Доклад

По Алгоритми и структури от данни

Тема – Хеширащи алгоритми

Изготвил: Валентин Кьосев 12 ,,а‘‘ клас

1. Хеширащи алгоритми. Хеш-таблица
2. Имплементация на хеш-таблица
3. **Хеширащи алгоритми. Хеш-таблица**

Хеширащите алгоритми са ключови компоненти в областта на компютърните науки, използвани за бързо търсене, сортиране и уникално идентифициране на данни. Хеш-таблицата е структура от данни, която използва хеш-функция за преобразуване на входни данни в индекс на масив. Това позволява бърз достъп до данни, тъй като времето за достъп е почти константно, независимо от броя на елементите в хеш-таблицата.

**Основните хеширащи алгоритми:**

**Метод на деление (Division Method)**

При този метод се изчислява остатъка при деление на ключа на броя на "слотовете" (размера на хеш-таблицата).

**Метод на умножение (Multiplication Method)**

При този метод се умножава ключа с число

A между 0 и 1, след което се изважда цялата част от резултата и този остатък се използва като индекс в таблицата:

h(k)=⌊m⋅(k⋅Amod1)⌋

Където m е размерът на таблицата, A е число между 0 и 1, а k е ключът.

**Универсални хеш-функции (Universal Hashing)**

Този метод използва семейство от хеш-функции и се избира случайно една от тях за всяко приложение. Това помага за избягване на колизии (конфликти при хеширане на различни ключове в един и същи слот).

**Хеш-функции с отворено разсейване (Open Addressing)**

При този вид хеш-таблица, при колизия (когато два ключа хешират до един и същи индекс), се извършва пробване на различни места в таблицата, докато се намери празно място. Представители на този метод са Linear Probing, Quadratic Probing, и др.

**Хеш-таблица**

Хеш-таблица (hash table) е структура от данни, използвана за ефективно съхранение и бързо търсене на двойки "ключ-стойност". Тя осигурява бърз достъп до данни, като комбинира предимствата на масива и асоциативния масив (познат още като речник или хеш-мап).

Основната идея на хеш-таблицата е да се използва функция, наречена хеш-функция, която приема ключ и го преобразува в индекс във вътрешен масив от данни. Този индекс отговаря на мястото в масива, където стойността, свързана с ключа, трябва да бъде съхранена или взета.

