|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №2  по дисциплине «Язык программирования Python»  Вариант 8 | | |
|  | | Работу выполнил  студент группы ИТ/О ИТ-6-20241 курса  Рясин В.А Фамилия И.О.  «15» июня 2025 г. |
| Работу проверил  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фамилия И.О.  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2025 г. |
| Пермь 2025 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc1)

[Алгоритм решения 4](#_Toc2)

[1. Структура данных 4](#_Toc3)

[2. Основные операции 4](#_Toc4)

[3. Парсинг строки 5](#_Toc5)

[4. Ввод данных 5](#_Toc6)

[5. Основная логика 5](#_Toc7)

[6. Последовательность выполнения 7](#_Toc8)

[7. Обработка ошибок 7](#_Toc9)

[Тестирование 8](#_Toc10)

[Код программы 15](#_Toc11)

# Постановка задачи

Многочлен P(x) = a\_n \* x^n + a\_(n-1) \* x^(n-1) + ... + a\_1 \* x + a\_0 с целыми коэффициентами можно представить в виде списка, в котором первое информационное поле соответствует коэффициенту при неизвестном, а второе показателю степени этого же неизвестного. Написать процедуры для ввода и вывода многочлена из строкового формата в список. В качестве строкового формата может использоваться запись типа: 52y^10 – 3y^8 + y, где знак ^ означает возведение в степень. Реализовать процедуру для приведения подобных слагаемых в многочлене. С помощью описанных процедур ввести многочлен из файла, привести подобные члены и вывести результирующий многочлен в конец исходного файла, расположив его члены в порядке убывания степеней. Использовать однонаправленный линейный список.

# Алгоритм решения

## 1. Структура данных

* Класс Node (узел многочлена):
  + Содержит коэффициент (coef).
  + Содержит степень (degree).
  + Содержит ссылку на следующий узел (next).
* Класс PolynomialList (список многочлена):
  + Содержит указатель на голову списка (head).

## 2. Основные операции

* Добавление члена:
  + Метод push(coef, degree).
  + Проверка на ненулевой коэффициент.
  + Добавление узла в конец списка.
* Приведение подобных:
  + Метод combine\_like\_terms().
  + Сравнение степеней.
  + Сложение коэффициентов.
  + Удаление нулевых членов.
* Сортировка:
  + Метод bubble\_sort().
  + Сортировка по убыванию степени.
  + Алгоритм пузырька.
* Вывод в строку:
  + Метод list\_to\_string().
  + Формирование строкового представления.
  + Учет знаков, коэффициентов и степеней.

## 3. Парсинг строки

* Функция parsing\_polynomial(input\_str):
  + Удаление пробелов.
  + Определение переменной.
  + Разбиение на члены.
  + Парсинг коэффициентов и степеней.
  + Добавление в список.

## 4. Ввод данных

* Функция input\_data(filename):
  + Выбор способа ввода:
    - Загрузка из файла.
    - Ввод с клавиатуры.
  + Возврат выражения.
* Функция read\_file(filename):
  + Чтение первой строки.
  + Проверка на пустоту.
* Функция write\_to\_file():
  + Проверяет существование файла.
  + Создает файл, если он не существует.
  + Записывает результат в файл, учитывая его текущее содержимое (добавляет перенос строки, если файл не пустой).

## 5. Основная логика

* Функция main():
  + Задание имени файла.
  + Получение входных данных.
  + Парсинг в список.
  + Приведение подобных.
  + Сортировка.
  + Формирование результата.
  + Вывод.
  + Запись в файл.

## 6. Последовательность выполнения

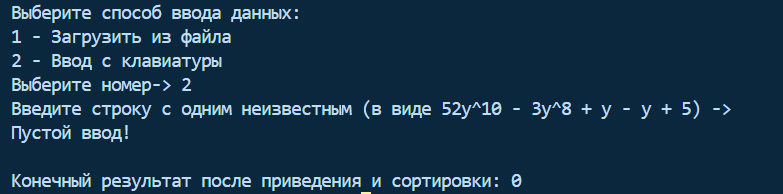
* Запуск через if \_\_name\_\_ == "\_\_main\_\_"::
  + Вызов main().
  + Обработка ввода.
  + Парсинг выражения.
  + Математические операции.
  + Вывод результата.
  + Сохранение в файл.

