Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Институт информационных технологий

Факультет компьютерных технологий

**КОНТРОЛЬНАЯ РАБОТА №2**

**Тема: Особенности работы с внешней памятью. Хешированные и индексированные файлы**

по учебной дисциплине «Алгоритмы и структуры данных»

Выполнил:

студент гр. 381574

Жгуновский О.Б.

Минск — 2024

**Интерпретация задачи**

На основе динамических списков реализовать словарь и сохранить его во внешней памяти. Реализовать операции, выполняемые над данными словаря: поиск, вставка и удаление. В сочетании со списками для построения и обработки словарей использовать хешированные файлы. Ввод должен производиться как в ручном режиме, так и автоматически.

**Код программы**

**using System;**

**using System.Collections.Generic;**

**using System.IO;**

**using System.Linq;**

**class DictionaryEntry**

**{**

**public string Key { get; set; }**

**public string Value { get; set; }**

**public DictionaryEntry(string key, string value)**

**{**

**Key = key;**

**Value = value;**

**}**

**}**

**class HashFileDictionary**

**{**

**private const string FileName = "dictionary.dat";**

**private const int TableSize = 100; // Размер хеш-таблицы**

**private List<DictionaryEntry>[] table;**

**public HashFileDictionary()**

**{**

**table = new List<DictionaryEntry>[TableSize];**

**for (int i = 0; i < TableSize; i++)**

**table[i] = new List<DictionaryEntry>();**

**LoadFromFile();**

**}**

**private int Hash(string key)**

**{**

**return Math.Abs(key.GetHashCode()) % TableSize;**

**}**

**public void Insert(string key, string value)**

**{**

**int index = Hash(key);**

**var entry = new DictionaryEntry(key, value);**

**table[index].Add(entry);**

**SaveToFile();**

**}**

**public string Search(string key)**

**{**

**int index = Hash(key);**

**var entry = table[index].FirstOrDefault(e => e.Key == key);**

**return entry?.Value ?? "Не найдено";**

**}**

**public