|  |  |  |
| --- | --- | --- |
| 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 1 | | |
|  | | Работу выполнил  студент группы ИТ-2,Сентябрь-2024 1 курса  Стариков Марат С.  «26» Июня 2025 г. |
| Работу проверил  Анисимова Светлана Игоревна  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г. |
| Пермь 2024 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc153130027)

[Алгоритм решения 3](#_Toc153130028)

[Тестирование 3](#_Toc153130029)

[Код программы 3](#_Toc153130030)

[Инструкция по применению стилей и оформлению работы 4](#_Toc153130031)

# Постановка задачи

Написать программу, которая:

1. Принимает математическое выражение в инфиксной записи со скобками (максимальная вложенность — 10 уровней).
2. Строит дерево разбора выражения, где:
   * Операнды: целые числа и однобуквенные переменные.
   * Операторы: +, -, \*, / (включая унарный минус).
3. Упрощает выражение, применяя алгебраические преобразования:
   * Заменяет (f1\*f3 ± f2\*f3) на (f1 ± f2)\*f3.
   * Заменяет (f1\*f2 ± f1\*f3) на (f1\*(f2 ± f3)).
4. Выводит исходное и упрощенное выражения в виде деревьев.

# Алгоритм решения

1. **Лексический анализ (токенизация)**:
   * Разбивает строку выражения на токены: числа, переменные, операторы, скобки.
   * Игнорирует пробелы.
2. **Синтаксический анализ (построение дерева)**:
   * Рекурсивный нисходящий парсер с приоритетами операций:
     + parse\_add\_sub: Обрабатывает + и -.
     + parse\_mul\_div: Обрабатывает \* и /.
     + parse\_unary: Обрабатывает унарный минус.
     + parse\_primary: Обрабатывает числа, переменные и вложенные выражения в скобках.
3. **Упрощение выражения**:
   * Рекурсивно обходит дерево.
   * Ищет паттерны для алгебраических преобразований:
     + Выносит общие множители за скобки.
   * Возвращает новое упрощенное дерево.

1. **Вычисление выражения**:
   * Рекурсивно вычисляет значение дерева с подстановкой значений переменных (если заданы).

# Тестирование

Проверка на примерах:

1. Вход: "a\*b+a\*c"  
   Упрощение: a\*(b+c)  
   **Тест пройден**.
2. Вход: "x\*y-z\*y"  
   Упрощение: (x-z)\*y  
   **Тест пройден**.
3. Вход: "a\*(b+c)-a\*d"  
   Упрощение: a\*(b+c-d)  
   **Тест пройден**.
4. Вход: "2\*(x+3)+4\*(x+3)"  
   Упрощение: (2+4)\*(x+3)  
   **Тест пройден**.
5. Вход: "a\*b+c\*d"(нет общих множителей)  
   Упрощение: Без изменений.  
   **Тест пройден**.

# Код программы

<https://github.com/SamuelRodriguesMGR/Expression-Factorizer>

# Инструкция по применению стилей и оформлению работы

Для оформления частей отчёта следует использовать заранее созданные стили. Все стили, которые могут пригодиться, начинаются с «ЛР. 1 …».



**ЛР. 1 Обычный** – для оформления текста задания и алгоритма решения.

**ЛР. 1 Текст программы** – для оформления кода программы.

**ЛР. 1 Заголовок 1** – заголовок первого уровня (для того, чтобы озаглавить основные разделы отчета).

**ЛР. 1 Заголовок 2** – заголовок второго уровня (для того, чтобы озаглавить подразделы).

Для того, чтобы перенести текст следующего блока на другую страницу, необходимо воспользоваться инструментов «Разрыв страницы» в разделе «Вставка».