## 7. Обработка ошибок

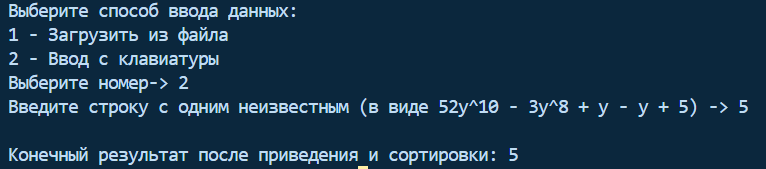
* Проверка на пустые вводы.
* Проверка корректности формата.
* Обработка некорректных символов.
* Проверка количества переменных.
* Обработка ошибок парсинга чисел.

# Тестирование

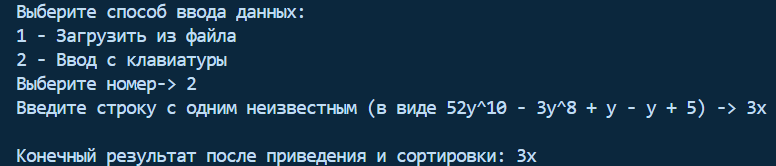
1. Тест на пустой ввод
   1. Входная строка: ""
   2. Ожидаемый результат: Пустой многочлен (вывод "0").
   3. Цель: Проверить обработку пустого ввода.



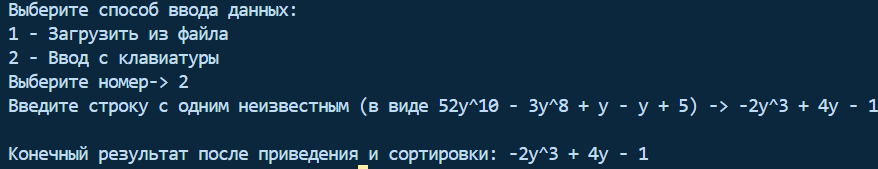
1. Тест на многочлен без переменной (только константа)
   1. Входная строка: "5"
   2. Ожидаемый результат: "5"
   3. Цель: Проверить обработку многочлена, состоящего только из константы.



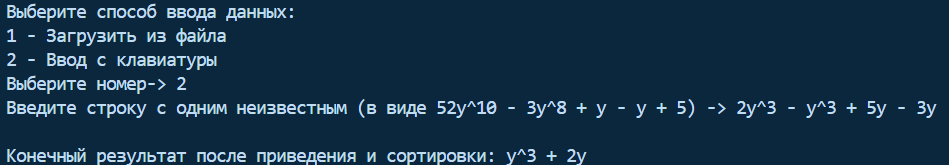
1. Тест на многочлен с одной переменной без степени
   1. Входная строка: "3x"
   2. Ожидаемый результат: "3x"
   3. Цель: Проверить обработку члена с переменной без явной степени.



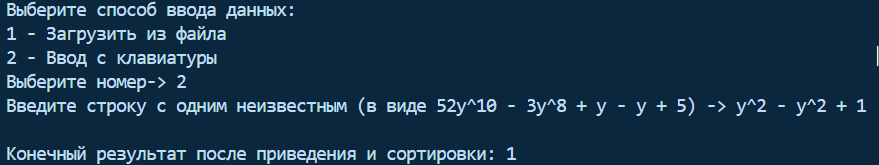
1. Тест на многочлен с отрицательными коэффициентами
   1. Входная строка: "-2y^3 + 4y - 1"
   2. Ожидаемый результат: "-2y^3 + 4y - 1"
   3. Цель: Проверить корректность обработки отрицательных коэффициентов.



