**Отчет по лабораторной работе №1(9)**  по курсу алгоритмы и структуры данных

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

Контакты e-mail

Работа выполнена: « » 202 г.

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

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

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

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

1. **Тема:** Сортировка и поиск

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

1. **Цель работы:**Цель данной работы заключается в практическом освоении алгоритмов сортировки и поиска по ключу в структуре данных с использованием сложных (составных) ключей. В ходе лабораторной работы необходимо реализовать и протестировать сортировку массива структур по составному ключу и организовать двоичный поиск по этому ключу, продемонстрировав корректность работы на различных исходных данных. Дополнительной целью является развитие навыков анализа поведения алгоритмов в различных сценариях (отсортированный массив, обратно-отсортированный, случайный порядок

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

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

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

1. **Задание** (*вариант №***1(9)**)**:**Разработать программу на языке C, которая:

* Оперирует структурой данных, элемент которой содержит составной ключ (две целые части) и строку текста.
* Имеет возможность заполнения массива таких структур тремя способами: исходный отсортированный по ключу порядок, обратный порядок, случайный порядок.
* Реализует алгоритм сортировки выбором (selection sort) для сортировки массива по составному ключу.
* Выполняет двоичный поиск по составному ключу в уже отсортированном массиве.
* Позволяет пользователю вводить ключ для поиска и выводит найденный элемент, если он есть, или сообщает о его отсутствии.
* Ведёт наглядный вывод массива до и после сортировки, а также результаты поиска.
* Корректно работает с русскими строками и поддерживает задания массива строк (например, стихотворение Пушкина).

1. **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 этой лабораторной работе рассматривается задача сортировки и поиска в массиве структур с составным ключом. Каждый элемент массива (Record) содержит ключ, состоящий из двух целых чисел (part1 и part2), и сопровождающий их текст. Основная трудность заключается в необходимости корректно сравнивать составные ключи как один логический объект, а также обеспечить устойчивую корректную работу алгоритмов сортировки и поиска для различных исходных состояний массива.

Для сортировки используется один из простейших и классических алгоритмов сортировки — сортировка выбором (selection sort). Принцип работы данного алгоритма основан на последовательном выборе минимального (с точки зрения ключа) элемента из неотсортированной части массива и обмена его с первым элементом этой части. Таким образом, каждый проход по массиву уменьшает размер неотсортированной части на один, а в результате весь массив оказывается отсортированным.

Сравнение ключей реализовано в отдельной функции compare\_keys, которая сравнивает сначала первую часть ключа (part1), а если они равны — вторую (part2). Такой подход оказывается полезным, так как в реальных задачах ключи часто состоят из нескольких полей, и требуется однозначный способ их сравнения.

Для поиска элемента по ключу после сортировки применяется алгоритм двоичного поиска (binary search), что существенно ускоряет работу по сравнению с линейным поиском — особенно на больших объемах данных. Двоичный поиск основан на делении отсортированного массива пополам и каждом шаге определяет, где следует искать ключ — в левой или правой части массива. Для этого вновь используется функция сравнения составных ключей.

Заполнение массива производится тремя способами: отсортированный порядок (инкремент от 0), обратный порядок (декремент от N-1 до 0) и случайный порядок (случайные значения частей ключа). Это позволяет проверить, как ведет себя алгоритм сортировки при различных исходных условиях — известно, что эффективность selection sort мало зависит от начального состояния массива, но на практике стоит в этом убедиться.

Для пользователя организован диалог поиска: после сортировки таблицы программа ожидает ввода ключа, по которому выполняет поиск и выдает результат. Наряду с этим оформлен подробный вывод массивов до и после сортировки для визуального контроля.

Особое внимание уделяется обработке ввода: программа строго ожидает два целых числа как ключ и выводит сообщения об отсутствии найденных элементов. Логика работы программы рассчитана на расширение — всегда можно заменить selection sort на другой алгоритм, или реализовать более сложные типы ключей.

Тестирование алгоритма проводится на трех сценариях (отсортированный, обратный и случайный порядок) и подразумевает попытки искать как заведомо существующие (например, те, что были в изначальных таблицах), так и отсутствующие ключи (например, за пределами диапазона заполнения или с несуществующими комбинациями частей ключа).

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

Выполнение работы осуществляется по следующему плану:

* Подготовка и запуск программного кода.

