**Отчет по лабораторной работе №25-26** по курсу практикум программирования

Студент группы М8О-104Б-22 Тесля Данила Сергеевич, № по списку 16

Контакты www, e-mail, icq, skype tesla\_2002@mail.ru

Работа выполнена: «30» сентября 2023г.

Преподаватель: \_\_\_\_\_\_ каф. 806 Потенко Максим Алексеевич

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

Отчет сдан «30» сентября 2023 г., итоговая оценка \_\_\_\_\_

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

* 1. **Тема :** Абстрактные типы данных. Рекурсия. Модульное программирование на Си. Автоматизация сборки программ модульной структуры на Си с использованием утилиты make.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. **Цель работы:** Составить и отладить программу на языке Си, содержащую модуль определений и модуль реализации по заданной схеме модуля определений для абстрактного типа данных. Автоматизировать сборку программы с помощью утилиты make.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

1. **Задание** (*вариант №* **Q6**)**: Процедура:** поиск в очереди первого от начала элемента, который меньше своего предшественника. Если такой элемент найден, смещение его к началу до тех пор, пока он не станет первым или больше своего предшественника. **Метод:** вариант сортировки методом 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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

*Оборудование ПЭВМ студента, если использовалось:*

Процессор \_\_Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz с ОП\_\_8192\_\_Мб, НМД 524 288 Мб. Монитор \_\_\_-\_\_\_\_\_\_\_\_\_

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

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

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

Операционная система семейства \_\_\_Unix\_\_\_\_, наименование \_\_\_\_\_Ubuntu\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_4.15.0\_\_\_\_\_\_

интерпретатор команд \_\_\_\_bash\_\_\_\_\_\_\_\_ версия \_\_\_4.4.20\_\_\_\_

Система программирования \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Редактор текстов \_\_\_\_\_emacs\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_25.2.2 \_\_\_\_\_\_\_\_\_

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

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

Прикладные системы и программы \_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Местонахождение и имена файлов программ и данных\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства \_\_\_Unix\_\_\_\_, наименование \_\_Ubuntu\_\_\_\_\_\_\_\_ версия \_\_\_22.04.3 LTS\_\_\_\_\_\_

интерпретатор команд \_\_\_\_bash\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_.

Система программирования \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_версия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Редактор текстов \_\_\_\_\_Sublime\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия 25.2.2\_\_\_\_\_\_\_\_\_\_\_\_

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

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

Прикладные системы и программы **\_\_\_**

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

Местонахождение и имена файлов программ и данных на домашнем компьютере \_\_\_\_\_\_/Documents\_\_\_\_\_\_\_\_

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

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

Перед выполнением работы необходимо изучить материалы по созданию и обработке абстрактного типа данных – очереди на языке Си, а также изучить основы модульного программирование на Си и автоматизацию сборки программ модульной структуры на Си с использованием утилиты make. Создадим программу из трех файлов, один из которых будет главным файлом программы, в нем будет функция main, во втором файле будет реализация всех необходимых функций программы, и третий – заголовочный файл с определениями структур и функций. Так же дополнительно создадим файл в котором будут команды – указания для корректной сборки и компиляции программы, запускаться данный файл будет с помощью утилиты make. В заголовочном файле (.h) определим структуру узла очереди и структуру самой очереди –реализуемой с помощью структуры данных односвязного списка, поля у структуры узла будут – значение узла и указатель на следующий узел, а в структуре очереди будут два поля – указатель на первый узел очереди, и на последний узел очереди. Далее в файле для реализации функций необходимо будет реализовать функции создания очереди, вычисления размера очереди, проверки очереди на пустоту, добавление элемента в конец очереди, извлечение и удаление элемента из начала очереди, печати очереди, процедуры согласно варианту задания, сортировки согласно варианту задания, удаление очереди. Когда все эти функции будут реализованы – перейдем главному файлу программы – в функции main сделаем контекстное меню, где пользователь сможет произвести действия с очередью по выбору – добавление узла в очередь, удаление узла, печать очереди, сортировка очереди, сделать это можно например конструкцией switch case в цикле while, и считывать выбор пользователя, пока он не захочет завершить работу с очередью.

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

Создадим текстовый файл Makefile, содержащий набор инструкций о том, как компилировать и линковать исходный код программы:

CC - переменная, указывающая компилятор C (gcc).

DEBUGGER - переменная, указывающая отладчик (gdb).

CFLAGS - флаги компиляции, включая -std=c99, -Wall и -Werror.

