**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра ФКТИ**

**ОТЧЕТ**

**по лабораторной работе №2**

**по дисциплине «Программирование»**

| Студент гр. 4335 |  | Тарасов И.А. |
| --- | --- | --- |
| Преподаватель |  | Калмычков B. A. |

Санкт-Петербург

2025

**Исходная формулировка задания**

Вставить заданное число элементов после последнего элемента; Удалить список.

**Структура данных для хранения строки**

Используется собственная структура String для представления строки символов:

* data: динамический массив символов с нуль-терминатором (\0) в конце.
* length: длина строки (количество значимых символов).

**Структура Node**

Узел односвязного списка

* data: Объект типа String, хранящий строку символов.
* next: Указатель (Node\*) на следующий узел в списке.
* Если узел последний, содержит nullptr.

**Структура LinkedList**

Односвязный список

* head: Указатель (Node\*) на первый узел списка.
* Если список пуст, значение равно nullptr.

**Формат входного и выходного файлов**

* Входной файл (input.txt):
  + Содержит исходные данные для списка.
  + Максимальная длина строки не ограничена.
  + Каждая строка представляет собой отдельный элемент списка.
* Входной файл (insert.txt):
  + Содержит элементы для вставки.
  + Формат аналогичен input.txt.
* Выходной файл (output.txt): Содержит результаты обработки
  + Исходный список.
  + Сообщения о состоянии обработки.
  + Результат после вставки элементов.

**Пример входного файла input.txt:**

Hello

World

**Пример входного файла insert.txt:**

New

Elements

Inserted

**Пример выходного файла output.txt:**

Исходный список:

Hello

World

После вставки 3 элементов:

Hello

World

New

Elements

Inserted

**Реализация функций**

| **Имя модуля (файла)** | **Имя структуры/класса или функции** | **Назначение** | **Параметры для функции** | **Возвращаемое значение** |
| --- | --- | --- | --- | --- |
| main.h/main.cpp | init | Инициализирует строку | String\* s | void |
| main.h/main.cpp | free | Освобождает память строки | String\* s | void |
| main.h/main.cpp | copy | Копирует содержимое строки | String String\* dest, const String\* src | void |
| main.h/main.cpp | readFromStdString | Читает строку из std::string | String\* s, const std::string& str | void |
| main.h/main.cpp | initList | Инициализирует список | LinkedList\* list | void |
| main.h/main.cpp | addNode | Добавляет узел в конец списка | LinkedList\* list, const String\* str | void |
| main.h/main.cpp | freeList | Удаляет весь список (задание 3.6) | LinkedList\* list | void |
| main.h/main.cpp | insertAfterLast | Вставляет элементы после последнего (задание 1.10) | LinkedList\* list, int count, String\* newElements | void |
| main.h/main.cpp | printList | Выводит список в файл | const LinkedList\* list, std::ofstream& out | void |

**Алгоритм добавления узла в список**

1. Создать новый узел newNode.
2. Инициализировать newNode->data.
3. Скопировать данные в newNode->data.
4. Установить newNode->next.
5. Если список пуст, то установить newNode в head, иначе найти последний узел и присоединить newNode к последнему узлу.