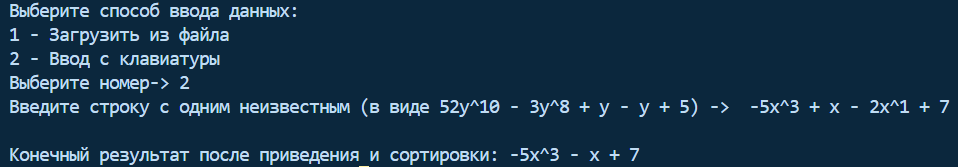
1. Тест на многочлен с приведением подобных членов
   1. Входная строка: "2y^3 - y^3 + 5y - 3y"
   2. Ожидаемый результат: "y^3 + 2y"
   3. Цель: Проверить работу функции combine\_like\_terms().



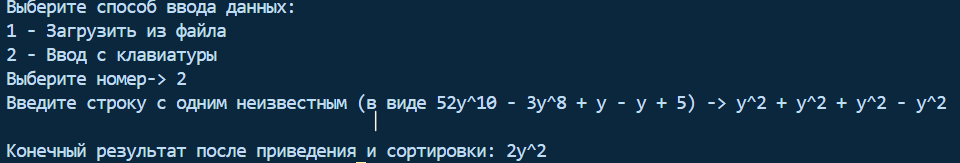
1. Тест на многочлен с нулевыми коэффициентами после приведения
   1. Входная строка: "y^2 - y^2 + 1"
   2. Ожидаемый результат: "1"
   3. Цель: Проверить удаление нулевых коэффициентов.



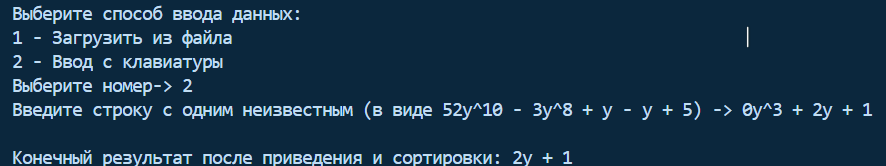
1. Тест на многочлен с пробелами и разными форматами записи
   1. Входная строка: " -5x^3 + x - 2x^1 + 7 "
   2. Ожидаемый результат: "-5x^3 - x + 7"
   3. Цель: Проверить корректность парсинга при наличии пробелов и разных форматов (x и x^1).



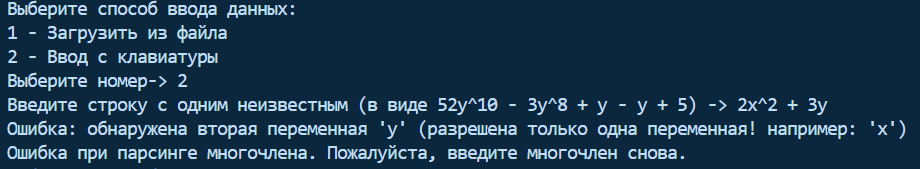
1. Тест на многочлен с дублирующимися подобными членами
   1. Входная строка: "y^2 + y^2 + y^2 - y^2"
   2. Ожидаемый результат: "2y^2"
   3. Цель: Проверить обработку нескольких одинаковых членов.



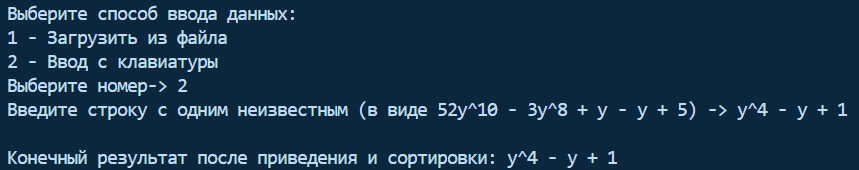
1. Тест на многочлен с нулевым коэффициентом в начале
   1. Входная строка: "0y^3 + 2y + 1"
   2. Ожидаемый результат: "2y + 1"
   3. Цель: Проверить удаление члена с нулевым коэффициентом.



