**Отчет по лабораторной работе №26** по курсу алгоритмы и структуры данных

Студент группы М8О-101БВ-24 Закревский Владислав Анатольевич, № по списку 10

Контакты e-mail

Работа выполнена: «25.05» 2025г.

Преподаватель: \_\_\_\_\_\_\_ каф. 806 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

1. **Тема:** Абстрактные типы данных. Рекурсия. Модульное программирование на языке Си.

1. **Цель работы:** изучить абстрактные типы данных, рекурсию и модульное программирование на языке Си. Разработать модули для работы с деком и реализовать сортировку методом пузырька, используя только операции из модуля односвязного списка
2. **Задание:** разработать и отладить модуль определений и модуль реализации для односвязного списка, хранящего числа. Составить программный модуль, сортирующий список методом пузырька, используя только операции, импортированные из модуля односвязного списка.
3. **Оборудование** (лабораторное):

ЭВМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ , процессор \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, имя узла сети \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ с ОП \_\_\_\_\_\_\_\_\_\_ Мб,

НМД \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мб. Терминал \_\_\_\_\_ адрес \_\_\_\_\_\_\_\_\_\_. Принтер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

*Оборудование ПЭВМ студента, если использовалось:*

Процессор \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ с ОП \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ГБ, НМД \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ГБ. Монитор \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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 \_\_\_\_\_\_\_\_\_\_, 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 на домашнем компьютере \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

В основе данной работы лежит реализация классического односвязного линейного списка (single-linked list) с основными операциями — добавление в конец (push\_back), получение элемента по индексу (list\_get) и сортировка (list\_bubble\_sort). Реализация ведётся на языке программирования C, что позволяет глубже понять управление памятью на низком уровне, работу с динамическими структурами данных и принципы построения абстрактных типов данных.

Организация структуры данных

Односвязный список представляет собой упорядоченную последовательность элементов (узлов), каждый из которых хранит некоторое значение (key) и указатель на следующий элемент. Такой подход обеспечивает динамический характер структуры: она может расти или уменьшаться в зависимости от необходимого объёма данных, при этом отсутствуют ограничения, связанные с размером буфера.

Каждый элемент списка реализован как структура node, содержащая две составляющих — поле key, хранящее непосредственно числовое значение, и next, являющееся указателем на следующий элемент списка либо NULL в случае последнего узла. Управление всей структурой осуществляется посредством отдельной структуры list, включающей указатель на начало списка (head) и дополнительное поле size, содержащее текущий размер структуры. Такая организация облегчает доступ к информации о количестве элементов и обеспечивает удобное управление памятью.

Ввод и построение списка

Пользовательский ввод реализован через стандартный поток ввода с использованием функции scanf. Программа циклически запрашивает ввод новых значений до момента достижения конца файла (EOF). Каждый новый элемент добавляется в конец списка с помощью функции list\_push\_back, реализующей проход до последнего узла с последующим присоединением нового элемента. Такой подход сохраняет порядок следования входных данных, обеспечивает гибкость и отсутствие ограничений на размер вводимой последовательности.

Сортировка методом "пузырька"

Для сортировки элементов используется один из простейших и наиболее интуитивно понятных алгоритмов — пузырьковая сортировка (bubble sort). Особенностью данной реализации является то, что перестановке подвергаются исключительно значения key внутри узлов, а не указатели next. Это исключает необходимость сложных манипуляций с адресами памяти и предотвращает риски возникновения ошибок, связанных с нарушением целостности списка при перестроении.

Сортировка построена на двух вложенных циклах: внешний контролирует число проходов по списку, внутренний сравнивает элементы попарно и производит обмен значениями, если они идут в неверном порядке (по возрастанию или убыванию). Несмотря на низкую эффективность пузырькового алгоритма по времени (O(n^2)), его простота и наглядность делают этот метод подходящим для демонстрационных целей и учебных заданий.

Визуализация данных

Успех восприятия структуры данных во многом зависит от средств её представления. В рамках данной работы реализованы два вида визуализации:

* Текстовое строковое отображение — список представляется в привычном виде [a1, a2, ..., an], где a\_i — значения элементов списка. Данный формат позволяет быстро оценить порядок следования и содержимое без необходимости анализировать сложные конструкции.
* Графическое отображение — гистограммы, формируемые посредством функции plot\_data. Эта функция строит "столбчатую" диаграмму по значениям элементов списка, что даёт наглядное представление о распределении и изменении данных в результате сортировки. Использование цветов и разметки делает изображение наполненным и понятным.

Работа с памятью

Особое внимание уделяется ручному управлению динамической памятью: при добавлении каждого нового элемента используется malloc, а по завершении работы применятся последовательное освобождение памяти с помощью free для каждого узла списка. Это предотвращает утечки памяти и способствует выработке аккуратного стиля программирования.

Разделение на модули

Программа структурирована посредством функционального разделения: определение структур и объявление функций вынесено в заголовочный файл, вспомогательные и сервисные процедуры — в отдельную реализационную часть, точка входа и логика взаимодействия с пользователем — в основном файле. Это способствует облегчению сопровождения и расширению программы, а также повышает её читаемость.

