МИНОРБНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение высшего образования*
***«МИРЭА - Российский технологический университет»***

**РТУ МИРЭА**

---

Отчёт по выполнению практического задания № 7
**Тема:**
«Рекурсивные алгоритмы и их реализация»
Дисциплина: «Структуры и алгоритмы обработки данных»


Выполнил студент: Фамилия И.О.
Фамилия И.О.

Группа: АААА-00-00
Номер группы


Москва 2023

# СОДЕРЖАНИЕ

# ЗАДАНИЕ 1

## 1.1 Условие задачи

Разработать и протестировать рекурсивные функции в соответствии с задачами варианта

1) Требования к выполнению первой задачи варианта:

- приведите итерационный алгоритм решения задачи
- реализуйте алгоритм в виде функции и отладьте его
- определите теоретическую сложность алгоритма
- опишите рекуррентную зависимость в решении задачи
- реализуйте и отладьте рекурсивную функцию решения задачи
- определите глубину рекурсии, изменяя исходные данные
- определите сложность рекурсивного алгоритма, используя метод подстановки и дерево рекурсии
- приведите для одного из значений схему рекурсивных вызовов
- разработайте программу, демонстрирующую выполнение обеих функций и покажите результаты тестирования.

## 1.2 Постановка задачи

Найти наибольший общий делитель двух целых чисел

## 1.3 Описание алгоритма

Итеративный алгоритм использует алгоритм Евклида для поиска НОД. Код алгоритма представлен в листинге 1.

Листинг 1 - Итеративный алгоритм НОД.

```
int gcd_i(int a, int b)
{
    while (b)
    {
        a %= b;
        swap(a, b);
    }
    return a;
}
```

Алгоритм Евклида применяется к паре положительных целых чисел и формирует новую пару, которая состоит из меньшего числа и остатка от деления большего числа на меньшее. Процесс повторяется, пока числа не станут равными.

Рекурсивный алгоритм представляет собой тот же самый алгоритм Евклида, и работает точно так же. Код представлен в листинге 2.

Листинг 2 - Рекурсивный алгоритм НОД.

```
int gcd_r(int a, int b)
{
    if (a == 0)
        return b;
    return gcd_r(b % a, a);
}
```

### 1.4 Анализ алгоритма

Теоретическая сложность алгоритма Евклида для обоих алгоритмов одинакова и известна, выводится и равна O(log(min(a,b)).

Запишем рекуррентную зависимость алгоритма Евклида. (формула 1).

$$f(a,b)=\begin{cases} b & \text{если } a=0 \\ f(b\%a,a) & \text{если } a\neq 0 \end{cases} \quad (1)$$

Анализируя значения тестов можно прийти к выводу, что глубина рекурсии также будет равна log(min(a,b).

Приведём схему рекуррентных вызовов для a = 98 и b = 54 (рис. 1).


Рисунок 1 - Схема рекуррентных вызовов

## 1.5 Результаты тестирования

Результаты тестирования представлены на рис. 2.

```
НОД чисел 98 и 56
Итеративно: 14
Глубина рекурсии: 4
Рекурсивно: 14

НОД чисел 642 и 432
Итеративно: 6
Глубина рекурсии: 9
Рекурсивно: 6
```

Рисунок 2 - Результаты тестирования

# ЗАДАНИЕ 2

## 2.1 Условие задачи

2) Требования к выполнению второй задачи варианта:

• рекурсивную функцию для обработки списковой структуры согласно варианту. Информационная часть узла – простого типа – целого;

• для создания списка может быть разработана простая или рекурсивная функция по желанию (в тех вариантах, где не требуется рекурсивное создание списка);

• определите глубину рекурсии

• определите теоретическую сложность алгоритма

• разработайте программу, демонстрирующую работу функций и покажите результаты тестов.

## 2.2 Постановка задачи

Создание и вывод линейного однонаправленного списка из n элементов.

