|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 «Записи с вариантами. Обработка таблиц» (ВАР. 7)**

Студент Смирнов Иван Владимирович

Группа ИУ7 – 32Б

Проверил Барышникова Марина Юрьевна

*2023 г.*

Оглавление

[Описание условия задачи 3](#_Toc147219892)

[Описание технического задания 3](#_Toc147219893)

[Описание внутренних структур данных 5](#_Toc147219894)

[Описание алгоритмов сортировок и ресурсных затрат 7](#_Toc147219895)

[Тесты 8](#_Toc147219896)

[Ответы на контрольные вопросы 11](#_Toc147219897)

[Вывод 13](#_Toc147219898)

# Описание условия задачи

Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами (объединениями)). Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ – любое невариантное поле (по выбору программиста), используя: а) саму таблицу, б) массив ключей. (Возможность добавления и удаления записей в ручном режиме обязательна). Ввести список абонентов, содержащий фамилию (20 англ. букв), имя (20 англ. букв), телефон (формат: +{11 цифр}), адрес (улица (20 англ. символов), дом), статус (вариантное поле: личный – дата рождения: день, месяц, год; служебный – должность, организация). Найти всех друзей, которых необходимо поздравить с днем рождения в ближайшую неделю (в качестве даты может быть выбрана текущая или она может быть введена по запросу).

Описание технического задания

1. **Исходные данные и результат**

**Входные данные:**

Перед пользователем появляется меню, в котором можно выбрать номера пунктов от 1 до 11. Для использования некоторых пунктов, пользователю необходимо ввести дополнительную информацию.

**Выходные данные:**

1. На экран выводятся результаты команд определённых пунктов меню (например, выводится таблица абонентов или таблица ключей).
2. **Задачи, реализуемой программой**

**Программа считывает из файла информацию об абонентах и записывает их в массив структур. Также создается массив ключей. Далее пользователю необходимо взаимодействовать с меню, выбирая различные пункты:  
Menu:**

**|1. Вывести таблицу**

**|2. Добавить запись в таблицу**

**|3. Удалить запись из таблицы (по индексу)**

**|4. Отсортировать по Фамилии**

**|5. Отсортировать по Имени**

**|6. Отсортировать по номеру Телефона**

**|7. Отсортировать по Адресу**

**|8. Сравнить сортировку массива записей и массива ключей**

**|9. Найти друзей, которых необходимо поздравить в течение недели**

**|10. Вывести таблицу ключей**

**|11. Выйти из программы**

**Способ обращения к программе**

**Программа запускается через терминал.**

1. **Возможные аварийные ситуации и ошибки пользователя**
2. **Пользователь ввел строку в качестве пункта меню**

**На выходе сообщение: «**Неправильный ввод пункта меню!**»**

1. **Пользователь ввел отрицательный пункт меню.**

**На выходе сообщение: «**Пункт меню должен быть неотрицательным!**»**

1. **Пользователь ввел целое или вещественное число, не следуя указанному формату.**

**На выходе сообщение: «**Вы выбрали несуществующий пункт меню!**)»**

1. **Пользователь ввел пустое поле**

**На выходе сообщение: «**Ошибка при вводе (ничего не введено).**»**

1. **Пользователь ввел много символов**

**На выходе сообщение: «**Ошибка при вводе (введено слишком много символов).»

1. **Пользователь ввел телефон не по указанному формату**

**На выходе сообщение: «**Некорректный номера телефона.»

1. **Пользователь ввел неизвестный тип статуса**

**На выходе сообщение: «**Ошибка: некорректный тип статуса.**»**

1. **Пользователь пытается добавить запись в заполненный массив записей**

**На выходе сообщение: «Количество абонентов максимально! (нельзя добавить)»**

1. **Пользователь ввел символы вместо индекса**

**На выходе сообщение: «**Неправильный ввод индекса!**»**

1. **Пользователь ввел несуществующий индекс**

**На выходе сообщение: «**Вы выбрали несуществующий индекс!**»**

1. Пользователь ввел несуществующий метод сортировки

На выходе сообщение: «Вы выбрали несуществующий тип сортировки!»

