EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Programování a algoritmizace

Binary search

Iterační řešení

Strategický projekt UTB ve Zlíně, reg. č. CZ.02.2.69/0.0/0.0/16_015/0002204

Ing. et Ing. Erik Král, Ph.D.
ÚPKS
Fakulta aplikované informatiky

Obsah

Binary search

Iterační implementace

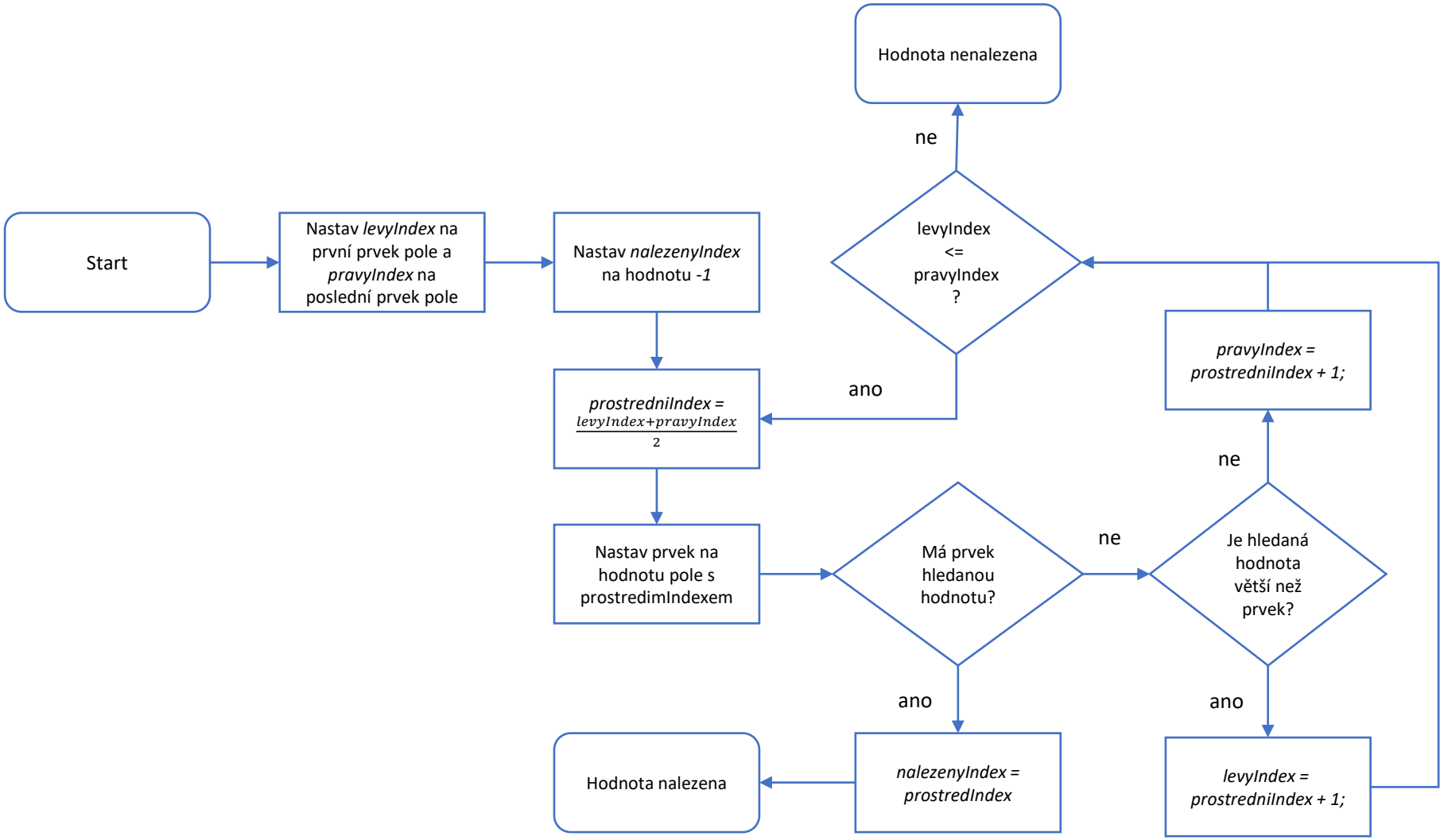
Úvod

- V následujících snímcích probereme algoritmus Binary Search.
- Na těchto příkladech si demonstrujeme práci s jednorozměrným polem s pevnou délkou [1] a iterační implementaci algoritmu.

Binary search

- Algoritmus binárního vyhledávání, který se také nazývá vyhledávání půlením intervalu.
- Vyhledává hodnotu v předem **seřazeném poli** prvků.
- Binární vyhledávání porovnává hledanou hodnotu s prvkem na pozici uprostřed pole. Pokud nejsou stejné, polovina, ve které hledaný prvek nemůže být, je vyloučena a hledání pokračuje na zbývající polovině pole, přičemž se prostřední prvek znovu porovná s hledanou hodnotou a postup opakuje se, dokud není nalezena cílová hodnota nebo už v poli nezbyly žádné prvky k prohledání.
- Algoritmus může být vyřešen s pomocí rekurzivní metody, ale i bez ní. V této prezentaci probereme iterační řešení.

Binary search

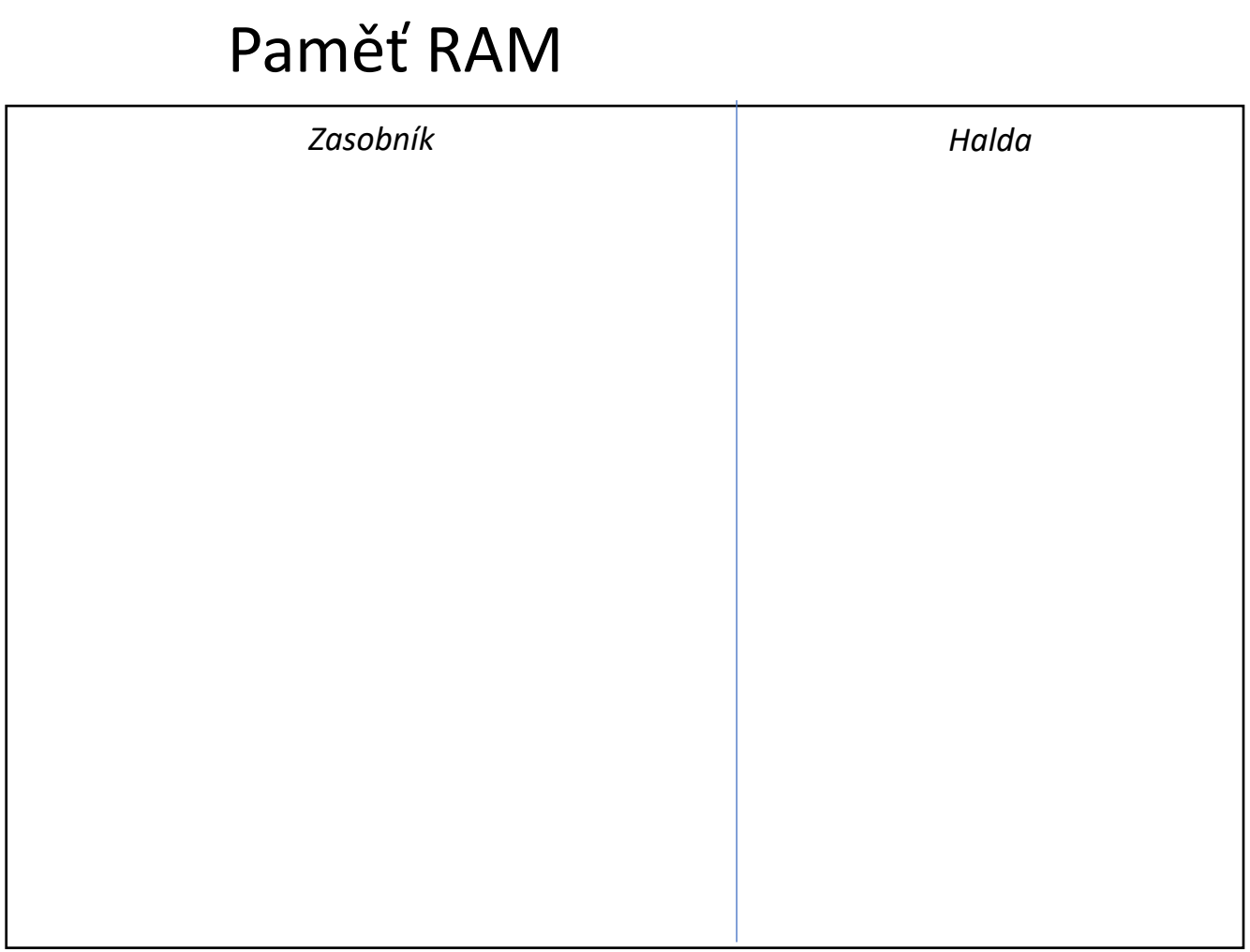


Algoritmus a paměť

- Algoritmus si alokuje paměť pro parametry, lokální proměnné a další hodnoty na zásobníku (Stack) a pro dynamicky alokované objekty alokuje paměť na haldě (Heap).
- V příkladech je **zjednodušeně** demonstrováno využití paměti z hlediska zásobníku a haldy.
- Práce se zásobníkem je ve skutečnosti složitější a v příkladech jsou zobrazeny **pouze proměnné přímo související s algoritmem** a jsou vynechány uložené hodnoty registrů nebo návratové hodnoty. Také pořadí předávaných argumentů a parametrů metody může být jiné.

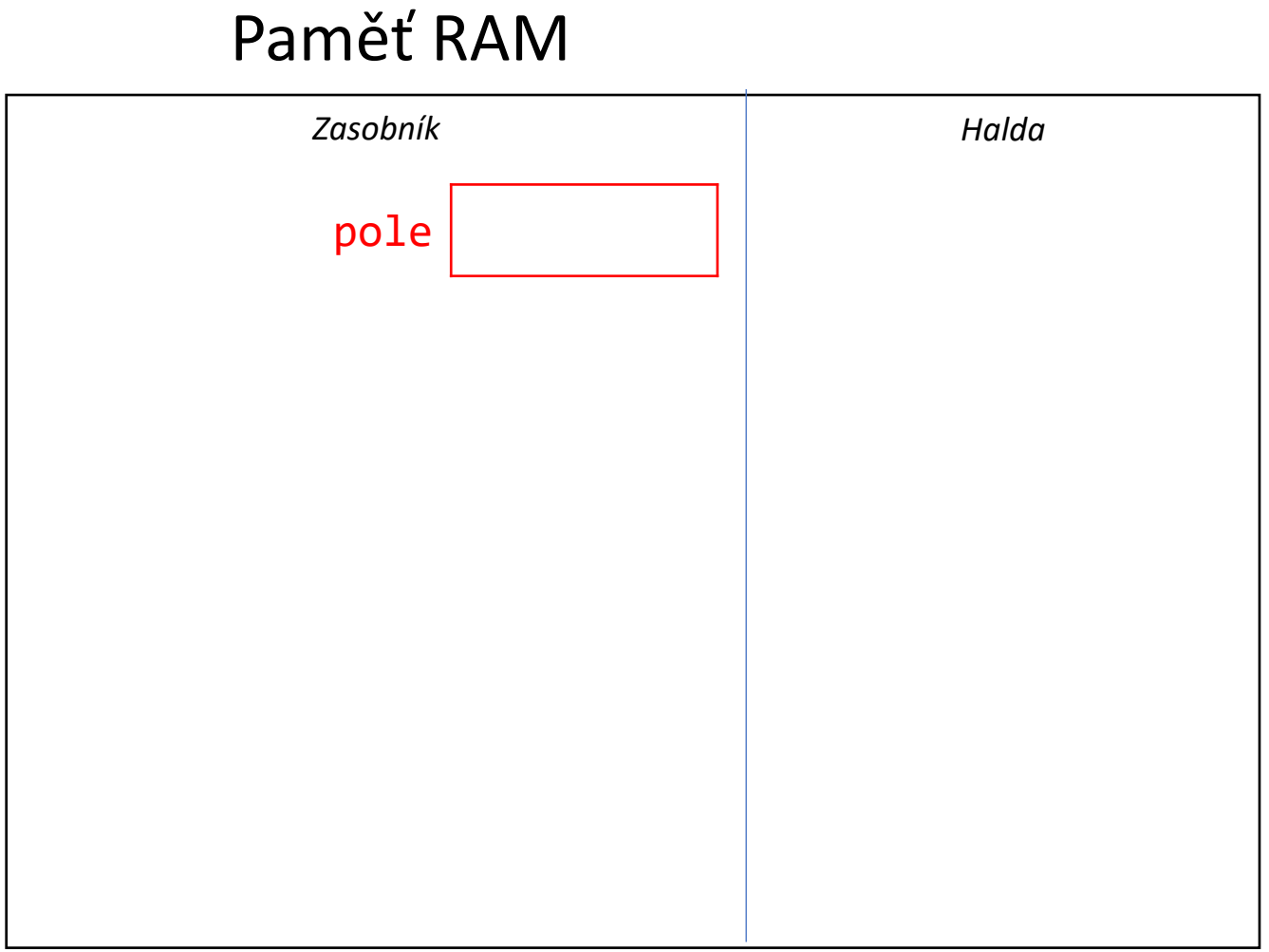
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    int prvek = pole[prostredniIndex];