void Delete(string key)**

**{**

**int index = Hash(key);**

**var entry = table[index].FirstOrDefault(e => e.Key == key);**

**if (entry != null)**

**{**

**table[index].Remove(entry);**

**SaveToFile();**

**}**

**}**

**private void LoadFromFile()**

**{**

**if (File.Exists(FileName))**

**{**

**foreach (var line in File.ReadAllLines(FileName))**

**{**

**var parts = line.Split('|');**

**if (parts.Length == 2)**

**{**

**Insert(parts[0], parts[1]);**

**}**

**}**

**}**

**}**

**private void SaveToFile()**

**{**

**using (var writer = new StreamWriter(FileName))**

**{**

**foreach (var bucket in table)**

**{**

**foreach (var entry in bucket)**

**{**

**writer.WriteLine($"{entry.Key}|{entry.Value}");**

**}**

**}**

**}**

**}**

**}**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**var dictionary = new HashFileDictionary();**

**while (true)**

**{**

**Console.WriteLine("Выберите операцию: 1 - Вставка, 2 - Поиск, 3 - Удаление, 0 - Выход");**

**var choice = Console.ReadLine();**

**switch (choice)**

**{**

**case "1":**

**Console.Write("Введите ключ: ");**

**string insertKey = Console.ReadLine();**

**Console.Write("Введите значение: ");**

**string insertValue = Console.ReadLine();**

**dictionary.Insert(insertKey, insertValue);**

**break;**

**case "2":**

**Console.Write("Введите ключ для поиска: ");**

**string searchKey = Console.ReadLine();**

**Console.WriteLine($"Значение: {dictionary.Search(searchKey)}");**

**break;**

**case "3":**

**Console.Write("Введите ключ для удаления: ");**

**string deleteKey = Console.ReadLine();**

**dictionary.Delete(deleteKey);**

**break;**

**case "0":**

**return;**

**default:**

**Console.WriteLine("Некорректный ввод.");**

**break;**

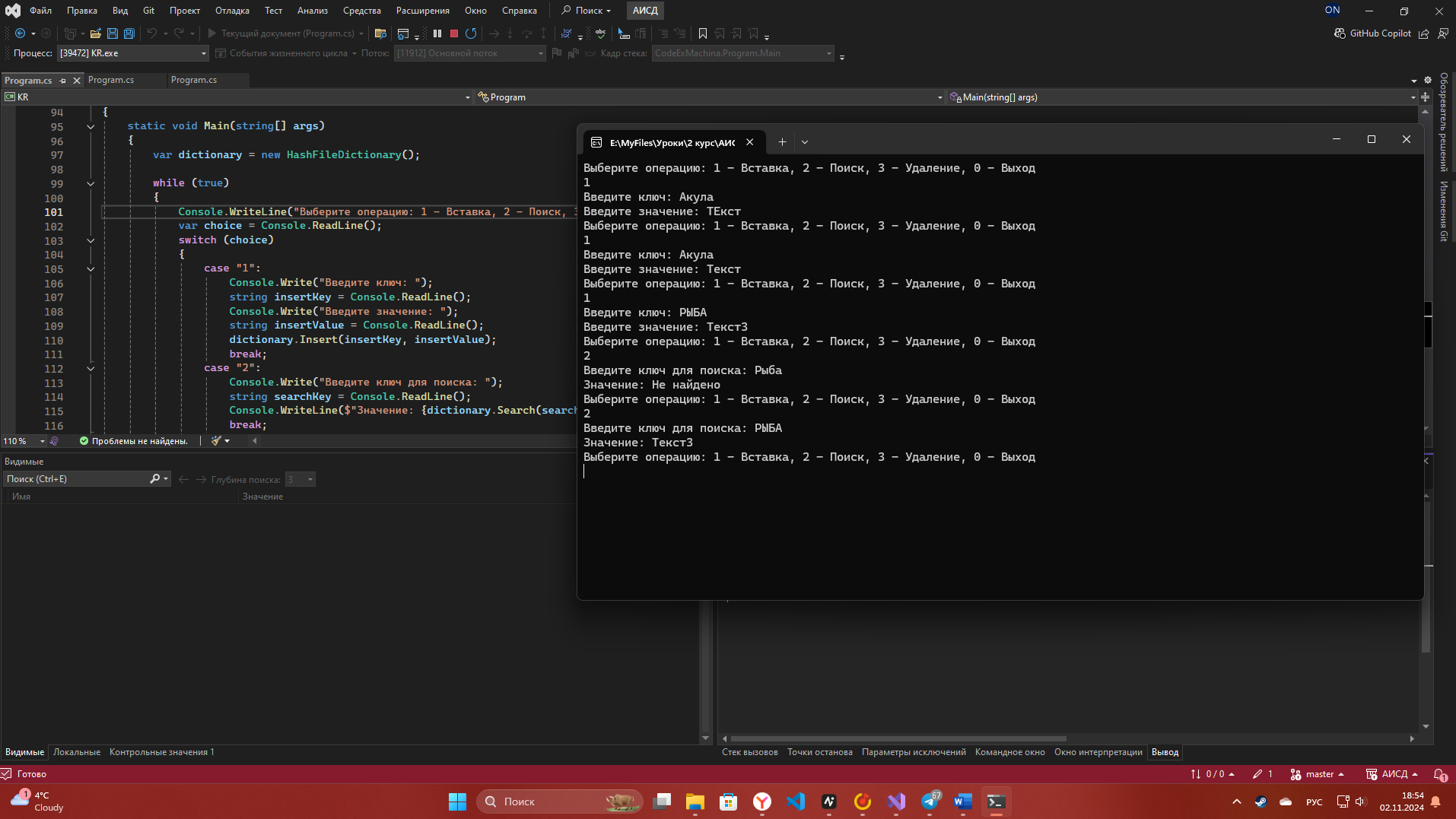
**}**

**}**

**}**

**}**

**Демонстрация работы программы**

Рисунок 1 – демонстрация вставки и поиска

После закрытия и открытия программы всё происходит как надо.

