**Отчет по лабораторной работе №25-26** по курсу 

Языки и методы программирования \_\_\_\_\_\_\_\_\_\_\_\_

Студент группы М8О-101Б-21 Постнов Александр Вячеславович, № по списку 17

Контакты www, e-mail: 61pav03@mail.ru

Работа выполнена: «» 202 2г.

Преподаватель: каф. 806 \_\_\_\_\_ Титов В.К.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Входной контроль знаний с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отчет сдан « » \_\_\_\_\_\_\_\_\_ 2022\_ г., итоговая оценка \_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Тема:** Абстрактные типы данных. Рекурсия. Модульное программирование на языке Си.

Утилита make \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

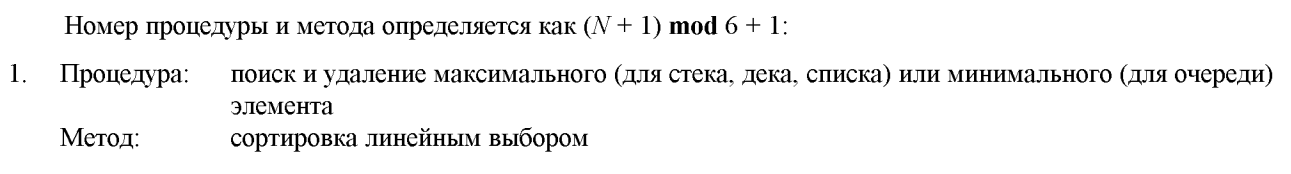
1. **Цель работы:** Составить и отладить модуль определений и модуль реализации по заданной схеме модуля определений для абстрактного(пользовательского) типа данных(стека, очереди, списка или дека, в зависимости от варианта задания). Составить программный модуль, сортирующий экземпляр указанного абстрактного типа данных заданным методом. с помощью утилиты make\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Задание** (*вариант №17* )**:**

Реализовать Дек ((17+1)%4+1=3)

Сортировать дек указанным методом: ((17+1)%6+1=1)



1. **Оборудование(лабораторное)**:

ЭВМ -, процессор -, имя узла сети - с ОП - ГБ,

НМД - ГБ, терминал- адрес -, принтер -

Другие устройства -

*Оборудование ПЭВМ студента, если использовалось:*

Процессор AMD Ryzen 5 4500U, с ОП 8 ГБ

Другие устройства -

1. **Программное обеспечение:**

Операционная система семейства -, наименование - версия - интерпретатор команд - версия

Система программирования - версия -

Редактор текстов - версия -

Утилиты операционной системы -

Прикладные системы и программы -

Местонахождение и имена файлов программ и данных -

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства GNU/Linux, наименование Manjaro версия 5-13-12-1

интерпретатор команд GNOME Terminal версия 3.38.2.

Система программирования \_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_версия\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Редактор текстов emacs версия 3.27.20

Утилиты операционной системы

Прикладные системы и программы -

Местонахождение и имена файлов программ и данных -

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

Дек (от англ. deque — double ended queue) — структура данных, представляющая из себя список элементов, в которой добавление новых элементов и удаление существующих производится с обоих концов. Эта структура поддерживает как FIFO, так и LIFO, поэтому на ней можно реализовать как стек, так и очередь. В первом случае нужно использовать только методы головы или хвоста, во втором — методы push и pop двух разных концов. Дек можно воспринимать как двустороннюю очередь. Он имеет следующие операции:

empty — проверка на наличие элементов,

pushBack (запись в конец) — операция вставки нового элемента в конец,

popBack (снятие с конца) — операция удаления конечного элемента,

pushFront (запись в начало) — операция вставки нового элемента в начало,

popFront (снятие с начала) — операция удаления начального элемента.

Реализую эту структуру данных двумя способами: 1) с отображением на массив 2) с отображением на динамические структуры.

Сортировка линейным выбором:

1. Реализую процедуру, которая будет искать максимальный элемент и удалять его
2. с помощью этой процедуры буду сортировать дек.

