Изображение выглядит как текст, керамическая посуда

Автоматически созданное описание

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №6**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема. Рекурсивные алгоритмы и их реализация

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-06-22 |  | Кликушин В.И. |
| Принял старший преподаватель |  | Скворцова Л.А. |

Москва 2023

ОГЛАВЛЕНИЕ

[1 ОТВЕТЫ НА ВОПРОСЫ 3](#_Toc133007842)

[2 ОТЧЁТ ПО ЗАДАЧЕ 6](#_Toc133007843)

[2.1 УСЛОВИЕ ЗАДАНИЯ, ЗАДАЧА ВАРИАНТА, ТРЕБОВАНИЯ ПО ВЫПОЛНЕНИЮ 6](#_Toc133007844)

[2.2 ПОСТАНОВКА ЗАДАЧИ 6](#_Toc133007845)

[2.3 ОПИСАНИЕ АЛГОРИТМА 6](#_Toc133007846)

[2.3.1 Рекуррентная зависимость 6](#_Toc133007847)

[2.3.2 Глубина рекурсии 6](#_Toc133007848)

[2.3.4 Анализ сложности алгоритма 6](#_Toc133007849)

[2.4 ДЕРЕВО РЕКУРСИВНЫХ ВЫЗОВОВ 8](#_Toc133007850)

[2.5 ТАБЛИЦА ТЕСТИРОВАНИЯ 8](#_Toc133007851)

[2.6 КОД ПРОГРАММЫ 9](#_Toc133007852)

[2.7 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ 9](#_Toc133007853)

[3 ВЫВОДЫ 11](#_Toc133007854)

[4 ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ 12](#_Toc133007855)

# 1 ОТВЕТЫ НА ВОПРОСЫ

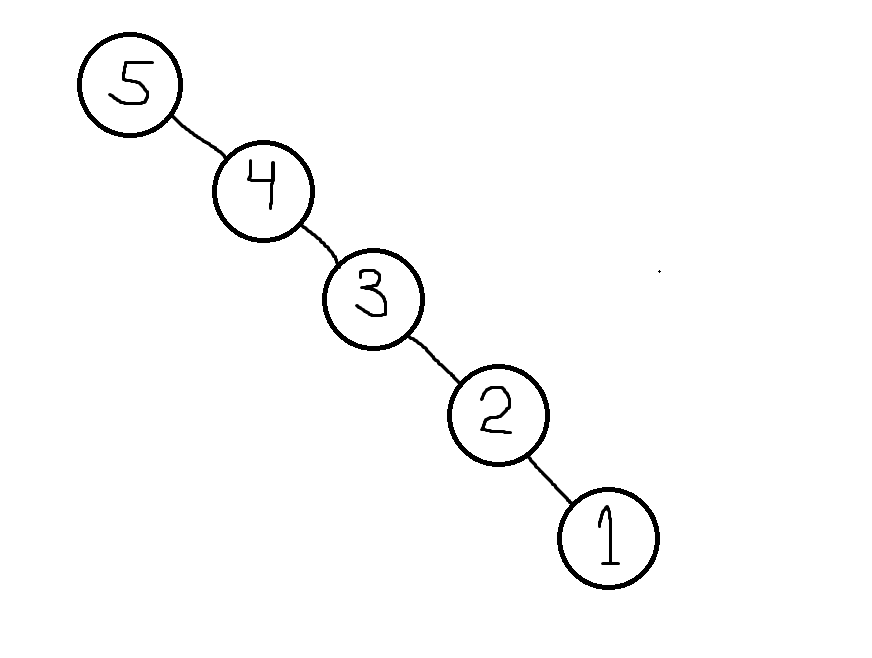
1. Рекурсивная функция – это функция, которая вызывает сама себя, и при каждом очередном вызове использует данные, созданные во время предыдущего вызова.
2. Шаг рекурсии – это активизация очередного рекурсивного выполнения алгоритма при других исходных данных.
3. Глубина рекурсии – наибольшее одновременное количество рекурсивных вызовов функции, определяющее максимальное количество слоев рекурсивного стека, в котором осуществляется хранение отложенных вычислений.
4. Условие завершения рекурсии – условие, которое определяет завершение рекурсии и формирование конкретного выходного значения вычислительного процесса.
5. В зависимости от количества и места размещения рекурсивных вызовов различаются следующие виды рекурсивных процессов. Линейная рекурсия: каждый вызов порождает ровно один новый вызов. Является простой и не приводит к большому количеству выполнения операций. Каскадная рекурсия: каждый вызов порождает несколько новых вызовов. Сложность алгоритма значительно возрастает по сравнению с каскадной рекурсией.
6. По организации обращения к подзадачам рекурсию разделяют на прямую и косвенную. Косвенная рекурсия имеет место, если алгоритм А вызывает алгоритм В, и алгоритм В вызывает алгоритм А. Прямая рекурсия имеет место, если решение задачи сводится к разделению ее на меньшие подзадачи, выполняемые с помощью одного и того же алгоритма. Процесс разбиения завершается, когда достигается простейшее возможное решение.
7. Данные функции в момент ее выполнения хранятся в локальных переменных в разделе оперативной памяти, организованной по правилу стека. Каждый рекурсивный вызов является самостоятельным вызовом функции и для него так же, как и для первого вызова, создаются локальные переменные, которые заполняются при этом вызове. При каждом новом рекурсивном вызове функции в стеке создается новое множество локальных переменных и форменных параметров, их имена одинаковы, но они имеют различные значения. Продолжается это до момента, пока алгоритм не дойдёт до условия выхода из цикла. Пример заполнения стека при вычислении 5! :