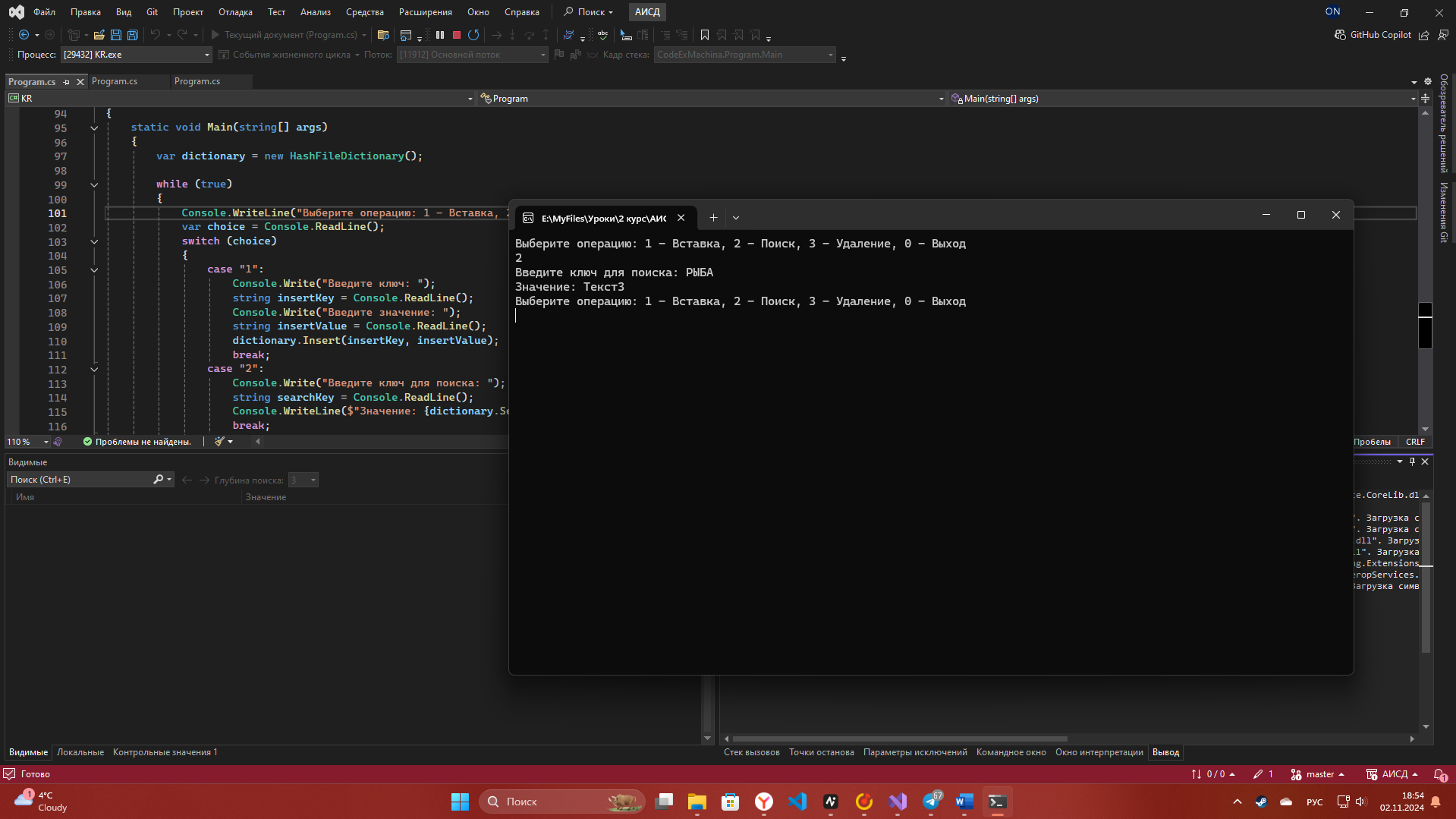


Рисунок 2 – демонстрация что файл не очищается при закрытии

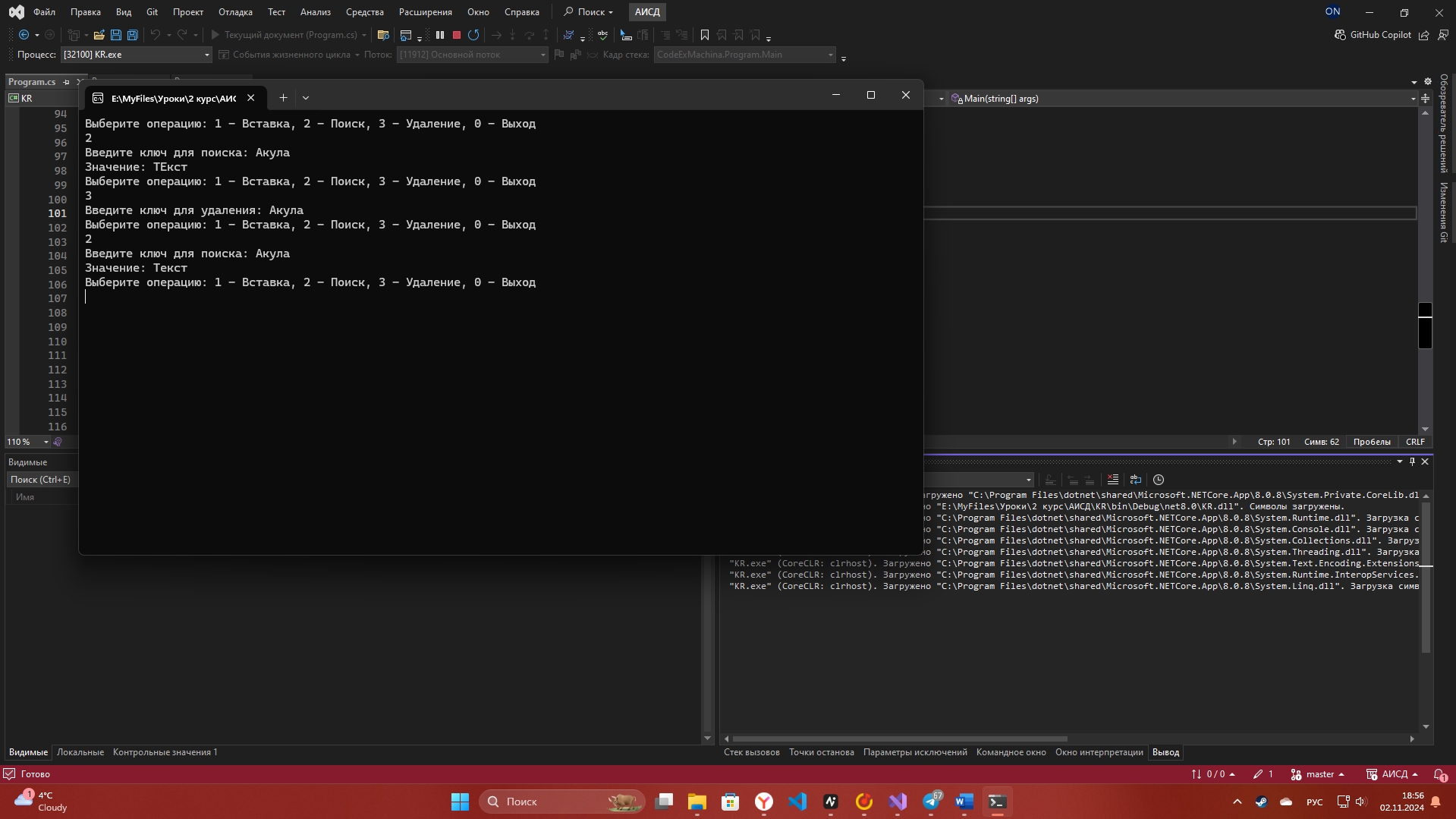


Рисунок 3 – демонстрация удаления элемента

**Контрольные вопросы**

*В чем заключаются особенности операций с данными, хранящимися*

*во внешней памяти?*

1. медленнее чем с оперативной памятью;
2. данные во внешней памяти организованы в блоки;
3. возможность многопоточности.

*Что общего и различного у хешированных файлов и алгоритма открытого хеширования, предназначенного для работы с данными в оперативной памяти?*

Из общего – оба используют хеш-функции, у них общая цель (эффективный доступ к данным) и схожая структура.

Из отличий – данные хранятся в разных местах (в ОЗУ и на диске), по-разному обрабатывают коллизии. Ну и естественно они отличаются по скорости.

*Почему удается ускорить обработку данных при использовании*

*индексированных файлов?*

Индексы позволяют быстро находить данные в файле, минимизируя необходимость последовательного сканирования. К тому же, уменьшают количество запросов для поиска значения по файлу.