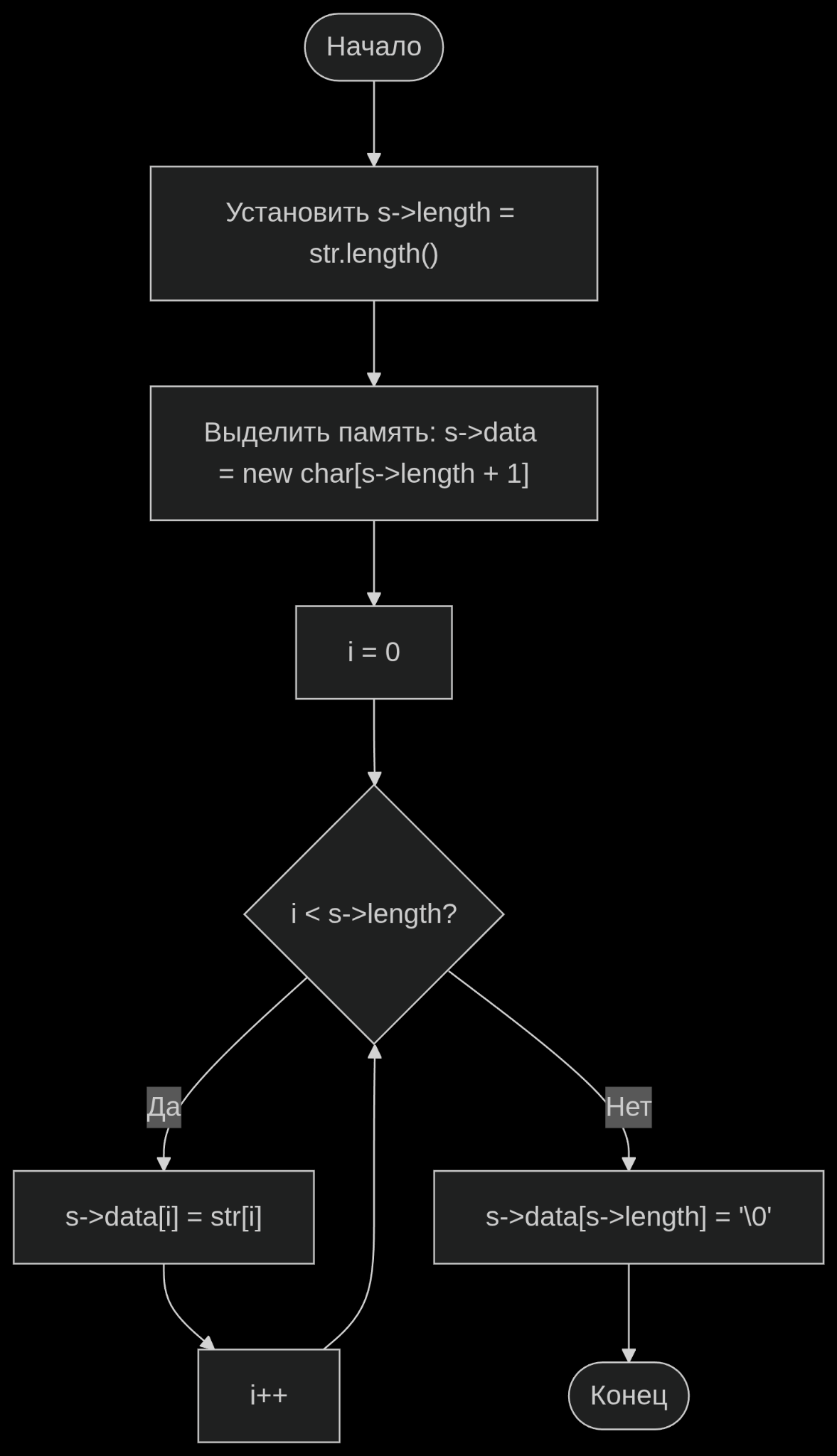
**Алгоритм вставки элемента в конец списка**

1. Для каждого элемента в массиве newElements вызвать addNode для добавления элемента в конец списка.

