**Отчет по лабораторной работе №25-26** по курсу практикум на ЭВМ

Студент группы М8О-101Б-20 Ядров Артем Леонидович, № по списку 28

Контакты www, e-mail, icq, skype temayadrow@gmail.com

Работа выполнена: « » \_\_\_\_\_\_\_\_\_\_\_\_202\_\_\_г.

Преподаватель: доцент каф. 806 Никулин Сергей Петрович

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

Отчет сдан « » \_\_\_\_\_\_\_\_\_202 \_\_ г., итоговая оценка \_\_\_\_\_

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

1. **Тема:** Автоматизация сборки программ модульной структуры на языке Си с использованием утилиты make. Абстрактные типы данных. Рекурсия. Модульное программирование на языке Си.
2. **Цель работы:** Изучить утилиту make, абстрактные типы данных, модульное программирование и рекурсию.

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

1. **Задание** (*вариант №* *28* )**:**  АТД: Очередь. Процедура: Поиск в очереди, списке, стеке или деке первого от начала элемента, который меньше своего непосредственного предшественника. Если такой элемент найден, смещение его к началу до тех пор, пока он не станет первым или больше своего предшественника. Метод: вариант метода вставки.
2. **Оборудование** (лабораторное):

ЭВМ Intel Pentium G2140, процессор 3.30 GHz , имя узла сети Cameron с ОП 8096 Мб, НМД 7906 Мб. Терминал ASUS адрес dev/pets/3 Принтер HP Laserjet 6P

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

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

*Оборудование ПЭВМ студента, если использовалось:*

Процессор \_\_Intel core i5-7300HQ 2.50 GHz с ОП 8096 Мб, НМД 131072 Мб. Монитор ASUS

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

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

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

Операционная система семейства Unix , наименование Ubuntu версия 18.15.0

интерпретатор команд bash версия 4.4.20

Система программирования CLion версия 2020.3

Редактор текстов emacs версия 25.2.2

Утилиты операционной системы cat, gcc

Прикладные системы и программы

Местонахождение и имена файлов программ и данных stud/208104

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

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства Unix , наименование Fedora версия 33

интерпретатор команд bash версия 5.0.17

Система программирования CLion версия 2020.3

Редактор текстов emacs версия 25.2.2 Утилиты операционной системы cat, gcc

Прикладные системы и программы

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

Местонахождение и имена файлов программ и данных на домашнем компьютере home/Temi4

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

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

25: Проанализируем файловую структуру модуля. На ее основе создадим Makefile с зависимостями программных файлов модуля.

26: Отдельно реализуем модуль дека на языке Си заголовочным файлом (deq.h) и реализуем методы модуля (deq.c). Модуль имеет операции добавления в конец, добавления в начало, удаления из начала, удаления из конца, печати очереди, получения элемента начала, конца.

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

Методы модуля:

udt \*create\_udt(udt \*d) {

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

d->first = d->last = -1;

d->size = 0;

return d;

}

bool udt\_empty(const udt \*d) {

return d->size == 0;

}

int udt\_size(const udt \*d) {

return d->size;

}

bool udt\_push\_back(udt \*d, data t) {

if (d->size == 10) {

return false;

}

if (!d->size) {

d->last = d->first = 0;

} else {

d->last = (d->last + 1) % 10;

}

d->arr[d->last] = t;

d->size++;

return true;

}

bool udt\_push\_front(udt \*d, data t) {

if (d->size == 10) {

return false;

}

if (!d->size) {

d->first = d->last = 0;

} else {

d->first = (10 + (d->first - 1) % 10) % 10;

}

d->arr[d->first] = t;

d->size++;

return true;

}

bool udt\_pop\_front(udt \*d) {

if (!d->size) {

return false;

}

if (d->size == 1) {

d->first = d->last = -1;

d->size--;

return true;

}

d->first++;

d->first %= 10;

d->size--;

return true;

}

bool udt\_pop\_back(udt \*d) {

if (!d->size) {

return false;

}

if (d->size == 1) {

d->first = d->last = -1;

d->size--;

return true;

}

d->last--;

d->last = (10 + d->last % 10) % 10;

d->size--;

return true;

}

data udt\_top\_left(udt \*d) {

if (d->size) {

return d->arr[d->first];

}

}

data udt\_top\_right(udt \*d) {

if (d->size) {

return d->arr[d->last];

}

}

void udt\_print(udt \*d) {

printf("Key\tValue\n");

int size = udt\_size(d);

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

data a = udt\_top\_left(d);

udt\_pop\_front(d);

printf("%d\t", a.key);

printf("%s\n", a.value);

udt\_push\_back(d, a);

}

}

**Тесты:**

Протестируем наихудший для сортировки случай: когда входные данные отсортированы в обратном порядке.