Изображение выглядит как стол

Автоматически созданное описаниеРисунок 1 – Заполнение стека при рекурсивных вызовах функции

Тв - точка возврата указывает на оператор в коде функции, куда вернуть результат, символ n над ячейкой таблицы на рисунке 1 отмечает переданное на обработку значение в каждом шаге рекурсии.

1. Дерево рекурсии – граф, помогающий отобразить глубину рекурсии. Дерево рекурсии – это совершенное дерево, т.е. на каждом уровне максимально допустимое количество узлов, кроме последнего. На последнем уровне размещаются вершины – листья, показывающие вызовы, приводящие к завершению вычислений. Уровни нумеруются с 0. Количество узлов на уровне зависит от степени дерева, так если степень дерева 2 (дерево бинарное), то количество узлов на i-ом уровне равно . Пример дерева рекурсии вычисления 5! :

Рисунок 2 – Дерево рекурсии вычисления 5!

Пример дерева рекурсии вычисления пятого числа Фибоначчи:

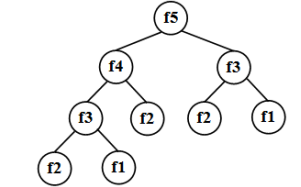


Рисунок 3 – Дерево рекурсии вычисления пятого числа Фибоначчи

# 2 ОТЧЁТ ПО ЗАДАЧЕ

## 2.1 УСЛОВИЕ ЗАДАНИЯ, ЗАДАЧА ВАРИАНТА, ТРЕБОВАНИЯ ПО ВЫПОЛНЕНИЮ

Условие задания: Разработать и протестировать рекурсивные функции в соответствии с задачами варианта.

Задача варианта: Дано десятичное натуральное число N. Выведите все его цифры по одной, в обычном порядке, разделяя их пробелами или новыми строками. Примечание. При решении этой задачи нельзя использовать строки, списки, массивы (ну и циклы, разумеется). Разрешена только рекурсия и целочисленная арифметика.

