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

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Programování a algoritmizace

Binary search

Rekurzivní implementace

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

Popis algoritmu Binary search

Popis rekurzivní implementace

Úvod

- V následujících snímcích probereme algoritmus binárního vyhledávání (Binary search).
- Na tomto příkladu si demonstrujeme práci s jednorozměrným polem s pevnou délkou [1] a rekurzivní metody [2].

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 se opakuje, dokud není nalezena cílová hodnota nebo už v poli nezbyly žádné prvky k prohledání.
- V této prezentaci probereme řešení s **rekurzivní metodou**.

Algoritmus a paměť

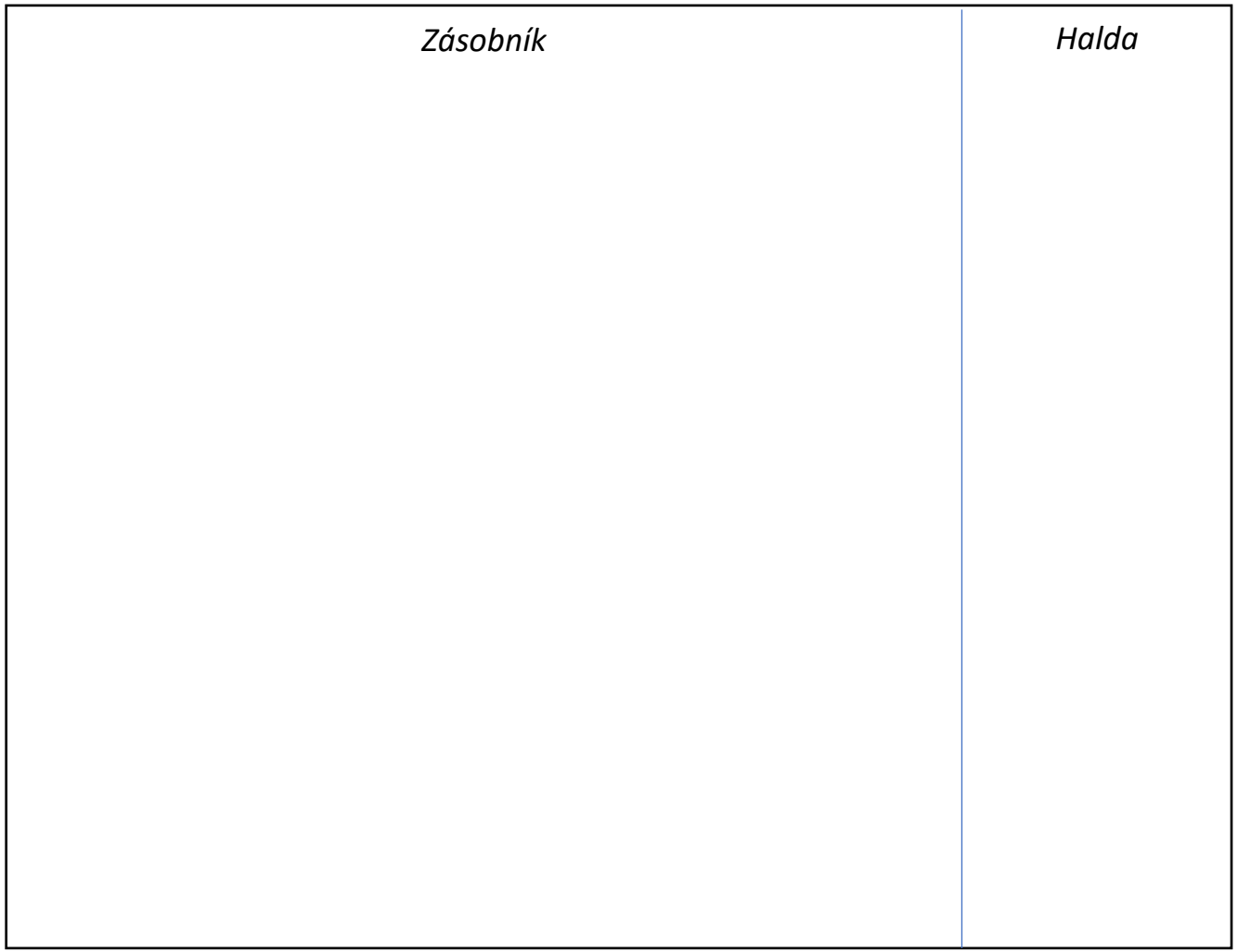
- Rekurzivní algoritmus alokuje pro každé volání metody novou paměť.
- V příkladech je **zjednodušeně** demonstrováno využití paměti z hlediska zásobníku (Stack) a haldy (Heap).
- Práce se zásobníkem je ve skutečnosti složitější a v příkladech jsou zobrazeny **pouze proměnné přímo související s algoritmem** a jsou vynechány uložené hodnoty registrů nebo návratové hodnoty. Také pořadí předávaných argumentů a parametrů metody může být jiné.
- Návratová hodnota metody se může předávat jak v registru, tak i jiným způsobem [3].

