторной работы было расширено приложение, разработанное для работы с полиномами (которое поддерживает действия с десятичными и целыми коэффициентами мономов, операциями сложения, вычитания, умножения полинома на константу и перемножения двух полиномов). Благодаря наличию трёх видов таблиц: неупорядоченной, упорядоченной и хеш-таблицы, появилась возможность добавлять полиномы в таблицы, удалять их по уникальному ключу и осуществлять поиск по ключу, кроме того, для всех видов операций по работе с таблицами была организована оценка количества операций.

# Список литературы

1. Кормен Т.Х., Лейзерсон Ч.И., Ривест Р.Л., Штайн К. Алгоритмы: построение и анализ. – 3-е изд. – М.: Вильямс, 2013.
2. https://learn.microsoft.com/ru-ru/cpp/cpp/?view=msvc-170
3. <https://metanit.com/cpp/>