*Федеральное государственное бюджетное образовательное учреждение высшего образования*

***«Московский государственный технический университет имени Н.Э. Баумана»***

***(национальный исследовательский университет)***

***(МГТУ им. Н.Э. Баумана)***

---

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

# Отчет

**по домашнему заданию № 3**

**Название домашнего задания: Динамические структуры данных и файловая система.**

**Дисциплина: Алгоритмизация и программирование**

Студент гр. ИУ6-14Б **22.11.2023** **А.С. Воеводин**

(Подпись, дата) (И.О. Фамилия)

Преподаватель **22.11.2023** **О.А. Веселовская**

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

**Цель задания –** Работа с динамическими структурами данных и файловой системой.

**Задание –** Часть 1. Списки

Создать список по типу очереди из вещественных чисел и их целых частей. Удалить числа с нечетными номерами. При завершении программы освободить динамическую память.

Часть 2. Файлы

Создать текстовый файл F, содержащий латинские слова. Переписать из файла F в файл G все слова, состоящие больше чем из трех букв.

**Ход работы:**

- Написание программы и схемы алгоритма в части 1
- Тестировка программы в части 1
- Написание программы и схемы алгоритма в части 2
- Тестировка программы в части 2
- Вывод

Для начала выполним первую часть задания, а именно – создадим список по типу очереди из вещественных чисел и их целых частей:

```cpp
double temp = 0.0;
List list;
size_t n = 0;
std::cout << "Enter amount of your nums: ";
std::cin >> n;

for (size_t i = 0; i < n; ++i) {
    std::cin >> temp;
    list.AddElement(temp);
}
std::cout << "List with odds element: ";
list.PrintList();
std::cout << std::endl;
std::cout << "List without odds element: ";
list.DeleteOdds();
list.PrintList();
```

**Рисунок 1** – Основная программа

```cpp
#ifndef DZ_3_NODE_HPP
#define DZ_3_NODE_HPP

class Node
{
public:
    Node(long long int int_, double fract_);
    ~Node();
    friend class List;

private:
    double fract;
    long long int integer;
    Node* next;
};


#endif//DZ_3_NODE_HPP
```

**Рисунок 2 –** Заголовочный файл Node.hpp

```cpp
#include "Node.hpp"

Node::Node(long long int_, double fract_) : fract(fract_), integer(int_), next(nullptr) {}

Node::~Node() {
    integer = 0;
    fract = 0.0;
    delete next;
}
```

**Рисунок 3 –** Файл реализации Node.cpp

```cpp
#ifndef DZ_3_LIST_HPP
#define DZ_3_LIST_HPP
#include "Node.hpp"

class List
{
public:
    List();
    ~List();
    void AddElement(double);
    void DeleteOdds();
    void PrintList();

private:
    Node* head;
    Node* tail;
};


#endif//DZ_3_LIST_HPP
```

**Рисунок 4 –** Заголовочный файл List.hpp

```cpp
#include "List.hpp"
#include <iostream>

List::List() : head(nullptr), tail(nullptr) {}
List::~List() {
    delete head;
}

void List::AddElement(double num) {
    if (tail == nullptr) {
        tail = new Node( int_: static_cast<long long int>(num), fract_: num - static_cast<long long int>(num));
        head = tail;
    } else {
        tail->next = new Node( int_: static_cast<long long int>(num), fract_: num - static_cast<long long int>(num));
        tail = tail->next;
    }
}


void List::PrintList() {
    Node* currELem = head;

    while (currELem) {
        std::cout << currELem->integer << " + " << currELem->fract << " | ";
        currELem = currELem->next;
    }
    std::cout << std::endl;
}

void List::DeleteOdds() {
    Node* currElem = head;
    Node* prev = currElem;
    size_t count = 0;

    while (currElem) {
        ++count;
        if (count == 1) {
            currElem = currElem->next;
```

**Рисунок 5 –** Файл реализации List.cpp часть 1

```
    while (currElem) {
        ++count;
        if (count == 1) {
            currElem = currElem->next;
            prev->next = nullptr;
            delete prev;
            prev = currElem;
            head = currElem;
        } else if (count & 1) {
            prev->next = currElem->next;
            currElem->next = nullptr;
            delete currElem;
            currElem = prev->next;
        } else {
            prev = currElem;
            currElem = currElem->next;
        }
    }
}
```

