Банкузов Михаил 7 группа Лабораторная работа №15 Вариант 1



HashTable.h

#ifndef HASHTABLE\_H

#define HASHTABLE\_H

#include <vector>

struct Item {

char name[50];

int year;

};

class HashTable {

private:

int tableSize;

std::vector<Item>\* table;

int hashFunction(int key);

public:

HashTable(int size);

~HashTable();

void insertItem(char name[], int year);

void searchItem(int year);

};

#endif

HashTable.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS // Защита от ошибок компиляции связанных с использованием небезопасных функций

#include "HashTable.h"

#include <iostream>

#include <cstdlib>

#include <cstring>

// Конструктор класса HashTable, который инициализирует таблицу заданного размера

HashTable::HashTable(int size) {

tableSize = size;

table = new std::vector<Item>[tableSize];

}

// Деструктор класса HashTable, который освобождает выделенную память

HashTable::~HashTable() {

delete[] table;

}

// Функция hashFunction, которая использует метод универсального хеширования для вычисления хеш-значения ключа

int HashTable::hashFunction(int key) {

int p = 1;

int a = rand() % (key - 1) + 1;

int b = rand() % key;

int sum = 0;

for (int i = 0; i < 4; i++) {

sum += ((key / p) % 10) \* a + b;

p \*= 10;

}

return sum % tableSize;

}

// Функция insertItem, которая вставляет элемент в таблицу

void HashTable::insertItem(char name[], int year) {

Item item;

strcpy(item.name, name);

item.year = year;

int index = hashFunction(year); // Вычисляем хеш-значение ключа

table[index].push\_back(item); // Вставляем элемент в соответствующий вектор

}

// Функция searchItem, которая ищет элемент в таблице

void HashTable::searchItem(int year) {

int index = hashFunction(year); // Вычисляем хеш-значение ключа

bool found = false;

for (int i = 0; i < table[index].size(); i++) {

if (table[index][i].year == year) { // Если элемент найден, выводим его название

std::cout << "Название товара: " << table[index][i].name << std::endl;

found = true;

}

}

if (!found) { // Если элемент не найден, выводим сообщение об ошибке

std::cout << "Элемент не найден" << std::endl;

}

}Main.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS // Защита от ошибок компиляции связанных с использованием небезопасных функций

#include <iostream>

#include <chrono>

#include "HashTable.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "Rus"); // Установка локали для вывода русских символов

// Создаем хеш-таблицы размерами 10, 100 и 1000

HashTable hashTable10(10);

HashTable hashTable100(100);

HashTable hashTable1000(1000);

// Вставляем случайные элементы в хеш-таблицы

srand(time(NULL));

for (int i = 0; i < 1000; i++) {

char name[50];

int year = rand() % 10000;

sprintf(name, "Товар %d", i);

hashTable10.insertItem(name, year);

hashTable100.insertItem(name, year);

hashTable1000.insertItem(name, year);

}

// Замеряем время поиска элемента в каждой хеш-таблице

int searchYear = rand() % 10000;

cout << "Ищем элемент с годом выпуска " << searchYear << endl;

auto start10 = chrono::high\_resolution\_clock::now();

hashTable10.searchItem(searchYear);

auto end10 = chrono::high\_resolution\_clock::now();

cout << "Время поиска в хеш-таблице размером 10: "

<< chrono::duration\_cast<chrono::microseconds>(end10 - start10).count() << " микросекунд" << endl;

auto start100 = chrono::high\_resolution\_clock::now();

hashTable100.searchItem(searchYear);

auto end100 = chrono::high\_resolution\_clock::now();

cout << "Время поиска в хеш-таблице размером 100: "

<< chrono::duration\_cast<chrono::microseconds>(end100 - start100).count() << " микросекунд" << endl;

auto start1000 = chrono::high\_resolution\_clock::now();

hashTable1000.searchItem(searchYear);

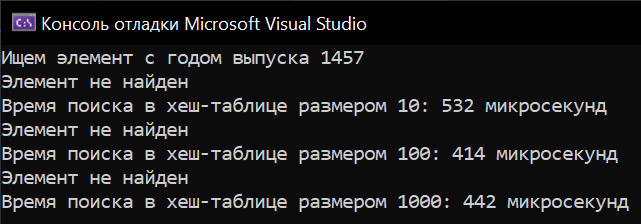
auto end1000 = chrono::high\_resolution\_clock::now();

cout << "Время поиска в хеш-таблице размером 1000: "

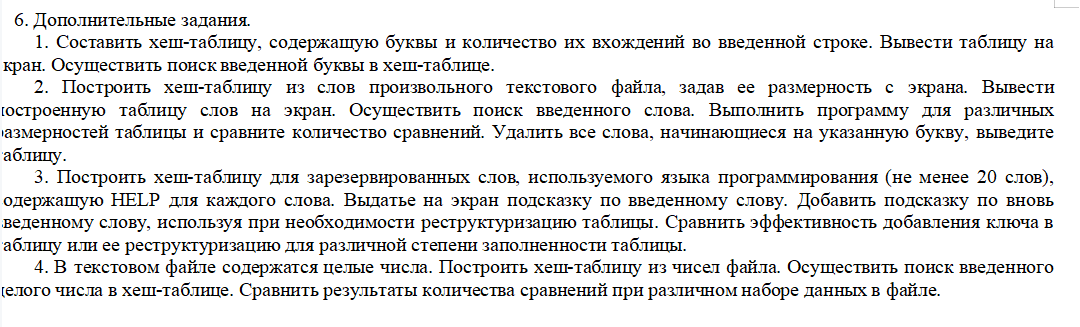
<< chrono::duration\_cast<chrono::microseconds>(end1000 - start1000).count() << " микросекунд" << endl;

return 0;