Binary search

- Následující příklad se skládá z volání metody *BinarySearch* v metodě *Main* a následujícím vyhodnocení návratové hodnoty.
- A dále obsahuje příklad samotnou rekurzivní implementaci metody *BinarySearch*.

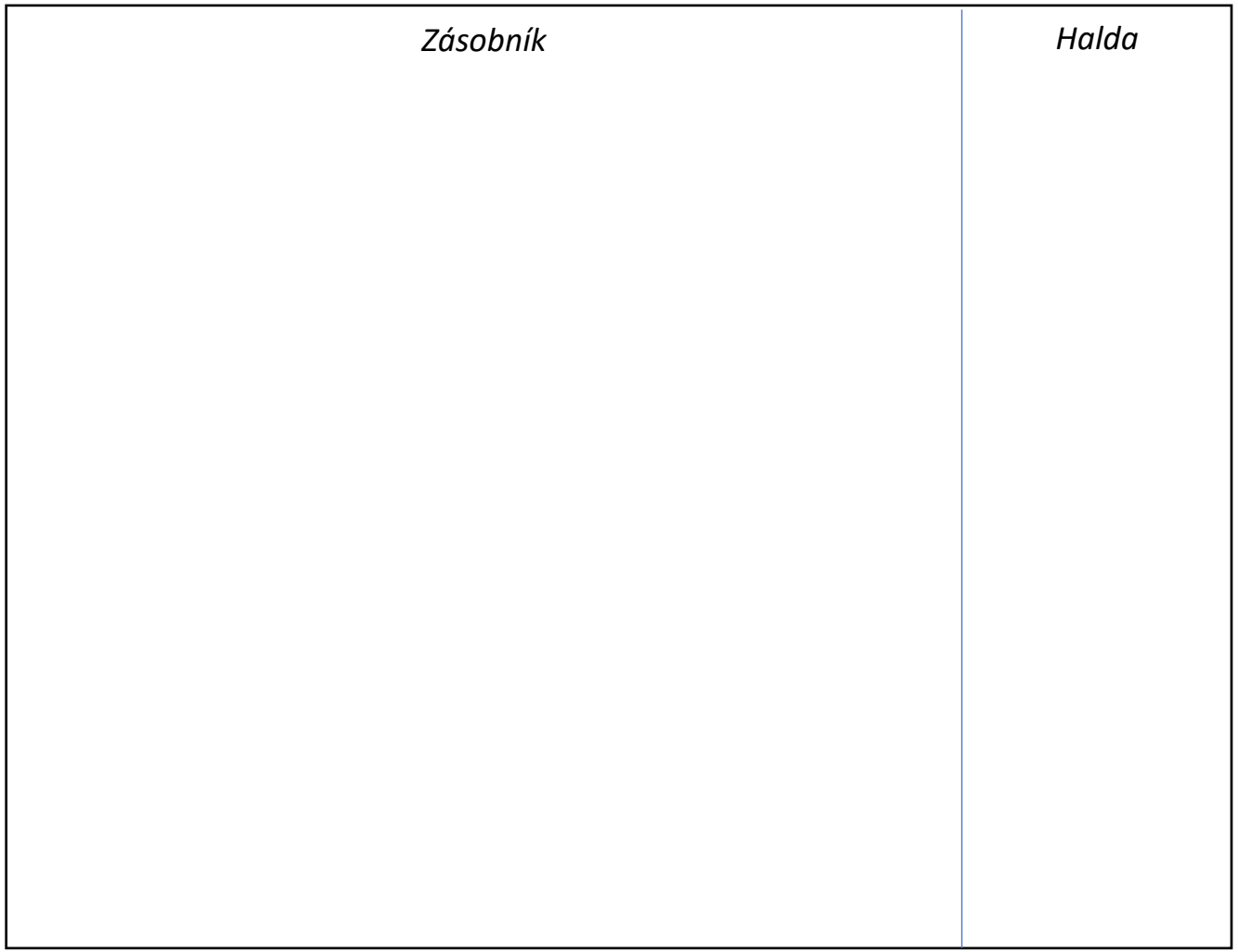