    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }

} while (levyIndex <= pravyIndex);
```



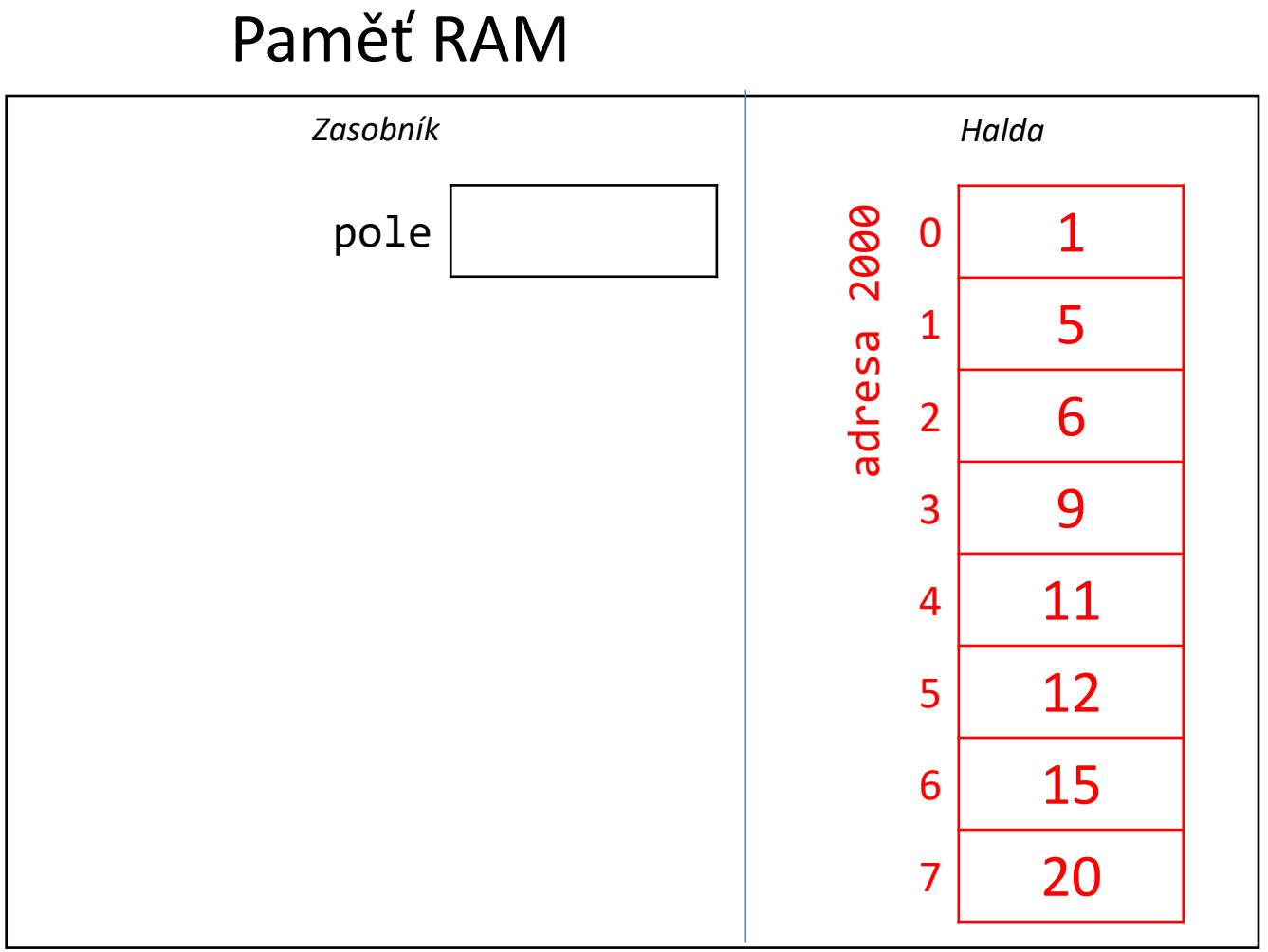
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
```



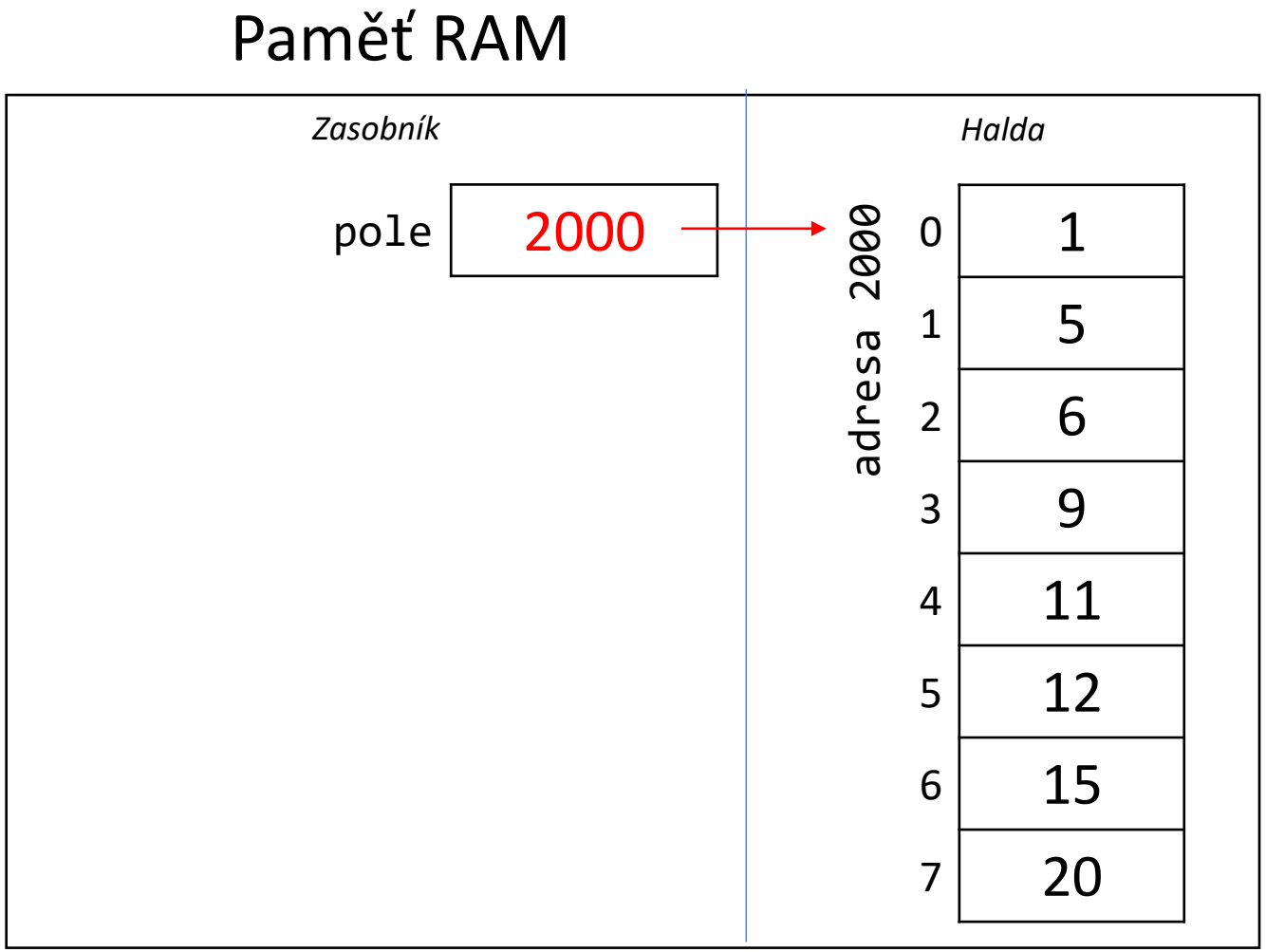
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
```



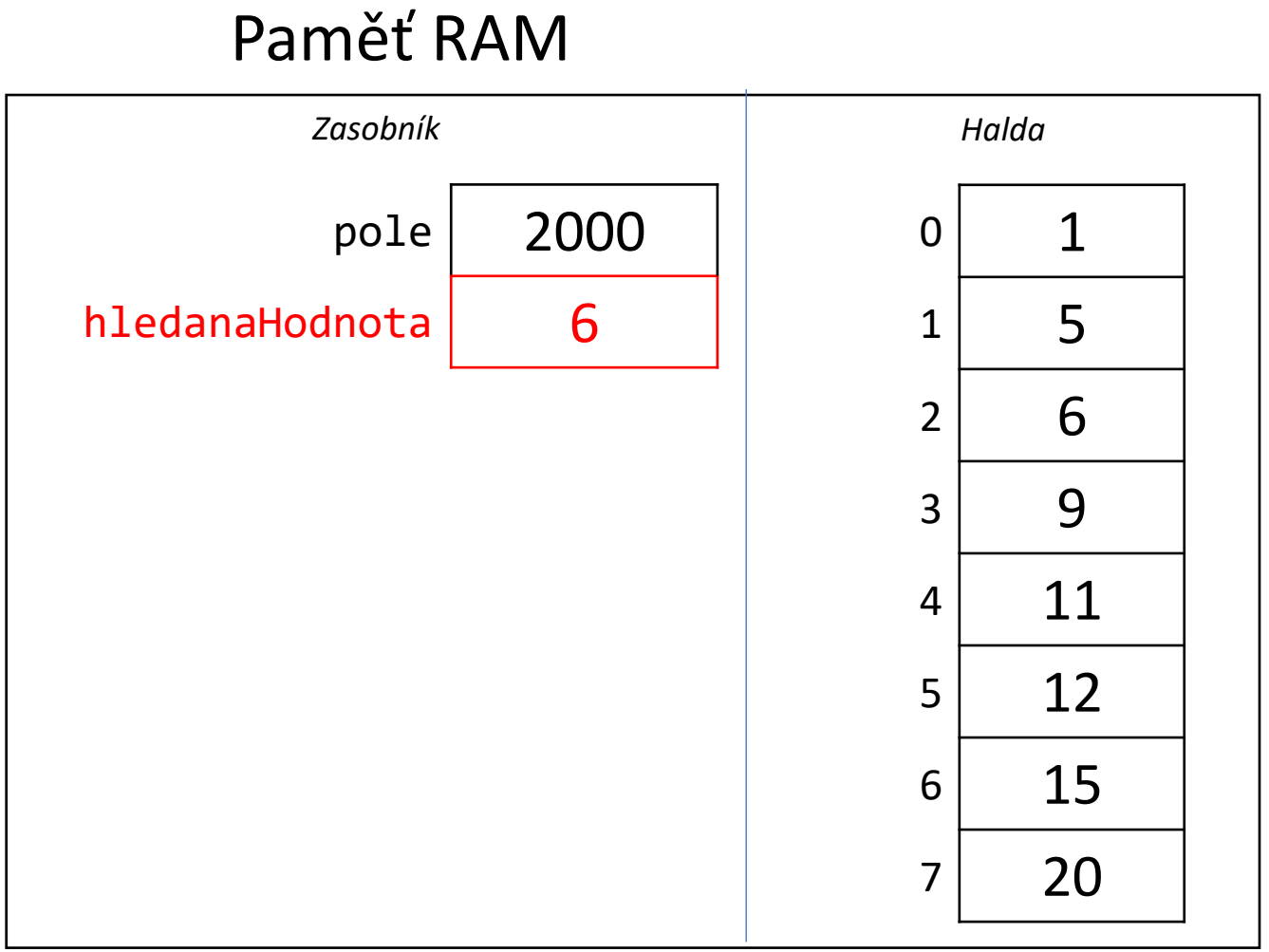
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
```



