Отчет по лабораторной работе № 2 по дисциплине

“Типы и структуры данных”

Сушина А.Д. ИУ7-31б

Работа № 2

Записи с вариантами. Обработка таблиц

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.

# Описание условия задачи

Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами). Упорядочить данные в ней по возрастанию ключей, 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 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 отдыха.

# Описание ТЗ

### Общая концепция системы

Программа предназначена для работы с таблицей стран. Таблицу можно загрузить как из файла, так и создать свою собственную. Из таблицы может быть удалена запись, или добавлена новая. Можно произвести поиск по таблице, в данной реализации можно вывести все страны с определенного материка, в которых наиболее распространен определенный вид туризма.

### Требования к функциональным характеристикам

Программа должна выполнять следующие функции:

* Производить поиск по таблице
* Сортировать таблицу
* Добавлять новую строку в таблицу
* Удалять строку из таблицы
* Выводить информацию о затраченном времени
* Сообщить об ошибке ввода
* Сообщать о отсутствии элементов, удовлетворяющих поиску
* Сообщать о проблемах с открытием файлов

На вход программа получает действие и имя файла. При необходимости на вход так же поступают параметры поиска или новые данные, которые необходимо добавить в таблицу.

Выход должен быть представлен в том виде, которое требует действие. Это может быть напечатанная таблица, данные о результатах сортировки, отсортированная таблица, неотсоритрованная 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 структур данных

Country - структура, состоящая из:

* name - строка из 31 элемента, название страны
* people - элемент int, количество проживающих людей
* capital - строка из 31 элемента, название столицы
* mainland - строка из 31 элемента, название материка
* turism - строка из 31 элемента, популярный вид спорта
* tur - объединение, состоящее из:

1. ex - структура, состоящая из:

* n - элемент int, количество экскурсий
* type - строка из 31 элемента, тип экскурсий

1. beach - структура, состоящая из:

* season - строка из 31 элемента, время года;
* tw - элемент int, температура воды
* ta - элемент int, температура воздуха
* time - элемент int, время перелета

1. sport - структура, состоящая из:

* type - строка из 31 элемента, вид спорта
* cost - элемент типа int, минимальная цена

struct country

{

char name[N+1];

int people;

char capital[N+1];

char mainland[N+1];

char turism[N+1];

union

{

struct

{

int n;

char type[N+1];

} ex;

struct

{

char season[N+1];

int tw;

int ta;

int time;

} beach;

struct

{

char type[N+1];

int cost;

} sport;

} tur;

};

keys - структура ключей, состоящая из:

* n - элемент int, нумерация строк
* people - элемент int, количество жителей

struct keys

{

int n;

int people;

};

all - массив из 50 элементов типа структура Country, таблица

k - массив из 50 элементов типа структура keys, таблица ключей

action - строка из 31 элемента, действие

filename - строка из 31 элемента, имя файла

int action;

char filename[N+1] = "1.txt";

struct country all[M];

struct keys key[M];

# Описание алгоритма

Пока не выбрана команда выход

* 1 - добавить строку  
  ввести страну

ввести количество жителей  
ввести столицу  
ввести материк  
ввести туризм :

1. спорт  
   ввести тип спорта  
   ввести цену на спорт
2. экскурсии  
   ввести количество экскурсий  
   ввести тип экскурсий
3. пляжи  
   ввести сезон  
   ввести температуру воды  
   ввести температуру воздуха  
   ввести время перелета до страны

открыть файл

записать в файл  
закрыть файл

* 2 - удалить строку  
  считать таблицу из файла  
  вывести таблицу на экран  
  ввести индекс

записать в новую таблицу все строки без строки с введенным индексом

открыть файл

записать в файл  
закрыть файл

* 3 - печать

считать таблицу из файла   
напечатать таблицу

* 4 - поиск

ввести материк  
ввести вид отдыха  
считать таблицу из файла   
Пока не конец таблицы

если материк совпадает   
 если вид отдыха совпадает  
 напечатать строку

если строк нет  
 вывести сообщение об отсутствии строк

* 5 - сортировка  
  выполнить сортировку таблицы пузырьком  
  напечатать таблицу   
  выполнить сортировку таблицы по ключам  
  напечатать таблицу ключей  
  напечатать отсортированную по ключам таблицу   
  выполнить сортировку по ключам вставками  
  вывести время затраченное на сортировку   
  вывести эффективность сортировки
* 6 - выход  
  выход из цикла

# Тесты

Обработка действий:

Неверная цифра

Ввод: 7

Вывод: inputt error

Ввод буквы

Ввод: u

Вывод: input error

Верное значение

Вывод: <действие по значению>

Добавление нового элемента

Ввод больше длины строки

Ввод:aaaaaaaaaaaaaaaaaa

Вывод:< считывание максимальной длины строки>

Ввод неверной цифры или буквы вместо цифры

Ввод: a

Вывод: input error <повторный ввод>

Ввод подходящего значения

Ввод : moscow

Вывод :<переход к следующему вводу или запись>

Удаление строки по индексу

Несуществующий индекс

Ввод : 50

Вывод: input error <возвращение к меню>

Существующий индекс

Ввод : 32

Вывод : done <переход к меню>

Поиск:

Ввод несуществующего значения

Ввод: 9

Вывод : input error

Ввод верного значения

Ввод : 1

Вывод: <вывод подходящего значения>

Сортировка:

Вывод: отсортированная таблица, таблица ключей

## Оценка эффективности:

## 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Table bubble sort time,такты | Table Insert sort time,такты | Key bubble sort  time,такты | Key Insert sort  time,такты |
| 50 | 125067 | 53196 | 19698 | 19074 |
| 100 | 443949 | 173196 | 63699 | 59250 |
| 150 | 931548 | 349434 | 129597 | 87786 |

## 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Table bubble vs Table insert | Key bubble vs key insert | table bubble vs key bubble | table insert vs key insert |
| 50 | 57% | 3% | 84% | 64% |
| 100 | 60% | 6% | 85% | 65% |
| 150 | 62% | 32% | 86% | 74% |

## 

|  |  |  |  |
| --- | --- | --- | --- |
| Table bubble sort memory,байт | Table Insert sort memory,байт | Key Bubble sort  memory,байт | Key Insert sort  memory,байт |
| 6528 | 6540 | 6808 | 6820 |

Размер одного элемента структуры 128 байт: соответственно для 50 элементов - 6400 байт, 100 -12800, 150 - 19200.

Сортировка таблицы ключей ~70-80% быстрее сортировки всей таблицы.

А время сортировки таблицу ключей зависит от способа реализации сортировки.

При большом количестве записей(от 100) сортировка вставками быстрее на 10-20%.

На сортировку таблицы ключей нужно на 4,1% больше памяти..

# Выводы по проделанной работе

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

Таблицы ключей дают выигрыш в скорости сортировки в 70-80% при этом требуя увеличение памяти до 10%. Это существенно уменьшает время работы программы.

Способ сортировки влияет на время сортировки. Сортировка вставками дает выигрыш на 50% при сортировке таблиц целиком. и 20% при сортировке таблицы ключей.

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 трех полей: номер строки, индекс в исходной таблице, поле сортировки.

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

Когда таблица небольшая, с маленьким количеством полей, эффективнее сортировать саму таблицу, а не таблицу ключей. В том случае, если таблица содержит множество полей, лучше использовать сортировку по ключам

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

Для таблиц предпочтительнее сортировка по ключам, так как она быстрее и требует меньше памяти.