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

Студентка группы М8О-113Б-22, № по списку 1, Астахова Анастасия Сергеевна

Контакты astakhovaanastasia0201@gmail.com

Работа выполнена: « » \_\_\_\_\_\_\_\_ 2023 г.

Преподаватель: доцент каф. 806 Никулин Сергей Петрович

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

Отчет сдан « » \_\_\_\_\_\_\_\_\_2023 г., итоговая оценка \_\_\_\_\_

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

* 1. **Тема:** Абстрактные типы данных. Рекурсия. Модульное программирование на языке Си. А также автоматизация сборки программ модульной структуры на языке Си с использованием утилиты make.

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

1. **Цель работы:** Составить и отладить модуль определений и модуль реализации по заданной схеме модуля определений для абстрактного типа данных. Составить программный модуль, сортирующий экземляр указанного абстрактного типа данных заданным методом, используя только операции, импортированные из модуля UUDT.\_Сборку программы осуществить с помощью утилиты make.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

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

1. **Задание (номер типа данных 3, номер процедуры 3):** Альтернативный тип данных - дек, выполнить быструю сортировку Хоара, используя конкатенацию двух деков. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. **Оборудование** (лабораторное):

ЭВМ \_\_\_\_\_\_\_\_\_, процессор \_\_\_\_ \_\_, имя узла сети \_\_\_\_\_\_ с ОП \_\_\_\_\_\_\_\_ Мб, НМД \_\_\_\_\_\_\_ Мб. Терминал \_\_\_\_\_ адрес \_\_\_\_\_\_\_\_\_\_. Принтер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Другие устройства \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

*Оборудование ПЭВМ студента, если использовалось:*

Ноутбук: Процессор Intel Core i5-10210U с ОП 16 ГБ, SSD 512 ГБ. Другие устройства: мышь Bloody R80

*Компьютер: Процессор Ryzen 5 3600 с ОП 16 ГБ, SSD 512 ГБ. Другие устройства: мышь* Razer BASILISK X HYPERSPEED, [*Logitech G910 (920-008019) Orion Spectrum RGB Mechanical Gaming*](https://www.komus.ru/katalog/tekhnika/kompyuternaya-tekhnika/klaviatury-i-myshi/klaviatury-provodnye/klaviatura-logitech-g910-920-008019-orion-spectrum-rgb-mechanical-gaming/p/659916/?from=n-13776), монитор [*LG UltraGear 27GP750*](https://www.lg.com/ru/monitors/lg-27gp750-b)

1. **Программное обеспечение (лабораторное):**

Операционная система семейства \_\_\_Unix\_\_\_\_, наименование \_\_\_\_\_Ubuntu\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_4.15.0\_\_\_\_\_\_

интерпретатор команд \_\_\_\_bash\_\_\_\_\_\_\_\_ версия \_\_\_4.4.20\_\_\_\_

Система программирования \_\_С\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_\_последняя\_\_\_\_\_\_\_\_\_

Редактор текстов \_\_\_gedit\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_25.2.2 \_\_\_\_\_\_\_\_\_

Утилиты операционной системы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Прикладные системы и программы \_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Местонахождение и имена файлов программ и данных\_\_stud/208104 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

**5.** *Программное обеспечение ПЭВМ студента, если использовалось****:***

Ноутбук: Операционная система семейства Ubuntu, наименование версия Ubuntu 20.04.3 LTS,

*интерпретатор команд bash версия 5.0.17. Система программирования C. Редактор текстов gedit версия 28.2. Утилиты: стандартные утилиты UNIX. Программное обеспечение: стандартное программное обеспечение системы UNIX.*

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

Дек будет реализован с использованием следующих функций:

1. Создание

2. Проверка на пустоту

3. Добавить элемент спереди

4. Добавить элемент сзади

5. Удалить элемент спереди

6. Удалить элемент сзади

7. Размер дека

8. Первый элемент дека

9. Последний элемент дека

10. Распечатка дека

Основной файл будет включать функции:

1. Конкатенация двух деков

Принимает на вход два дека, склеивает из, возвращает один дек

2. Быстрая сортировка Хоара

На вход принимает дек, сортирует его, используя заданный метод. Функция выполнена рекурсивно

Будет написан makefile, реализация приведена в следующем пункте.