# **Описание внутренних структур данных**

Информация о каждом из абонентов хранится в структуре Abonent:

В данной структуре присутствуют: фамилия как массив из 20 char, имя – массив из 20 char, номер телефона – массив из 12 char. Также хранится улица – массив из 20 char, номер дома типа int в подструктуре Address. Информация о типе статуса хранится в StatusType, а поля разных типов хранятся в объединении Status.

|  |
| --- |
| // Структура для адреса struct Address { char street[OTHER\_INFO\_LEN]; int house\_number; };  // Объединение для статуса union Status { struct { int day; int month; int year; } personal; struct { char position[OTHER\_INFO\_LEN]; char organization[OTHER\_INFO\_LEN]; } professional; };  typedef enum { PERSONAL, PROFFESSIONAL } StatusType;  // Структура для абонента typedef struct { char last\_name[MAX\_NAME\_LEN]; char first\_name[MAX\_NAME\_LEN]; char phone[MAX\_PHONE\_LEN]; struct Address address; StatusType status\_type; union Status status; } Abonent; |

Для структуры Abonent в качестве статуса был выбран вариантный тип данных. Это позволило сэкономить память на обработке различных типов данных, так как union занимает такое количество байт, которое занимает вариант, занимающий наибольшее количество байт среди всех вариантов. Также из-за вариантой части в программе можно сделать больше функций (например, как в задании: вывести всех абонентов с личным статусом, которых можно поздравить в течение недели).

# Сравнение времени работы и ресурсных затрат сортировок

При сортировке записей использовались 2 алгоритма сортировки (сортировка “расческой” и быстрая сортировка) и 2 подхода (метода) сортировки (сортировать таблицу, сортировать массив ключей). Пользователь может выбрать в программе, каким именно алгоритмом и каким именно методом необходимо отсортировать записи. В программе предусмотрен сравнительный анализ всех возможных вариантов сортировки записей.

|  |
| --- |
| Сравнение ресурсных и временных затрат сортировки абонентов:  Сравнение для (n = 18):  +---------+---------------------++-------------------++-------------++  | | Расческа || Быстрая ++-------------++  +---------+---------------------++-------------------++-------------++  | Таблица | 0.00000411с. 3888Б || 0.00000127с. 3888Б|| 69.09975669%||  +---------+---------------------++-------------------++-------------++  | Ключи | 0.00000286с. 4032Б || 0.00000099с. 4032Б|| 65.38461538%||  +---------+---------------------++-------------------++-------------++  |---------| 30.41362530% || 22.04724409% || 75.91240876%||  +---------+---------------------++-------------------++-------------++ |

Для каждого из 4 видов сортировки производилось 100 замеров. В таблице указано среднее время от всех 100 замеров данного вида.

В таблице указано время выполнения сортировки заданным методом и заданным алгоритмом, а также использованная для этого память. Отдельно по прямым линиям расположены проценты эффективности того или иного метода перед предыдущим. Так, использование алгоритма быстрой сортировки, вместо “расчески”, для сортировки самой таблицы эффективнее на 62.5%, однако при этом под массив ключей выделяется дополнительная память. Сортировать массив ключей вместо таблицы алгоритмом “расчески” эффективнее на 50%, так как при замене значений на этапе сортировки в массиве ключей необходимо заменить одно поле int, в то время как в таблице придется поменять все поля структуры.

А если взять хороший алгоритм и метод сортировки массива ключей, то эффективность – 75%. И при всем при этом данная статистика показана только для n = 18 записей. При больших количествах записей эффективность вырастет в разы.

# Тесты

В следующих тестах символ «>» обозначает ввод пользователем пункта меню.

1. **Позитивные тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Входные данные | Выходные данные | Что проверяется |
| pos\_01 | >1 | -Таблица записей- | Умеет ли программа выводить записи в виде таблицы |
| pos\_02 | >2  Smirnov Ivan  +79689542522  Alabama St.  10  personal  25  2  2004  >11 | Сообщение: «Запись успешно добавлена в таблицу» | Умеет ли программа проверять корректность данных о записи, введенных пользователем и добавить их в таблицу |
| pos\_03 | >3  0  >11 | Сообщение: «Запись успешно удалена!» | Умеет ли программа удалять записи из таблицы по индексу |
| pos\_04 | >4  1  1  >11 | Сообщение: «Массив абонентов успешно отсортирован!» | Умеет ли программа сортировать записи по массиву абонентов (таблице) |
| pos\_05 | >4  1  2  >11 | Сообщение: «Массив ключей успешно отсортирован!» | Умеет ли программа сортировать записи по массиву ключей |
| pos\_06 | >8  >11 | -Сравнительный анализ алгоритмов и методов сортировки записей- | Умеет ли программа замерять время выполнения каждого из способов сортировки и на его основе выдавать сравнительные характеристики (эффективность) |
| pos\_07 | >9  >11 | -Все абоненты, у которых день рождения в течение недели- | Умеет ли программа вычислять, у каких абонентов день рождения в течение недели (7 дней от текущей даты) |
| pos\_08 | >10  >11 | -Таблица ключей- | Умеет ли программа выводить таблицу ключей отдельно от таблицы записей |
| pos\_09 | >11 | Сообщение: «До встречи!» | Умеет ли программа завершать меню и тем самым себя |

