МИНОБРНАУКИ РОСИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания №1.7
Тема:
Рекурсивные алгоритмы и их реализация
Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Павлов Н.С.

Группа: ИКБО-30-23

Москва 2024

СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ

Получить знания и практические навыки по разработке и реализации рекурсивных процессов

1 ЗАДАНИЕ №1

1.1 ФОРМУЛИРОВКА ЗАДАЧИ

Разработать и протестировать рекурсивные функции в соответствии с задачами варианта

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.

Персональный вариант:

Определить является ли текст – палиндромом

1.2 РАБОТА С АЛГОРИТМОМ

1.2.1 Рекуррентная зависимость

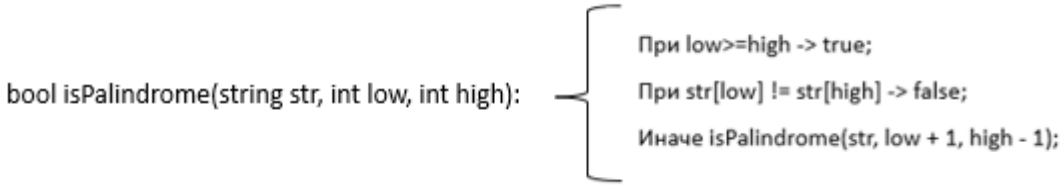

Рисунок 1 – Рекуррентная зависимость

1.2.2 Реализация используемых функций

```
bool Palindrome(string str)
{
    int low = 0;
    int high = str.length() - 1;
    while (low < high) {
        if (str[low] != str[high]) {
            return false;
        }
        low++;
        high--;
    }
    return true;
}
```

Рисунок 2 – Реализация итерационного алгоритма

```
bool Palindrome_recursion(string str, int low, int high)
{
    if (low >= high) {
        return true;
    }
    if (str[low] != str[high]) {
        return false;
    }
    return Palindrome_recursion(str, low + 1, high - 1);
}
```

