МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №11

**«Бинарные и текстовые файлы»**

Выполнил:

студент 1 курса

группы ПО-9

Мисиюк Алексей Сергеевич

Проверила:

Хацкевич М. В.

Брест 2022

**Цель работы:** Изучить принципы программирования с использованием бинарных файлов. Ознакомиться с основными функциями в Си для работы с бинарными файлами.

**Порядок выполнения работы**

*Текст задания*

В программу разработанную в лабораторной работе 10 добавить чтение и сохранение данных массива структур при помощи бинарных файлов следующим образом:

1. При первом запуске программы должен создаваться бинарный или текстовый файл на выбор пользователя для хранения данных из массива структур.

2. При добавлении новой записи в массив структур в файл должна дописываться новая запись, без изменения остальных записей.

3. При повторном запуске программы, если файл уже существует, то информация в массив структур должна читаться из этого файла. Если файл отсутствует, то он должен создаваться (см. Пункт 1).

4. Все изменения (сортировка, изменения полей записи, удаление записи) – сохраняются в файле при помощи полной перезаписи содержимого.

5. Сделать вывод о том, какие преимущества использования конкретного типа файлов (бинарные или текстовые) в решаемой вами задаче.

**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);

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);

int main()

{

struct city cities[1000];

int arr\_length = 0;

menu(cities, arr\_length);

return 0;

}

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

cout << "Type file name:" << endl;

string fileName;

getline(cin, fileName);

loadFromFile(fileName, cities, arr\_length);

string input;

cout << "Working with menu:\n0 - exit\t1 - get country with max teens\n2 - input new city\t3 - display cities\n4 - sorting\t5 - delete city\n6 - load from file\n";

while (input != "0") {

getline(cin, input);

if (input == "1") {

if (arr\_length > 0)

cout << "Country with max teens: " << getCountryWithMaxTeens(cities, arr\_length) << endl;

else

cout << "There's no city yet.\n";

}

if (input == "2") {

cout << "Country:\n";

string country;

getline(cin, country);

printf("City:\n");

string title;

getline(cin, title);

cout << "Amount of people:\n";

int people;

cin >> people;

cout << "Teens in %:\n";

double teens\_part;

cin >> teens\_part;

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

}

if (input == "3") {

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

displayCity(cities[i]);

}

if (arr\_length <= 0) {

cout << "There's no city yet.\n";

}

}

if (input == "4") {

cout << "Type field for sorting (country, title, people, teens\_part):\n";

string field;

getline(cin, field);

while (!(field == "country" || field == "title" || field == "people" || field == "teens\_part")) {

cout << "Theres no such field!\n";

getline(cin, field);

}

sortByField(field, cities, arr\_length);

saveToFile(fileName, cities, arr\_length);

cout << "Done!\n";

}

if (input == "5") {

if (arr\_length <= 0) {

cout << "There's no city yet.\n";

}

else {

cout << "Type field for deleting (country, title, people, teens\_part):\n";

string field;

getline(cin, field);

while (!(field == "country" || field == "title" || field == "people" || field == "teens\_part")) {

cout << "Theres no such field!\n";

getline(cin, field);

}

cout << "Type value:\n";

int key;

if (field == "country" || field == "title") {

string value;

getline(cin, value);

key = findIdByField(field, value, cities, arr\_length, 0);

}

if (field == "people") {

int value;

cin >> value;

key = findIdByField(field, value, cities, arr\_length, 0);

}

if (field == "teens\_part") {

double value;

cin >> value;

key = findIdByField(field, value, cities, arr\_length, 0);

}

if (key >= 0) {

displayCity(cities[key]);

cout << "Got deleted!\n";

deleteById(key, cities, arr\_length, fileName);

saveToFile(fileName, cities, arr\_length);

}

else cout << "Value not found!\n";

}

}

if (input == "6") {

loadFromFile(fileName, cities, arr\_length);

}

if (input == "7") {

}

}

}

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();

}

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);

}

file.close();

}

}

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

fstream file;

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

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

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

throw exception();

}

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

saveToFileOne(file, cities[i]);

}

}

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

file << city.country << endl;

file << city.title << endl;

file << city.people << endl;

file << city.teens\_part << endl;

}

**Вывод:** был изучен синтаксис и принципы работы с текстовыми и бинарными файлами. Реализована программа с использованием текстовых файлов, что дает возможность читать структуры глазами, так же переписывать от руки напрямую.