|  |  |
| --- | --- |
| C:\Documents and Settings\admin\Рабочий стол\МИРЭА_ЭМБЛЕМА_приказ.JPG | |
| МИНОБРАНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  «**МОСКОВСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»**  **МИРЭА** | |
| Институт информационных технологий  Кафедра математического обеспечения и стандартизации информационных технологий | |
| **Отчет по лабораторной работе №8**  **Тема работы «Файлы»** | |
| Выполнил: |  |
| Студент группы ИКБО-07-17  Отчет принял:  Преподаватель кафедры МОСИТ | Акжигитов Р. Р.  Миронов Антон Николаевич |
| Москва  2018 | |

Оглавление

[Задание 1 3](#_Toc517780758)

[Условия задания 3](#_Toc517780759)

[Декомпозиция 3](#_Toc517780760)

[Определения классов/функций 3](#_Toc517780761)

[Тестирование 4](#_Toc517780762)

[Задание 2 4](#_Toc517780763)

[Условия задания 4](#_Toc517780764)

[Декомпозиция 4](#_Toc517780765)

[Определения классов/функций 5](#_Toc517780766)

[Тестирование 5](#_Toc517780767)

[Кодирование алгоритма программы 7](#_Toc517780768)

[Вывод 10](#_Toc517780769)

# Задание 1

# Условия задания

Удаление значения строки с заданным номером, путем создания нового файла, удаление старого и переименование нового, указав ему имя удаленного файла.

# Декомпозиция

1. Открыть на чтение и запить два файла.
2. Последовательно считывать из одного файла, и если номер строки не равен номеру удаляемой строки, то записать эту строку в другой файл.

# Определения классов/функций

vector<string> getLines(string filename, string delim = "\n") {

ifstream file(filename);

vector<string> total;;

string temp;

if (file.is\_open())

while (!file.eof()) {

getline(file, temp, '\n');

total.push\_back(temp);

}

file.close();

return total;

}

void deletedStringCopyFile(string name, string tempName, int lineNumber) {

ifstream fin(name);

ofstream fout(tempName);

string temp;

int i = 0;

if (fin.is\_open() && fout.is\_open()) {

while (!fin.eof()) {

getline(fin, temp, '\n');

if (lineNumber != i++)

fout << temp << '\n';

}

}

fin.close();

fout.close();

remove(name.c\_str());

rename(tempName.c\_str(), name.c\_str());

}

template<class T>

void createFile(string filename, vector<T>data) {

ofstream file(filename, ios::trunc);

if (file.is\_open()) {

for (int i = 0; i < data.size() - 1; i++)

file << data[i] << "\n";

file << data.back();

}

file.close();

}

template<class T>

void appendToFile(string filename, vector<T>data) {

ofstream file(filename, ios::app);

if (file.is\_open()) {

file << "\n";

for (int i = 0; i < data.size() - 1; i++)

file << data[i] << "\n";

file << data.back();

}

file.close();

}

void printFile(string filename, string delim = "\n") {

ifstream file(filename);

string temp;

if (file.is\_open())

while (!file.eof()) {

getline(file, temp, '\n');

cout << temp << delim;

}

file.close();

}

# Тестирование

createFile("Hello.txt", vector<int>{1, 2, 5, 10});

//printFile("Hello.txt");

appendToFile("Hello.txt", vector<int>{3, 14, 15, 92});

printFile("Hello.txt", " ");

std::cout << std::endl;

deletedStringCopyFile("Hello.txt", "temp.txt", 1);

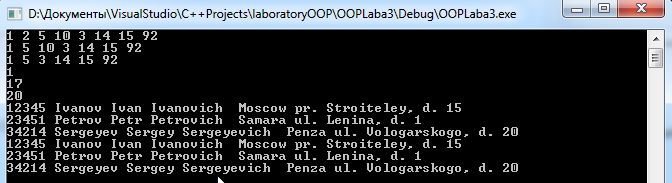
printFile("Hello.txt", " ");

std::cout << std::endl;

deletedStringCopyFile("Hello.txt", "temp.txt", 2);

printFile("Hello.txt", " ");

std::cout << std::endl;



Записываем 1, 2, 5, 10 в файл, добавляем к нему в конце 3, 14, 15, 92.

Удаляем 1 и 2 строку (строки начинаются с 0).

# Задание 2

# Условия задания

**Структура записи:**

Читательский абонемент: номер читательского - целое пятизначное число, ФИО, Адрес

**Доп. операция:**

1. Поиск записи с заданным значением ключа.
2. Удаление найденного значения.

# Декомпозиция

1. Считать бинарный файл в вектор по строкам.
2. Искать по структуре (классу) по ключу.
3. Удалить необходимый элемент.
4. Записать в бинарный файл с этим же именем.