**Рисунок 6 –** Файл реализации List.cpp часть 2

Теперь протестируем данную программу на тестовых данных:

```
Enter amount of your nums: 6
1.1 2.2 3.3 4.4 5.5 6.6
List with odds element: 1 + 0.1 | 2 + 0.2 | 3 + 0.3 | 4 + 0.4 | 5 + 0.5 | 6 + 0.6 |

List without odds element: 2 + 0.2 | 4 + 0.4 | 6 + 0.6 |

Process finished with exit code 0
```

**Рисунок 7 –** Тестовые данные часть 1

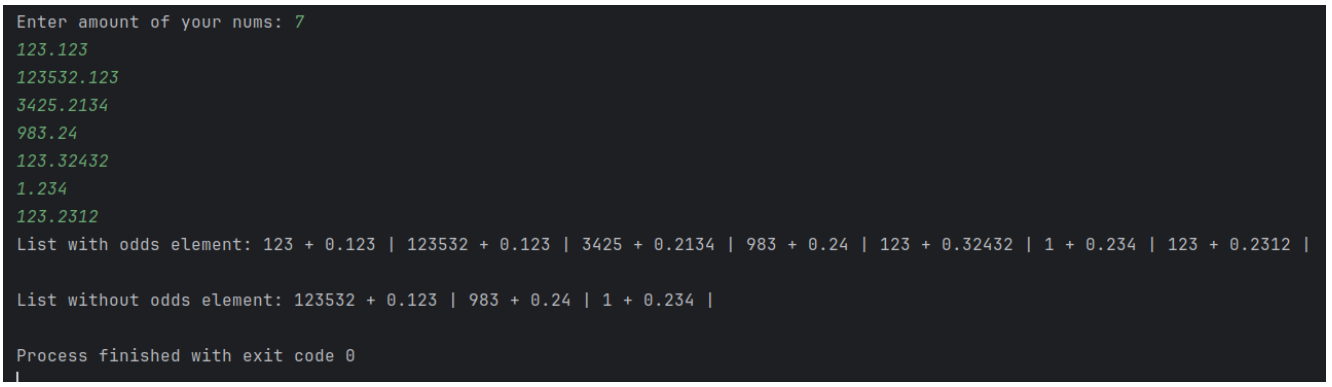


**Рисунок 8 –** Тестовые данные часть 2

Как видно из тестов, программа работает верно. Теперь изобразим программу в виде схемы и диаграммы класса:

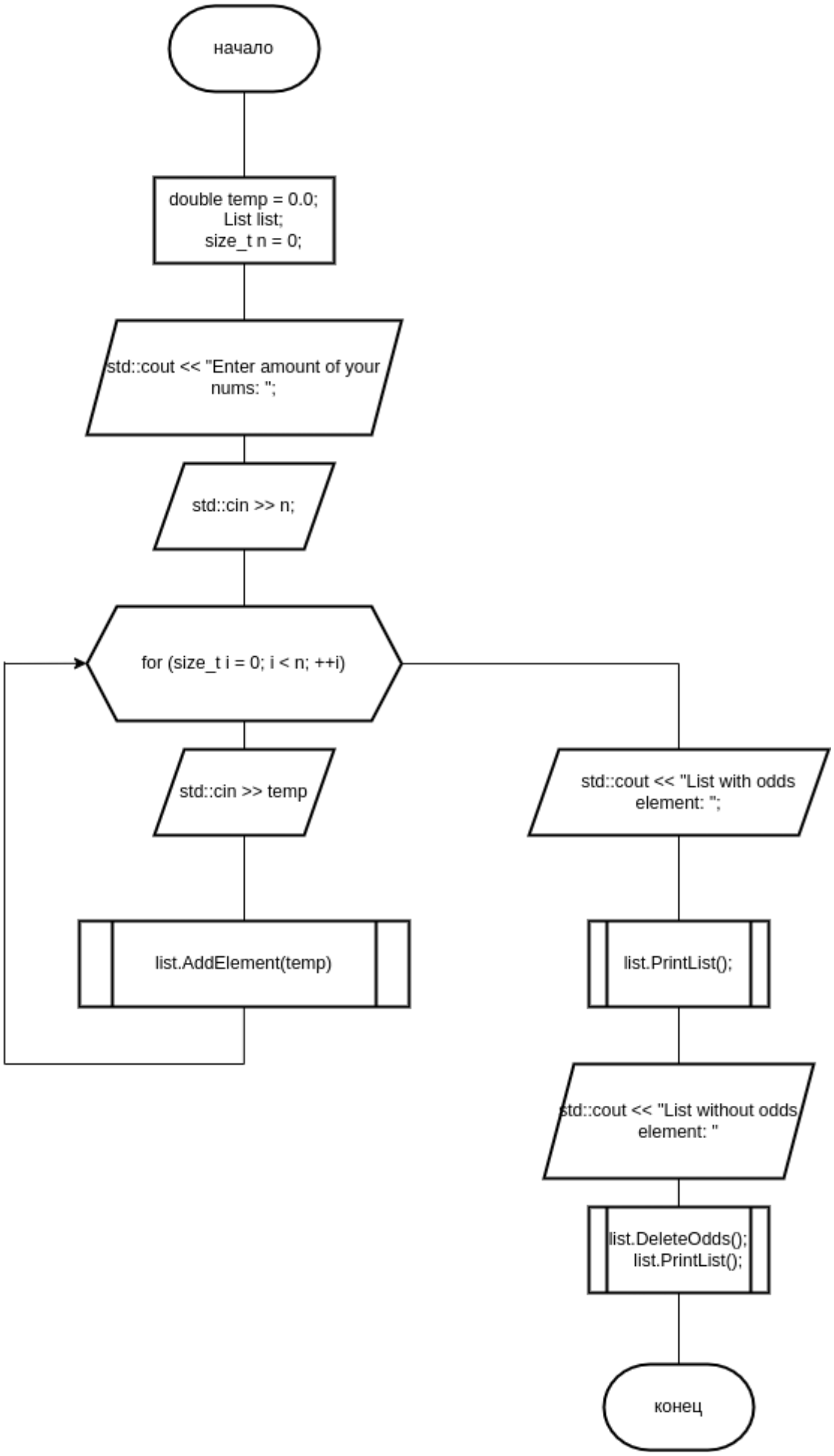


**Рисунок 9 –** Схема алгоритма основной программы

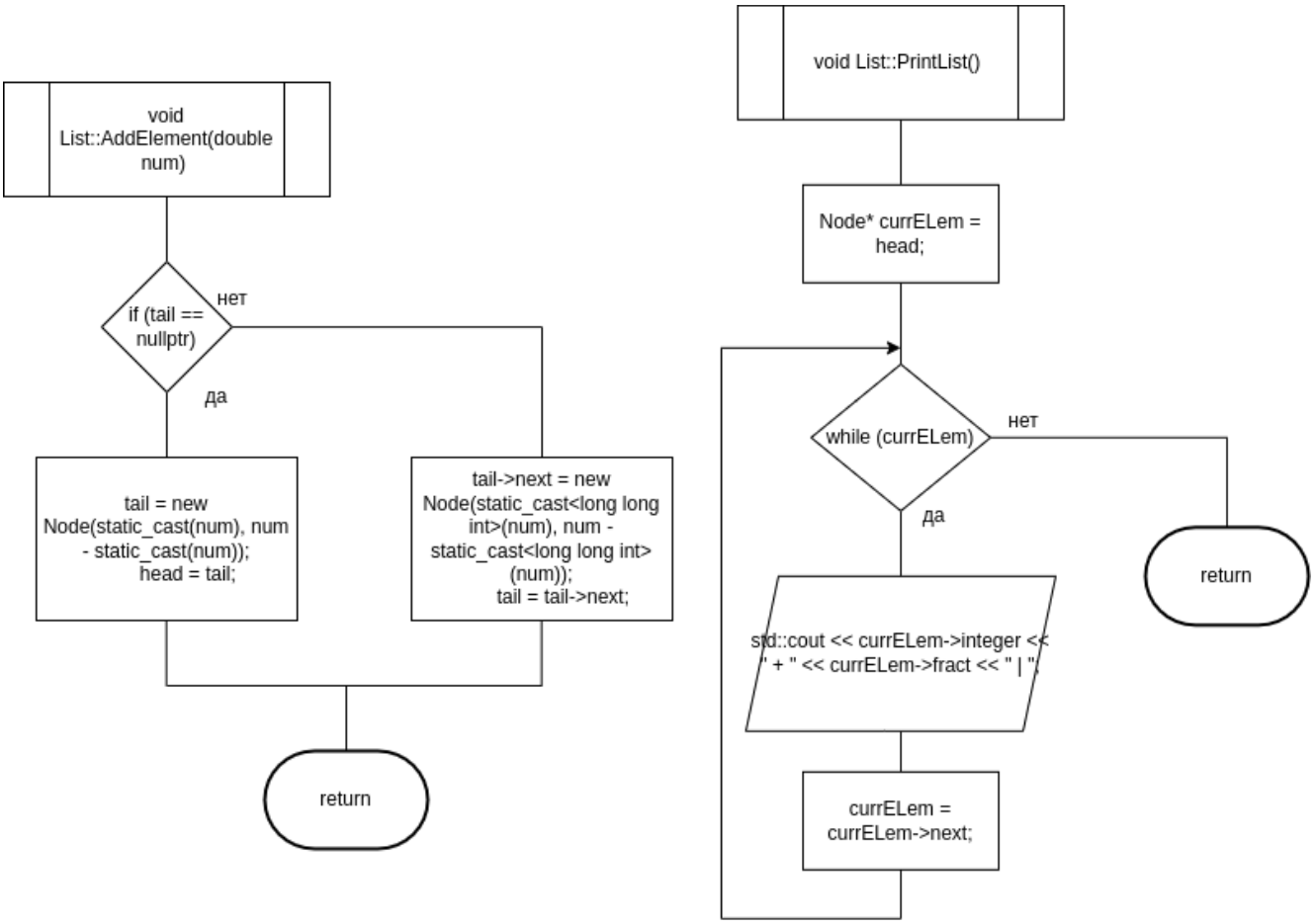


**Рисунок 10 –** Схема алгоритмов процедур, используемых в программе

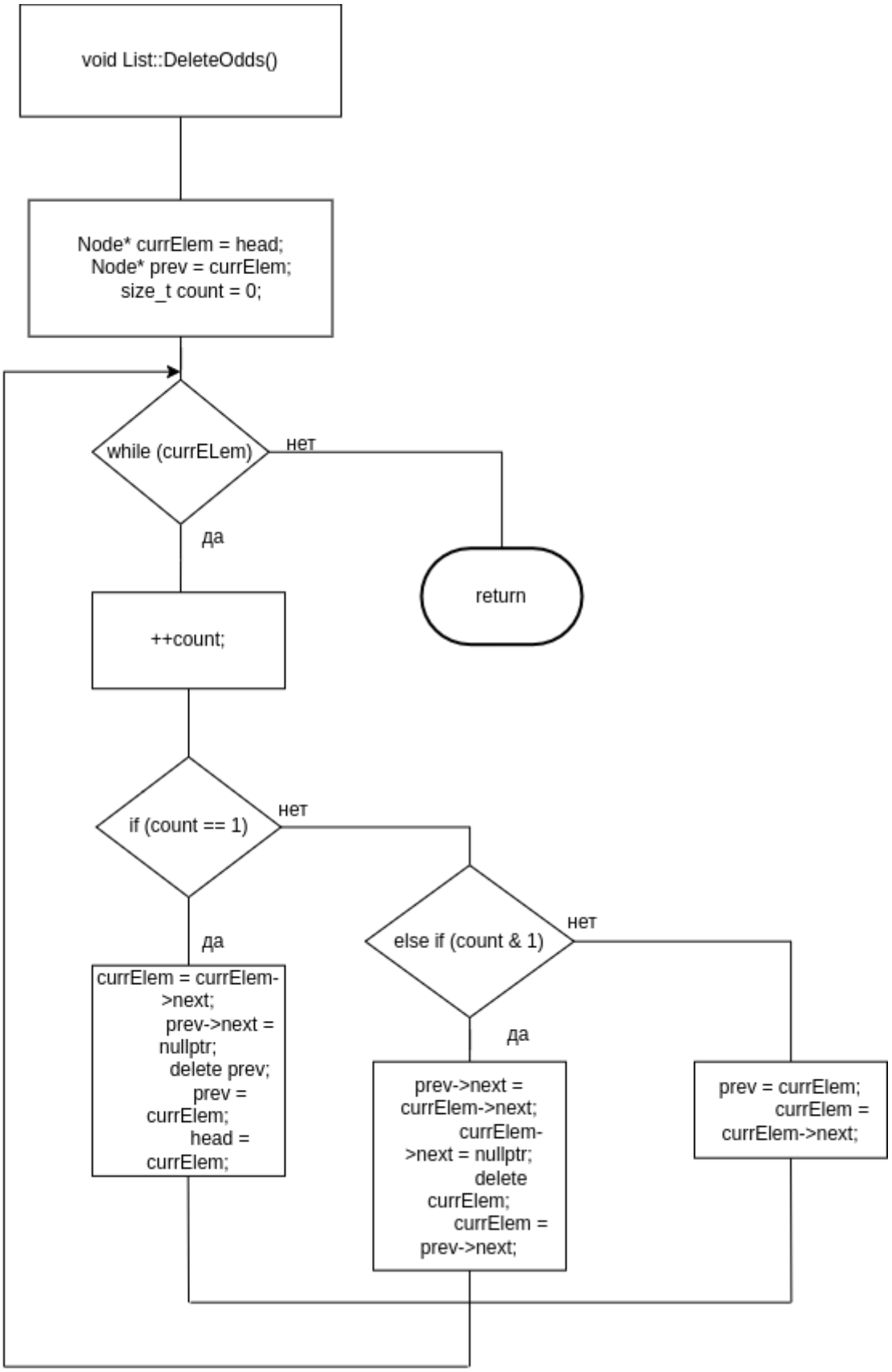


**Рисунок 11 –** Схема алгоритма процедуры, используемой в программе

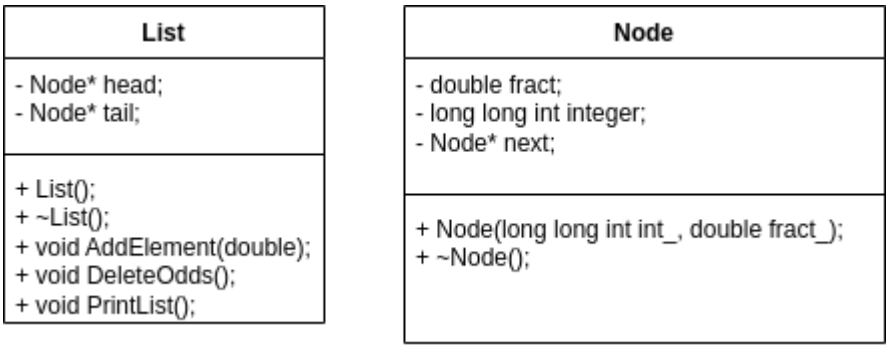


**Рисунок 12 –** Диаграммы классов, используемых в программе

Теперь перейдем ко второй части задания, а именно: напишем код программы:

```
    FILE *toRead;
    FILE *toWrite;
    toRead = fopen( filename: "..//F.txt",  modes: "r");
    toWrite = fopen( filename: "..//G.txt",  modes: "w");
    std::string temp;
    char symb = ' ';

    while (symb = fgetc( stream: toRead), symb != EOF) {
        if (symb == ' ' && !temp.empty()) {
            if (temp.length() > 3) {
                for (size_t i = 0; i < temp.length(); ++i) {
                    fputc( c: temp[i],  stream: toWrite);
                }
                fputc( c: ' ',  stream: toWrite);
            }
            temp.clear();
        } else if (std::isalpha(symb)) {
            temp += symb;
        }
    }

    if (temp.length() > 3) {
        for (size_t i = 0; i < temp.length(); ++i) {
            fputc( c: temp[i],  stream: toWrite);
        }
    }
    fclose( stream: toRead);
    fclose( stream: toWrite);
    return 0;
}
```