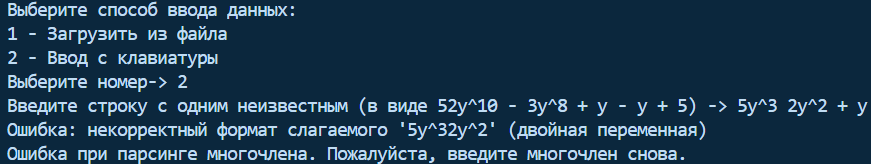
1. Тест на многочлен с некорректным форматом (двойная переменная)
   1. Входная строка: "2x^2 + 3y"
   2. Ожидаемый результат: Сообщение об ошибке ("Ошибка: обнаружена вторая переменная").
   3. Цель: Проверить обработку некорректного ввода.



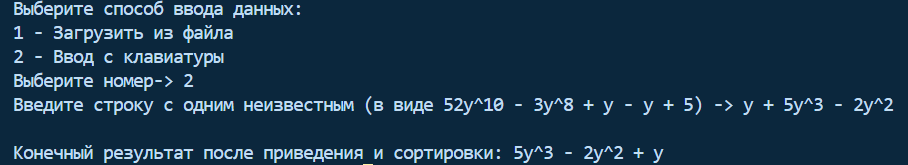
1. Тест на многочлен с пропущенными коэффициентами (неявная 1)
   1. Входная строка: "y^4 - y + 1"
   2. Ожидаемый результат: "y^4 - y + 1"
   3. Цель: Проверить обработку членов с неявным коэффициентом 1.



1. Тест на многочлен с пропущенным знаком перед первым членом
   1. Входная строка: "5y^3 2y^2 + y"
   2. Ожидаемый результат: Сообщение об ошибке ("Ошибка: некорректный формат слагаемого").
   3. Цель: Проверить обработку отсутствующего знака между членами.



1. Тест на сортировку членов по убыванию степени
   1. Входная строка: "y + 5y^3 - 2y^2"
   2. Ожидаемый результат: "5y^3 - 2y^2 + y"
   3. Цель: Проверить работу функции bubble\_sort().

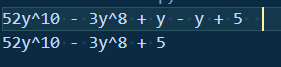
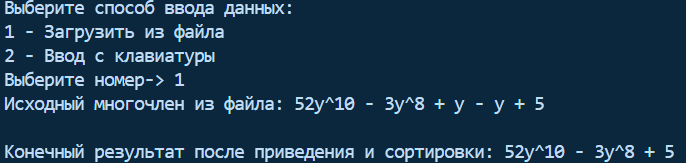


1. Тест на запись результата в файл
   1. Действие: Запустить программу с корректным многочленом и проверить, что результат записывается в конец файла.
   2. Ожидаемый результат: В файле появляется строка с упрощенным и отсортированным многочленом.
   3. Цель: Проверить корректность работы с файлами.

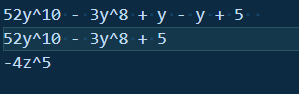
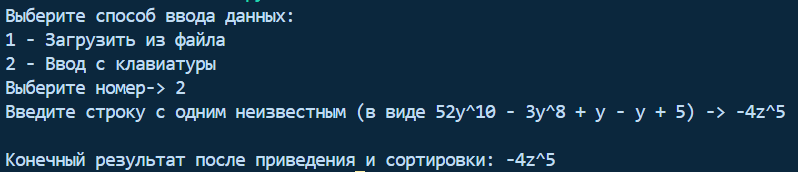
Многочлен записанный в файл:



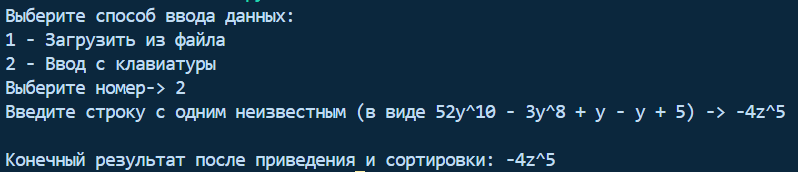
Выполнение программы и запись результата в файл:



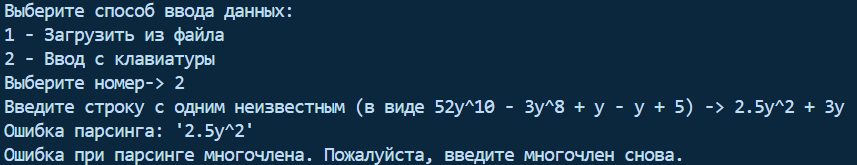
Ввод с клавиатуры и запись результата в файл:



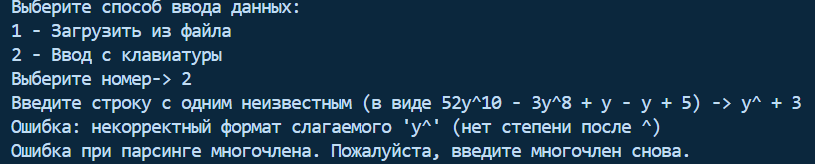
1. Тест на многочлен с единственным отрицательным членом
   1. Входная строка: "-4z^5"
   2. Ожидаемый результат: "-4z^5"
   3. Цель: Проверить корректность обработки многочлена с одним членом.



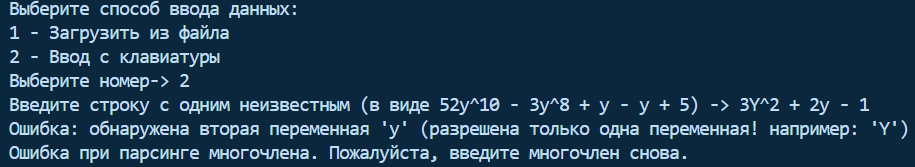
1. Тест на многочлен с дробными коэффициентами (ошибка)
   1. Входная строка: "2.5y^2 + 3y"
   2. Ожидаемый результат: Сообщение об ошибке ("Ошибка парсинга: '2.5y^2'").
   3. Цель: Проверить обработку нецелых коэффициентов.



1. Тест на многочлен с символом '^' без степени
   1. Входная строка: "y^ + 3"
   2. Ожидаемый результат: Сообщение об ошибке ("Ошибка: нет степени после ^").
   3. Цель: Проверить обработку некорректного использования символа степени.

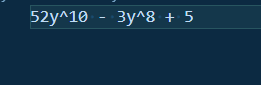
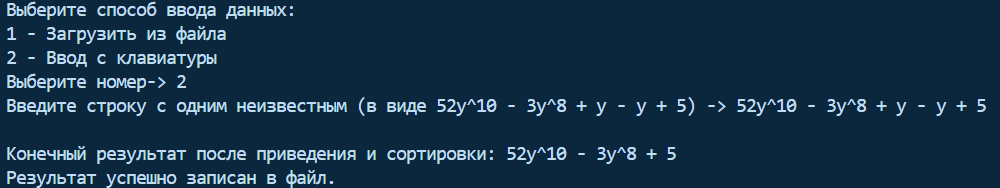
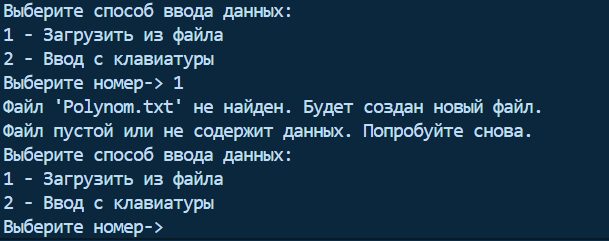


1. Тест на многочлен с переменной в разных регистрах
   1. Входная строка: "3Y^2 + 2y - 1"
   2. Ожидаемый результат: Сообщение об ошибке ("Ошибка: обнаружена вторая переменная 'y'").
   3. Цель: Проверить чувствительность к регистру переменных.



19. Проверка создания файла, если он не существует

1. Действие: Удалить файл (если существует) и запустить программу.
2. Ожидаемый результат: Программа создает файл и записывает в него результат.
3. Цель: Проверить корректность работы с несуществующим файлом.



