|  |  |
| --- | --- |
|  | **Отчёт по лабораторной работе** №25-26 по курсу Языки и методы программирования­  студента группы М08-107Б-18 Хренова Геннадия Николаевича , № по списку 24  Адреса www, e-mail, jabber, skype [khrenov.gena@yandex.ru](mailto:khrenov.gena@yandex.ru)  Работа выполнена: “6“ апреля 2019 г.  Преподаватель: аспирант каф.806 Ридли А. Н.  Входной контроль знаний с оценкой  Отчёт сдан “ “ 2019г., итоговая оценка  Подпись преподавателя |

1. **Тема**: Абстрактные типы данных. Рекурсия. Модульное программирование на языке си.
2. **Цель работы**: Составить и отладить модуль определений и модуль реализации по заданной схеме модуля определений для абстрактного типа данных(очередь). Составить модуль сортирующий экземпляр заданным методом.
3. **Задание** (*вариант №* 2,2): очередь, сортировка простой вставкой
4. **Оборудование** (*лабораторное*):

ЭВМ Pentium , процессор 2.7 ГГц , имя узла сети с ОП 4096 МБ

НМД 2 ГБ. Терминал адрес . Принтер

Другие устройства

*Оборудование ПЭВМ студента, если использовалось:*

Процессор intel core i3 , ОП 3072 , НМД 50 ГБ. Монитор

Другие устройства

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

Операционная система семейства Unix , наименование версия

Интерпретатор команд bash версия

Система программирования версия

Редактор текстов Emacs версия

Утилиты операционной системы

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства Unix , наименование Ubuntu версия 18.04LTS

Интерпретатор команд bash версия

Система программирования версия

Редактор текстов Emacs версия

Утилиты операционной системы

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

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

Программа состоит из следующих составляющих:

1) **queue.h** - описание структуры очереди и обозначения всех функций

2) **queue.c** — изложены все операции очереди, а также функция, выполняющая задание

3) **sort.c & sort.h** — изложен метод сортировки при помощи операций, описанных ранее

3) **main.c** — главная часть, вызывающая функции

объединения всех программ в единый проект происходит с помощью команды **makefile**

lab26

sort.o

queue.o

queue.c

queue.h

sort.c

sort.h

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

Тесты:

1) 1 2 4 новый — 3

2)12 44 54 новый — 1

3) 1 2 3 4 новый — 4

4)\_\_\_ - новый 0

5) Pop в нулевой очереди

6) проверить возможные выходы за границы массива

7) Sort 10 -1 4 8 2 -23 11 0

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

Допущен к выполнению работы. Подпись преподавателя

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

gennadii@lenovo-b560:~$ cd workdir/26

gennadii@lenovo-b560:~/workdir/26$ cat queue.h

#ifndef \_QUEUE\_H\_

#define \_QUEUE\_H\_

#include <stdbool.h>

#define POOL\_SIZE 100

typedef struct

{

int first;

int size;

int data[POOL\_SIZE];

}queue;

void Create(queue\* q);

bool Empty(const queue\* q);

int Size(queue\* q);

int Top(queue\* q);

int Push(queue\* q, int t);

int Pop(queue\* q);

int Pushfew (queue\* q);

void printq(queue\* q);

#endifgennadii@lenovo-b560:~/workdir/26$ cat queue.c

//очередь

#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

#include "queue.h"

void Create(queue\* q)

{

q->first = 0;

q->size = 0;

}

bool Empty(const queue\* q)

{

return q->size == 0;

}

int Size(queue\* q)

{

printf("%d\n", q->size);

}

int Top(queue\* q)

{

if(q->size)

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

}

int Push(queue\* q, int t)

{

if (q->size == POOL\_SIZE){

fprintf(stderr, "this queue is full: " );

return 1;

}

q-> data[(q->first + q->size++) % POOL\_SIZE] = t;

return 0;

}

int Pushfew (queue\* q)

{

int t, n;

printf("number added elements = ");

scanf("%d", &n);

