Отчёт по лабораторных работах №25-26.

по курсу «Языки и методы программирования».

Выполнил студент группы **М8О-112Б-21**: **Вещин Марат Константинович** № по списку **1**.

Контакты: **mafiramul@mail.ru**

Работа выполнена: «25» мая 2022 г.

Преподаватель: **каф. 806 Никулин Сергей Петрович**

Входной контроль знаний с оценкой: _____

Отчет сдан «25» мая 2022 г.

Итоговая оценка: _____

Подпись преподавателя: _____

1. **Тема:** абстрактные типы данных, рекурсия, модульное программирование на ЯП Си. Автоматизация сборки программ модульной структуры с использованием утилиты make.

2. **Цель работы:** применение различных сортировок к различным типам данных и обучение по работе с утилитой make.
3. **Задание:** АТД - дек, процедура - конкатенация, метод - быстрая сортировка Хоара.
4. **Оборудование**: *Оборудование ПЭВМ студента, если использовалось:*
Процессор **AMD Ryzen 5 5600H**, ОП **16** ГБ, SSD **250** ГБ, мониторы **15'' Full Hd Display** и **27" BenQ BL2780T**. Другие устройства: принтер **Canon MG4520S**, мышь **Logitech g403**, наушники **HyperX Cloud Alpha**.
5. **Программное обеспечение:** *Программное обеспечение ПЭВМ студента, если использовалось:*
Операционная система семейства **Ubuntu**, наименование версия **VirtualBox Ubuntu 20.04.3 LTS**,
интерпретатор команд **bash** версия **5.0.17**. Система программирования **C**. Редактор текстов **VI** версия **8.1**
6**. Идея, метод, алгоритм** *решения задачи [в формах: словесной, псевдокода, графической (блок-схема, диаграмма, рисунок, таблица) или формальные спецификации с пред- и постусловиями]:*

1. Продумать основную модульную структуру будущей программы.
2. Написать код.
3. Написать make файл.
4. Протестировать.

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

Заголовочный файл deck.h:

```
#ifndef _DECK_H_
#define _DECK_H_

#define N 100
#define Tvalue int

typedef struct Deck Deck;
typedef struct El_D El_D;

struct El_D {
   Tvalue V;
   El_D* next;
   El_D* previous;
};

struct Deck {
```

```
    El_D* first;
    El_D* last;
    int size;
};

void Init(Deck D);//
int Empty(Deck D);
void PushFront(Deck D, Tvalue V);//
void PushBack(Deck D, Tvalue V);//
Tvalue PopFront(Deck D);//
Tvalue PopBack(Deck D);//
Tvalue Top(Deck D);
int Size(Deck D);
void Display(Deck D);
void Cat(Deck D1, Deck D2);//
void Append(Deck D1, Deck D2);//

#endif
```

Файл с функциями дека deck.c:

```
#include "deck.h"
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#define Tvalue int

void Init(Deck D) {//
    D.first = 0;
    D.size = 0;
    D.last = 0;
}
int Empty(Deck D) {
    return D.size == 0;
}

void PushFront(Deck D, Tvalue V) {//
    El_D* t=(El_D*) malloc (sizeof (El_D));
    t->next=D.first;
    if (!Empty(D))
        D.first->previous=t;

    D.first=t;
    D.first->V=V;
    if (!D.size)
        D.last=t;

    D.size++;
}

void PushBack(Deck D, Tvalue V) {//
    El_D* t=(El_D*) malloc (sizeof (El_D));
    t->previous=D.last;
    if (!Empty(D))
        D.last->next=t;

    D.last=t;
    D.last->V=V;
    if (!D.size)
        D.first=t;

    D.size++;
}

Tvalue PopFront(Deck D) {//
    if (Empty(D))
        printf("Дек пуст");
    else {
        Tvalue V=D.first->V;
        D.first=D.first->next;
        D.size--;
        return V;
    }
```

```
    return -1;
}

Tvalue PopBack(Deck D) {//
    if (Empty(D))
        printf("Дек пуст");
    else {
        Tvalue V=D.last->V;
        D.last=D.last->previous;
        D.last->next=NULL;
        D.size--;
        return V;
    }
    return -1;
}

Tvalue Top(Deck D) {
    if (Empty(D))
        printf("Дек пуст");
    else
        return D.first->V;
    return -1;
}

int Size(Deck D) {
    return D.size;
}

void Display(Deck D) {
    if(Empty(D))
        printf("\nДек пуст\n");
    else {
        printf("\n[ ");
        El_D* t=D.first;
        while(t) {
            printf("%d ",t->V);
            t=t->next;
        }
        printf("]\n");
    }
}

void Cat(Deck D1, Deck D2) {//
    while (!Empty(D2))
        PushBack(D1, PopFront(D2));
}

void Append(Deck D1, Deck D2) {//
    int k = Size(D1);
    Cat(D1, D2);
    for (int i = 0; i < k; i++)
        PushBack(D1, PopFront(D1));
}
```

