**Отчет по лабораторной работе № 26**

**по практикуму по циклу дисциплин "Информатика"**

Выполнил студент группы М8О-114БВ-24: Дробышев Егор Павлович, № по списку 9

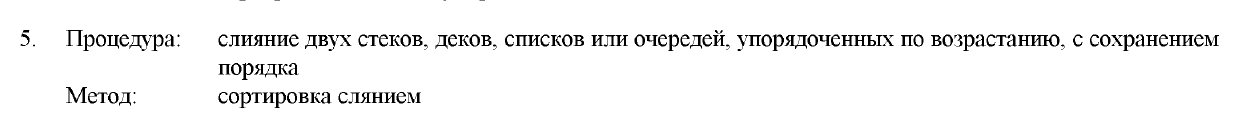
Контакты e-mail: [tru.899@yandex.ru](mailto:tru.899@yandex.ru)

Работа выполнена: «6» апреля 2025 г.

Преподаватель: доцент каф. 806 Никулин Сергей Петрович

Отчет сдан « » 2025г., итоговая оценка

Подпись преподавателя

1. **Тема:** «Абстрактные типы данных. Рекурсия. Модульное программирование на Си. Автоматизация сборки программ модульной структуры на языке Си с использованием утилиты make»
2. **Цель работы:** Составить и отладить модуль определений и модуль реализации по заданной схеме модуля определений для дека. Составить программный модуль, сортирующий дек методом слияния, используя только операции, импортированные из модуля дека.
3. **Задание**: 3. Дек

# Оборудование: оборудование ПЭВМ студента, если использовалось:

Процессор \_Intel Core i5\_ с ОП 8 Гб НМД 256 Гб. Монитор 1920x1080~60Hz. Другие устройства не использовались

# Программное обеспечение: программное обеспечение ЭВМ студента:

Операционная система семейства \_Linux\_, наименование \_Ubuntu\_ версия \_24.04

интерпретатор команд \_GNU bash\_ версия \_5.2.21(1)\_.

Редактор текстов emacs версия 29.3

Утилиты операционной системы: gcc, gdb

Местонахождение и имена файлов программ и данных на домашнем компьютере:

/home/tru\_\_

1. **Идея, метод, алгоритм решения задачи** (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями).

**Описание функций дека:**

1. **initDeck(StaticDeck \*D)** — инициализация дека: front устанавливается в 0, rear — в -1, size — в 0.
2. **size(StaticDeck \*D)** — возвращает текущее количество элементов в деке.
3. **empty(StaticDeck \*D)** — проверяет, пуст ли дек.
4. **full(StaticDeck \*D)** — проверяет, полон ли дек.
5. **pushFront(StaticDeck \*D, int key, int value)** — добавление элемента в начало.
6. **pushBack(StaticDeck \*D, int key, int value)** — добавление элемента в конец.
7. **popFront(StaticDeck \*D)** — удаляет и возвращает значение первого элемента: сохраняет значение с позиции front и сдвигает его циклически вперед. Уменьшает size.
8. **popBack(StaticDeck \*D)** — удаляет и возвращает значение последнего элемента: сохраняет значение с позиции rear и уменьшает rear циклически назад. Уменьшает size.
9. **topFront(StaticDeck \*D)** — возвращает значение первого элемента без удаления.
10. **topBack(StaticDeck \*D)** — возвращает значение последнего элемента без удаления.
11. **frontKey(StaticDeck \*D)** — возвращает ключ первого элемента без удаления.
12. **clearDeck(StaticDeck \*D)** — очищает дек: сбрасывает front, rear, size в начальные значения.
13. **printDeck(StaticDeck \*D) —** выводит все элементы дека в виде: [(key:value)], начиная с front и двигаясь циклически.

**Сортировка слиянием:**

**mergeSort(StaticDeck \*D) —** дек рекурсивно разбивается пополам и копирует первую половину в left, а вторую в right. Далее также рекурсивно объединяем в дек, пока не опустеют: сравниваются первые элементы left и right и вставляется меньший в дек D. Если элементы из left, right остаются, то просто добавляем их.

1. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].

**deck.h**

#ifndef DECK\_H

#define DECK\_H

#include <stdio.h>

#include <stdbool.h>

#define MAX\_SIZE 10

typedef int key\_type;

typedef int value\_type;

typedef struct {

    key\_type key;

    value\_type value;

} KeyValuePair;

typedef struct {

    KeyValuePair data[MAX\_SIZE];

    int front;

    int rear;

    int size;

} StaticDeck;

void initDeck(StaticDeck \**D*);

int size(StaticDeck \**D*);

bool empty(StaticDeck \**D*);

bool full(StaticDeck \**D*);

void pushFront(StaticDeck \**D*, key\_type *key*, value\_type *value*);

void pushBack(StaticDeck \**D*, key\_type *key*, value\_type *value*);

value\_type popFront(StaticDeck \**D*);

value\_type popBack(StaticDeck \**D*);

value\_type topFront(StaticDeck \**D*);

value\_type topBack(StaticDeck \**D*);

void clearDeck(StaticDeck \**D*);

void printDeck(StaticDeck \**D*);

key\_type frontKey(StaticDeck \**D*);