**Основните операции, които могат да бъдат изпълнявани върху хеш-таблица:**

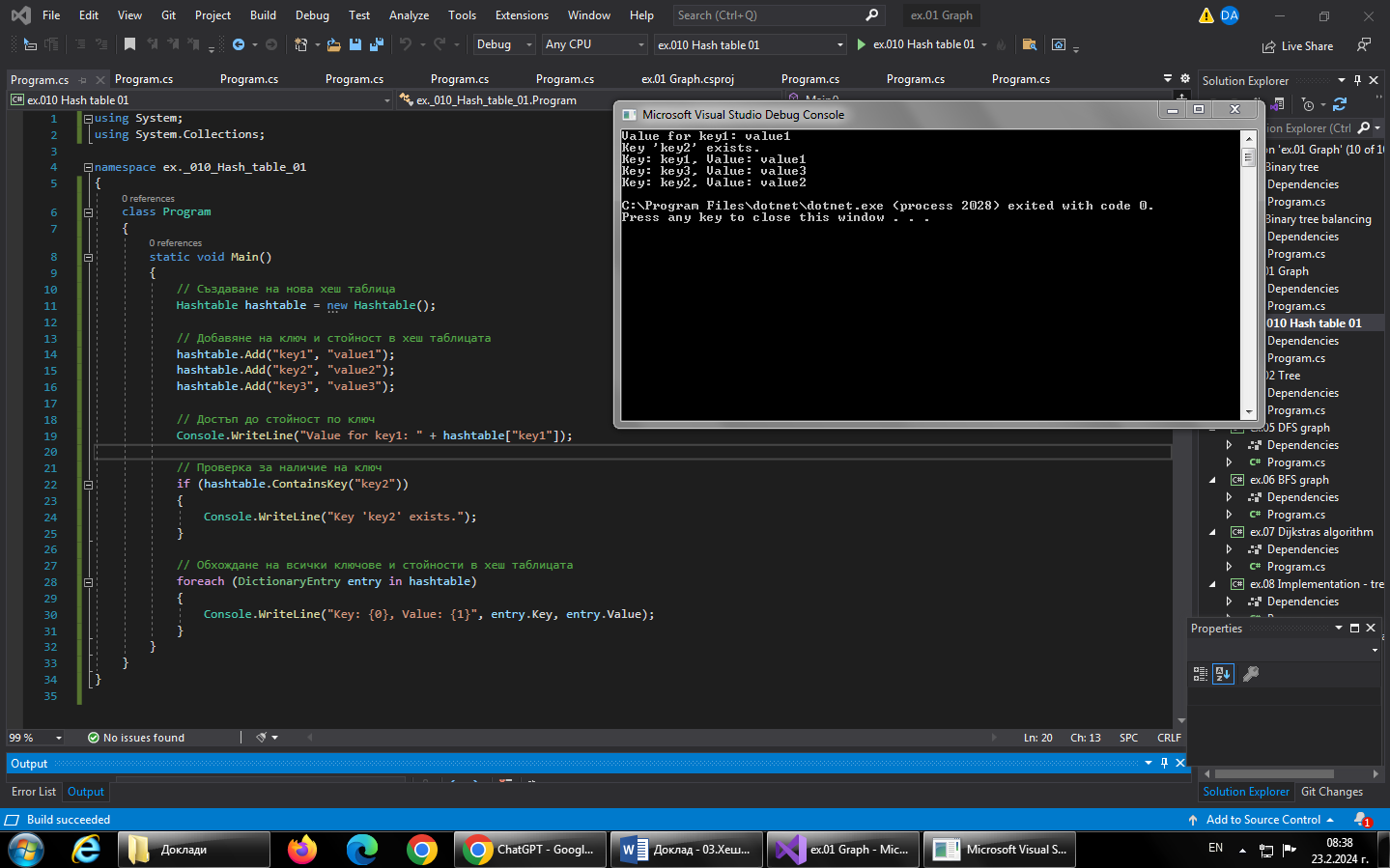
* **Добавяне (Insert):** За да добавим нова стойност в хеш-таблицата, първо изчисляваме хеш от ключа. След това използваме този хеш като индекс в масива от данни и съхраняваме стойността на този индекс.
* **Търсене (Search):** За да намерим стойността, свързана с даден ключ, отново изчисляваме хеш от ключа. След това отиваме на индекса, който е получен от хеша, и връщаме стойността, ако такава съществува.
* **Премахване (Delete):** За да премахнем стойност от хеш-таблицата, най-напред търсим стойността по ключа и след това я изтриваме от масива.

Важно е да се отбележи, че при хеш-таблиците могат да възникнат колизии - ситуации, когато два различни ключа имат един и същ хеш. За да се справим с това, съществуват различни методи за обработка на колизии, като например методът на отворената адресация и методът на цепене.

