**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЯ ОБРАЗОВАНИЯ**

**“ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет информационных технологий

Кафедра технологий программирования

**Алгоритмы: теория и практика. Структуры данных**

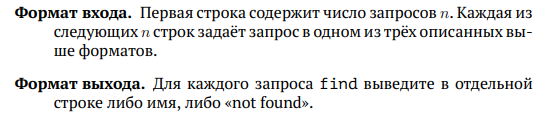
**Модуль №3**

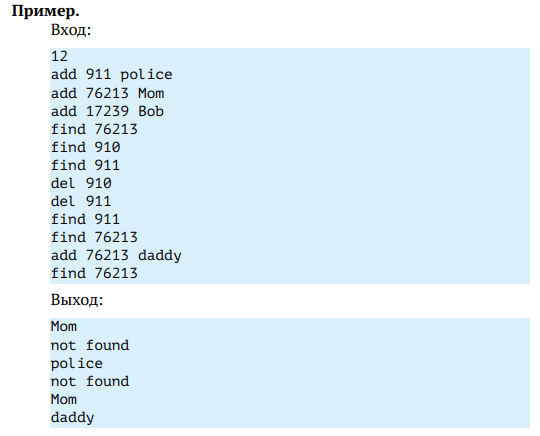
Выполнила студент 2 курса, группа 21-ИТ-1 Катушёнок И.В

Проверила Виноградова А.Д

Полоцк, 2022 г.

Телефонная книга





Решение

#include <iostream>

#include <string>

#include <map>

using namespace std;

int main()

{

long n, key;

string cmd, value;

map<long, string> db;

cin >> n;

for (int i = 0; i < n; ++i)

{

cin >> cmd;

if (cmd == "add")

{

cin >> key >> value;

db[key] = value;

}

else if (cmd == "find")

{

cin >> key;

if (db.find(key) != db.end()) { cout << db[key] << endl; }

else { cout << "not found" << endl; }

}

else if (cmd == "del")

{

cin >> key;

db.erase(key);

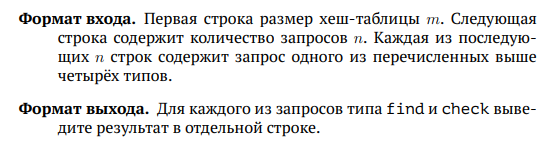
}

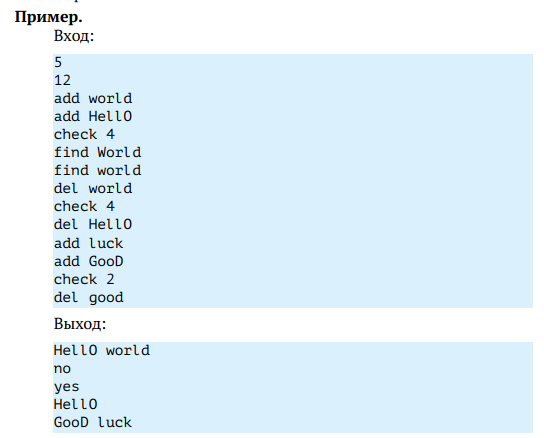
}

return 0;

}

Хеширование цепочками





Решение

using System;

using System.Collections.Generic;

public class MainClass

{

static int m = 1;

static int n = 1;

static int p = 1000000007;

static int x = 263;

static List<String>[] map;

static int GetHash(string str)

{

long result = 0;

for (int i = str.Length - 1; i >= 0; i--)

{

result = ((int)str[i] + result \* x) % p;

//Console.WriteLine("[" + i.ToString() + "]=" + ((int)str[i]).ToString());

//Console.WriteLine("result=" + result.ToString());

}

result = result % m;

return (int)result;

}

static void Add(string word)

{

if (Find(word) == "no")

{

int index = GetHash(word);

map[index].Insert(0, word);

}

}

static string Find(string word)

{

int index = GetHash(word);

int flag = map[index].IndexOf(word);

if (flag == -1)

{

return "no";

}

else

{

return "yes";

}

}

static void Del(string word)

{

int index = GetHash(word);

map[index].Remove(word);

}

static string Check(string number)

{

int index;

int.TryParse(number, out index);

string result = "";

Array.ForEach(map[index].ToArray(), element => {

result += element + " ";

});

return result;

}

public static void Main()

{

string mInput = Console.ReadLine();

int.TryParse(mInput, out m);

map = new List<String>[m];

for (var i = 0; i < m; i++)

{

map[i] = new List<String>();

}

string nInput = Console.ReadLine();

int.TryParse(nInput, out n);

for (var i = 0; i < n; i++)

{

string commandInput = Console.ReadLine();

string[] commandSplit = commandInput.Split(' ');

string command = commandSplit[0];

if (command == "add")

{

Add(commandSplit[1]);

}

if (command == "find")

{

string result = Find(commandSplit[1]);

Console.WriteLine(result);

}

if (command == "del")

{

Del(commandSplit[1]);

}

if (command == "check")

{

string result = Check(commandSplit[1]);

Console.WriteLine(result);

}

}

// Console.WriteLine(lines.Count.ToString());

// Array.ForEach(lines.ToArray(), x => Console.WriteLine(x));

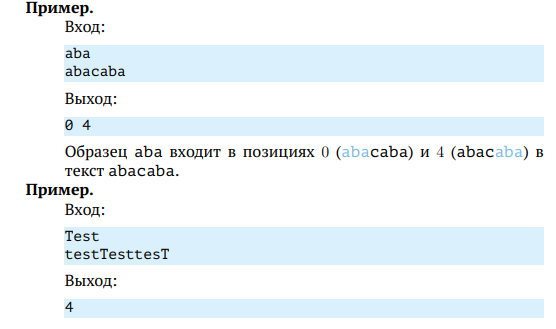
}

}

Поиск образца в тексте

**Формат входа.** Образец Pattern и текст Text.

**Формат выхода.** Индексы вхождений строки Pattern в строку Text в возрастающем порядке, используя индексацию с нуля.



Решение

#include <iostream>

#include <string>

using namespace std;

long M = 1000000007;

long p = 263;

int main()

{

string pattern, text;

cin >> pattern >> text;

long patternHash = 0, curHash = 0;

long power = 1;

for (char c : pattern)

{

patternHash = ((patternHash % M \* p % M) % M + c) % M;

power = (power \* p) % M;

}

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

{

curHash = ((curHash \* p) % M + text[i]);

if (i >= pattern.length())

{

long last = (text[i - pattern.length()] \* power) % M;

curHash = (curHash + M - last) % M;

}

if (curHash == patternHash) { cout << i - pattern.length() + 1 << " "; }

}

return 0;

}