



Национална програма
"Обучение за ИТ умения и кариера"
<https://it-kariera.mon.bg>

Министерството на
образованието и науката
<https://www.mon.bg>



Хеширане и хеш таблицы

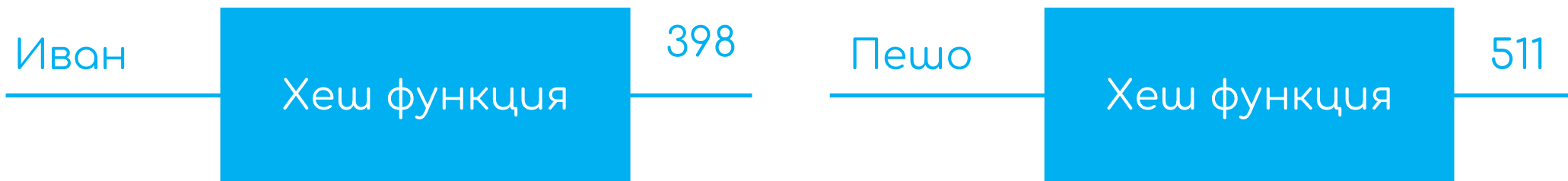
Алгоритми и структури от данни

Съдържание

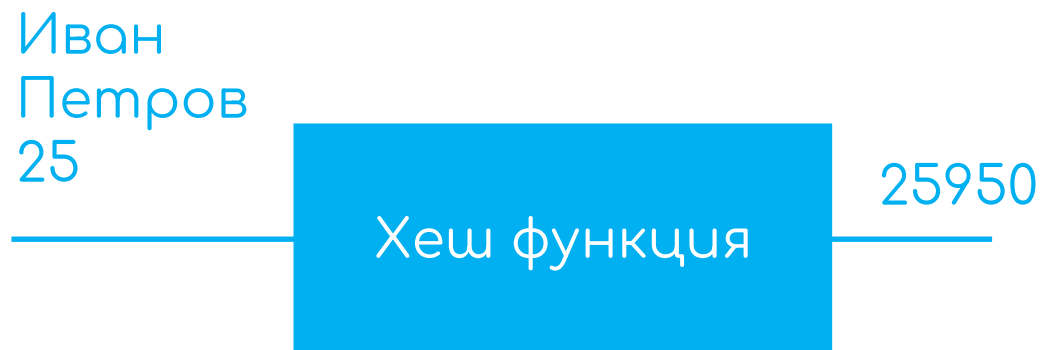
- Хеширащи функции
- Хеш таблици
- Управление на колизии в хеш таблици
- Упражнения: хеш таблици

Хеширащи функции [1/2]

Хеширащите функции конвертират ключ от произволен тип до стойност от целочислен тип



```
class Person
{
    string firstName;
    string lastName;
    int age;
```

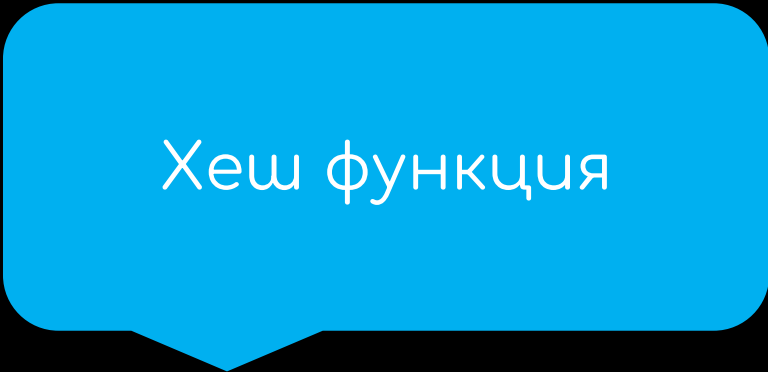


Хешираци функции [2/2]

```
class Person
{
    string firstName;
    string lastName;
    int age;

    public override int GetHashCode()
    {
        int firstNameHash = firstName.GetHashCode() * age;
        int lastNameHash = lastName.GetHashCode() * age;

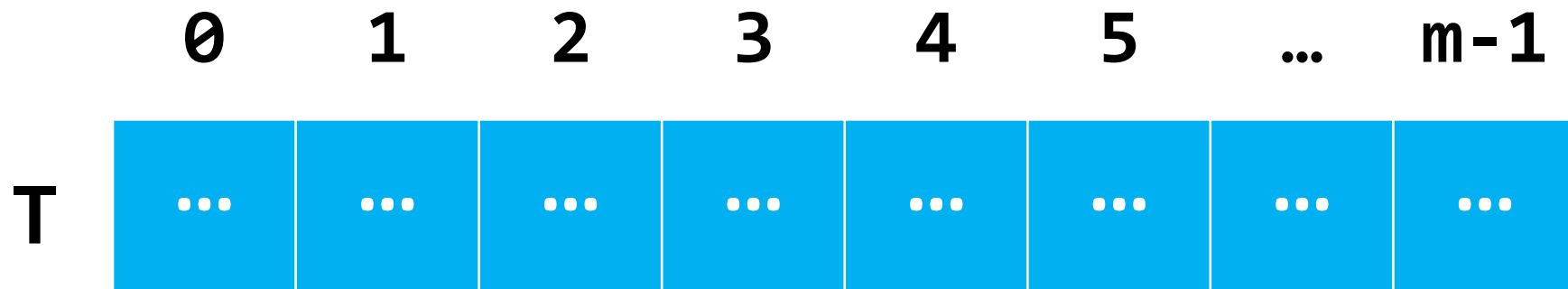
        return firstNameHash + lastNameHash;
    }
}
```



