МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Частное учреждение образования

«Гродненский колледж бизнеса и права»

**Лабораторная работа № 23**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

**Тема:** Открытое хеширование

для учащихся 2 курса специальности

2-40 01 01 «Программное обеспечение информационных технологий»

**ЛАБОРАТОРНАЯ РАБОТА № 23**

Тема: Открытое хеширование.

Цель:

Образовательная**:**

* Обучить основным алгоритмам обхода графа и научиться решать задачи обхода графа на основе поиска в ширину и поиска в глубину,

Развивающая:

* научить анализировать алгоритмы обхода графа и научить решать задачи обхода графа на основе поиска в ширину и поиска в глубину,
* создать условия для развития способности четко формулировать свои мысли.

Воспитательная:

* воспитывать в обучающихся средствами урока уверенность в своих силах;

воспитывать сознательное и серьёзного отношения обучающихся к учебной дисциплине, убеждая их в том, что полученные знания пригодятся им в будущей деятельности.

Задачи: Освоение теоретического материала и выполнение индивидуального задания.

**ЗАДАЧИ**

Условие: Для хеш-таблицы m = 10 со списками.

Алгоритм: Предоставлен преподавателю в письменном виде.

Решение:

using System;

using System.Collections;

namespace Program22

{

class Program

{

static void Main()

{

Console.WriteLine("Otkrytor Heshirovanie m=10");

int m = 10;

int[] chisla = new int[m];

Random rand = new Random(((int)DateTime.Now.Ticks));

for (int i = 0; i < chisla.Length; i++)

{

chisla[i] = rand.Next(1, 100);

}

Hashtable ht1 = new Hashtable();

for (int i = 0; i < 10; i++)

{

ht1.Add(chisla[i] + " ", " "+ chisla[i]/2+6);

}

ICollection keys = ht1.Keys;

foreach (string s in keys)

Console.WriteLine(s + ": " + ht1[s]);

Console.WriteLine("Heshirovanie so spiskami m=10");

int[] chisla1 = new int[m];

Random rand1 = new Random(((int)DateTime.Now.Ticks));

for (int i = 0; i < chisla1.Length; i++)

{

chisla1[i] = rand.Next(1, 1000);

}

Hashtable ht2 = new Hashtable();

for (int i = 0; i < 10; i++)

{

ht2.Add(chisla1[i] + " ", " " + chisla1[i] / 2 + 6);

}

ICollection keys1 = ht2.Keys;

foreach (string s1 in keys1)