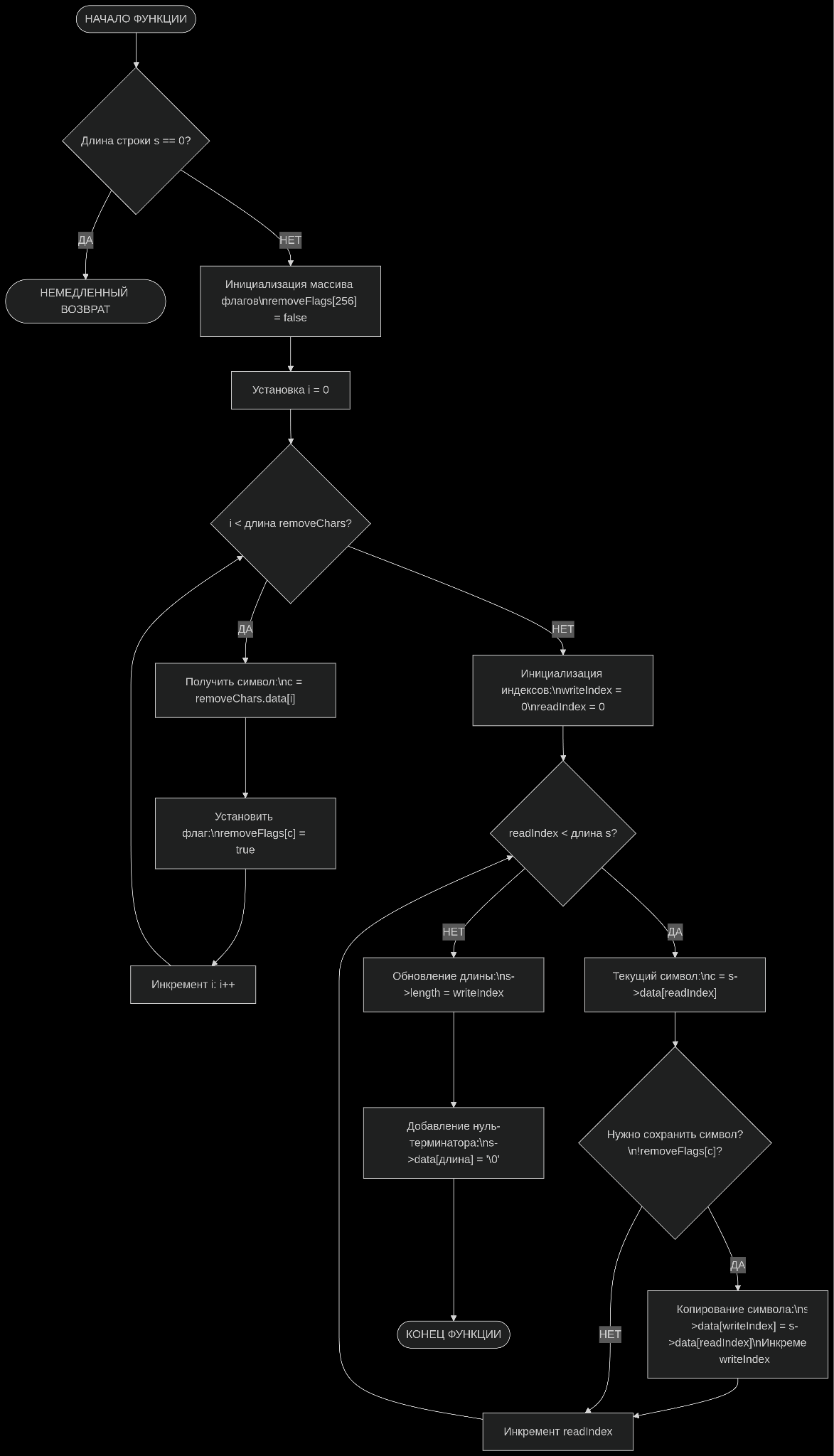
**Алгоритм освобождения списка**

1. Установить current = head.
2. Пока current != nullptr, сохранить next = current->next.
3. Освободить current->data вызвав free.
4. Удалить текущий узел.
5. Установить head = nullptr

