**Отчет по лабораторной работе № 25-26**

по курсу “Фундаментальная информатика”.

Студент группы М8О-111Б-23 Ласточкин Максим Владимирович, № по списку 12 Контакты e-mail: lastochkin.maks05@mail.ru

Работа выполнена: «10» мая 2024 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 языке Си.

1. **Цель работы:** составить модуль определений и модуль реализации по заданной схеме модуля определений для очереди. Составить программный модуль сортирующий очередь указанным методом.
2. **Задание**: АТД - очередь, процедура - конкатенация, метод - сортировка Хоара.

1. **Оборудование** (лабораторное):

ЭВМ \_\_\_\_\_\_\_\_\_, процессор \_\_\_\_\_\_\_. имя узла сети\_\_\_\_\_\_\_с ОП\_\_\_\_\_ Мб, НМД \_\_\_\_ Мб. Терминал \_\_\_\_\_\_\_\_адрес \_\_\_\_\_\_\_\_. Принтер\_\_\_\_\_\_\_

Другие устройства

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

*Оборудование ПЭВМ студента, если использовалось:*

Процессор \_Intel Core i5 4690\_ с ОП 8 Гб НМД \_\_\_256+1024\_\_ Гб. Монитор 1920x1080~60Hz 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 \_Linux\_, наименование \_Ubuntu\_ версия 22.04.2 интерпретатор команд \_GNU bash\_ версия 5.1.16.

Система программирования С.

Редактор текстов emacs версия 29.1

Утилиты операционной системы \_\_\_\_\_\_\_\_

Прикладные системы и программы Emacs, gcc

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

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

Для выполнения данного задания требуются структуры:

1. data\_type: {int key, int value}
2. queue: { int first; int size; data\_type data[POOL\_SIZE];}

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. Concatenation: объединение двух очередей.
10. Qsort: сортировка методом Хоара.

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

**Текст программы:**

**main.c:**

#include <stdio.h>

#include <stdlib.h>

#include "queue.h"

#include "qsort.h"

void print\_menu();

int main() {

queue q;

data\_type t;

Create(&q);

queue q2;

data\_type y;

Create(&q2);

print\_menu();

int number;

scanf("%d", &number);

while (1){

switch (number) {

case 0:

Destroy(&q);

puts("Программа завершена!");

exit(0);

case 1:

printf("Печатаю список...\n");

Print(&q);

break;

case 2:

printf("Введите элемент для вставки (ключ - значение): \n");

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

Push(&q, t);

printf("Элемент успешно добавлен.\n");

break;

case 3:

if (!Empty(&q)) {

printf("Элемент <%d - %d> удалён.\n", Top(&q).key, Top(&q).value);

Pop(&q);

} else {

puts("Очередь пуста!");

}

break;

case 4:

printf("Идет подсчет длины...\n");

printf("Длина = %d", Size(&q));

printf("\n");

break;

case 5:

printf("Сортирую очередь...\n");

q = Qsort(&q);

break;

default:

printf("Введены неверные данные, попробуйте ввести снова!\n");

}

print\_menu();

scanf("%d", &number);

}

return 0;

}

void print\_menu(){

printf("Введите 0, чтобы завершить программу.\n");

printf("Введите 1, чтобы напечатать очередь.\n");

printf("Введите 2, чтобы вставить новый элемент в очередь.\n");

printf("Введите 3, чтобы удалить элемент из очереди.\n");

printf("Введите 4, чтобы подсчитать длину очереди.\n");

printf("Введите 5, чтобы отсортировать очередь.\n");

}

**qsort.h:**

//

// Created by lasto on 10.05.2024.

//

#ifndef LABA25\_26\_QSORT\_H

#define LABA25\_26\_QSORT\_H

#include <stdio.h>

#include <stdbool.h>

queue Concatenation(queue \*first, queue \*second);

queue Qsort(queue \*q);

#endif //LABA25\_26\_QSORT\_H

**queue.h:**

//

// Created by lasto on 09.05.2024.

//

#ifndef LABA25\_26\_QUEUE\_H

#define LABA25\_26\_QUEUE\_H

#include <stdio.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;

queue Concatenation(queue \*first, queue \*second);

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 //LABA25\_26\_QUEUE\_H

**queue.c:**

//

// Created by lasto on 09.05.2024.