Требования по выполнению: описать рекуррентную зависимость в решении задачи, реализовать и отладить рекурсивную функцию решения задачи, определить глубину рекурсии при введенных исходных данных, определить сложность рекурсивного алгоритма, используя метод подстановки и дерево рекурсии, привести для одного из значений схему рекурсивных вызовов, разработать программу, демонстрирующую выполнение функции, и показать результаты тестирования.

## 2.2 ПОСТАНОВКА ЗАДАЧИ

Необходимо изучить теоретический материал о рекурсивных алгоритмах и разработать программу для решения задачи варианта, используя только рекурсивный алгоритм и целочисленную арифметику.

## 2.3 ОПИСАНИЕ АЛГОРИТМА

### 2.3.1 Рекуррентная зависимость

Опишем рекуррентную зависимость функции. Функция будет вызывать саму себя до момента, пока n не станет равным нулю. В этом случае начнется печать c пробелом последней цифры для каждой «новой n», для рекурсий из стека.

### 2.3.2 Глубина рекурсии

Глядя на дерево рекурсии, можно сделать вывод о том, что глубина заданной рекурсии равна O(n).

### 2.3.4 Анализ сложности алгоритма

Лучший случай: вводится однозначное число.

За худший случай примем ввод максимально допустимого числа, которое может быть записано в типе int в языке C++. «Правая граница» возможного диапазона целых чисел: 2 147 483 647. Числа, большие, чем максимально допустимое будут приводиться к этому числу (результат работы рекурсии для них будет аналогичен). Составим рекуррентное уравнение для времени работы функции. Каждый рекурсивный вызов требует константного времени плюс время на рекурсивный вызов, в свою очередь осуществляемый им, что приводит к рекуррентному соотношению:

Решим данное уравнение методом подстановки (определим верхнюю границу). Предположим, что решение имеет вид T(n) = O(n). Докажем, применяя метод индукции, что при подходящем выборе константы с > 0 выполняется неравенство Т (n) ≤ с\*n. Предположим справедливость этого неравенства для величины T(n-1):

T(n) <= c\*(n-1) + 1 = c\*n – c+1. Тогда, T(n) <= c\*n + (1-c). Итого T(n) <= c\*n. Теперь построим дерево рекурсии, чтобы убедиться в правильности оценки сложности алгоритма.

## 2.4 ДЕРЕВО РЕКУРСИВНЫХ ВЫЗОВОВ

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 4 – Дерево рекурсии вывода цифр числа 1234

## 2.5 ТАБЛИЦА ТЕСТИРОВАНИЯ

Таблица 1 – Таблица тестирования рекурсивного алгоритма

|  |  |  |
| --- | --- | --- |
| Вывод цифр десятичного числа в обычном порядке через пробел  print\_numbers(int n) | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | n = 1234567899 | 1 2 3 4 5 6 7 8 9 9 |
| 2 | n = 1000000000 | 1 0 0 0 0 0 0 0 0 0 |
| 3 | n = 7 | 7 |
| 4 | n = 878 | 8 7 8 |
| 5 | n = 4321 | 4 3 2 1 |

## 2.6 КОД ПРОГРАММЫ

#include <iostream>

using namespace std;

int print\_numbers(int n)

{

if (n == 0)

{

return 0;

}

print\_numbers(n / 10);

cout << n % 10 << " ";

}

int main()

{

system("chcp 1251");

int n;

cout << " Введите десятичное число число: \n";

cin >> n;

print\_numbers(n);

}

## 2.7 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 1 – Результат теста 1

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 2 – Результат теста 2

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 3 – Результат теста 3

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Результат теста 4

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Результат теста 5

# 3 ВЫВОДЫ

В ходе практической работы я изучил понятие рекурсивного алгоритма, познакомился с видами рекурсии, решил практическую задачу с использованием рекурсии, а также научился оценивать сложность рекурсивной функции.

# 4 ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ

1. Учебно-методическое пособие СиАОД (часть 1)

2. Приложение к практическим работам – СДО (online-edu.mirea.ru)