1. **Негативные тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Входные данные | Выходные данные | Что проверяется |
| neg\_01 |  | Сообщение: «Ошибка при вводе (ничего не введено).» | Пустой ввод |
| neg\_02 | >-1 | Сообщение: «Пункт меню должен быть неотрицательным!» | Правильность ввода пункта меню |
| neg\_03 | >100 | Сообщение: «Вы выбрали несуществующий пункт меню!)» | Правильность ввода пункта меню |
| neg\_04 | >2  (пустой ввод в любом поле) | Сообщение: «Ошибка при вводе (ничего не введено).» | Пустой ввод |
| neg\_05 | >2  (переполнение любой строки) | Сообщение: «Ошибка при вводе (введено слишком много символов).» | Правильность ввода строковых полей для записи |
| neg\_06 | >2  (ввод символа вместо цифр в численных типах) | Сообщение: «Ошибка при вводе (введено не число).» | Правильность ввода численных полей для записи |
| neg\_07 | >2  (ввод неправильного статуса) | Сообщение:  «Ошибка: некорректный тип статуса.» | Выбор существующего типа статуса |
| neg\_08 | >2  (ввод номера телефона не по формату) | Сообщение: «Некорректный номера телефона.» | Правильный формат введенного телефона |
| neg\_09 | >2  (статический массив переполнен) | Сообщение: «Количество абонентов максимально! (нельзя добавить)» | Превышение длины массива абонентов |
| neg\_10 | >3  (любой несуществующий индекс) | Сообщение: «Неправильный ввод индекса!» | В ходе деление получилось число, выходящее за границы |
| neg\_11 | >4  (выбор несуществующего алгоритма сортировки) | Сообщение: «Вы выбрали несуществующий тип сортировки!» | Проверяется выбор предусмотренного алгоритма сортировки |
| neg\_12 | >4  1  (выбор несуществующего метода сортировки) | Сообщение: «Вы выбрали несуществующий метод сортировки!» | Проверяется выбор предусмотренного метода сортировки |

# Ответы на контрольные вопросы

* 1. **Как выделяется память под вариантную часть записи?**

Выделение памяти под вариантную часть записи зависит от используемой структуры данных. Обычно вариантная часть записи выделяется как массив байтов и ее размер определяется заранее или в процессе создания структуры данных.

* 1. **Что будет, если в вариантную часть ввести данные, несоответствующие описанным?**

Если в вариантную часть ввести данные, несоответствующие описанным, то это может привести к ошибкам в работе программы. В зависимости от способа обработки вариантной части, это может привести к сбоям программы или некорректным результатам операций.

* 1. **Кто должен следить за правильностью выполнения операций с вариантной частью записи?**

За правильностью выполнения операций с вариантной частью записи должен следить программист или разработчик, который создал структуру данных и определил правила доступа к вариантной части. В некоторых случаях это может быть автоматизировано с использованием проверок и обработки ошибок в коде программы.

* 1. **Что представляет собой таблица ключей, зачем она нужна?**

Таблица ключей представляет собой структуру данных, которая содержит информацию о ключах (идентификаторах) и их связанных с ними данных. Она нужна для быстрого доступа к данным по ключу. Таблица ключей обычно используется в контексте баз данных или индексированных структур данных.

* 1. **В каких случаях эффективнее обрабатывать данные в самой таблице, а когда – использовать таблицу ключей?**

Эффективность обработки данных в самой таблице или использование таблицы ключей зависит от конкретной задачи и требований. Если нужно быстро выполнять операции поиска, сортировки и фильтрации данных, то таблица ключей может быть более эффективной, так как она обеспечивает быстрый доступ к данным по ключам. Однако, если данные редко изменяются, и требуется минимизировать использование памяти, то обработка данных непосредственно в таблице может быть предпочтительнее.

* 1. **Какие способы сортировки предпочтительнее для обработки таблиц и почему?**

Выбор способа сортировки зависит от типа данных, их объема и конкретных задач. Например, для небольших массивов данных может быть эффективно использовать сортировку вставками или сортировку пузырьком. Для больших объемов данных часто используют сортировку слиянием или быструю сортировку, так как они имеют более высокую производительность. Выбор оптимального метода сортировки требует анализа конкретной ситуации и тестирования разных вариантов.

# Вывод

В рамках задания были реализованы алгоритмы и методы­ сортировки записей (а на их основе сравнительный анализ) по невариантному полю, был реализован вывод таблицы записей и таблицы ключей. В программе реализован поиск абонентов по заданному условию, ручное удаление записи по позиции, а также добавление новой записи в конец.