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

Тесты:

PS D:\c\_labs\lab\_26> make

gcc –c deque.c

gcc –c main.c

PS D:\c\_labs\lab\_26> ./main

Input your command:

1. Create deq

2. Empty

3. Size

4. Push back

5. Push front

6. Top left

7. Top right

8.Pop back

9. Pop front

10.Print

11. Sort

12. Exit

1

4

Input num, that is going to be pushed

5

4

Input num, that is going to be pushed

1

4

Input num, that is going to be pushed

3

4

Input num, that is going to be pushed

2

4

Input num, that is going to be pushed

4

3

The size of deque is equal to 5.

10

5 1 3 2 4

11

10

1 2 3 4 5

0

PS D:\c\_labs\lab\_26> ./main

Input your command:

1. Create deq

2. Empty

3. Size

4. Push back

5. Push front

6. Top left

7. Top right

8.Pop back

9. Pop front

10.Print

11. Sort

12. Exit

1

4

Input num, that is going to be pushed

3

4

Input num, that is going to be pushed

2

4

Input num, that is going to be pushed

1

10

3 2 1

5

Input num, that is going to be pushed

4

5

Input num, that is going to be pushed

5

10

5 4 3 2 1

5

Input num, that is going to be pushed

1

Deque is full.

10

5 4 3 2 1

11

10

1 2 3 4 5

0

PS D:\c\_labs\lab\_26> ./main

Input your command:

1. Create deq

2. Empty

3. Size

4. Push back

5. Push front

6. Top left

7. Top right

8.Pop back

9. Pop front

10.Print

11. Sort

12. Exit

1

2

Yes, the deque is empty.

3

The size of deque is equal to 0.

4

Input num, that is going to be pushed

3

4

Input num, that is going to be pushed

2

4

Input num, that is going to be pushed

5

4

Input num, that is going to be pushed

1

3

The size of deque is equal to 4.

5 2

Input num, that is going to be pushed

2

No, the deque is not empty.

3

The size of deque is equal to 5.

10

2 3 2 5 1

6

The front elem is 2

7

The back elem is 1

8

2

No, the deque is not empty.

9

10

3 2 5

6

The front elem is 3

7

The back elem is 5

11

10

2 3 5

0

*Пункты 1-7 отчета составляются сторого до начала лабораторной работы.*

*Допущен к выполнению работы.* **Подпись преподавателя****\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

Код лабораторной работы:

MakeFile

**laba: deque.o main.o**

**gcc deque.o main.o**

**queue.o : deque.h deque.c**

**gcc -c deque.c**

**main.o : deque.h main.c**

**gcc -c main.c**

deque.c

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#include "deque.h"

deque \*deque\_create(deque \*d) {

d = (deque\*)malloc(sizeof(deque));

d->size = 0;

}

bool deque\_isempty(deque \*d) {

if (d->size == 0) {

return true;

}

else {

return false;

}

}

void deque\_push\_back(deque \*d, int elem) {

if (d->size == MAXSIZE) {

printf("Deque is full.\n");

return;

}

d->arr[d->size] = elem;

d->size++;

}

void deque\_push\_front(deque \*d, int elem) {

if (d->size == MAXSIZE) {

printf("Deque is full.\n");

return;

}

for (int i = d->size; i >= 1; i--) {

d->arr[i] = d->arr[i-1];

}

d->arr[0] = elem;

d->size++;

}

void deque\_pop\_front(deque \*d) {

if (d->size == 0) {

printf("Дек пуст.\n");

return;

}

for (int i = 1; i < d->size; i++) {

d->arr[i-1] = d->arr[i];

}

d->size--;

}

void deque\_pop\_back(deque \*d) {

if (d->size == 0) {

printf("Дек пуст.\n");

return;

}

d->size--;

}

int deque\_sizeof(deque \*d) {

return d->size;

}

int deque\_front(deque \*d) {

return d->arr[0];

}

int deque\_back(deque \*d) {

return d->arr[d->size-1];

}

void deque\_print(const deque \*d) {

for (int i = 0; i < d->size; i++) {

printf("%d ", d->arr[i]);

}

printf("\n");

}

deque.h  
#ifndef DEQUE\_H

#define DEQUE\_H

#include <stdbool.h>

#define MAXSIZE 5

typedef struct deque {

int size;

int arr[MAXSIZE];

} deque;

deque \*deque\_create(deque \*);

bool deque\_isempty(deque \*);

void deque\_push\_back(deque \*, int);

void deque\_push\_front(deque \*, int);

void deque\_pop\_front(deque \*);

void deque\_pop\_back(deque \*);

int deque\_sizeof(deque \*);

int deque\_front(deque \*);

int deque\_back(deque \*);

void deque\_print(const deque \*);