Binary search - rekurzivní

```
static void Main(string[] args)
{
    int[] pole = new int[]
    { 1, 5, 6, 9, 11, 12, 15, 20 };

    int hledanaHodnota = 15;

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

    int nalezenyIndex = BinarySearch(pole,
        hledanaHodnota,
        levyIndex,
        pravyIndex);

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



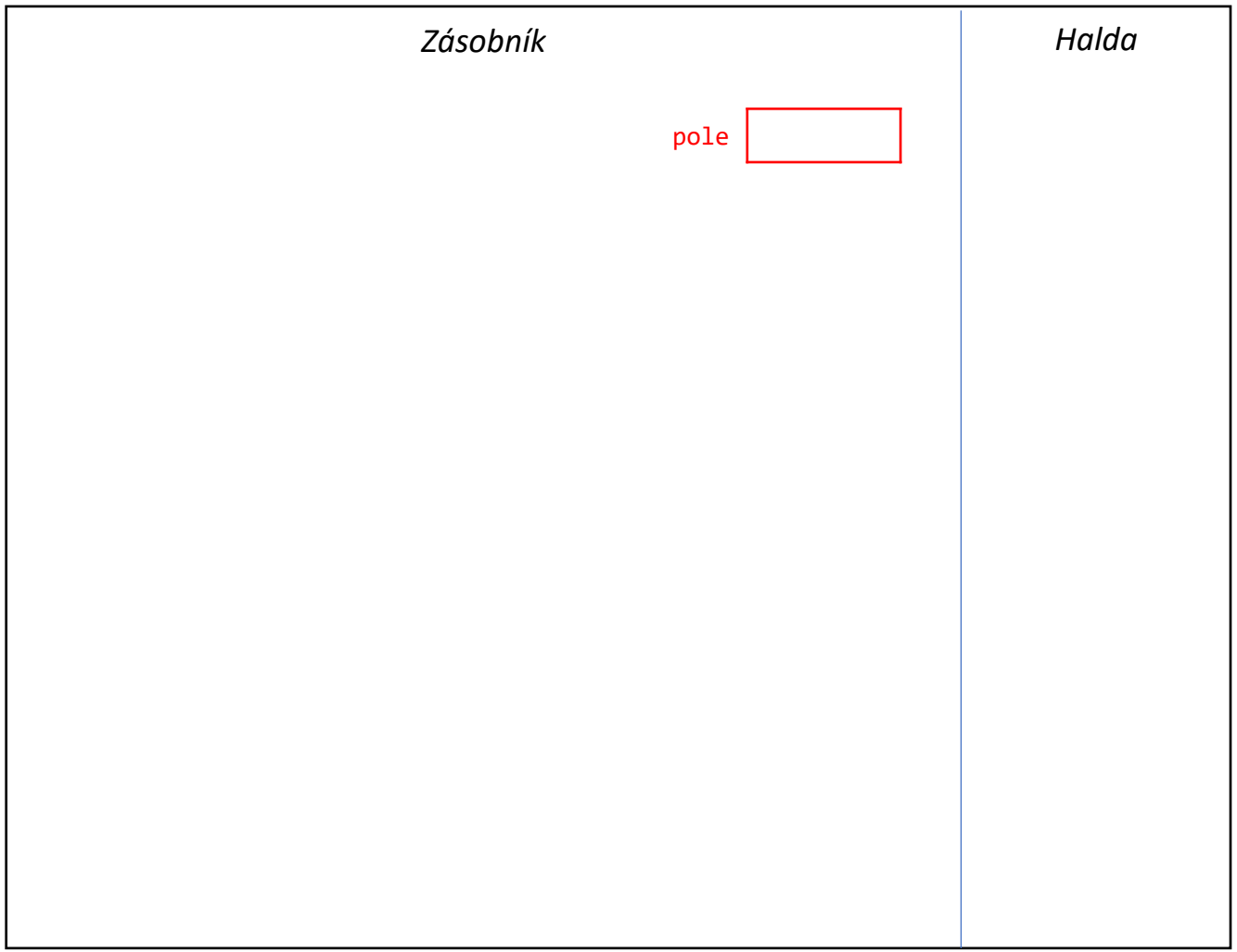
Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```



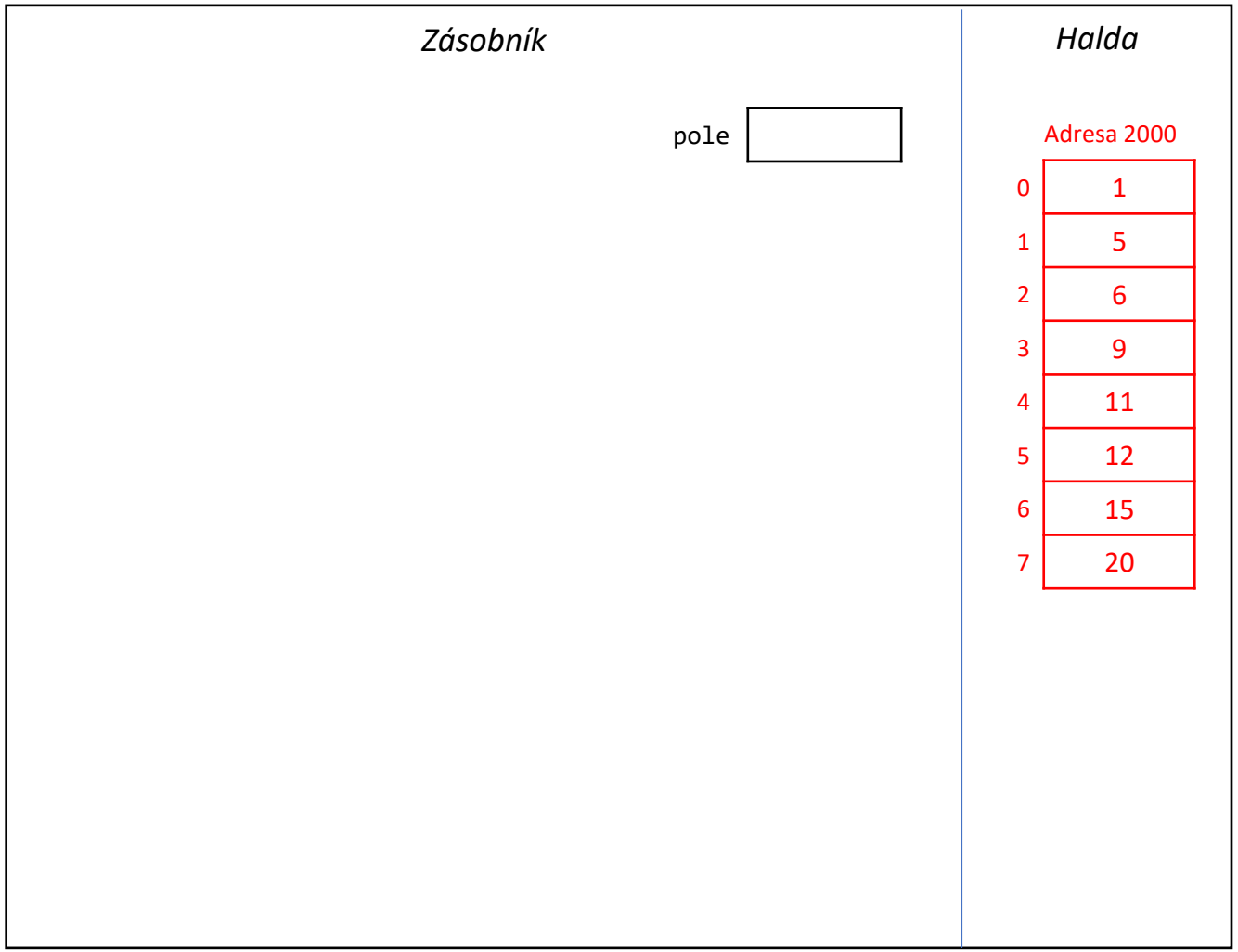
Binary search - rekurzivní