Хеш функция

Хеш таблица

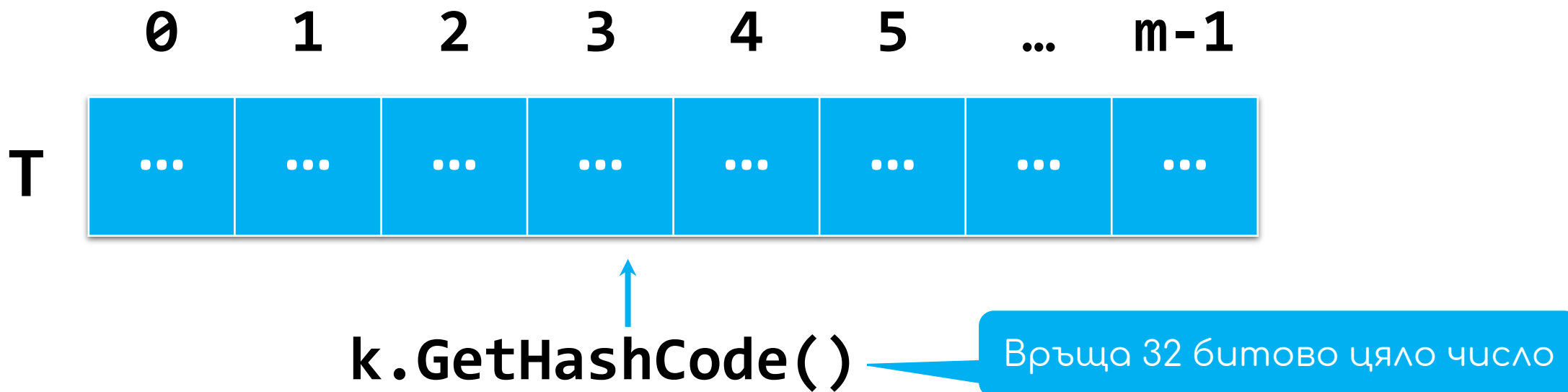
- Хеш таблица е стандартен масив, който съдържа набор от наредени двойки {ключ, стойност}
- Техниката, с която се определя кой ключ на коя позиция в масива да се съхрани се нарича хеширане



Хеш таблица с
размер m

Хеш функции и хеширане

- Хеш таблицата (масива) има **m** позиции, индексирани от **0** до **m-1**
- Хеш функцията конвертира **ключовете** до **индекси в масив**



Хеширащи функции [1/2]

- Перфектно хеширане
 - Перфектно хешираща функция е тази $f(k)$, която прави 1:1 съответствие за всяко уникално k към уникално число в интервала $[0, m-1]$
 - Перфектно хеширащата функция свързва всеки ключ към **уникално** цяло число в рамките на конкретен интервал
- В повечето случаи перфектното хеширане е невъзможно

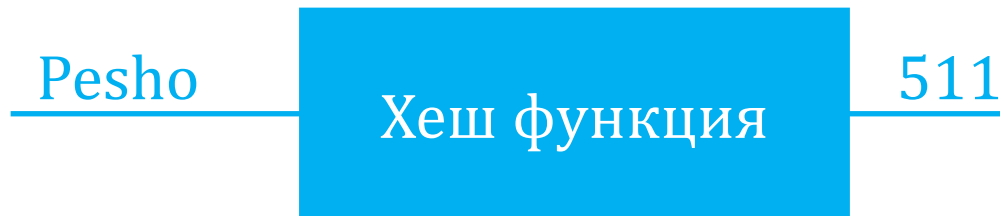
Хеширащи функции [2/2]

- Свойства на добрата хешираща функция
 - **Консистентност** - еднакви ключове трябва да произвеждат един и същ хеш
 - **Ефективност** - ефективни при изчисляването на хеш
 - **Равномерност** - хешовете, произведени от хеширащата функция трябва да се равномерно разпределени

Модулна аритметика и хеш таблици

- Имаме масив с размер 16
- Въвеждаме "Pesho"

511 е извън
размера на хеш
таблицата



- Използваме остатъка от делението за да извлечем валидна позиция:

`GetHashCode() / Array.Length`

$$511 \% 16 = 15$$

0

1

2

3

4

5

6

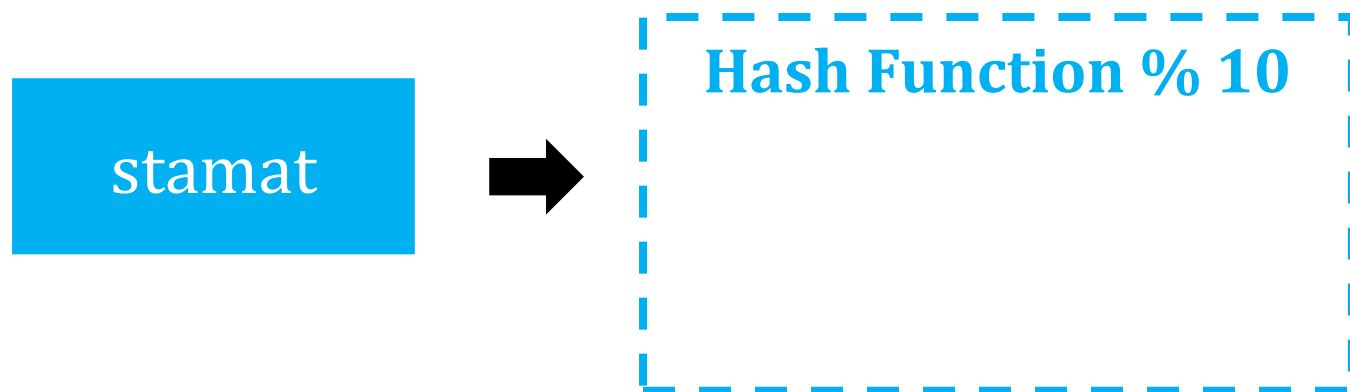
7

...