Сборка программы с модулями будет реализована с помощью утилиты make

В итоге будут 2 программы main.cpp, в которой реализована сортировка для дека, они будут отличаться только подключением дека (на массиве или на динам. структуре).

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

**deck1.h**

#ifndef DECK1\_H

#define DECK1\_H

#include <stdlib.h>

#define N 100

#define Tvalue int

struct Deck { // реализация дека с отображением на массив

int first, last, size;

Tvalue body[N];

};

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

**deck1.cpp**

#include <stdio.h>

#include <stdlib.h>

#include "deck1.h"

void Init(Deck &D) { // инициализация дека

D.first = D.size = 0;

D.last = N - 1;

}

int Empty(Deck D) { return D.size == 0; } // проверка на то, что дек пустой

void PushFront(Deck &D, Tvalue V) { // добавление элемента в начало

if (D.size == N)

printf("\nDECK IS OVERFLOW\n");

else {

D.body[(D.first) ? --D.first : (D.first = N - 1)] = V;

D.size++;

}

}

void PushBack(Deck &D, Tvalue V) { // добавление элемента в конец

if (D.size == N)

printf("\nDECK IS OVERFLOW\n");

else {

D.body[++D.last %= N] = V;

D.size++;

}

}

Tvalue PopFront(Deck &D) { // удаление верхнего элемента и возвращение его

if (Empty(D))

printf("\nDECK IS EMPTY\n");

else {

D.size--;

int i = D.first++;

D.first %= N;

return D.body[i];

}

return -1;

}

Tvalue PopBack(Deck &D) { // удаление последнего элемента и возвращение его

if (Empty(D))

printf("\nDECK IS EMPTY\n");

else {

D.size--;

int i = D.last;

(D.last) ? D.last-- : (D.last = N - 1);

return D.body[i];

}

return -1;

}

Tvalue Top(Deck D) { // возвращает первое значение

if (Empty(D))

printf("\nDECK IS EMPTY\n");

else

return D.body[D.first];

return -1;

}

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

void Display(Deck D) { // вывод дека

if (Empty(D)) {

printf("\nDECK IS EMPTY\n");

}

printf("\n[");

for (int i = D.first; i < D.first + D.size; i++)

printf("%d%s", D.body[i % N], (i < D.first + D.size - 1) ? "," : "");

printf("]\n");

}

void Cat(Deck &D1, Deck &D2) { // добавляет элементы второго дека(с начала) в конец первого дека

while (!Empty(D2))

PushBack(D1, PopFront(D2));

}

void Append(Deck &D1, Deck &D2) { // добавление элементов второго дека в начало первого

int k = Size(D1);

Cat(D1, D2);

for (int i = 0; i < k; i++)

PushBack(D1, PopFront(D1));

}

**main1.cpp**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "deck1.h"

void randomize() {

long a = time(0);

srand(a);

}

int RemoveMax(Deck &S, Tvalue M1, Tvalue &M) {

Tvalue V;

int Max\_removed;

if (Empty(S)) {

M = M1;

return 0;

} else {

V = PopFront(S);

if (V > M1)

M1 = V;

}

Max\_removed = RemoveMax(S, M1, M);

if (Max\_removed || (V != M))

PushFront(S, V);

else

Max\_removed = 1;

return Max\_removed;

}

void FetchSort(Deck &S) { //сортировка выбором

Deck S1;

Tvalue Max;

Init(S1);

while (!Empty(S)) {

RemoveMax(S, Top(S), Max);

PushFront(S1, Max);

}

Cat(S, S1);

}

int main() {

randomize();

Deck S1, S2;

Init(S1);

Tvalue v;

for (int i = 0; i < 10; i++) {

v = rand() % 25;

PushBack(S1, v);

}

Display(S1);

Init(S2);

for (int i = 0; i < 7; i++) {

v = rand() % 25;

PushBack(S2, v);

}

Display(S2);

Append(S1, S2);

Display(S1);

FetchSort(S1);

Display(S1);

}

**deck2.h**

#ifndef DECK2\_H

#define DECK2\_H

#include <stdlib.h>

#define N 100

#define Tvalue int

struct El\_d {

Tvalue V;

El\_d \*next;

