**Отчёт по лабораторной работе № 26**

по курсу Алгоритмы и структуры данных

Студент группы М8О-103Б-23, Махмутов Дэниз Ирикович, № по списку 17

Контакты www, e-mail, icq, skype e-mail: lustmodesta@gmai.com

Работа выполнена «\_15» мая 2024 г.

Преподаватель: Никулин Сергей Петрович каф. 806

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

Отчет сдан «16» мая 2024 г., итоговая оценка \_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_

1. **Тема:** Абстрактные типы данных. Модули. Make.
2. **Цель работы:**

Составить программу-модуль на языке Си, реализующую абстрактный тип данных, программу-модуль, содержащую функцию его сортировки определённым методом, указанном в варианте

1. **Задание (***вариант №17***):**

#### Дек - Поиск и удаление максимального элемента - Сортировка линейным выбором

**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 5600x c ОП 16 Гб, НМД 1024 Гб. Монитор 1920×1080 пикс.

Другие устройства –

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

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 Ubuntu версия 20.04.3 LTS

Интерпретатор команд GNOME Terminal версия 3.36.2

Система программирования – версия –

Редактор текстов emacs версия 3.24.14

Прикладные системы и программы –

Местонахождение и имена файлов программ и данных на домашнем компьютере –

1. **Идея, метод, алгоритм**  решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)
2. Реализуем четыре файла:
   1. Файл stack.h, в котором описаны такие функции как:
      1. udtCreate(Udt udt, const int capacity)
   * Создает дек (двустороннюю очередь) с заданной емкостью.
   * Инициализирует структуру Udt, выделяя память для массива \_data, устанавливает начальные значения для полей \_capacity, \_size, \_first и \_last.
3. udtPushFront(Udt udt, const UDT\_TYPE value)
   * Добавляет элемент в начало дека.
   * Рассчитывает позицию для вставки нового элемента, обновляет соответствующие индексы и увеличивает размер дека.
4. udtPushBack(Udt udt, const UDT\_TYPE value)
   * Добавляет элемент в конец дека.
   * Рассчитывает позицию для вставки нового элемента, обновляет соответствующие индексы и увеличивает размер дека.
5. udtPopFront(Udt udt)
   * Удаляет элемент из начала дека.
   * Обновляет индексы и уменьшает размер дека. Если дек пуст, ничего не делает.
6. udtPopBack(Udt udt)
   * Удаляет элемент из конца дека.
   * Обновляет индексы и уменьшает размер дека. Если дек пуст, ничего не делает.
7. udtTopFront(const Udt udt)
   * Возвращает значение первого элемента дека.
   * Не изменяет дек.
8. udtTopBack(const Udt udt)
   * Возвращает значение последнего элемента дека.
   * Не изменяет дек.
9. udtSize(const Udt udt)
   * Возвращает количество элементов в деке.
10. udtCapacity(const Udt udt)
    * Возвращает емкость дека.
11. udtEmpty(const Udt udt)
    * Проверяет, пуст ли дек.
    * Возвращает 1, если дек пуст, иначе 0.
12. udtPrint(Udt udt)
    * Печатает содержимое дека в виде таблицы.
    * Форматирует вывод, чтобы показать номер элемента, ключ и строку для каждого элемента дека.
13. udtDestroy(Udt udt)
    * Освобождает память, выделенную под дек, и сбрасывает все поля структуры Udt в начальное состояние.
14. task(Udt udt)
    * Находит и удаляет строку максимальной длины в деке.
    * Проходит по всем элементам дека, определяет элемент с максимальной длиной строки, перемещает его в конец и удаляет.
    1. Файл sort.h
    2. Обменивает содержимое двух деков.
    3. Использует временную переменную для хранения одного из деков, затем осуществляет обмен.
    4. Реализует сортировку дека методом выбора.
    5. Создает два временных дека.
    6. Пока дек не пуст, перемещает элементы между деком и временными деками, сортируя их по ключам.
    7. Использует временные деки для сохранения порядка и обмена элементов.
    8. После сортировки уничтожает временные деки и выводит сообщение "sorted".
    9. Файл main.c Программа предоставляет пользователю несколько опций для работы с деком, таких как добавление и удаление элементов, сортировка элементов, определение размера дека и другие.
    10. Makefile
15. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].

