Отчет по лабораторной работе №25-26
по курсу Практикум программирования

Студент группы М8О-111Б-23 Бугренков Владимир Петрович,
№ по списку 4, Контакты e-mail: vladimir.bugrenkov@yandex.ru
Работа выполнена: «3» мая 2023 г.
Преподаватель: доцент каф. 806 Никулин Сергей Петрович
Отчет сдан « » _____20 ___ 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 – Очередь. Процедура – поиск и удаление минимального элемента. Метод – сортировка линейным выбором.
4. **Оборудование** (лабораторное):

ЭВМ _____, процессор _____. имя узла сети_____с ОП_____ Мб, НМД ____ Мб.
Терминал _____адрес _____. Принтер_____
Другие устройства
_____

Оборудование ПЭВМ студента, если использовалось:

Процессор _AMD Ryzen 5_ с ОП __16_ Гб НМД ___512 Гб. Монитор _1920x1080~60Hz
Другие устройства
_____

5. **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 _Linux_, наименование _Ubuntu_ версия 22.04.2
интерпретатор команд _GNU bash_ версия 5.1.16.
Система программирования C.
Редактор текстов emacs версия 29.1
Утилиты операционной системы
_____
Прикладные системы и программы Emacs

Местонахождение и имена файлов программ и данных на домашнем компьютере /home/

6. **Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица]

Для выполнения данного задания требуются структуры data_type и queue:

```
typedef struct data_type {
   int key;
   int value;
} data_type;

typedef struct queue {
   int first;
   int size;
   data_type data[POOL_SIZE];
} queue;
```

Функции:

1) Create: инициализирует очередь.
2) Empty: проверка на пустоту.
3) Size: размер очереди.
4) Push: добавление элемента в конец очереди.
5) Pop: удаление первого элемента очереди.
6) Top: возвращает значение первого элемента.
7) Print: печать очереди.
8) Destroy: удаляет очередь.
9) min_pop: поиск и удаление минимального элемента
10) qsort: сортировка линейным выбором по ключам

Сортировка линейным выбором над очередью (qsort) реализована следующим образом:

1. Создаем вспомогательную очередь
2. Ищем в оригинальной очереди минимальный ключ
3. Добавляем минимальный ключ-элемент в вспомогательную очередь
4. Удаляем из оригинальной очереди минимальный ключ
5. Повторяем 2-3-4 пока оригинальная очередь не опустеет
6. Меняем оригинальную очередь и временную местами
7. Получаем отсортированную по возрастанию ключей очередь

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

Код программы:

main.c:

```
#include <stdio.h>
#include <stdbool.h>
#include "queue.c"
#include "qsort.c"

void menu() {
   puts("-------------------------------------------");
   puts("  1) Добавить элемент в очередь.");
```

```
    puts("| 2) Удаление первого элемента в очереди.");
    puts("| 3) Размер очереди.");
    puts("| 4) Удаление минимального элемента.");
    puts("| 5) Печать очереди.");
    puts("| 6) Сортировка очереди.");
    puts("| 0) Выход.");
    puts("-----------------------------------------");
    printf("▷▷▷ ");
}

int main() {
    queue q;
    data_type t;
    Create(&q);
    char c;
    menu();
    scanf("%c", &c);
    while (c != '0') {
        switch (c) {
            case '1':
                printf("Введите элемент(ключ - значение):\n▷▷▷ ");
                scanf("%d - %d", &t.key, &t.value);
                Push(&q, t);
                break;

            case '2':
                if (!Empty(&q)) {
                    printf("Элемент <%d - %d> удалён.\n", Top(&q).key, Top(&q).value);
                    Pop(&q);
                } else {
                    puts("Очередь пуста!");
                }
                break;

            case '3':
                printf("Размер очереди: %d", Size(&q));
                break;

            case '4':
                q = min_pop(&q);
                puts("Минимальный элемент был удален");
                break;

            case '5':
                Print(&q);
                break;

            case '6':
                q = qsort(&q);
                puts("Очередь отсортирована!");
                break;

            default:
                puts("Неизвестное значение!");
                break;
        }
        menu();
        scanf("\n%c", &c);
    }
    Destroy(&q);
    puts("Программа завершена!");
}
```

queue.h:

```
#ifndef _QUEUE_H_
#define _QUEUE_H_
```

```
#include <stdbool.h>

#define POOL_SIZE 100

typedef struct data_type {
   int key;
   int value;
} data_type;

typedef struct queue {
   int first;
   int size;
   data_type data[POOL_SIZE];
} queue;

void Create(queue *q);

bool Empty(queue *q);

int Size(queue *q);

bool Push(queue *q, const data_type t);

bool Pop(queue *q);

void Print(queue *q);

data_type Top(const queue *q);

void Destroy(queue *q);

#endif
```

queue.c:

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

void Create(queue *q) {
   q->first = 0;
   q->size = 0;
}

bool Empty(queue *q) {
   return q->size == 0;
}

int Size(queue *q) {
   return q->size;
}

bool Push(queue *q, const data_type t) {
   if (q->size == POOL_SIZE)
      return false;
   q->data[(q->first + q->size++) % POOL_SIZE] = t;
   return true;
}

bool Pop(queue *q) {
   if (!q->size)
      return false;
   q->first++;
```

```
    q->first %= POOL_SIZE;
    q->size--;
    return true;
}

data_type Top(const queue *q) {
    if (q->size)
        return q->data[q->first];
}

void Print(queue *q) {
    queue cur = *q;
    putchar('[');
    while (!Empty(&cur)) {
        printf(" %d:%d", Top(&cur).key, Top(&cur).value);
        Pop(&cur);
    }
    puts(" ]");
}

void Destroy(queue *q) {
    q->size = 0;
}
```

qsort.c:

```
#include "queue.h"
#include "stdio.h"


int min_key_search(queue *q) {
    //   Нашли минимальный элемент
    queue cur = *q;
    int min = 2147483647;

    while (!Empty(&cur)) {
        if (min > Top(&cur).key)
            min = Top(&cur).key;
        Pop(&cur);
    }
    return min;
}

int min_value_search(queue *q) {
    //   Нашли минимальный элемент
    queue cur = *q;
    int min = 2147483647;

    while (!Empty(&cur)) {
        if (min > Top(&cur).value)
            min = Top(&cur).value;
        Pop(&cur);
    }
    return min;
}

queue minimal_key_search_and_pop(queue *q) {
    if (Empty(q))
        return *q;
    if (Size(q) == 1)
        return *q;

    int min = min_value_search(q);
```

```
    queue cur = *q;
    queue new_temp_que;
    Create(&new_temp_que);


    printf("До: ");
    Print(&cur);


    int flag = 0;

    while (!Empty(&cur)) {
        if ((min == Top(&cur).value) && (flag == 0)) {
            flag = 1;
        } else {
            Push(&new_temp_que, Top(&cur));
        }
        Pop(&cur);
    }
    return new_temp_que;

}

queue min_pop(queue *q) {
    if (Empty(q))
        return *q;
    if (Size(q) == 1) {
        return *q;
    }
    *q = minimal_key_search_and_pop(q);
    printf("После: ");
    Print(q);
    return *q;
}


// линейная сортировка очереди
// ищем минимальный элемент, добавляем его во временную очередь и удаляем из оригинальной. Повторяем так пока оригинальная очередь не опустеет
queue qsort(queue *q) {
    if (Empty(q))
        return *q;
    if (Size(q) == 1) {
        return *q;
    }
    printf("ДО: ");
    Print(q);

    queue cur = *q;

    queue help_que;
    Create(&help_que);
    const int size_que = Size(&cur);

    int min_value;
    for (int i = 0; i < size_que; i++) {
        // Ищем минимальный элемент в очереди
        min_value = min_key_search(&cur);
        // Создаем временную очередь для удаления из оригинальной минимального текущего элемента
        queue temp_que;
        Create(&temp_que);

        int flag = 0;

        while (!Empty(&cur)) {
```

```
        if ((min_value == Top(&cur).key) && (flag == 0)) {
          flag = 1;
          // Добавляем min_value в вспомогательную очередь
          Push(&help_que, Top(&cur));
        } else {
          Push(&temp_que, Top(&cur));
        }
        Pop(&cur);
      }
      cur = temp_que;

    }

    printf("После: ");
    Print(&help_que);
    return help_que;

}
```

make:

```
main: main.c qsort.o queue.o
gcc -o main main.c qsort.o queue.o

qsort.o: qsort.c queue.h
gcc -c qsort.c

queue.o: queue.c queue.h
gcc -c queue.c

clean:
rm *.o main
```

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

```
qwental@DESKTOP-NKF1EUK:/mnt/c/Users/Holiday/Desktop/Мои
 материалы/Информатика/2 Семестр/ЛР25-26_$ make main
cc     main.c   -o main
qwental@DESKTOP-NKF1EUK:/mnt/c/Users/Holiday/Desktop/Мои
 материалы/Информатика/2 Семестр/ЛР25-26_$ ./main
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 5
[ ]
-------------------------------------------
▌ 1) Добавить элемент в очередь.
```

```
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 2
Очередь пуста!
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 1
Введите элемент(ключ - значение):
▷▷▷ 12-1423
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 1
Введите элемент(ключ - значение):
▷▷▷ 354-345
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 1
Введите элемент(ключ - значение):
▷▷▷ 453-567
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
```

```
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
---------------------------------------------
▷▷▷ 1
Введите элемент(ключ - значение):
▷▷▷ 123-56456
---------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
---------------------------------------------
▷▷▷ 1
Введите элемент(ключ - значение):
▷▷▷ 5-234
---------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
---------------------------------------------
▷▷▷ 5
[ 12:1423 354:345 453:567 123:56456 5:234 ]
---------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
---------------------------------------------
▷▷▷ 6
ДО: [ 12:1423 354:345 453:567 123:56456 5:234 ]
После: [ 5:234 12:1423 123:56456 354:345 453:567 ]
Очередь отсортирована!
---------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
```