Рисунок 3 – Реализация рекурсивного алгоритма

1.2.3 Ответы на задания по задаче 1

1. Итерационный алгоритм решения задачи представлен на рисунке 4

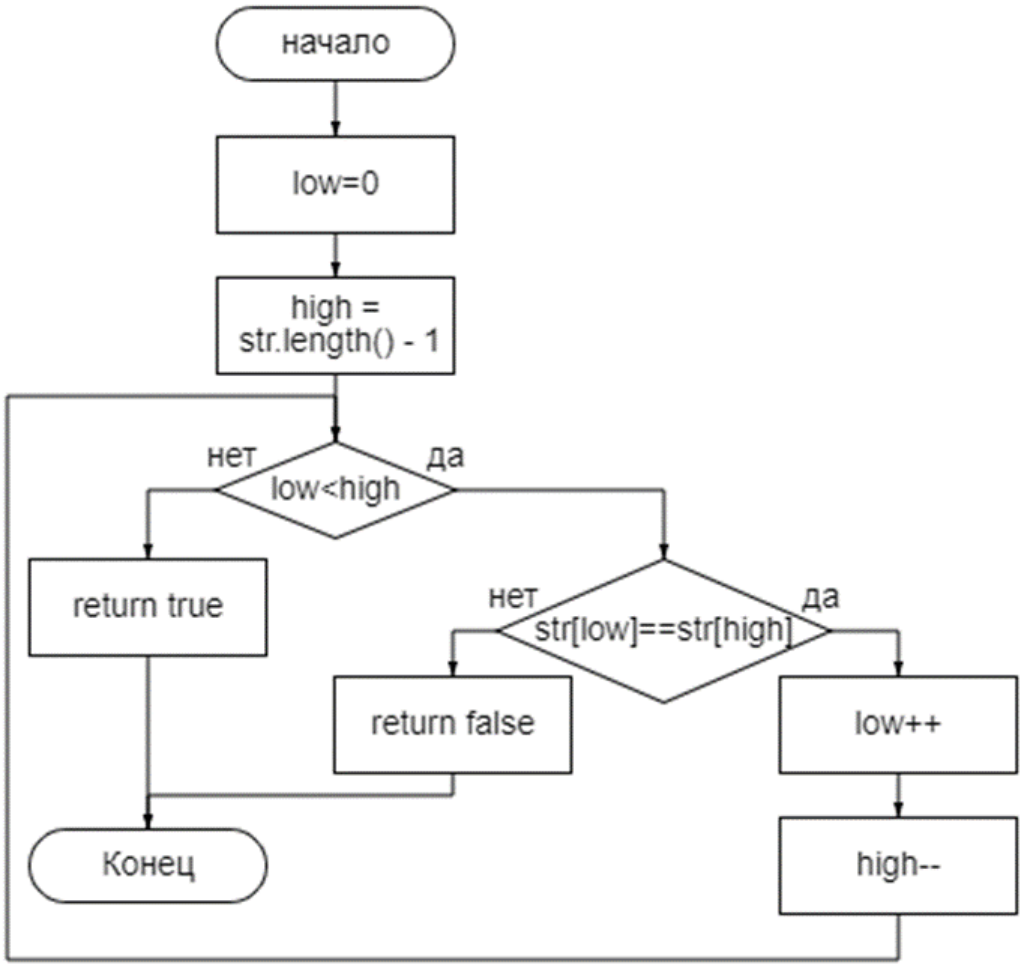

Рисунок 4 – Блок-схема итерационного алгоритма

2. Теоретическая сложность алгоритма – линейная, так как в худшем случае количество операций линейно зависит от длины введенной строки

3. Глубина рекурсии зависит от входных данных. Худшим случаем является строка-палиндром, при котором глубина рекурсии составляет половину длины введенной строки.

4. С помощью методов подстановки и дерева рекурсии ясно, что сложность рекурсивного алгоритма также линейна, это можно заметить на примере (рис. 5)

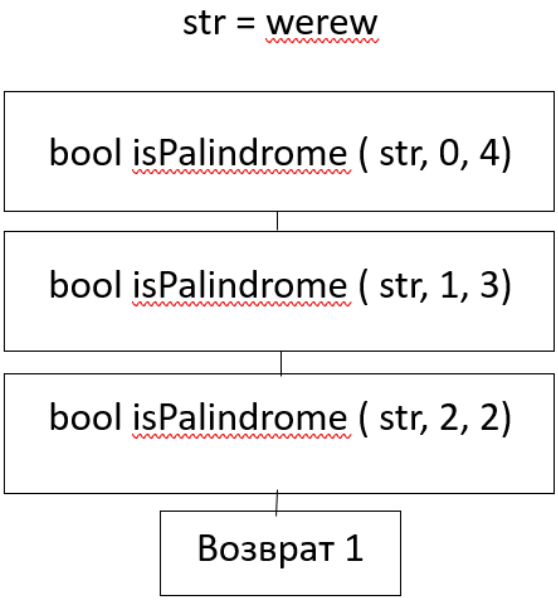

Рисунок 5 – Дерево рекурсии для одного из значений

1.2.4 Реализация программы и тестирование

```
cout << "Задание №1" << endl << endl;
cout << "Введите строку: ";
string str;
getline(cin, str);
cout << "Введенная строка: " << str << endl;
int len = str.length();
cout << "Результат итерационного алгоритма: ";
if (Palindrome(str)) {
    cout << "Палиндром" << endl;
}
else {
    cout << "Не палиндром" << endl;
}
cout << "Результат рекурсивного алгоритма: ";
if (Palindrome_recursion(str, 0, len - 1)) {
    cout << "Палиндром" << endl;
}
else {
    cout << "Не палиндром" << endl;
}
```

