**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и технологий

Образовательная программа «Информационные системы и технологии»

Дисциплина «Теория и технология программирования»

Отчет к лабораторной работе № 8

«Методы хеширования»

Выполнил:

студент группы 3530902/10001

Лобченко А.Р.

Принял:

Доцент к.т.н Хлопин. С. В.

Санкт-Петербург, 2023 год

**Задание лабораторной № 8:**

Составить программу для поиска по хешам данных. Хеширование проводить в соответствии с индивидуальными заданиями. В модуле поиска, предусмотреть реализацию обработки случая, при котором хэш-коды различных данных совпадают.

В некоторых заданиях слово «хеширование» и производные от него употреблены не совсем корректно, поняв разницу между хешированием, шифрованием и кодировкой вы сможете корректно определить, где допущены неточности, это не влияет на условие в целом, но важно для понимания и ответы на вопросы. Например, вариант 6 – предложенный вариант преобразования информации не удовлетворяет определению хеширования.

База данных – это могут просто данные из блокнота, формат .txt, проверку на ввод корректного формата файла по желанию. То же относится и к вариантам, где «дан текст» и синонимичные фразы.

В конце данного документа есть ссылки на материалы, но вы можете пользоваться своими материалами.

Дана таблица текстовой базы данных записями: фамилия; имя; отчество. Произвести хеширование вместе трех полей (в обратном порядке) и поиск по запросу «Фамилия Имя Отчество» и «Фамилия И.О.»

**Текст программы:**

Файл laba\_8.cpp:

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

#include "Hash\_Search.h"

#include "Hash\_Function.h"

#include <Windows.h>

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Hash\_Search search;

search.Start();

}

Файл ConsoleRead.h:

#pragma once

#include <iostream>

#include <string>

#include <vector>

namespace Console {

std::string GetWord();

std::vector<std::string> GetWordReduc();

}

Файл Data.h:

#pragma once

#include <string>

#include <iostream>

struct Data {

std::string name = "Unknown";

std::string surname = "surname";

std::string patronymic = "patronymic";

};

std::ostream& operator<<(std::ostream& out, Data data);

Файл FileClient.h:

#pragma once

#include <string>

#include <locale.h>

#include <fstream>

#include <vector>

#include <deque>

#include <regex>

#include "Data.h"

#include <iostream>

class FileClient

{

public:

FileClient();

std::vector<Data> Read();

private:

std::deque<std::string> buffer;

std::ifstream file;

std::string path\_to\_file\_;

bool end\_flag = false;

std::string ReadWord();

bool ReadFileToBuffer();

void GetFilePath();

bool CheckPath(const std::string& path);

};

Файл Hash\_Function.h:

#pragma once

#include <string>

namespace hash {

int GetHashCode(std::string key);

}

Файл Hash\_Search.h:

#pragma once

#include <string>

#include <vector>

#include "Data.h"

#include "FileClient.h"

#include "ConsoleRead.h"

class Hash\_Search

{

public:

Hash\_Search();

void Start();

void Search(const Data&, const int& );

private:

std::vector<Data> data\_;

std::vector<int> hash\_fulls;

std::vector<int> hash\_initials;

void Hashing();

struct Result {

Data data;

int hash;

bool IsCoincid;

};

void PrintSearchResult (std::vector<Result>);

Data GetRequest();

};

Файл Hash\_Search.cpp:

#include "Hash\_Search.h"

#include "FileClient.h"

#include "Hash\_Function.h"

#include <string>

#include <vector>

#include "Data.h"

Hash\_Search::Hash\_Search() {

FileClient file;

data\_ = file.Read();

Hashing();

for (auto& elem : data\_) {

std::cout << elem << std::endl;