**udt.h**

#ifndef udt\_h

#define udt\_h

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct \_Item

{

int \_key;

char \_str[31];

} Item;

typedef Item UDT\_TYPE;

typedef struct \_Udt

{

UDT\_TYPE \*\_data;

int \_capacity;

int \_size;

int \_first;

int \_last;

} Udt;

void udtCreate(Udt \*udt, const int capacity);

int udtPushFront(Udt \*udt, const UDT\_TYPE value);

int udtPushBack(Udt \*udt, const UDT\_TYPE value);

void udtPopFront(Udt \*udt);

void udtPopBack(Udt \*udt);

UDT\_TYPE udtTopFront(const Udt \*udt);

UDT\_TYPE udtTopBack(const Udt \*udt);

int udtSize(const Udt \*udt);

int udtCapacity(const Udt \*udt);

int udtEmpty(const Udt \*udt);

void udtPrint(Udt \*udt);

void udtDestroy(Udt \*udt);

void task(Udt \*udt);

#endif

**udt.c**

#include "udt.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// создать дек

void udtCreate(Udt \*udt, const int capacity)

{

    int i;

    UDT\_TYPE item;

    item.\_key = 0;

    item.\_str[0] = '\0';

    if (capacity <= 0)

        return;

    udt->\_data = (UDT\_TYPE \*)malloc(sizeof(UDT\_TYPE) \* capacity);

    for (i = 0; i < capacity; i++)

        udt->\_data[i] = item;

    udt->\_capacity = capacity;

    udt->\_size = 0;

    udt->\_first = (capacity == 1 ? 0 : 1);

    udt->\_last = 0;

}

//добавить элемент в начало

int udtPushFront(Udt \*udt, const UDT\_TYPE value)

{

    const int pos = (udt->\_first + udt->\_capacity - 1) % udt->\_capacity;

    if (udt->\_size == udt->\_capacity)

        return 0;

    udt->\_data[pos] = value;

    udt->\_first = pos;

    udt->\_size++;

    return 1;

}

// добавить элемент в конец

int udtPushBack(Udt \*udt, const UDT\_TYPE value)

{

    const int pos = (udt->\_last + udt->\_capacity + 1) % udt->\_capacity;

    if (udt->\_size == udt->\_capacity)

        return 0;

    udt->\_data[pos] = value;

    udt->\_last = pos;

    udt->\_size++;

    return 1;

}

// удалить элемент из начала

void udtPopFront(Udt \*udt)

{

    const int pos = (udt->\_first + udt->\_capacity + 1) % udt->\_capacity;

    UDT\_TYPE item;

    item.\_key = 0;

    item.\_str[0] = '\0';

    if (udt->\_size == 0)

        return;

    udt->\_data[udt->\_first] = item;

    udt->\_first = pos;

    udt->\_size--;

}

// удалить элемент из конца

void udtPopBack(Udt \*udt)

{

    const int pos = (udt->\_last + udt->\_capacity - 1) % udt->\_capacity;

    UDT\_TYPE item;

    item.\_key = 0;

    item.\_str[0] = '\0';

    if (udt->\_size == 0)

        return;

    udt->\_data[udt->\_last] = item;

    udt->\_last = pos;

    udt->\_size--;

}

// значение первого элемента

UDT\_TYPE udtTopFront(const Udt \*udt)

{

    return udt->\_data[udt->\_first];

}

// значение последнего элемента

UDT\_TYPE udtTopBack(const Udt \*udt)

{

    return udt->\_data[udt->\_last];

}

// количество элементов в деке

int udtSize(const Udt \*udt)

{

    return udt->\_size;

}

// размер дека

int udtCapacity(const Udt \*udt)

{

    return udt->\_capacity;

}

int udtEmpty(const Udt \*udt)