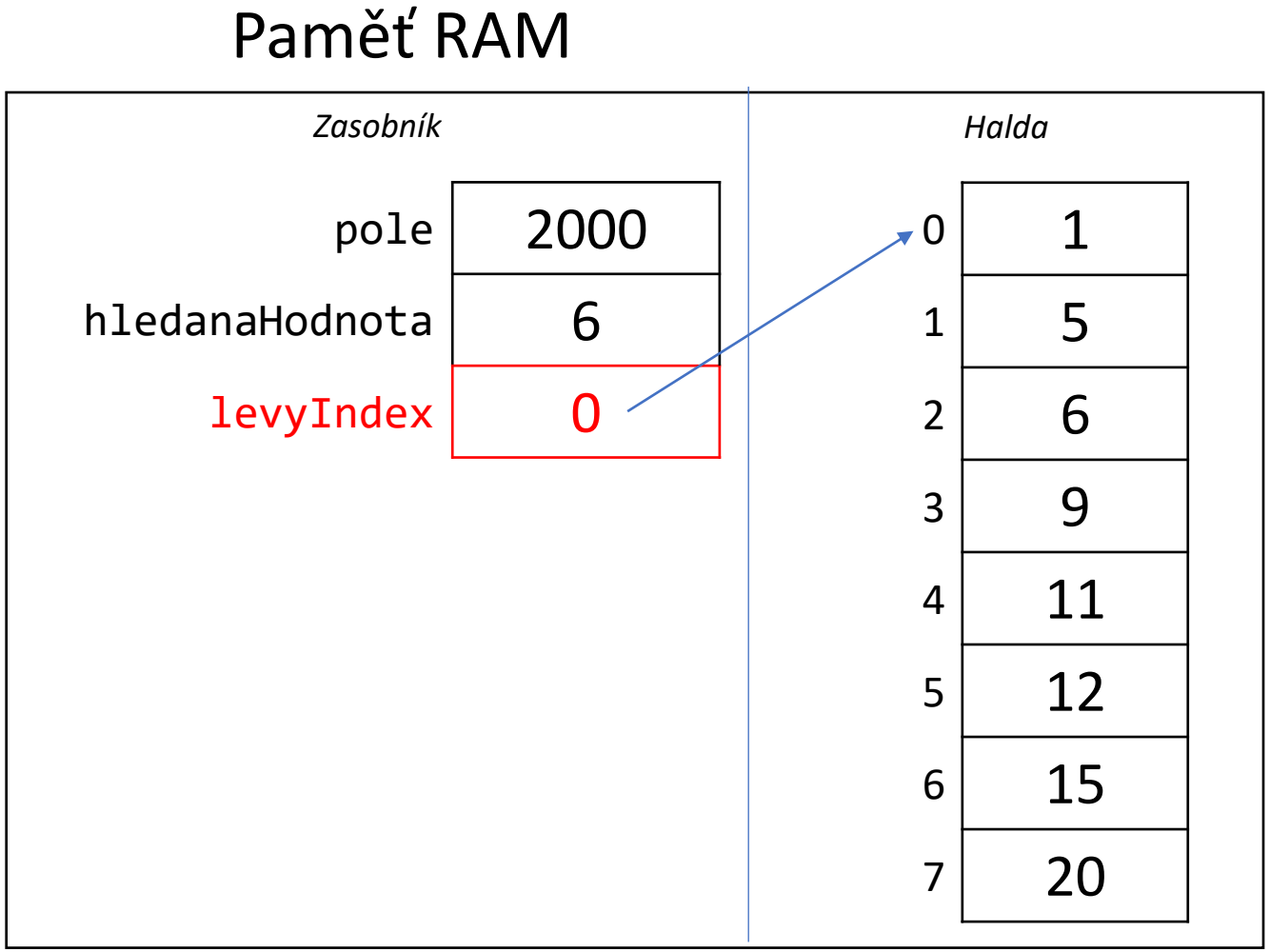
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;
```



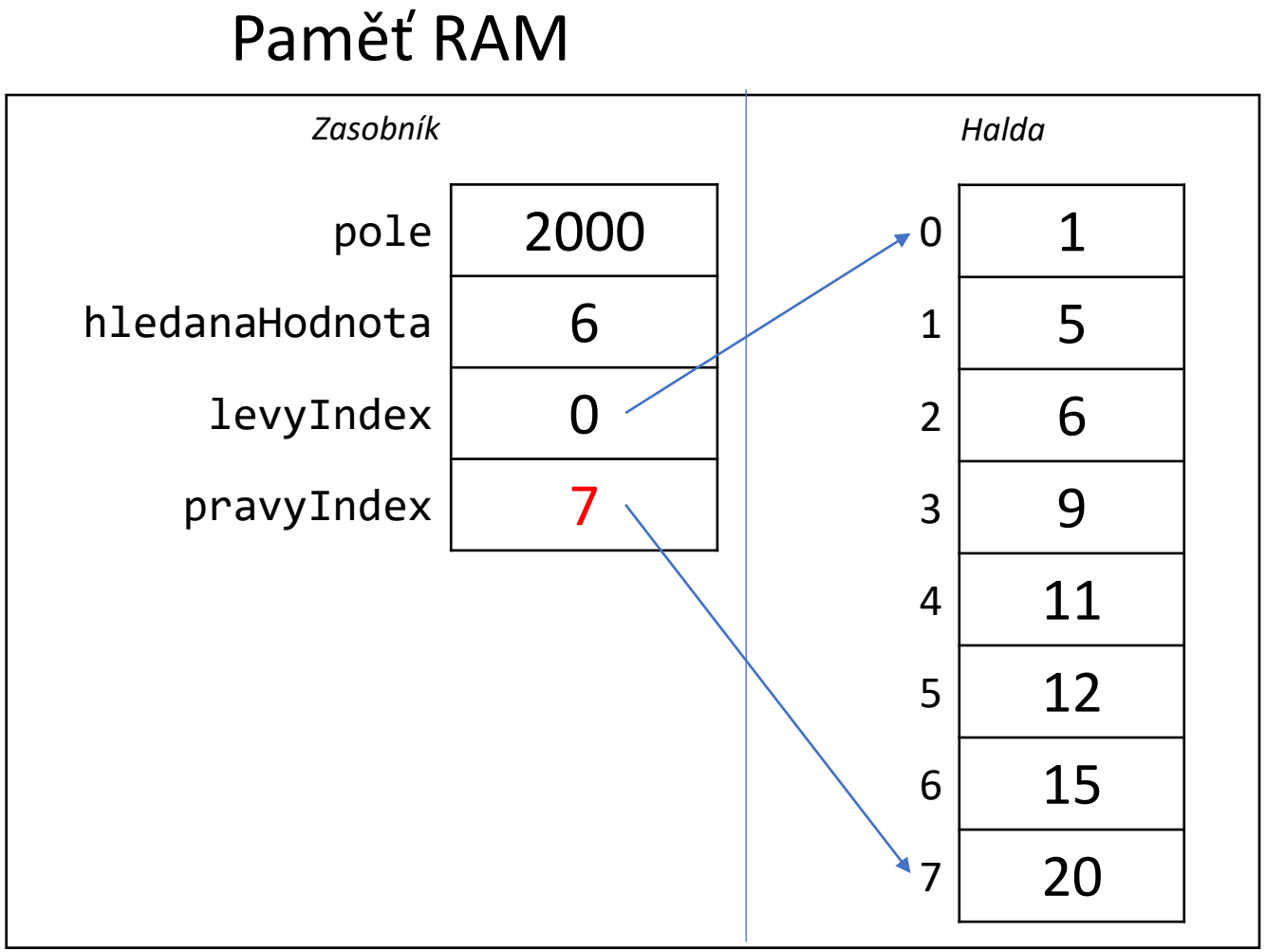
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
```



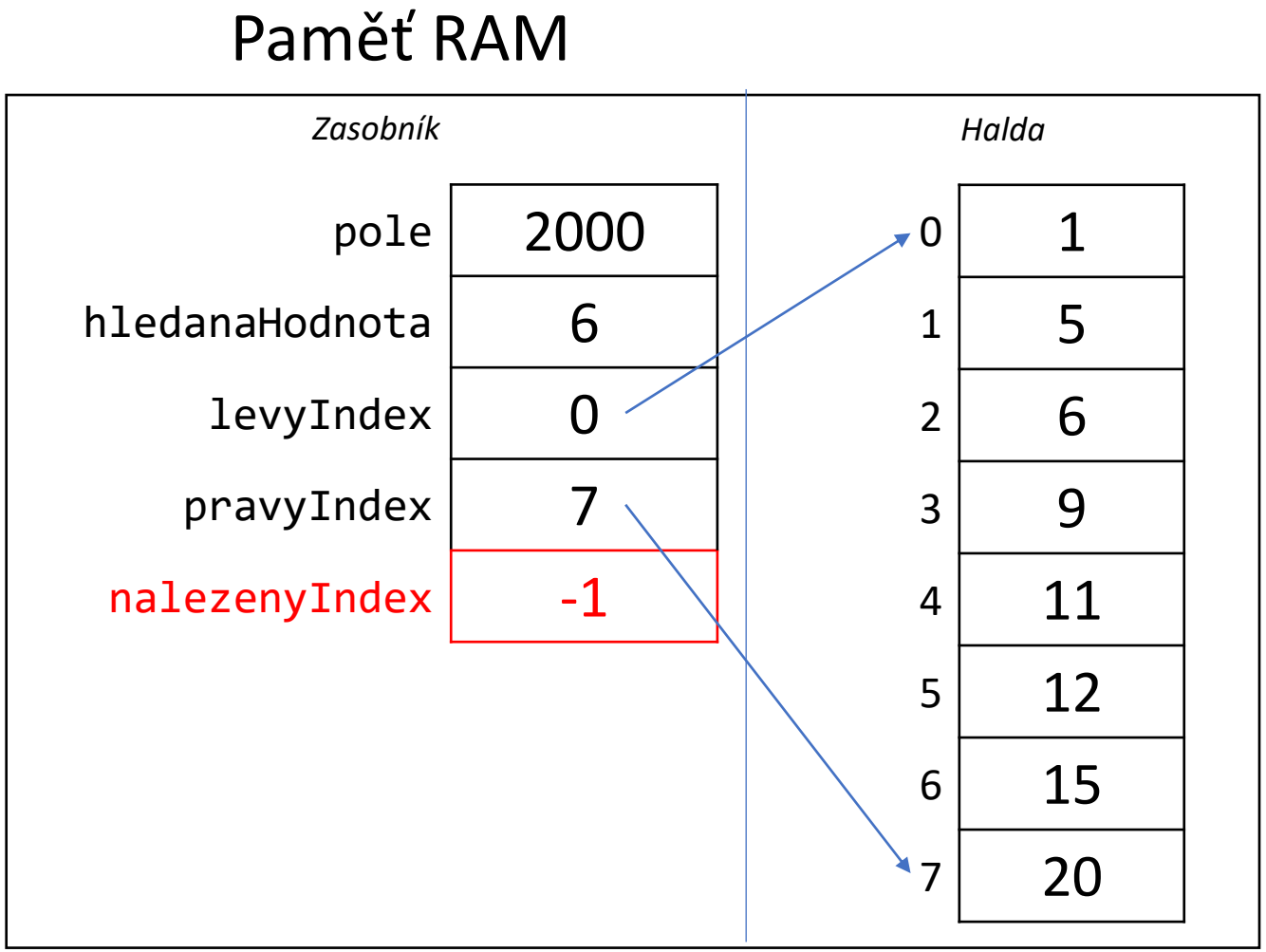
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;
```



Binary search

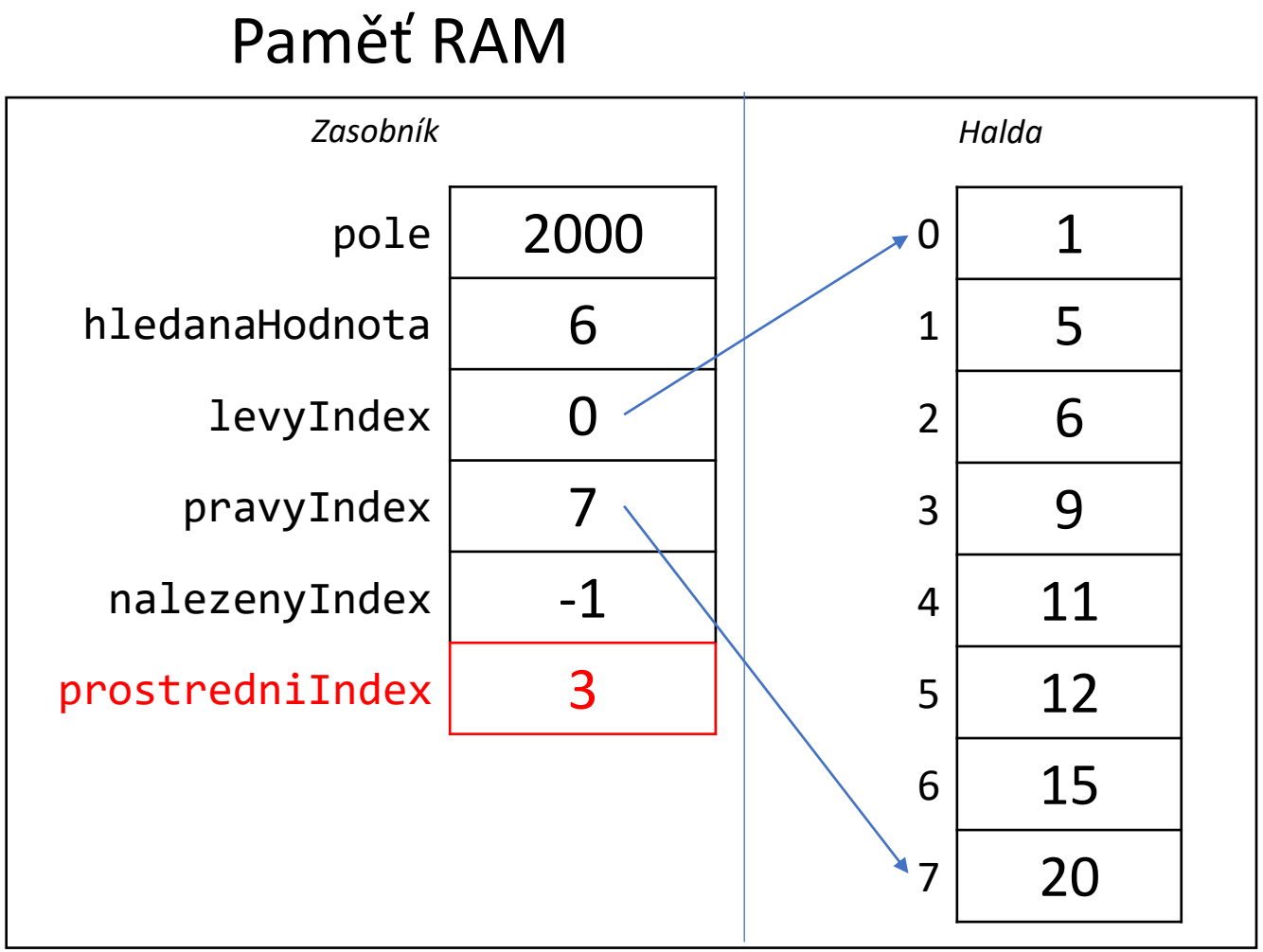
```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;
```



Binary search

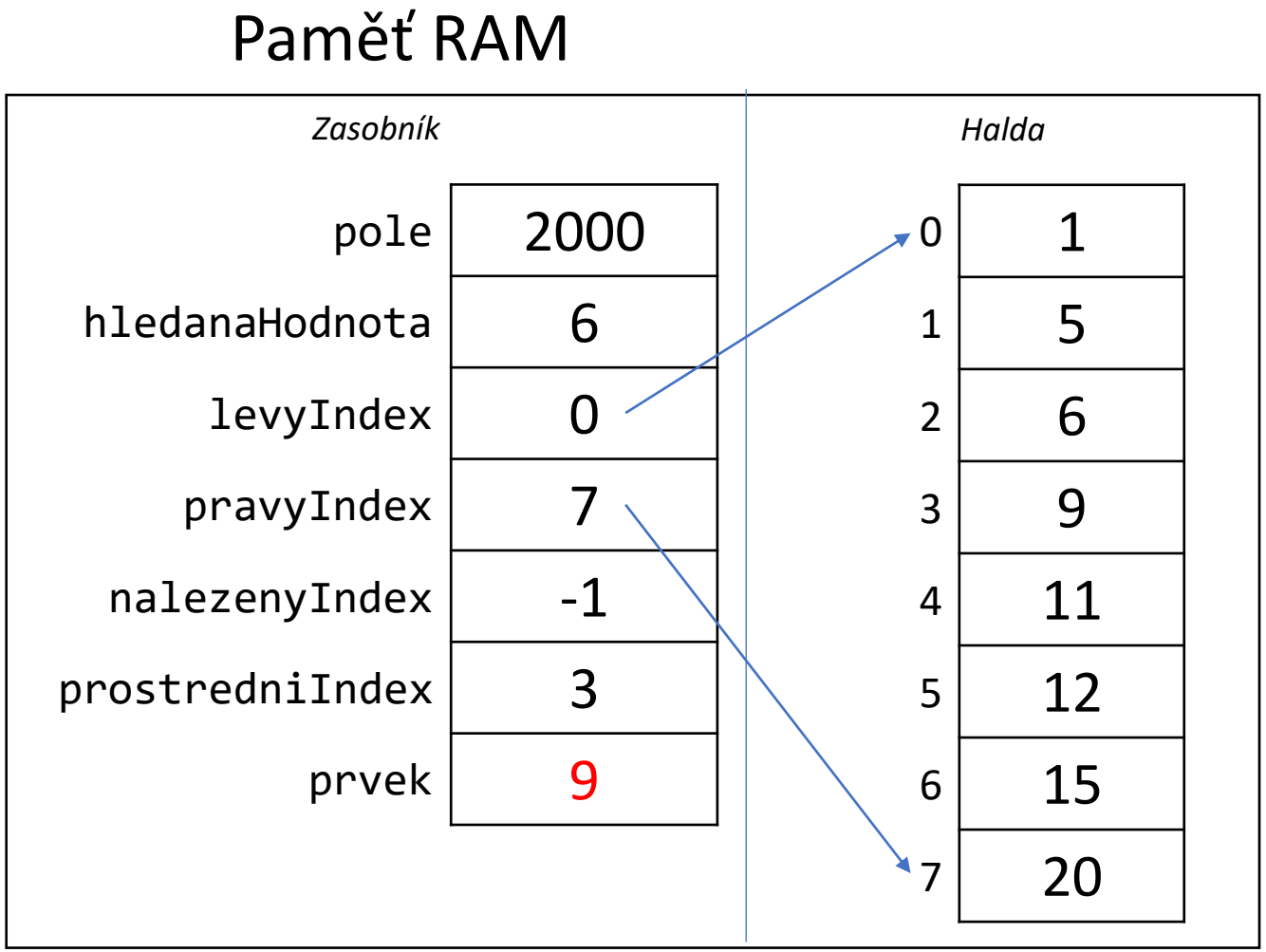
```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

