БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра программного обеспечения информационных технологий

Факультет ФНиДО

Специальность ПОИТ

Индивидуальная практическая работа № 1.1

«Построение словарей на основе метода открытого хеширования данных»

по дисциплине «Структуры и алгоритмы обработки данных»

Выполнила: Карпеко Н. Г.

Договор № 941 от 20.02.2019 г.

Минск 2020

**Цель работы**: научиться строить словари на базе линейных списков и

открытого хеширования данных.

**Задание.** На основе динамических списков (линейных списков) реализовать словарь. Реализовать выполнение операций: поиск, вставка и удаление данных словаря. В сочетании со списками использовать открытое хеширование данных для построения словарей.

Листинг программы:

**program** IPR1\_1;

**Uses** crt;

**const**

MAXn = 3000; //размерность таблицы

c = 2; //c - фиксированный шаг

**type**

arr = **record**

mas: string[7]; //элемент в таблице

flag: boolean; //для пометки просмотров

**end**;

**var**

table: **array** [0..MAXn] **of** arr; //табл представлена массивом записей

{---- Процедура инициализации массива (хеш-таблицы) --------}

**procedure** initArray;

**var**

j: integer;

**begin**

**for** j := 0 **to** MAXn **do**

**begin**

table[j].mas := ''; //Массив типа string, '' - пустая ячейка.

table[j].flag := true;

**end**; //приготовили место под массив

**end**;

{------- Хеш-функция. Задает адрес для эл-та в массиве. ------}

**function** hash(str: string): integer;

**begin** //для занесения эл-та в таблицу надо, чтобы эл-т был из 2 символов}

**if** length(str) = 1 **then** // str - эл-т, которому надо дать адрес

str := ' ' + str + ' ';

//допустим, сумма кодов 2й и последней букв эл-та

hash := ord(str[2]) + ord(str[length(str)]);

**end**;

{------- Разрешение коллизий (новый адрес, если место занято) --------}

**function** rhash(k: integer; str: string): integer;

//k - k-й эл-т последоват-сти проб; str - текущий эл-т

**begin**

rhash := (hash(str) + c \* k) **mod** MAXn;

**end**;

{---- 1)Процедура добавления эл-та в таблицу --------------------}

**procedure** AddHash(**var** n: integer; **var** str: string);

{str - вводимая строка

n – разм-ть табл = к-во ячеек}

**var**

j, i, k: integer; //j - счетчик эл-тов,

**begin** //если подбирать свободн место для эл-та в табл, начинать с начала

i := 1; { переменная для rhash }

j := 0;

writeln('Размерность таблицы (<=3000): ');

readln(n);

**if** n = 0 **then**

writeln('Таблица заполнена')

**else** // табл не заполнена

**begin**

inc(j); //счетчик для эл-тов

writeln('Введите добавляемый элемент: ');

writeln('Эл-т № ', j); readln(str); // str - вводимая строка

k := hash(str); //получение адреса для хранения эл-та в хеш-табл

**while** true **do**

**if** k <= maxn **then**

**begin**

**if** (table[k].mas = '') **or** (table[k].flag = true) **then**

**begin** //если ячейка свободна

table[k].mas := str; //добавляем эл-т в ячейку

table[k].flag := false; //помечаем как занятую

dec(n);

writeln('Элемент добавлен.');

**break**; //выход для дальнейших операций

**end**;

//если по указанному адресу лежит эл-т-клон

**if** ((table[k].mas <> '') **and** (table[k].mas = str)) **or** (table[k].flag = true) **then**

**begin**

table[k].mas := str; //заменяем его

table[k].flag := false; //помечаем ячейку как занятую

dec(j); //эл-т заменился, значит счетчик не увеличиваем

**break**; //выходим для дальнейших операций

**end**;

//по указанному адресу уже есть эл-т, отличный от текущего

**if** ((table[k].mas <> '') **and** (table[k].mas <> str)) **or**

(table[k].flag = true) **then**

k := rhash(i, str); {меняем адрес эл-ту во избежание коллизии }

//проверяем таблицу на наличие свободного места

**end**

**else**

k := rhash(i, str); //меняем код элемента (случай коллизии)

inc(i); //если адрес, выданный rhash, тоже занят, продолжаем поиск

**end**;

**end**;

{------- 2)Процедура удаления эл-та по адресу в табл ------------}

**procedure** DeleteHash(**var** str: string);

**var**

i, k, n: integer;

**begin**

i := 1; //переменная для rhash

writeln('Введите элемент для удаления:');readln(str);

k := hash(str); //вычисляем адрес этого элемента

**while** true **do**

**begin**

**if** k <= maxn **then**

**begin** //ищем в табл

**if** (table[k].mas = '') **or** ((table[k].mas = str) **and** (table[k].flag = true)) **then**

**begin**

writeln('Элемент не найден');

**break**;

**end**;

//находим

**if** (table[k].mas = str) **and** (table[k].flag = false) **then**

**begin** //Помечаем ячейку как свободную, эл-т в ней не удаляем:

//добавляемый в нее эл-т перезапишет предыдущ. значение

table[k].flag := true;

writeln('Элемент удален');

inc(n);