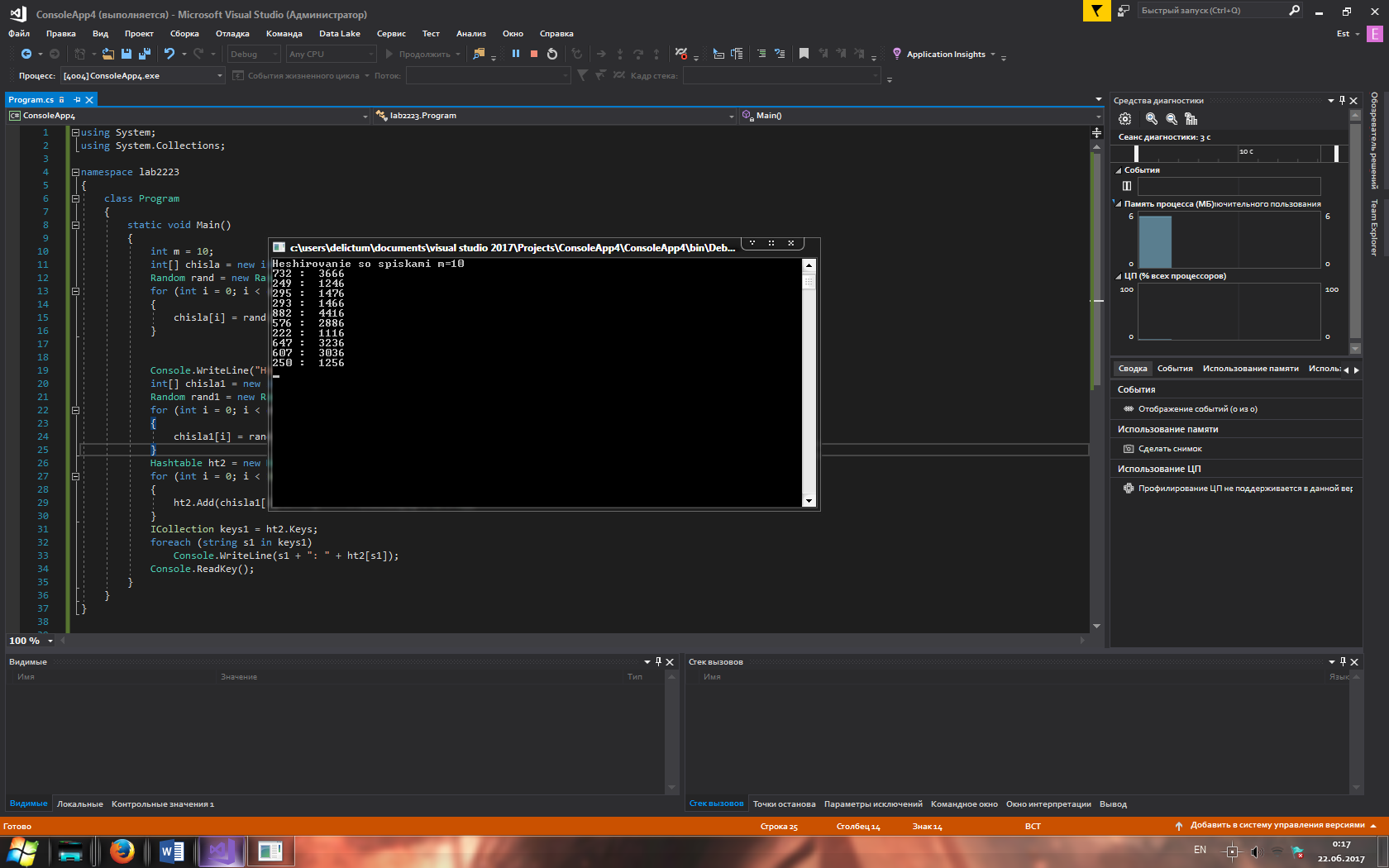
Console.WriteLine(s1 + ": " + ht2[s1]);

Console.ReadKey();

}

}

}



**ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ**

* 1. «Хеш-код» как остаток от деления на число всех возможных «хешей». Хеш-код» как набор коэффициентов получаемого полинома. Криптографические хеш-функции. Контрольные суммы.
  2. Открытая адресация - В массиве H хранятся сами пары ключ-значение. Алгоритм вставки элемента проверяет ячейки массива H в некотором порядке до тех пор, пока не будет найдена первая свободная ячейка, в которую и будет записан новый элемент. Этот порядок вычисляется на лету, что позволяет сэкономить на памяти для указателей, требующихся в хеш-таблицах с цепочками. Последовательность, в которой просматриваются ячейки хеш-таблицы, называется последовательностью проб. В общем случае, она зависит только от ключа элемента, то есть это последовательность h0(x), h1(x), …, hn — 1(x), где x — ключ элемента, а hi(x) — произвольные функции, сопоставляющие каждому ключу ячейку в хеш-таблице. Первый элемент в последовательности, как правило, равен значению некоторой хеш-функции от ключа, а остальные считаются от него одним из приведённых ниже способов. Для успешной работы алгоритмов поиска последовательность проб должна быть такой, чтобы все ячейки хеш-таблицы оказались просмотренными ровно по одному разу.
  3. Идеальное хеширование используется в задачах со статическим множеством ключей (т.е. после того, как все ключи сохранены в таблице, их множество никогда не изменяется) для обеспечения хорошей асимптотики даже в худшем случае. При этом мы можем дополнительно хотеть, чтобы размер таблицы зависел от количества ключей линейно.В таком хешировании для доступа к данным потребуется лишь вычисление хеш-функций (одной или нескольких), что делает данный подход наибыстрейшим для доступа к статическим данным. Данная технология применяется в различных словарях и базах данных, в алгоритмах со статической (известной заранее) информацией.Будем использовать двухуровневую схему хеширования с универсальным хешированием на каждом уровне.
  4. Важное свойство хеш-таблиц состоит в том, что, при некоторых разумных допущениях, все три операции (поиск, вставка, удаление элементов) в среднем выполняются за время. Но при этом не гарантируется, что время выполнения отдельной операции мало́. Это связано с тем, что при достижении некоторого значения коэффициента заполнения необходимо осуществлять перестройку индекса хеш-таблицы: увеличить значение размера массива H и заново добавить в пустую хеш-таблицу все пары.