# Определения классов/функций

template<class T>

void writeToBinary(string path, vector<T>& data) {

ofstream fout(path, ios::binary | ios::trunc);

for (auto& i : data)

fout.write((char\*)(&i), sizeof(T));

fout.close();

}

template<class T>

vector<T> readFromBinary(string path) {

vector<T> total;

ifstream fin(path, ios::in | ios::binary);

while (!fin.eof()) {

T temp;

fin.read((char\*)(&temp), sizeof(T));

if (fin.eof()) break;

total.push\_back(temp);

}

fin.close();

return total;

}

template<class T>

T searchRecordById(vector<T> arr, int key = 0) {

for (T a : arr)

if (a.id == key)

return a;

throw std::exception("Record doesn't found...");

}

template<class T>

void deleteRecord(vector<T>& arr, T elem) {

auto it = std::find(arr.begin(), arr.end(), elem);

if (it != arr.end())

arr.erase(it);

}

class Abonement {

public:

Abonement() {};

int id = 0;

char f[256] = "", i[256] = "", o[256] = "", address[256] = "";

friend bool operator==(Abonement elem1, Abonement elem2) {

return elem1.id == elem2.id

&& strcmp(elem1.f, elem2.f) == 0

&& strcmp(elem1.i, elem2.i) == 0

&& strcmp(elem1.o, elem2.o) == 0

&& strcmp(elem1.address, elem2.address) == 0;

}

};

# Тестирование

vector<double> numbers = { 2.45, 17e-2 };

writeToBinary<double>("sample1.dat", numbers);

vector<int> ints = { 1, 17, 20 };

writeToBinary<int>("sample2.dat", ints);

for (auto i : readFromBinary<int>("sample2.dat"))

cout << i << endl;

vector<Abonement> ab;

ifstream file("sample.txt");

while (!file.eof()) {

Abonement temp;

file >> temp.id;

file >> temp.f >> temp.i >> temp.o;

string str = "";

getline(file, str, '\n');

strcpy\_s(temp.address, str.c\_str());

ab.push\_back(temp);

}

for (auto i : ab) {

cout << i.id << " " << i.f << " " << i.i << " " << i.o << " " << i.address << endl;

}

writeToBinary("sampleStruct.dat", ab);

auto str = readFromBinary<Abonement>("sampleStruct.dat");

for (auto i : str) {

cout << i.id << " " << i.f << " " << i.i << " " << i.o << " " << i.address << endl;

}

cout << "\nSearching:\n";

try {

Abonement t = searchRecordById<Abonement>(ab, 23451);

cout << t.id << " " << t.f << " " << t.i << " " << t.o << " " << t.address << endl;

deleteRecord(ab, t);

}

catch (std::exception e) {

cout << e.what();

}

try {

Abonement t = searchRecordById<Abonement>(ab, 11111);

cout << t.id << " " << t.f << " " << t.i << " " << t.o << " " << t.address << endl;

}

catch (std::exception e) {

cout << e.what() << " with id: " << 11111 << endl;

}

writeToBinary("sampleStruct.dat", ab);

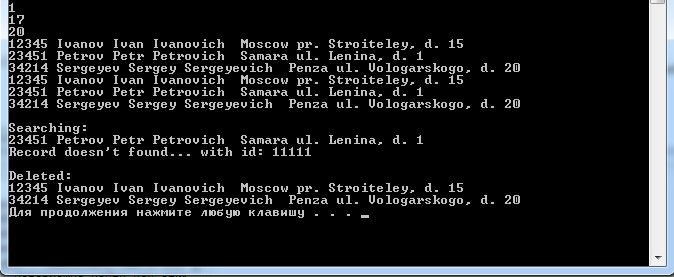
cout << "\nDeleted:\n";

str = readFromBinary<Abonement>("sampleStruct.dat");

for (auto i : str) {

cout << i.id << " " << i.f << " " << i.i << " " << i.o << " " << i.address << endl;

}



Запись в бинарный файл массив типа int, double.

Считывание из текстового файла массив структур.

Запись, считывание из бинарного файла.

Поиск, удаление записи.

# Кодирование алгоритма программы

#include <iostream>

#include <vector>

#include <fstream>

#include <string>

using namespace std;

template<class T>

void createFile(string filename, vector<T>data) {

ofstream file(filename, ios::trunc);

if (file.is\_open()) {

for (int i = 0; i < data.size() - 1; i++)

file << data[i] << "\n";

file << data.back();

}

file.close();

}

template<class T>

void appendToFile(string filename, vector<T>data) {

ofstream file(filename, ios::app);

if (file.is\_open()) {

file << "\n";

for (int i = 0; i < data.size() - 1; i++)

file << data[i] << "\n";

file << data.back();

}

file.close();

}

void printFile(string filename, string delim = "\n") {

ifstream file(filename);

string temp;

if (file.is\_open())

while (!file.eof()) {

getline(file, temp, '\n');

cout << temp << delim;

}

file.close();