{

    return udt->\_size == 0;

}

// распечатать дек

void udtPrint(Udt \*udt)

{

    int i;

    Item item;

    printf("+--------+-----------+------------------------------+\n");

    printf("| Number |    Key    |            String            |\n");

    printf("+--------+-----------+------------------------------+\n");

    for (i = 0; i < udtSize(udt); i++)

    {

        item = udt->\_data[(i + udt->\_first) % udt->\_capacity];

        printf("|%8d|%11d|%30s|\n", i + 1, item.\_key, item.\_str);

    }

    printf("+--------+-----------+------------------------------+\n");

}

// удалить дек

void udtDestroy(Udt \*udt)

{

    if (udt->\_data != NULL)

    {

        free(udt->\_data);

        udt->\_data = NULL;

    }

    udt->\_capacity = 0;

    udt->\_size = 0;

    udt->\_first = 0;

    udt->\_last = 0;

}

// поиск и удаление максимального элемента

void task(Udt \*udt)

{

    if (!udt || !udt->\_data || udtEmpty(udt))

        return;

    Udt tmp;

    udtCreate(&tmp, udtCapacity(udt));

    int max\_length = 0;

    UDT\_TYPE max\_item;

    UDT\_TYPE item;

    // Пройти по всем элементам и найти максимальный элемент

    int size = udtSize(udt);

    for (int i = 0; i < size; i++)

    {

        item = udtTopFront(udt);

        udtPopFront(udt);

        udtPushBack(&tmp, item);

        if (strlen(item.\_str) > max\_length)

        {

            max\_length = strlen(item.\_str);

            max\_item = item;

        }

    }

    for (int i = 0; i < size; i++)

    {

        item = udtTopFront(&tmp);

        udtPopFront(&tmp);

        if (strcmp(item.\_str, max\_item.\_str) != 0)

        {

            udtPushBack(udt, item);

        }

    }

    udtDestroy(&tmp);

    printf("deleted\n");

}

**sort.h**

#ifndef UDT\_SORT\_H

#define UDT\_SORT\_H

#include "udt.h"

void udtSwap(Udt \*udt1, Udt \*udt2);

void udtSelectionSort(Udt \*udt);

#endif

**sort.c**

**#include "sort.h"**

// обмен местами двух элементов

void udtSwap(Udt \*udt1, Udt \*udt2)

{

    Udt tmp;

    tmp = \*udt1;

    \*udt1 = \*udt2;

    \*udt2 = tmp;

}

// сортировка линейным образом

void udtSelectionSort(Udt \*udt)

{

    const int cap = udtCapacity(udt);

    Udt sorted, tmp;

    UDT\_TYPE item;

    if (cap < 2)

        return;

    udtCreate(&sorted, cap);

    udtCreate(&tmp, cap);

    while (!udtEmpty(udt))

    {

        udtPushFront(&tmp, udtTopFront(udt)); // добавляем первый элемент udt в начало tmp

        udtPopFront(udt);                     // удаляем первый элемент udt

        //

        while (!udtEmpty(udt))

        {

            item = udtTopFront(udt); // берем значение первого элемента udt

            udtPopFront(udt); // удаляем элемент из начала udt

            if (item.\_key < udtTopFront(&tmp).\_key) // если ключ первого элемента udt   <   ключа первого элемента tmp

                udtPushFront(&tmp, item);           // добавляем item в начало tmp

            else

                udtPushBack(&tmp, item); // иначе добавляем item в конец tmp

        }

        //

        udtPushBack(&sorted, udtTopFront(&tmp)); // берем значение первого элемента tmp и добавляем в конец sorted

        udtPopFront(&tmp);                       // удаляем элемент из начала tmp

        udtSwap(udt, &tmp); // обмен элементов между деками udt и tmp

    }

    udtSwap(udt, &sorted); // обмен элементов между деками udt и sorted

    udtDestroy(&sorted);

    udtDestroy(&tmp);

    printf("sorted\n");

}

**main.c**

**#include <stdio.h>**

#include "sort.h"