} while (levyIndex <= pravyIndex);
```

3 0 7



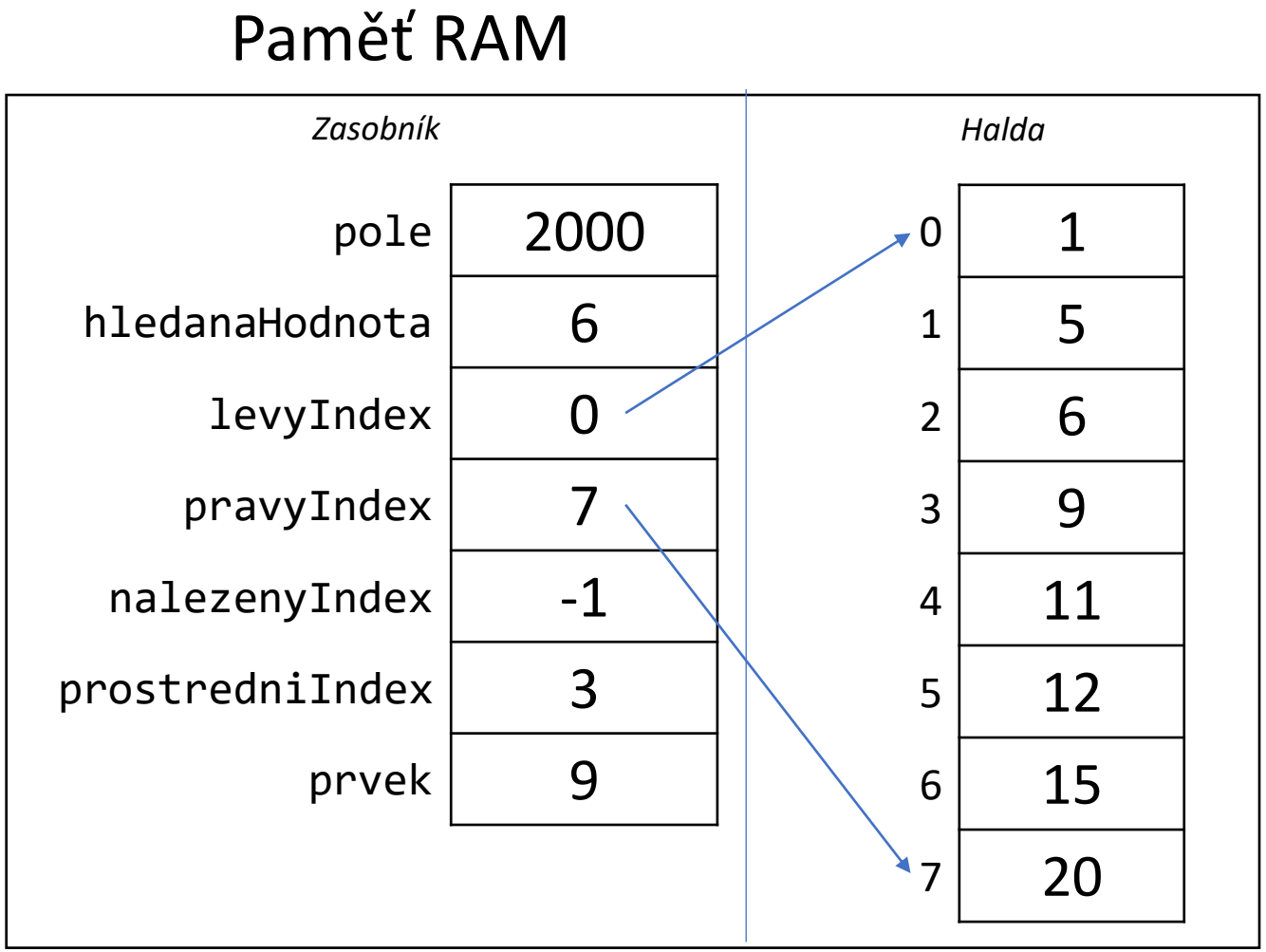
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;
                                                  3
    int prvek = pole[prostredniIndex];









} while (levyIndex <= pravyIndex);
```



Binary search

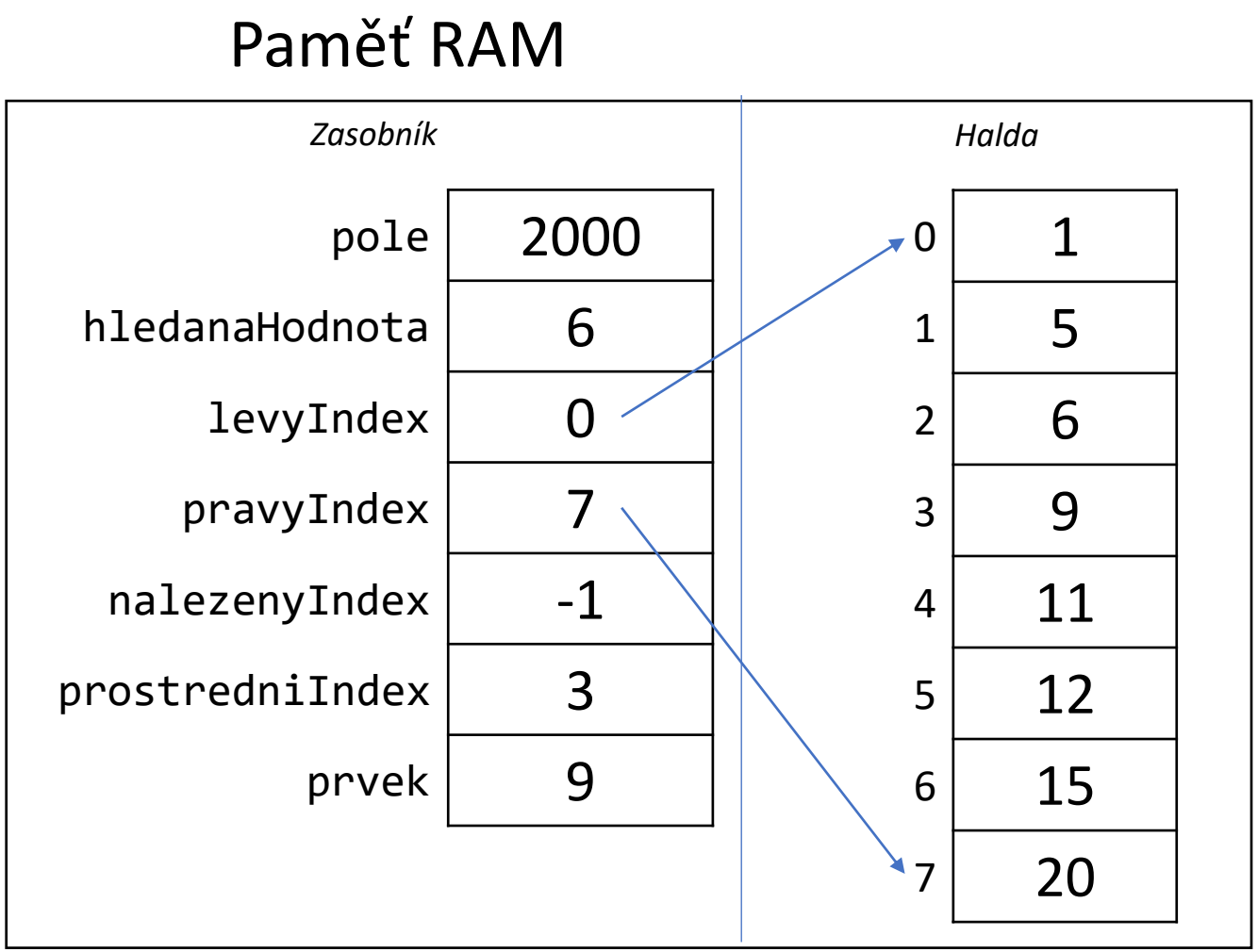
```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    int prvek = pole[prostredniIndex];
            6           9
    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }

} while (levyIndex <= pravyIndex);
```



Binary search

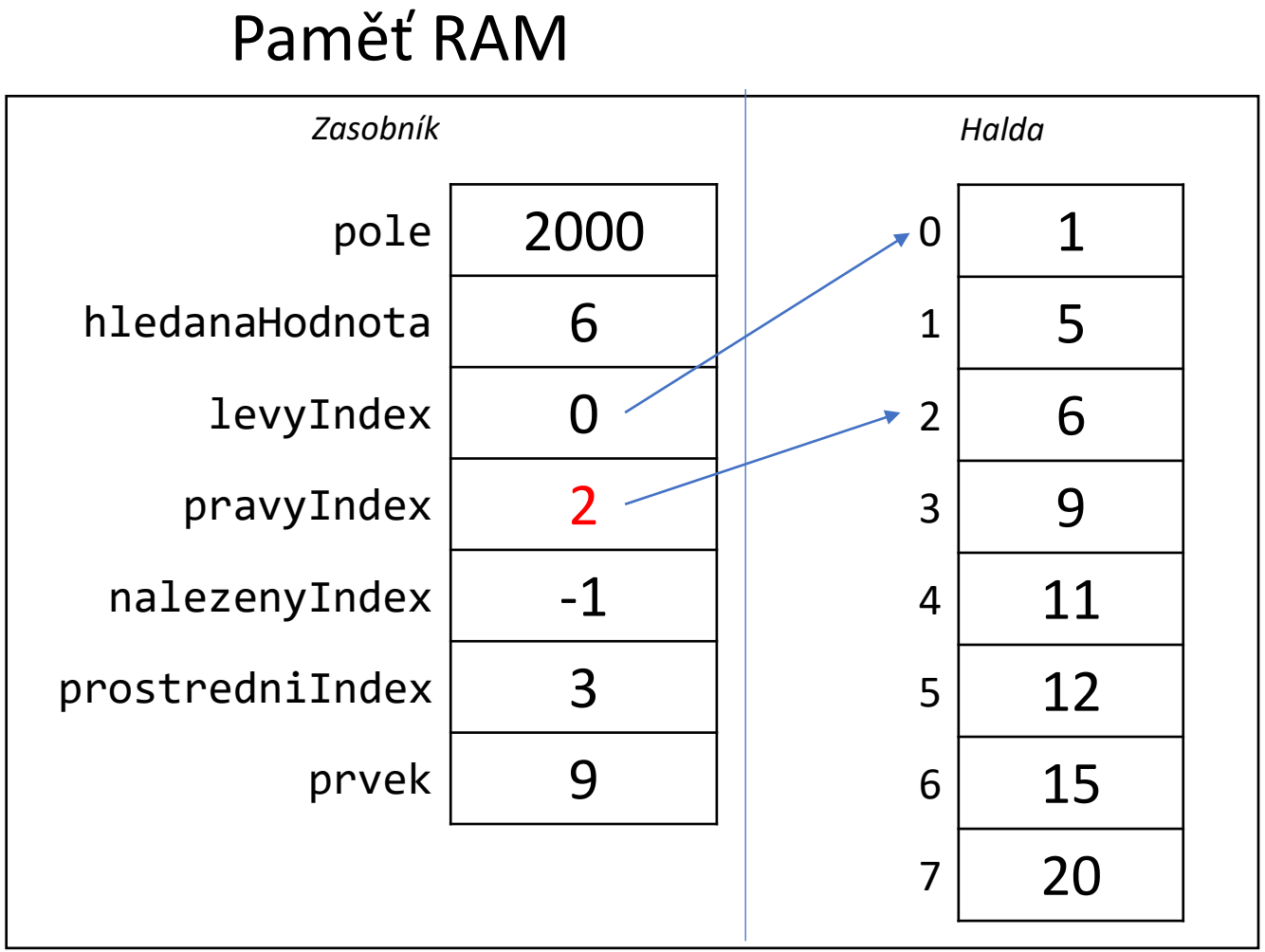
```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    int prvek = pole[prostredniIndex];

    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }

} while (levyIndex <= pravyIndex);
```

