Г. М. Якимович, А. И. Маршалович

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЭФФЕКТИВНОСТИ РЕКУРСИВНЫХ И ИТЕРАТИВНЫХ АЛГОРИТМОВ

В данной работе проводится сравнительный анализ рекурсии и итерации как двух фундаментальных парадигм в программировании.

Введение

Решение повторяющихся задач в программировании базируется на двух парадигмах: рекурсии (самовызове функции) и итерации (циклическом выполнении). Выбор между ними определяет производительность, читаемость кода и эффективность использования ресурсов.

I. Рекурсивные алгоритмы

Рекурсивные алгоритмы решают задачу путем самовызова функции, разделяя её на более мелкие однотипные части. Процесс продолжается до тех пор, пока не будет достигнут простейший случай — базовое условие, после чего результат собирается в обратном порядке.

II. Итеративные алгоритмы

Итеративные алгоритмы — это способ решения задачи путем многократного повторения одних и тех же действий в цикле. На каждом шаге состояние переменных обновляется, постепенно приближая алгоритм к результату, пока не будет достигнуто условие выхода.

III. Различия и сравнения алгоритмов

Рекурсивные алгоритмы обеспечивают лаконичность и прозрачность кода, но создают высокую нагрузку на стек, что чревато ошибкой stack overflow.

Итеративный подход отличается ресурсной эффективностью и стабильностью, исключая системные риски переполнения памяти. Выбор между методами требует баланса между архитектурной ясностью и вычислительными ограничениями системы.

IV. Экспериментальное исследование на базе расчета факториала

Для демонстрации структурных и ресурсных различий парадигм на языке C++ проанализировано время вычисления факториалов чисел от 1 до 20, на основе 10 000 повторных расчетов для каждого числа. Рекурсивная версия лаконична (2 строки кода), но инициирует линейный рост стека вызовов, что при больших значениях создает риск stack overflow. Итеративный подход требует на 1–2 строки кода больше, но гарантирует константное потребление памяти и более высокую производительность за счет отсутствия накладных расходов на создание стековых фреймов. (см. рис. 1).

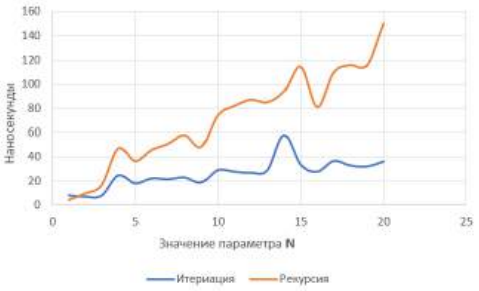

Рис. 1 – Диаграмма времени выполнения задачи для двух парадигм

Диаграмма показывает зависимость времени от числа итераций, демонстрируя преимущество итеративного метода. В отличие от рекурсии, он исключает риск stack overflow и накладные расходы на стековые фреймы. Это делает итерацию более быстрым и стабильным решением для масштабируемых систем.

V. Выводы

Выбор между рекурсией и итерацией базируется на балансе читаемости и ресурсной эффективности. Рекурсия упрощает код при работе со сложными нелинейными структурами (например, деревьями), в то время как итерация гарантирует стабильность и производительность в ресурсоемких линейных вычислениях. Оптимальное решение требует учета глубины вложенности и лимитов стека для предотвращения критических ошибок.

Список литературы

1. Recursion Versus Iteration with the List as a Data Structure / Informatics in Education. – I. Foltynowicz, 2007.
2. Slash [Электронный ресурс]. – Режим доступа : https://slash.co/articles/5-brief-explanation-about-the-difference-between-recursion-vs-iteration/.

Маршалович Артем Игоревич, студент ФИТиУ, БГУИР, jker82217@gmail.com,
Якимович Георгий Максимович, студент ФИТиУ БГУИР, georgijakimovic347@gmail.com.
Научный руководитель: Кукин Дмитрий Петрович, заведующий кафедрой вычислительных методов и программирования БГУИР, кандидат технических наук, доцент, kukin@bsuir.by.