```
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 2
Элемент <5 - 234> удалён.
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 2
Элемент <12 - 1423> удалён.
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 4
До: [ 123:56456 354:345 453:567 ]
После: [ 123:56456 453:567 ]
Минимальный элемент был удален
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 2
Элемент <123 - 56456> удалён.
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
```

```
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 2
Элемент <453 - 567> удалён.
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 2
Очередь пуста!
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 1
Введите элемент(ключ - значение):
▷▷▷ 8-1
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 1
Введите элемент(ключ - значение):
▷▷▷ 7-2
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
```

```
▌ 0) Выход.
-------------------------------------------
▷▷▷ 1
Введите элемент(ключ - значение):
▷▷▷ 6-3
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 1
Введите элемент(ключ - значение):
▷▷▷ 5-4
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 1
Введите элемент(ключ - значение):
▷▷▷ 4-5
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 5
[ 8:1 7:2 6:3 5:4 4:5 ]
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
```

```
-------------------------------------------
▷▷▷ 6
ДО: [ 8:1 7:2 6:3 5:4 4:5 ]
После: [ 4:5 5:4 6:3 7:2 8:1 ]
Очередь отсортирована!
-------------------------------------------
▉ 1) Добавить элемент в очередь.
▉ 2) Удаление первого элемента в очереди.
▉ 3) Размер очереди.
▉ 4) Удаление минимального элемента.
▉ 5) Печать очереди.
▉ 6) Сортировка очереди.
▉ 0) Выход.
-------------------------------------------
▷▷▷ 4
До: [ 4:5 5:4 6:3 7:2 8:1 ]
После: [ 4:5 5:4 6:3 7:2 ]
Минимальный элемент был удален
-------------------------------------------
▉ 1) Добавить элемент в очередь.
▉ 2) Удаление первого элемента в очереди.
▉ 3) Размер очереди.
▉ 4) Удаление минимального элемента.
▉ 5) Печать очереди.
▉ 6) Сортировка очереди.
▉ 0) Выход.
-------------------------------------------
▷▷▷ 4
До: [ 4:5 5:4 6:3 7:2 ]
После: [ 4:5 5:4 6:3 ]
Минимальный элемент был удален
-------------------------------------------
▉ 1) Добавить элемент в очередь.
▉ 2) Удаление первого элемента в очереди.
▉ 3) Размер очереди.
▉ 4) Удаление минимального элемента.
▉ 5) Печать очереди.
▉ 6) Сортировка очереди.
▉ 0) Выход.
-------------------------------------------
▷▷▷ 1
Введите элемент(ключ - значение):
▷▷▷ 1-0
-------------------------------------------
▉ 1) Добавить элемент в очередь.
▉ 2) Удаление первого элемента в очереди.
▉ 3) Размер очереди.
▉ 4) Удаление минимального элемента.
```

```
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 4
До: [ 4:5 5:4 6:3 1:0 ]
После: [ 4:5 5:4 6:3 ]
Минимальный элемент был удален
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 1
Введите элемент(ключ - значение):
▷▷▷ 2-3
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 6
ДО: [ 4:5 5:4 6:3 2:3 ]
После: [ 2:3 4:5 5:4 6:3 ]
Очередь отсортирована!
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 4
До: [ 2:3 4:5 5:4 6:3 ]
После: [ 4:5 5:4 6:3 ]
Минимальный элемент был удален
-------------------------------------------
▌ 1) Добавить элемент в очередь.
```

```
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
--------------------------------------------
▷▷▷ 4
До: [ 4:5 5:4 6:3 ]
После: [ 4:5 5:4 ]
Минимальный элемент был удален
--------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
--------------------------------------------
▷▷▷ 4
До: [ 4:5 5:4 ]
После: [ 4:5 ]
Минимальный элемент был удален
--------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
--------------------------------------------
▷▷▷ 4
Минимальный элемент был удален
--------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
--------------------------------------------
▷▷▷ 2
Элемент <4 - 5> удалён.
--------------------------------------------
▌ 1) Добавить элемент в очередь.
```

```
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 2
Очередь пуста!
-------------------------------------------
▌ 1) Добавить элемент в очередь.
▌ 2) Удаление первого элемента в очереди.
▌ 3) Размер очереди.
▌ 4) Удаление минимального элемента.
▌ 5) Печать очереди.
▌ 6) Сортировка очереди.
▌ 0) Выход.
-------------------------------------------
▷▷▷ 0
Программа завершена!
qwental@DESKTOP-NKF1EUK:/mnt/c/Users/Holiday/Desktop/Мои
 материалы/Информатика/2 Семестр/ЛР25-26_$ exit
```

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

_____

_____

11 Выводы

Я научился работать с абстрактными типами данных и сортировками в них.

Подпись студента _____