**Рисунок 13 –** Код программы

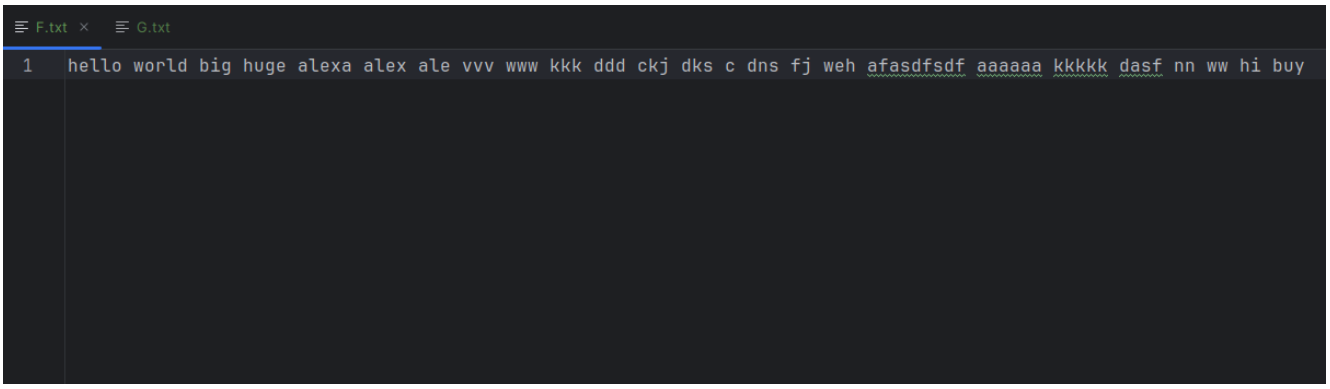


**Рисунок 14 –** Текстовый файл F.txt, содержащий латинские слова

Также создадим пустой файл G.txt. Теперь запустим программу и посмотрим, как выглядит теперь файл G.txt:

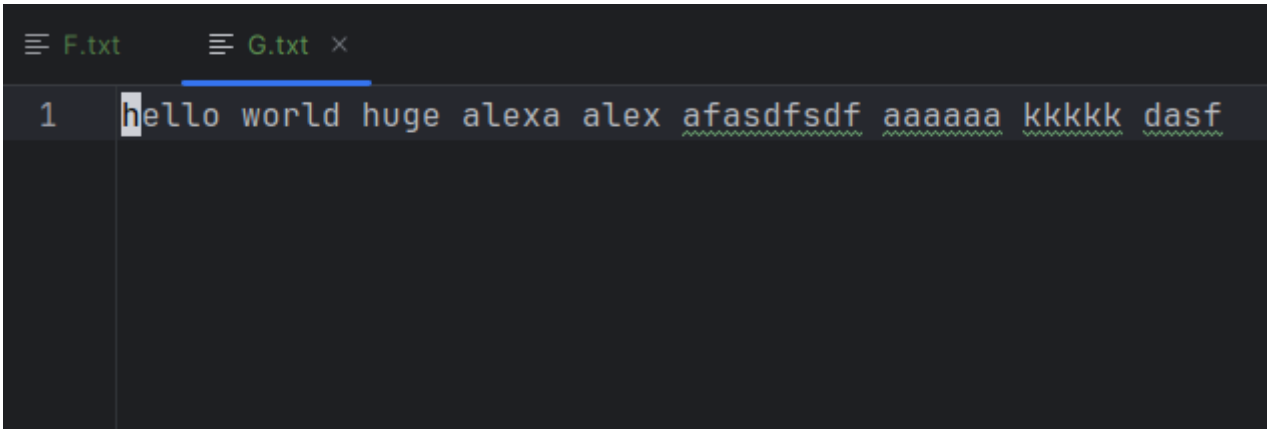


**Рисунок 15 –** Файл G.txt после выполнения программы

Как можем увидеть, в файл G.txt записались только те слова, длина которых больше 3, что и требовалось по заданию. Программа работает корректно.