#endif

**deck.c**

#include "deck.h"

void initDeck(StaticDeck \**D*) {

*D*->front = 0;

*D*->rear = -1;

*D*->size = 0;

}

int size(StaticDeck \**D*) {

    return *D*->size;

}

bool empty(StaticDeck \**D*) {

    return size(*D*) == 0;

}

bool full(StaticDeck \**D*) {

    return *D*->size >= MAX\_SIZE;

}

void pushFront(StaticDeck \**D*, key\_type *key*, value\_type *value*) {

    if (full(*D*)) {

        printf("Ошибка: дек переполнен\n");

        return;

    }

*D*->front = (*D*->front - 1 + MAX\_SIZE) % MAX\_SIZE;

*D*->data[*D*->front].key = *key*;

*D*->data[*D*->front].value = *value*;

    if (*D*->rear == -1) *D*->rear = *D*->front;

*D*->size++;

}

void pushBack(StaticDeck \**D*, key\_type *key*, value\_type *value*) {

    if (full(*D*)) {

        printf("Ошибка: дек переполнен\n");

        return;

    }

*D*->rear = (*D*->rear + 1) % MAX\_SIZE;

*D*->data[*D*->rear].key = *key*;

*D*->data[*D*->rear].value = *value*;

    if (*D*->size == 0) *D*->front = *D*->rear;

*D*->size++;

}

value\_type popFront(StaticDeck \**D*) {

    if (empty(*D*)) {

        printf("Дек пуст!\n");

        return -1;

    }

    value\_type val = *D*->data[*D*->front].value;

*D*->front = (*D*->front + 1) % MAX\_SIZE;

*D*->size--;

    if (*D*->size == 0) {

*D*->front = 0;

*D*->rear = -1;

    }

    return val;

}

value\_type popBack(StaticDeck \**D*) {

    if (empty(*D*)) {

        printf("Дек пуст!\n");

        return -1;

    }

    value\_type val = *D*->data[*D*->rear].value;

*D*->rear = (*D*->rear - 1 + MAX\_SIZE) % MAX\_SIZE;

*D*->size--;

    if (*D*->size == 0) {

*D*->front = 0;

*D*->rear = -1;

    }

    return val;

}

value\_type topFront(StaticDeck \**D*) {

    if (empty(*D*)) {

        printf("Дек пуст!\n");

        return -1;

    }

    return *D*->data[*D*->front].value;

}

value\_type topBack(StaticDeck \**D*) {

    if (empty(*D*)) {

        printf("Дек пуст!\n");

        return -1;

    }

    return *D*->data[*D*->rear].value;

}

void clearDeck(StaticDeck \**D*) {

*D*->front = 0;

*D*->rear = -1;

*D*->size = 0;

}

void printDeck(StaticDeck \**D*) {

    if (empty(*D*)) {

        printf("Дек пуст!\n");

        return;

    }

    printf("Дек: ");

    int i = *D*->front;

    for (int count = 0; count < *D*->size; count++) {

        printf("[%d:%d] ", *D*->data[i].key, *D*->data[i].value);

        i = (i + 1) % MAX\_SIZE;

    }

    printf("\n");

}

key\_type frontKey(StaticDeck \**D*) {

    if (empty(*D*)) {

        printf("Дек пуст!\n");

        return -1;

    }

    return *D*->data[*D*->front].key;

}

**main.c**

#include "deck.h"

void mergeSort(StaticDeck \**D*) {

    if (size(*D*) <= 1) return;

    StaticDeck left, right;

    initDeck(&left);

    initDeck(&right);

    for (int i = 0; i < (size(*D*) / 2); i++) {

        pushBack(&left, frontKey(*D*), topFront(*D*));

        popFront(*D*);

    }

    while (!empty(*D*)) {

        pushBack(&right, frontKey(*D*), topFront(*D*));

        popFront(*D*);

    }

    mergeSort(&left);

    mergeSort(&right);

    initDeck(*D*);

    while (!empty(&left) && !empty(&right)) {

        if (frontKey(&left) <= frontKey(&right)) {

            pushBack(*D*, frontKey(&left), topFront(&left));

            popFront(&left);

        } else {

            pushBack(*D*, frontKey(&right), topFront(&right));

            popFront(&right);

        }

    }

    while (!empty(&left)) {

        pushBack(*D*, frontKey(&left), topFront(&left));

        popFront(&left);

    }

    while (!empty(&right)) {

        pushBack(*D*, frontKey(&right), topFront(&right));

        popFront(&right);