};

struct Deck { // реализация дека с отображением на динамические структуры

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

**deck2.cpp**

#include <stdio.h>

#include <stdlib.h>

#include "deck2.h"

void Init(Deck &S) { // инициализация дека

S.first = 0;

S.last = 0;

S.size = 0;

}

int Empty(Deck S) { return S.first == 0; } //проверка на пустоту дека

void PushFront(Deck &S, Tvalue V) { // добавление элемента в начало дека

if (Empty(S)) { // когда дек пустой

El\_d \*t = new El\_d;

t->next = 0;

t->V = V;

S.first = t;

S.last = t;

S.size++;

return;

}

El\_d \*t = new El\_d;

t->next = S.first;

S.first = t;

S.first->V = V;

S.size++;

}

void PushBack(Deck &S, Tvalue V) { //добавление элемента в конец дека

if (Empty(S)) { // когда дек пустой

El\_d \*t = new El\_d;

t->next = 0;

t->V = V;

S.size++;

S.first = t;

S.last = t;

return;

}

El\_d \*t = new El\_d;

t->next = 0;

t->V = V;

S.last->next = t;

S.last = t;

if (S.size == 1) {

S.first->next = S.last;

}

S.size++;

}

Tvalue PopFront(Deck &S) { //удаление с начала

if (Empty(S))

printf("\nDeck IS EMPTY\n");

else {

Tvalue V = S.first->V;

S.first = S.first->next;

S.size--;

if (Empty(S)) {

S.last = 0;

}

return V;

}

return -1;

}

Tvalue PopBack(Deck &S) { //eудаление с конца

if (Empty(S)) {

printf("\nDeck IS EMPTY\n");

}

else {

El\_d \*t = S.first;

if (S.size == 1) {

Tvalue V = S.first->V;

S.last = 0;

S.first = 0;

S.size = 0;

return V;

}

else {

while (1) {

if (t->next == S.last) {

Tvalue V = S.last->V;

S.last = 0;

S.last = t;

return V;

}

else {

t = t->next;

}

}

}

}

return -1;

}

Tvalue Top(Deck S) { //первое значение

if (Empty(S))

printf("\nDeck IS EMPTY\n");

else

return S.first->V;

return -1;

}

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

void Display(Deck S) {

if (Empty(S))

printf("\nDeck IS EMPTY\n");

else {

printf("\n[");

El\_d \*t = S.first;

while (t->next != S.last) {

printf("%d,", t->V);

t = t->next;

}

printf("%d]\n", t->next->V);

}

}

void Cat(Deck &S1, Deck &S2) {

Tvalue V;

if (!Empty(S2)) {

V = PopFront(S2);

Cat(S1, S2);

PushFront(S1, V);

}

}

void Append(Deck &S1, Deck &S2) {

Tvalue V;

if (Empty(S1))

Cat(S1, S2);

else {

V = PopFront(S1);

Append(S1, S2);

PushFront(S1, V);

}

}

**main2.cpp**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "deck2.h"

void randomize() {

long a = time(0);

srand(a);

}

int RemoveMax(Deck &S, Tvalue M1, Tvalue &M) {

Tvalue V;

int Max\_removed;

if (Empty(S)) {

M = M1;

return 0;

} else {

V = PopFront(S);

if (V > M1)

M1 = V;

}

Max\_removed = RemoveMax(S, M1, M);

if (Max\_removed || (V != M))

PushFront(S, V);

else

Max\_removed = 1;

return Max\_removed;

}

void FetchSort(Deck &S) { //сортировка выбором

Deck S1;

Tvalue Max;

Init(S1);

while (!Empty(S)) {

RemoveMax(S, Top(S), Max);

PushFront(S1, Max);

}

Cat(S, S1);

}

int main() {

randomize();

Deck S1, S2;

Init(S1);

Tvalue v;

for (int i = 0; i < 10; i++) {

v = rand() % 25;

PushBack(S1, v);

}

Display(S1);

Init(S2);

for (int i = 0; i < 7; i++) {

v = rand() % 25;

PushBack(S2, v);

}

Display(S2);

Append(S1, S2);

Display(S1);

FetchSort(S1);

Display(S1);

}

**makefile**

CC = g++

CFLAGS = -g -Wall -Wextra -pedantic -std=c99 -w -pipe -O3 -lm

lab26\_a: deck1.o main1.o

$(CC) $^ -o $@ $(CFLAGS)

lab26\_d: deck2.o main2.o

$(CC) $^ -o $@ $(CFLAGS)

clean:

rm -f \*.o lab26\_a

rm -f \*.o lab26\_d

Тестирование: сортировка дека будет проверяться с помощью генератора случайных чисел.