```
static void Main(string[] args)
{
    int[] pole = new int[]
    { 1, 5, 6, 9, 11, 12, 15, 20 };




}
```



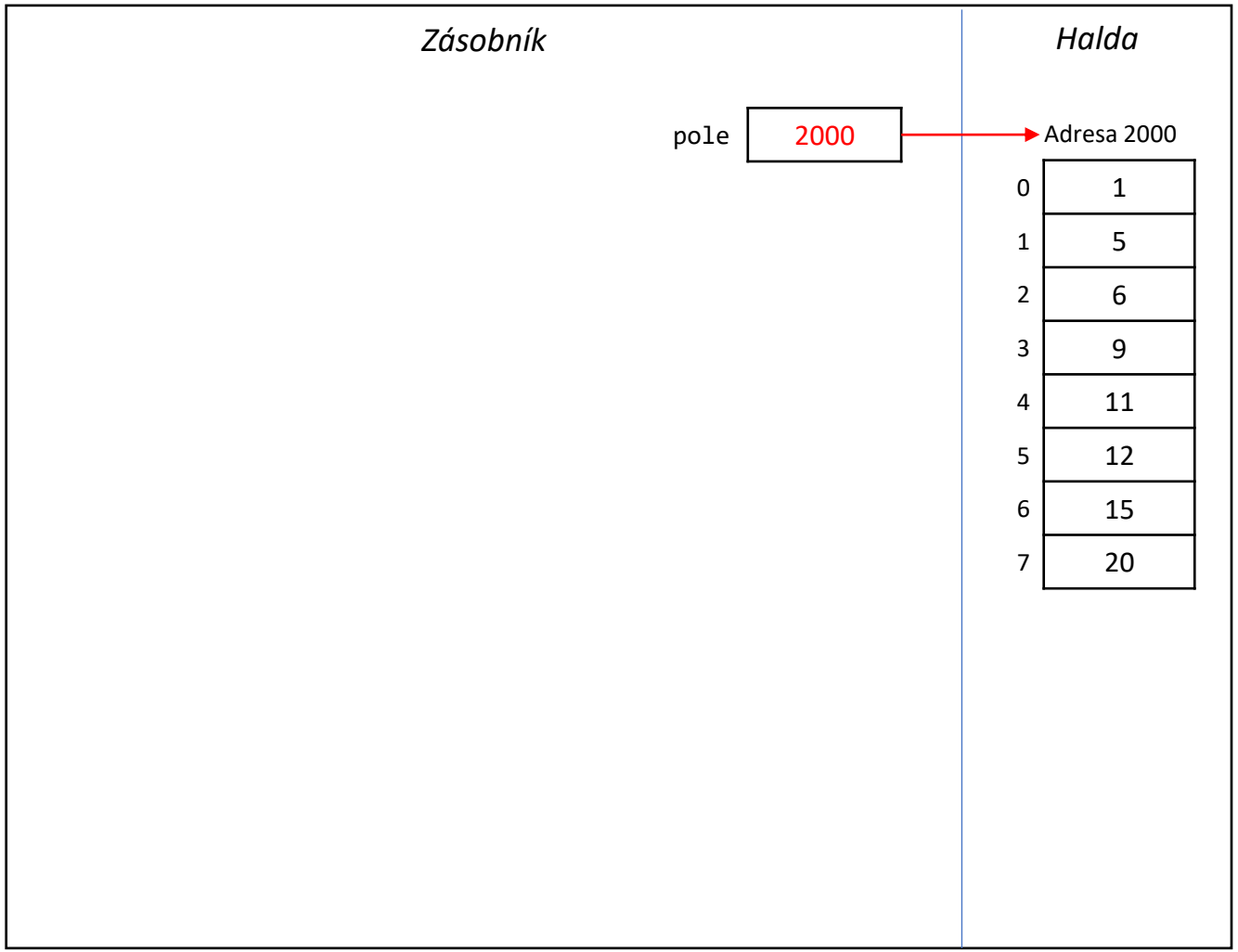
Binary search - rekurzivní

```
static void Main(string[] args)
{
    int[] pole = new int[]
    { 1, 5, 6, 9, 11, 12, 15, 20 };