# Код программы

"""

Многочлен P(x) = a\_n \* x^n + a\_(n-1) \* x^(n-1) + ... + a\_1 \* x + a\_0 с целыми коэффициентами можно представить в виде списка,

в котором первое информационное поле соответствует коэффициенту при неизвестном, а второе показателю степени этого же неизвестного.

Написать процедуры для ввода и вывода многочлена из строкового формата в список.

В качестве строкового формата может использоваться запись типа: 52y^10 – 3y^8 + y, где знак ^ означает возведение в степень. Реализовать процедуру для приведения подобных слагаемых в многочлене.

С помощью описанных процедур ввести многочлен из файла, привести подобные члены и вывести результирующий многочлен в конец исходного файла, расположив его члены в порядке убывания степеней.

Использовать однонаправленный линейный список.

"""

# Класс узла односвязного списка для хранения члена многочлена.

class Node:

def \_\_init\_\_(self, coef: int, degree: int) -> None:

self.coef = coef # Коэффициент

self.degree = degree # Степень

self.next = None # Ссылка на следующий элемент

# Класс для работы с многочленом в виде односвязного списка

class PolynomialList:

def \_\_init\_\_(self) -> None:

self.head = None # Голова списка

# Добавление нового члена в конец списка

def push(self, coef: int, degree: int) -> None:

if coef == 0:

return

new\_node = Node(coef, degree)

if self.head is None:

self.head = new\_node

else:

current = self.head

while current.next:

current = current.next

current.next = new\_node

# Приведение подобных членов (с одинаковой степенью)

def combine\_like\_terms(self) -> None:

current = self.head

while current:

prev = current

checker = current.next

while checker:

if checker.degree == current.degree:

current.coef += checker.coef

prev.next = checker.next

checker = checker.next

else:

prev = checker

checker = checker.next

current = current.next

# Удаление нулевых коэффициентов

while self.head and self.head.coef == 0:

self.head = self.head.next

current = self.head

while current and current.next:

if current.next.coef == 0:

current.next = current.next.next

else:

current = current.next

# Сортировка членов по убыванию степени (пузырьком)

def bubble\_sort(self) -> None:

if self.head is None:

return

swapped = True

while swapped:

swapped = False

prev = None

current = self.head

while current and current.next:

if current.degree < current.next.degree:

swapped = True

nxt = current.next

current.next = nxt.next

nxt.next = current

if prev:

prev.next = nxt

else:

self.head = nxt

prev = nxt

else:

prev = current

current = current.next

# Преобразование списка в строку для вывода

def list\_to\_string(self, variable: str = 'y') -> str:

current = self.head

if not current:

return "0"

result = ""

while current:

c, d = current.coef, current.degree

sign = " + " if c > 0 else " - "

if result == "":

sign = "-" if c < 0 else ""

result += sign

abs\_coef = abs(c)

if d == 0:

result += str(abs\_coef)

elif d == 1:

result += f"{'' if abs\_coef == 1 else abs\_coef}{variable}"

else:

result += f"{'' if abs\_coef == 1 else abs\_coef}{variable}^{d}"

current = current.next

return result.strip(" +")

# Функция преобразования строки в односвязный список многочлена

def parsing\_polynomial(input\_str: str) -> tuple['PolynomialList', str]:

input\_str = input\_str.replace(' ', '')

if not input\_str:

print("Пустой ввод!")

return None, None

# Определяем переменную (первая буква)

variable = 'y'

for ch in input\_str:

if ch.isalpha():

variable = ch

break

# Проверка на наличие других переменных

for ch in input\_str:

if ch.isalpha() and ch != variable:

print(f"Ошибка: обнаружена вторая переменная '{ch}' (разрешена только одна переменная! например: '{variable}')")

return None, None

input\_str = input\_str.lower().replace('-', '+-')

terms = input\_str.split('+')

poly = PolynomialList()

for term in terms:

if not term:

continue

# Проверка на некорректные случаи

if term.count(variable) > 1:

print(f"Ошибка: некорректный формат слагаемого '{term}' (двойная переменная)")

return None, None

if term.count('^') > 1:

print(f"Ошибка: некорректный формат слагаемого '{term}' (двойная степень)")

return None, None

if '^' in term and variable not in term:

print(f"Ошибка: некорректный формат слагаемого '{term}' (степень без переменной)")

return None, None

if '^' in term and term.index('^') < term.index(variable):

print(f"Ошибка: некорректный формат слагаемого '{term}' (степень до переменной)")

return None, None

if '^' in term and term.endswith('^'):

print(f"Ошибка: некорректный формат слагаемого '{term}' (нет степени после ^)")

return None, None

if variable in term and '^' in term:

idx\_var = term.index(variable)

idx\_pow = term.index('^')

if idx\_pow != idx\_var + 1:

print(f"Ошибка: некорректный формат слагаемого '{term}' (неправильное расположение ^)")

return None, None

if idx\_var == 0 and (len(term) == 1 or term[0] == variable):

coef = 1

else:

coef = term[:idx\_var]

degree = term[idx\_pow+1:]

elif variable in term:

idx\_var = term.index(variable)

if idx\_var == 0:

coef = 1

else:

coef = term[:idx\_var]

degree = 1

else:

coef = term

degree = 0

# Преобразование коэффициента и степени к числу

if coef == "" or coef == "+":

coef = 1

elif coef == "-":

coef = -1

try:

coef = int(coef)

degree = int(degree)

except ValueError:

print(f"Ошибка парсинга: '{term}'")

return None, None

poly.push(coef, degree)

return poly, variable

# Функция для ввода данных

def input\_data(filename: str) -> str:

while True:

n = input("Выберите способ ввода данных: \n1 - Загрузить из файла " \

"\n2 - Ввод с клавиатуры\nВыберите номер-> ")

if n == "1":

try:

data = read\_file(filename)

if not data:

print("Файл пустой или не содержит данных. Попробуйте снова.")

continue

print(f"Исходный многочлен из файла: {data}")

return data

except FileNotFoundError:

print(f"Файл '{filename}' не найден. Попробуйте снова.")

except Exception as e:

print(f"Ошибка при чтении файла: {e}. Попробуйте снова.")

elif n == "2":

expr = input(

"Введите строку с одним неизвестным "\

"(в виде 52y^10 - 3y^8 + y - y + 5) -> "

).strip()

if not expr:

print("Ввод не может быть пустым. Попробуйте снова.")

continue

return expr

else:

print("Введите цифру 1 или 2!")

# Функция для считывания данных из файла

def read\_file(filename: str) -> str:

try:

with open(filename, "r") as f:

expression = f.readline().strip()

if not expression:

print("Файл пустой!")

return ""

return expression

except FileNotFoundError:

print(f"Файл '{filename}' не найден. Будет создан новый файл.")

return ""

except Exception as e:

print(f"Ошибка при чтении файла: {e}")

return ""

# Функция для записи результата в файл

def write\_to\_file(filename: str, result: str) -> None:

try:

# Проверяем существование файла

try:

with open(filename, "r") as f:

content = f.read()

except FileNotFoundError:

content = ""

# Записываем результат

with open(filename, "a") as file:

if content: # Если файл не пустой, добавляем перенос строки

file.write("\n" + result)

else: # Если файл пустой, записываем без переноса

file.write(result)

print("Результат успешно записан в файл.")

except Exception as e:

print(f"Ошибка при записи в файл: {e}")

# Главная функция программы

def main() -> None:

filename = "Polynom.txt"

while True:

input\_string = input\_data(filename)

if not input\_string:

continue

poly, variable = parsing\_polynomial(input\_string)

if poly is None or variable is None:

print("Ошибка при парсинге многочлена. Пожалуйста, введите многочлен снова.")

continue

poly.combine\_like\_terms()

poly.bubble\_sort()

result = poly.list\_to\_string(variable)

print(f"\nКонечный результат после приведения и сортировки: {result}")

# Запись результата в файл

write\_to\_file(filename, result)

break

if \_\_name\_\_ == "\_\_main\_\_":

main()