FILE\_OUT - имя выходного исполняемого файла (lab26.out).

SOURCES - исходные файлы программы (26main.c и 26source.c).

all - компиляция программы с использованием $(CC) $(CFLAGS) $(SOURCES) -o $(FILE\_OUT).

clean - удаление всех файлов с расширением .out.

debug - компиляция программы с отладочной информацией (-g) для последующей отладки с помощью gdb.

comp - компиляция программы без отладочной информации.

run - выполнение скомпилированной программы.

Таким образом мы автоматизировали сборку программ модульной структуры на Си, осталось только применить утилиту make для применения этих инструкций к программе.

Создадим программу из трех файлов, один из которых будет главным файлом программы – 26main.c, в нем будет функция main, во втором файле 26source.c будет реализация всех необходимых функций программы, и третий – заголовочный файл 26header.h с определениями структур и функций. В заголовочном файле для начала воспользуемся директивами #ifndef \_FILE\_H / #define \_FILE\_H /#endif методом условной компиляции, для предотвращения повторной компиляции кода, подключим стандартные библиотеки языка Си. Далее в этом же файле введем определение структуры узла, и структуры самой очереди typedef struct \_node \*Node; typedef struct \_queue \*Queue; , опишем поля этих структур - поля у структуры узла будут – значение узла int value; и указатель на следующий узел struct \_node\* next\_node;, а в структуре очереди будут два поля – указатель на первый узел очереди Node front; и на последний узел очереди Node last. Следует отметить, что тип данных Node – указатель на структуру Node, а тип данных Queue – указатель на структуру Queue. Также в заголовочном файле определим конструкцию enum для ее дальнейшего использования в опреторе switch в основном файле программы. Далее в файле для реализации функций необходимо будет реализовать функции. Создание очереди - create\_queue – функция, создающая пустую очередь и возвращающая указатель на себя, в теле функции выделяется память под структуру очереди и полям структуры очереди присваиваются пустые указатели. Вычисление размера очереди – getsize – функция принимает указатель на очередь, возвращает размер очереди, в теле функции происходит проход по всем узлам очереди – от первого к последнему, и их подсчёт. Проверка очереди на пустоту –isempty принимает указатель на очередь, возвращает true если очередь пуста, и false если не пуста, в теле функции происходит проверка поля структуры очереди front – головы очереди на нулевой указатель, если он таков, то очередь пуста – иначе нет. Добавление элемента в конец очереди – queue-push-back – функция типа void принимающая указатель на очередь и значение узла, который будет добавлен, в теле функции выделяется память под новый узел типа Node, затем сменяется значение хвоста очереди last – теперь новый узел становится последним. Извлечение и удаление элемента из начала очереди – queue\_pop\_front – функция принимает указатель на очередь, возвращает указатель на извлеченный узел, в теле функции создается временный указатель temp – указывающий на голову очереди, голове очереди front присваивается указатель на следующий, после первого узел, а временный указатель удаляется.

Печать очереди - print\_queue – функция типа void – принимает указатель на очередь, в теле функции происходит прозод по узлам очереди от первого к последнему и их вывод на экран, а также выводится размер очереди. Процедуры согласно варианту задания – procedure – функция типа bool – принимающая указатель на очередь , в теле функции происходит поиск первого от начала узла очереди, который меньше своего непосредственного предшественника и перетаскивание его назад до тех пор, пока он не станет первым или меньше своего предшественника, с использованием временного указателя this\_node, так как в очереди нет ссылки на предыдущий узел, то узел this\_node в данной функции отвечает за сохранение значения предыдущего узла для его сравнения с рассматриваемым, затем в случае нахождения с помощью временного указателя происходит обмен значениями предыдущего узла и нынешнего, если узел, удовлетворяющий условию был найден и прошел процедуру перетаскивания по условию, то функция возвращает true, если же он не был найден, а все узлы очереди до последнего были рассмотрены, то возвращает false. Cортировка согласно варианту задания – Insertsort – функция – вызывающая процедуру procedure, пока есть узлы, подлежащие перетаскиванию т.е. пока очередь не отсортируется. Удаление очереди - delete\_queue – принимает указатель на очередь, в теле функции происходит проход по узлам очереди от первого к последнему и очищение памяти, выделенной под них с помощью создания и удаления временного указателя temp, указывающего на каждый из узлов.

