Беларускі дзяржаўны тэхналагічны ўніверсітэт

Факультэт інфармацыйных тэхналогій

Кафедра праграмнай інжэнерыі

Лабараторная работа №14

Па дысцыпліне «Асновы алгарытмізацыі і праграмавання»

На тэму «Хэш-табліцы»

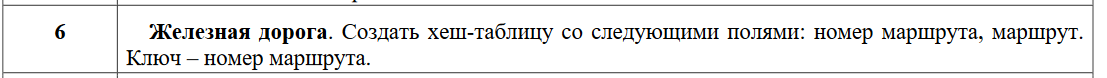
Выканала:

Студэнтка 1 курса 6 группы

Жучкевіч Кацярына Сяргееўна

Выкладчык: асс. Андронава М.В.

Мінск, 2024



#include <iostream>

#include <vector>

#include <Windows.h>

#include <list>

#include <chrono>

using namespace std;

using namespace std::chrono;

const int TABLE\_SIZE = 3;

struct Railway {

int number;

string rute;

};

int universalHash(int key) {//ствараем універсальную хэш табліцу

const int a = 4;

const int b = 9;

const int p = 52;

return ((a \* key + b) % p) % TABLE\_SIZE;

}

void insert(vector<list<Railway>>& table, const Railway& school) { //функцыя для таго, каб уставіць элемент ў табліцу

int ind = universalHash(school.number);

table[ind].push\_back(school);

}

string search(const vector<list<Railway>>& table, int key) { //функцыя для таго, каб знайсці элемент

int ind = universalHash(key);

auto start = high\_resolution\_clock::now();

for (const auto& entry : table[ind]) { //цыкл for each для таго, каб знайсці ключ

if (entry.number == key) { //і калі яго знайшлі...

auto end = high\_resolution\_clock::now();

double time = duration\_cast<microseconds>(end - start).count();

cout << "Час: " << time << endl;

return entry.rute; //вяртаем маршрут

}

}

return "Няма такога маршруту!!";

}

void show(const vector<list<Railway>>& table) { //функцыя для вываду табліцы

for (int i = 0; i < TABLE\_SIZE; ++i) {

if (!table[i].empty()) { //калі спіс не пусты

for (const auto& entry : table[i]) {

cout << entry.number << " - " << entry.rute; //выводзім

}

}

else {

cout << "Пуста";

}

cout << endl;

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

vector<list<Railway>> hashTable(TABLE\_SIZE);

insert(hashTable, { 1, "Ліда-Мінск" });

insert(hashTable, { 34, "Гомель-Маладзечна" });

insert(hashTable, { 52, "Валожын-Гродна" });

cout << "Вось тое, што атрымалася :\n";

show(hashTable);

int searchKey;

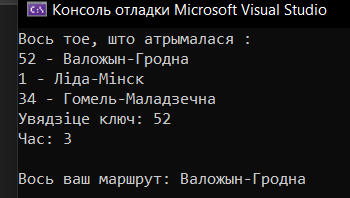
cout << "Увядзіце ключ: ";

cin >> searchKey;

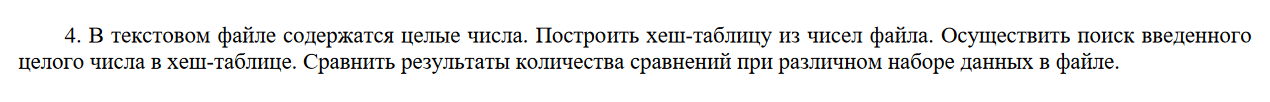
cout << "\nВось ваш маршрут: " << search(hashTable, searchKey) << endl;

return 0;

}



Дадатковыя варыянты:



#include <iostream>

#include <fstream>

#include <vector>

#include <list>

#include <Windows.h>

#define MAX\_SIZE = 10

using namespace std;

struct Nums { //структура для лічб

int number;

};

struct HashTable { // структура для стварэння хэш-табліцы

vector<list<Nums>> table; //будзем рабіць праз вектар, каб запісваць ўсё неекалькі разоў

int size;

int count;

HashTable(int size) : size(size), count(0) {

table.resize(size);

}

void insert(int number) {//функцыя для таго, каб уставіць лічбы ў хэш-табліцу

int ind = hashFunction(number);

for (auto& entry : table[ind]) {

if (entry.number == number) {

return;

}

}

Nums newEntry = { number };

table[ind].push\_back(newEntry);

count++;

}

int search(int number, int& findings) {//функцыя для пошуку лічб

int ind = hashFunction(number);

findings = 0;

for (const auto& entry : table[ind]) {

findings++; //лічым колькасць параўнанняў

if (entry.number == number) {

return findings;

}

}

return findings;

}

int hashFunction(int number) {//рэалізацыя хэш-табліцы

return number % size;

}

};

int main() {

SetConsoleCP(1251);//для таго, каб не было памылак з літарай "і"

SetConsoleOutputCP(1251);