}



Дополнительные задания



Задание 1

#include <iostream>

#include <string>

#include <unordered\_map>

using namespace std;

int main() {

setlocale(LC\_ALL, "Rus");

string str;

cout << "Введите строку: ";

cin >> str;

unordered\_map<char, int> hashTable; //инициализируем хеш-таблицу

for (char c : str) { //проходим по строке

hashTable[c]++; //увеличиваем значение ключа на 1

}

cout << "Таблица:" << endl; //выводим хеш-таблицу

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

cout << it->first << " -> " << it->second << endl;

}

cout << "Введите символ: "; //поиск символа

char c;

cin >> c;

if (hashTable.find(c) != hashTable.end()) { //если символ найден

cout << "Символ " << c << " встречается " << hashTable[c] << " раз" << endl;

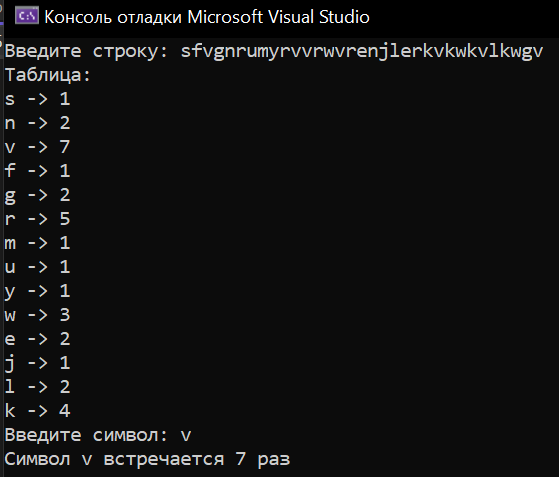
}

else {

cout << "Символ " << c << " не найден." << endl;

}

}



Задание 2

#include <iostream>

#include <fstream>

#include <string>

#include <unordered\_map>

using namespace std;

int main() {

setlocale(LC\_ALL, "Rus");

ifstream fin("text.txt"); //открываем файл

string word;

cout << "Введите размер таблицы: "; //вводим размер таблицы

int size;

cin >> size;

unordered\_map<string, int> hashTable(size); //инициализируем хеш-таблицу

while (fin >> word) { //заполняем таблицу

hashTable[word]++;

}

cout << "Таблица:" << endl; //выводим элементы таблицы

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

cout << it->first << " -> " << it->second << endl;

}

cout << "Введите слово: ";

string searchWord;

cin >> searchWord;

int comparisons = 0;

for (auto it = hashTable.begin(); it != hashTable.end(); ++it) { //ищем количество шагов до слова

comparisons++;

if (it->first == searchWord) {

cout << "Слово " << searchWord << " найдено после " << comparisons << " сравнений" << endl;

break;

}

}

cout << "Введите букву: "; //удаляем все слова с выбранной буквой

char letter;

cin >> letter;

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

if (it->first[0] == letter) {

it = hashTable.erase(it);

}

else {

++it;

}

}

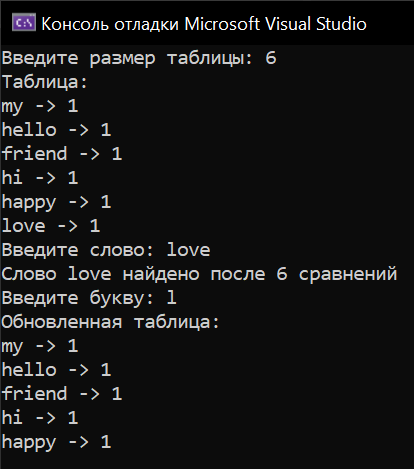
cout << "Обновленная таблица:" << endl; //выводим обновлённую таблицу

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

cout << it->first << " -> " << it->second << endl;

}

}



Задание 3

#include <iostream>

#include <fstream>

#include <unordered\_set>

using namespace std;

int main() {

setlocale(LC\_ALL, "Rus");

std::unordered\_set<int> hashTable;

std::ifstream file("text.txt");

int number;

while (file >> number) {

hashTable.insert(number); // добавление числа в хеш-таблицу

}

std::cout << "Таблица:" << std::endl;

for (const auto& item : hashTable) {

std::cout << item << std::endl;

}

int searchNumber;

std::cout << "Введите число: ";

std::cin >> searchNumber;

auto it = hashTable.find(searchNumber); // поиск числа в хеш-таблице

if (it != hashTable.end()) {

std::cout << "Число найдено" << std::endl;

}

else {

std::cout << "Число не найдено" << std::endl;

}

return 0;

}