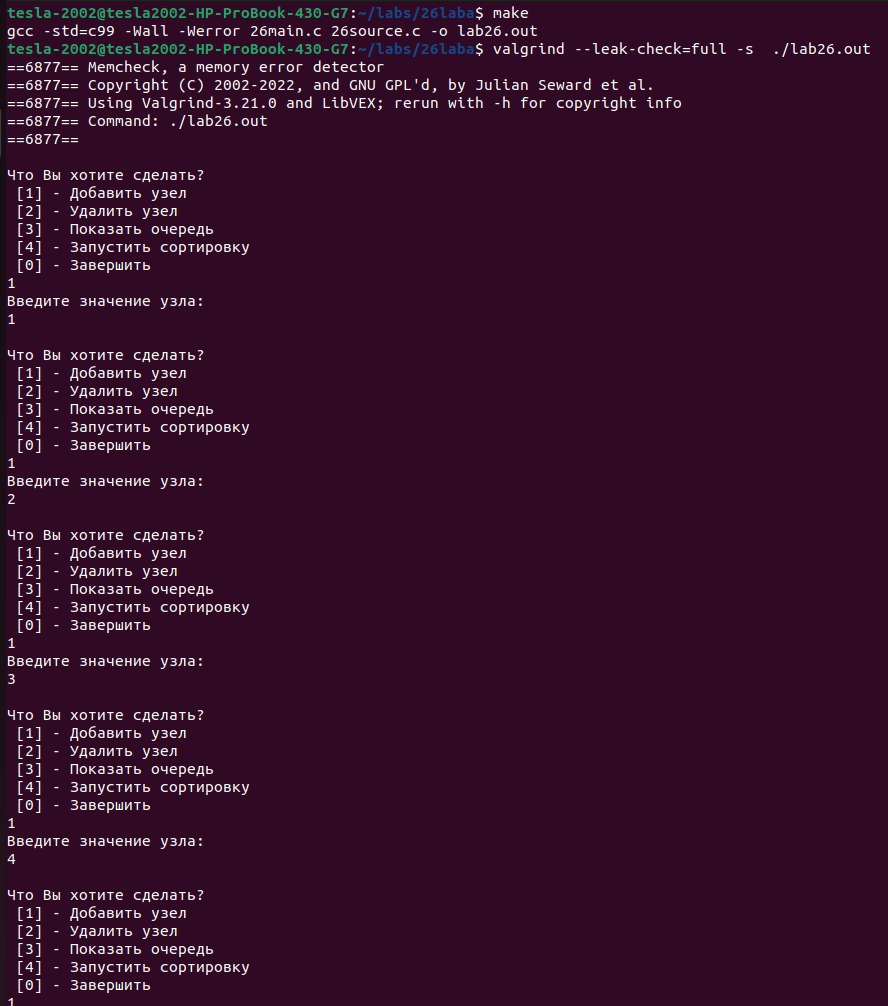
Перейдем главному файлу программы – в функции main сделаем контекстное меню, где пользователь сможет произвести действия с очередью по выбору – добавление узла в очередь, удаление узла, печать очереди, сортировка очереди, сделать это можно например конструкцией switch case в цикле while, и считывать выбор пользователя, пока он не захочет завершить работу с очередью, и после этого вывести окончательный вид очереди.