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

printf("element %d = ", i + 1);

scanf("%d", &t);

if (q->size == POOL\_SIZE){

fprintf(stderr, "this queue is full: " );

return 1;

}

q-> data[(q->first + q->size++) % POOL\_SIZE] = t;

}

return 0;

}

int Pop(queue\* q)

{

q-> size--;

if (q->size <= 0){

fprintf(stderr,"there`s no elements in queue");

return 2;

}

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

q->data[i] = q->data[i+1];

}

return 0;

}

void printq(queue\* q)

{

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

printf("%d ", q->data[i]);

}

printf("\n");

}

gennadii@lenovo-b560:~/workdir/26$ cat sort.c

#include <stdio.h>

#include <stdlib.h>

#include "queue.h"

void procedure (queue\* q, int d)

{

if (Size(q) == 0){

Push(q, d);

}else if (d < Top(q)){

Push(q, d);

while (d != Top(q)){

Push(q, Top(q));

Pop(q);

}

}else{

int one = Top(q);

while (Top(q) <= d){

Push(q, Top(q));

Pop(q);

if (one == Top(q)){

break;

}

}

Push(q, d);

while (Top(q) != one){

Push(q, Top(q));

Pop(q);

}

}

}

void sort(queue\* q)

{

queue w;

Create(&w);

while(Size(q) > 0){

int d = Top(q);

Pop(q);

procedure(&w, d);

}

while(Size(&w) > 0){

Push(q, Top(&w));

Pop(&w);

}

}

gennadii@lenovo-b560:~/workdir/26$ cat main.c

#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

#include "queue.h"

#include "sort.h"

int main(void)

{

queue q;

int t;

Create(&q);

Pushfew (&q);

printf("new element= ");

scanf("%d", &t);

sort(q, t);

return 0;

}gennadii@lenovo-b560:~/workdir/26$ cat makifile

cat: makifile: Нет такого файла или каталога

gennadii@lenovo-b560:~/workdir/26$ cat makefile

CC = gcc

FLAGS = -g -std=c99 -w -pipe -O3

all: lab26

lab26: queue.o sort.o

$(CC) $(FLAGS) main.c queue.o sort.o -o lab26

queue.o: queue.h queue.c

$(CC) $(FLAGS) -c queue.c

sort.o: queue.h sort.h sort.c

$(CC) $(FLAGS) -c sort.c

clean:

rm -f \*.ogennadii@lenovo-b560:~/workdir/26$ make

gcc -g -std=c99 -w -pipe -O3 -c queue.c

gcc -g -std=c99 -w -pipe -O3 -c sort.c

gcc -g -std=c99 -w -pipe -O3 main.c queue.o sort.o -o lab26

gennadii@lenovo-b560:~/workdir/26$ ./lab26

queue created

Pushfew

number added elements = 5

element 1 = 2

element 2 = -1

element 3 = 10

element 4 = 0

element 5 = 21

queue: 2 -1 10 0 21

Sort

-1 0 2 10 21

Pop

queue: 0 2 10 21

Push 3

queue: 0 2 10 21 3

Sort

0 2 3 10 21

qq

gennadii@lenovo-b560:~/workdir/26$ make

gcc -c -o main.o main.c

gcc -g -std=c99 -w -pipe -O3 main.c queue.o sort.o -o lab26

gennadii@lenovo-b560:~/workdir/26$

**9.Дневник отладки** должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) 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. Замечание автора по существу работы
2. Выводы: Очередь – структура данных, позволяющая добавлять элементы лишь в конец списка, и извлекать их из его начала. Она функционирует по принципу «первым пришёл — первым вышел». Очереди часто используются в программах для реализации буфера, в который можно положить элемент для последующей обработки, сохраняя порядок поступления. Makefile — это набор инструкций для программы make, которая помогает собирать программный проект буквально в одно касание. Makefile незаменим работе с большими проектами, ведь компиляция вручную очень неудобна и занимает много времени.

