Практическая работа №5

ОЦЕНКА СЛОЖНОСТИ РЕКУРСИВНЫХ АЛГОРИТМОВ

Цель работы

Научиться разрабатывать рекурсивные алгоритмы и оценить их сложность.

2. Пояснения к работе

Перед выполнением задания изучить лекционный материал и теоретические сведения.

При выполнении практической работы обучающийся должен

Знать:

- Основные этапы разработки программного обеспечения

-Основные принципы технологии структурного и объектно-ориентированного программирования

Уметь:

- осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней

3. Теоретические сведения

Рекурсия – фундаментальное понятие в математике и компьютерных науках. В языках программирования рекурсивной программой называется программа, которая обращается сама к себе (подобно тому, как в математике рекурсивная функция определяется через понятия самой этой функции). Рекурсивная программа не может вызывать себя до бесконечности, следовательно, вторая важная особенность рекурсивной программы – наличие условия завершения, позволяющее программе прекратить вызывать себя. Таким образом рекурсия в программировании может быть определена как сведение задачи к такой же задаче, но манипулирующей более простыми данными. Как следствие, рекурсивная программа должна иметь как минимум два пути выполнения, один из которых предполагает рекурсивный вызов (случай «сложных» данных), а второй – без рекурсивного вызова (случай «простых» данных).

Рекурсивный алгоритм – это алгоритм, в описании которого прямо или косвенно содержится обращение к самому себе. В технике процедурного программирования данное понятие распространяется на функцию, которая реализует решение отдельного блока задачи посредством вызова из своего тела других функций, в том числе и себя самой. Если при этом на очередном этапе работы функция организует обращение к самой себе, то такая функция является рекурсивной.

Прямое обращение функции к самой себе предполагает, что в теле функции содержится вызов этой же функции, но с другим набором фактических параметров. Такой способ организации работы называется прямой рекурсией. Например, чтобы найти сумму первых n натуральных чисел, надо сумму первых (n-1) чисел сложить с числом n, то есть имеет место зависимость: Sn=Sn-1+n. Вычисление происходит с помощью аналогичных рассуждений. Такая цепочка взаимных обращений в конечном итоге сведется к вычислению суммы одного первого элемента, которая равна самому элементу.

При косвенном обращении функция содержит вызовы других функций из своего тела. При этом одна или несколько из вызываемых функций на определенном этапе обращаются к исходной функции с измененным набором входных параметров. Такая организация обращений называется косвенной рекурсией. Например, поиск максимального элемента в массиве размера n можно осуществлять как поиск максимума из двух чисел: одно их них – это последний элемент массива, а другое является максимальным элементом в массиве размера (n-1). Для нахождения максимального элемента массива размера (n-1) применяются аналогичные рассуждения. В итоге решение сводится к поиску максимального из первых двух элементов массива.

Рекурсивный метод в программировании предполагает разработку решения задачи, основываясь на свойствах рекурсивности отдельных объектов или закономерностей.При этом исходная задача сводится к решению аналогичных подзадач, которые являются более простыми и отличаются другим набором параметров.

Разработке рекурсивных алгоритмов предшествует рекурсивная триада – этапы моделирования задачи, на которых определяется набор параметров и соотношений между ними. Рекурсивную триаду составляют параметризация, выделение базы и декомпозиция.

На этапе параметризации из постановки задачи выделяются параметры, которые описывают исходные данные. При этом некоторые дальнейшие разработки решения могут требовать введения дополнительных параметров, которые не оговорены в условии, но используются при составлении зависимостей. Необходимость в дополнительных параметрах часто возникает также при решении задач оптимизации рекурсивных алгоритмов, в ходе которых сокращается их временная сложность.

Выделение базы рекурсии предполагает нахождение в решаемой задаче тривиальных случаев, результат для которых очевиден и не требует проведения расчетов. Верно найденная база рекурсии обеспечивает завершенность рекурсивных обращений, которые в конечном итоге сводятся к базовому случаю. Переопределение базы или ее динамическое расширение в ходе решения задачи часто позволяют оптимизировать рекурсивный алгоритм за счет достижения базового случая за более короткий путь обращений.

Декомпозиция представляет собой сведение общего случая к более простым подзадачам, которые отличаются от исходной задачи набором входных данных. Декомпозиционные зависимости описывают не только связь между задачей и подзадачами, но и характер изменения значений параметров на очередном шаге. От выбранных отношений зависит трудоемкость алгоритма, так как для одной и той же задачи могут быть составлены различные зависимости. Пересмотр отношений декомпозиции целесообразно проводить комплексно, то есть параллельно с корректировкой параметров и анализом базовых случаев.

4. Задание

Разработать рекурсивный алгоритм вычисления факториала и оценить его сложность.

5. Порядок выполнения работы

1. Проведите анализ предметной области в соответствии с выбранной темой по следующему плану:

- описать пользователей базы данных;
- составить перечень функций базы данных;
- описать входную и выходную информацию.

2. Дайте ответы на контрольные вопросы.

3. Оформите отчет.

6. Содержание отчета

Отчет должен быть выполнен в соответствии с Общими требованиями к оформлению документов учебной деятельности обучающихся. Отчет должен содержать следующие разделы:

1. Наименование работы.
2. Цель работы.
3. Конечные результаты выполненной работы в виде скриншотов.
4. Ответы на контрольные вопросы.
5. Вывод.

7. Контрольные вопросы

Что такое рекурсивный алгоритм?