*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).

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ cat head.txt

-----------------------------------------------------

| Лабораторная работа №25-26 |

| Абстрактные типы данных |

| Выполнил: студент группы М8О-101Б-21 |

| Постнов Александр Вячеславович |

-----------------------------------------------------

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ cat deck1.h

#ifndef DECK1\_H

#define DECK1\_H

#include <stdlib.h>

#define N 100

#define Tvalue int

struct Deck { // реализация дека с отображением на массив

int first, last, size;

Tvalue body[N];

};

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

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ cat deck1.cpp

#include <stdio.h>

#include <stdlib.h>

#include "deck1.h"

void Init(Deck &D) { // инициализация дека

D.first = D.size = 0;

D.last = N - 1;

}

int Empty(Deck D) { return D.size == 0; } // проверка на то, что дек пустой

void PushFront(Deck &D, Tvalue V) { // добавление элемента в начало

if (D.size == N)

printf("\nDECK IS OVERFLOW\n");

else {

D.body[(D.first) ? --D.first : (D.first = N - 1)] = V;

D.size++;

}

}

void PushBack(Deck &D, Tvalue V) { // добавление элемента в конец

if (D.size == N)

printf("\nDECK IS OVERFLOW\n");

else {

D.body[++D.last %= N] = V;

D.size++;

}

}

Tvalue PopFront(Deck &D) { // удаление верхнего элемента и возвращение его

if (Empty(D))

printf("\nDECK IS EMPTY\n");

else {

D.size--;

int i = D.first++;

D.first %= N;

return D.body[i];

}

return -1;

}

Tvalue PopBack(Deck &D) { // удаление последнего элемента и возвращение его

if (Empty(D))

printf("\nDECK IS EMPTY\n");

else {

D.size--;

int i = D.last;

(D.last) ? D.last-- : (D.last = N - 1);

return D.body[i];

}

return -1;

}

Tvalue Top(Deck D) { // возвращает первое значение

if (Empty(D))

printf("\nDECK IS EMPTY\n");

else

return D.body[D.first];

return -1;

}

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

void Display(Deck D) { // вывод дека

if (Empty(D)) {

printf("\nDECK IS EMPTY\n");

}

printf("\n[");

for (int i = D.first; i < D.first + D.size; i++)

printf("%d%s", D.body[i % N], (i < D.first + D.size - 1) ? "," : "");

printf("]\n");

}

void Cat(Deck &D1, Deck &D2) { // добавляет элементы второго дека(с начала) в конец первого дека

while (!Empty(D2))

PushBack(D1, PopFront(D2));

}

void Append(Deck &D1, Deck &D2) { // добавление элементов второго дека в начало первого

int k = Size(D1);

Cat(D1, D2);

for (int i = 0; i < k; i++)

PushBack(D1, PopFront(D1));

}

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ cat main.cpp

cat: main.cpp: Нет такого файла или каталога

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ cat main1.cpp

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "deck1.h"

void randomize() {

long a = time(0);

srand(a);

}

int RemoveMax(Deck &S, Tvalue M1, Tvalue &M) {

Tvalue V;

int Max\_removed;

if (Empty(S)) {

M = M1;

return 0;

} else {

V = PopFront(S);

if (V > M1)

M1 = V;

}

Max\_removed = RemoveMax(S, M1, M);

if (Max\_removed || (V != M))

PushFront(S, V);

else

Max\_removed = 1;

return Max\_removed;

}

void FetchSort(Deck &S) { //сортировка выбором

Deck S1;