**Ключ Строка**

5 5

4 4

3 3

2 2

1 1

*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,подписанныйпреподавателем).

**[Temi4@localhost 25-26 (17 var)]$ cat Makefile**

**laba: deq.o main.o**

**gcc deq.o main.o**

**deque.o : deq.h deq.c**

**gcc -c deq.c**

**main.o : deq.h main.c**

**gcc -c main.c**

**[Temi4@localhost 25-26 (17 var)]$ cat deq.h**

**#ifndef \_UDT\_H\_**

**#define \_UDT\_H\_**

**#include <stdbool.h>**

**typedef struct {**

**int key;**

**char value[40];**

**} data;**

**typedef struct {**

**int first;**

**int last;**

**int size;**

**data arr[10];**

**} udt;**

**udt\* create\_udt(udt \*);**

**bool udt\_empty(const udt \*);**

**bool udt\_push\_front(udt \*, data);**

**bool udt\_push\_back(udt \*, data);**

**bool udt\_pop\_front(udt \*);**

**bool udt\_pop\_back(udt \*);**

**data udt\_top\_left(udt \*);**

**data udt\_top\_right(udt \*);**

**void udt\_print(udt \*);**

**int udt\_size(const udt \*);**

**#endif[Temi4@localhost 25-26 (17 var)]$ cat deq.c**

**#include <stdio.h>**

**#include <malloc.h>**

**#include "deq.h"**

**udt \*create\_udt(udt \*d) {**

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

**d->first = d->last = -1;**

**d->size = 0;**

**return d;**

**}**

**bool udt\_empty(const udt \*d) {**

**return d->size == 0;**

**}**

**int udt\_size(const udt \*d) {**

**return d->size;**

**}**

**bool udt\_push\_back(udt \*d, data t) {**

**if (d->size == 10) {**

**return false;**

**}**

**if (!d->size) {**

**d->last = d->first = 0;**

**} else {**

**d->last = (d->last + 1) % 10;**

**}**

**d->arr[d->last] = t;**

**d->size++;**

**return true;**

**}**

**bool udt\_push\_front(udt \*d, data t) {**

**if (d->size == 10) {**

**return false;**

**}**

**if (!d->size) {**

**d->first = d->last = 0;**

**} else {**

**d->first = (10 + (d->first - 1) % 10) % 10;**

**}**

**d->arr[d->first] = t;**

**d->size++;**

**return true;**

**}**

**bool udt\_pop\_front(udt \*d) {**

**if (!d->size) {**

**return false;**

**}**

**if (d->size == 1) {**

**d->first = d->last = -1;**

**d->size--;**

**return true;**

**}**

**d->first++;**

**d->first %= 10;**

**d->size--;**

**return true;**

**}**

**bool udt\_pop\_back(udt \*d) {**

**if (!d->size) {**

**return false;**

**}**

**if (d->size == 1) {**

**d->first = d->last = -1;**

**d->size--;**

**return true;**

**}**

**d->last--;**

**d->last = (10 + d->last % 10) % 10;**

**d->size--;**

**return true;**

**}**

**data udt\_top\_left(udt \*d) {**

**if (d->size) {**

**return d->arr[d->first];**

**}**

**}**

**data udt\_top\_right(udt \*d) {**

**if (d->size) {**

**return d->arr[d->last];**

**}**

**}**

**void udt\_print(udt \*d) {**

**printf("Key\tValue\n");**

**int size = udt\_size(d);**

**for (int i = 0; i < size; i++) {**

**data a = udt\_top\_left(d);**

**udt\_pop\_front(d);**

**printf("%d\t", a.key);**

**printf("%s\n", a.value);**

**udt\_push\_back(d, a);**

**}**

**}**

**[Temi4@localhost 25-26 (17 var)]$ cat main.c**

**#include <stdio.h>**

**#include <string.h>**

**#include "deq.h"**

**#include <limits.h>**

**#define INF INT\_MAX**

**data procedure(udt \*d) {**

**data t, a;**

**int max = -INF - 1;**

**int size = d->size;**

**for (int i = 0; i < size; i++) {**

**a = udt\_top\_left(d);**

**if (a.key > max) {**

**max = udt\_top\_left(d).key;**

**t = udt\_top\_left(d);**

**}**

**udt\_pop\_front(d);**

**udt\_push\_back(d, a);**

**}**

**udt \*d1 = create\_udt(NULL);**

**while (udt\_top\_left(d).key != t.key && udt\_top\_right(d).key != t.key) {**

**a = udt\_top\_left(d);**

**udt\_push\_front(d1, a);**

**udt\_pop\_front(d);**

**if (!udt\_empty(d)) {**

**a = udt\_top\_right(d);**

**udt\_push\_back(d1, a);**

**udt\_pop\_back(d);**

**}**

**}**

**if (udt\_top\_left(d).key == t.key) {**

**udt\_pop\_front(d);**

**} else {**

**udt\_pop\_back(d);**

**}**

**while (!udt\_empty(d1)){**

**udt\_push\_front(d, udt\_top\_left(d1));**

**udt\_pop\_front(d1);**

**}**

**return t;**

**}**

**udt\* sort(udt \*d) {**

**int size = udt\_size(d);**

**udt \*d1 = create\_udt(NULL);**

**data t;**

**for (int i = 0; i < size; i++) {**

**t = procedure(d);**

**udt\_push\_front(d1, t);**

**}**

**return d1;**

**}**

**int main() {**

**int c = 1, ans;**

**udt \*d = NULL;**

**while (c) {**

**printf("1. Create deq\t 2. Empty\t 3. Size\t 4. Push back\t 5. Push front\t 6. Top left\t 7. Top right\t 8.Pop back\t 9. Pop front\t 10.Print\t 11. Sort\t 12. Exit\n");**

