

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

Дисциплина: «Программирование»

Отчёт по лабораторной работе № 3

**Лабораторная работа №3. Методы хэширования**

**Выполнил студент группы №485:**

**Проверили:**

Корниенко Иван Григорьевич

Федин Алексей Константинович

Санкт-Петербург

2019

# Постановка задачи

Необходимо составить программу для поиска по хэшам данных. В модуле поиска предусмотреть реализацию обработки случая, при котором хэш-коды различных данных совпадают. Дана текст. Произвести хэширование по блокам, содержащих в себе 10 элементов (символов).

# Исходные данные

В качестве исходных данных программа использует вводимый пользователем путь к файлу с текстом и введенный в консоль текст.

# Особые ситуации

* Если пользователь при указании пути к файлу использует запрещённые имена, например: con, aux и т.д., программа попросит ввести путь заново.
* Если пользователь укажет путь к файлу, к которому программа не может получить доступ из-за недостатка прав, она попросит ввести путь заново.
* Если пользователь укажет путь к несуществующему файлу, из которого программа должна получить данные, она попросит ввести путь заново.

# Математические методы и алгоритмы решения задач

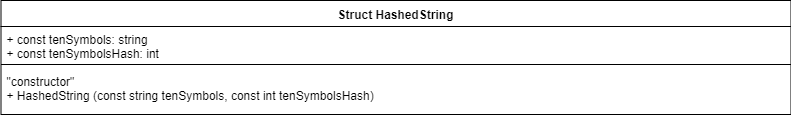


Рисунок 1 – структура HashedString, содержащая в себе блок из 10 символов и его хэш

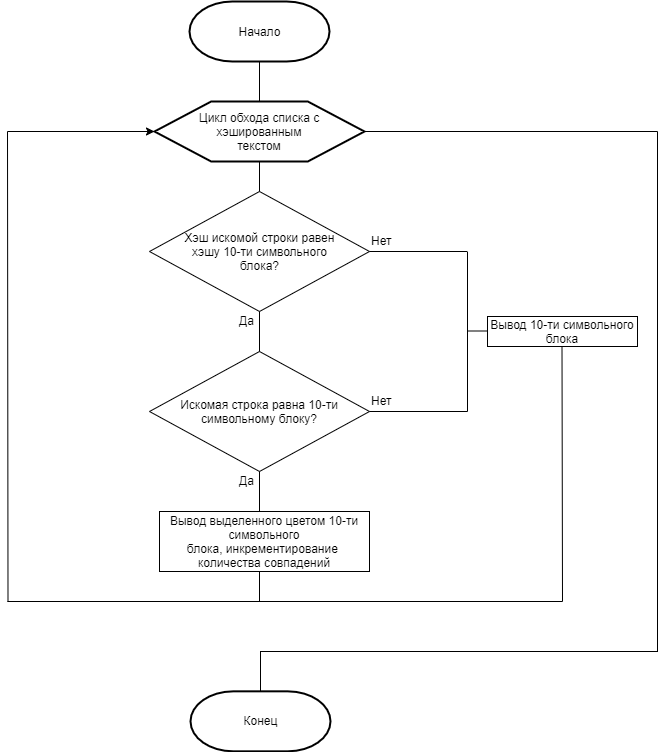


Рисунок 2 – функция поиска строки в тексте

Согласно постановке задачи, для составления программы будут использоваться алгоритмы, блок-схемы которых представлены выше.

Функция хэширования заключается в умножении ASCII кода символа на его позицию в блоке. В связи с простотой алгоритма было решено не строить его блок-схему.