6 9



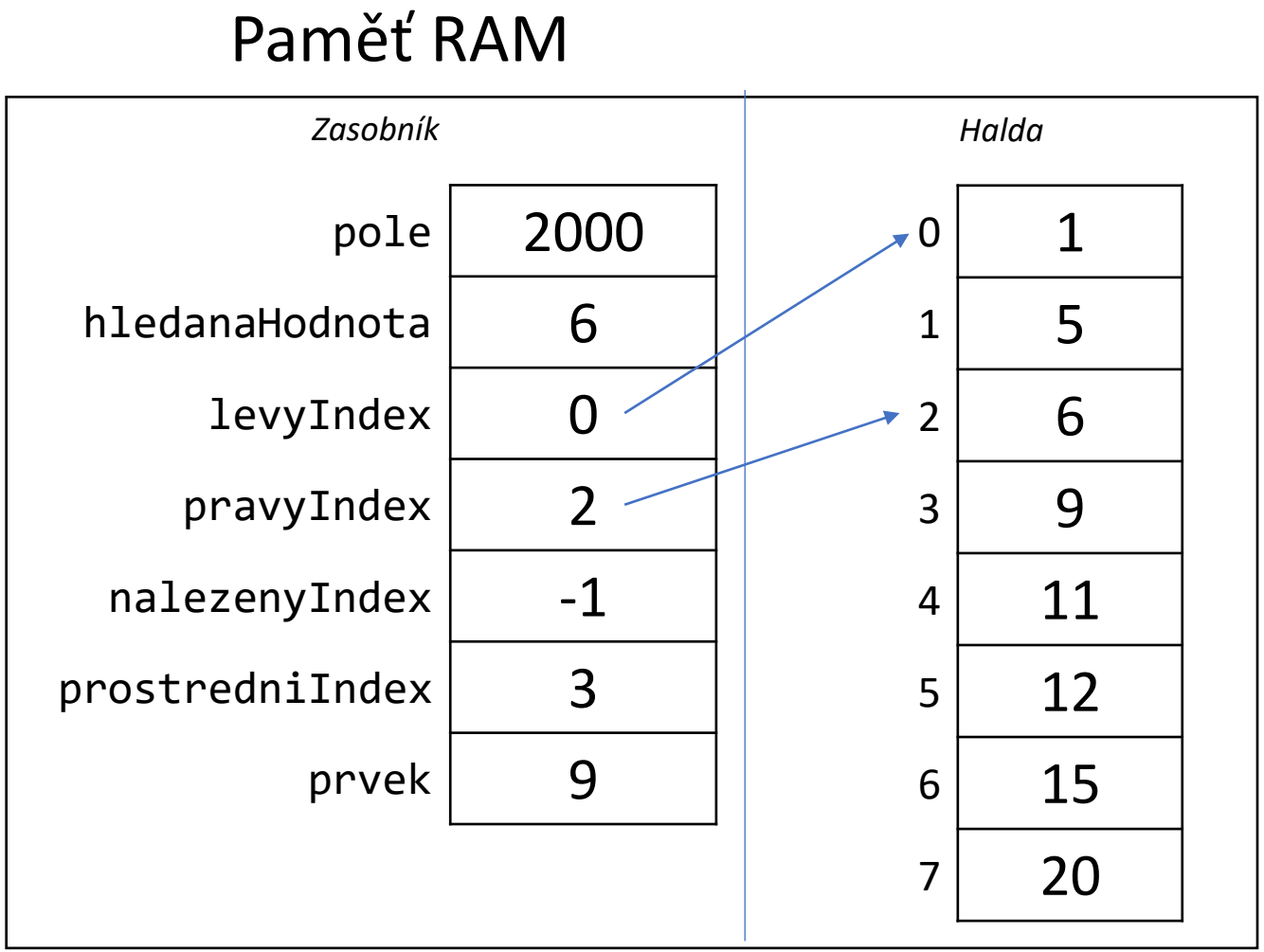
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    int prvek = pole[prostredniIndex];

    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {                          3
        pravyIndex = prostredniIndex - 1;
    }

} while (levyIndex <= pravyIndex);
```



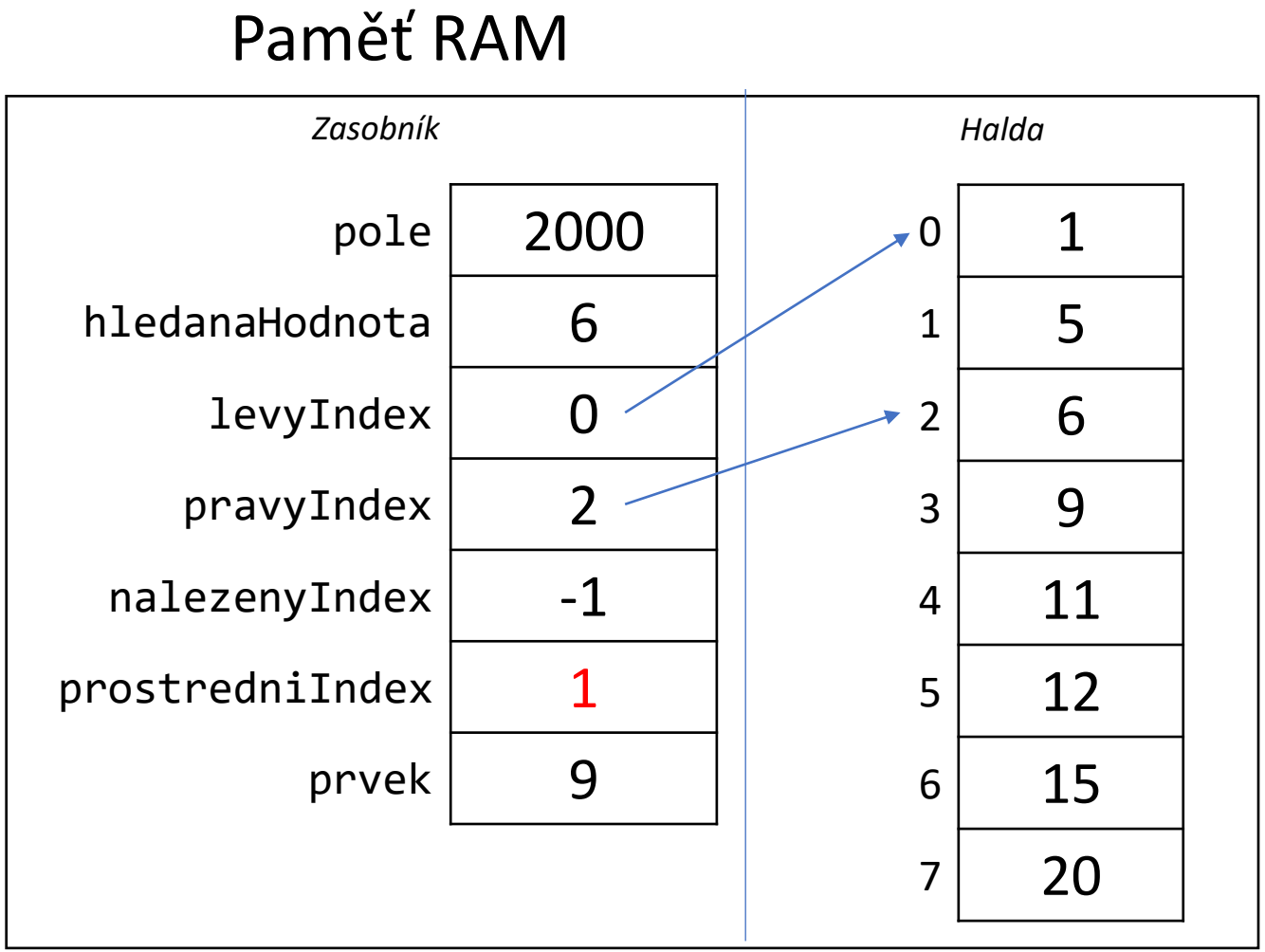
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    int prvek = pole[prostredniIndex];

    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }
            0           2
} while (levyIndex <= pravyIndex);
```



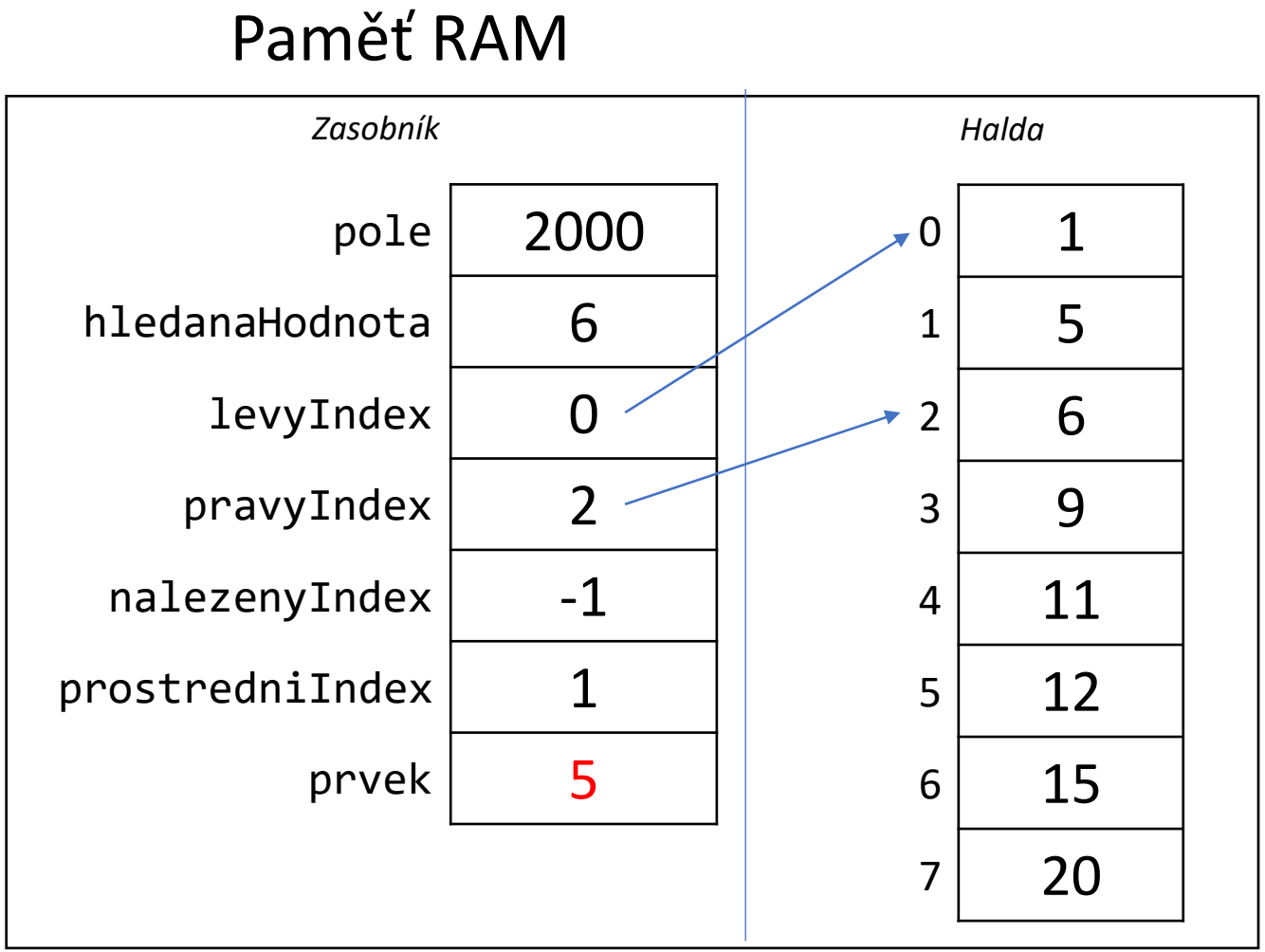
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{                                   0         2
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    int prvek = pole[prostredniIndex];

    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }

} while (levyIndex <= pravyIndex);
```



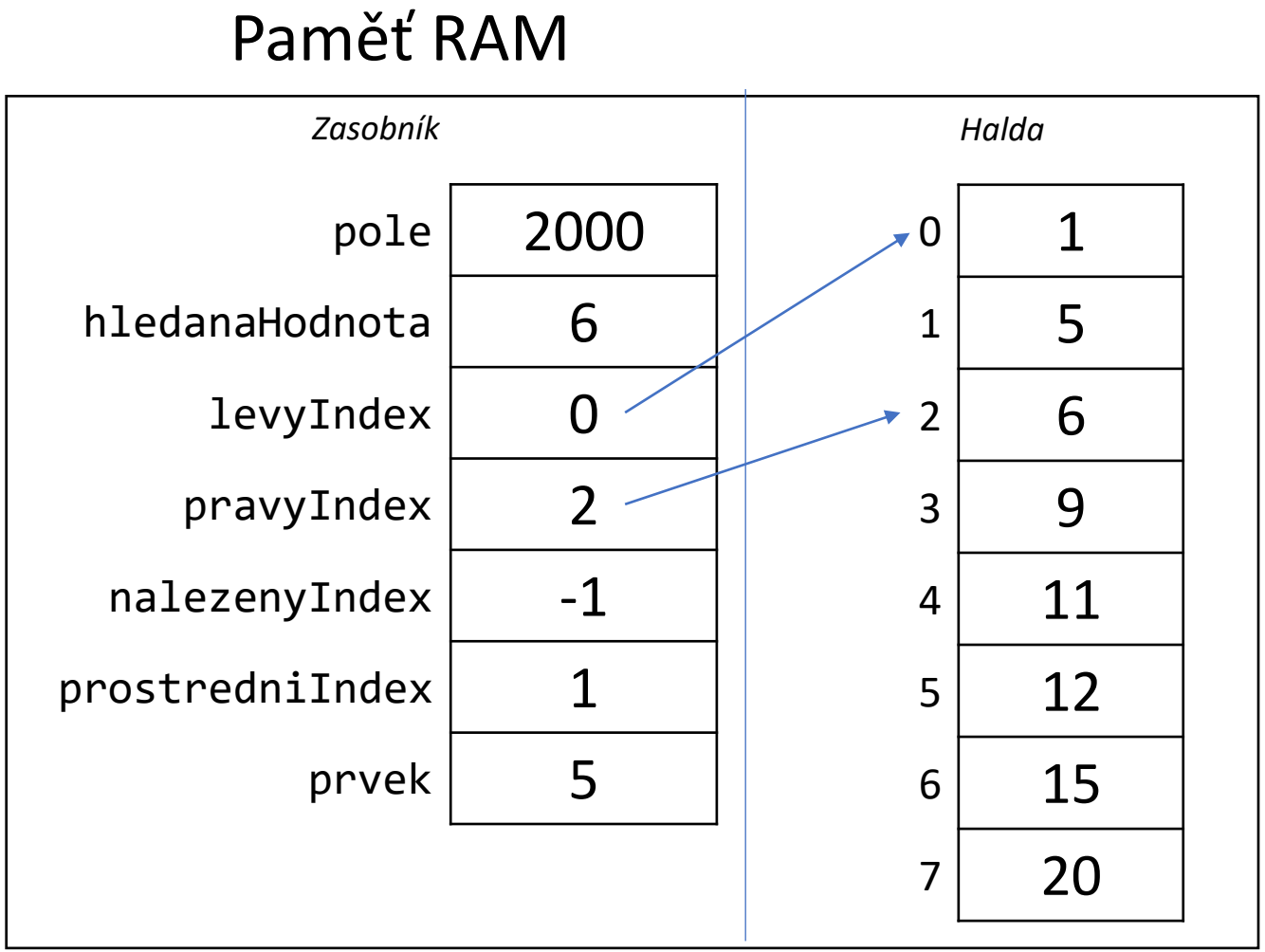
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;
                                                  1
    int prvek = pole[prostredniIndex];

    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }

} while (levyIndex <= pravyIndex);
```