...

15

Работа с хеш таблица [1/6]



0

1

2

3

4

5

6

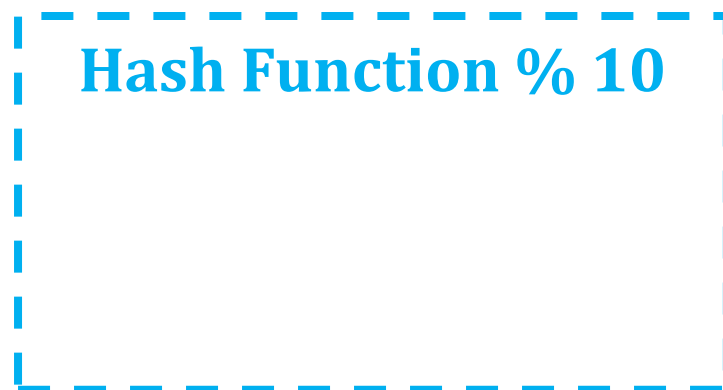
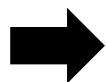
7

8

9

Работа с хеш таблица [2/6]

mitko



stamat



0

1

2

3

4

5

6

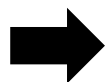
7

8

9

Работа с хеш таблица [3/6]

ivan



Hash Function % 10

stamat



0

1

2

3

4

5

6

mitko

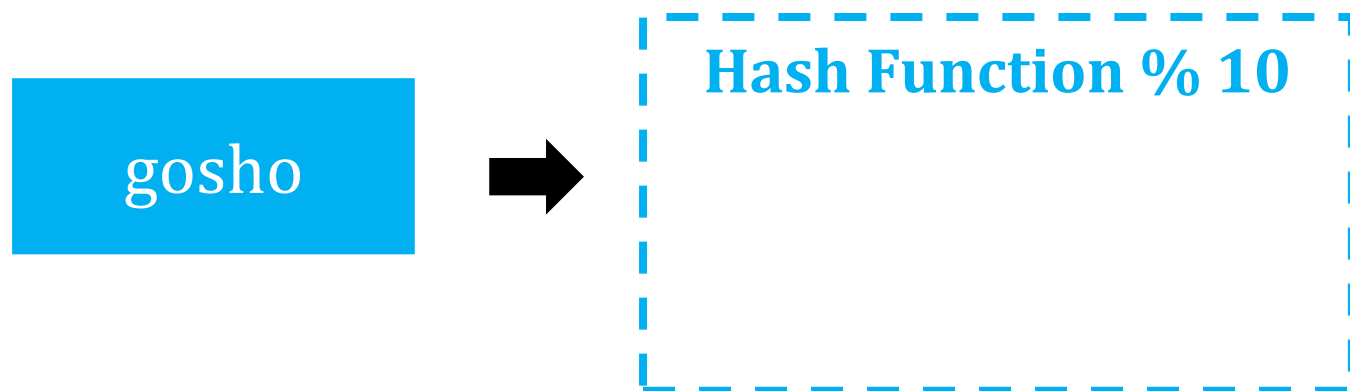


7

8

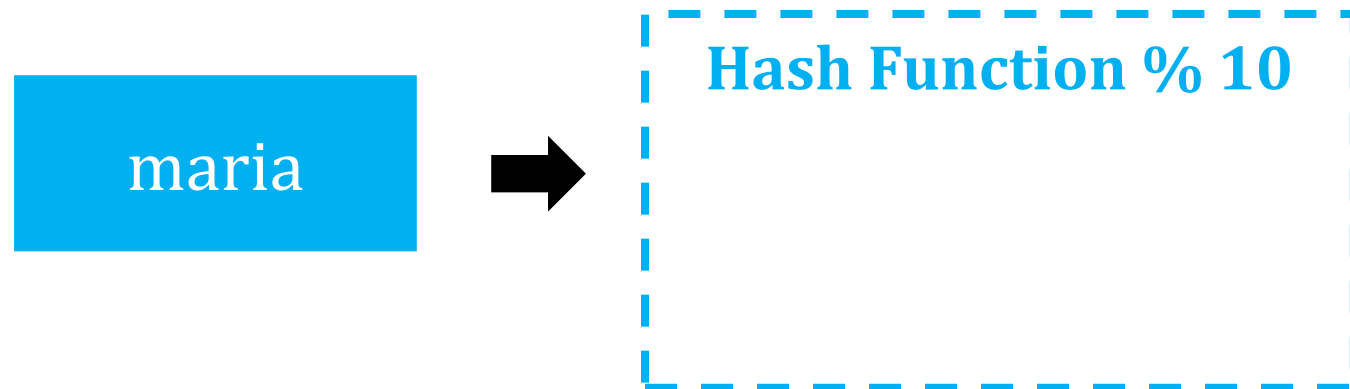
9

Работа с хеш таблица [4/6]



stamat	→	0
		1
		2
		3
		4
ivan	→	5
		6
mitko	→	7
		8
		9

Работа с хеш таблица [5/6]



stamat	→	0
		1
		2
		3
		4
ivan	→	5
		6
mitko	→	7
		8
gosho	→	9

Работа с хеш таблица [6/6]