}

}

void Hash\_Search::Hashing() {

for (Data& elem : data\_) {

std::string for\_hash = elem.patronymic + elem.name + elem.surname;

hash\_fulls.push\_back(hash::GetHashCode(for\_hash));

std::string for\_initial\_hash;

for\_initial\_hash = elem.patronymic[0];

for\_initial\_hash += elem.name[0];

for\_initial\_hash += elem.surname;

hash\_initials.push\_back(hash::GetHashCode(for\_initial\_hash));

}

}

void Hash\_Search::Search(const Data& request, const int& hash) {

std::vector<Result> result;

for (size\_t i = 0; i < hash\_fulls.size(); i++) {

if (hash == hash\_fulls[i]) {

if (request.name == data\_[i].name

&& request.patronymic == data\_[i].patronymic

&& request.surname == data\_[i].surname) {

result.push\_back(Result{ data\_[i], hash, false });

}

else {

result.push\_back(Result{ data\_[i], hash, true });

}

}

}

for (size\_t i = 0; i < hash\_initials.size(); i++) {

if (hash == hash\_initials[i]) {

if (request.name[0] == data\_[i].name[0]

&& request.patronymic[0] == data\_[i].patronymic[0]

&& request.surname == data\_[i].surname) {

result.push\_back(Result{data\_[i], hash, false });

}

else {

result.push\_back(Result{ data\_[i], hash, true });

}

}

}

PrintSearchResult(result);

}

void Hash\_Search::PrintSearchResult(std::vector<Result> date) {

using namespace std;

cout << endl;

if (date.size()) {

cout << "Вот что мне удалось найти: "s << endl;

for (const auto res : date) {

if (res.IsCoincid) {

cout << "Произошла коллизия :"s;

}

else {

cout << "искомый элемент: "s;

}

cout << res.data << " "s << res.hash << endl;

}

}

else {

cout << "В базе данных нет данных по этому человеку";

}

cout << endl;

}

void Hash\_Search::Start() {

Data request = GetRequest();

int hash = hash::GetHashCode(request.patronymic + request.name + request.surname);

Search(request, hash);

}

Data Hash\_Search::GetRequest() {

do {

Data request;

std::cout << std::endl << "Введите ФИО полностью, или через инициалы: ";

request.surname = Console::GetWord();

auto word\_red = Console::GetWordReduc();

if (word\_red.size() == 1) {

request.name = word\_red.front();

word\_red = Console::GetWordReduc();

if (word\_red.size() == 1) {

request.patronymic = word\_red.back();

}

else {

std::cout << "Вы ввели данные не в верном формате. Пожалуйста используйте один из слудующих формата ввода: Фамилия Имя Отчество, Фимилия И.О., Фамилия И О" << std::endl;

std::string c;

std::getline(std::cin, c);

continue;

}

}

else if (word\_red.size() == 2) {

request.name = word\_red.front();

request.patronymic = word\_red.back();

}

else {

std::cout << "Вы ввели данные не в верном формате. Пожалуйста используйте один из слудующих формата ввода: Фамилия Имя Отчество, Фимилия И.О., Фамилия И О" << std::endl;

std::string c;

std::getline(std::cin, c);

continue;

}

std::transform(request.surname.begin(), request.surname.end(), request.surname.begin(), tolower);

std::transform(request.name.begin(), request.name.end(), request.name.begin(), tolower);

std::transform(request.patronymic.begin(), request.patronymic.end(), request.patronymic.begin(), tolower);

return request;

} while (true);

}

Файл Hash\_Function.cpp:

#include "Hash\_Function.h"

namespace hash {

int GetHashCode(std::string key)

{

double k = 0;

for (char ch : key) {

k += abs(ch);

}

int N = 13; double A = 0.618033;

int h = N \* fmod(k \* A, 23);

return h;

}

}

Файл FileClient.cpp:

#include <algorithm>

#include <fstream>

#include <codecvt>

#include "ConsoleRead.h"

#include "FileClient.h"

std::deque<std::string> SplitIntoWordsView(std::string str) {

std::string sep = ";:/.,";

std::deque<std::string> result;

std::string word;

for (const auto& ch : str) {

if (sep.find(ch) != sep.npos) {

if (!word.empty()) {

result.push\_back(word);

word.clear();