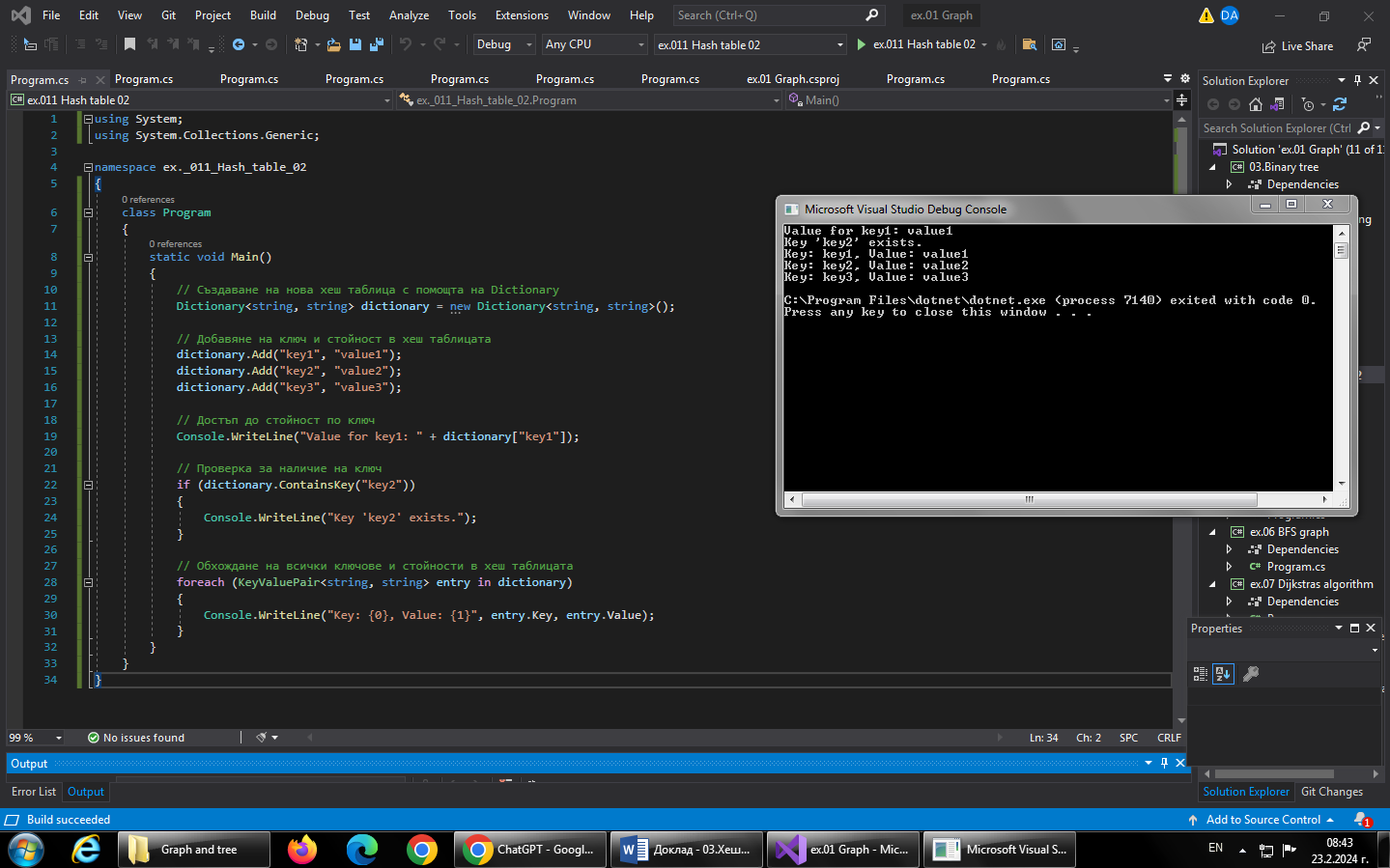
**Хеш-таблица – Пример**

Хеш таблиците в C# обикновено се реализират с помощта на класовете Hashtable или Dictionary. Ето няколко примера за това как да използвате хеш таблиците в C#:

**Пример с `Hashtable`:**

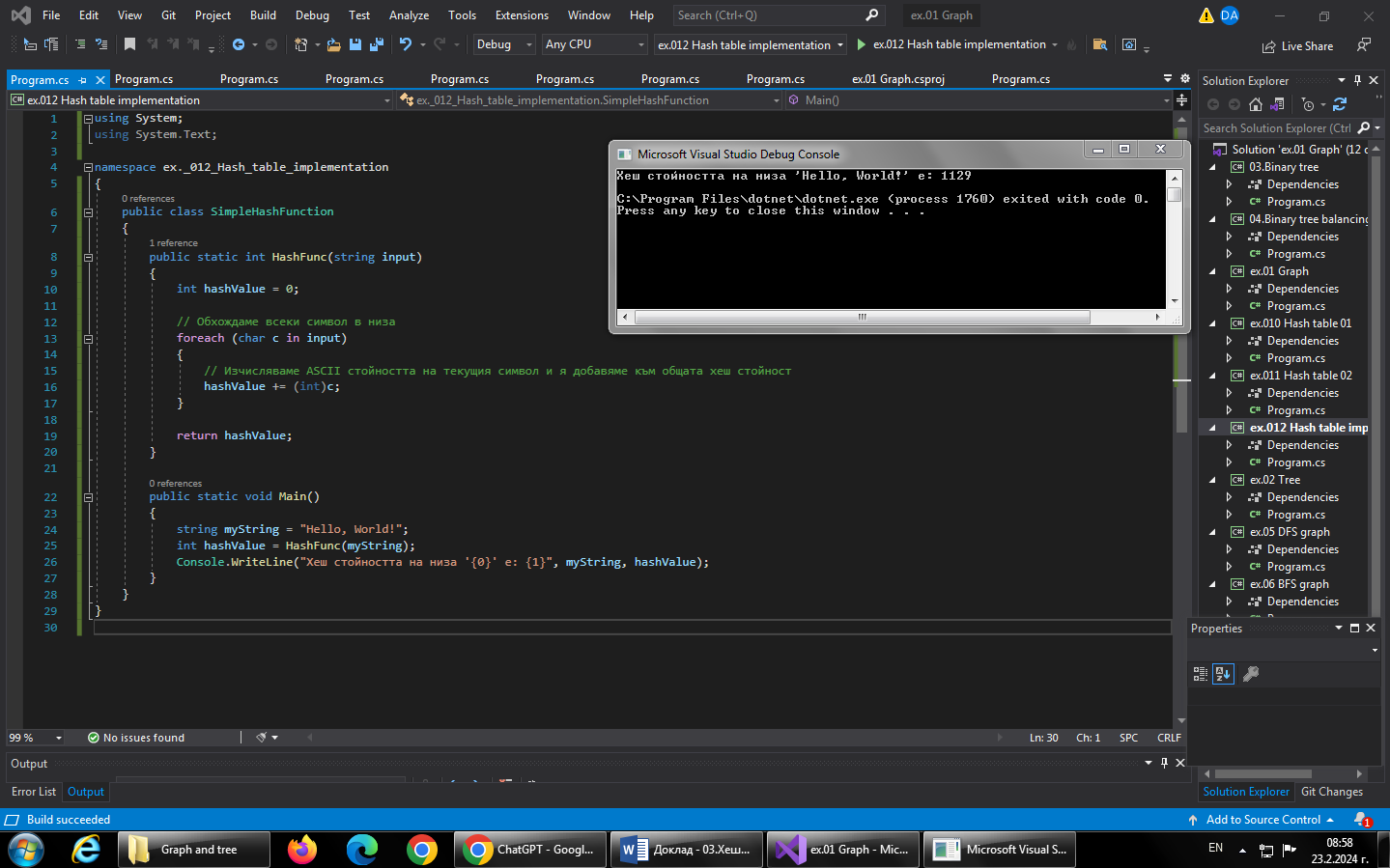
****

**Пример с `Dictionary`:**

****

1. **Имплементация на хеш-таблица**

**Имплементация на проста хеш функция на C# - Пример**

****

**Използвана литература/Източници:**

<https://introprogramming.info/intro-java-book/read-online/glava18-rechnici-hesh-tablici-i-mnojestva/>

<https://learn.fmi.uni-sofia.bg/pluginfile.php/187838/mod_resource/content/1/SDA_2018-2019_lecture_9.pdf>

<https://softuni.bg/blog/what-is-hash-table>

<https://academy.binance.com/bg/articles/what-is-hashing>