Колузия

Hash Function % 10

stamat

→ 0

1

2

maria

3

4

ivan

→ 5

6

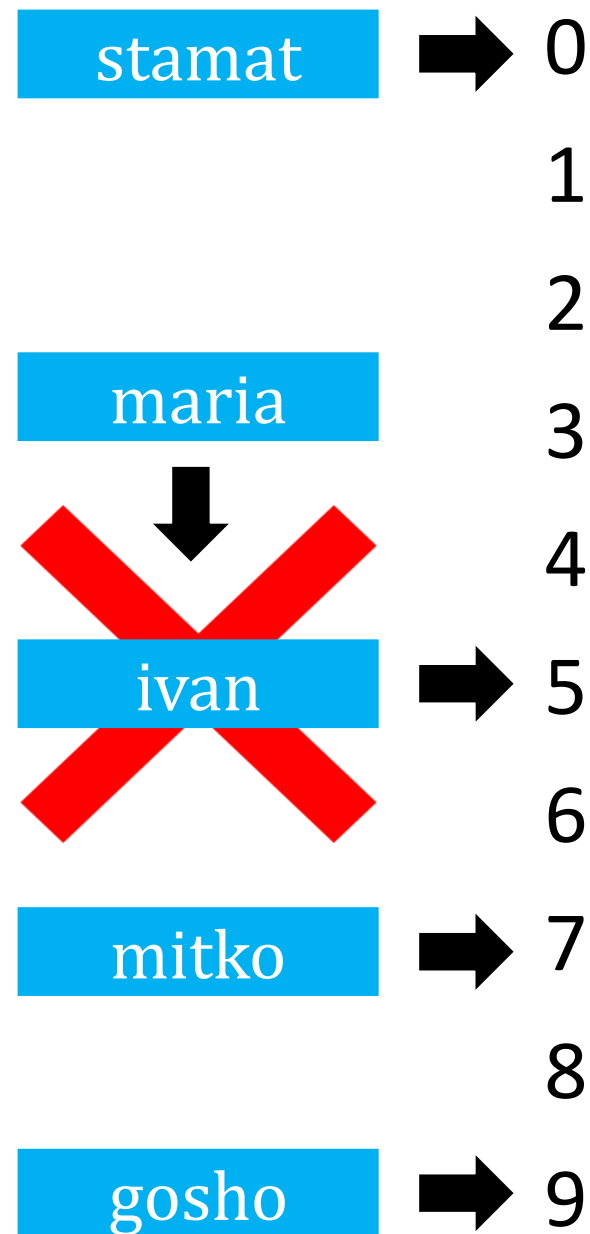
mitko

→ 7

8

gosho

→ 9



Колизии в хеш таблици [1/2]

- Колизия настъпва, когато хеш функцията генерира един и същ хеш за различни ключове

$$h(k_1) = h(k_2) \text{ for } k_1 \neq k_2$$

- При нисък брой колизии, бързодействието на хеш таблиците не се афектира

Колизии в хеш таблици [2/2]

- Стратегии за разрешаване на колизии
 - Свързване на елементите в колизия
 - Използване на други клетки от таблицата
 - Cuckoo хеширане
 - други...

Свързване на елементи [1/9]

Хеш функция

maria

0

1

2

3

4

5

6

7

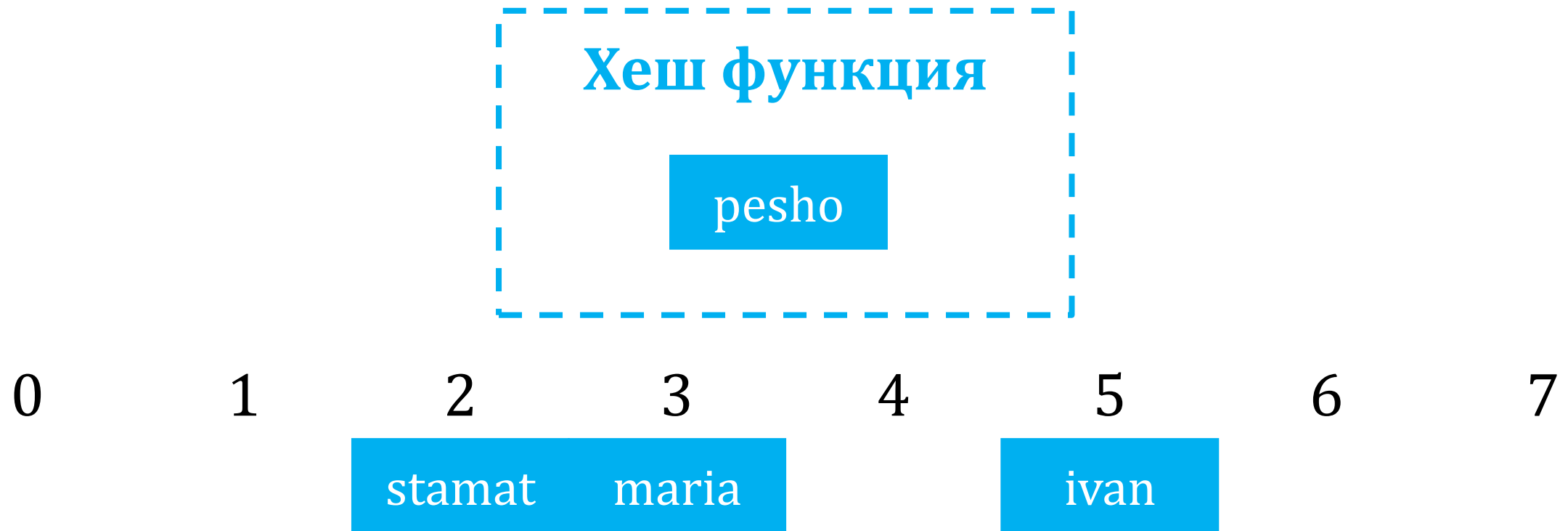
Свързване на елементи [2/9]