Файл основной программы main.c:

```
#include "deck.h"
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#define Tvalue int

void Init(Deck D) {//
    D.first = 0;
    D.size = 0;
    D.last = 0;
}
int Empty(Deck D) {
    return D.size == 0;
}

void PushFront(Deck D, Tvalue V) {//
    El_D* t=(El_D*) malloc (sizeof (El_D));
```

```
    t->next=D.first;
    if (!Empty(D))
        D.first->previous=t;

    D.first=t;
    D.first->V=V;
    if (!D.size)
        D.last=t;

    D.size++;
}

void PushBack(Deck D, Tvalue V) {//
    El_D* t=(El_D*) malloc (sizeof (El_D));
    t->previous=D.last;
    if (!Empty(D))
        D.last->next=t;

    D.last=t;
    D.last->V=V;
    if (!D.size)
        D.first=t;

    D.size++;
}

Tvalue PopFront(Deck D) {//
    if (Empty(D))
        printf("Дек пуст");
    else {
        Tvalue V=D.first->V;
        D.first=D.first->next;
        D.size--;
        return V;
    }
    return -1;
}

Tvalue PopBack(Deck D) {//
    if (Empty(D))
        printf("Дек пуст");
    else {
        Tvalue V=D.last->V;
        D.last=D.last->previous;
        D.last->next=NULL;
        D.size--;
        return V;
    }
    return -1;
}

Tvalue Top(Deck D) {
    if (Empty(D))
        printf("Дек пуст");
    else
        return D.first->V;
    return -1;
}

int Size(Deck D) {
    return D.size;
}

void Display(Deck D) {
    if(Empty(D))
        printf("\nДек пуст\n");
    else {
        printf("\n[ ");
        El_D* t=D.first;
        while(t) {
            printf("%d ",t->V);
            t=t->next;
        }
```

```
        printf("]\n");
    }
}

void Cat(Deck D1, Deck D2) {//
    while (!Empty(D2))
        PushBack(D1, PopFront(D2));
}

void Append(Deck D1, Deck D2) {//
    int k = Size(D1);
    Cat(D1, D2);
    for (int i = 0; i < k; i++)
        PushBack(D1, PopFront(D1));
}
```

Пункты 1-7 отчета составляются строго до начала лабораторной работы.

Допущен к выполнению работы. **Подпись преподавателя _____**

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

```
maratik@maratik:~/Рабочий стол/Рабочие$ ls
deck.c  deck.h  main.c  makefile
maratik@maratik:~/Рабочий стол/Рабочие$ make
gcc -c deck.c
gcc -c main.c
gcc deck.o main.o
maratik@maratik:~/Рабочий стол/Рабочие$ ls
a.out  deck.c  deck.h  deck.o  main.c  main.o  makefile
maratik@maratik:~/Рабочий стол/Рабочие$ ./a.out

Выберите действие:
1. Создать случайный дек
2. Распечатать дек
3. Вывести размер дека
4. Вставить в начало дека
5. Вставить в конец дека
6. Удалить первый элемент дека
7. Удалить последний элемент дека
8. Сортировка дека
9. Очистить дек
10. Конкатенация двух деков
0. Выйти
Введите номер действия =>1
Введите количество элементов =>5

[ 6 10 1 2 6 ]

Выберите действие:
1. Создать случайный дек
2. Распечатать дек
3. Вывести размер дека
4. Вставить в начало дека
5. Вставить в конец дека
6. Удалить первый элемент дека
7. Удалить последний элемент дека
8. Сортировка дека
9. Очистить дек
10. Конкатенация двух деков
0. Выйти
Введите номер действия =>2

[ 6 10 1 2 6 ]

Выберите действие:
1. Создать случайный дек
2. Распечатать дек
3. Вывести размер дека
4. Вставить в начало дека
5. Вставить в конец дека
6. Удалить первый элемент дека
7. Удалить последний элемент дека
8. Сортировка дека
9. Очистить дек
10. Конкатенация двух деков
```

