Отчёт по лабораторной работе №

по курсу «Языки и методы программирования».

Выполнил студент группы М8О-111Б-23: Воробьев Глеб Янович № по списку 5.

Контакты: koshastet13@gmail.com

Работа выполнена: «28» февраля 2024 г.

Преподаватель: каф. 806 Никулин Сергей Петрович

Входной контроль знаний с оценкой: _____

Отчет сдан «29» февраля 2024 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 - поиск и удаление максимального и минимального элемента, метод - сортировка линейным выбором

4. Оборудование:
Процессор AMD Ryzen 5 7640HS.
ОП 16 ГБ.
SSD 512 ГБ.
Монитор 2560x1600~165Hz.

5. Программное обеспечение:
Операционная система семейства Unix.
Наименование Ubuntu версия 22.04.3.
Интерпретатор команд GNU bash версия 6.2.0.
Система программирования -.
Редактор текстов Visual Studio Code.

6. Идея, метод, алгоритм решения задачи:
Код реализует поиск максимального и минимального элемента в деке, который представлен как кольцевой буфер. Реализация дека представлена в файлах udt.c и udt.h. RemoveMin для удаления минимального элемента из дека и selectionSort для сортировки дека методом сортировки выбором.которая представлена в файле main.c
содержание main.c:

1. FindMin(deque *d): Начинается с сохранения первого элемента дека как текущего минимального. Проходит через все элементы дека, перемещая их с "головы" на "хвост" и сравнивая их с текущим минимальным значением. Если новый элемент меньше текущего минимального, обновляет минимальное значение и индекс минимального элемента. После прохода по всему деку, проходит по деку второй раз, удаляя минимальный элемент (не добавляя его обратно). Возвращает значение минимального элемента.
2. selectionSort(deque *d): Создает новый пустой дек, который будет использоваться для хранения отсортированных элементов. Пока исходный дек не пуст, вызывает RemoveMin для нахождения и удаления минимального элемента и добавляет этот элемент в конец нового отсортированного дека. После того как все элементы перемещены в отсортированный дек, перекладывает их обратно в исходный дек (этот шаг на самом деле не нужен, если мы хотим сохранить отсортированный дек). Возвращает новый отсортированный дек.
3. printDeque(deque *d): Функция печати элементов дека на экран. Выводит все элементы дека с начала до конца, учитывая кольцевую структуру дека и начальный индекс head.
4. main(): Главная функция, которая предоставляет пользователю меню для взаимодействия с деком. В меню есть опции для добавления элементов в голову или хвост дека, печати минимального и максимального элемента, удаления минимального и максимального элемента, проверки размера и пустоты дека, а также для печати всех элементов дека.

Для компиляции используется утилита make

7. Сценарий выполнения работы:

makefile

```
all:
	gcc main.c udt.c
```

udt.h

```
#ifndef _UDT_H_
#define _UDT_H_

#define POOL_SIZE 100

typedef struct {
	int head;
	int tail;
	int size;
	int data[POOL_SIZE];
} deque;

void PushHeadDeque(deque *d, int key);
void PushTailDeque(deque *d, int key);
void PopHeadDeque(deque *d);
void PopTailDeque(deque *d);
int TopHeadDeque(deque *d);
int TopTailDeque(deque *d);
void CreateDeque(deque *d);
void deleteDeque(deque *d);
int getSizeDeque(deque *d);
int isEmptyDeque(deque *d);

#endif
```

udt.c

```
#include <stdio.h>

#include "udt.h"

void CreateDeque(deque *d)
{
	d->head = 0;
	d->tail = 0;
	d->size = 0;
}

int isEmptyDeque(deque *d)
{
	return d->size == 0;
}
```

```c
int getSizeDeque(deque *d)
{
    return d->size;
}

void PushHeadDeque(deque *d, int key)
{
    if (d->size == POOL_SIZE) // Проверяем, не заполнен ли дек.
        return; // Если дек полон, мы не можем добавить новый элемент.

    // Если дек пуст, добавляем элемент в начало, не меняем индекс head.
    if (d->size == 0) {
        d->data[d->head] = key; // Вставляем элемент в текущую позицию
head, которая равна 0.
    } else {
        // Уменьшаем head на 1 с учетом кольцевой структуры.
        d->head = (d->head - 1 + POOL_SIZE) % POOL_SIZE;
        d->data[d->head] = key; // Вставляем элемент в позицию перед
текущим head.
    }
    d->size++; // Увеличиваем размер дека.
}


void PushTailDeque(deque *d, int key)
{
    if (d->size == POOL_SIZE) // Проверяем, не заполнен ли дек.
        return; // Если дек полон, мы не можем добавить новый элемент.

    // Вычисляем индекс для вставки элемента в конец дека.
    int index = (d->head + d->size) % POOL_SIZE;
    d->data[index] = key; // Вставляем элемент.
    d->size++; // Увеличиваем размер дека.
    d->tail = index; // Обновляем индекс tail.
}


void PopHeadDeque(deque *d)
{
    if (!d->size)
        return;
    d->head++;
    d->head %= POOL_SIZE;
    d->size--;
}

void PopTailDeque(deque *d)
{
```

```
    if (!d->size)
        return;
    d->tail = (d->tail - 1 + POOL_SIZE) % POOL_SIZE;
    d->size--;
}

int TopHeadDeque(deque *d)
{
    if (d->size)
        return d->data[d->head];
    else
        return '\0';
}

int TopTailDeque(deque *d)
{
    if (d->size)
        return d->data[d->tail];
    else
        return '\0';
}

void deleteDeque(deque *d)
{
    d->head = 0;
    d->tail = 0;
    d->size = 0;
}
```