## 2.3 Описание алгоритма

Алгоритм создания списка представляет собой добавление элемента в конец списка до тех пор, пока массив переданных коэффициентов не опустеет.

Алгоритм вывода списка представляет собой вывод переданного элемента и передача в рекурсию следующего до тех пор, пока следующий элемент существует.

Коды алгоритмов представлены в листингах 3 и 4.

Листинг 3 - Рекурсивное создание списка

```
List create_list(List list, vector<int> values)
{
    if (values.size() == 0)
    {
        return list;
    }

    int _val = values[0];
    values.erase(values.begin());
    Node *p = new Node(_val);

    if (list.is_empty())
```

```
    {
        list.first = p;
        list.last = p;
    }
    else
    {
        list.last->next = p;
        list.last = p;
    }
    return create_list(list, values);
}
```

Листинг 4 - Рекурсивный вывод списка

```
void print_list(Node *p)
{
    cout << p->val << " ";
    if (p->next == nullptr)
        return;
    print_list(p->next);
}
```

### 2.4 Анализ алгоритма

Теоретическая сложность для обоих алгоритмов одинакова и будет равна длине создаваемого/выводимого списка, поскольку вызов выполняемые операции производятся поэлементно, поэтому сложность равна O(n).

Запишем рекуррентную зависимость создания списка. (формула 2).

$$f(\text{list}, \text{value}) = \begin{cases} \text{list} & \text{если values} = 0 \\ f(\text{list}, \text{values}[1:]) & \text{если values} \neq 0 \end{cases} \quad (2)$$

Запишем рекуррентную зависимость вывода списка. (формула 3).

$$f(p) = \begin{cases} \text{выход} & \text{если } p \to \text{next} = 0 \\ f(p \to \text{next}) & \text{если } p \to \text{next} \neq 0 \end{cases} \quad (3)$$

Анализируя значения тестов можно прийти к выводу, что глубина рекурсии также будет равна количеству элементов — n.

Приведём схему рекуррентных вызовов для массива 1 2 3 4 5 (рис. 4). В данном случае схемы будут выглядеть идентично для обоих алгоритмов.


Рисунок 3 - Схема рекуррентных вызовов

## 2.5 Результаты тестирования

Результаты тестирования представлены на рис. 4.

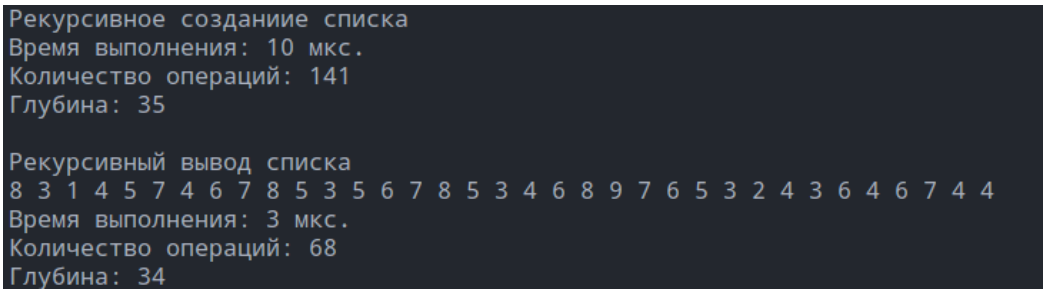


Рисунок 4 - Результаты тестирования

# ВЫВОД

Была проделана работа по изучению рекуррентных алгоритмов. Были выведены теоретическая сложность данных алгоритмов, рекуррентная зависимость и глубина рекурсии.

## ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Структуры и алгоритмы обработки данных : Лекционные материалы / Рысин М. Л. МИРЭА — Российский технологический университет, 2022/23. – 77 с.

2. Алгоритм Евклида — Материал из Википедии — свободной энциклопедии [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Алгоритм_Евклида (дата обращения 10.04.2024)