# Форматы представления данных

Программа использует следующие переменные:

Таблица 1 – Переменные, используемы в программе

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| running | bool | Определяет запущена ли программа |
| tenSymbolsHash | int | Значение хэша строки |
| path | string | Путь к файлу с текстом |
| file | ifstream | Открытие файла с текстом |
| hashedText | list<HashedString> | Структура данных для сохранения текста и его хэша |
| tenSymbols | string | Хранение 10-ти символьного блока |
| isPathValid | bool | Проверка на валидность введённого пути |

Для задания размера минимального блока символов, а также обозначения количества символов, стираемых из буфера cin, в случае некорректного ввода пользователя используются следующие константы:

Таблица 2 – Константы, используемы в программе

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Тип** | **Значение** | **Описание** |
| INT\_MAX | const int | 2147483647 | Максимальное целое число |
| minBlockSize | const int | 10 | Минимальный блок символов |

# Структура программы

В силу большого количества функций программа разделена на 4 исполняемых модуля, из которых один является основным и отвечает за запуск программы и содержит меню, двое оставшихся содержат в себе функции, необходимые для работы программы.

Модуль Source:

Таблица 3 – Функции, составляющие модуль Source

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| main | Начало программы |
| Menu | Отображение меню программы, ввод пункта меню |

Модуль FileInputOutput:

Таблица 4 – Функции, составляющие модуль FileInputOutput

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| FilePathCheckReturnForInput | Ввод, проверка и возврат пути к файлу с текстом |
| FilePathCheckReturnForOutput | Проверка и возврат пути файла |
| OutputTextInFile | Вывод введенного вручную текста в файл |
| OutputResultInFile | Вывод результата работы программы в файл |

Модуль VarCheckAndColorSet:

Таблица 5 – Функции, составляющие модуль VarCheckAndColorSet

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| SetColor | Изменение цвета текста, выводимого в консоль |
| GetInt | Проверка на валидность данных, вводимых пользователем, типа int |
| GetBool | Проверка на валидность данных, вводимых пользователем, типа bool |
| CoutWithColor | Вывод сообщения с заданным цветом |
| InputCheck | Проверка на валидность данных любого типа, вводимых пользователем |

Модуль HashAndSearch:

Таблица 6 – Функции, составляющие модуль HashAndSearch

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Hash | Хэширование символа |
| InputAndHash | Ввод текста, его хэширование |
| Search | Ввод строки, её поиск в тексте |

# Описание хода выполнения лабораторной работы

* В ходе лабораторной работы было создано решение (Lab3) в интегрированной среде разработки Microsoft Visual Studio C++ 2019. В нём был создан проект.
* В начале было решено использовать одномерный динамический массив для хранения текста и его хэша, однако, из-за непредсказуемости размера текста нужно было бы постоянно очищать и выделять память, что серьёзно ухудшило бы быстродействие программы. Чтобы этого избежать, было принято решение использовать структуру данных list из библиотеки STL.
* При работе программы с файлами нужно было добавить проверки на валидность имени файла.
* При получении пользовательского ввода необходимо было добавить проверку на его соответствие предполагаемому типу данных и условиям выбора.

# Результат работы программы

В результате работы программа выводит 10-ти символьные блоки и их хэши, а также текст и выделенные совпадения и их количество.