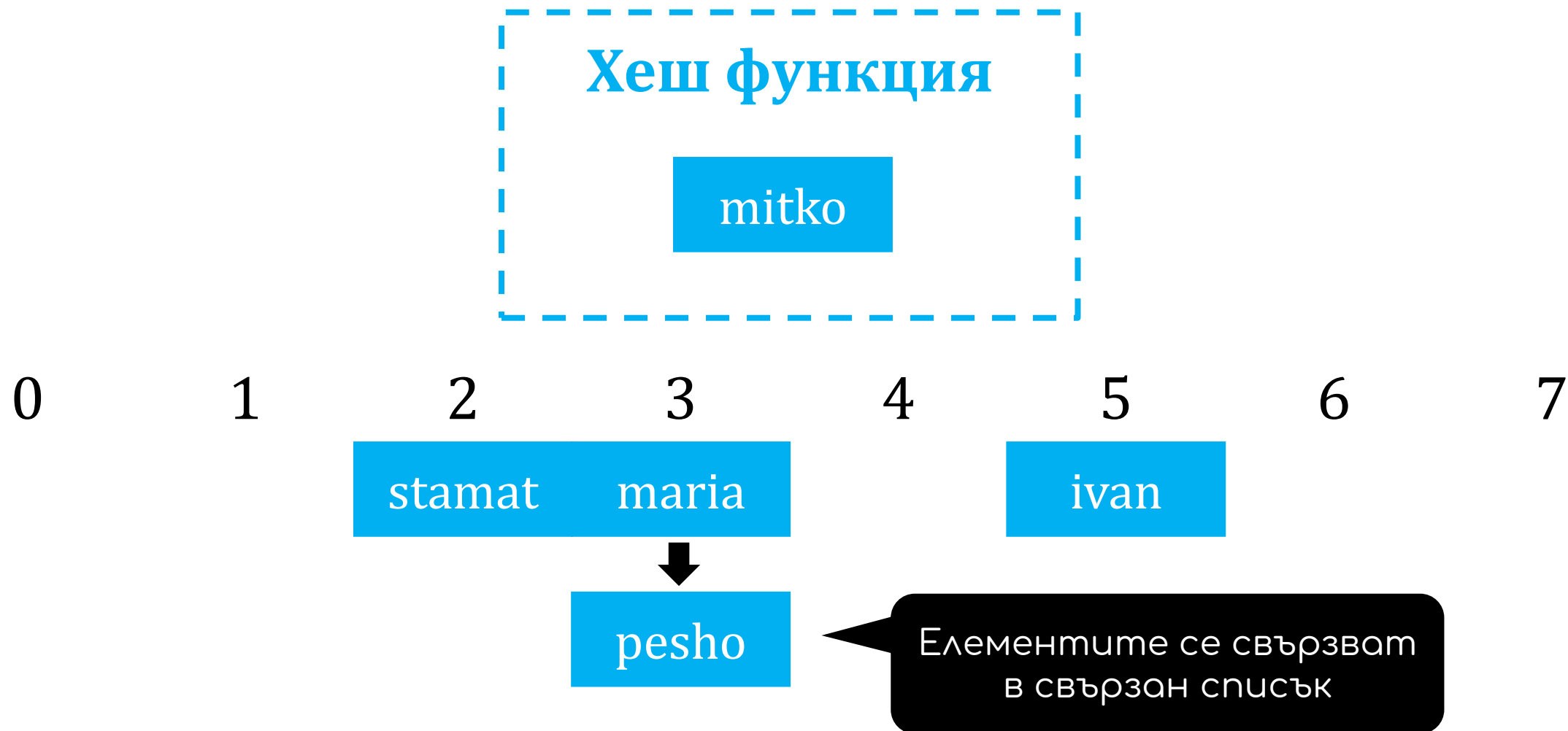
Свързване на елементи [3/9]



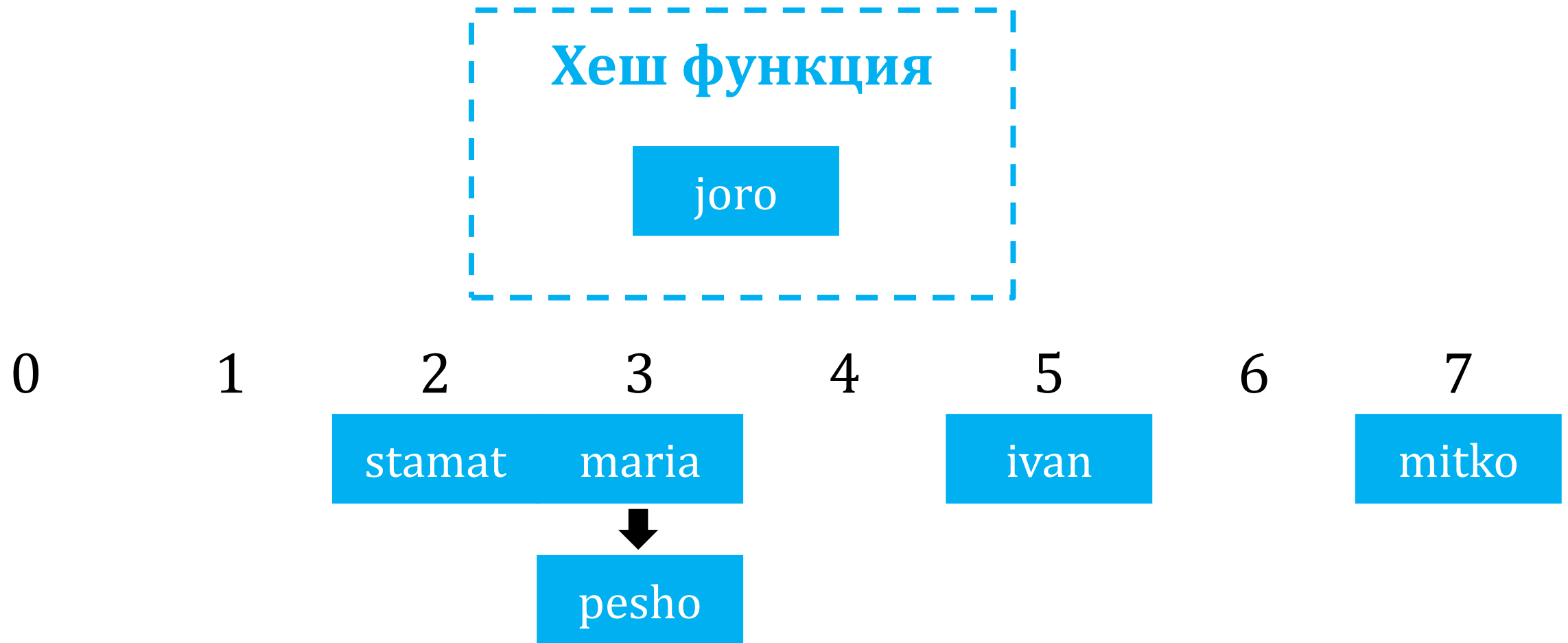
Свързване на елементи [4/9]