//

#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 (data\_type){-1, -1};

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:**

//

// Created by lasto on 09.05.2024.

//

#include "queue.h"

queue Concatenation(queue \*first, queue \*second)

{

queue f = \*first, s = \*second;

while (!Empty(&s))

{

Push(&f, Top(&s));

Pop(&s);

}

return f;

}

queue Qsort(queue \*q){

if (Empty(q))

return \*q;

if (Size(q) == 1)

return \*q;

int num = Top(q).key;

queue left, right;

Create(&left);

Create(&right);

while(!Empty(q)){

if (Top(q).key <= num){

Push(&left, Top(q));

}else{

Push(&right, Top(q));

}

Pop(q);

}

if(Empty(&right)){

Push(&right, Top(&left));

Pop(&left);

}

left = Qsort(&left);

right = Qsort(&right);

return Concatenation(&left, &right);

}

**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

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

max@DESKTOP-L04A0IM:/mnt/c/Users/lasto/CLionProjects/laba25-26$ make main

gcc -c queue.c

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

max@DESKTOP-L04A0IM:/mnt/c/Users/lasto/CLionProjects/laba25-26$ ./main

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

2

Введите элемент для вставки (ключ - значение):

5-1

Элемент успешно добавлен.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

2

Введите элемент для вставки (ключ - значение):

100-2

Элемент успешно добавлен.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

2

Введите элемент для вставки (ключ - значение):

10-3

Элемент успешно добавлен.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

2

Введите элемент для вставки (ключ - значение):

200-5

Элемент успешно добавлен.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

2

Введите элемент для вставки (ключ - значение):

1-10

Элемент успешно добавлен.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

1

Печатаю список...

[ 5:1 100:2 10:3 200:5 1:10 ]

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

3

Элемент <5 - 1> удалён.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

1

Печатаю список...

[ 100:2 10:3 200:5 1:10 ]

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

2

Введите элемент для вставки (ключ - значение):

1000-11

Элемент успешно добавлен.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

2

Введите элемент для вставки (ключ - значение):

29-1230

Элемент успешно добавлен.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

1

Печатаю список...

[ 100:2 10:3 200:5 1:10 1000:11 29:1230 ]

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

5

Сортирую очередь...

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

1

Печатаю список...

[ 1:10 10:3 29:1230 100:2 200:5 1000:11 ]

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

3

Элемент <1 - 10> удалён.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

3

Элемент <10 - 3> удалён.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

3

Элемент <29 - 1230> удалён.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

3

Элемент <100 - 2> удалён.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

3

Элемент <200 - 5> удалён.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

3

Элемент <1000 - 11> удалён.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

3

Очередь пуста!

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

2

Введите элемент для вставки (ключ - значение):

5-10

Элемент успешно добавлен.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

2

Введите элемент для вставки (ключ - значение):

4-11

Элемент успешно добавлен.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

2

Введите элемент для вставки (ключ - значение):

3-2

Элемент успешно добавлен.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

2

Введите элемент для вставки (ключ - значение):

2-2

Элемент успешно добавлен.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

2

Введите элемент для вставки (ключ - значение):

1-1

Элемент успешно добавлен.

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

1

Печатаю список...

[ 5:10 4:11 3:2 2:2 1:1 ]

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

5

Сортирую очередь...

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

1

Печатаю список...

[ 1:1 2:2 3:2 4:11 5:10 ]

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

4

Идет подсчет длины...

Длина = 5

Введите 0, чтобы завершить программу.

Введите 1, чтобы напечатать очередь.

Введите 2, чтобы вставить новый элемент в очередь.

Введите 3, чтобы удалить элемент из очереди.

Введите 4, чтобы подсчитать длину очереди.

Введите 5, чтобы отсортировать очередь.

0

Программа завершена!

max@DESKTOP-L04A0IM:/mnt/c/Users/lasto/CLionProjects/laba25-26$

**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 |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |

1. **Замечания автора** по существу работы

1. **Выводы**

Научился работать с абстрактными типами данных, составлять модули определений и модули реализации. Написал алгоритм конкатенации очередей и сортировки очереди по методу Хоара.

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