**scanf("%d", &ans);**

**switch (ans) {**

**case 1: {**

**d = create\_udt(d);**

**break;**

**}**

**case 2: {**

**if (d == NULL) {**

**printf("Deq doesn't exist\n");**

**} else {**

**udt\_empty(d) ? printf("Deq is empty\n") : printf("Deq isn't empty\n");**

**}**

**break;**

**}**

**case 3: {**

**if (d == NULL) {**

**printf("Deq doesn't exist\n");**

**} else {**

**printf("%d\n", udt\_size(d));**

**}**

**break;**

**}**

**case 4: {**

**if (d == NULL) {**

**printf("Deq doesn't exist\n");**

**} else {**

**data tb;**

**printf("Print key\n");**

**scanf("%d", &tb.key);**

**printf("Print string\n");**

**scanf("%s", tb.value);**

**if (!udt\_push\_back(d, tb)) {**

**printf("Deq is full\n");**

**}**

**}**

**break;**

**}**

**case 5: {**

**if (d == NULL) {**

**printf("Deq doesn't exist\n");**

**} else {**

**data tf;**

**printf("Print key\n");**

**scanf("%d", &tf.key);**

**printf("Print string\n");**

**scanf("%s", tf.value);**

**if (!udt\_push\_front(d, tf)) {**

**printf("Deq is full\n");**

**}**

**}**

**break;**

**}**

**case 6: {**

**if (d == NULL) {**

**printf("Deq doesn't exist\n");**

**} else {**

**if (udt\_empty(d)) {**

**printf("Deq is empty\n");**

**} else {**

**data a = udt\_top\_left(d);**

**printf("Key\n%d\nValue\n%s\n", a.key, a.value);**

**}**

**}**

**break;**

**}**

**case 7: {**

**if (d == NULL) {**

**printf("Deq doesn't exist\n");**

**} else {**

**if (udt\_empty(d)) {**

**printf("Deq is empty\n");**

**} else {**

**data a = udt\_top\_right(d);**

**printf("Key\n%d\nValue\n%s\n", a.key, a.value);**

**}**

**}**

**break;**

**}**

**case 8: {**

**if (d == NULL) {**

**printf("Deq doesn't exist\n");**

**} else {**

**if (!udt\_pop\_back(d)) {**

**printf("Deq is empty\n");**

**}**

**}**

**break;**

**}**

**case 9: {**

**if (d == NULL) {**

**printf("Deq doesn't exist\n");**

**} else {**

**if (!udt\_pop\_front(d)) {**

**printf("Deq is empty\n");**

**}**

**}**

**break;**

**}**

**case 10: {**

**if (d == NULL) {**

**printf("Deq doesn't exist\n");**

**} else {**

**udt\_print(d);**

**}**

**break;**

**}**

**case 11: {**

**if (d == NULL) {**

**printf("Deq doesn't exist\n");**

**} else {**

**d = sort(d);**

**}**

**break;**

**}**

**case 12: {**

**c = 0;**

**break;**

**}**

**default: {**

**printf("Wrong answer\n");**

**}**

**}**

**}**

**return 0;**

**}**

**[Temi4@localhost 25-26 (17 var)]$ make**

**cc -c -o deq.o deq.c**

**gcc -c main.c**

**gcc deq.o main.o**

**[Temi4@localhost 25-26 (17 var)]$ ./a.out**

**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**

**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**

**2**

**Deq is empty**

**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**

**3**

**0**

**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**

**6**

**Deq is empty**

**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**

**7**

**Deq is empty**

**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**

**4**

**Print key**

**5**

**Print string**

**5**

**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**

**4**

**Print key**

**4**

**Print string**

**4**