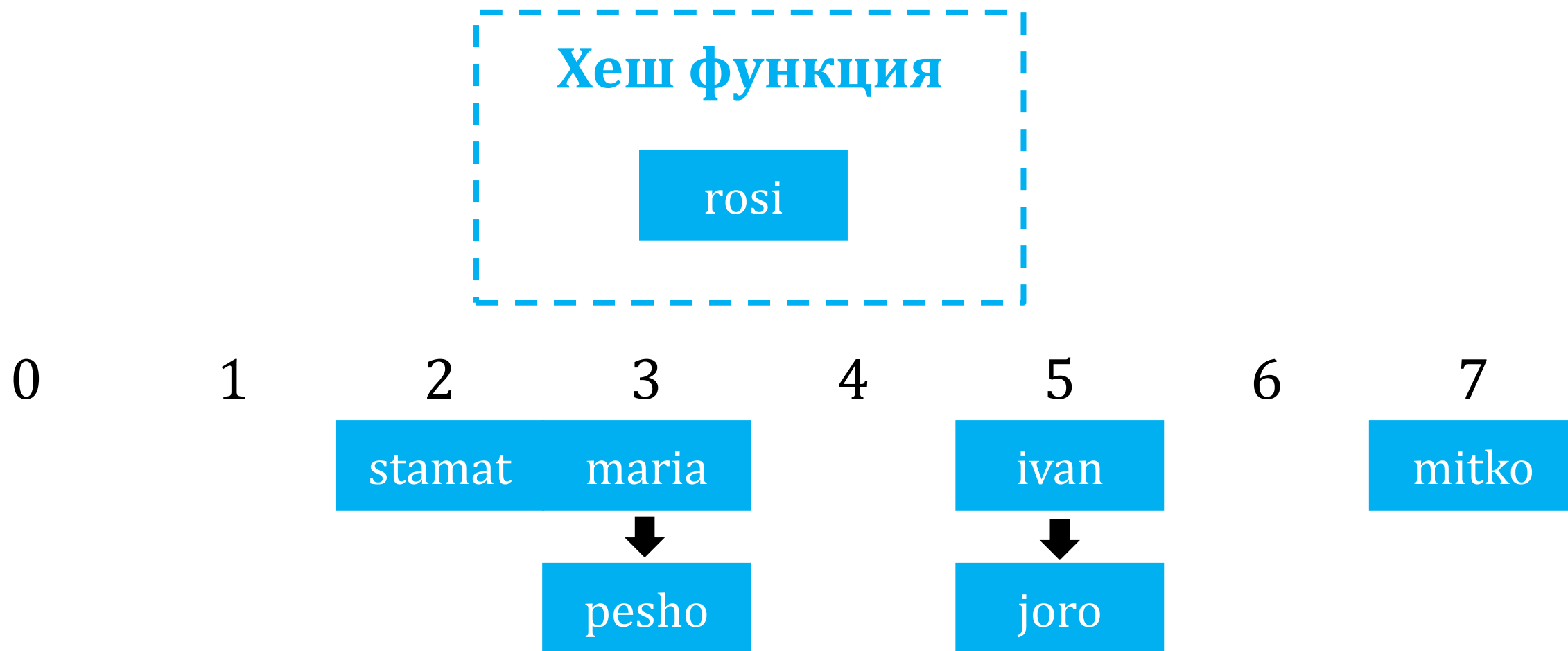
Свързване на елементи [5/9]



Свързване на елементи [6/9]



Свързване на елементи [7/9]



Свързване на елементи [8/9]