Корректность и обработка исключений

Дополнительно реализована обработка случаев отсутствия элементов (пустого списка), обработка ошибок выделения памяти и завершение работы с возвращением всех ресурсов системе, что соответствует стандартам надёжного программного обеспечения.

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

План работы

Работа строится последовательно, каждый этап направлен на формирование законченного и надежного программного продукта. Общий сценарий включает в себя следующие шаги:

* Постановка задачи и проектирование структуры данных.

На первом этапе анализируется задание и формируются требования к будущей программе. Решается, какие структуры и функции необходимы для реализации списка, хранения данных, сортировки и визуализации. Делается выбор в пользу односвязного списка, разрабатывается его структурная схема.

* Реализация базовых операций со списком.

Далее реализуется каркас односвязного списка: функции инициализации списка, добавления элементов в конец (push\_back), получения размера списка (list\_size), доступа к элементам по индексу (list\_get), а также освобождения памяти (list\_free). Проверяется корректность работы каждого этапа путем запуска промежуточных тестовых фрагментов.

* Реализация пользовательского ввода и построения списка.

Программа снабжается возможностью принимать произвольное количество числовых значений с клавиатуры или через подачу данных со стандартного ввода. По каждому введенному числу вызывается функция добавления в список.

* Реализация сортировки списка пузырьковым методом.

Имплементируется функция сортировки, в которой значения узлов переставляются местами в ходе попарных сравнений. Особое внимание уделяется предотвращению ошибок при работе с указателями, особенно для крайних случаев (пустой список, единственный элемент).

* Реализация визуализации списка.

Для повышения наглядности разрабатываются две функции вывода:

Текстовое представление списка: [a1, a2, ...].

Графическая функция plot\_data — для рисования гистограммы по высоте значений элементов. Тестируется поведение функций на различных входных данных.

* Организация основного цикла работы, вывод результатов до и после сортировки.

Основной сценарий предполагает: ввод данных, отображение двух видов представления до сортировки, сортировку, повторное отображение, завершение работы с полным освобождением памяти.

* Управление памятью и обработка ошибок.

В ходе разработки проводится тестирование на корректность работы с памятью (проверки с помощью инструментов типа valgrind), обработку граничных ситуаций (отсутствие элементов, ввод некорректных значений).

Тестирование и проверка работы

Тестирование охватывает различные аспекты, включая правильность работы с последовательностями разной длины, крайние случаи и корректность сортировки. Примеры тестов:

* Пустой ввод: программа должна корректно опознать отсутствие элементов и не совершать ошибочных операций.
* Один элемент: список должен отображаться корректно, а сортировка не повлияет на порядок.
* Повторяющиеся значения: программа должна корректно их обработать и корректно вывести, без смешения позиций.
* Упорядоченные и обратные последовательности: Проверка самой сортировки на готовых отсортированных или обратно-отсортированных наборах.
* Произвольный порядок и наличие отрицательных/нулевых/положительных чисел: Воспроизводятся различные графические варианты.
* Граничные значения (Int\_MAX, Int\_MIN): Проверка устойчивости к крайним случаям.

При проведении тестов особое внимание уделяется тому, чтобы программа не допускала утечек памяти, не выходила за границы допустимых диапазонов и корректно освобождала все ресурсы. Проверяются возможные исключительные ситуации обработки ошибочного ввода. В качестве инструментов контроля используются вывод диагностической информации и статические анализаторы.

Итоговый сценарий работы пользователя

* Запуск программы.
* Ввод (вручную или через файл/конвейер) последовательности чисел.
* Просмотр списка в формате [a1, a2, ...] и соответствующей гистограммы до сортировки.
* Просмотр результатов после выполнения сортировки (текстовое и графическое представление).
* Завершение работы без ошибок, утечки памяти полностью предотвращены.

*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)

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. или дом. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  |  |  |  |  |  |  |

1. **Замечания автора** по существу работы

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

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

\_\_\_\_\_

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

1. **Выводы**

В ходе выполнения работы были исследованы, реализованы и проанализированы основные методы работы с динамическими структурами данных на языке программирования C. Получены практические навыки построения односвязного линейного списка, включая операции добавления, выбора и сортировки элементов, а также освобождения динамической памяти. На реальном примере закреплены принципы применения простейших алгоритмов сортировки, что позволило наглядно видеть их недостатки и преимущества в условиях ручного управления структурами.

Важной частью стало внедрение механизмов визуализации, позволивших не только анализировать содержимое и порядок следования элементов, но и оценить изменения, происходящие в ходе сортировки по графическому представлению (гистограмме). Такой подход улучшает интуитивное восприятие работы алгоритмов и способствует лучшему пониманию их внутреннего устройства.

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

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

\_\_\_\_\_

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

\_\_\_\_\_

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

\_\_\_\_\_

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

\_\_\_\_\_

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

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