Компиляция и Запуск

* Тестирование на отсортированном массиве.

Программа заполняет массив структур в возрастающем порядке — то есть, ключи идут от (0,0), (1,2), (2,4) и так далее. Сначала выводится этот массив, далее он проходит через функцию сортировки selection sort, и выводится снова — для контроля корректности сортировки в изначально отсортированном массиве. Ожидается, что результат после сортировки не изменится, что подтверждает корректность работы алгоритма.

* Тестирование на обратно отсортированном массиве.

На втором этапе массив заполняется в обратном порядке — (18,36), (17,34), ... (0,0). После этого снова запускается сортировка, и снова делается вывод до и после — ожидается, что алгоритм приведет массив к возрастающему порядку, идентичному первому тесту.

* Тестирование на случайном массиве.

Следующим этапом производится заполнение массива случайными значениями составного ключа. Выводится массив, проводится сортировка, затем выводится отсортированный массив для наглядности — ожидается, что все элементы будут отсортированы по ключам.

* Поиск по пользовательскому вводу.

После сортировки программа приглашает пользователя ввести ключи (две целые части). Пользователь может поочередно искать ключи:

* + заведомо существующие (например, те, которые были выведены ранее, найти для них строки из стихотворения);
  + заведомо отсутствующие (например, части сильно за пределами диапазона, или несуществующие комбинации);
  + пограничные случаи, совпадающие только по одной части ключа (одинаковый part1, но разный part2). Каждый раз программа должна либо выдать найденный элемент, либо сообщить об отсутствии.
* Анализ результатов.

По итогам работы анализируется, как ведет себя алгоритм сортировки на разных типах исходных данных:

* + изменился ли порядок элементов (например, для отсортированного ничего не поменялось, для обратного он "починил" порядок, для случайного — аккуратно все отсортировал).
  + Анализируется поведение поиска: как быстро и корректно находится нужная запись, не возникает ли ошибок при вводе, корректно ли программа сообщает об отсутствии элементов.
* Дополнительные тест.

Для полноты проверки можно варьировать размеры массива, либо модифицировать наполнением строк (можно добавить проверку на длинные строки, граничные ситуации).

Также возможно изменить алгоритм сортировки (например, на пузырьковую или быструю) и сравнить поведение.

Соображения по тестированию:

Обязательно тестировать сценарии поиска:

* По первому, последнему и среднему ключу отсортированного массива.
* По заведомо несуществующему ключу.
* По одинаковой первой части ключа, но разной второй.
* Проверить обработку некорректного ввода (ввести только одно число или строку вместо чисел).
* Оценить устойчивость программы к повторяющимся ключам (если заполнить массив с дубликатами).
* Проверить, что сортировка действительно приводит массив к однозначному порядку по ключам независимо от исходного состояния данных.

*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 лабораторной работы были реализованы и протестированы базовые алгоритмы сортировки и поиска в массиве структур с составным ключом. Практическая часть показала, что сортировка выбором корректно и однозначно упорядочивает элементы по сложному ключу вне зависимости от начального состояния данных. Двоичный поиск, реализованный для составных ключей, позволяет эффективно искать элементы в отсортированном массиве и корректно определяет, найден ли элемент, соответствующий введенному пользователем ключу.

Тестирование на разных сценариях (отсортированные, обратно отсортированные и случайные данные) показало предсказуемое поведение алгоритмов и подтвердило их работоспособность на практике. Дополнительные тесты на различные комбинации ключей дали понять, насколько важно четкое определение функции сравнения для составных ключей.

В процессе работы были закреплены навыки структурирования кода, написания функций сравнения и сортировки, проведения комплексного тестирования программ, рассматривающих сложные типы ключей. Работа развила навыки системного анализа и дала понимание того, как реализуются базовые алгоритмы поиска, сравнения и сортировки в реальных прикладных задачах с составными структурами данных.

В заключение, поставленные задачи выполнены в полном объеме: разработан, реализован и протестирован программный комплекс, позволяющий уверенно сортировать и искать структуры с составным ключом, а также выявлены основные подходы к тестированию и анализу таких программ.

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

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

\_\_\_\_\_

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

\_\_\_\_\_

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

\_\_\_\_\_

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

\_\_\_\_\_

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

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