}
```



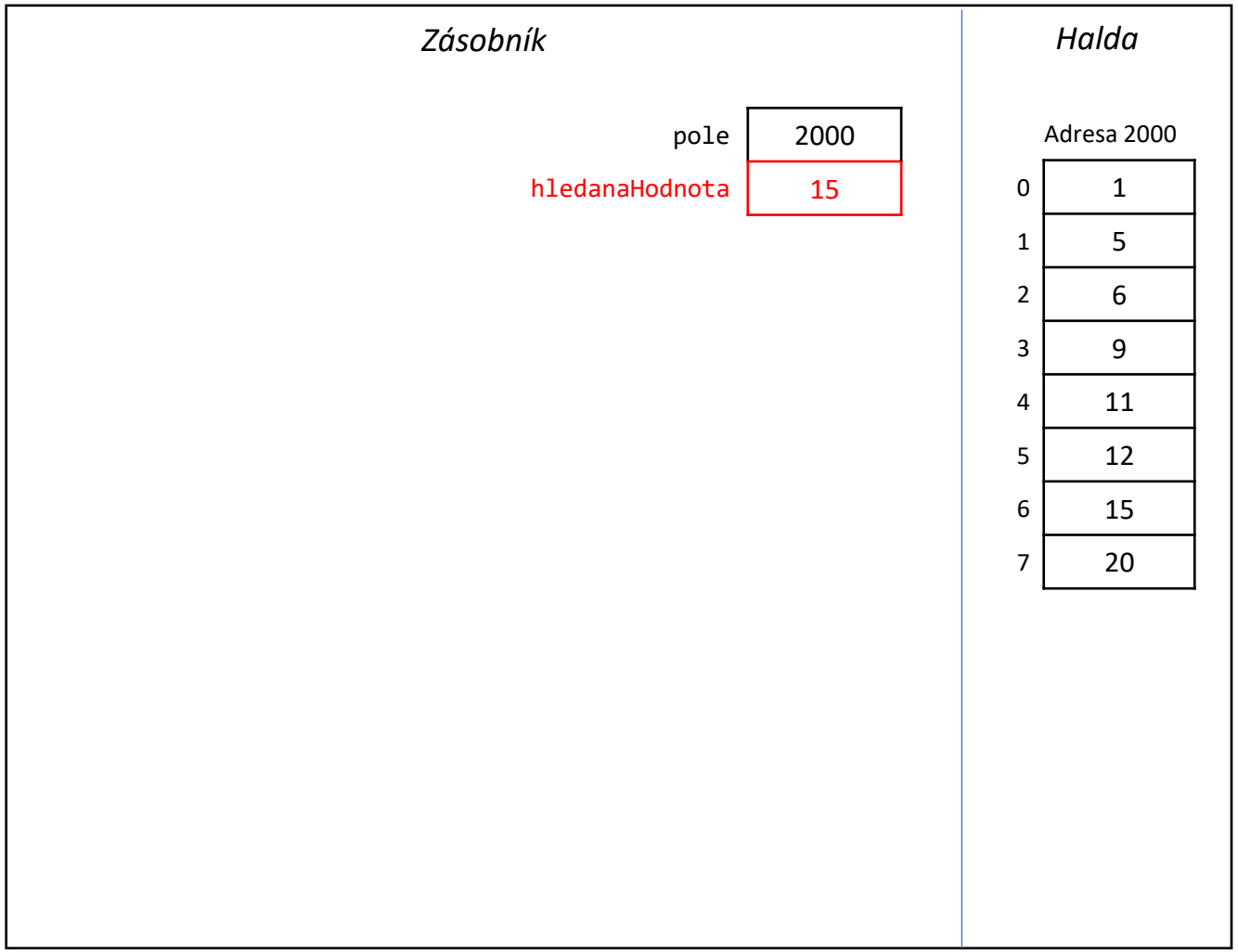
Binary search - rekurzivní

```
static void Main(string[] args)
{
    int[] pole = new int[]
    { 1, 5, 6, 9, 11, 12, 15, 20 };