Tvalue Max;

Init(S1);

while (!Empty(S)) {

RemoveMax(S, Top(S), Max);

PushFront(S1, Max);

}

Cat(S, S1);

}

int main() {

randomize();

Deck S1, S2;

Init(S1);

Tvalue v;

for (int i = 0; i < 10; i++) {

v = rand() % 25;

PushBack(S1, v);

}

Display(S1);

Init(S2);

for (int i = 0; i < 7; i++) {

v = rand() % 25;

PushBack(S2, v);

}

Display(S2);

Append(S1, S2);

Display(S1);

FetchSort(S1);

Display(S1);

}

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ cat deck2.h

#ifndef DECK2\_H

#define DECK2\_H

#include <stdlib.h>

#define N 100

#define Tvalue int

struct El\_d {

Tvalue V;

El\_d \*next;

};

struct Deck { // реализация дека с отображением на динамические структуры

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

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ cat deck2.cpp

#include <stdio.h>

#include <stdlib.h>

#include "deck2.h"

void Init(Deck &S) { // инициализация дека

S.first = 0;

S.last = 0;

S.size = 0;

}

int Empty(Deck S) { return S.first == 0; } //проверка на пустоту дека

void PushFront(Deck &S, Tvalue V) { // добавление элемента в начало дека

if (Empty(S)) { // когда дек пустой

El\_d \*t = new El\_d;

t->next = 0;

t->V = V;

S.first = t;

S.last = t;

S.size++;

return;

}

El\_d \*t = new El\_d;

t->next = S.first;

S.first = t;

S.first->V = V;

S.size++;

}

void PushBack(Deck &S, Tvalue V) { //добавление элемента в конец дека

if (Empty(S)) { // когда дек пустой

El\_d \*t = new El\_d;

t->next = 0;

t->V = V;

S.size++;

S.first = t;

S.last = t;

return;

}

El\_d \*t = new El\_d;

t->next = 0;

t->V = V;

S.last->next = t;

S.last = t;

if (S.size == 1) {

S.first->next = S.last;

}

S.size++;

}

Tvalue PopFront(Deck &S) { //удаление с начала

if (Empty(S))

printf("\nDeck IS EMPTY\n");

else {

Tvalue V = S.first->V;

S.first = S.first->next;

S.size--;

if (Empty(S)) {

S.last = 0;

}

return V;

}

return -1;

}

Tvalue PopBack(Deck &S) { //eудаление с конца

if (Empty(S)) {

printf("\nDeck IS EMPTY\n");

}

else {

El\_d \*t = S.first;

if (S.size == 1) {

Tvalue V = S.first->V;

S.last = 0;

S.first = 0;

S.size = 0;

return V;

}

else {

while (1) {

if (t->next == S.last) {

Tvalue V = S.last->V;

S.last = 0;

S.last = t;

return V;

}

else {

t = t->next;

}

}

}

}

return -1;

}

Tvalue Top(Deck S) { //первое значение

if (Empty(S))

printf("\nDeck IS EMPTY\n");

else

return S.first->V;

return -1;

}

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

void Display(Deck S) {

if (Empty(S))

printf("\nDeck IS EMPTY\n");

else {

printf("\n[");

El\_d \*t = S.first;

while (t->next != S.last) {

printf("%d,", t->V);

t = t->next;

}

printf("%d]\n", t->next->V);

}

}

void Cat(Deck &S1, Deck &S2) {

Tvalue V;

if (!Empty(S2)) {

V = PopFront(S2);

Cat(S1, S2);

PushFront(S1, V);

}

}

void Append(Deck &S1, Deck &S2) {

Tvalue V;

if (Empty(S1))

Cat(S1, S2);

else {

V = PopFront(S1);

Append(S1, S2);

PushFront(S1, V);

}

}

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ cat main2.cpp

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "deck2.h"

void randomize() {

long a = time(0);

srand(a);

}

int RemoveMax(Deck &S, Tvalue M1, Tvalue &M) {

Tvalue V;

int Max\_removed;

if (Empty(S)) {

M = M1;

return 0;

} else {

V = PopFront(S);

if (V > M1)

