**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 очереди на языке Си заголовочным файлом (queue.h) и реализуем методы модуля (queue.c). Модуль имеет операции добавления в конец, удаления из начала, печати очереди, получения элемента начала очереди, сортировки

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

Методы модуля:

void udt\_create(udt \*q) {

q->first = q->size = 0;

}

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

return q->size == 0;

}

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

return q->size;

}

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

if (q->size == 100) {

return false;

}

q->arr[(q->first + q->size++) % 100] = t;

return true;

}

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

if (!q->size) {

return false;

}

q->first++;

q->first %= 100;

q->size--;

return true;

}

data udt\_top(udt \*q) {

if (q->size) {

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

}

}

void udt\_print(udt \*q) {

printf("Key\tValue\n");

int size = udt\_size(q);

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

data a = udt\_top(q);

udt\_pop\_front(q);

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

for (int j = 0; j < 40; j++) {

if (a.value[j] != '\n') {

printf("%c", a.value[j]);

} else {

break;

}

}

printf("\n");

udt\_push\_back(q, a);

}

}

bool udt\_procedure(udt \*q) {

if (udt\_empty(q)) {

return false;

}

bool ok = false;

int size = udt\_size(q);

data a[size];

a[0] = udt\_top(q);

data prev = a[0];

udt\_pop\_front(q);

int n = 1;

while (!udt\_empty(q)) {

data cur = udt\_top(q);

udt\_pop\_front(q);

a[n] = cur;

if (!ok && cur.key < prev.key) {

for (int i = n - 1; i >= 0; i--) {

if (a[i].key > a[i + 1].key) {

data tmp = a[i];

a[i] = a[i + 1];

a[i + 1] = tmp;

} else {

break;

}

}

ok = true;

}

n++;

prev = cur;

}

udt\_create(q);

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

udt\_push\_back(q, a[i]);

}

return ok;

