МИНОБРНАУКИ РОССИИ

Федеральное бюджетное образовательное учреждение высшего профессионального образования

«Ижевский государственный технический университет

имени М.Т. Калашникова»

(ФГБОУ ВПО «ИжГТУ имени М.Т. Калашникова»)

Факультет: «Программное обеспечение»

Кафедра: «Программное обеспечение»

Лабораторная работа №3

Выполнил:

Студент группы Б04-191-1з

Байков Даниил Олегович

Проверил:

Старший преподаватель

Еланцев М.О.

Ижевск, 2020

# Оглавление

[Оглавление 2](#_Toc39239143)

[1. ЗАДАНИЕ 3](#_Toc39239144)

[Реализовать открытую хэш таблицу: 3](#_Toc39239145)

[Хэш таблица должна поддерживать следующие операции: 3](#_Toc39239146)

[2. КОД ПРОГРАММЫ 4](#_Toc39239147)

[Lab3.cpp: 4](#_Toc39239148)

[3. ПРИМЕР РАБОТЫ 8](#_Toc39239149)

# ЗАДАНИЕ

## Реализовать открытую хэш таблицу:

* Ключом хэш-таблицы должна быть строка
* При заполнении хеш-таблицы более чем на половину, должен увеличиваться внутренний буфер (в два раза) и производиться операция перехэширования

## Хэш таблица должна поддерживать следующие операции:

* Инициализация
* Вставка элемента (ключ+значение)
* Получение значения элемента по ключу
* Удаление элемента по ключу
* Освобождение памяти от структуры данных

# КОД ПРОГРАММЫ

## Lab3.cpp:

#include <iostream>

#include <string>

#include <ctime>

using namespace std;

struct Data {

string key = "";

int value;

Data\* next = NULL;

Data(string key, int value) {

this->key = key;

this->value = value;

}

Data() {

}

};

struct HashTable {

private:

Data\* array = NULL;

int size = 0;

int elements = 0;

void isInitialized() {

if (array == NULL) {

cout << "Table not initialized." << endl;

return;

}

}

int h(string key) {

int hash = 0;

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

hash += (int)key[i];

}

hash %= size;

return hash;

}

public:

void initialize() {

if (array) delete array;

array = new Data[10];

size = 10;

elements = 0;

}

void put(string key, int value) {

isInitialized();

int hash = h(key);

if (array[hash].key == "") {

array[hash].key = key;

array[hash].value = value;

elements++;

}

else {

if (array[hash].key == key) {

array[hash].value = value;

return;

}

Data\* data = (array + hash);

while (data->next != NULL) {

if (data->key == key) {

data->value = value;

return;

}

data = data->next;

}

Data\* newData = new Data(key, value);

data->next = newData;

elements++;

}

if (elements > (size / 2)) resize();

}

Data\* get(string key) {

isInitialized();

int hash = h(key);

if (array[hash].key == key) {

Data\* result = new Data(array[hash].key, array[hash].value);

return result;

}

Data\* point = array + hash;

while (point != NULL) {

if (point->key == key) {

Data\* result = new Data(point->key, point->value);

return result;

}

point = point->next;

}

cout << "No element with key = " << key << endl;

return NULL;

}

void remove(string key) {

int hash = h(key);

if (array[hash].key == "") {

cout << "No element with key = " << key << endl;

return;

}

if (array[hash].key == key) {

if (array[hash].next != NULL) {

Data\* toRemove = array[hash].next;

array[hash].key = toRemove->key;

array[hash].value = toRemove->value;

array[hash].next = toRemove->next;

delete toRemove;

elements--;

return;

}

else {

array[hash].key = "";

elements--;

return;

}

}

else {

Data\* point = array + hash;

while (point->next != NULL) {

if (point->next->key == key) {

Data\* toRemove = point->next;

point->next = toRemove->next;

delete toRemove;

elements--;

return;

}

point = point->next;

}

}

cout << "No element with key = " << key << endl;

}

void deleteTable() {

deleteTable(array, size);

}

void deleteTable(Data\* array, int size) {

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

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

Data\* next = array[i].next;

while (next) {

Data\* toRemove = next;

next = next->next;

delete toRemove;

}

}

}

delete[] array;

}

void resize() {

Data\* oldArray = array;

int oldSize = size;

array = new Data[size \* 2];

size = size \* 2;

elements = 0;

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

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

put(oldArray[i].key, oldArray[i].value);

Data\* point = oldArray[i].next;

while (point) {

put(point->key, point->value);

point = point->next;

}

}

}

deleteTable(oldArray, oldSize);

}

void print() {

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

cout << "hash = " << i << " elements: ";

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

Data\* point = array + i;

while (point) {

cout << "[" <<point->key << "]:" << point->value << " ";

point = point->next;

}

}

cout << endl;

}

cout << "Elements = " << elements << " Size = " << size << endl;

}

};

string randomString(string s) {

string result = "";

int length = (rand() % s.length()) / 5 + 1;

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

result = result + s[rand() % s.length()];

}

return result;

}

int randomValue(int length) {

return rand() % length;

}

int main() {

srand(time(NULL));

string\* strings;

int count = 4;

HashTable t;

t.initialize();

string s = "asdfghjkklzxcvbnm,.!@#$%^&asdpo8uinzxc)(&@\*($@)";

strings = new string[count];

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

strings[i] = randomString(s);

t.put(strings[i], randomValue(1000));

}

t.print();

cout << "remove " << count / 2 << " random keys and 2 invalid keys: " << endl;

for (int i = 0; i < count / 2; i++) {

t.remove(strings[randomValue(count)]);

}

t.remove("ASDASDASDASDASDASDASD");

t.remove("AS34#$#$#$DASDASDASDASD");

t.print();

cout << "Add new " << count / 2 - 1 << " keys." << endl;

for (int i = 0; i < count / 2 - 1; i++) {

strings[i] = randomString(s);

t.put(strings[i], randomValue(1000));

}

t.print();

return 0;

}

# ПРИМЕР РАБОТЫ

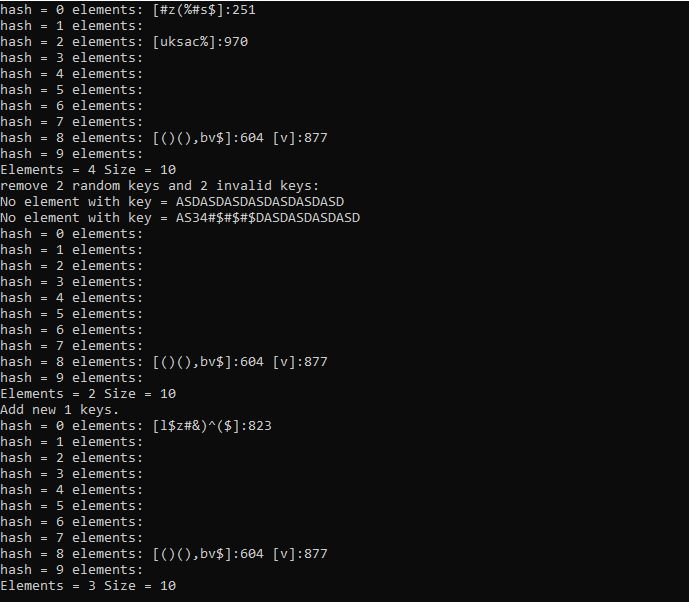


Рисунок 1