// ДЕК - ПОИСК И УДАЛЕНИЕ МАКСИМАЛЬНОГО ЭЛЕМЕНТА - СОРТИРОВКА ЛИНЕЙНЫМ ОБРАЗОМ

void getLine(char \*str, const int size)

{

    int cnt = 0, ch;

    while ((ch = getchar()) != '\n' && cnt < size - 1)

        str[cnt++] = ch;

    str[cnt] = '\0';

}

void print\_help()

{

    printf("\n1) Добавить элемент слева\n");

    printf("2) Добавить элемент справа\n");

    printf("3) Удалить элемент слева\n");

    printf("4) Удалить элемент справа\n");

    printf("5) Размер дека\n");

    printf("6) Сортировка\n");

    printf("7) Поиск и удаление строки максимальной длины\n");

    printf("8) Печать\n");

    printf("9) Выход\n\n");

}

int main(void)

{

    system("chcp 1251 > nul");

    print\_help();

    const int N = 10; // общий размер дека

    char tmpCh;

    Udt udt;

    Item item;

    udtCreate(&udt, N);

    char c;

    while ((c = getchar()) != EOF)

    {

        if (c == '\n')

            continue;

        switch (c)

        {

        case '1':

        {

            printf("Введите ключ (только числа): ");

            if (scanf("%d", &item.\_key))

            {

                scanf("%c", &tmpCh);

                printf("Введите Строку: ");

                getLine(item.\_str, sizeof(item.\_str));

                if (udtPushFront(&udt, item))

                    printf("\nЭлемент с ключом %d и строкой '%s' добавлен\n", item.\_key, item.\_str);

                else

                    printf("Дек полон\n");

                print\_help();

                break;

            }

            else

            {

                printf("\nerror\n");

                print\_help();

                break;

            }

        }

        case '2':

        {

            printf("Введите ключ (только числа): ");

            if (scanf("%d", &item.\_key))

            {

                scanf("%c", &tmpCh);

                printf("Введите Строку: ");

                getLine(item.\_str, sizeof(item.\_str));

                if (udtPushBack(&udt, item))

                    printf("\nЭлемент с ключом %d и строкой '%s' добавлен\n", item.\_key, item.\_str);

                else

                    printf("Дек полон\n");

                print\_help();

                break;

            }

            else

            {

                printf("\nerror\n");

                print\_help();

                break;

            }

        }

        case '3':

        {

            if (udtSize(&udt) > 0)

            {

                item = udtTopFront(&udt);

                udtPopFront(&udt);

                printf("\nЭлемент с ключом %d и строкой '%s' удален\n", item.\_key, item.\_str);

            }

            else

                printf("Дек пуст\n");

            print\_help();

            break;

        }

        case '4':

        {

            if (udtSize(&udt) > 0)

            {

                item = udtTopBack(&udt);

                udtPopBack(&udt);

                printf("\nЭлемент с ключом %d и строкой '%s' удален\n", item.\_key, item.\_str);

            }

            else

                printf("Дек пуст\n");

            print\_help();

            break;

        }

        case '5':

        {

            printf("Размер дека: %d (Общий размер: %d)\n", udtSize(&udt), N);

            print\_help();

            break;

        }

        case '6':

        {

            udtSelectionSort(&udt);

            print\_help();

            break;

        }

        case '7':

        {

            task(&udt);

            print\_help();

            break;

        }

        case '8':

        {

            if (udtSize(&udt) > 0)

            {

                printf("\nДек:\n");

                udtPrint(&udt);

            }

            else

                printf("\nДек пуст\n");

            print\_help();

            break;

        }

        case '9':

            return 0;

        default:

        {

            break;

        }

        }

    }

    udtDestroy(&udt);

    return 0;

}

Makefile

TARGET = my\_program

CC = gcc

CFLAGS = -Wall -I.