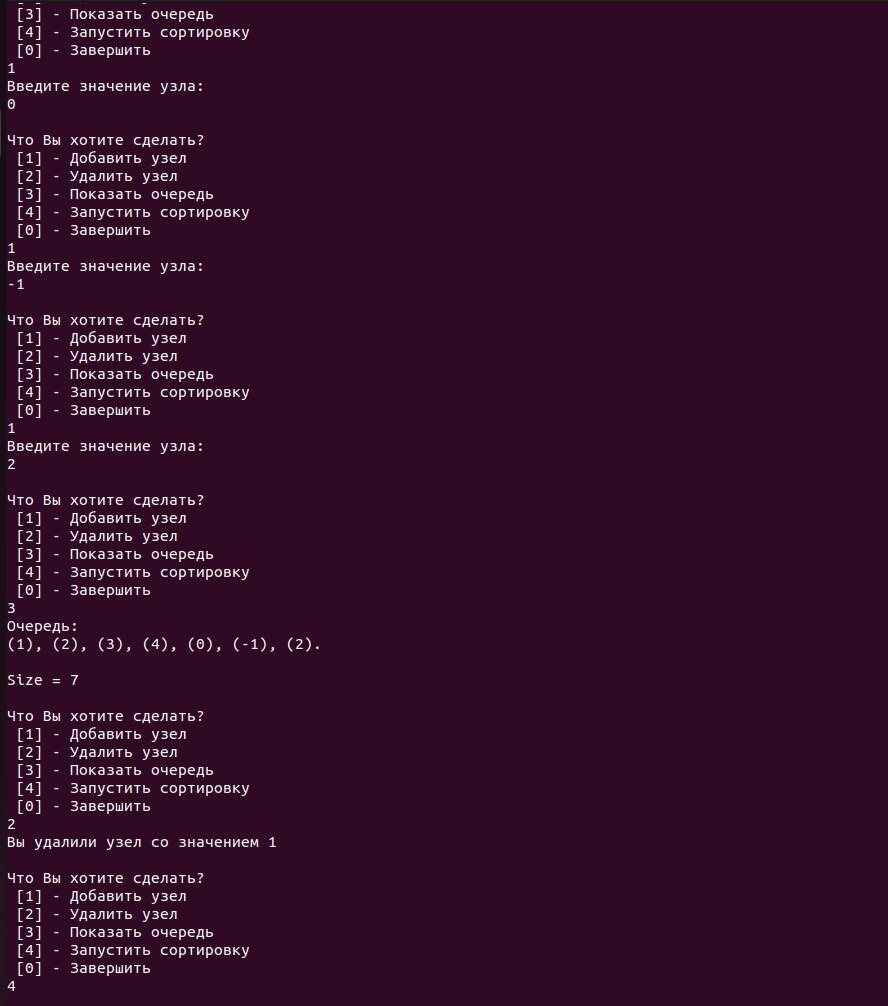
*Пункты 1-7 отчета составляются строго до начала лабораторной работы.*

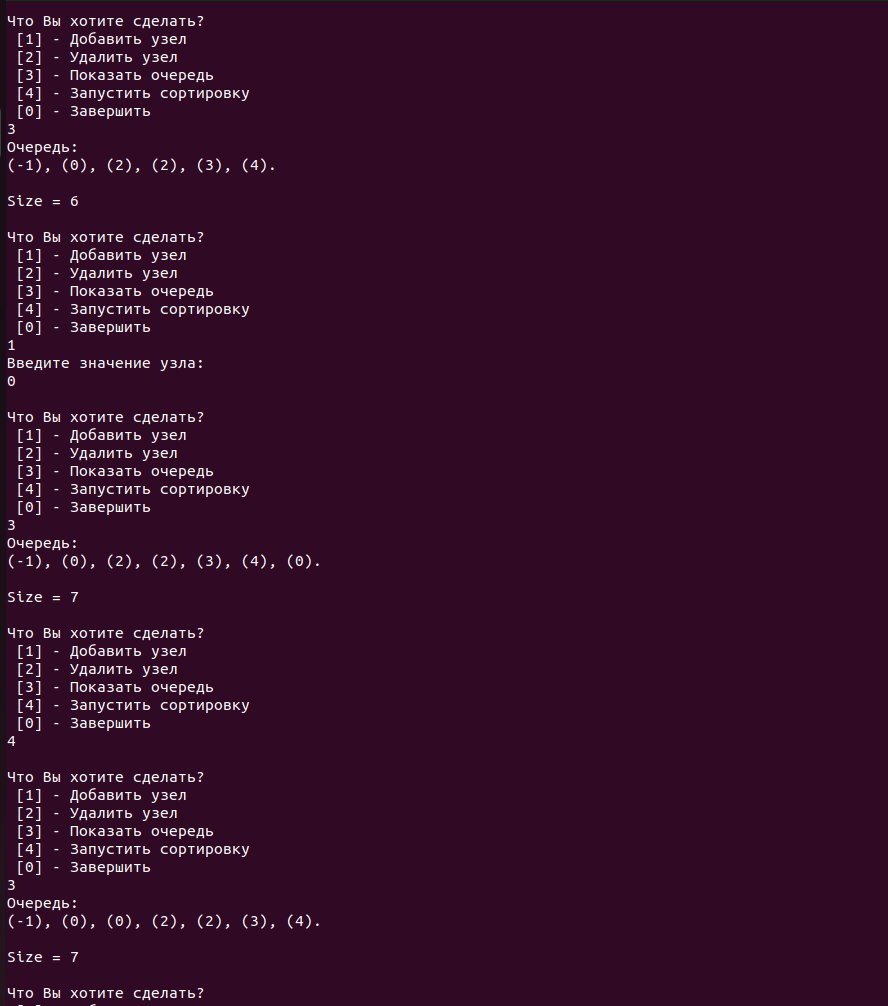
*Допущен к выполнению работы.* **Подпись преподавателя****\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