}

void deletedStringCopyFile(string name, string tempName, int lineNumber) {

ifstream fin(name);

ofstream fout(tempName);

string temp;

int i = 0;

if (fin.is\_open() && fout.is\_open()) {

while (!fin.eof()) {

getline(fin, temp, '\n');

if (lineNumber != i++)

fout << temp << '\n';

}

}

fin.close();

fout.close();

remove(name.c\_str());

rename(tempName.c\_str(), name.c\_str());

}

template<class T>

void writeToBinary(string path, vector<T>& data) {

ofstream fout(path, ios::binary | ios::trunc);

for (auto& i : data)

fout.write((char\*)(&i), sizeof(T));

fout.close();

}

template<class T>

vector<T> readFromBinary(string path) {

vector<T> total;

ifstream fin(path, ios::in | ios::binary);

while (!fin.eof()) {

T temp;

fin.read((char\*)(&temp), sizeof(T));

if (fin.eof()) break;

total.push\_back(temp);

}

fin.close();

return total;

}

vector<string> getLines(string filename, string delim = "\n") {

ifstream file(filename);

vector<string> total;;

string temp;

if (file.is\_open())

while (!file.eof()) {

getline(file, temp, '\n');

total.push\_back(temp);

}

file.close();

return total;

}

template<class T>

T searchRecordById(vector<T> arr, int key = 0) {

for (T a : arr)

if (a.id == key)

return a;

throw std::exception("Record doesn't found...");

}

template<class T>

void deleteRecord(vector<T>& arr, T elem) {

auto it = std::find(arr.begin(), arr.end(), elem);

if (it != arr.end())

arr.erase(it);

}

class Abonement {

public:

Abonement() {};

int id = 0;

char f[256] = "", i[256] = "", o[256] = "", address[256] = "";

friend bool operator==(Abonement elem1, Abonement elem2) {

return elem1.id == elem2.id

&& strcmp(elem1.f, elem2.f) == 0

&& strcmp(elem1.i, elem2.i) == 0

&& strcmp(elem1.o, elem2.o) == 0

&& strcmp(elem1.address, elem2.address) == 0;

}

};

int main() {

createFile("Hello.txt", vector<int>{1, 2, 5, 10});

//printFile("Hello.txt");

appendToFile("Hello.txt", vector<int>{3, 14, 15, 92});

printFile("Hello.txt", " ");

std::cout << std::endl;

deletedStringCopyFile("Hello.txt", "temp.txt", 1);

printFile("Hello.txt", " ");

std::cout << std::endl;

deletedStringCopyFile("Hello.txt", "temp.txt", 2);

printFile("Hello.txt", " ");

std::cout << std::endl;

vector<double> numbers = { 2.45, 17e-2 };

writeToBinary<double>("sample1.dat", numbers); //getLines("sample.txt")

vector<int> ints = { 1, 17, 20 };

writeToBinary<int>("sample2.dat", ints);

//Abonement ab = {id = 12312, f = ""}

//writeToBinary("sample2.txt", vector<Abonement>{ { 12343, "123", "234", "234", "4534fs" } });

for (auto i : readFromBinary<int>("sample2.dat"))

cout << i << endl;

vector<Abonement> ab;

ifstream file("sample.txt");

while (!file.eof()) {

Abonement temp;

file >> temp.id;

file >> temp.f >> temp.i >> temp.o;

string str = "";

getline(file, str, '\n');

strcpy\_s(temp.address, str.c\_str());

ab.push\_back(temp);

}

for (auto i : ab) {

cout << i.id << " " << i.f << " " << i.i << " " << i.o << " " << i.address << endl;

}

writeToBinary("sampleStruct.dat", ab);

auto str = readFromBinary<Abonement>("sampleStruct.dat");

for (auto i : str) {

cout << i.id << " " << i.f << " " << i.i << " " << i.o << " " << i.address << endl;

}

cout << "\nSearching:\n";

try {

Abonement t = searchRecordById<Abonement>(ab, 23451);

cout << t.id << " " << t.f << " " << t.i << " " << t.o << " " << t.address << endl;

deleteRecord(ab, t);

}

catch (std::exception e) {

cout << e.what();

}

try {

Abonement t = searchRecordById<Abonement>(ab, 11111);

cout << t.id << " " << t.f << " " << t.i << " " << t.o << " " << t.address << endl;

}

catch (std::exception e) {

cout << e.what() << " with id: " << 11111 << endl;

}

writeToBinary("sampleStruct.dat", ab);

cout << "\nDeleted:\n";

str = readFromBinary<Abonement>("sampleStruct.dat");

for (auto i : str) {

cout << i.id << " " << i.f << " " << i.i << " " << i.o << " " << i.address << endl;

}

file.close();

system("pause");

return 0;

}

# Вывод

В данной лабораторной работе я научился обращаться с файлами и файловыми потоками, узнал, как сохранять и создавать файлы, переименовывать их. Сериализовывать структуры в бинарные файлы, а потом снова получить структуры из файлов.