HashTable numbersTable(10);

ifstream fin("FILE.txt");

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

cout << "Памылка!" << endl;

return 1;

}

int number;

while (fin >> number) {

numbersTable.insert(number);

}

fin.close();

int searchNumber;

cout << "Увядзіце: ";

cin >> searchNumber;

int findings;

int result = numbersTable.search(searchNumber, findings);

if (result != 0) {

cout << "Колькасць параўнанняў: " << findings << endl;

}

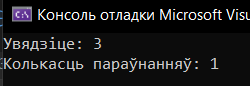
else {

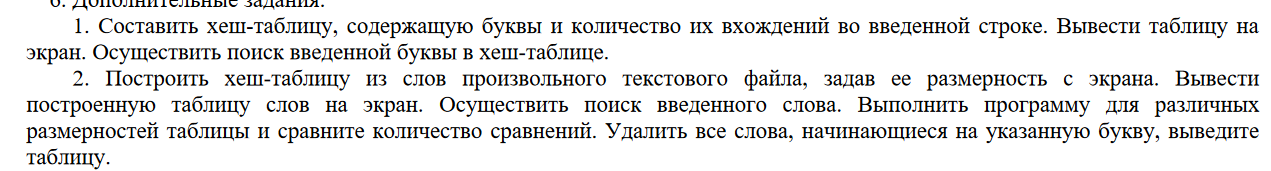
cout << "Такога нумара няма((" << endl;

}

return 0;

}





#include <iostream>

#include <unordered\_map>

#include <string>

#include <Windows.h>

using namespace std;

unordered\_map<char, int> createHashTable(const string& str) {//хэш-функцыя

unordered\_map<char, int> hashTable;

for (char ch : str) {

if (isalpha(ch)) { //праглядаем, каб сімвал быў літарай

char key = tolower(ch);

hashTable[key]++;

}

}

return hashTable;

}

int main() {

SetConsoleCP(1251);//для таго, каб не было памылак з літарай "і"

SetConsoleOutputCP(1251);

string input;

cout << "Увядзіце штосьці: ";

getline(cin, input);

unordered\_map<char, int> hashTable = createHashTable(input);//адпрацоўваем хэш-табліцу

cout << "Хэш-табліца:\n";

for (const auto& i : hashTable) {

cout << "'" << i.first << "': " << i.second << '\n';

}

char look;//пошук літар

cout << "Увядзіце тое, што трэба шукаць: ";

cin >> look;

look = tolower(look);//робім літары маленькімі

if (hashTable.find(look) != hashTable.end()) { //шукаем

cout << "Колькасць зваротаў да літары '" << look << ": " << hashTable[look];

}

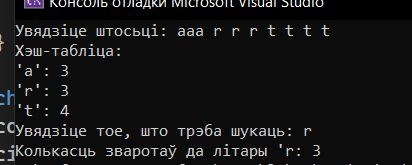
else {

cout << "Такой літары няма((\n";

}

return 0;

}



#include <iostream>

#include <fstream>

#include <unordered\_map>

#include <Windows.h>

#include <vector>

#include <algorithm>

using namespace std;

unordered\_map<string, int> createHashTable(const string& filename, size\_t tableSize) {//хэш-функцыя

ifstream fin(filename);//адчыняем файл

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

cout << "Памылка!!!\n";

return {};

}

unordered\_map< string, int> hashTable(tableSize);

string word;

while (fin >> word) {//адпрацоўваем словы, каб знайсці

word.erase( remove\_if(word.begin(), word.end(), ispunct), word.end());

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

hashTable[word]++;

}

return hashTable;

}

void show(const unordered\_map< string, int>& hashTable) {

cout << "Увядзіце імя файла:\n";

for (const auto& i : hashTable) {

cout << "'" << i.first << "': " << i.second << '\n';

}

}

void removeLetter(unordered\_map< string, int>& hashTable, char letter) {//функцыя, яка выдаляе словы

vector< string> removingWords;

for (auto it = hashTable.begin(); it != hashTable.end(); ++it) {

if (it->first.front() == letter) {

removingWords.push\_back(it->first);

}

}

for (const auto& word : removingWords) {

hashTable.erase(word);

}

}

int main() {

string inputFile;

cout << "Увядзіце імя файла: ";

cin >> inputFile;

size\_t size;

cout << "Увядзіце памер хэш-табліцы: ";

cin >> size;

unordered\_map< string, int> hashTable = createHashTable(inputFile, size);

show(hashTable);

string searchWord;

cout << "Увядзіце слова для пошуку: ";

cin >> searchWord;

if (hashTable.find(searchWord) != hashTable.end()) {

cout << "Мы знайшлі гэта слова " << hashTable[searchWord] << " разоў\n";

}

else {

cout << "Няма такога слова!!\n";

}

char removingLet;

cout << "Увядзіце літару: ";

cin >> removingLet;

removeLetter(hashTable, removingLet);//выдаленне слова

show(hashTable);

return 0;

}