main.c

```
#include <stdio.h>
#include "udt.h"


int FindMin(deque *d)
{
    int min_el = TopHeadDeque(d);
    int cur_el;
    int min_ind = 0;

    for (int i=0; i < d->size; ++i) {
        cur_el = TopHeadDeque(d);
        PopHeadDeque(d);
        PushTailDeque(d, cur_el);

        if (cur_el < min_el) {
            min_el = cur_el;
            min_ind = i;
```

```
        }
    }

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

        if (min_ind != i)
            PushTailDeque(d, cur_el);
    }

    return min_el;
}


void selectionSort(deque *d)
{
    deque sorted_d;
    CreateDeque(&sorted_d);
    int min_el;

    while (!isEmptyDeque(d))
    {
        min_el = FindMin(d);
        PushTailDeque(&sorted_d, min_el);
    }

    while (!isEmptyDeque(&sorted_d)) {
        min_el = TopHeadDeque(&sorted_d);
        PopHeadDeque(&sorted_d);
        PushTailDeque(d, min_el);
    }

}


void printDeque(deque *d)
{
    int i, index;
    for (i = 0; i < d->size; i++) {
        index = (d->head + i) % POOL_SIZE;
        printf("%d ", d->data[index]);
    }
    printf("\n");
}


int main()
```

```
{
    deque deck;

    CreateDeque(&deck);

    int state, key;

    do {
        printf("Choose the option: 1-push to head, 2-push to tail, 3 - sort
deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print
deque\n");
        scanf("%d", &state);
        if (state == 1) {
            printf("Type value: ");
            scanf(" %d", &key);
            PushHeadDeque(&deck, key);
            printf("%d added to head\n", key);
        } else if (state == 2) {
            printf("Type value: ");
            scanf(" %d", &key);
            PushTailDeque(&deck, key);
            printf("%d added to tail\n", key);
        } else if (state == 3) {
            if (!isEmptyDeque(&deck)) {
                selectionSort(&deck);
                printDeque(&deck);
            } else {
                printf("Deque is empty\n");
            }
         } else if (state == 4) {
            PopHeadDeque(&deck);
            printf("Delete head\n");
        } else if (state == 5) {
            PopTailDeque(&deck);
            printf("Delete tail\n");
        } else if (state == 6) {
            int size = getSizeDeque(&deck);
            printf("Size is %d\n", size);
        } else if (state == 7) {
            int empty = isEmptyDeque(&deck);
            printf("Deque is %s\n", empty ? "empty" : "not empty");
        } else if (state == 8) {
            printf("Deque:\n");
            printDeque(&deck);
        }
    } while(state);

    deleteDeque(&deck);
```

```
    return 0;
}
```

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

8. **Распечатка протокола:**

```
xxxkoshaster@YES-MAN:~/Documents/Zayks/lb6$ make
gcc main.c udt.c
xxxkoshaster@YES-MAN:~/Documents/Zayks/lb6$ ./a.out
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
1
Type value: 10
10 added to head
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
2
Type value: 20
20 added to tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
2
Type value: 30
30 added to tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
8
Deque:
10 20 30
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
4
Delete head
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
8
Deque:
20 30
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
5
Delete tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
8
Deque:
20
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
1
Type value: 89
89 added to head
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
2
Type value: 32
32 added to tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
8
Deque:
89 20 32
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
3
20 32 89
```

Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
4
Delete head
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
8
Deque:
32 89
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
6
Size is 2
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
7
Deque is not empty
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
4
Delete head
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
4
Delete head
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
7
Deque is empty
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
6
Size is 0
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
8
Deque:

Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
1
Type value: 21
21 added to head
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
1
Type value: 30
30 added to head
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
2
Type value: 9
9 added to tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
2
Type value: 9
9 added to tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
8
Deque:
30 21 9 9
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
3
9 9 21 30
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
4
Delete head
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
8

Deque:
9 21 30
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
2
Type value: -12
-12 added to tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
2
Type value: 0
0 added to tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
2
Type value: -1
-1 added to tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
8
Deque:
9 21 30 -12 0 -1
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
3
-12 -1 0 9 21 30
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
5
Delete tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
8
Deque:
-12 -1 0 9 21
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
6
Size is 5
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
4
Delete head
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
8
Deque:
-1 0 9 21
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
5
Delete tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
7
Deque is not empty
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
6
Size is 3
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
5
Delete tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
6
Size is 2
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
5
Delete tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque

6
Size is 1
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
5
Delete tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
7
Deque is empty
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
8
Deque:

Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
1
Type value: 30
30 added to head
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
2
Type value: 20
20 added to tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
2
Type value: 20
20 added to tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
2
Type value: 30
30 added to tail
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
8
Deque:
30 20 20 30
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
3
20 20 30 30
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
6
Size is 4
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
4
Delete head
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
8
Deque:
20 30 30
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
6
Size is 3
Choose the option: 1-push to head, 2-push to tail, 3 - sort deque, 4 - pop head, 5 - pop tail, 6 - size, 7 - check if empty, 8 - print deque
0

9. Дневник отладки

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

10. Замечания автора:

По существу работы: замечания отсутствуют.

11. Выводы:

Подпись студента _____