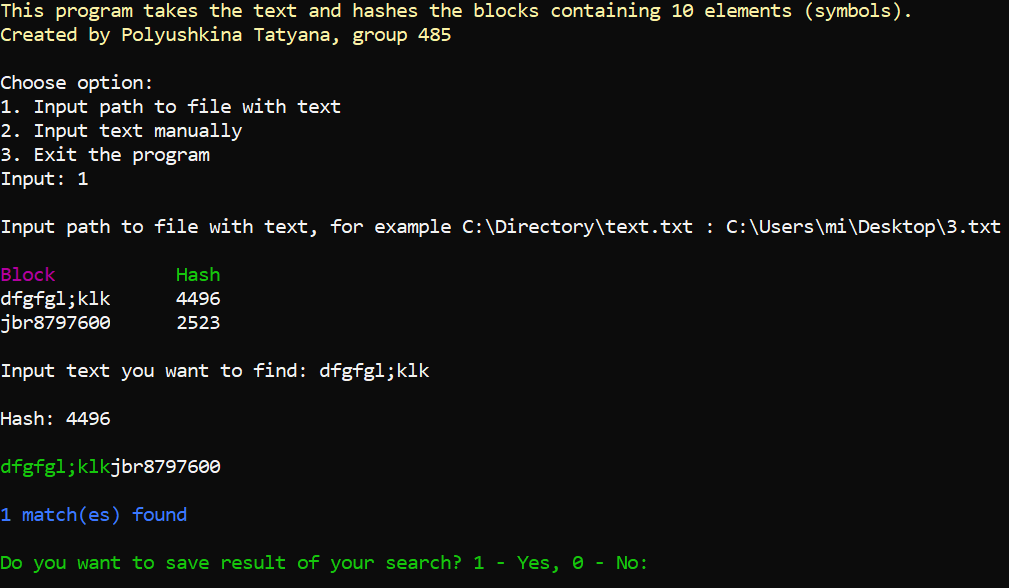


Рисунок 3 – Поиск и вывод совпадений в тексте

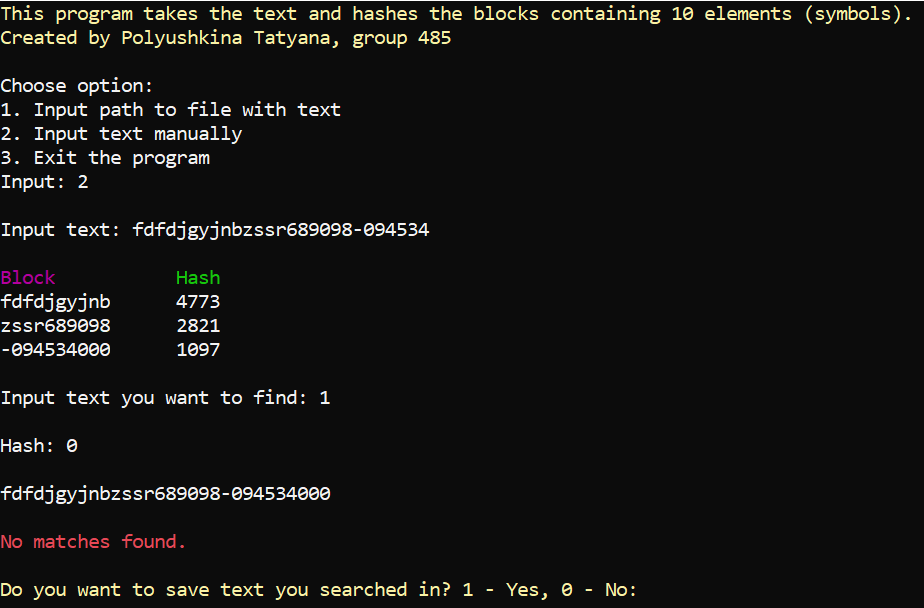


Рисунок 4 – Поиск и вывод сообщения о том, что совпадений не найдено

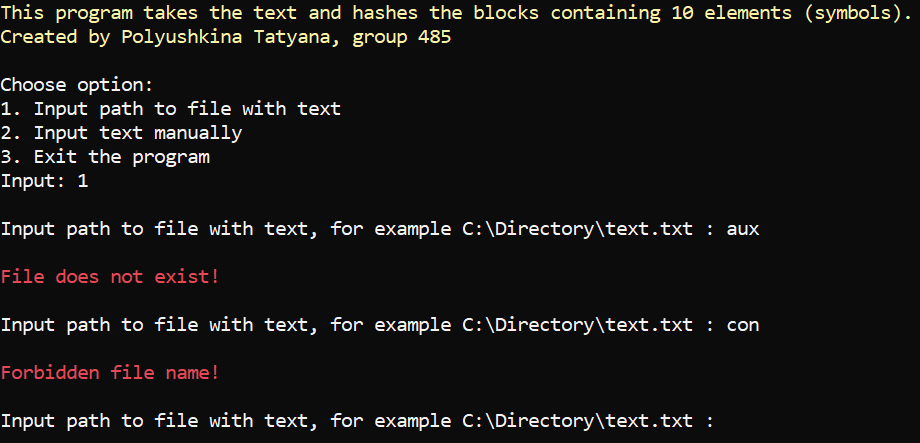


Рисунок 5 – Проверка путей к тексту

# Текст программы

# [--- Начало программы ---]

**// FileInputOutput.h**

**// Лабораторная работа №3.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#pragma once

#include <string>

#include <list>

#include <vector>

#include "HashAndSearch.h"