SRCS = main.c sort.c udt.c

OBJS = $(SRCS:.c=.o)

$(TARGET): $(OBJS)

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

%.o: %.c

    $(CC) $(CFLAGS) -c $< -o $@

clean:

    rm -f $(OBJS) $(TARGET)

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

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

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

unix@DESKTOP-MPQDBS2:~/labs$ make

gcc -Wall -I. -c udt.c -o udt.o

gcc -Wall -I. -o my\_program main.o sort.o udt.o

unix@DESKTOP-MPQDBS2:~/labs$ ./my\_program

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

1

Введите ключ (только числа): 10

Введите Строку: 15

Элемент с ключом 10 и строкой '15' добавлен

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

2

Введите ключ (только числа): 10

Введите Строку: 15

Элемент с ключом 10 и строкой '15' добавлен

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

5

Размер дека: 2 (Общий размер: 10)

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

6

sorted

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

8

Дек:

+--------+-----------+------------------------------+

| Number | Key | String |

+--------+-----------+------------------------------+

| 1| 10| 15|

| 2| 10| 15|

+--------+-----------+------------------------------+

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

1

Введите ключ (только числа): 11

Введите Строку: 15

Элемент с ключом 11 и строкой '15' добавлен

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

1

Введите ключ (только числа): 13

Введите Строку: 13

Элемент с ключом 13 и строкой '13' добавлен

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

2

Введите ключ (только числа): 8

Введите Строку: 13

Элемент с ключом 8 и строкой '13' добавлен

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

2

Введите ключ (только числа): 14

Введите Строку: 13

Элемент с ключом 14 и строкой '13' добавлен

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

8

Дек:

+--------+-----------+------------------------------+

| Number | Key | String |

+--------+-----------+------------------------------+

| 1| 13| 13|

| 2| 11| 15|

| 3| 10| 15|

| 4| 10| 15|

| 5| 8| 13|

| 6| 14| 13|

+--------+-----------+------------------------------+

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

6

sorted

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

8

Дек:

+--------+-----------+------------------------------+

| Number | Key | String |

+--------+-----------+------------------------------+

| 1| 8| 13|

| 2| 10| 15|

| 3| 10| 15|

| 4| 11| 15|

| 5| 13| 13|

| 6| 14| 13|

+--------+-----------+------------------------------+

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

7

deleted

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

8

Дек:

+--------+-----------+------------------------------+

| Number | Key | String |

+--------+-----------+------------------------------+

| 1| 10| 15|

| 2| 10| 15|

| 3| 11| 15|

+--------+-----------+------------------------------+

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

1

Введите ключ (только числа): 16

Введите Строку: 1

Элемент с ключом 16 и строкой '1' добавлен

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

2

Введите ключ (только числа): 100

Введите Строку: 10

Элемент с ключом 100 и строкой '10' добавлен

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

8

Дек:

+--------+-----------+------------------------------+

| Number | Key | String |

+--------+-----------+------------------------------+

| 1| 16| 1|

| 2| 10| 15|

| 3| 10| 15|

| 4| 11| 15|

| 5| 100| 10|

+--------+-----------+------------------------------+

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

7

deleted

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

8

Дек:

+--------+-----------+------------------------------+

| Number | Key | String |

+--------+-----------+------------------------------+

| 1| 16| 1|

| 2| 100| 10|

+--------+-----------+------------------------------+

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

1

Введите ключ (только числа): 10

Введите Строку: 100

Элемент с ключом 10 и строкой '100' добавлен

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

7

deleted

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

8

Дек:

+--------+-----------+------------------------------+

| Number | Key | String |

+--------+-----------+------------------------------+

| 1| 16| 1|

| 2| 100| 10|

+--------+-----------+------------------------------+

1) Добавить элемент слева

2) Добавить элемент справа

3) Удалить элемент слева

4) Удалить элемент справа

5) Размер дека

6) Сортировка

7) Поиск и удаление строки максимальной длины

8) Печать

9) Выход

9

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 |
|  |  |  |  |  |  |  |

1. **Замечания автора**
2. **Выводы**

За время выполнения ЛР я научился работать с абстрактными типами данных, реализовал стек и метод её сортировки.

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