Теперь представим эту программу в виде схемы:

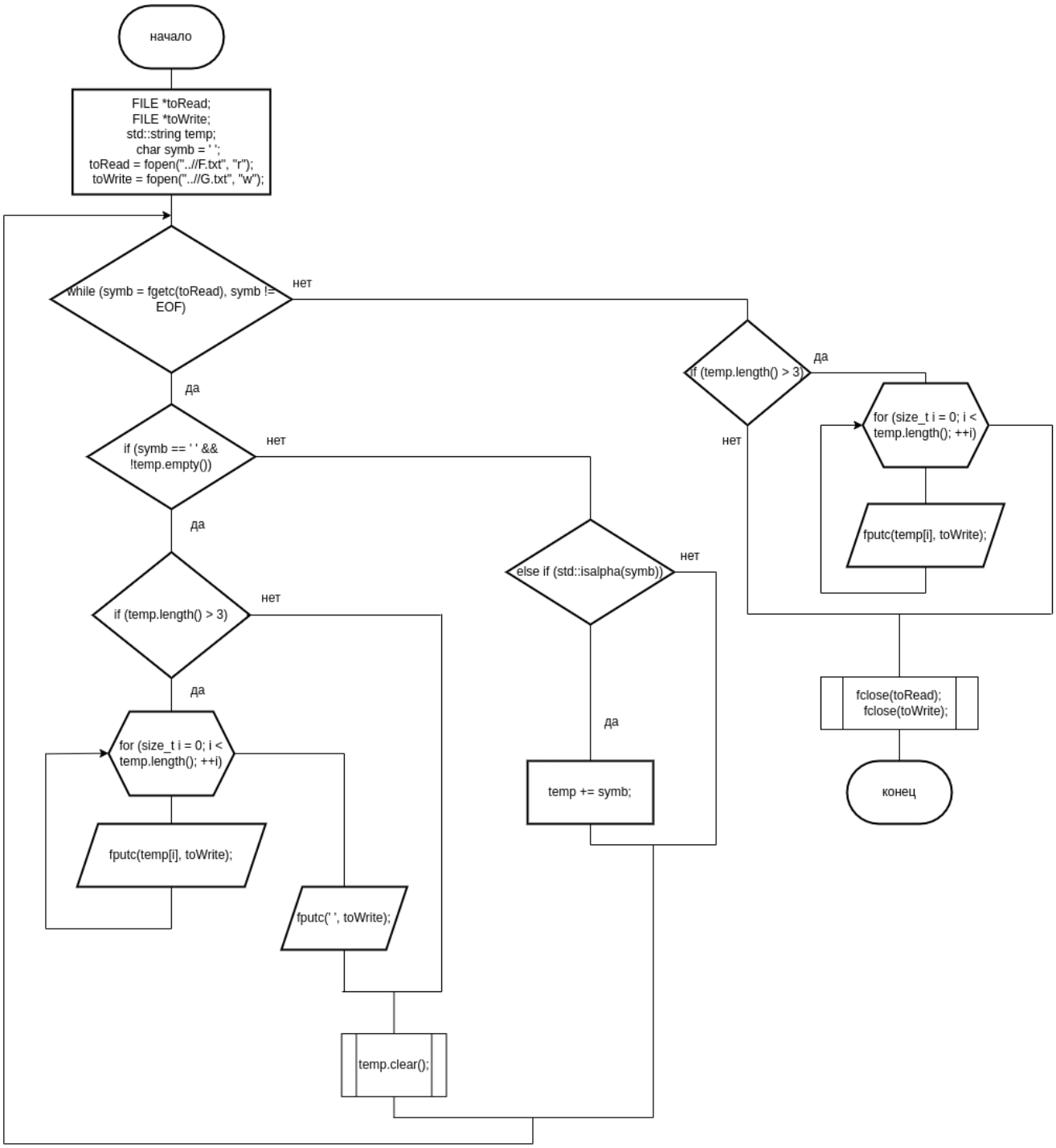


**Рисунок 16 –** Схема алгоритма

**Вывод:** В ходе домашнего задания были получены навыки работы с динамическими структурами данных, а именно: списковыми, а также с файловой системой, с их открытием, обработкой, вводом в них и выводом из них.