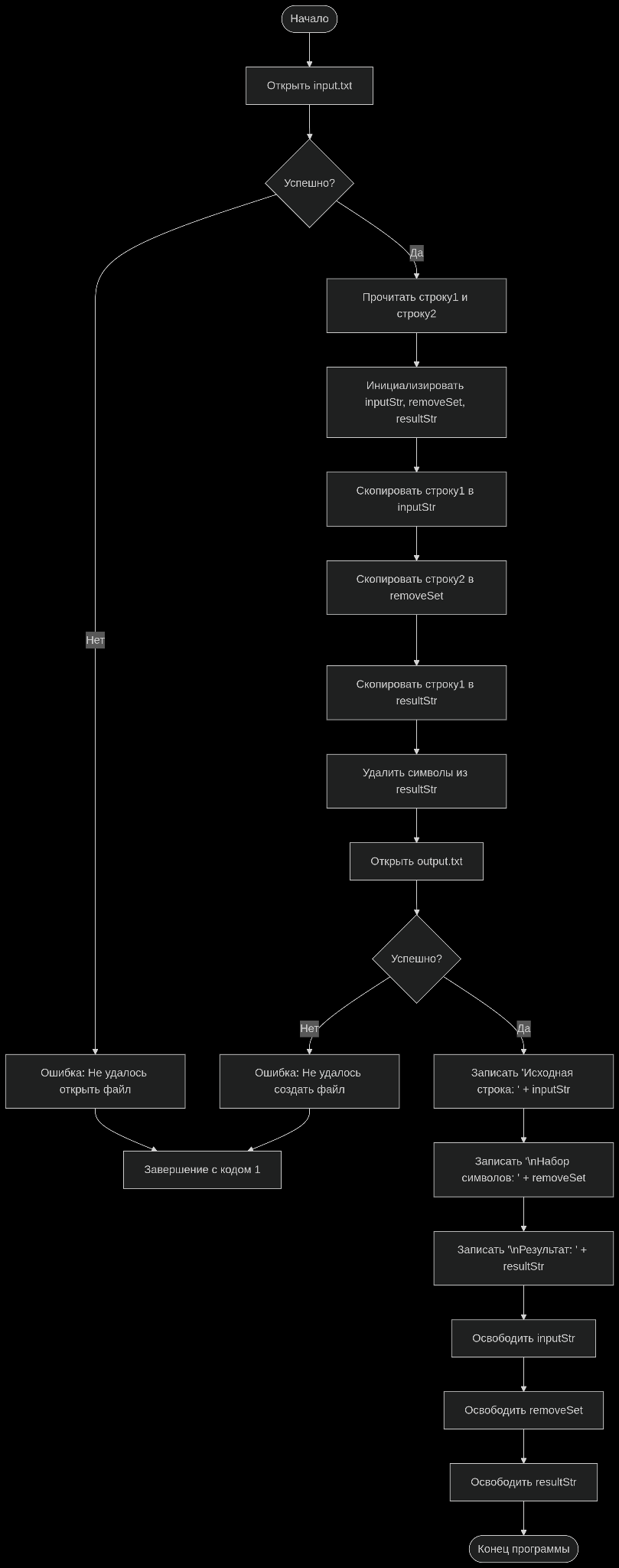
**Блок-схема функции readFromStdString**

****

**Блок-схема функции removeCharacters**

****

**Блок-схема программы**

****

**Примеры работы программы**

**Пример 1: Обычный случай**

// input.txt

Hello

World

// insert.txt

New

Elements

// output.txt

Исходный список:

Hello

World

После вставки 2 элементов:

Hello

World

New

Elements

**Пример 2: Пустой исходный файл**

// input.txt (пустой)

// insert.txt

New

// output.txt

Исходный список:

Исходный список пуст

После вставки 1 элементов:

New

**Пример 3: Файл вставки не найден**

// input.txt

New

// insert.txt отсутствует

// output.txt

Исходный список:

New

Файл для вставки не найден

**Пример 4: Пустой файл вставки**

// input.txt

New

// insert.txt (пустой)

// output.txt

Исходный список:

New

Нет элементов для вставки (файл пуст)

**Пример 5: Несколько вставок**

// input.txt

First

Second

// insert.txt

Third

Fourth

Fifth

// output.txt

Исходный список:

First

Second

После вставки 3 элементов:

