Н. С. Минчуков, И. Д. Усольцев, В. А. Герасимов

РЕКУРСИЯ В ПРОГРАММИРОВАНИИ И РАЗРАБОТКЕ ПРИЛОЖЕНИЙ

В этой работе рассказывается о рекурсии, её плюсах и недостатках. Рассматривается вид «хвостовой рекурсии»

Введение

Рекурсия – это общее понятие, которое может быть присуще чему угодно и встречаться в повседневной жизни, но больше всего она распространена в информатике и математике. Для программистов же умение применять рекурсию — большой плюс в коллекцию полезных навыков. Самое известное программисту применение рекурсии – задачи на вычисление чисел Фибоначчи или факториала[1].

I. Способ реализации

Реализация рекурсивных вызовов функций в практически применяемых языках и средах программирования, как правило, опирается на механизм стека вызовов — адрес возврата и локальные переменные функции записываются в стек, благодаря чему каждый следующий рекурсивный вызов этой функции пользуется своим набором локальных переменных и за счёт этого работает корректно. Обратной стороной этого довольно простого по структуре механизма является то, что на каждый рекурсивный вызов требуется некоторое количество оперативной памяти компьютера, и при чрезмерно большой глубине рекурсии может наступить переполнение стека вызовов. Вопрос о желательности использования рекурсивных функций в программировании неоднозначен: с одной стороны, рекурсивная форма может быть структурно проще и нагляднее, в особенности, когда сам реализуемый алгоритм по сути рекурсивен. Кроме того, в некоторых декларативных или чисто функциональных языках, таких как Prolog или Haskell, просто нет синтаксических средств для организации циклов, из-за чего мы вынуждены использовать рекурсию, т.к. рекурсия в них — единственный доступный механизм организации повторяющихся вычислений. С другой стороны, обычно рекомендуется избегать рекурсивных программ, которые могут приводить к слишком большой глубине рекурсии. Так, широко распространённый в учебной литературе пример рекурсивного вычисления факториала является, скорее, примером того, как не надо применять рекурсию, так как приводит к достаточно большой глубине рекурсии и имеет очевидную реализацию в виде обычного циклического алгоритма.

II. Сфера применения

Выбор между рекурсивным и итеративным методом может в значительной степени зависеть от языка, который используется, или от задачи, которую программист намерен решить. Например, в JavaScript рекурсия может привести к ошибкам stack frame errors, когда предел стека уже достигнут, а базовое условие ещё не выполнено. В таком случае, итеративный подход будет работать лучше.

В качестве эксперимента была написана программа на языке C++, в которой с помощью библиотечной функции clock() было засечено время, за которое вычисляются выражения итерационным и рекурсивным методами. Программа производит каждое вычисление 10000 раз, так как время одного вычисления слишком мало и его не может засечь функция clock(). Полученные значения были поделены на 10000, для получения результата времени одного измерения (Таблица 1).

Таблица 1 – Результаты эксперимента

Вычисление	С рекурсией	Без рекурсии
$\sum_{n=1}^{65} n!$	0.0000013	0.0000002
$\sum_{n=1}^{3000} n$	0.0000596	0.0000063
$\sum_{n=1}^{65} \frac{1}{n!}$	0.0000418	0.0000026

Заключение

Использование рекурсии не во всех случаях будет считаться лучшим вариантом, она тратит меньше памяти и не использует циклы, но в то же время, выполняется дольше. Так же существует проблема переполнения стека, что ограничивает нас в возможностях, в связи с чем, некоторых вычисление некоторых значений будет являться приблизительным.

1. Статья, посвященная рекурсии и примерам программ с ее использованием [Электронный ресурс]. – Режим доступа: `https://habr.com/ru/post/319790/`. – Дата доступа: 30.03.2022.

Минчуков Никита Сергеевич, Усольцев Иван Дмитриевич, студенты факультета радиотехники и электроники БГУИР, nikitamin07@gmail.co, van_987@bk.ru.

Научный руководитель: Герасимов Вячеслав Александрович, ассистент кафедры вычислительных методов и программирования БГУИР, v.gerasimov@bsuir.by.