}

void udt\_sort(udt \*q) {

while (udt\_procedure(q)) {}

}

**Тесты:**

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

**Ключ Строка**

10 10

9 9

8 8

7 7

6 6

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]$ cat queue.h**

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

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

**#include <stdbool.h>**

**typedef struct {**

**int key;**

**char value[40];**

**} data;**

**typedef struct {**

**int first;**

**int size;**

**data arr[10];**

**} udt;**

**void udt\_create(udt \*);**

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

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

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

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

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

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

**#endif[Temi4@localhost 25-26]$ cat queue.c**

**#include <stdio.h>**

**#include "queue.h"**

**void udt\_create(udt \*q) {**

**q->first = q->size = 0;**

**}**

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

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

**}**

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

**return q->size;**

**}**

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

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

**return false;**

**}**

**q->arr[(q->first + q->size++) % 10] = t;**

**return true;**

**}**

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

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

**return false;**

**}**

**q->first++;**

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

**q->size--;**

**return true;**

**}**

**data udt\_top(udt \*q) {**

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

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

**}**

**}**

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

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

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

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

**data a = udt\_top(q);**

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

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

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

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

**}**

**}[Temi4@localhost 25-26]$ cat main.c**

**#include <stdio.h>**

**#include <malloc.h>**

**#include <string.h>**

**#include "queue.h"**

**bool udt\_procedure(udt \*q) {**

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

**return false;**

**}**

**bool ok = false;**

**udt \*q1 = (udt \*) malloc(sizeof(udt)); // очередь из обработанных элементов**

**udt\_create(q1);**

**udt \*q2 = (udt \*) malloc(sizeof(udt)); // вспомогательная очередь для обмена позициями**

**udt\_create(q2);**

**data cur, prev = udt\_top(q);**

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

**cur = udt\_top(q);**

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

**if (cur.key < prev.key) {**

**ok = true;**

**bool ok\_1 = false;**

**while (!udt\_empty(q1)) {**

**if (ok\_1) {**

**udt\_push\_back(q2, udt\_top(q1));**

**udt\_pop\_front(q1);**

**} else {**

**if (udt\_top(q1).key < cur.key) {**

**udt\_push\_back(q2, udt\_top(q1));**

**udt\_pop\_front(q1);**

**} else {**

**udt\_push\_back(q2, cur);**

**ok\_1 = true;**

**}**

**}**

**}**

**break;**

**}**

**udt\_push\_back(q1, cur);**

**prev = cur;**

**}**

**if (ok) { // заканчиваем "перекладывание очереди q в q2"**

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

**udt\_push\_back(q2, udt\_top(q));**

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

**}**

**q1 = q2;**

**}**

**udt\_create(q); // очистим буфер очереди**

**while (!udt\_empty(q1)) { // возвращение элементов в очередь**

**data a = udt\_top(q1);**

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

**udt\_pop\_front(q1);**

**}**

**return ok;**

**}**

**void udt\_sort(udt \*q) {**

**while (udt\_procedure(q)) {}**

**}**

**int main() {**

**int c = 1, ans;**

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

**while (c) {**

**printf("1. Create queue\t 2. Empty\t 3. Size\t 4. Push back\t 5. Top\t 6.Pop\t 7.Print\t 8. Sort\t 9. Exit\n");**

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

**switch (ans) {**

**case 1: {**

**udt\_create(q);**

**break;**

**}**

**case 2: {**

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

**break;**

**}**

**case 3: {**

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

**break;**

**}**

**case 4: {**

**data t;**

**char a[40];**

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

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

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

**scanf("%s", a);**

**strcpy(t.value, a);**

**if (!udt\_push\_back(q, t)) {**

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

**}**

**break;**

**}**

**case 5: {**

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

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

**} else {**

**data a = udt\_top(q);**

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

**}**

**break;**

**}**

**case 6: {**

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

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

**}**

**break;**

**}**

**case 7: {**

**udt\_print(q);**

**break;**

**}**

**case 8: {**

**udt\_sort(q);**

**break;**

**}**

**case 9: {**

**c = 0;**

**break;**

**}**

**default: {**

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

**}**

**}**

**}**

**return 0;**

**}[Temi4@localhost 25-26]$ cat Makefile**

**laba: queue.o main.o**

**gcc queue.o main.o**

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

**gcc -c queue.c**

**main.o : queue.h main.c**

**gcc -c main.c**

**[Temi4@localhost 25-26]$ make**

**gcc -c queue.c**

**gcc -c main.c**

**gcc queue.o main.o**

**[Temi4@localhost 25-26]$ ./a.out**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**1**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**4**

**Print key**

**10**

**Print string**

**10**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**4**

**Print key**

**9**

**Print string**

**9**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**4**

**Print key**

**8**

**Print string**

**8**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**4**

**Print key**

**7**

**Print string**

**7**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**4**

**Print key**

**6**

**Print string**

**6**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**4**

**Print key**

**5**

**Print string**

**5**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**4**

**Print key**

**4**

**Print string**

**4**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**4**

**Print key**

**3**

**Print string**

**3**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**4**

**Print key**

**2**

**Print string**

**2**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**4**

**Print key**

**1**

**Print string**

**1**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**2**

**Queue isn't empty**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**3**

**10**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**5**

**Key**

**10**

**Value**

**10**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**7**

**Key Value**

**10 10**

**9 9**

**8 8**

**7 7**

**6 6**

**5 5**

**4 4**

**3 3**

**2 2**

**1 1**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**8**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**7**

**Key Value**

**1 1**

**2 2**

**3 3**

**4 4**

**5 5**

**6 6**

**7 7**

**8 8**

**9 9**

**10 10**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**4**

**Print key**

**111**

**Print string**

**abcddf**

**Queue is full**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**5**

**Key**

**1**

**Value**

**1**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**6**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**4**

**Print key**

**1**

**Print string**

**abcdefg**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**7**

**Key Value**

**2 2**

**3 3**

**4 4**

**5 5**

**6 6**

**7 7**

**8 8**

**9 9**

**10 10**

**1 abcdefg**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**8**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**7**

**Key Value**

**1 abcdefg**

**2 2**

**3 3**

**4 4**

**5 5**

**6 6**

**7 7**

**8 8**

**9 9**

**10 10**

**1. Create queue 2. Empty 3. Size 4. Push back 5. Top 6.Pop 7.Print 8. Sort 9. Exit**

**9**

**[Temi4@localhost 25-26]$ make**

**gcc queue.o main.o**

**[Temi4@localhost 25-26]$ touch queue.h**

**[Temi4@localhost 25-26]$ ls -l**

**итого 156**

**-rwxrwxr-x. 1 Temi4 Temi4 25840 апр 30 08:10 a.out**

**-rw-r--r--. 1 Temi4 Temi4 18601 апр 29 16:26 laba\_25-26.docx**

**-rw-r--r--. 1 Temi4 Temi4 81707 апр 29 16:25 laba\_25-26.pdf**

**-rw-r--r--. 1 Temi4 Temi4 3648 апр 30 08:05 main.c**

**-rw-rw-r--. 1 Temi4 Temi4 5952 апр 30 08:06 main.o**

**-rw-rw-r--. 1 Temi4 Temi4 122 апр 30 07:57 Makefile**

**-rw-r--r--. 1 Temi4 Temi4 867 апр 30 07:40 queue.c**

**-rw-r--r--. 1 Temi4 Temi4 382 апр 30 08:10 queue.h**

**-rw-rw-r--. 1 Temi4 Temi4 2952 апр 30 08:06 queue.o**

**[Temi4@localhost 25-26]$ make**

**gcc -c queue.c**

**gcc -c main.c**

**gcc queue.o main.o**

**[Temi4@localhost 25-26]$ touch main.c**

**[Temi4@localhost 25-26]$ ls -l**

**итого 156**

**-rwxrwxr-x. 1 Temi4 Temi4 25840 апр 30 08:11 a.out**

**-rw-r--r--. 1 Temi4 Temi4 18601 апр 29 16:26 laba\_25-26.docx**

**-rw-r--r--. 1 Temi4 Temi4 81707 апр 29 16:25 laba\_25-26.pdf**

**-rw-r--r--. 1 Temi4 Temi4 3648 апр 30 08:11 main.c**

**-rw-rw-r--. 1 Temi4 Temi4 5952 апр 30 08:11 main.o**

**-rw-rw-r--. 1 Temi4 Temi4 122 апр 30 07:57 Makefile**

**-rw-r--r--. 1 Temi4 Temi4 867 апр 30 07:40 queue.c**

**-rw-r--r--. 1 Temi4 Temi4 382 апр 30 08:10 queue.h**

**-rw-rw-r--. 1 Temi4 Temi4 2952 апр 30 08:11 queue.o**

**[Temi4@localhost 25-26]$ make**

**gcc -c main.c**

**gcc queue.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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_