Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа №2 по дисциплине «Функциональное и логическое программирование»

Тема Динамическая база знаний

Студент Александров Э.И.

Группа ИУ7-53БВ

Преподаватель Строганов Ю.В.

Москва, 2024

Содержание

ВВЕДЕНИЕ

Цель данной работы заключается в разработке программы на языке SWI-Prolog для вычисления факториала числа как с использованием мемоизации, так и без нее. Задачи, поставленные в ходе работы, включают реализацию рекурсивного алгоритма для вычисления факториала, а также оптимизацию этого алгоритма с помощью мемоизации для повышения эффективности.

1 Аналитическая часть

Факториал числа N (обозначается как $N!$) — это произведение всех положительных целых чисел от 1 до N. Факториал широко используется в комбинаторике, теории вероятностей и других областях математики. Рекурсивный подход к вычислению факториала прост и интуитивно понятен, однако он может быть неэффективным для больших значений N из-за повторных вычислений. Мемоизация — это техника оптимизации, которая позволяет сохранять результаты вычислений и повторно использовать их, что значительно ускоряет выполнение программы, особенно для больших значений N.

Мемоизация в Prolog реализуется с помощью динамических предикатов, которые позволяют сохранять результаты вычислений для последующего использования. Это позволяет избежать повторных вычислений для одних и тех же входных данных, что значительно повышает эффективность программы.

Вывод

В аналитической части мы рассмотрели теоретические аспекты факториала и обосновали необходимость использования мемоизации для повышения производительности.

2 Конструкторская часть

Для реализации задачи были выбраны два подхода:

a. Вычисление факториала без мемоизации — Алгоритм для вычисления факториала без мемоизации реализует рекурсивный подход, где факториал N вычисляется как N умноженное на факториал (N-1).

b. Вычисление факториала с мемоизацией — В случае мемоизации используется динамический предикат для хранения уже вычисленных значений факториала, что позволяет избежать повторных вычислений.

Архитектура программы включает два основных предиката: один для вычисления факториала без мемоизации и другой — с мемоизацией.

Алгоритм вычисления без мемоизации:

— если $N = 0$, возвращаем результат 1;

— если $N > 0$, вычисляем факториал для $N - 1$ и умножаем результат на N.

Алгоритм с мемоизацией:

— если значение факториала для N уже сохранено в базе данных, программа возвращает сохранённое значение;

— если значение не сохранено, программа вычисляет его и сохраняет для последующего использования

Вывод

В конструкторской части мы разработали алгоритмы для вычисления факториала с мемоизацией и без и определили их архитектуру.

3 Технологическая часть

Реализация программы была выполнена на языке программирования SWI-Prolog. Для разработки использовалась среда разработки SWI-Prolog, которая предоставляет удобные инструменты для написания и отладки Prolog-кода.

Программа без мемоизации:

```
factorial(0, 1).
% Recursion: factorial of N is N * factorial of (N-1)
factorial(N, Result) :-
N > 0,
N1 is N - 1,
factorial(N1, Result1),
Result is N * Result1.
```

Программа с мемоизацией:

```
:- dynamic factorial_memo/2.
% База: factorial of 0 is 1
factorial_memo(0, 1).
factorial_memo(N, Result) :-
N > 0,
N1 is N - 1,
factorial_memo(N1, Result1),
Result is N * Result1,
assertz(factorial_memo(N, Result)).
factorial(N, Result) :-
factorial_memo(N, Result).
```

В программе используются следующие команды:

— dynamic/1 — это директива в Prolog, которая используется для объявления предикатов как динамических. Динамические предикаты могут быть добавлены, удалены или изменены во время выполнения программы.

— assertz/1 — это встроенный предикат в Prolog, который используется для добавления нового факта или правила в базу данных программы. Он добавляет указанный факт или правило в конец базы данных, что позволяет сохранять информацию для последующего использования.

Тестовые данные:

Пример тестов:

```
?- factorial(5, Result).
```

```
Result = 120.

?- factorial(8, Result).
Result = 40320.
```

Все тесты пройдены успешно.

Результаты тестов показывают корректные данные как с мемоизацией, так и без неё.

Вывод

В технологической части мы реализовали программу для вычисления факториала с мемоизацией и без на языке Prolog и успешно протестировали ее.

ЗАКЛЮЧЕНИЕ

В ходе работы была поставлена цель разработать программу для вычисления факториала на SWI-Prolog с использованием мемоизации и без нее. Мы реализовали два алгоритма, проанализировали их эффективность и протестировали программу. Сделали вывод, что мемоизация значительно ускоряет расчет факториала для больших значений N. Все задачи, поставленные в начале работы, были успешно выполнены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Сергиевский Г. М., Волченков Н. Г. Функциональное и логическое программирование: учебник. – Москва: Издательский центр Академия, 2010. – 320 с.

2. Рекурсия и мемоизация. — [https://www.studyplan.dev/pro-cpp/recursion].

Приложение А

Код программы:

Факториал без меморизации

```
factorial(0, 1).
% Recursion: factorial of N is N * factorial of (N-1)
factorial(N, Result) :-
N > 0,
N1 is N - 1,
factorial(N1, Result1),
Result is N * Result1.

% ?- factorial(5, Result).
```

Факториал с меморизацией

```
:- dynamic factorial_memo/2.
% База: factorial of 0 is 1
factorial_memo(0, 1).
% Проверяем, вычислен ли уже факториал.
factorial_memo(N, Result) :-
N > 0,
N1 is N - 1,
factorial_memo(N1, Result1),
Result is N * Result1,
assertz(factorial_memo(N, Result)).
factorial(N, Result) :-
factorial_memo(N, Result).

% ?- factorial(5, Result).
```