|  |  |
| --- | --- |
| 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 «Программная инженерия»

**ОТЧЁТ**

**«**Работа №2, Записи с вариантами, обработка таблиц**»**

Выполнил студент: \_\_***Бугаенко Андрей Павлович***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа: \_\_\_\_***ИУ7-35Б***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверила.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*подпись, дата*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2020 г.*

**Цель работы**

Приобрести навыки работы с типом данных «запись» (структура), содержащим вариантную часть (объединение, смесь), и с данными, хранящимися в таблицах, произвести сравнительный анализ реализации алгоритмов сортировки и поиска информации в таблицах, при использовании записей с большим числом полей, и тех же алгоритмов, при использовании таблицы ключей; оценить эффективность программы по времени и по используемому объему памяти при использовании различных структур и эффективность использования различных алгоритмов сортировок.

**Условие задания**

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 разных 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 %).

**Техническое задание**

Ввести список стран, содержащий название страны, количество жителей, столицу, материк, основной вид туризма (экскурсионный - количество объектов, основной вид (природа, история, искусство); пляжный – основной сезон, температура воздуха и воды, время полета до страны; спортивный – вид спорта (горные лыжи, серфинг, восхождения), минимальная стоимость отдыха). Вывести список стран на выбранном материке, где можно заняться указанным видом спорта.

**Исходные данные и результат**:

В качестве изначальных входных данных программа получает текстовый файл, в котором содержится информация о содержимом полей таблицы.  
  
В качестве результата пользователь получает в зависимости от введённой команды либо таблицу с данными, либо таблицу ключей, либо статистические измерения времени и эффективности.

**Описание задачи, реализуемой программой:**

Программа должна на основе файла, название которого было получено от пользователя, создать таблицу с данными и таблицу ключей. Также программа должна предоставить возможность проводить с этими структурами данных такие операции как:

* Поиск информации по вариантному полю
* Упорядочивание таблицы с помощью исходной таблицы
* Упорядочивание таблицы с помощью массива ключей используя два способа сортировки (простой и ускоренный)
* Оценка эффективности этих алгоритмов по времени и памяти при сортировке исходной таблицы и таблицы ключей

**Способ обращения к программе:**

Для работы с программой пользователь должен использовать консольный клавиатурный ввод. Для запуска программы пользователь использует команду ./app.exe. После этого пользователь попадёт в меню, в котором каждое действие определено как цифра, при вводе которой это действие происходит. Также для выполнения некоторых операций программе могут потребоваться дополнительные данные, о необходимости ввода которых она уведомит пользователя. Данные вводятся в соответствии с запросом программы и если данные введены неправильно, то программа выведет сообщение об ошибке.

**Описание возможных аварийных ситуаций и ошибок пользователя:**

В случае ошибки во вводе программа прекратит выполнение команды и выведет описание ошибки, которая произошла.

Типы ошибок ввода:

* неправильно была введена команда - в этом случае программа уведомит пользователя о том, что была совершена ошибка при попытке вызова команды.
* неправильный ввод дополнительных данных - введённые данные не соответствуют формату ввода, либо их использование в программе не представляется возможным в силу их свойств.

**Описание внутренних структур данных:**

В данной программе использовались две структуры данных, отвечающие соответственно за элемент основной таблицы с данными и элемент таблицы ключей.

Структура для поля данных основной таблицы:

typedef struct

{

    char name[COUNTRY\_MAX\_NAME\_LENGTH];

    char capital[CAPITAL\_MAX\_NAME\_LENGTH];

    unsigned int population;

    continent\_t continent;

    tourism\_t tourism\_spec;

    union

    {

        struct

        {

            unsigned int objects\_amount;

            sightseeing\_t type;

        } sightseeing;

        struct

        {

            season\_t season;

            float air\_temperature;

            float water\_temperature;

            float travel\_time;

        } beach;

        struct

        {

            sport\_t type;

            unsigned int min\_cost;

        } sport;

    } tourism;

} country\_t;

В качестве структуры данных для хранения основной таблицы была выбрана структура с вариантным полем, поскольку она позволяет реализовать хранение похожих записей, которые не идентичны по форме. В нашем случае это виды отдыха со своими параметрами. Однако у такой реализации есть свои проблемы, а именно:

* Память такой структуры определяется как сумма памяти фиксированной части и максимальной длины вариантной записи, поэтому если нам требуется памяти меньше, чем эта величина, то мы будем терять память, поскольку она будет выделена, но использоваться не будет.
* Тип данных в вариантной части при компиляции не проверяется, поэтому, контроль за правильностью ее использования приходится реализовывать вручную.

Структура для поля данных таблицы ключей:

В качестве поля для ключей мы используем обычно стандартную структуру си, в которой мы определяем два поля - индекс элемента, и ключ. В нашем случае ключом будет невариантное поле - население. При сортировке таблицы ключей, которой ранее была сопоставлена основная таблица данных, мы экономим время, но не экономим память, так как нам требуется помимо основной таблицы таблица ключей.

typedef struct

{

    unsigned int index;

    unsigned int population;

} tkey\_t;

**Описание алгоритма:**

* В качестве алгоритмов сортировки были выбраны алгоритмы qsort - как быстра сортировка и bubble sort - как медленная. Выбор qsort обусловлен тем, что этот алгоритм сравнительно универсален и является одним из самых быстрых алгоритмов сортировки, имея сложность выполнения O(n log n).   
  Работа алгоритма:  
   Выбрать из массива элемент, называемый опорным. Это может быть любой из элементов массива. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность.   
   Сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующих друг за другом: «элементы меньшие опорного», «равные» и «большие».   
   Для отрезков «меньших» и «больших» значений выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы.
* В качестве медленной сортировки была выбрана сортировка пузырьком, поскольку он достаточно прост в реализации и имеет сложность O(n2)  
  Работа алгоритма:  
   Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются {\displaystyle N-1}N - 1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива.