using namespace std;

string FilePathCheckReturnForInput();

string FilePathCheckReturnForOutput(const string& message);

void OutputTextInFile(const list<HashedString>& text);

void OutputResultInFile(const vector<int>& matchedBlockNums, const string& searchedMessage);

**// HashAndSearch.h**

**// Лабораторная работа №3.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#pragma once

#include <string>

using namespace std;

struct HashedString {

const string tenSymbols;

const int tenSymbolsHash;

HashedString(const string& tenSymbols, const int& tenSymbolsHash);

};

list<HashedString> InputAndHash(const int& inputWay);

void Search(const list<HashedString>& hashedText, const int& inputWay, bool& askedToSaveText);

**// VarCheckAndColorSet.h**

**// Лабораторная работа №3.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#pragma once

#include <string>

using namespace std;

enum Color { blue = 9, green, azure, red, purple, yellow, white };

int GetInt();

bool GetBool();

void CoutWithColor(const int& color, const string& message);

void SetColor(const int& color);

**// VarCheckAndColorSet.cpp**

**// Лабораторная работа №3.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#include <iostream>

#include <Windows.h>

#include <string>

#include "VarCheckAndColorSet.h"

using namespace std;

template <typename T> // создание шаблона T

T InputCheck() { // проверка ввода пользователя

T userInput; // создание переменной заданного типа для записи ввода пользователя

cin >> userInput;

while (cin.fail()) { // пока введенные данные не соответствуют типу

CoutWithColor(red, "\nTry again: ");

cin.clear(); // очищает cin.fail()

cin.ignore(INT\_MAX, '\n'); // очистка буфера

cin >> userInput;

}

cin.ignore(INT\_MAX, '\n');

return userInput;

}

int GetInt() { // ввод пользователем, проверка и возврат значения int

return InputCheck<int>();

}

bool GetBool() { // ввод пользователем, проверка и возврат значения bool

return InputCheck<bool>();

}

void SetColor(const int& color) { // изменение цвета в консоли

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), color);

}

void CoutWithColor(const int& color, const string& message) { // вывод сообщения message с цветом color

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), color); // изменение цвета на color

cout << message; // вывод сообщения

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), white); // изменение цвета на белый

}

**// HashAndSearch.cpp**

**// Лабораторная работа №3.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#include <iostream>

#include <string>

#include <fstream>

#include <list>

#include <vector>

#include "FileInputOutput.h"

#include "HashAndSearch.h"

#include "VarCheckAndColorSet.h"

using namespace std;

enum {fileInput = 1, manualInput};

const int minBlockSize = 10;

HashedString::HashedString(const string& tenSymbols, const int& tenSymbolsHash) : tenSymbols(tenSymbols), tenSymbolsHash(tenSymbolsHash) {}

int Hash(const char& symbol, const int& pos) {

return symbol \* pos;

}

list<HashedString> InputAndHash(const int& inputWay) {

list<HashedString> hashedText;

if (inputWay == fileInput) {

string path = FilePathCheckReturnForInput();

ifstream file(path);

string tenSymbols = "";

int tenSymbolsHash = 0;

char symbol;

int i = 0;

while (true) {

if (file >> symbol) {

tenSymbols += symbol;

tenSymbolsHash += Hash(symbol, i);

++i;

if (i == minBlockSize) {

hashedText.emplace\_back(tenSymbols, tenSymbolsHash);

tenSymbols = "";

tenSymbolsHash = 0;

i = 0;

