МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра прикладной математики и информатики

ОТЧЁТ

по лабораторной работе №1 «Хэш-таблица»

по дисциплине

Структуры данных

(наименование дисциплины)

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Санников А.Н.

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Зырянов Е.А.

(подпись) (фамилия, и.,о.)

22-ПМ-1

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Задание на лабораторную работу:*** h-функция — середина квадратов, коллизии — квадратичное опробование, ключ — строка.

Реализация:

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

#define FULL\_FACTOR 75

struct Bucket {

string key = "";

string data = "";

};

struct Hashmap {

int count = 0;

Bucket\* element;

int Capacity = 32;

long int hash\_function(string key) {

long int sum = 0;

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

if ((key[i] != ' ') and (key[i] != ',')) {

sum += key[i] \* (i + 1);

}

}

sum \*= sum;

sum >>= 11;

sum = sum % 1024;

long int Hash = abs(sum);

return Hash % Capacity;

}

void init() {

this->element = new Bucket[this->Capacity];

}

void realloc() {

if ((static\_cast<double>(this->count) / this->Capacity) \* 100 >= FULL\_FACTOR) {

int oldCapacity = this->Capacity;

Bucket\* oldElement = this->element;

this->Capacity \*= 2;

this->element = new Bucket[this->Capacity];

this->count = 0;

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

if (oldElement[i].key != "") {

add(oldElement[i].key, oldElement[i].data);

}

}

delete[] oldElement;

}

}

void add(string key, string data) {

realloc();

long int Hash = hash\_function(key);

if (this->element[Hash].key == "") {

this->element[Hash].key = key;

this->element[Hash].data = data;

}

else {

int i = 1;

long int temp = (Hash + i) % this->Capacity;

while (this->element[temp].key != "") {

i += 1;

temp = (Hash + (i \* i)) % this->Capacity;

}

this->element[temp].key = key;

this->element[temp].data = data;

}

this->count += 1;

}

string get(string key) {

int Hash = hash\_function(key);

if (this->element[Hash].key == key) {

return this->element[Hash].data;

}

else {

int j = 1;

for (int i = Hash + 1; i != Hash; i = (i + (j\*j)) % this->Capacity) {

if (this->element[i].key == key) {

return this->element[i].data;

}

j += 1;

}

return "-1";

}

}

void delite(string key) {

long int Hash = hash\_function(key);

if (this->element[Hash].key == key) {

this->element[Hash].key = "";

this->element[Hash].data = "";

this->count -= 1;

}

else {

int j = 1;

for (int i = Hash + 1; i != Hash; i = (i + (j \* j)) % this->Capacity) {

if (this->element[i].key == key) {

this->element[i].key = "";

this->element[i].data = "";

this->count -= 1;

break;

}

j++;

}

}

}

void printAll() {

for (int i = 0; i < this->Capacity; i++) {

if (this->element[i].key != "") {

cout << "Место: " << i << ", Ключ: " << this->element[i].key << ", Значение: " << this->element[i].data << endl;

}

}

cout << " " << endl;

}

void inputFromFile() {

string filePlace;

cout << "Введите имя файла: " << endl;

cin >> filePlace;

ifstream file;

file.open(filePlace);

string str;

string key;

string value;

while (!file.eof()) {

str = "";

key = "";

value = "";

bool flag = false;

getline(file,str);

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

if (str[i] == ':') {

flag = true;

}

if (flag != true) {

key += str[i];

}

if ((flag == true) and (str[i] != ':')) {

value += str[i];

}

}

add(key, value);

}

file.close();

cout << " " << endl;

}

void getFromHashTable() {

cout << "Хотите что-то получить из таблицы?" << endl;

cout << "1 - Да" << endl << "2 - Нет " << endl;

int chose;

string key;

cin >> chose;

if (chose == 1) {

cout << "Введите ключ: ";

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

getline(std::cin, key);

cout << "Значение: " << get(key) << endl;

}

cout << " " << endl;

}

void setToHashTable() {

cout << "Хотите что-то вставить в таблицу?" << endl;

cout << "1 - Да" << endl << "2 - Нет " << endl;

int chose;

string key;

string value;

cin >> chose;

if (chose == 1) {

cout << "Введите ключ: ";

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

getline(std::cin, key); '\n';

cout << "Введите значение: ";

//cin.ignore(numeric\_limits<streamsize>::max(), '\n');

getline(std::cin, value);

add(key, value);

}

cout << " " << endl;

}

void delFromHashTable() {

cout << "Хотите что-то удалить из таблицы?" << endl;

cout << "1 - Да" << endl << "2 - Нет " << endl;

int chose;

string key;

cin >> chose;

if (chose == 1) {

cout << "Введите ключ: ";

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

getline(std::cin, key);

delite(key);

cout << "Значение удалено! " << endl;

}

cout << " " << endl;

}

};

int main() {

setlocale(LC\_ALL, "Russian\_Russia.1251");

Hashmap HT;

HT.init();

HT.inputFromFile();

HT.printAll();

HT.getFromHashTable();

HT.setToHashTable();

HT.printAll();

HT.delFromHashTable();

HT.printAll();

delete[] HT.element;

return 0;

}

Вывод: Реализовали хэш-таблицу.