МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Институт математики и информационных систем
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Отчёт по лабораторной работе №4

по дисциплине

«Программирование»

Выполнил студент гр. ИВТб-1301-06-00 _____/Бигрин А.С./
Проверил доцент кафедры ЭВМ _____/Должженкова М.Л./

Киров
2025

Цель

Цель работы: Изучение механизмов эффективного управления динамической памятью, создание элементарных структур данных.

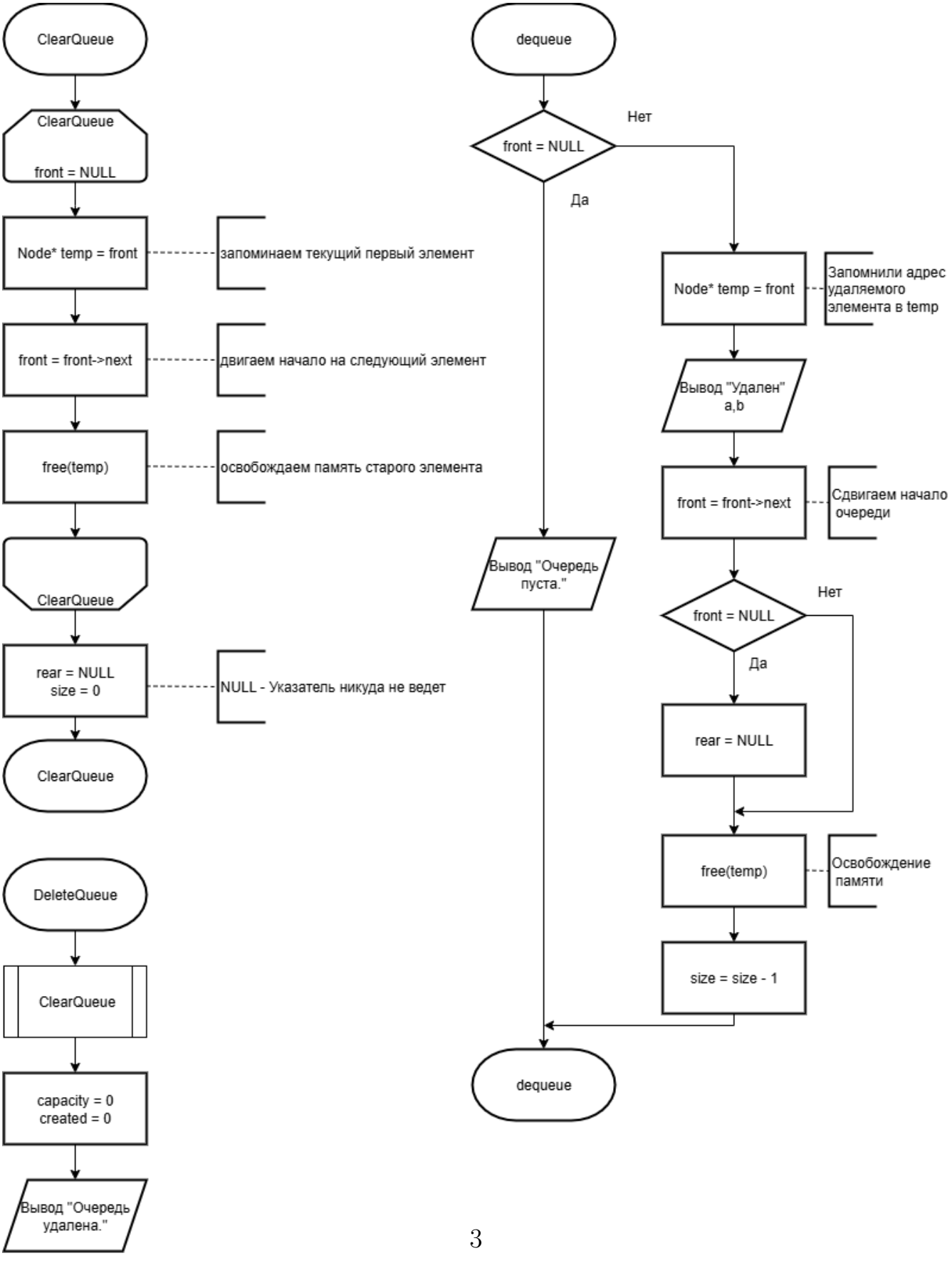
Требования

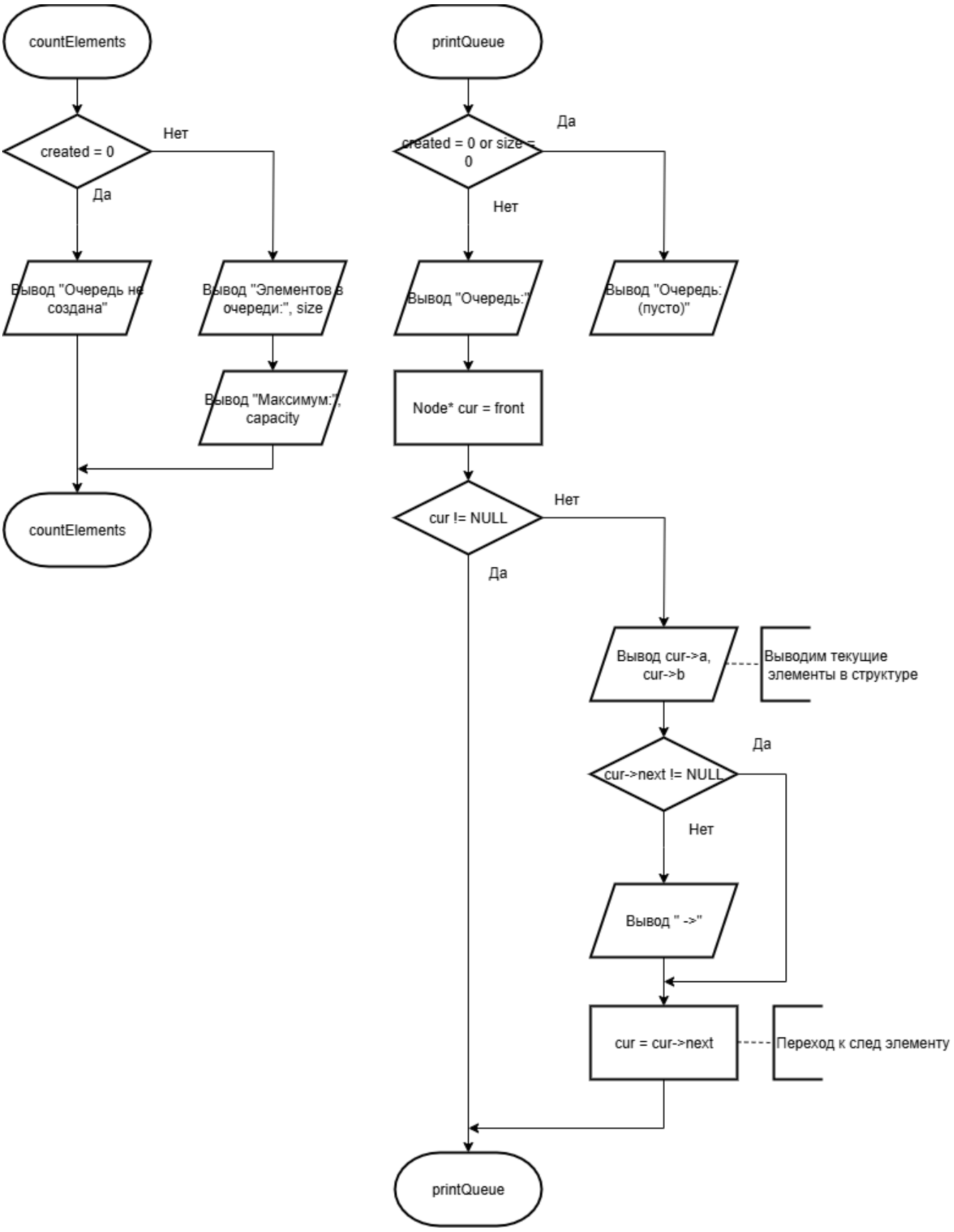
1. Интерфейс взаимодействия с пользователем организован как меню. Навигация в меню реализуется через клавиши (в виде стрелок)
2. Применяются ТОЛЬКО структуры данных динамической памяти
3. Работа пунктов меню в коде реализована через применение подпрограмм;
4. Предусмотреть отдельные пункты меню для добавления и удаления элементов динамической структуры