}

}

else {

word += ch;

}

}

if (!word.empty()) {

result.push\_back(word);

word.clear();

}

return result;

}

FileClient::FileClient() {

GetFilePath();

}

std::vector<Data> FileClient::Read() {

do

{

file.open(path\_to\_file\_);

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

std::cout << "Не удалось открыть файл" << std::endl;

GetFilePath();

}

else {

break;

}

} while (true);

std::vector<Data> result;

while (true) {

Data date;

date.surname = ReadWord();

date.name = ReadWord();

date.patronymic = ReadWord();

if (end\_flag) {

if (!date.name.empty()) {

result.push\_back(date);

}

file.close();

return result;

}

result.push\_back(date);

}

}

std::string FileClient::ReadWord() {

if (buffer.empty()) {

if (!ReadFileToBuffer()) return "";

}

std::string result = buffer.front();

buffer.pop\_front();

return result;

}

bool FileClient::ReadFileToBuffer() {

std::string word;

if (file >> word)

{

std::transform(word.begin(), word.end(), word.begin(), tolower);

buffer = SplitIntoWordsView(word);

}

else {

end\_flag = true;

return false;

}

return true;

}

bool FileClient::CheckPath(const std::string& path)

{

const char\* reservedWords[] = { "CON", "CONSOLE", "PRN", "AUX", "COM1", "COM2", "COM3", "COM4", "COM5", "COM6", "COM7", "COM8", "COM9",

"LPT1", "LPT2", "LPT3", "LPT4", "LPT5", "LPT6", "LPT7", "LPT8", "LPT9" };

char name[\_MAX\_FNAME];

char ext[\_MAX\_EXT];

\_splitpath\_s(path.c\_str(), nullptr, 0, nullptr, 0, name, \_MAX\_FNAME, ext, \_MAX\_EXT);

std::transform(name, name + strlen(name), name, ::toupper);

for (int i = 0; i < sizeof(reservedWords) / sizeof(reservedWords[0]); i++)

{

if (strcmp(name, reservedWords[i]) == 0)

{

return false;

}

}

return true;

}

void FileClient::GetFilePath() {

do

{

std::cout << "Введите имя файла: ";

std::string path = Console::GetWord();

std::string c;

std::getline(std::cin, c);

if (CheckPath(path)) {

path\_to\_file\_ = path;

return;

}

std::cout << "Путь содержит запрещенные символы" << std::endl;

} while (true);

}

Файл Data.cpp:

#include "Data.h"

std::ostream& operator<<(std::ostream& out, Data data) {

out << data.surname << ' ' << data.name << ' ' << data.patronymic;

return out;

}

Файл ConsoleRead.cpp:

#include "ConsoleRead.h"

namespace Console {

std::string GetWord() {

do

{

std::string word;

std::cin >> word;

if (!word.empty()) {

std::string c;

return word;

}

} while (true);

}

std::vector<std::string> SplitIntoDots(std::string str) {

std::string sep = ".";

std::vector<std::string> result;

std::string word;

for (const auto& ch : str) {

if (sep.find(ch) != sep.npos) {

if (!word.empty()) {

result.push\_back(word);

word.clear();

}

}

else {

word += ch;

}

}

if (!word.empty()) {

result.push\_back(word);

word.clear();

}

return result;

}

std::vector<std::string> GetWordReduc() {

std::string word = GetWord();

return SplitIntoDots(word);

}

}

**Пример работы:**

Изображение выглядит как текст

Автоматически созданное описание

**Выводы:** в ходе лабораторной работе были изучены методы хеширования данных. Применив один из таких методов на практике, были выявлены все тонкости работы, например, как программа должна реагировать на коллизии данных. вследствие того, что числа сравниваются быстрее, чем, например строки, осуществляя поиск данных по хешу, мы заметно сокращаем время работы программы.