M1 = V;

}

Max\_removed = RemoveMax(S, M1, M);

if (Max\_removed || (V != M))

PushFront(S, V);

else

Max\_removed = 1;

return Max\_removed;

}

void FetchSort(Deck &S) { //сортировка выбором

Deck S1;

Tvalue Max;

Init(S1);

while (!Empty(S)) {

RemoveMax(S, Top(S), Max);

PushFront(S1, Max);

}

Cat(S, S1);

}

int main() {

randomize();

Deck S1, S2;

Init(S1);

Tvalue v;

for (int i = 0; i < 10; i++) {

v = rand() % 25;

PushBack(S1, v);

}

Display(S1);

Init(S2);

for (int i = 0; i < 7; i++) {

v = rand() % 25;

PushBack(S2, v);

}

Display(S2);

Append(S1, S2);

Display(S1);

FetchSort(S1);

Display(S1);

}

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ cat Makefile

CC = g++

CFLAGS = -g -Wall -Wextra -pedantic -std=c99 -w -pipe -O3 -lm

lab26\_a: deck1.o main1.o

$(CC) $^ -o $@ $(CFLAGS)

lab26\_d: deck2.o main2.o

$(CC) $^ -o $@ $(CFLAGS)

clean:

rm -f \*.o lab26\_a

rm -f \*.o lab26\_d

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ make lab26\_a

g++ -c -o main1.o main1.cpp

g++ deck1.o main1.o -o lab26\_a -g -Wall -Wextra -pedantic -std=c99 -w -pipe -O3 -lm

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ ./lab26\_a

[3,5,16,10,0,19,6,23,21,10]

[8,0,18,3,17,16,12]

[8,0,18,3,17,16,12,3,5,16,10,0,19,6,23,21,10]

[0,0,3,3,5,6,8,10,10,12,16,16,17,18,19,21,23]

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ ./lab26\_a

[6,10,5,17,9,23,8,18,22,15]

[23,1,12,13,13,18,14]

[23,1,12,13,13,18,14,6,10,5,17,9,23,8,18,22,15]

[1,5,6,8,9,10,12,13,13,14,15,17,18,18,22,23,23]

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ ./lab26\_a

[3,5,7,5,11,18,7,24,23,13]

[11,3,9,9,10,18,16]

[11,3,9,9,10,18,16,3,5,7,5,11,18,7,24,23,13]

[3,3,5,5,7,7,9,9,10,11,11,13,16,18,18,23,24]

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ make lab26\_d

g++ -c -o deck2.o deck2.cpp

g++ -c -o main2.o main2.cpp

g++ deck2.o main2.o -o lab26\_d -g -Wall -Wextra -pedantic -std=c99 -w -pipe -O3 -lm

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ ./lab26\_d

[18,9,7,21,8,13,18,18,22]

[19,21,23,3,24,21]

[18,9,7,21,8,13,18,18,15,22,19,21,23,3,24,21]

[3,7,8,9,13,15,18,18,18,19,21,21,21,21,22,24]

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ ./lab26\_d

[12,23,15,18,22,11,13,7,22]

[2,10,16,24,3,10]

[12,23,15,18,22,11,13,7,23,22,2,10,16,24,3,10]

[2,3,7,10,10,11,12,13,15,16,16,18,22,22,23,24]

alex@alex-VivoBook-ASUSLaptop-X421IA-M433IA:~/mai\_labs/2 семестр/25-26(?)$ ./lab26\_d

[14,20,17,24,3,24,11,8,1]

[24,16,3,0,17,17]

[14,20,17,24,3,24,11,8,9,1,24,16,3,0,17,17]

[0,1,3,3,8,9,11,13,14,16,17,17,17,20,24,24]

**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 |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | дом | 28.05.2022 | 16:35 | Забыл поставить точку с запятой | Поставил точку с запятой |  |

**10. Замечания автора**

**11**. **Вывод**

В ходе лабораторной работы я изучил абстрактные структуры данных, научился работать с ними \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Недочеты при выполнении задания могут быть устранены следующим образом: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись студента \_\_ Постнов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_