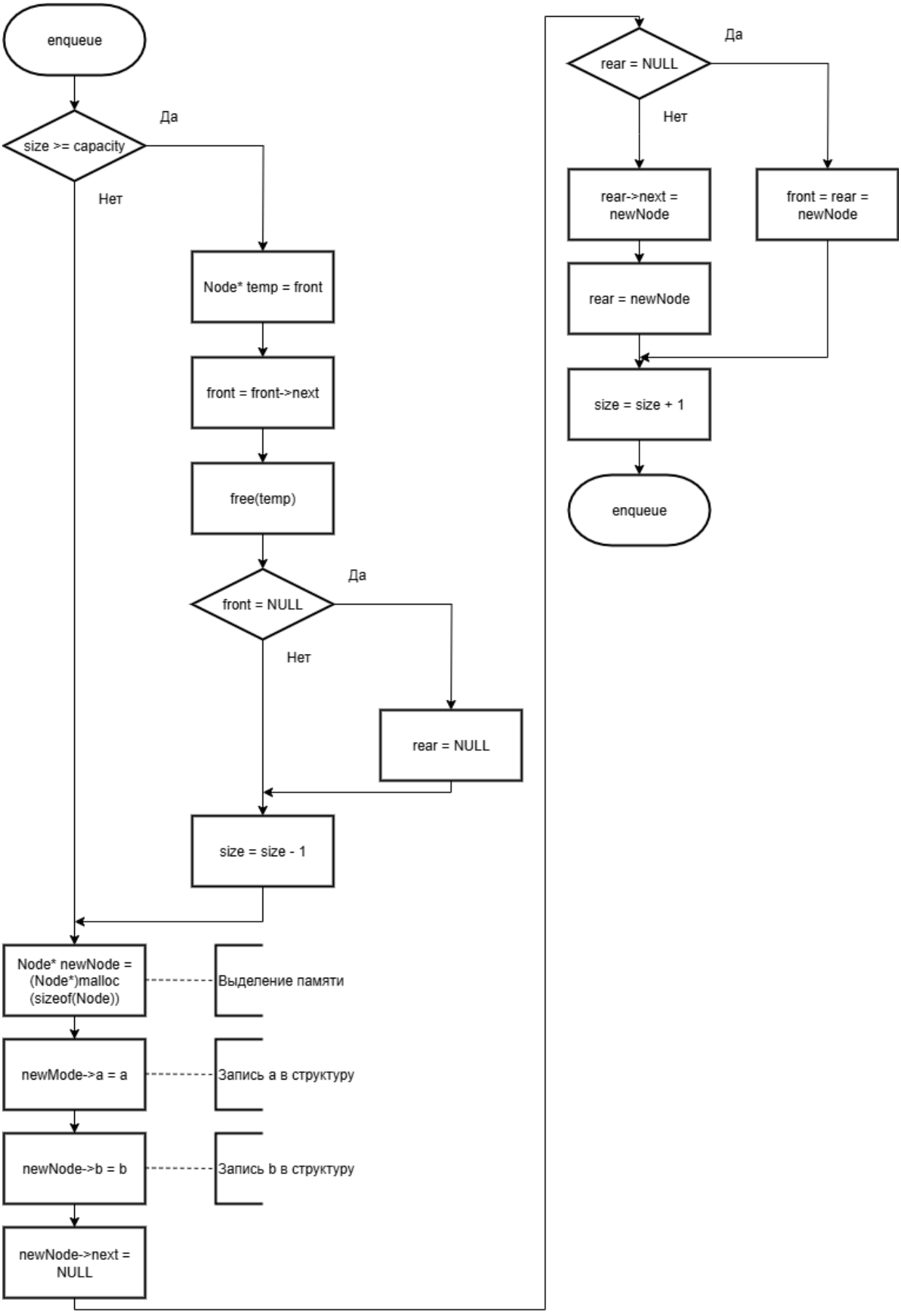
Задание

Разработать консольное приложение для демонстрации особенностей организации элементарных структур данных. Для организации взаимодействия с пользователем использовать case-меню.