    }

}

void menu() {

    printf("\n");

    printf("1. Добавить в начало ");

    printf("2. Добавить в конец ");

    printf("3. Удалить из начала ");

    printf("4. Удалить с конца ");

    printf("5. Очистить дек ");

    printf("6. Отсортировать по ключу ");

    printf("7. Показать дек ");

    printf("8. Показать размер дека ");

    printf("0. Выход\n");

    printf("Выбор: ");

}

int main() {

    StaticDeck d;

    initDeck(&d);

    int choice;

    key\_type key;

    value\_type value;

    do {

        menu();

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                printf("Введите ключ и значение: ");

                scanf("%d %d", &*key*, &*value*);

                pushFront(&*d*, key, value);

                break;

            case 2:

                printf("Введите ключ и значение: ");

                scanf("%d %d", &*key*, &*value*);

                pushBack(&*d*, key, value);

                break;

            case 3:

                printf("Удалено из начала: %d\n", popFront(&*d*));

                break;

            case 4:

                printf("Удалено с конца: %d\n", popBack(&*d*));

                break;

            case 5:

                clearDeck(&*d*);

                printf("Дек очищен\n");

                break;

            case 6:

                mergeSort(&*d*);

                printf("Дек отсортирован\n");

                printDeck(&*d*);

                break;

            case 7:

                printDeck(&*d*);

                break;

            case 8:

                printf("Размер дека: %d\n", size(&*d*));

                break;

            case 0:

                printf("Выход\n");

                break;

            default:

                printf("Неверный выбор\n");

        }

    } while (choice != 0);

    return 0;

}

**Makefile**

CC = gcc

CFLAGS = -g -Wall -std=c99 -Werror

DEBUGGER = gdb

TARGET = deck.exe

SOURCES = deck.c main26.c

all: $(TARGET)

$(TARGET): $(SOURCES)

    $(CC) $(CFLAGS) $(SOURCES) -o $(TARGET)

debug: $(TARGET)

    $(DEBUGGER) $(TARGET)

run: $(TARGET)

    ./$(TARGET)

clean:

    rm -f $(TARGET)

1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем).

tru@empty:/mnt/c/c proj/vsc/sem2$ make

gcc -g -Wall -std=c99 -Werror deck.c main26.c -o deck.exe

tru@empty:/mnt/c/c proj/vsc/sem2$ make run

./deck.exe

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 1

Введите ключ и значение: 0 10

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 1

Введите ключ и значение: 1 12

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 1

Введите ключ и значение: 2 7

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 1

Введите ключ и значение: 3 234

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 2

Введите ключ и значение: 4 8

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 2

Введите ключ и значение: 5 1

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 2

Введите ключ и значение: 6 2

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 2

Введите ключ и значение: 7 3

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 1

Введите ключ и значение: 8 21

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 7

Дек: [8:21] [3:234] [2:7] [1:12] [0:10] [4:8] [5:1] [6:2] [7:3]

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 8

Размер дека: 9

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 1

Введите ключ и значение: 9 11

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 1

Введите ключ и значение: 10 123

Ошибка: дек переполнен

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 8

Размер дека: 10

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 7

Дек: [9:11] [8:21] [3:234] [2:7] [1:12] [0:10] [4:8] [5:1] [6:2] [7:3]

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 6

Дек отсортирован

Дек: [0:10] [1:12] [2:7] [3:234] [4:8] [5:1] [6:2] [7:3] [8:21] [9:11]

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 3

Удалено из начала: 10

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 3

Удалено из начала: 12

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 4

Удалено с конца: 11

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 4

Удалено с конца: 21

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 4

Удалено с конца: 3

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 3

Удалено из начала: 7

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 8

Размер дека: 4

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 7

Дек: [3:234] [4:8] [5:1] [6:2]

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 1

Введите ключ и значение: 9 9

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 1

Введите ключ и значение: 10 55

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 2

Введите ключ и значение: 11 98

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 7

Дек: [10:55] [9:9] [3:234] [4:8] [5:1] [6:2] [11:98]

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 6

Дек отсортирован

Дек: [3:234] [4:8] [5:1] [6:2] [9:9] [10:55] [11:98]

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 8

Размер дека: 7

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 5

Дек очищен

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 7

Дек пуст!

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 8

Размер дека: 0

1. Добавить в начало 2. Добавить в конец 3. Удалить из начала 4. Удалить с конца 5. Очистить дек 6. Отсортировать по ключу 7. Показать дек 8. Показать размер дека 0. Выход

Выбор: 0

Выход

tru@empty:/mnt/c/c proj/vsc/sem2$

1. **Дневник отладки** должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб. или дом. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

# Замечания автора по существу работы

1. **Выводы:**

В ходе выполнения лабораторной работы я научился составлять и отлаживать модуль определений и модуль реализации по заданной схеме модуля определений для дека, а также составлять программный модуль, сортирующий экземпляр дека методом слияния. Недочёты при выполнении задания могут быть устранены следующим образом:

**Подпись студента** \_\_\_\_\_\_\_\_\_\_\_\_