#endif

main.c

#include <stdio.h>

#include <stdlib.h>

#include "deque.h"

deque \*concatenation(deque \*d1, deque \*d2) {

deque \*d = (deque\*)malloc(sizeof(deque));

\*d = \*d1;

int s1 = deque\_sizeof(d1);

int s2 = deque\_sizeof(d2);

if (s1 + s2 > MAXSIZE) {

printf("Произойдет переполнение.\n");

return NULL;

}

for (int i = 0; i < s2; i++) {

int f2 = deque\_front(d2);

deque\_push\_back(d, f2);

deque\_pop\_front(d2);

deque\_push\_back(d2, f2);

}

return d;

}

deque \*mass\_to\_deque(int \*numbers, int l, int r) {

deque \*d;

d = deque\_create(d);

for (int i = l; i <= r; i++) {

deque\_push\_back(d, numbers[i]);

}

return d;

}

deque \*qs(deque \*d, int left, int right) {

deque \*d1\_c;

d1\_c = deque\_create(d1\_c);

deque \*d2\_c;

d2\_c = deque\_create(d2\_c);

int numbers[MAXSIZE];

int sz = deque\_sizeof(d);

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

numbers[i] = deque\_back(d);

deque\_pop\_back(d);

}

deque \*res;

res = deque\_create(res);

if (left == right) {

deque\_push\_back(res, numbers[0]);

return res;

}

int base;

int l = left;

int r = right;

base = numbers[left];

while (left < right) {

while ((numbers[right] >= base) && (left < right)) {

right--;

}

if (left != right) {

numbers[left] = numbers[right];

left++;

}

while ((numbers[left] <= base) && (left < right)) {

left++;

}

if (left != right) {

numbers[right] = numbers[left];

right--;

}

}

numbers[left] = base;

base = left;

left = l;

right = r;

if (left < base) {

deque \*d1;

d1 = deque\_create(d1);

d1 = mass\_to\_deque(numbers, left, base-1);

//deque\_print(d1);

d1\_c = qs(d1, 0, base-1 - left);

//deque\_print(d1\_c);

}

if (right > base) {

deque \*d2;

d2 = deque\_create(d2);

d2 = mass\_to\_deque(numbers, base + 1, right);

d2\_c = qs(d2, 0, right - base - 1);

}

deque \*tmp;

tmp = deque\_create(tmp);

deque\_push\_back(tmp, numbers[base]);

res = concatenation(d1\_c, tmp);

res = concatenation(res, d2\_c);

return res;

}

void pr\_interface() {

printf("Input your command:\n");

printf("1. Create deq\n2. Empty\n3. Size\n4. Push back\n5. Push front\n6. Top left\n7. Top right\n8.Pop back\n9. Pop front\n10.Print\n11. Sort\n0. Exit\n");

}

int main() {

deque \*d;

pr\_interface();

while(true) {

int q;

scanf("%d", &q);

if (q == 1) {

d = deque\_create(d);

}

if (q == 2) {

bool ans = deque\_isempty(d);

if (ans) {

printf("Yes, the deque is empty.\n");

}

else {

printf("No, the deque is not empty.\n");

}

}

if (q == 3) {

int ans = deque\_sizeof(d);

printf("The size of deque is equal to %d.\n", ans);

}

if (q == 4) {

printf("Input num, that is going to be pushed\n");

int n;

scanf("%d", &n);

deque\_push\_back(d, n);

}

if (q == 5) {

printf("Input num, that is going to be pushed\n");

int n;

scanf("%d", &n);

deque\_push\_front(d, n);

}

if (q == 6) {

int ans = deque\_front(d);

printf("The front elem is %d\n", ans);

}

if (q == 7) {

int ans = deque\_back(d);

printf("The back elem is %d\n", ans);

}

if (q == 8) {

deque\_pop\_back(d);

}

if (q == 9) {

deque\_pop\_front(d);

}

if (q == 10) {

deque\_print(d);

}

if (q == 11) {

d = qs(d, 0, deque\_sizeof(d)-1);

}

if (q == 0) {

return 0;

}

}

}

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. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  | или |  |  |  |  |  |
|  | дом. |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Замечания автора** по существу работы:

Замечания отсутствуют.

1. **Выводы**

Я научилась писать свои собственные структуры данных, программировать программы модульно и собирать их с помощью утилиты make

Недочёты при выполнении задания могут быть устранены следующим образом:

Недочеты отсутствуют.

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