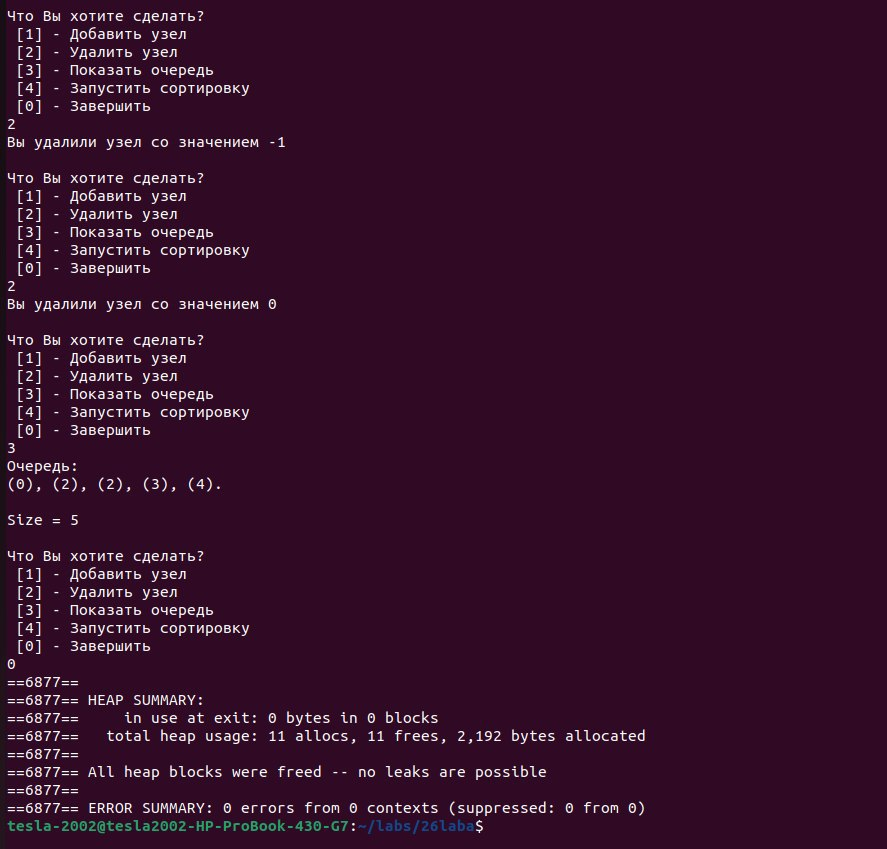
1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами,подписанныйпреподавателем).

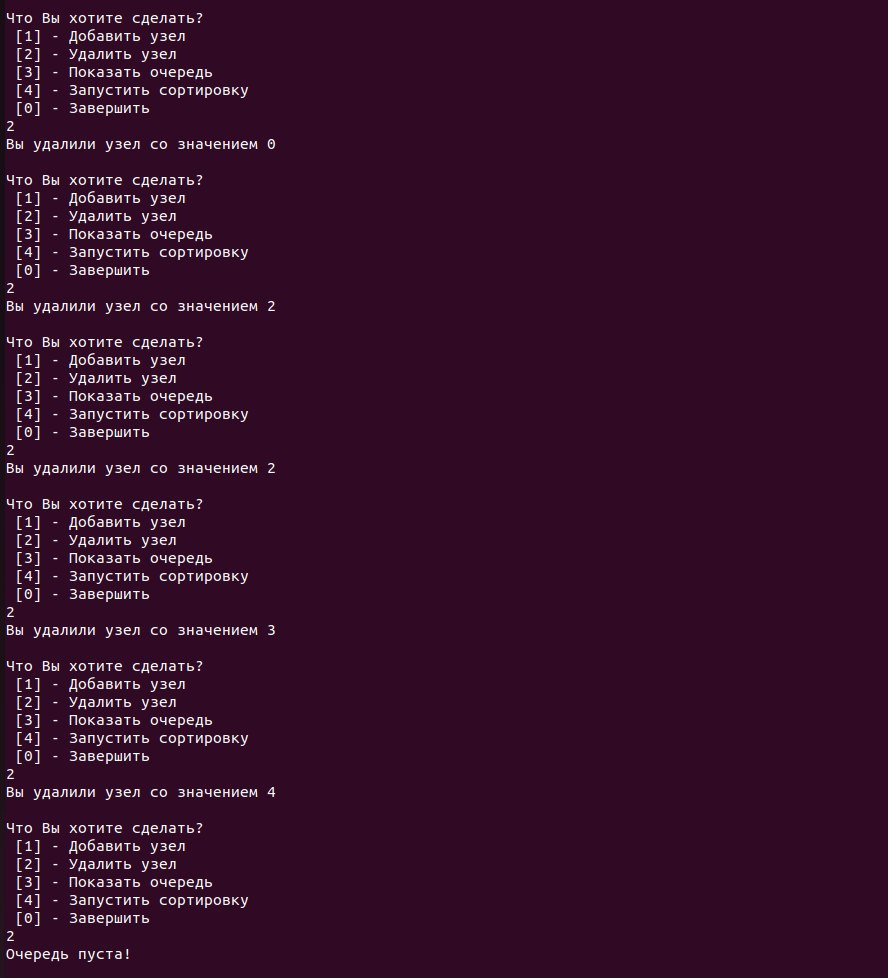
**Ссылка для просмотра кода:** https://github.com/tesla-2002/lab26