Решение


ClearQueue
ClearQueue
front = NULL
Node* temp = front
запоминаем текущий первый элемент
front = front->next
двигаем начало на следующий элемент
free(temp)
освобождаем память старого элемента
ClearQueue
rear = NULL
size = 0
NULL - Указатель никуда не ведет
ClearQueue
DeleteQueue
ClearQueue
capacity = 0
created = 0
Вывод "Очередь удалена."
dequeue
front = NULL
Нет
Да
Вывод "Очередь пуста."
Node* temp = front
Запомнили адрес удаляемого элемента в temp
Вывод "Удален" a,b
front = front->next
Сдвигаем начало очереди
front = NULL
Нет
Да
rear = NULL
free(temp)
Освобождение памяти
size = size - 1
dequeue


countElements
created = 0
Нет
Да
Вывод "Очередь не создана"
Вывод "Элементов в очереди:", size
Вывод "Максимум:", capacity
countElements
printQueue
created = 0 or size = 0
Да
Нет
Вывод "Очередь:"
Вывод "Очередь: (пусто)"
Node* cur = front
cur != NULL
Нет
Да
Вывод cur->a, cur->b
Выводим текущие элементы в структуре
cur->next != NULL
Да
Нет
Вывод " ->"
cur = cur->next
Переход к след элементу
printQueue


