Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Институт информационных технологий

Факультет компьютерных технологий

Кафедра информационных систем и технологий

**Лабораторная работа №5**

по дисциплине

«Алгоритмы и структуры данных»

|  |  |
| --- | --- |
|  | Выполнил студент группы 181073  Шерстнёв Артур Александрович |
|  | Проверил ассистент кафедры ИСиТ Потоцкий Дмитрий Сергеевич |

Минск 2022

**Вариант 13**

**Цель работы:** научиться выполнять операции с данными, используя методы закрытого хеширования.

**Задание:** ввести массив из n целых чисел из заданного диапазона. Создать хеш-таблицу из М элементов. Осуществить поиск элемента в хеш-таблице. Вывести на экран исходный массив, хеш-таблицу и результат поиска. Задание выбрать в соответствии с номером варианта в таблице 1.

Table

Description automatically generated

**Код программы:**

using System;

using System.Collections.Generic;

namespace lab\_five

{

internal class Program

{

public static int[] hash { get; set; }

public static void AddValue(int value)

{

var k = GetHash(value);

if (hash[k].Equals(0))

{

hash[k] = value;

}

else

{

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

{

if (i == k) continue;

if (hash[i].Equals(0))

{

hash[i] = value;

break;

}

}

}

}

public static string Search(int value)

{

var k = GetHash(value);

if (hash[k] != 0)

{

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

{

if (value.Equals(hash[i]))

{

k = i;

}

}

return $"Hash: {k}; ";

}

else

{

return "Not found";

}

}

private static int GetHash(int value)

{

int result = 0;

if (Type.GetTypeCode(value.GetType()) == TypeCode.Int32)

{

int temp = Convert.ToInt32(value);

while (temp != 0)

{

result += (temp % 10);

temp /= 10;

}

}

return result % hash.Length;

}

private static void Main(string[] args)

{

int arrayLength = 11;

int hashLength = 15;

int start = 44000;

int end = 73000;

Console.WriteLine($"\nRange: {start} - {end}");

Console.Write($"n: {arrayLength}");

Console.WriteLine($"\nM: {hashLength}");

int[] array = new int[arrayLength];

hash = new int[hashLength];

Console.WriteLine();

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

{

Console.Write($"{i + 1}: ");

int value = -1;

while (value < start || value > end)

{

value = Convert.ToInt32(Console.ReadLine());

}

array[i] = value;

AddValue(value);

}

Console.WriteLine("\nArray");

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

{

Console.Write($"{array[i]}, ");

}

Console.WriteLine("\n\nhash");

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

{

if (hash[i] == 0)

{

Console.WriteLine($"{i}: ");

}

else

{

Console.WriteLine($"{i}: {hash[i]} ");

}

}

Console.WriteLine("\nSearch:");

int search = Convert.ToInt32(Console.ReadLine());

Console.WriteLine($"{Search(search)}");

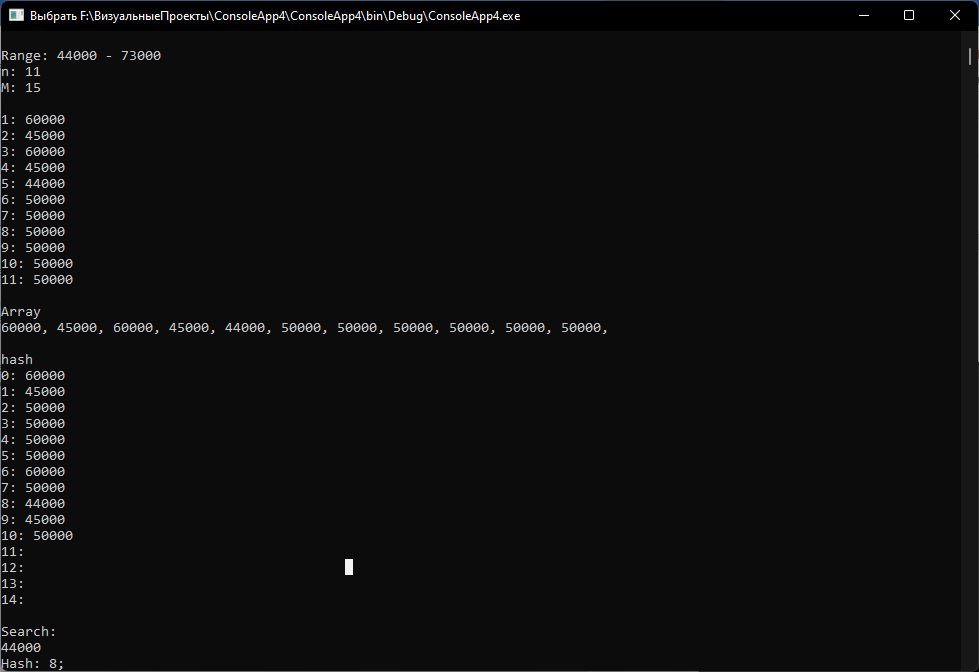
Console.ReadKey();

}

}

}

**Результат:**



**Контрольные вопросы:**

1. В чем отличия между открытым и закрытым хешированием данных?

При закрытом хешировании в таблице сегментов хранятся непосредственно элементы словаря, а не заголовки списков. Поэтому в каждом сегменте может храниться только один элемент словаря. При закрытом хешировании применяется методика повторного хеширования.

Открытое хеширование или Метод цепочек – это технология разрешения коллизий, которая состоит в том, что элементы множества с равными хеш-значениями связываются в цепочку-список.

2. Объясните суть коллизий при закрытом хешировании данных.

Если произойдет попытка поместить элемент х в сегмент с номером h(x), который уже занят другим элементом (такая ситуация называется коллизией), то в соответствии с методикой повторного хеширования выбирается последовательность других номеров сегментов.

3. Перечислите способы борьбы с коллизиями.

При заполнении таблицы могут возникать коллизии, для борьбы с которыми разработаны специальные методы, которые в основном сводятся к методам "цепочек" и "открытой адресации".

В методе цепочек для разрешения коллизий во все записи вводятся указатели, используемые для организации списков – "цепочек переполнения". В случае возникновения коллизии при заполнении таблицы в список для требуемого адреса хеш-таблицы добавляется еще один элемент.

Метод открытой адресации состоит в том, чтобы, пользуясь каким-либо алгоритмом, обеспечивающим перебор элементов таблицы, просматривать их в поисках свободного места для новой записи.

**Вывод:** в ходе данной лабораторной работы я закрепил знания в написании хэш-таблиц используя методы закрытого хэширования. Изучил схемы хэширования с открытой адресацией и на основе цепочек.