Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Институт информационных технологий

Факультет компьютерных технологий

Лабораторная № 2

«Построение словарей на основе метода открытого хеширования данных»

Вариант № 17

Выполнил: студент гр. 881064 Колесников Р.Д.

Проверила: Воробей В.А.

Минск 2019

Лабораторная № 2

«Построение словарей на основе метода открытого хеширования данных»

Вариант № 17

1. Условие:

На основе динамических списков необходимо реализовать словарь. Основные операции, выполняемые над данными словаря:

- поиск,

- вставка;

- удаление.

1. Реализация программы:

#include <vcl.h>

#pragma hdrstop

#include <iostream.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#pragma argsused

struct link {

char\* value;

link\* next;

};

class dict {

private:

link\*\* spisok;

int Hash(char\* value);

public:

dict();

link\* Find(char\* value);

void Insert(char\* value);

bool Delete(char\* value);

~dict();

};

void SearchV(dict\* sp);

void InsertV(dict\* sp);

void DeleteV(dict\* sp);

const int maxstrlen = 100;

dict::dict() {

this->spisok = new link\*[256];

for (int i = 0; i < 256; i++) {

this->spisok[i] = NULL;

}

}

int dict::Hash(char\* value) {

return value[0];

}

link\* dict::Find(char\* value) {

link\* rez = spisok[Hash(value)];

if (rez) {

while (strcmp(rez->value, value) != 0) {

if (rez->next)

rez = rez->next;

else

break;

}

if (strcmp(rez->value, value) == 0)

return rez;

}

return NULL;

}

void dict::Insert(char\* value) {

int hash = Hash(value);

link\* current = spisok[hash];

if (current) {

while (current->next)

current = current->next;

current->next = new link();

current->next->value = value;

} else {

current = new link();

current->value = value;

spisok[hash] = current;

}

}

bool dict::Delete(char\* value) {

int hash = Hash(value);

link\* current = spisok[hash];

if (current) {

link\* prev = current;

while (current->next && strcmp(current->next->value, value) != 0) {

prev = current;

current = current->next;

}

if (current->value) {

if (current->next) {

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

link\* next = current->next->next;

current->next = next;

return true;

}

} else {

if (current == spisok[hash])

spisok[hash] = NULL;

else if (prev) {

prev->next = NULL;

current = NULL;

}

return true;

}

}

}

return false;

}

dict::~dict() {

delete[] spisok;

}

int main(int argc, char\* argv[]) {

dict spisok = dict();

while (1) {

cout << endl << "Dejstvie: " << endl;

cout << "1) Search" << endl;

cout << "2) Insert" << endl;

cout << "3) Delete" << endl;

cout << "4) Exit" << endl

int value = 0;

cin >> value;

cout << endl;

switch (value) {

case 1:

SearchV(&spisok);

break;

case 2:

InsertV(&spisok);

break;

case 3:

DeleteV(&spisok);

break;

case 4:

return 0;

}

}

}

void SearchV(dict\* sp) {

char\* input = new char[maxstrlen];

cout << "Vvedite znachenie" << endl;

cin >> input;

link\* value = sp->Find(input);

if (value)

cout << "Element najden" << endl;

else

cout << "Element ne najden" << endl;

}

void InsertV(dict\* sp) {

char\* input = new char[maxstrlen];

cout << "Vvedite znachenie" << endl;

cin >> input;

sp->Insert(input);

cout << "Element dobavlen" << endl;

}

void DeleteV(dict\* sp) {

char\* input = new char[maxstrlen];

cout << "Vvedite znachenie" << endl;

cin >> input;

if (sp->Delete(input))

cout << "Element udalen" << endl;

else

cout << "Element ne najden" << endl;

}

1. Скрин программы, в котором отражены все результаты работы программы;
2. Контрольные вопросы:

1. С какой целью выполняется хэширование данных

Основная цель хэширования состоит в вычислении по входным данным некоторого числа фиксированного размера, которое как бы будет указывать адрес данных в таблице.

2. Какая хэш-функция является идеальной

Это такая хеш-функция, которая преобразует заранее известное статическое множество ключей в диапазон целых чисел [0, В — 1] без коллизий, т. е. один ключ соответствует только одному уникальному значению.

3. В чем отличия между открытым и закрытым хэшированием данных

В отличие от открытого хеширования закрытое не требует каких-либо дополнительных структур данных. В ячейках таблицы хранятся не указатели, а элементы исходного массива, доступ к каждому из которых осуществляется по хеш-значению ключа, при этом одна ячейка может содержать только один элемент.

1. Объясните суть коллизий при закрытом хэшировании данных

Коллизия – это ситуация, когда происходит попытка поместить элемент *х* в сегмент с номером *h(x)*, который уже занят другим элементом.

1. Перечислите способы борьбы с коллизиями

Существуют два основных подхода к проблеме коллизий:

* Не допускать коллизии вообще, то есть пользоваться детерминированными методами.
* Допускать коллизии и каким-тообразом выходить из них, что достижимо только использованием случайных методов.

Во втором случае также можно выделить два подхода:

* Не обращать внимание на причины возникновения коллизий, а упор делать на способ выхода из них.
* Пытаться предотвращать коллизии тем самым максимально снижая их количество, ну а если коллизии все-такивозникают, то «тяжело» выходить из них.

Все случайные методы основаны на использовании генератора случайных чисел, который позволяют с определенной степенью вероятности избегать коллизий.