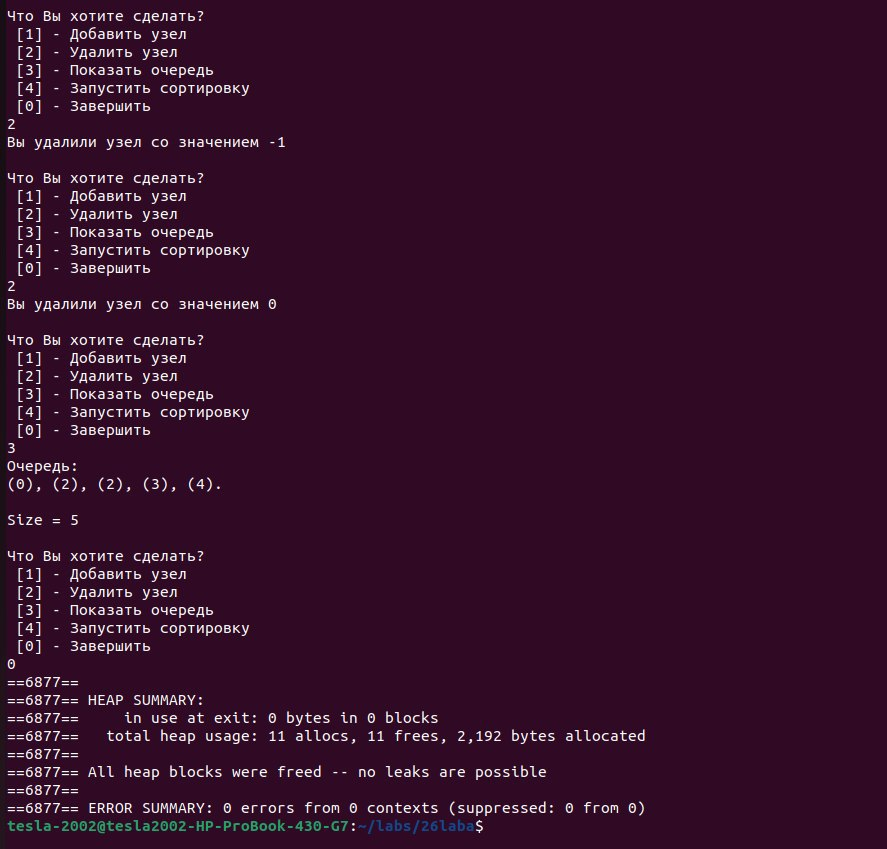












1. **Дневник отладки** должен содержать дату и время сеансов отладки, и основные события(ошибки в сценарии и программе,нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  | или |  |  |  |  |  |
|  | дом. |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Замечания автора** по существу работы

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

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

1. **Выводы**

\_В ходе выполнения данной лабораторной работы я изучил и освоил на практике такой абстрактный тип данных на языке Си , как очередь, представленную как односвязный список, изучил и применил в своей программе основные методы обработки очереди – добавление узла в конец, извлечение узла из начала очереди, поиск элемента в очереди, печать очереди, очистка памяти , выделенной под очередь. Также в данной лабораторной работе освоил один из вариантов метода сортировки вставками, применил его для очереди. Кроме того в данной лабораторной работе я научился создавать программы модульной структуры на Си, и автоматизировать сборку таких программ используя утилиту make и файл с инструкциями для этой утилиты. Все полученные мной знания и навыки пригодятся в выполнении лабораторных и курсовых работ в будущем.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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