Хеш функция

alex

0

rosi

1

2

stamat

3

maria

4

5

ivan

6

7

mitko



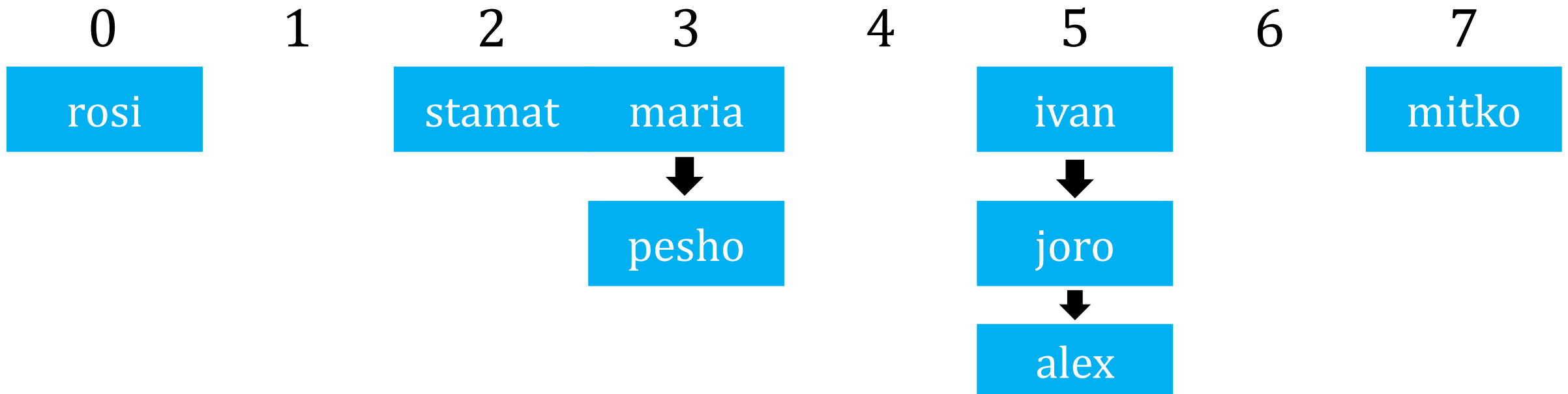
pesho



joro

Свързване на елементи [9/9]

Хеш функция



Отворена адресация [1/2]

Отворена адресация е стратегия за разрешаване на колизии, при която конфликтните елементи се съхраняват в друга клетка на хеш таблицата

- **Линейно пробване** - взима се следващия празен слот след позицията на колизията

$$h(\text{key}, i) = h(\text{key}) + i$$

където i е поредния брой на опита: 0, 1, 2, ...

$h(\text{key}) + 1$, $h(\text{key}) + 2$, $h(\text{key}) + 3$, и т.н.

Отворена адресация [2/2]

- Квадратично пробване - $i^{\text{та}}$ следваща позиция се определя от квадратна функция (c_1 и c_2 са константи и от тях зависи кои позиции ще бъдат пробвани)

$$h(\text{key}, i) = h(\text{key}) + c_1 * i + c_2 * i^2$$

$$h(\text{key}) + 1^2, h(\text{key}) + 2^2, h(\text{key}) + 3^2, \text{ etc.}$$

- Двойно хеширане - използване на втора хеш функция за колизиите

$$h(\text{key}, i) = h_1(\text{key}) + i * h_2(\text{key})$$

Линейно пробване [1/18]

Хеш функция

maria

0

1

2

3

4

5

6

7

Линейно пробване [2/18]

Хеш функция

ivan

0

1

2

3

4

5

6

7

maria

Линейно пробване [3/18]

Хеш функция

stanat

0

1

2

3

4

5

6

7

maria

ivan

Линейно пробване [4/18]

Хеш функция

resho

0

1

2

3

4

5

6

7

stanat

maria

ivan

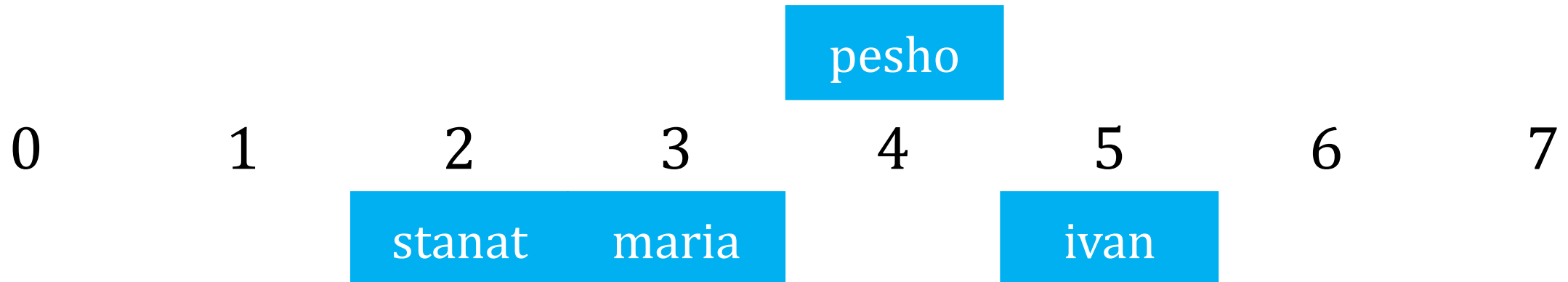
Линейно пробване [5/18]

Хеш функция



Линейно пробване [6/18]

Хеш функция



Линейно пробване [7/18]

Хеш функция

mitko

0

1

2

3

4

5

6

7

stanat

maria

pesho

ivan

Линейно пробване [8/18]

Хеш функция

joro

0

1

2

3

4

5

6

7

stanat

maria

pesho

ivan

mitko

Линейно пробване [9/18]

Хеш функция

0

1

2

3

4

5

6

7

stanat

maria

pesho

ivan

joro

mitko

Линейно пробване [10/18]

Хеш функция

0

1

2

3

4

5

6

7

stanat

maria

pesho

ivan

joro

mitko

Линейно пробване [11/18]

Хеш функция

rosi

0

1

2

3

4

5

6

7

stanat

maria

pesho

ivan

joro

mitko

Линейно пробване [12/18]



0	1	2	3	4	5	6	7
rosi		stanat	maria	pesho	ivan	joro	mitko

Линейно пробване [13/18]

Хеш функция

0	1	2	3	4	alex	6	7
rosi		stanat	maria	pesho	ivan	joro	mitko

Линейно пробване [14/18]

Хеш функция

0	1	2	3	4	5	alex	7
rosi		stanat	maria	pesho	ivan	joro	mitko

Линейно пробване [15/18]

Хеш функция

								alex
0	1	2	3	4	5	6	7	
rosi		stanat	maria	pesho	ivan	joro	mitko	

Линейно пробване [16/18]