Рисунок 6 – Реализация программы на языке C++

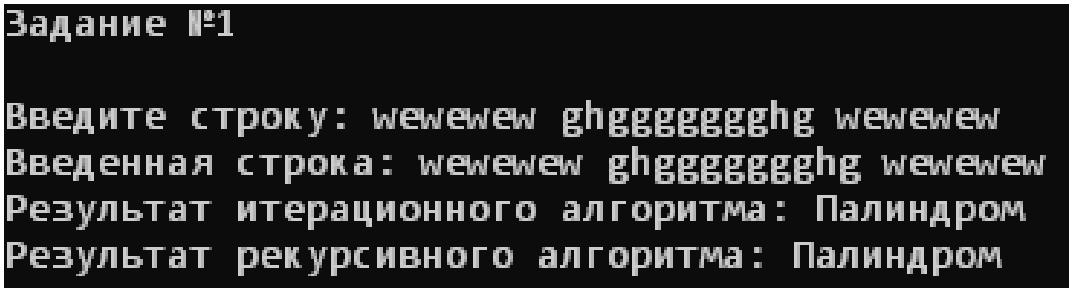

Рисунок 7 – Тестирование алгоритма для строки-палиндрома

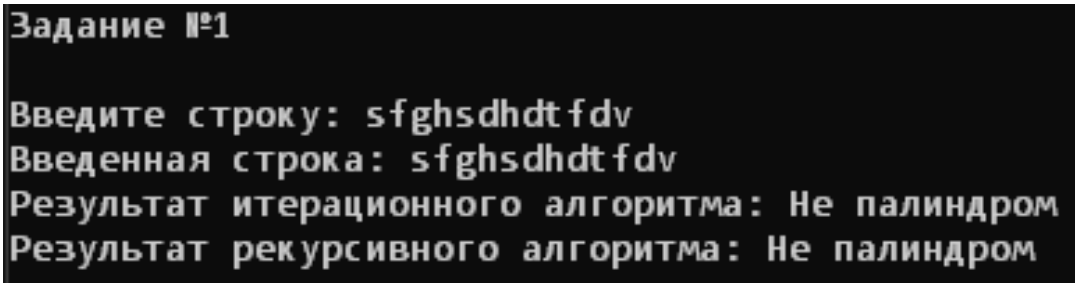

Рисунок 8 – Тестирование для произвольной строки

2 ЗАДАНИЕ №2

2.1 ПОСТАНОВКА ЗАДАЧИ

Разработать и протестировать рекурсивные функции в соответствии с задачами варианта

Требования к выполнению первой 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.

Персональный вариант:

Удалить из связанного однонаправленного списка все элементы, равные заданному.

2.2 РАБОТА С АЛГОРИТМОМ

2.2.1 Рекуррентная зависимость

del(Node* &first, char key)

- При first = nullptr -> возврат;
- При first.val = key -> а) удалить first;
  б) del(first, key);
- При first.val != key -> del(first->next, key)

Рисунок 9 – Рекуррентная зависимость

2.2.2 Реализация используемых функций

```
struct Node
{
    int val;
    Node* next;
    Node(char _val) : val(_val), next(nullptr) {}
};

struct list
{
    Node* first;
    Node* last;
    list() : first(nullptr), last(nullptr) {}

    bool is_empty() { ... }
    void push_back(int _val) { ... }
    void print() { ... }
};
```

Рисунок 10 – Структура узла и списка

```
void del(Node*& first, char key)
{
    if (first == nullptr)
        return;
    if (first->val == key) {
        Node* temp = first;
        first = first->next;
        delete temp;
        del(first, key);
    }
    else {
        del(first->next, key);
    }
}
```

Рисунок 11 – Функция удаления ключа из списка

2.2.3 Ответы на задания по задаче 1

1. Глубина рекурсии зависит от входных данных и равняется количеству введенных элементов (размер списка)

2. Теоретическая сложность алгоритма – линейная, так как в любом случае количество операций линейно зависит от количества введенных элементов.

2.2.4 Реализация программы и тестирование

```
cout << endl << "Задание №2" << endl << endl;
int key = 1;
list a;
for (int i = 0; i < 5; i++) {
    a.push_back(1);
}
for (int i = 0; i < 7; i++) {
    a.push_back(2);
}
for (int i = 0; i < 3; i++) {
    a.push_back(1);
}
for (int i = 0; i < 3; i++) {
    a.push_back(5);
}
for (int i = 0; i < 2; i++) {
    a.push_back(4);
}
a.push_back(1);
cout << "Удаляемое значение: " << key << endl;
cout << "Исходный список: ";
a.print();
cout << "Полученный список: ";
del(a.first, key);
a.print();
```

Рисунок 12 – Реализация алгоритма на языке C++

```
Задание №2

Удаляемое значение: 1
Исходный список: 1 1 1 1 1 2 2 2 2 2 2 2 1 1 1 5 5 5 4 4 1
Полученный список: 2 2 2 2 2 2 2 5 5 5 4 4
```

Рисунок 13 – Результат тестирования алгоритма

3 ВЫВОД

Мы получили знания и практические навыки по разработке и реализации рекурсивных процессов

СПИСОК ИНФОРМАЦТОННЫХ ИСТОЧНИКОВ

1. Рысин, М. Л. Введение в структуры и алгоритмы обработки данных. Часть 1. Сложность алгоритмов. Сортировки. Линейные структуры данных. Поиск в таблице. : учеб. пособие / М. Л. Рысин, М. В. Сартаков, М. Б. Туманова ; МИРЭА – Российский технологический университет, 2022. – 109 с. – ISBN 978-5-7339-1612-5.

2. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения : межгосударственный стандарт : дата введения 1992-01-01 / Федеральное агентство по техническому регулированию. – Изд. официальное. – Москва : Стандартинформ, 2010. – 23 с.