НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра Обчислювальної Техніки

Лабораторна робота №2

з Програмного забезпечення комп’ютерних систем

Виконав:
Студент 5 курсу, групи ІМ-42мп
БАРДІН Владислав Дмитрович

_____

_____

Перевірив:
Асистент кафедри обчислювальної техніки,
КОБИЛЮК Андрій Григорович

Київ — 2024 року

Мета роботи

Виконати автоматичне розпаралелювання арифметичного виразу.

Вхідні дані

Коректний арифметичний вираз після успішного виконання лексичного та синтаксичного аналізу (результат виконання лабораторної роботи №1).

Завдання роботи

За аналітичним записом арифметичного виразу (AB) побудувати його дерево паралельної форми максимальної ширини (максимальна кількість операцій в одному ярусі) та мінімальної довжини (мінімально можлива кількість ярусів). Для виконання цієї лабораторної роботи необхідно розробити алгоритм побудови дерева паралельної форми за записом AB та реалізувати цей алгоритм на будь-якій мові програмування. Спосіб зображення отриманого дерева визначається студентом. Окремої уваги заслуговують фрагменти AB типу $A/S/D/F/G/H$ або $A-S-D-F-G-H$. У цих випадках для того, щоб розпаралелювання стало можливим потрібно частину операцій / замінити на * , а частину операцій – відповідно на +.

Опис алгоритму виконання роботи

Для побудови паралельного дерева використовується алгоритм топологічного сортування, який за своєю природою розділяє операції на рівні в залежності від кількості їх залежностей. Так, якщо це операція додавання, від якої нічого не залежить – ця операція буде знаходитися на 0 рівні, якщо є це операція ділення чи множення від якої залежать інші операції вона відповідно буде опускатися на рівень нижче. Також цей алгоритм дозволяє враховувати положення дужок без перетворення дерева у постфіксну форму. Окрім цього для оптимізації обрахунку та розпаралелювання виразу використовуються препроцесори, що виконують наступні операції:

- видалення операції, що не впливають на кінцевий результат, наприклад, ділення чи множення на 1, додавання/віднімання 0, чи ділення 0 на змінну;
- обчислення констант.

Для візуальної репрезентації дерева доступні декілька форматів:

- JSON дерево виводиться у JSON форматі, що зручно для його зберігання;
- текстовий формат, що використовується за замовчуванням;
- mermaid формат, що вимагає використання сторонніх сервісів для відображення.

Результати виконання

```
"/Users/vbardin/Documents/GitHub/KPI/KPI-Masters/Computer System Software/Source Code/Peat.Cli/bin/Debug/net9.0/Peat.Cli" tree "(a) + b * (c - d)" -v true
🌳 Tree Visualization:
└── +
    ├── a
    └── *
        ├── b
        └── -
            ├── c
            └── d


📊 Performance Metrics:
⚡ Parallelization Level: 3
🖥️  Estimated Processors: 1
⏱️ Processing Time: 10,97ms

🔄 Optimization Steps:
  ✨UnaryOperatorsOptimizer: 12 → 12 tokens
  ✨RedundantBracketsOptimizer: 12 → 10 tokens
  Transformation Report for RedundantBracketsOptimizer
Total Steps: 2

🔄 Step ID: 76dae58d-b3e4-45cf-aeee-1317ced47e31
⏱️ Time: 2024-12-25 11:49:43.884
📁 Type: RedundantBracketsOptimization
📝 Description: Removed redundant brackets
📍 Position: 1-3
⚪Original: ( a )
🟢 Result: a

🔄 Step ID: b9a9f3c0-989f-4314-87c2-9733c5f871de
⏱️ Time: 2024-12-25 11:49:43.885
📁 Type: RedundantBracketsOptimization
📝 Description: Keeping brackets due to operator precedence: inner=- prev=* next=
📍 Position: 7-11
⚪Original: ( c - d )
🟢 Result: ( c - d )


  ✨ConstantsOptimizer: 10 → 10 tokens
```

```
"/Users/vbardin/Documents/GitHub/KPI/KPI-Masters/Computer System Software/Source Code/Peat.Cli/bin/Debug/net9.0/Peat.Cli" tree "((a) + b) * (c - d)" -v true
🌳 Tree Visualization:
└── *
    ├── +
    │   ├── a
    │   └── b
    └── -
        ├── c
        └── d


📊 Performance Metrics:
⚡ Parallelization Level: 2
🖥️  Estimated Processors: 1
⏱️ Processing Time: 9,40ms

🔄 Optimization Steps:
  ✨UnaryOperatorsOptimizer: 14 → 14 tokens
  ✨RedundantBracketsOptimizer: 14 → 12 tokens
  Transformation Report for RedundantBracketsOptimizer
Total Steps: 3

🔄 Step ID: 5e4f0047-5dc2-4829-9ec0-6482f30cb13d
⏱️ Time: 2024-12-25 11:50:50.236
📁 Type: RedundantBracketsOptimization
📝 Description: Keeping brackets due to operator precedence: inner=+ prev= next=*
📍 Position: 1-7
⚪Original: ( ( a ) + b )
🟢 Result: ( ( a ) + b )

🔄 Step ID: 68059b91-a1e0-4319-977b-c26dbcc1e99f
⏱️ Time: 2024-12-25 11:50:50.236
📁 Type: RedundantBracketsOptimization
📝 Description: Removed redundant brackets
📍 Position: 2-4
⚪Original: ( a )
🟢 Result: a

🔄 Step ID: bf81816b-a3bf-4c0d-af14-b6e7c0101cf4
⏱️ Time: 2024-12-25 11:50:50.236
📁 Type: RedundantBracketsOptimization
📝 Description: Keeping brackets due to operator precedence: inner=- prev=* next=
📍 Position: 9-13
⚪Original: ( c - d )
🟢 Result: ( c - d )


  ✨ConstantsOptimizer: 12 → 12 tokens
```

Лістинг програмного коду

Код лабораторної роботи розміщено на веб-ресурсі GitHub, і доступний для ознайомлення за посиланням: https://github.com/Bardin08/KPI-Masters/pull/5 а також https://github.com/Bardin08/KPI-Masters/pull/9.