Хеш функция

alex

0

rosi

1

2

3

4

5

6

7

stanat

maria

pesho

ivan

joro

mitko

Линейно пробване [17/18]

Хеш функция

alex

0

1

2

3

4

5

6

7

rosi

stanat

maria

pesho

ivan

joro

mitko

Линейно пробване [18/18]

Хеш функция

0

1

2

3

4

5

6

7

rosi

alex

stanat

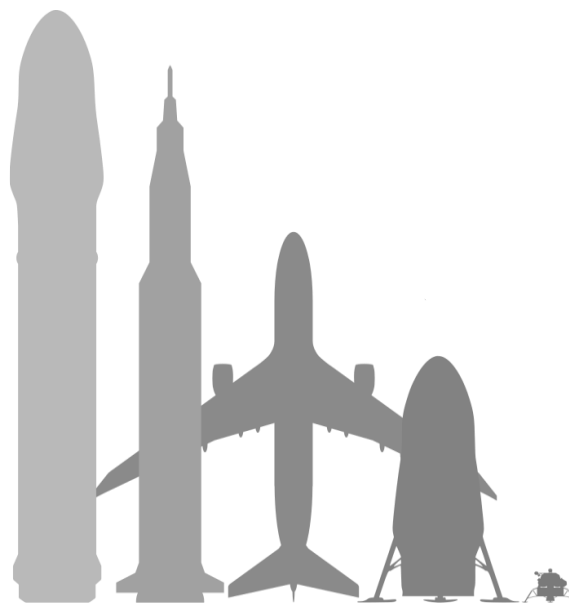
maria

pesho

ivan

joro

mitko



Сравняване на ключове
при работа със собствени класове

Упражнение

Сравняване на ключове

- `Dictionary<TKey,TValue>` използва:
 - `Object.Equals()` – за сравнение на ключове
 - `Object.GetHashCode()` – за изчисляване на ключове
- `SortedDictionary<TKey,TValue>` използва
 - `IComparable<T>` за подредба на ключове

Реализация на Equals() и GetHashCode()

```
public class Point
{
    public int X { get; set; }
    public int Y { get; set; }

    public override bool Equals(Object obj)
    {
        if (!(obj is Point) || (obj == null)) return false;
        Point p = (Point)obj;
        return (X == p.X) && (Y == p.Y);
    }

    public override int GetHashCode()
    {
        return (X << 16 | X >> 16) ^ Y;
    }
}
```

Реализация на IComparable<T>

```
public class Point : IComparable<Point>
{
    public int X { get; set; }
    public int Y { get; set; }

    public int CompareTo(Point otherPoint)
    {
        if (X != otherPoint.X)
        {
            return this.X.CompareTo(otherPoint.X);
        }
        else
        {
            return this.Y.CompareTo(otherPoint.Y);
        }
    }
}
```

key	value
John Smith	+1-555-8976
Sam Doe	+1-555-5030
Sam Smith	+1-555-4542
John Doe	+1-555-3527

Речници

Дефиниция и функционалност

Речник: Dictionary (MAP)

- Абстрактния тип данни **речник** асоциира стойности с уникални ключове
 - Тази структура е позната като **карта** или **асоциативен масив**
 - Съдържа набор от наредени двойки от тип **{key, value}**
- Имплементации
 - Хеш таблици, балансиран дървета, списъци, масиви и др.

Dictionary<TKey, TValue> [1/2]

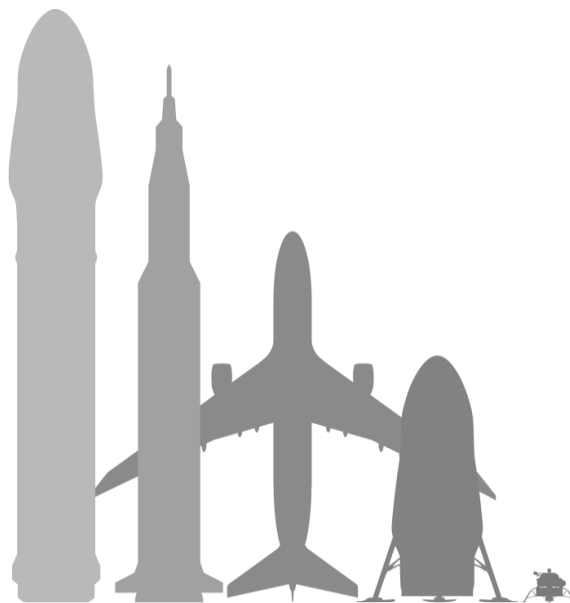
- Основна функционалност:
 - `Add(key, value)` – добавя елемент
 - `Remove(key)` – премахва елемент
 - `this[key] = value` – добавя или подменя елемент
 - `this[key]` – извлича елемент
 - `Keys` – връща всички ключове (по ред на добавяне)
 - `Values` – връща всички стойности (по ред на добавяне)

Dictionary<TKey, TValue> [2/2]

- Основна функционалност:
 - **ContainsKey(key)** – проверява дали ключа е в речника
 - **ContainsValue(value)** – проверява дали стойността е в речника
 - **TryGetValue(key, out value)**

Ако намери стойността я записва във параметъра **value** и връща **true**

Иначе връща **false**



Упражнение: Реализация на хеш таблица
стратегия за колизии - свързване на елементи

Обобщение

- Хеширащи функции
- Хеш таблици
- Управление на колизии в хеш таблици
- Упражнения: хеш таблици



Национална програма
"Обучение за ИТ умения и кариера"
<https://it-kariera.mon.bg>

Министерството на
образованието и науката
<https://www.mon.bg>



Документът е разработен за нуждите на Национална програма "Обучение за ИТ умения и кариера" на Министерството на образованието и науката (МОН) и се разпространява под свободен лиценз CC-BY-NC-SA (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).