Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп’ютерно-інформаційних систем та програмної інженерії

Кафедра програмної інженерії

**З В І Т**

до виконання лабораторної роботи №4

з дисципліни «Алгоритми та структури даних»

на тему «Хешування»

Виконав:

студент групи СП-11

Бондар Віталій

Варіант 4

Тернопіль 2022

**Мета роботи:**набути практичних навичок по реалізації хеш-таблиць з відкритим хешуванням

**Теоретичні відомості**

* **Відкрите хешування** — метод об'єднання всіх елементів, хешованих в одну і ту ж комірку в зв'язаний список, а в  саму комірку записувати покажчик на заголовок такого списку;

**Завдання**

1. Реалізувати функції для робот из хеш-таблицею на основі методу відкритого хешування. В якості елементів хеш-таблиці вибрати стрічки (масив символів). Заповнити хеш-таблицю 15тьма елементами назв міст. Написати функцію PRINT, яка виводить вміст хеш-таблиці таким чином, що елементи, для яких хеш-функція повертає однакові значення (колізії) розташовуються в одному рядку і розділяються пробілами.

IDE: Microsoft Visual Studio 2019

Лістинг програми

//main.cpp

#include <iostream>

#include "funcRealisation.h"

using namespace std;

void PRINT(DICTIONARY A) {

celltype\* current;

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

current = A[i];

if (i)printf("\n");

cout << "[" << i << "] ";

while (current != NULL) {

printf("%s", current->element);

if (current->next != NULL)printf(" ");

current = current->next;

}

}

printf("\n");

}

char cities[][11] = { "Ternopil", "Kharkiv", "Kyiv", "Odessa", "Mariupol", "Lviv", "Lutsk", "Kherson", "Chernivtsi", "Poltava", "Mykolayiv", "Vinnytsia", "Donetsk", "Dnipro", "Rivne" };

int main()

{

DICTIONARY A;

MAKENULL(A);

for (int i = 0; i < 15; i++)

{

INSERT(cities[i], A);

}

PRINT(A);

int index;

start:

cout << endl;

cout << "Enter the index of element which you want to delete(0 to 14): ";

cin >> index;

cout << endl;

if (index >= 0 && index <= 14) {

if (MEMBER(cities[index], A)) {

cout << "You choosed " << cities[index] << endl;

}

DELETE(cities[index], A);

if (!MEMBER(cities[index], A)) {

cout << "New table without " << cities[index] << endl;

cout << endl;

}

PRINT(A);

}

else {

cout << "Incorrect num.Try again!";

cout << endl;

goto start;

}

return 0;

}

/funcRealisation.h

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

typedef char nametype[20];

struct celltype {

nametype element;

celltype\* next;

};

const int DICT\_SIZE = 5;

typedef celltype\* DICTIONARY[DICT\_SIZE];

int hash(nametype x) {

int sum = 0;

for (unsigned int i = 0; i < strlen(x); i++)

sum += (int)x[i];

return sum % (DICT\_SIZE);

}

void MAKENULL(DICTIONARY A) {

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

A[i] = nullptr;

}

bool MEMBER(nametype x, DICTIONARY A) {

celltype\* current;

current = A[hash(x)];

while (current != nullptr) {

if (strcmp(current->element, x) == 0) {

return true;

}

else {

current = current->next;

}

}

return false;

}

void INSERT(nametype x, DICTIONARY A) {

int bucket;

celltype\* old\_header;

if (!MEMBER(x, A)) {

bucket = hash(x);

old\_header = A[bucket];

A[bucket] = new celltype;

strcpy\_s(A[bucket]->element, x);

A[bucket]->next = old\_header;

}

}

void DELETE(nametype x, DICTIONARY A) {

int bucket = hash(x);

celltype\* delElem;

celltype\* current;

if (A[bucket] != nullptr) {

if (strcmp(A[bucket]->element, x) == 0)

{

delElem = A[bucket];

A[bucket] = A[bucket]->next;

delete delElem;

}

else {

current = A[bucket];

while (current->next != nullptr) {

if (strcmp(current->next->element, x) == 0) {

delElem = current->next;

current->next = current->next->next;

delete delElem;

break;

}

else {

current = current->next;

}

}

}

}

}

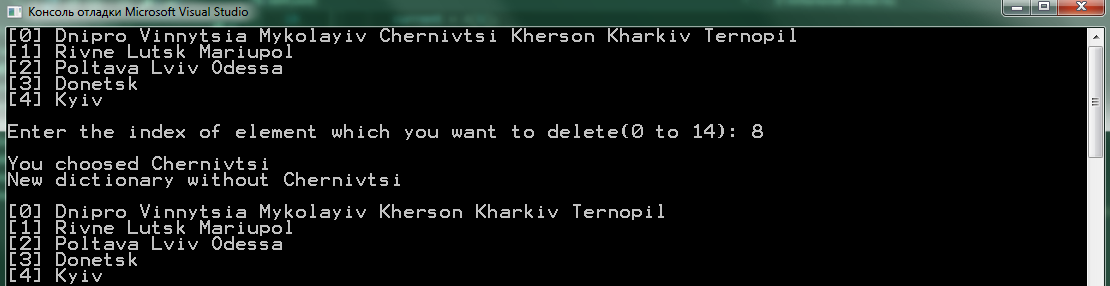


Рис.1 – результат виконання завдання

**Висновок.** Набув практичних навичок по реалізації хеш-таблиць з відкритим хешуванням