enqueue
size >= capacity
Да
Нет
Node* temp = front
front = front->next
free(temp)
front = NULL
Да
Нет
rear = NULL
size = size - 1
Node* newNode = (Node*)malloc (sizeof(Node))
Выделение памяти
newMode->a = a
Запись a в структуру
newNode->b = b
Запись b в структуру
newNode->next = NULL
rear = NULL
Да
Нет
rear->next = newNode
front = rear = newNode
rear = newNode
size = size + 1
enqueue

```
#include <stdio.h>
#include <stdlib.h>  // для malloc, free
#include <string.h>  // для strcpy
#include <conio.h>   // для getch()

// Структура элемента в очереди
typedef struct Node {
    int a;
    char b[21];
    struct Node* next; // указатель на следующий элемент в очереди
} Node;

// Глобальные переменные
Node* front = NULL;   // указатель на первый элемент
Node* rear = NULL;    // указатель на последний
int size = 0;         // текущее количество элементов в очереди
int capacity = 0;     // максимальное количество элементов
int created = 0;      // флаг создана ли очередь


void clearQueue() {
    // Пока есть элементы в начале очереди.
    while (front != NULL) {
        Node* temp = front;      // запоминаем текущий первый элемент
        front = front->next;     // двигаем начало на следующий элемент
        free(temp);              // освобождаем память старого элемента
    }
    rear = NULL;  // если очередь пуста, то и конец должен быть NULL
    size = 0;
}
```

```
void deleteQueue() {
    clearQueue();
    capacity = 0;
    created = 0;
}

// Добавить элемент в очередь
void enqueue(int a, char* b) {
    if (size >= capacity) {
        Node* temp = front;        // запоминаем первый элемент
        front = front->next;       // начинаем очередь со второго
        free(temp);                // освобождаем память
        if (front == NULL)         // если очередь стала пустой
            rear = NULL;           // то и конец = NULL
        size--;
    }

    // Выделяем память под элемент
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->a = a;                // записываем число
    strcpy(newNode->b, b);         // копируем строку (откуда, куда)
    newNode->next = NULL;

    // Добавляем элемент в конец очереди
    if (rear == NULL) {
    // Если очередь была пуста - новый элемент и первый, и последний
        front = rear = newNode;
    } else {
```

```
        // Иначе присоединяем к текущему концу
        rear->next = newNode;
        rear = newNode;
    }
    size++;
}

// Удалить элемент
void dequeue() {
    if (front == NULL) {
        printf("Очередь пуста!\n");
        return;
    }

    Node* temp = front;
    printf("Удалён: %d %s\n", temp->a, temp->b);

    front = front->next;          // сдвигаем начало очереди
    if (front == NULL)            // если всё удалили
        rear = NULL;              // конец тоже обнуляем
    free(temp);                   // освобождаем память
    size--;
}

// Кол-во элементов в очереди
void countElements() {
    if (!created) {
        printf("Очередь не создана!\n");
    } else {
        printf("Элементов в очереди: %d\n", size);
```

```
        if (capacity > 0) {
            printf("(максимум: %d)\n", capacity);
        }
    }
}

// Вывод очереи
void printQueue() {
    if (!created || size == 0) {
        printf("Очередь: (пусто)\n\n");
        return;
    }

    printf("Очередь:");
    Node* cur = front; // начинаем с первого элемента
    while (cur != NULL) {
        printf(" %d %s", cur->a, cur->b);
        if (cur->next != NULL) {
            printf(" →");
        }
        cur = cur->next;
    }
    printf("\n\n");
}

// Основная функция
int main() {
    system("chcp 65001 > nul");

    int choice = 0; // текущий выбор в меню
```

```
while (1) {
    system("cls");
    printQueue();


    const char* menu[] = {
        "1. Создать очередь",
        "2. Очистить очередь",
        "3. Запись элемента в очередь",
        "4. Удалить элемент из очереди",
        "5. Удалить очередь",
        "6. Количество элементов",
        "7. Выход"
    };

    printf("=== МЕНЮ ===\n");
    for (int i = 0; i < 7; i++) {
        if (i == choice)
            printf(">>> %s\n", menu[i]);
        else
            printf("    %s\n", menu[i]);
    }
    printf("\nСтрелки - перемещение, Enter - выбрать\n");

    char key = getch();

    // Если нажата стрелка
    if (key == 0 || key == -32) {
        key = getch();
```

```
        if (key == 72) { // стрелка вверх
            if (choice == 0) {
                choice = 6;
            } else {
                choice--;
            }
        }
        if (key == 80) { // стрелка вниз
            if (choice == 6) {
                choice = 0;
            } else {
                choice++;
            }
        }
    }
    else if (key == 13) {
        if (choice == 0) {
            if (created) deleteQueue();
            printf("Ёмкость: ");
            scanf("%d", &capacity);
            if (capacity <= 0) {
                printf("Неверная ёмкость!\n");
            } else {
                created = 1;
                printf("Очередь создана (ёмкость: %d)\n", capacity);
            }
            getch();
        }
        else if (choice == 1) {
            clearQueue();
```

```
        printf("Очищено.\n");
        getch();
    }
    else if (choice == 2) {
        if (!created) {
            printf("Сначала создайте очередь!\n");
            getch();
            continue; // вернуться в начало цикла
        }
        int a;
        char b[21];
        printf("Число: ");
        scanf("%d", &a);
        printf("Строка: ");
        scanf("%20s", b); //
        enqueue(a, b); // добавить в очередь
        printf("Добавлено.\n");
        getch();
    }
    else if (choice == 3) {
        dequeue();
        getch();
    }
    else if (choice == 4) {
        if (!created) {
            printf("Очередь не создана!\n");
        } else {
            deleteQueue();
        }
        getch();
```

```
            }
            else if (choice == 5) {
                countElements();
                getch();
            }
            else if (choice == 6) {
                break;
            }
        }
    }

    return 0;
}
```

Вывод

В ходе выполнения лабораторной работы я изучил и закрепил навыки работы с динамической памятью, указателями, а также создал программу case-menu о структуре данных - очередь, работу с ней.