|  |  |
| --- | --- |
| 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)** |

**Лабораторная работа №2**

**По дисциплине «Типы и структуры данных»  
  
Тема: «Записи с вариантами, обработка таблиц»**

**Вариант 9**

* + 1. Выполнил: студент группы ИУ7-36(Б)  
       Ковель Александр Денисович

**Москва, 2021**

**Цель работы**

приобрести навыки работы с типом данных «запись» («структура») содержащим

вариантную часть, и с данными, хранящимися в таблицах. Оценить относительную эффективность

программы (в процентах) по времени и по используемому объему памяти в зависимости от

используемого алгоритма и от объема сортируемой информации

**Задание**

Создать таблицу, содержащую не менее 40 записей с вариантной частью. Произвести поиск информации по вариантному полю. Упорядочить таблицу, по возрастанию ключей (где ключ – любое невариантное поле по выбору программиста), используя: а) исходную таблицу; б) массив ключей, используя 2 разны алгоритма сортировки (простой, ускоренный). Оценить эффективность этих алгоритмов (по времени и по используемому объему памяти) при различной реализации программы, то есть, в случаях а) и б). Обосновать выбор алгоритмов сортировки. Оценка эффективности должна быть относительной (в %).

Ввести список абонентов, содержащий фамилию, имя, телефон,

адрес, статус (личный – дата рождения: день, месяц, год;

служебный – должность, организация). Найти всех друзей, которых

необходимо поздравить с днем рождения в ближайшую неделю.

**Техническое задание**

**Общая концепция системы**

Программа предназначена для работы с таблицей абонентов. Таблица загружается из файла, затем с таблицей возможно произвести, требуемые от пользователя действия.

**Требования к функциональным характеристикам**

Программа должна выполнять следующие функции:

* Сортировать таблицу
* Добавлять новую строку в таблицу
* Удалить строку из таблицы
* Сообщить ошибку при неправильном вводе
* Выводить таблицу
* Отсортировать по именам
* Отсортировать по именам массив ключей
* Сравнить сортировки по ключам и сортировку обычную
* Сравнить быструю сортировку и пузырек
* Вывести друзей, у которых день рождения на этой неделе

**Вход**

На вход программа получает ключи для обработки запросов. При необходимости на вход так же поступают параметры поиска или новые данные, которые необходимо добавить в таблицу.

**Ограничения**

Имена, фамилии, адрес, телефон, организация и должность – это строки не больше 16 символов. День рождения это три числа: день – до 2 знаков и положительное число больше 0 и меньше 31, месяц – до 2 знаков и положительное число больше 0 и меньше 13, год – любое число до 4 знаков.

**Выход**

Выход должен быть представлен в том виде, которое требует действие. Это может быть напечатанная таблица, данные о результатах сортировки, отсортированная таблица, неотсортированная таблица, таблица ключей, результат удаления, добавления элемента или поиска.

Программа должна выдавать корректный ответ при вводе любых данных. Если найти или удалить элемент невозможно, она должна выдавать ошибку.

Программа завершается после того, как пользователь введет необходимое для завершения действие.

**Аварийные ситуации**

1. Ввод строки вместо числа.

Программа выведет сообщение “Input error”.

2. Ввод несуществующего пункта меню

Программа выведет сообщение “Input error”.

3. Ввод несуществующего индекса при удалении

Программа выведет сообщение “Input error”.

4. Ввод несуществующего действия при поиске

Программа выведет сообщение “Input error”.

**Способ обращения к программе**

Программа представляет собой файл app.exe. Запускается в консоли. для запуска достаточно команды ./app.exe. Если файл отсутствует, можно собрать его с помощью утилиты make.

**Описание структуры данных**

typedef struct

{

    int day; // День

    int month; // Месяц

    int year; // Год

} birthday; // Рождения

typedef struct

{

    char post[STR\_LEN]; // Должность

    char organ[STR\_LEN]; // Организация

} service;

typedef struct

{

    int index; // Индекс структуры

    char surname[STR\_LEN]; // Фамилия

    char name[STR\_LEN]; // Имя

    char phone[STR\_LEN]; // Телефон

    char adr[STR\_LEN]; // Адресс

    int status; // Статус (коллега или знакомый)

    union

    {

        birthday bd; // Знакомый

        service serv; // Коллега

    } info;

} people;

typedef struct

{

    int index; // Индекс массива ключей

    char name[STR\_LEN]; // Связка массива ключей и структуры

} key;

STR\_LEN = 17 символам

**Описание алгоритма**

Пока не выбрана команда выход:

1. Добавить строку в базу
   1. Ввести имя
   2. Ввести фамилию
   3. Ввести адрес
   4. Ввести телефон
   5. Ввести статус
   6. Если статус 1 (знакомый) – ввести день, месяц, год рождения
   7. Если статус 2 (коллега) – ввести должность и организацию
2. Удалить строку (-и)
   1. Пользователь вводит имя абонента
   2. Идет поиск подходящего имени
   3. Если имя найдено, сдвинуть всю базу на -1 одно строку, начиная с позиции найденного имени
   4. Уменьшить количество на один
3. Вывести таблицу
4. Отсортировать ключи по именам и вывести их
   1. Считать ключи, взяв индексы из базы
   2. Сортировать ключи по полю имени
   3. Вывести таблицу ключей
5. Отсортировать таблицу по именам и вывести ее
   1. Считать ключи, взяв индексы из базы
   2. Сортировать базу по именам
   3. Вывод базы
6. Отсортировать ключи и вывести базу
   1. Считать ключи, взяв индексы из базы
   2. Сортировать ключи по полю имени
   3. Вывести базу
7. Сравнить сортировку по ключам и сортировку базы
   1. Провести сортировку по ключам
   2. Провести сортировку базы
   3. Вывести процентную разницу
8. Сравнить быструю сортировку и сортировку пузырьком
   1. Провести быструю сортировку
   2. Провести сортировку пузырьком
   3. Вывести процентную разницу
9. Вывести друзей, у которых день рождения на этой недели
   1. Считать введенную дату
   2. Вычесть даты друг из друга, если один и тот же месяц, то разница по дням не должна превышать семи дней, или месяц отличается на единицу, тогда прибавляем количество дней в месяце и вычитаем. (В этой программе в каждом месяце 30 дней)
   3. Вывести друзей, если они нашлись, иначе вывести, что таких друзей нет
10. Выход

**Тесты**

|  |  |  |
| --- | --- | --- |
| Пункт | Ввод | Выход |
| 1 – выбор статуса | 0 | Input error |
| 1 – ввод имени | 01234567890123456 | Input error |
| 1 – ввод статуса | 0 | Input error |
| 1 – ввод дня рождения | один | Input error |
| 1 – выбор статуса | 1 | Введите имя |
| 1 – ввод имени | Alexander | Введите фамилию |
| 1 – ввод фамилии | Kovel | Введите адрес |
| 1 – ввод адреса | Moscow | Введите статус |
| 1 – ввод статуса | 1 | Введите день |
| 1 – ввод дня | 21 | Введите месяц |
| 1 – ввод месяца | 7 | Введите год |
| 1 – ввод года | 2002 | <Меню программы> |
| 2 | Alexander | Запись номер 1 удалена |
| 9 | 17 7 2021 | Kovel Alexander Moscow 21 7 2002 |
| 9 | 22 7 2021 | Вам некого поздравить |

**Оценка эффективности**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Bubble sort таблицы, такты | Bubble sort ключей, такты | Quick sort ключей, такты | Quick sort таблицы, такты |
| 50 | 110400 | 67840 | 23552 | 12850 |
| 100 | 400384 | 313120 | 58432 | 22280 |
| 1000 | 41502688 | 28876832 | 434336 | 240638 |

|  |  |  |  |
| --- | --- | --- | --- |
| N | Bubble sort таблицы и  Bubble sort ключей, % | Bubble sort ключей Quick sort ключей, % | Quick sort Таблицы и Bubble sort таблицы, % |
| 50 | 30.453 | 81.539 | 81.003 |
| 100 | 26.565 | 88.816 | 90.031 |
| 1000 | 31.218 | 98.706 | 98.023 |

|  |  |  |  |
| --- | --- | --- | --- |
| N | Bubble sort таблицы память, байт | Bubble sort ключей, память | Quick sort ключей, память |
| 50 | 6404 | 1204 | 1204 |

Размер одного элемента массива структуры: 112 байт, массива из 50 элементов: 5 600 байт, из 100: 11200 байт, из 1000: 112000 байт

Размер массива ключей (из 50 элементов): 1200 байт.

Сортировка таблицы ключей ~30% быстрее сортировки всей таблицы.

Сортировка таблицы ключей на ~81% меньше по памяти сортировки всей таблицы.

**Выводы по проделанной работе**

В ходе проделанной работы была создана программа для работы с таблицей.

Таблицы ключей дают выигрыш в скорости сортировки в 30%. Сортировка таблицы ключей на 1200 байт больше по памяти сортировки всей таблицы. Это существенно уменьшает время работы программы.

Способ сортировки влияет на время сортировки. Быстрая сортировка на 90% быстрее сортировки пузырьком.

Тип данных, по которым сортируется таблица так же имеет значение. Так целочисленные поля сортируются в разы быстрее, чем строчные. При этом выигрыш по способу сортировки уменьшается.

**Ответы на вопросы**

1.Как выделяется память под вариантную часть записи?

Компилятор ищет вариант, который занимает больше всего памяти и выделяет память только под него. Таким образом, памяти хватит для любого варианта.

2.Что будет, если в вариантную часть ввести данные, не соответствующие описанным?

Компилятор не проверяет правильность внесения данных в область памяти, поэтому за этим должен следить программист.

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

Разработчик должен следить за правильностью выполнения операций с вариантной частью.

4.Что представляет собой таблица ключей, зачем она нужна?

Таблица ключей представляет собой таблицу из двух полей: индекс в исходной таблице, поле ключа.

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

Когда таблица состоит из малого количества столбцов, тогда сортировать таблицу будет эффективнее по памяти

6.Какие способы сортировки предпочтительнее для обработки таблиц и почему?

Предпочтительнее стабильные сортировки, так как они эффективнее по времени, так как не происходит лишних обменов строк массива структур.