Лабораторная работа №3
Оценка сложности рекурсивных алгоритмов

1Цель работы

1.1 Научиться разрабатывать и оценивать сложность рекурсивных функций в программах на C#.

2 Литература

2.1 Фленов, М. Е. Библия C#. – 3 изд. – Санкт-Петербург: БХВ-Петербург, 2016. – URL: https://ibooks.ru/bookshelf/353561/reading. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный. – п.3.3.6.

3 Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).

3.2 Изучить описание лабораторной работы.

4 Основное оборудование

4.1 Персональный
компьютер.

5 Задание

Для упрощения подсчета количества вызовов рекурсивной функции выводить в консоль отладки информацию, какая рекурсивная функция вызвана и количество ее вызовов. Для реализации вывода в начале рекурсивной функции добавить точку останова с действием:

$HITCOUNT – количество срабатываний точки останова

$FUNCTION – имя функции

Если точка останова не должна участвовать в подсчете, выключать ее, но не удалять.

Для тестирования корректности работы методов использовать Debug.Assert(), подобрав минимально необходимый набор тестов. Предусмотреть проверку на корректные и некорректные значения.

5.1 Написать и протестировать рекурсивную функцию Power для вычисления x^n, где n – любое целое.

Поиск x^n, где n – отрицательное, осуществлять по формуле: $x^n = 1/x^{-n}$

Стандартный метод возведения в степень не использовать.

Оценить сложность разработанного алгоритма.

5.2 Написать и протестировать рекурсивную функцию FastPower для быстрого вычисления x^n, где n – неотрицательное целое, используя возведение в квадрат.

Для ускорения работы рекурсия должна вызываться в ветке алгоритма не более одного раза (для этого использовать сохранение промежуточного результата $x^{n/2}$ перед возведением в квадрат).

Пример (вместо 15 заходов будет 4 или 7):

$$a^{15} = a*(a^7)^2 = a*(a*(a^3)^2)^2 = a*(a*(a*(a^2)^2)^2$$

Для некорректных данных возвращать -1.

Стандартный метод возведения в степень не использовать.

Оценить сложность разработанного алгоритма.

5.3 Написать и протестировать рекурсивную функцию вычисления числа Фибоначчи двумя способами:

- 1 способ: Fibonacci(int n) – рекурсивное вычисление с вызовом рекурсивной функции для двух предыдущих чисел,

- 2 способ: FibonacciTail(int n, long a=0, long b=1) – вычисление с хвостовой рекурсией. Для реализации n на каждом шаге уменьшается, а значения a (первое слагаемое в формуле) и b (второе слагаемое в формуле) заменяются на следующие по возрастанию, например:

n=7 0 1 1

n=6 1 1 2

n=5 1 2 3

n=4 2 3 5

n=3 3 5 8

n=2 5 8 13

Для некорректных данных возвращать -1.

Оценить сложность разработанных алгоритмов.

6 Порядок выполнения работы

6.1 Запустить MS Visual Studio и использовать решение с названием LabWork3.

6.2 Выполнить все задания из п.5 в решении LabWork3. Для каждого задания создать в консольном проекте рекурсивную функцию в виде статического метода (по возможности используя тернарный оператор).

При выполнении заданий использовать минимально возможное количество команд и переменных, выполнять форматирование и рефакторинг кода.

6.3 Ответить на контрольные вопросы.

7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Ответы на контрольные вопросы

7.4 Вывод

8 Контрольные вопросы

8.1 Что такое «рекурсия»?

8.2 Какие проблемы могут возникать при реализации рекурсивных алгоритмов?

8.3 Что такое «глубина рекурсии»?

8.4 Что такое «рекурсивный спуск»?

8.5 Что такое «рекурсивный подъём»?

8.6 Что такое «хвостовая рекурсия»?

9 Приложение

Рекурсивный вызов метода – случай, когда метод вызывает сам себя. В рекурсии обязательно должен быть предусмотрен базовый случай.

Пример:

```
static void MethodName()
{
   MethodName();
}
```