First

Second

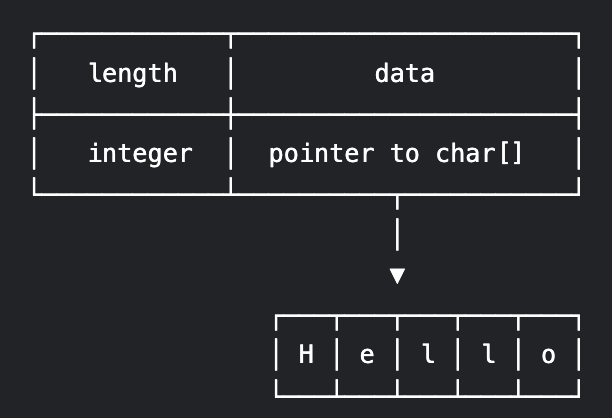
Third

Fourth

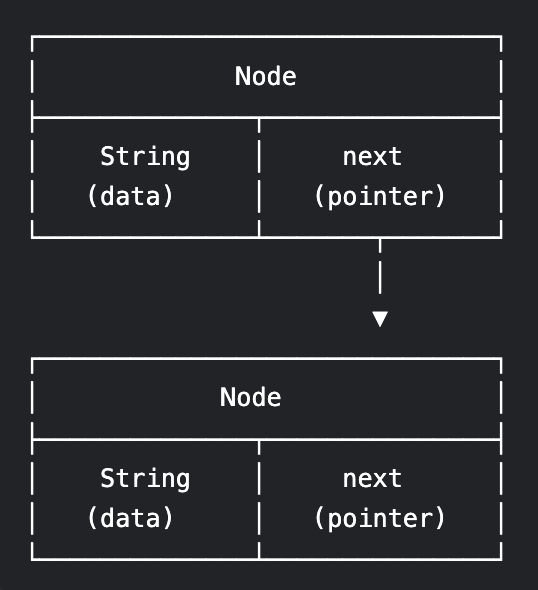
Fifth

**Визуализация внутреннего представления**

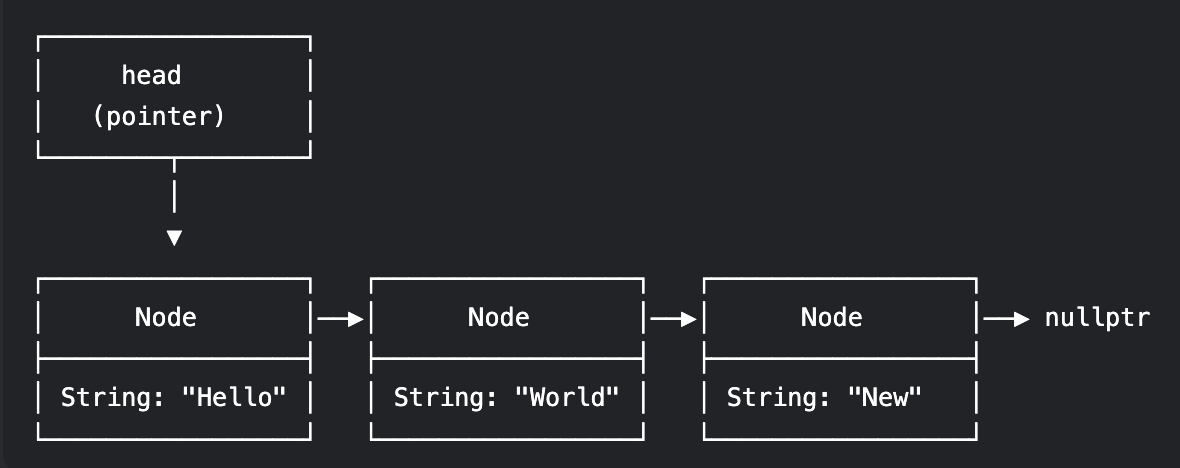
Структура String в памяти:



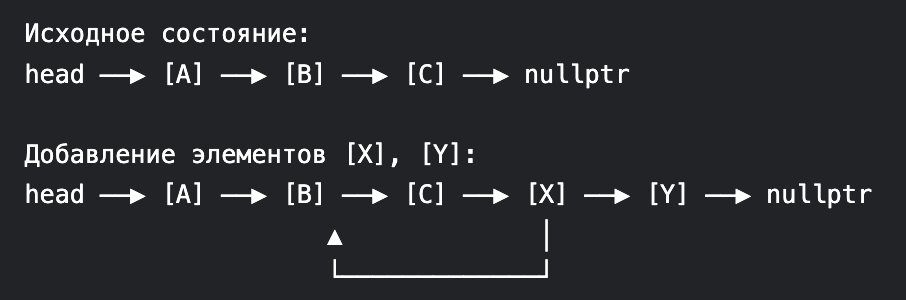
Структура Node:



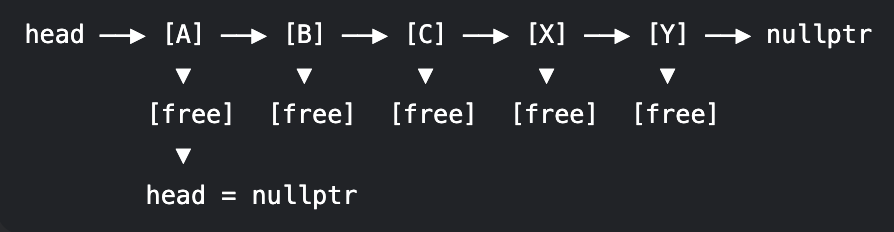
Структура LinkedList:



Вставка элементов после последнего элемента:



Удаление списка:



**Текст программы**

#include <iostream>

#include <fstream>

#include <string>

struct String {

char\* data;

int length;

};

void init(String\* s) {

s->data = nullptr;

s->length = 0;

}

void free(String\* s) {

if (s->data) {

delete[] s->data;

}

s->data = nullptr;

s->length = 0;

}

void copyString(String\* dest, const String\* src) {

dest->length = src->length;

if (dest->length > 0) {

dest->data = new char[dest->length + 1];

for (int i = 0; i < dest->length; ++i) {

dest->data[i] = src->data[i];

}

dest->data[dest->length] = '\0';

} else {

dest->data = nullptr;

}

}

void readFromStdString(String\* s, const std::string& str) {

s->length = str.length();

if (s->length > 0) {

s->data = new char[s->length + 1];

for (int i = 0; i < s->length; ++i) {

s->data[i] = str[i];

}

s->data[s->length] = '\0';

} else {

s->data = nullptr;

}

}

struct Node {

String data;

Node\* next;

};

struct LinkedList {

Node\* head;

};

void initList(LinkedList\* list) {

list->head = nullptr;

}

void addNode(LinkedList\* list, const String\* str) {

Node\* newNode = new Node;

init(&newNode->data);

copyString(&newNode->data, str);

newNode->next = nullptr;

if (list->head == nullptr) {

list->head = newNode;

} else {

Node\* current = list->head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

}

}

void freeList(LinkedList\* list) {

Node\* current = list->head;

while (current != nullptr) {

Node\* next = current->next;

free(&current->data);

delete current;

current = next;

}

list->head = nullptr;

}

void insertAfterLast(LinkedList\* list, int count, String\* newElements) {

for (int i = 0; i < count; ++i) {

addNode(list, &newElements[i]);

}

}

void printList(const LinkedList\* list, std::ofstream& out) {

Node\* current = list->head;

while (current != nullptr) {

if (current->data.data) {

out << current->data.data << "\n";

}

current = current->next;

}

}

int main() {

LinkedList list;

initList(&list);

std::ofstream out("output.txt");

if (!out) {

std::cerr << "Cannot open output file: " << "output.txt" << std::endl;

return 1;

}

std::ifstream in("input.txt");

if (in) {

out << "Исходный список:\n";

std::string line;

int count = 0;

while (std::getline(in, line)) {

count++;

}

if (count > 0) {

in.close();

in.open("input.txt");

while (std::getline(in, line)) {

String str;

init(&str);

readFromStdString(&str, line);

addNode(&list, &str);

free(&str);

}

in.close();

printList(&list, out);

} else {

out << "Исходный список пуст\n";

}

} else {

out << "Исходный список не найден\n";

}

std::ifstream insert("insert.txt");

if (insert) {

std::string line;

int count = 0;

while (std::getline(insert, line)) {

count++;

}

if (count > 0) {

insert.close();

insert.open("insert.txt");

String\* newElements = new String[count];

for (int i = 0; i < count; ++i) {

std::getline(insert, line);

init(&newElements[i]);

readFromStdString(&newElements[i], line);

}

insertAfterLast(&list, count, newElements);

out << "\nПосле вставки " << count << " элементов:\n";

printList(&list, out);

for (int i = 0; i < count; ++i) {

free(&newElements[i]);

}

delete[] newElements;

} else {

out << "\nНет элементов для вставки (файл пуст)\n";

}

insert.close();

} else {

out << "\nФайл для вставки не найден\n";

}

freeList(&list);

out.close();

return 0;

}

**Вывод:**

Реализованная программа успешно демонстрирует работу с односвязным списком строк на базе пользовательской структуры String, включая динамическое управление памятью, чтение данных из файлов input.txt и insert.txt с обработкой ошибок, вставку элементов в конец списка, и вывод результатов в output.txt, полностью соответствуя требованиям лабораторной работы по реализации динамических структур данных.