**Набор тестов:**

|  |  |  |
| --- | --- | --- |
| Наименование теста | Входные данные | Выходные данные |
| Неправильная команда | 20 | Неверная опция |
| Вывод таблицы | 1 | Таблица |
| Вывод таблицы по ключам | 2 | Таблица по ключам |
| Добавление записи вручную с правильными параметрами | 3, Russia ... правильные параметры | Сообщение о корректном добавлении записи |
| Добавление записи вручную с неправильными параметрами | 3, Russia, -1 ... неверные параметры | Сообщение об ошибке при вводе данных |
| Перемешивание таблицы стран | 4 | Перемешанная таблица стран |
| Перемешивание таблицы ключей | 5 | Перемешанная таблица ключей |
| Быстрая сортировка основной таблицы | 6 | Сообщение о сортировке и время сортировки |
| Медленная сортировка основной таблицы | 7 | Сообщение о сортировке и время сортировки |
| Быстрая сортировка таблицы ключей | 8 | Сообщение о сортировке и время сортировки |
| Медленная сортировка таблицы ключей | 9 | Сообщение о сортировке и время сортировки |
| Поиск по вариантному полю с правильными параметрами | 11, континент, вид спорта - корректные данные | Список элементов, удовлетворяющих условиям поиска |
| Поиск по вариантному полю с неправильными параметрами | 11, некорректные данные | Сообщение об ошибке ввода |
| Удаление элементов с корректным индексом | 12, индекс - корректный | Сообщение об успешно удалённом элементе |
| Удаление элементов с некорректным индексом | 12, -1 | Сообщение об ошибке ввода |
| Эффективность алгоритмов | 13 | Сообщение с параметрами эффективности алгоритмов |

**Анализ скорости обработки:**

При 100 элементах:

Эффективность быстрой сортировки по времени (ключи относительно таблицы): 15.27%

Эффективность медленной сортировки по времени (ключи относительно таблицы): 6.02%  
Эффективность сортировки ключей по времени (быстрая относительно медленной): 102.63%

Эффективность сортировки таблицы по времени (быстрая относительно медленной): 51.42%

Эффективность сортировки по памяти (ключи относительно таблицы): 92%

При 80 элементах:  
Эффективность быстрой сортировки по времени (ключи относительно таблицы): 16.23%  
Эффективность медленной сортировки по времени (ключи относительно таблицы): 6.68%  
Эффективность сортировки ключей по времени (быстрая относительно медленной): 104.63%

Эффективность сортировки таблицы по времени (быстрая относительно медленной): 49.58%  
Эффективность сортировки по памяти (ключи относительно таблицы): 92%

При 60 элементах:  
Эффективность быстрой сортировки по времени (ключи относительно таблицы): 17.50%  
Эффективность медленной сортировки по времени (ключи относительно таблицы): 8.61%

Эффективность сортировки ключей по времени (быстрая относительно медленной): 101.68%

Эффективность сортировки таблицы по времени (быстрая относительно медленной): 52.32%  
Эффективность сортировки по памяти (ключи относительно таблицы): 92%

При 40 элементах:  
Эффективность быстрой сортировки по времени (ключи относительно таблицы): 18.95%  
Эффективность медленной сортировки по времени (ключи относительно таблицы): 12.68%  
Эффективность сортировки ключей по времени (быстрая относительно медленной): 104.81%  
Эффективность сортировки таблицы по времени (быстрая относительно медленной): 50.48%  
Эффективность сортировки по памяти (ключи относительно таблицы): 92%

**Вывод:**Использование таблиц ключей в таблицах данных позволяет значительно сэкономить время сортировки элементов за счёт сокращения времени доступа к данным и времени на перестановку элементов. Однако при этом использование таблицы данных более эффективно по памяти, чем использование таблицы ключей, так как не нужно выделять дополнительную память под таблицу ключей.

**Контрольные вопросы:**

1.Как выделяется память под вариантную часть записи?

Память под вариантную часть записи равна памяти под самый большой элемент вариантной части.

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

При вводе в вариантную часть данных несоответствующих описанных в программе возникнет сбой при попытке их использовать.

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

Компилятор не отслеживает типы данных вариантной части, поэтому за правильностью её использования следит программист.

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

Таблица ключей - вспомогательная таблица данных, которая нужна для того, чтобы облегчить работу с данными при больших размерах таблицы. Таблица ключей состоит из значения индекса элемента и значения ключа - поля данных, которое используется для работы с таблицей данных.

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

Обрабатывать данные в самой таблице имеет смысл, когда она имеет относительно небольшой размер. В этом случае не требуется большого количества времени на доступ и обработку ключевых полей. Однако когда размеры таблицы достигают существенных размеров, то будет целесообразным создать таблицу ключей для экономии времени поиска.

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

Для обработки таблиц будет более эффективным использовать алгоритмы, требующие минимального количества перестановок элементов таблицы, т.е. алгоритмы с низкой сложностью (О(n log n) и менее).