**break**;

**end**;

**if** n = MAXn **then** writeln('Таблица пуста');

**end**;

//Поиск другого адреса: значит, эл-т был добавлен в таблицу функцией rhash

k := rhash(i, str); //переменная для rhash

inc(i);

**end**;

**end**;

{----- 3)Процедура поиска эл-та по адресу в табл --------------}

**function** SearchHash(**var** str: string): integer;

**var**

i, k: integer;

**begin**

writeln('Введите искомый элемент:');readln(str);

k := hash(str); //вычисляем адрес этого элемента

**while** true **do**

**begin**

**if** k <= maxn **then**

**begin** //ищем в табл

**if** (table[k].mas = '') **or** ((table[k].mas = str) **and** (table[k].flag = true)) **then**

writeln('Элемент не найден');

**break**;

**end**;

//находим

**if** (table[k].mas = str) **and** (table[k].flag = false) **then**

writeln('Элемент', str, 'найден. Его адрес: ', k);

SearchHash := k;

**end**;

**end**;

//---- Основная программа ---------------------------

**VAR**

n: integer;

str: string;

ch: char; //для работы меню

**BEGIN**

**repeat**

clrscr;

Write('Программа для работы с ');TextColor(3);Writeln('хеш-таблицей.');

TextColor(7);Writeln('Выберите желаемое действие:');

Writeln('1) Добавить элемент.');

Writeln('2) Удалить элемент из хеш-таблицы');

Writeln('3) Поиск элемента');

Writeln('4) Выход.');

writeln;

ch := readkey; //ожидаем нажатия клавиши

**case** ch **of** //выбираем клавишу

'1':

**begin** //1) Добавить эл-т

AddHash(n, str);

readkey; //ожидаем нажатия клавиши

**end**;

'2':

**begin** //2) Удалить элемент из хеш-таблицы

DeleteHash(str);

readkey;

**end**;

'3':

**begin** //3) Поиск эл-та

SearchHash(str);

readkey;

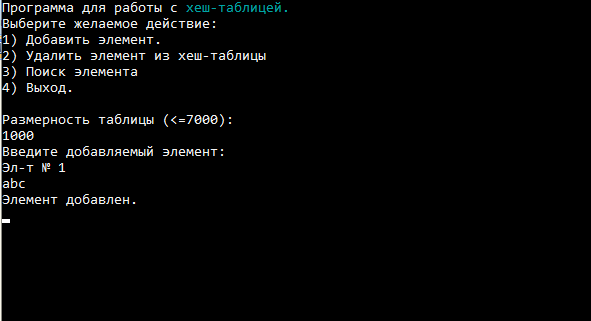
**end**;

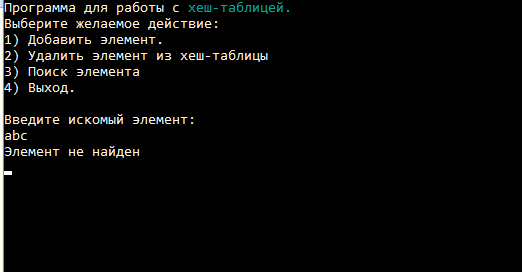
**end**;

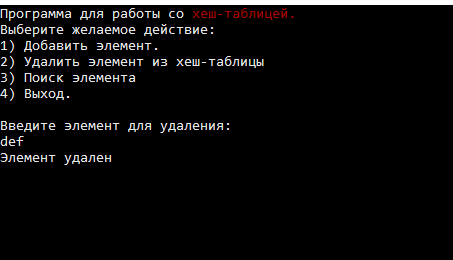
**until** ch = '4';

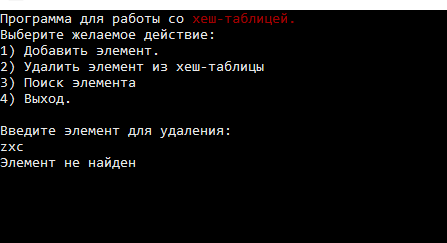
**END**.

Результаты работы программы:









**Контрольные вопросы и задания**

1. С какой целью выполняется хеширование данных?

 Основная цель хэширования – вычисление по входным данным некоторого числа фиксированного размера, которое будет указывать адрес данных в таблице.

2. В чем отличия между открытым и закрытым хешированием 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.

3. Объясните суть коллизий при закрытом хешировании данных.

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

4. Перечислите способы борьбы с коллизиями.

Для борьбы с коллизиями – спец методы, которые сводятся к м-дам "цепочек" и "открытой адресации". Ключи, выдающие одинаковые адреса в таблице – называются ключи-синонимы.

В методе цепочек для разрешения коллизий во все записи вводят указатели для организации списков – "цепочек переполнения". При коллизии (при заполнении таблицы) в список для требуемого адреса хеш-таблицы добавляют еще 1 эл-нт. Метод открытой адресации: пользуясь каким-то алгоритмом, перебирающий эл-нты таблицы, просматривать их в поисках свободного места для новой записи.

Линейное опробование – последовательный перебор эл-тов таблицы с некоторым фиксированным шагом a = h(key) + c\*i*,*

где i – № попытки разрешить коллизию. При шаге, =1 идет последоват-ый перебор всех эл-тов после текущего.