Binary search

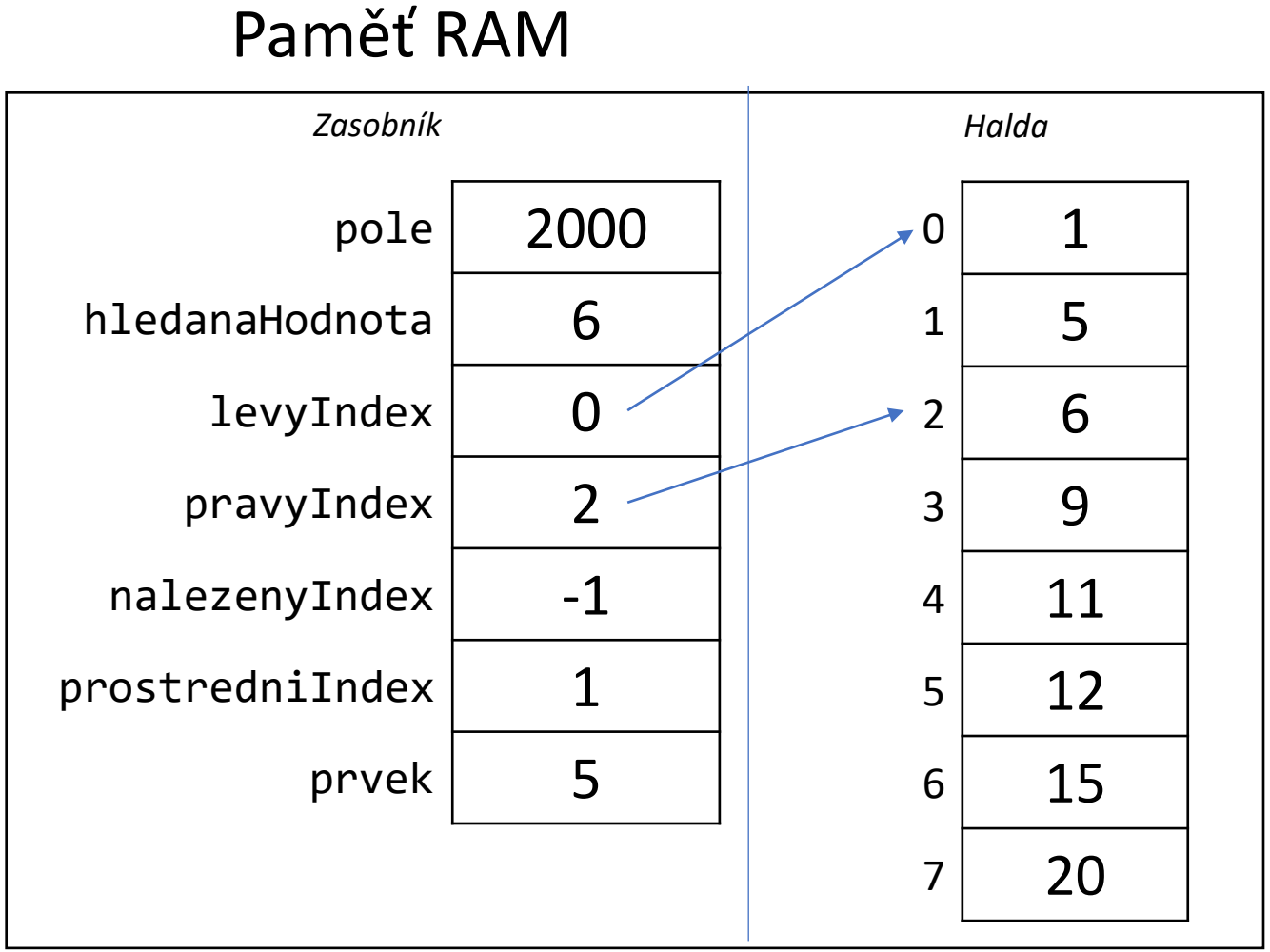
```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    int prvek = pole[prostredniIndex];
            6           5
    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }

} while (levyIndex <= pravyIndex);
```



Binary search

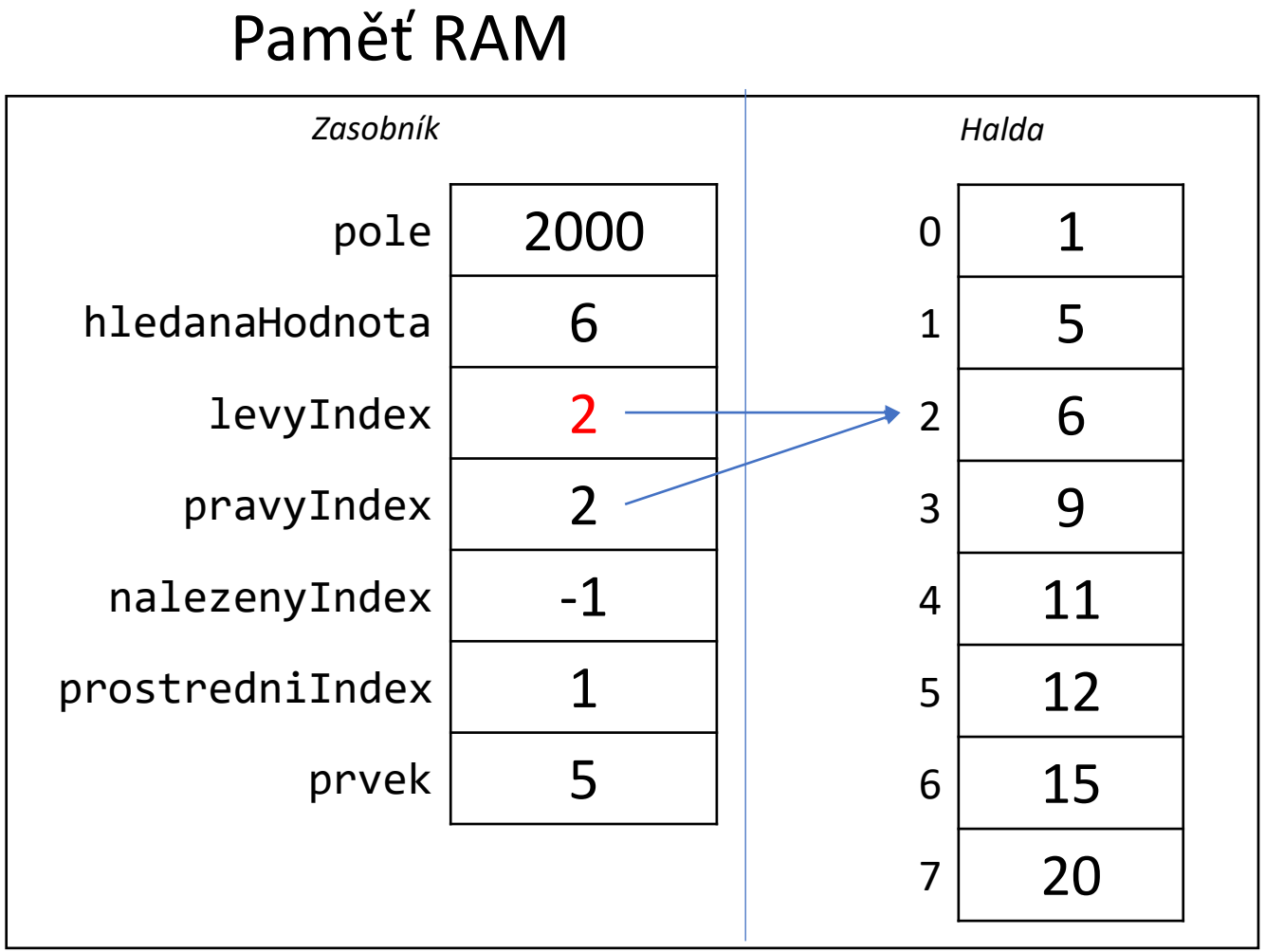
```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    int prvek = pole[prostredniIndex];

    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }

} while (levyIndex <= pravyIndex);
```

6 5



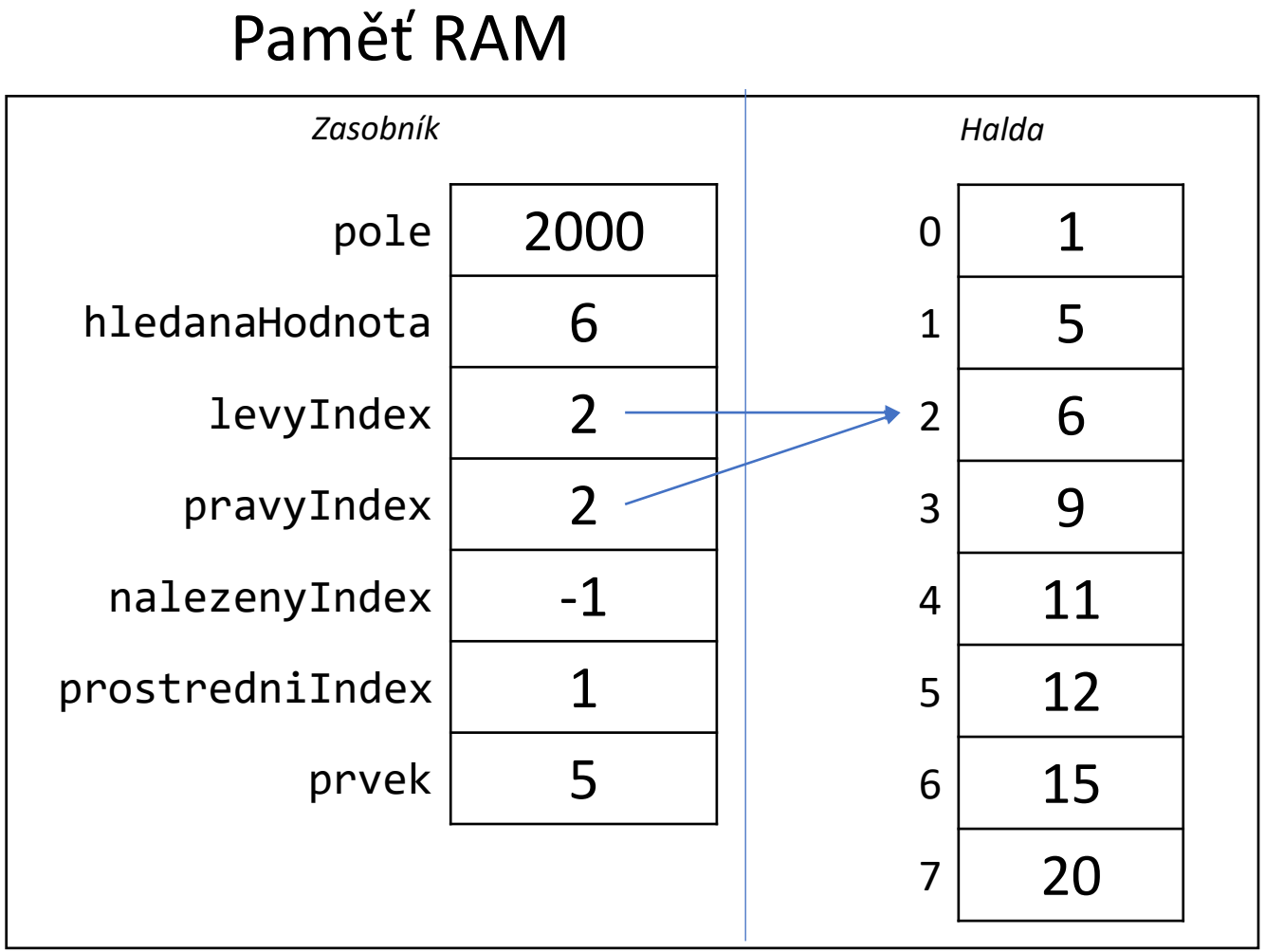
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    int prvek = pole[prostredniIndex];

    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
                                  1
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }

} while (levyIndex <= pravyIndex);
```



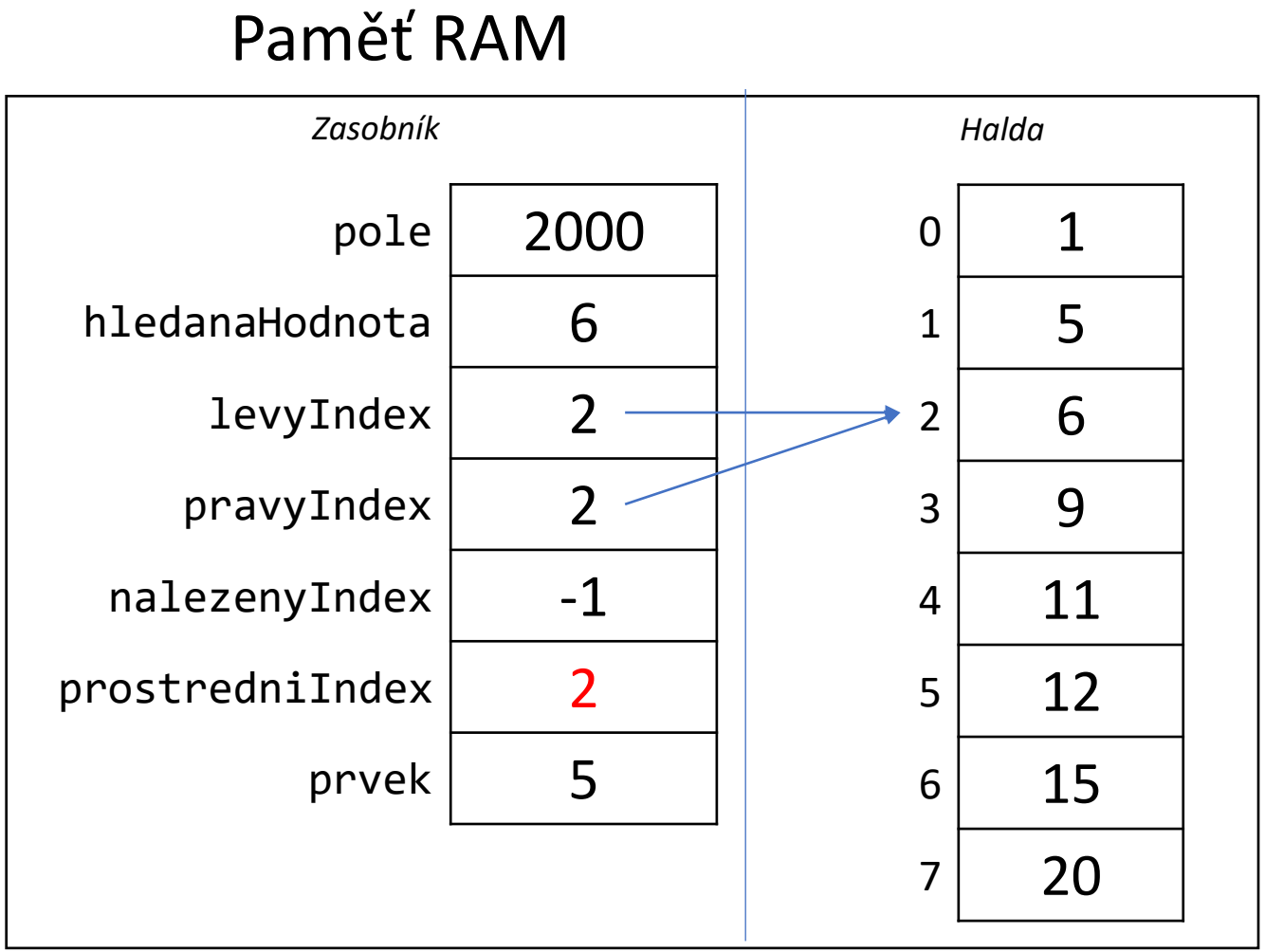
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    int prvek = pole[prostredniIndex];

    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }
            2           2
} while (levyIndex <= pravyIndex);
```



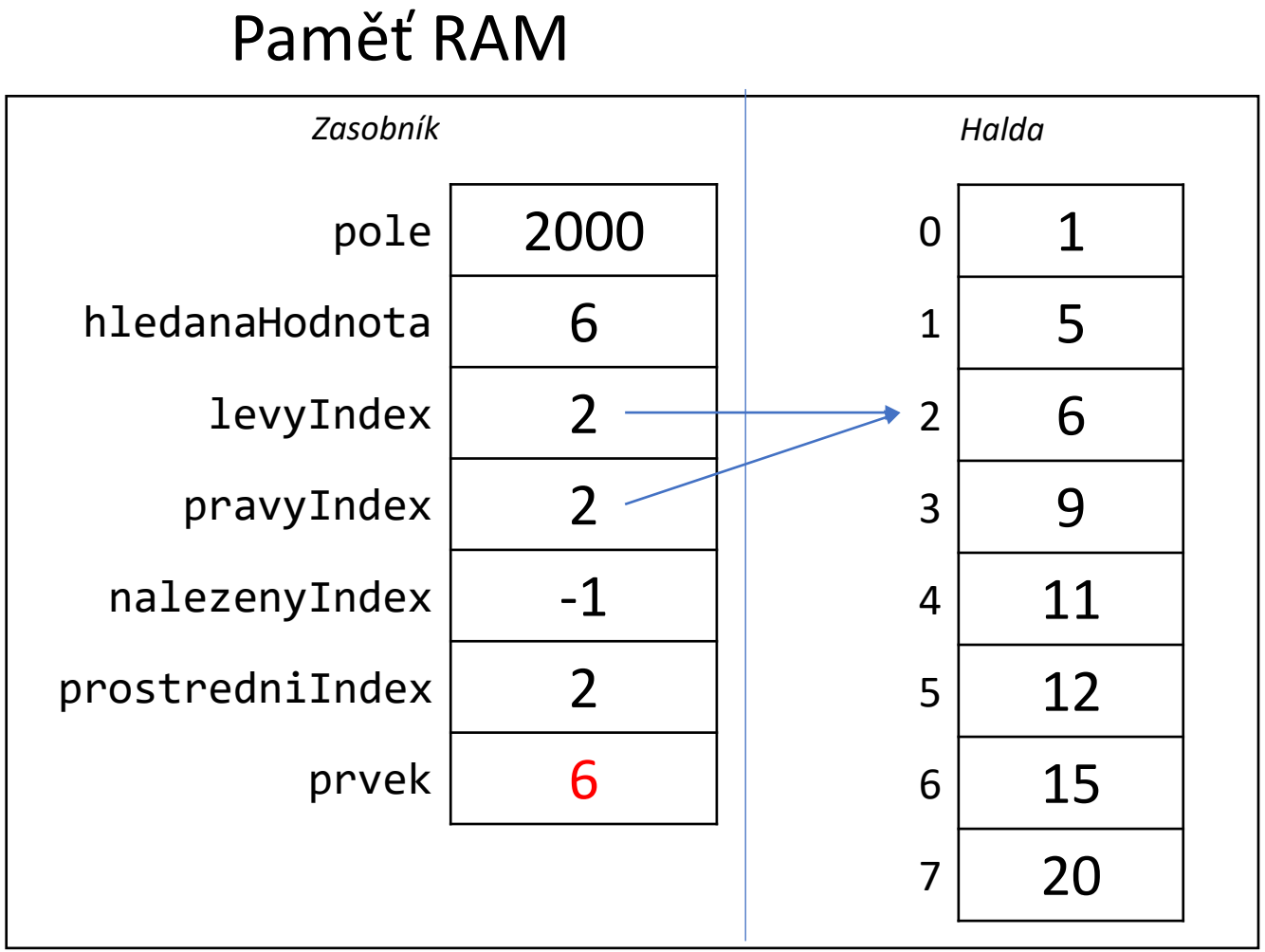
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{                                   2         2
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    int prvek = pole[prostredniIndex];

    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }

} while (levyIndex <= pravyIndex);
```



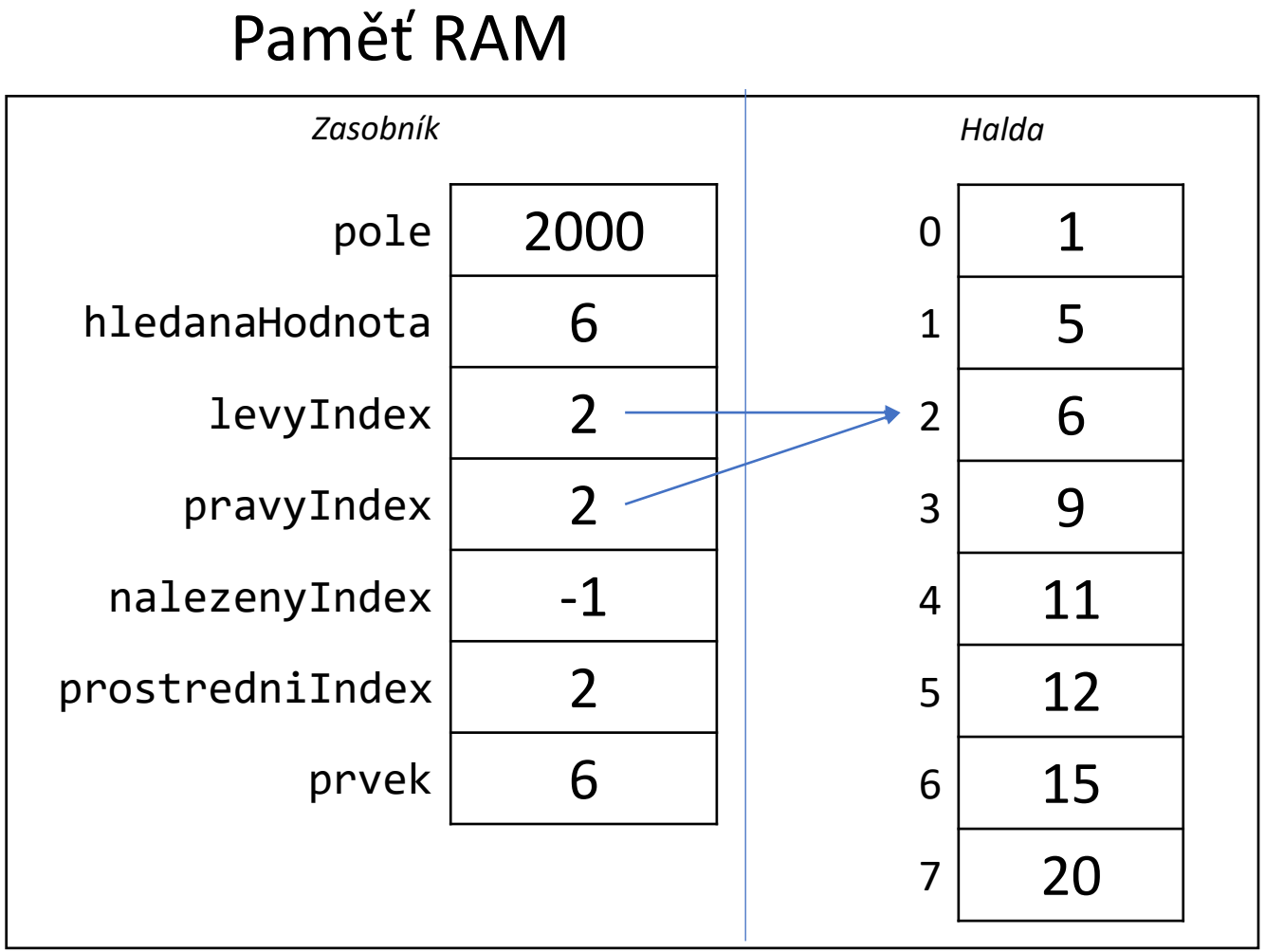
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;
                                                     2
    int prvek = pole[prostredniIndex];

    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }

} while (levyIndex <= pravyIndex);
```



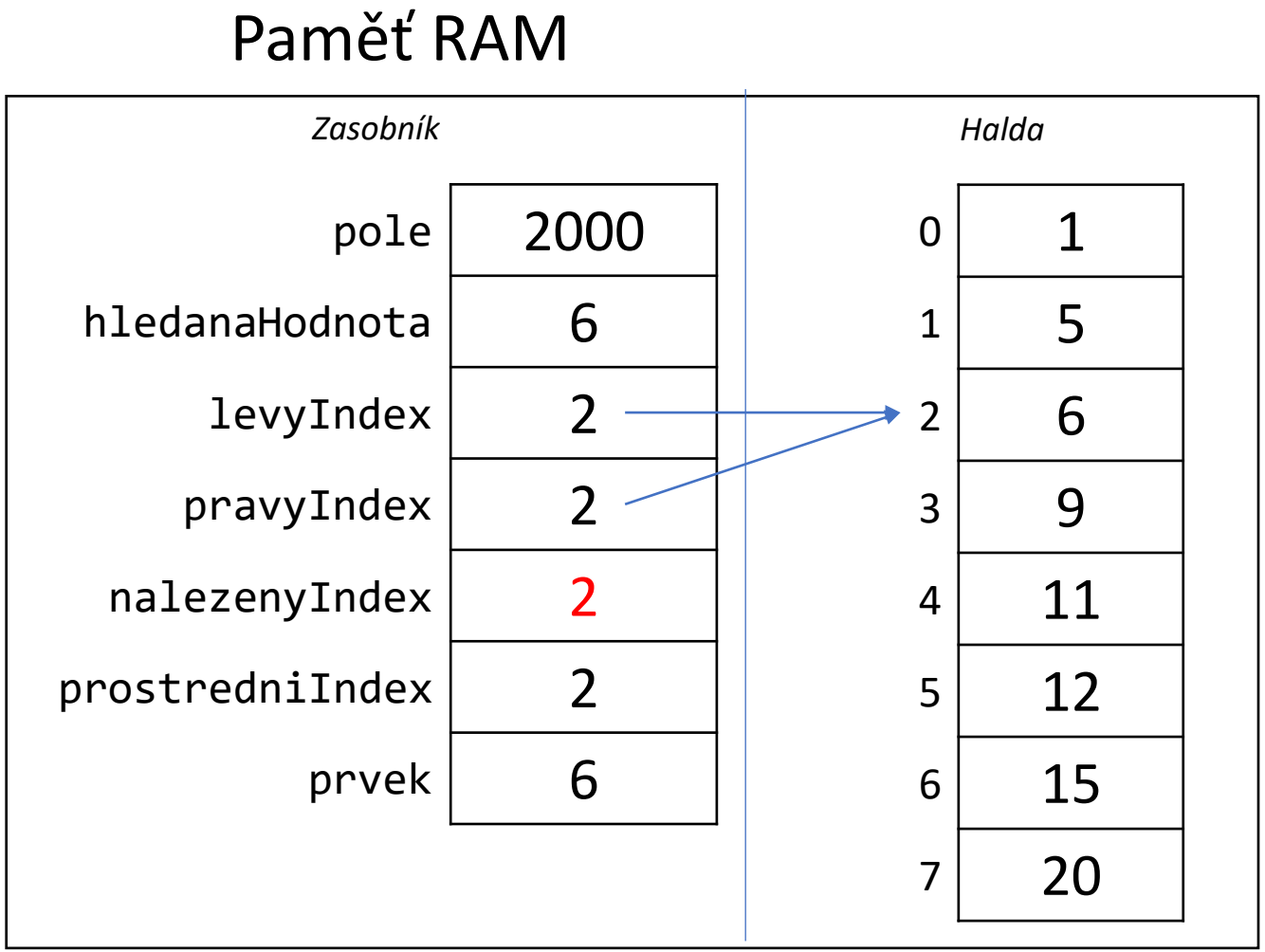
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    int prvek = pole[prostredniIndex];
            6           6
    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }

} while (levyIndex <= pravyIndex);
```



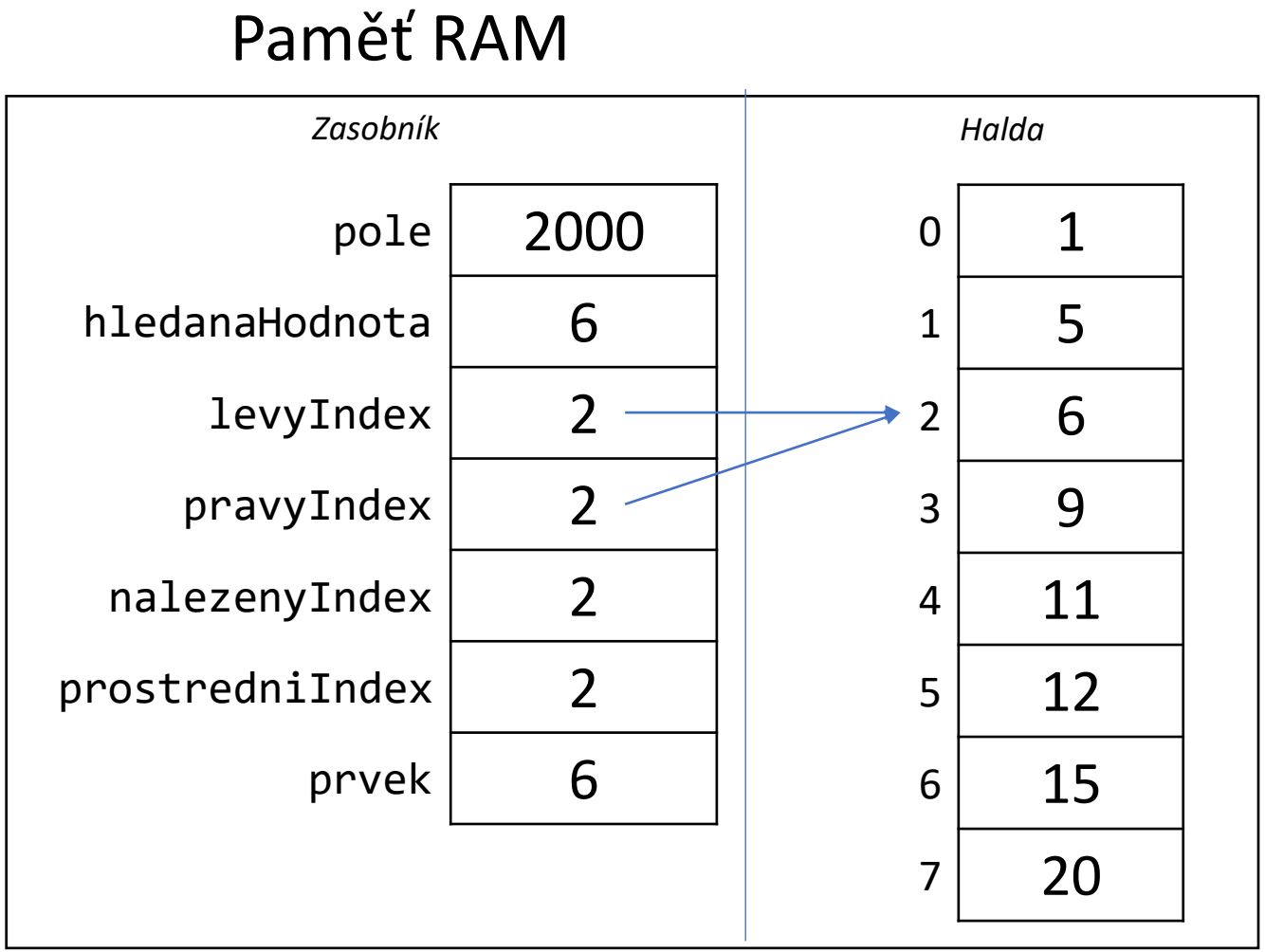
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    int prvek = pole[prostredniIndex];

    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }

} while (levyIndex <= pravyIndex);
```



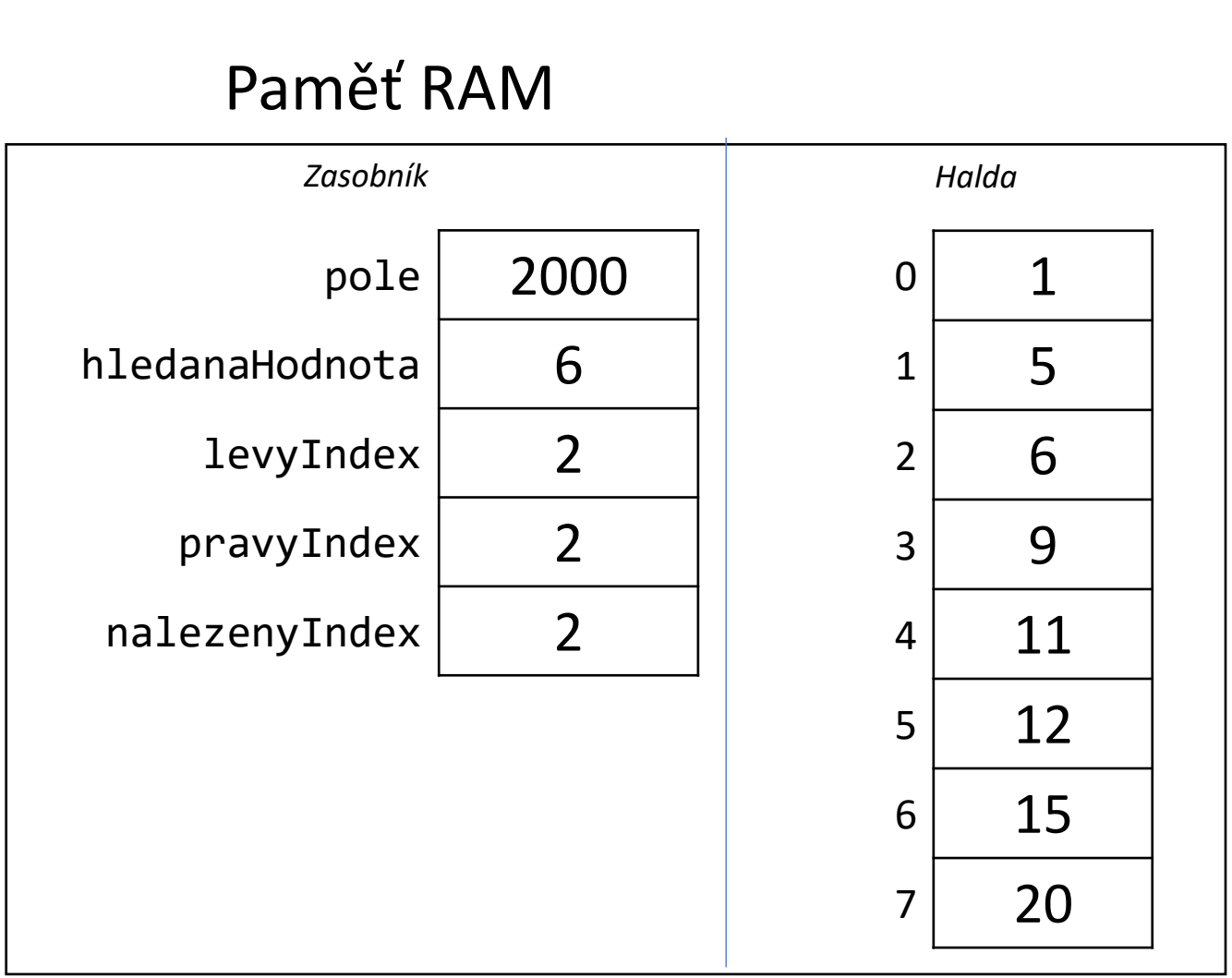
Binary search

```
int[] pole = new int[] { 1, 5, 6, 9, 11, 12, 15, 20 };
int hledanaHodnota = 6;

int levyIndex = 0;
int pravyIndex = pole.Length - 1;

int nalezenyIndex = -1;

do
{
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    int prvek = pole[prostredniIndex];

    if(hledanaHodnota == prvek)
    {
        nalezenyIndex = prostredniIndex;
        break;
    }
    else if(hledanaHodnota > prvek)
    {
        levyIndex = prostredniIndex + 1;
    }
    else
    {
        pravyIndex = prostredniIndex - 1;
    }

} while (levyIndex <= pravyIndex);
```



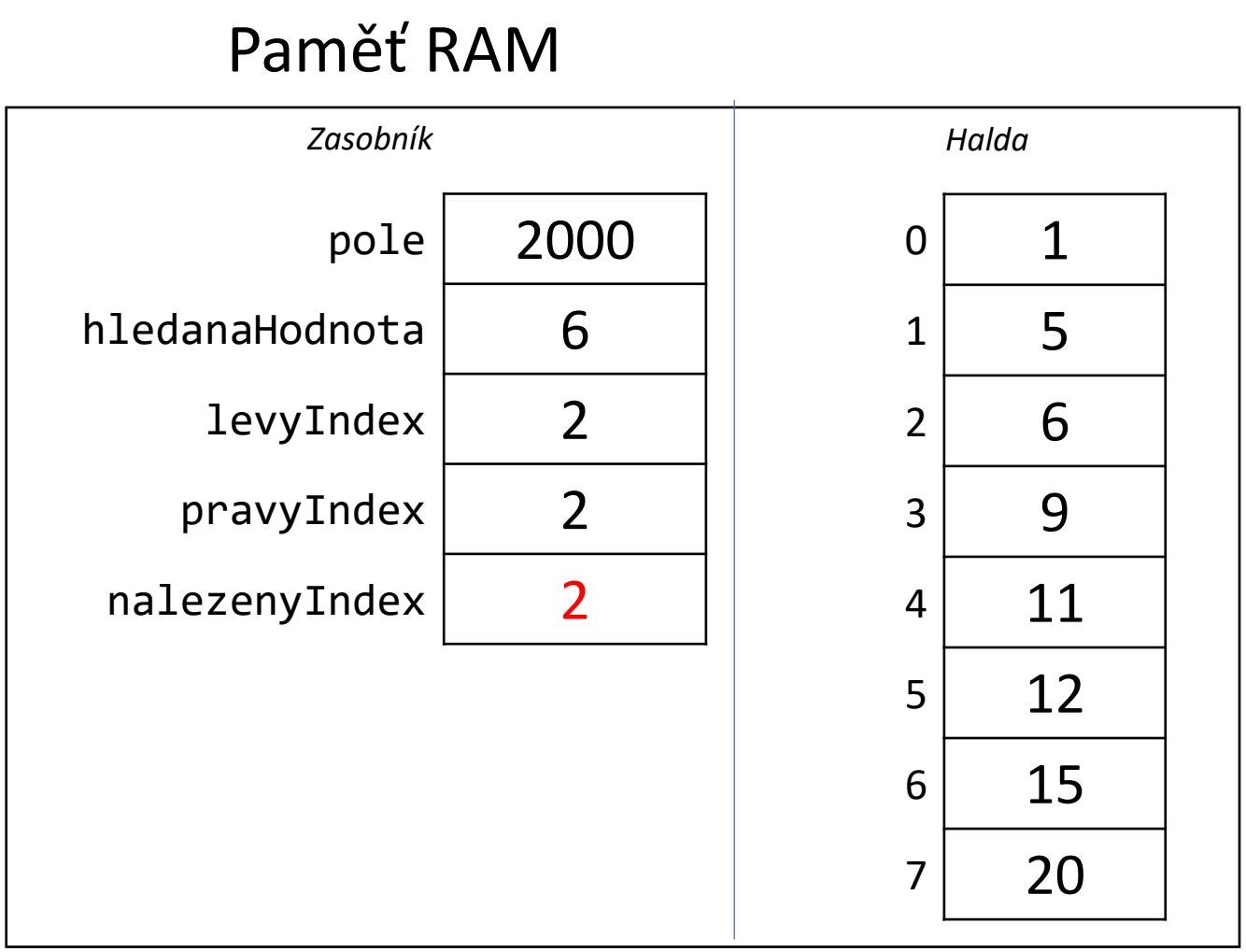
Binary search

```
        2
if (nalezenyIndex < 0)
{
    Console.WriteLine($"Hodnota nenalezena");
}
else
{
    Console.WriteLine(nalezenyIndex);
}
```



Binary search

```
if (nalezenyIndex < 0)
{
    Console.WriteLine($"Hodnota nenalezena");
}
else
{
    Console.WriteLine(nalezenyIndex);
}
```



Použité zdroje

[1] Single-Dimensional Arrays - C# Programming Guide | Microsoft Docs. [online]. Copyright © Microsoft 2021 [cit. 02.02.2021]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/single-dimensional-arrays

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Programování a algoritmizace

Děkuji za pozornost

Strategický projekt UTB ve Zlíně, reg. č. CZ.02.2.69/0.0/0.0/16_015/0002204

Ing. et Ing. Erik Král, Ph.D.
FAI, ÚPKS