**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**

**4**

**Print key**

**3**

**Print string**

**3**

**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**

**4**

**Print key**

**2**

**Print string**

**2**

**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**

**4**

**Print key**

**1**

**Print string**

**1**

**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**

**10**

**Key Value**

**5 5**

**4 4**

**3 3**

**2 2**

**1 1**

**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**

**11**

**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**

**10**

**Key Value**

**1 1**

**2 2**

**3 3**

**4 4**

**5 5**

**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**

**2**

**Deq isn't empty**

**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**

**3**

**5**

**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**

**6**

**Key**

**1**

**Value**

**1**

**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**

**7**

**Key**

**5**

**Value**

**5**

**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**

**8**

**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**

**9**

**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**

**10**

**Key Value**

**2 2**

**3 3**

**4 4**

**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**

**5**

**Print key**

**1**

**Print string**

**1**

**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**

**4**

**Print key**

**5**

**Print string**

**5**

**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**

**10**

**Key Value**

**1 1**

**2 2**

**3 3**

**4 4**

**5 5**

**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**

**12**

**[Temi4@localhost 25-26 (17 var)]$ make**

**gcc deq.o main.o**

**[Temi4@localhost 25-26 (17 var)]$ touch deq.h**

**[Temi4@localhost 25-26 (17 var)]$ ls -l**

**итого 64**

**-rwxrwxr-x. 1 Temi4 Temi4 29992 мая 20 20:10 a.out**

**-rw-r--r--. 1 Temi4 Temi4 1805 мая 20 19:23 deq.c**

**-rw-r--r--. 1 Temi4 Temi4 492 мая 20 20:11 deq.h**

**-rw-rw-r--. 1 Temi4 Temi4 4064 мая 20 20:09 deq.o**

**-rw-r--r--. 1 Temi4 Temi4 5260 мая 20 19:24 main.c**

**-rw-rw-r--. 1 Temi4 Temi4 7608 мая 20 20:09 main.o**

**-rw-rw-r--. 1 Temi4 Temi4 110 мая 20 19:23 Makefile**

**[Temi4@localhost 25-26 (17 var)]$ make**

**gcc -c main.c**

**gcc deq.o main.o**

**[Temi4@localhost 25-26 (17 var)]$ touch main.c**

**[Temi4@localhost 25-26 (17 var)]$ ls -l**

**итого 64**

**-rwxrwxr-x. 1 Temi4 Temi4 29992 мая 20 20:11 a.out**

**-rw-r--r--. 1 Temi4 Temi4 1805 мая 20 19:23 deq.c**

**-rw-r--r--. 1 Temi4 Temi4 492 мая 20 20:11 deq.h**

**-rw-rw-r--. 1 Temi4 Temi4 4064 мая 20 20:09 deq.o**

**-rw-r--r--. 1 Temi4 Temi4 5260 мая 20 20:11 main.c**

**-rw-rw-r--. 1 Temi4 Temi4 7608 мая 20 20:11 main.o**

**-rw-rw-r--. 1 Temi4 Temi4 110 мая 20 19:23 Makefile**

**[Temi4@localhost 25-26 (17 var)]$ make**

**gcc -c main.c**

**gcc deq.o main.o**

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

1. **Выводы**

Я изучил принцип работы утилиты make, а также абстрактный тип данных, рекурсию и модульное программирование.

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

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