МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №12

**«Индексные файлы»**

Выполнил:

студент 1 курса

группы ПО-9

Мисиюк Алексей Сергеевич

Проверила:

Хацкевич М. В.

Брест 2022

**Цель работы:** Изучить принципы программирования с использованием индексных файлов.

**Порядок выполнения работы**

*Текст задания*

В программу разработанную в лабораторной работе 12 внести следующие изменения и дополнения:

1. При запуске программы данные читаются в массив структур из файла, при добавлении новой записи в массив структур в файл должна дописываться новая запись, без изменения остальных записей.
2. Все изменения (изменения полей записи, удаление записи) – сохраняются в файле при помощи перезаписи содержимого всего файла.
3. Сортировка должна выполняться по двум полям на выбор при помощи создания индексных файлов. Содержимое индексного файла переписывается в случае изменения значения ключевого поля (поля, по которому выполняется сортировка) или в случае удаления, добавления записей.

**Блок-схема**



#include <iostream>

#include <map>

#include <math.h>

#include <fstream>

#include <string>

using namespace std;

struct city {

string country;

string title;

int people;

double teens\_part; //храним числовую часть, выводя число добавляем %

enum additionalEnum { First, Second };

union additionalUnion {

int unsigned natural;

int intger;

double rational;

};

int additionalBitField : 2;

};

string getCountryWithMaxTeens(struct city\* cities, int arr\_length);

void menu(struct city\* cities, int arr\_length);

void displayCity(struct city& city);

void addCity(string& country, string& title, int people, double teens\_part, struct city\* cities, int& arr\_length, string fileName);

void sortByField(string field, struct city\* cities, int arr\_length, string fileName);

void quickSortByField(string field, struct city\* cities, int arr\_length, string fileName);

int findIdByField(string field, string value, struct city\* cities, int arr\_length, int pos = 0);

int findIdByField(string field, int value, struct city\* cities, int arr\_length, int pos = 0);

int findIdByField(string field, double value, struct city\* cities, int arr\_length, int pos = 0);

void deleteById(int key, struct city\* cities, int& arr\_length, string fileName);

void loadFromFile(string fileName, struct city\* cities, int& arr\_length);

void saveToFile(string fileName, struct city\* cities, int& arr\_length);

void saveToFileOne(fstream& file, struct city city);

void getArrayOfIdxByField(string field, int\* ArrayOfIdx, int arr\_length);

void updateIdx(struct city\* cities, int arr\_length);

int main()

{

struct city cities[1000];

int arr\_length = 0;

menu(cities, arr\_length);

return 0;

}

void addCity(string& country, string& title, int people, double teens\_part, struct city\* cities, int& arr\_length, string fileName = "") {

cities[arr\_length].country = country;

cities[arr\_length].title = title;

cities[arr\_length].people = people;

cities[arr\_length].teens\_part = teens\_part;

if (fileName != "") {

fstream file;

file.open(fileName, ios::app);

saveToFileOne(file, cities[arr\_length]);

file.close();

updateIdx(cities, arr\_length+1);

}

arr\_length++;

}

void quickSortByField(string field, struct city\* cities, int arr\_length, string fileName) {

struct city tmp[1000];

for (int i = 0; i < arr\_length; i++) {

tmp[i] = cities[i];

}

int ArrayOfIdx[1000];

getArrayOfIdxByField(field, ArrayOfIdx, arr\_length);

for (int i = 0; i < arr\_length; i++) {

cities[ArrayOfIdx[i]] = tmp[i];

}

saveToFile(fileName, cities, arr\_length);

}

void deleteById(int key, struct city\* cities, int& arr\_length, string fileName) {

for (int i = key; i < arr\_length; i++) {

cities[i] = cities[i + 1];

}

arr\_length--;

saveToFile(fileName, cities, arr\_length);

updateIdx(cities, arr\_length);

}

void loadFromFile(string fileName, struct city\* cities, int& arr\_length) {

fstream file;

file.open(fileName, ios::in);

if (!file.is\_open()) {

saveToFile(fileName, cities, arr\_length);

}

else {

arr\_length = 0;

string country;

string title;

int people;

double teens\_part;

getline(file, country);

while (country != "") {

getline(file, title);

file >> people;

file >> teens\_part;

string tmp;

getline(file, tmp);

addCity(country, title, people, teens\_part, cities, arr\_length);

getline(file, country);

}

updateIdx(cities, arr\_length);

file.close();

}

}

void updateIdx(struct city\* cities, int arr\_length) {

fstream country, title;

country.open("idx\_country.txt", ios::out);

title.open("idx\_title.txt", ios::out);

if (!title.is\_open() && !country.is\_open()) {

cerr << "Error while creating file!\n";

throw exception();

}

int idx[1000];

for (int i = 0; i < arr\_length; i++) {

idx[i] = i;

}

for (int i = 0; i < arr\_length; i++) {

for (int j = 0; j < arr\_length - 1; j++) {

if (cities[j].country > cities[j + 1].country) {

swap(cities[j], cities[j + 1]);

swap(idx[j], idx[j + 1]);

}

}

}

for (int i = 0; i < arr\_length; i++) {

country << idx[i] << endl;

}

for (int i = 0; i < arr\_length; i++) {

idx[i] = i;

}

for (int i = 0; i < arr\_length; i++) {

for (int j = 0; j < arr\_length - 1; j++) {

if (cities[j].title > cities[j + 1].title) {

swap(cities[j], cities[j + 1]);

swap(idx[j], idx[j + 1]);

}

}

}

for (int i = 0; i < arr\_length; i++) {

title << idx[i] << endl;

}

country.close();

title.close();

}

void getArrayOfIdxByField(string field, int\* ArrayOfIdx, int arr\_length) {

fstream file;

if (field == "country") {

file.open("idx\_country.txt", ios::in);

}

if (field == "title") {

file.open("idx\_title.txt", ios::in);

}

if (field != "country" && field != "title") {

cerr << "Field " << field << " doesnt have indexes!\n";

throw exception();

}

if (!file.is\_open()) {

cerr << "Error while creating file!\n";

throw exception();

}

for (int i = 0; i < arr\_length; i++) {

file >> ArrayOfIdx[i];

}

file.close();

}

**Вывод:** были изучены принципы программирования и хранения данных с использованием индексных файлов.

.