}

}

else {

if (i) {

for (int j = 0; j < minBlockSize - i; ++j) {

tenSymbols += "0";

}

hashedText.emplace\_back(tenSymbols, tenSymbolsHash);

}

break;

}

}

}

else

if (inputWay == manualInput){

string text;

string tenSymbols = "";

int tenSymbolsHash = 0;

cout << "\nInput text: ";

getline(cin, text);

int j = 0;

for (int i = 0; i < text.size(); ++i) {

tenSymbols += text[i];

tenSymbolsHash += Hash(text[i], j);

++j;

if (j == minBlockSize) {

hashedText.emplace\_back(tenSymbols, tenSymbolsHash);

tenSymbols = "";

tenSymbolsHash = 0;

j = 0;

}

}

if (j) {

for (int i = 0; i < minBlockSize - j; ++i) {

tenSymbols += "0";

}

hashedText.emplace\_back(tenSymbols, tenSymbolsHash);

}

}

CoutWithColor(purple, "\nBlock");

CoutWithColor(green, " \tHash\n");

for (auto textPart = hashedText.begin(); textPart != hashedText.end(); ++textPart)

cout << textPart->tenSymbols << '\t' << textPart->tenSymbolsHash << endl;

cout << endl;

return hashedText;

}

void Search(const list<HashedString>& hashedText, const int& inputWay, bool& askedToSaveText) {

string input;

cout << "Input text you want to find: ";

getline(cin, input);

string copyInputToSaveInFile = input;

int hash = 0;

for (int i = 0; i < input.size(); ++i) {

hash += Hash(input[i], i);

}

if (input.size() < minBlockSize) {

int inputSize = input.size();

for (int i = 0; i < minBlockSize - inputSize; ++i)

input += "0";

}

cout << "\nHash: " << hash << "\n\n";

vector<int> matchedBlockNums;

int blockNum = 1;

for (auto textPart = hashedText.begin(); textPart != hashedText.end(); ++textPart, ++blockNum) {

if (hash == textPart->tenSymbolsHash && input == textPart->tenSymbols) {

matchedBlockNums.push\_back(blockNum);

CoutWithColor(green, textPart->tenSymbols);

}

else

cout << textPart->tenSymbols;

}

if (matchedBlockNums.size()) {

SetColor(blue);

cout << "\n\n" << matchedBlockNums.size() << " match(es) found\n";

SetColor(white);

}

else

CoutWithColor(red, "\n\nNo matches found.\n");

if (inputWay == manualInput && !askedToSaveText) {

SetColor(yellow);

cout << "Do you want to save text you searched in? 1 - Yes, 0 - No: ";

bool saveText = GetBool();

SetColor(white);

askedToSaveText = true;

if (saveText)

OutputTextInFile(hashedText);

}

SetColor(green);

cout << "\nDo you want to save result of your search? 1 - Yes, 0 - No: ";

bool saveResult = GetBool();

SetColor(white);

if (saveResult)

OutputResultInFile(matchedBlockNums, copyInputToSaveInFile);

SetColor(azure);

cout << "\nDo you want to search another text in this text file? 1 - Yes, 0 - No: ";

bool anotherSearch = GetBool();

cout << "\n";

SetColor(white);

if (anotherSearch)

Search(hashedText, inputWay, askedToSaveText);

}

**// Source.cpp**

**// Лабораторная работа №3.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#include <iostream>

#include <list>

#include "VarCheckAndColorSet.h"

#include "HashAndSearch.h"

using namespace std;

enum MenuItems {

minMenuItem = 0, file, manually, close, maxMenuItem

};

bool Menu() { // Вывод меню и проверка введённых данных

cout << "Choose option:\n" << "1. Input path to file with text\n" << "2. Input text manually\n" <<

