



МИНОБРНАУКИ РОССИИ

*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: Враженко Д.О.

Группа: ИКБО-10-23

Вариант: 10

Москва – 2024

ЦЕЛЬ РАБОТЫ

Получить знания и практические навыки по разработке и реализации рекурсивных процессов.

ХОД РАБОТЫ

Индивидуальный вариант:

Вариант 10: 1. Вычислить значение цифрового корня для некоторого целого числа N. 2. Найти в двунаправленном списке количество четных элементов.

1. Задача 1

1.1. Условие задачи

Вычислить значение цифрового корня для некоторого целого числа N.

1.2. Описание алгоритма

Этот алгоритм является рекурсивной функцией, которая выполняет следующие действия:

- 1) Принимает на вход одно целое число `number`.
- 2) Внутри функции создается переменная `sum`, которая инициализируется значением 0.
- 3) Затем начинается цикл `while`, который продолжается до тех пор, пока значение переменной `number` больше 0.
- 4) В каждой итерации цикла к переменной `sum` прибавляется последняя цифра числа `number` (получается с помощью операции остатка от деления на 10), а затем само значение переменной `number` делится на 10, чтобы удалить последнюю цифру.
- 5) После завершения цикла проверяется значение переменной `sum`. Если оно меньше 10, функция возвращает это значение.
- 6) Иначе функция вызывает саму себя, передавая в неё значение переменной `sum` в качестве аргумента.

1.3. Коды используемых функций

На рис. 1 представлен код дополнительной функции для переработки числа из вещественного в целочисленное.

```

int func_1(int number)
{
    int sum = 0;
    while (number > 0)
    {
        sum += number % 10;
        number /= 10;
    }
    if (sum < 10)
        return sum;
    return func_1(sum);
}

```

Рисунок 1 - Код дополнительной функции

1.4. Ответы на задания по задаче 1

Теоретическая сложность алгоритма: $O(n \cdot \log_{10}(sum))$, где n - кол-во цифр в исходном числе, а sum - сумма цифр числа $number$.

На рис. 2 представлен итерационный алгоритм решения задачи.

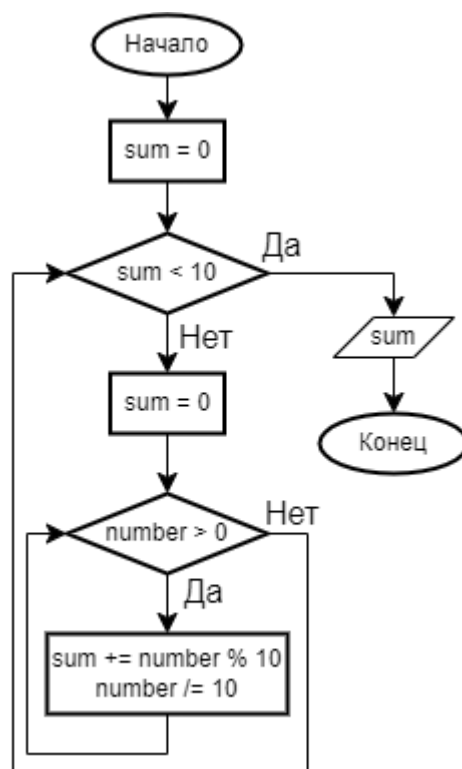


Рисунок 2 - Итерационный алгоритм

На рис. 3 представлена схема рекурсивных вызовов.

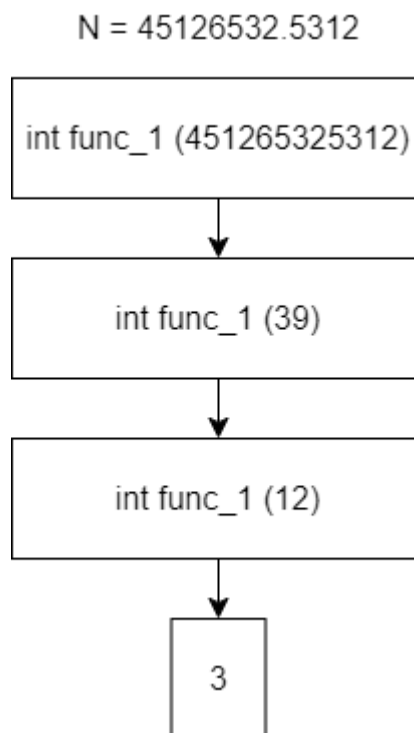


Рисунок 3 - Схема
рекурсивных вызовов

1.5. Код программы и скриншоты результатов тестирования

На рис. 4 представлен код алгоритма для извлечения цифрового корня.

```

int func_1(int number)
{
    int sum = 0;
    while (number > 0)
    {
        sum += number % 10;
        number /= 10;
    }
    if (sum < 10)
        return sum;
    return func_1(sum);
}
  
```

Рисунок 4 - Код алгоритма

На рис. 5-7 представлены тесты программы.

```

Введите номер задания (1, 2): 1
Введите число: 123
Цифровой корень: 6
  
```

Рисунок 5 - Пример 1

```
Введите номер задания (1, 2): 1  
Введите число: 123.4  
Цифровой корень: 1
```

Рисунок 6 - Пример 2

```
Введите номер задания (1, 2): 1  
Введите число: 412.86  
Цифровой корень: 3
```

Рисунок 7 - Пример 3

2. Задача 2

2.1. Условие задачи

Найти в двунаправленном списке количество четных элементов.

2.2. Описание алгоритма

- 1) Принимает указатель на текущий узел списка в качестве параметра.
- 2) Проверяет, является ли значение данных текущего узла четным.
- 3) Если значение данных четное, то функция возвращает 1, иначе возвращает значение 0.
- 4) Затем функция вызывает саму себя для следующего узла списка.
- 5) Результаты рекурсивных вызовов суммируются и возвращаются как общее количество четных элементов в списке.

2.3. Коды используемых функций

На рис. 8 представлены коды структуры данных и класса двунаправленного списка, необходимых для решения задания.

```

struct Node {
    int data;
    Node* prev;
    Node* next;
    Node(int val) : data(val), prev(nullptr), next(nullptr) {}
};

class DLL {
private:
    Node* head;
    Node* tail;
    int size;
    int count(Node* node) {
        if (!node) return 0;
        return (node->data % 2 == 0 ? 1 : 0) + count(node->next);
    }
public:
    DLL() : head(nullptr), tail(nullptr), size(0) {}
    void add(int val)
    {
        Node* newNode = new Node(val);
        if (!head) head = tail = newNode;
        else
        {
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
        }
        size++;
    }
    int count()
    { return count(head); }
    void print()
    {
        Node* current = head;
        while (current)
        {
            cout << current->data << ' ';
            current = current->next;
        }
        cout << endl;
    }
};

```

Рисунок 8 - Структура данных и класс двунаправленного списка

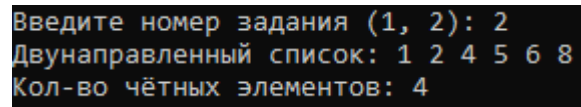
2.4. Ответы на задания по задаче 2

Глубина рекурсии равна количеству элементов в двунаправленном списке.

Теоретическая сложность равна $O(n)$.

2.5. Код программы и скриншоты результатов тестирования

На рис. 9 представлен результат тестирования программы.

A screenshot of a terminal window showing the output of a program. The text is as follows:

```
Введите номер задания (1, 2): 2
Двунаправленный список: 1 2 4 5 6 8
Кол-во чётных элементов: 4
```

Рисунок 9 - Результат тестирования

Вывод:

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