0. Выйти
Введите номер действия =>3

Размер дека =5

Выберите действие:
1. Создать случайный дек
2. Распечатать дек
3. Вывести размер дека
4. Вставить в начало дека
5. Вставить в конец дека
6. Удалить первый элемент дека
7. Удалить последний элемент дека
8. Сортировка дека
9. Очистить дек
10. Конкатенация двух деков
0. Выйти
Введите номер действия =>4

Введите значение элемента =>23

Выберите действие:
1. Создать случайный дек
2. Распечатать дек
3. Вывести размер дека
4. Вставить в начало дека
5. Вставить в конец дека
6. Удалить первый элемент дека
7. Удалить последний элемент дека
8. Сортировка дека
9. Очистить дек
10. Конкатенация двух деков
0. Выйти
Введите номер действия =>5

Введите значение элемента =>32

Выберите действие:
1. Создать случайный дек
2. Распечатать дек
3. Вывести размер дека
4. Вставить в начало дека
5. Вставить в конец дека
6. Удалить первый элемент дека
7. Удалить последний элемент дека
8. Сортировка дека
9. Очистить дек
10. Конкатенация двух деков
0. Выйти
Введите номер действия =>2

[23 6 10 1 2 6 32]

Выберите действие:
1. Создать случайный дек
2. Распечатать дек
3. Вывести размер дека
4. Вставить в начало дека
5. Вставить в конец дека
6. Удалить первый элемент дека
7. Удалить последний элемент дека
8. Сортировка дека
9. Очистить дек
10. Конкатенация двух деков
0. Выйти
Введите номер действия =>6

Элемент =23 удалён

Выберите действие:
1. Создать случайный дек
2. Распечатать дек
3. Вывести размер дека
4. Вставить в начало дека
5. Вставить в конец дека
6. Удалить первый элемент дека
7. Удалить последний элемент дека
8. Сортировка дека
9. Очистить дек
10. Конкатенация двух деков
0. Выйти

Введите номер действия =>2

[6 10 1 2 6 32]

Выберите действие:
1. Создать случайный дек
2. Распечатать дек
3. Вывести размер дека
4. Вставить в начало дека
5. Вставить в конец дека
6. Удалить первый элемент дека
7. Удалить последний элемент дека
8. Сортировка дека
9. Очистить дек
10. Конкатенация двух деков
0. Выйти
Введите номер действия =>7

Элемент =32 удалён

Выберите действие:
1. Создать случайный дек
2. Распечатать дек
3. Вывести размер дека
4. Вставить в начало дека
5. Вставить в конец дека
6. Удалить первый элемент дека
7. Удалить последний элемент дека
8. Сортировка дека
9. Очистить дек
10. Конкатенация двух деков
0. Выйти
Введите номер действия =>2

[6 10 1 2 6]

Выберите действие:
1. Создать случайный дек
2. Распечатать дек
3. Вывести размер дека
4. Вставить в начало дека
5. Вставить в конец дека
6. Удалить первый элемент дека
7. Удалить последний элемент дека
8. Сортировка дека
9. Очистить дек
10. Конкатенация двух деков
0. Выйти
Введите номер действия =>10

Введите размер второго дека =>2

Выберите действие:
1) Создать рандомный дек
2) Задать элементы дека вручную
Введите номер действия =>2

Введите значения элементов =>4 0

[6 10 1 2 6 4 0]

Выберите действие:
1. Создать случайный дек
2. Распечатать дек
3. Вывести размер дека
4. Вставить в начало дека
5. Вставить в конец дека
6. Удалить первый элемент дека
7. Удалить последний элемент дека
8. Сортировка дека
9. Очистить дек
10. Конкатенация двух деков
0. Выйти
Введите номер действия =>3

Размер дека =7

Выберите действие:
1. Создать случайный дек
2. Распечатать дек
3. Вывести размер дека

4. Вставить в начало дека
5. Вставить в конец дека
6. Удалить первый элемент дека
7. Удалить последний элемент дека
8. Сортировка дека
9. Очистить дек
10. Конкатенация двух деков
0. Выйти
Введите номер действия =>8

[0 1 2 4 6 6 10]

Выберите действие:
1. Создать случайный дек
2. Распечатать дек
3. Вывести размер дека
4. Вставить в начало дека
5. Вставить в конец дека
6. Удалить первый элемент дека
7. Удалить последний элемент дека
8. Сортировка дека
9. Очистить дек
10. Конкатенация двух деков
0. Выйти
Введите номер действия =>0
maratik@maratik:~/Рабочий стол/Рабочие$

9. **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

10. **Замечания автора** по существу работы: замечания отсутствуют.

11. **Выводы:**

Подпись студента _____