"3. Exit the program\n" << "Input: ";

int menuItem; // пункт меню для ввода пользователем

bool running = true; // Для возврата в main, проверка запущена ли программа

while (true) { // ввод пользователем элемента меню и его проверка

menuItem = GetInt(); // ввод элемента меню, проверка на int

if (menuItem > minMenuItem && menuItem < maxMenuItem) { // проверка на выход за пределы меню

break;

}

else CoutWithColor(red, "Try again: ");

}

if (menuItem != close) {

bool askedToSaveText = false;

Search(InputAndHash(menuItem), menuItem, askedToSaveText);

}

else

running = false;

return running; // если running = true - программа продолжается, если false, то программа завершается

}

int main() { // работа программы

CoutWithColor(yellow, "This program takes the text and hashes the blocks containing 10 elements (symbols).\nCreated by Polyushkina Tatyana, group 485\n\n");

bool running = true; // Для определения запущенности программы

while (running)

{

running = Menu();

}

return 0;

}

**// FileInputOutput.cpp**

**// Лабораторная работа №3.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#include <string>

#include <iostream>

#include <fstream>

#include <filesystem>

#include <list>

#include <vector>

#include "FileInputOutput.h"

#include "VarCheckAndColorSet.h"

#include "HashAndSearch.h"

using namespace std;

using namespace std::filesystem;

string FilePathCheckReturnForInput() { // проверка пути для чтения из файла

string filePath;

bool isPathValid = false;

while (!isPathValid) { // while isPathValid == false, проверка валидности введенного пути

cout << "\nInput path to file with text, for example C:\\Directory\\text.txt : ";

cin >> filePath;

cin.ignore(INT\_MAX, '\n');

if (!ifstream(filePath)) { // существует ли файл

CoutWithColor(red, "\nFile does not exist!\n");

continue;

}

try {

is\_regular\_file(filePath);

}

catch (...) {

CoutWithColor(red, "\nForbidden file name!\n");

continue;

}

ifstream file(filePath);

if (!file) { //проверка доступа к файлу

CoutWithColor(red, "\nAccess denied!\n");

continue;

}

isPathValid = true;

}

return filePath;

}

string FilePathCheckReturnForOutput(const string& message) { // проверка пути для сохранения в файл

string filePath;

bool isPathValid = false;

while (!isPathValid) { // while isPathValid == false, проверка валидности введенного пути

cout << "\n" << message;

cin >> filePath;

cin.ignore(INT\_MAX, '\n');

if (ifstream(filePath)) { // существует ли файл

try {

if (is\_regular\_file(filePath)) { // проверка на запрещенные имена (aux, con..)

SetColor(yellow);

cout << "\nFile already exists. Do you want to overwrite it? 1 - Yes, 0 - No: ";

bool toOverwrite = GetBool();

SetColor(white);

if (!toOverwrite) {

continue;

}

}

}

catch (...) {

CoutWithColor(red, "\nForbidden file name!\n");

continue;

}

}

ofstream file(filePath); // создание файла по заданному пути

if (!file) { // проверка на доступ к созданию файла

CoutWithColor(red, "\nAccess denied!\n");

continue;

}

isPathValid = true;

}

return filePath; // возврат корректного пути

}

void OutputTextInFile(const list<HashedString>& text) {

string path = FilePathCheckReturnForOutput("Input path to file to save text: ");

ofstream output(path);

for (auto textPart = text.begin(); textPart != text.end(); ++textPart) {

output << textPart->tenSymbols;

}

output.close();

}

void OutputResultInFile(const vector<int>& matchedBlockNums, const string& searchedMessage) {

string path = FilePathCheckReturnForOutput("Input path to file to save result: ");

ofstream output(path);

output << "You searched for: " << searchedMessage;

if (matchedBlockNums.size()) {

output << "\n\n" << matchedBlockNums.size() << " match(es) found.\n\nBlock(s) number(s):";

for (int i = 0; i < matchedBlockNums.size(); ++i) {

output << " " << matchedBlockNums[i];

}

}

else

output << "\n\nNo matches found.";

output.close();

}

**[--- Конец программы ---]**