}
```



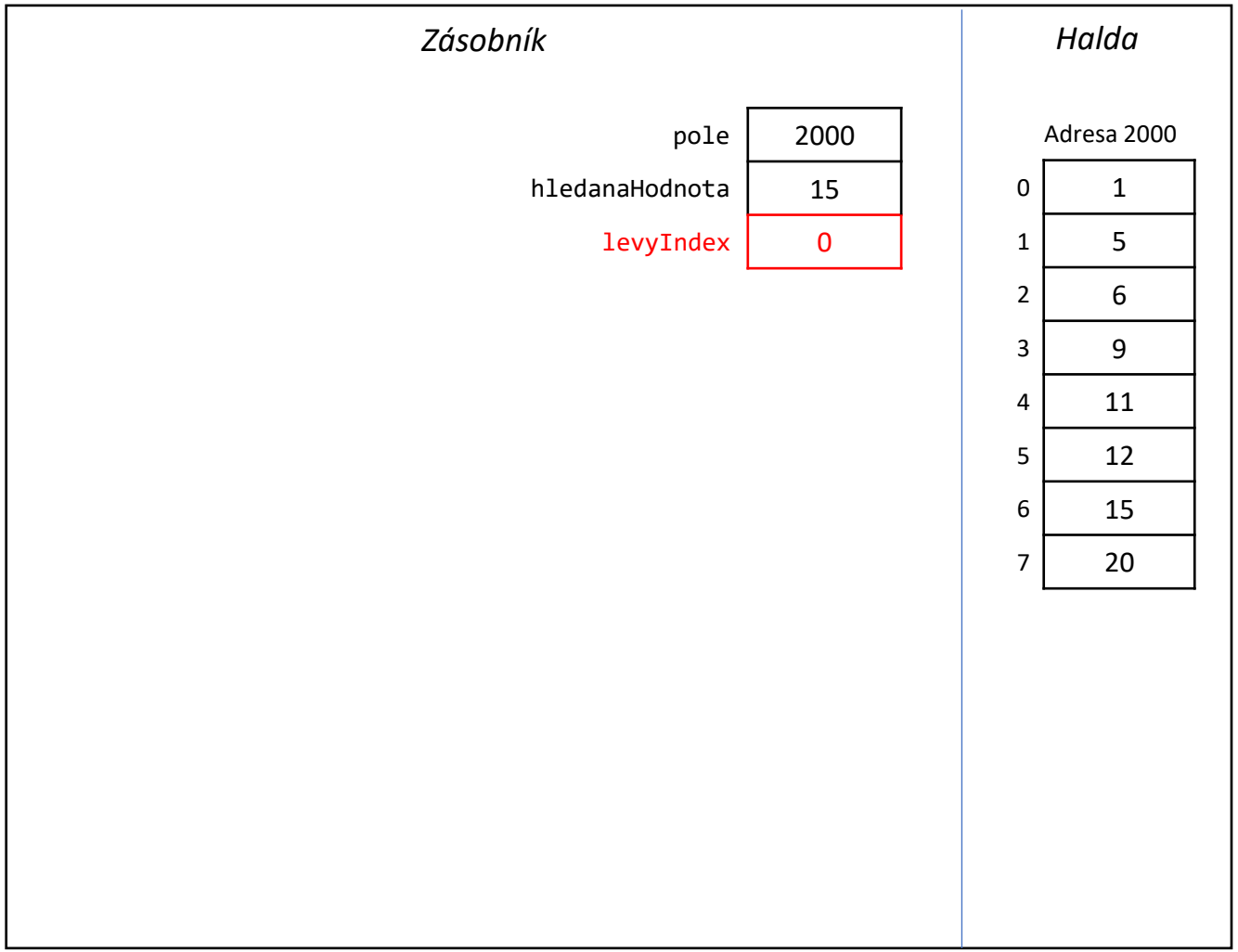
Binary search - rekurzivní

```
static void Main(string[] args)
{
    int[] pole = new int[]
    { 1, 5, 6, 9, 11, 12, 15, 20 };

    int hledanaHodnota = 15;









}
```



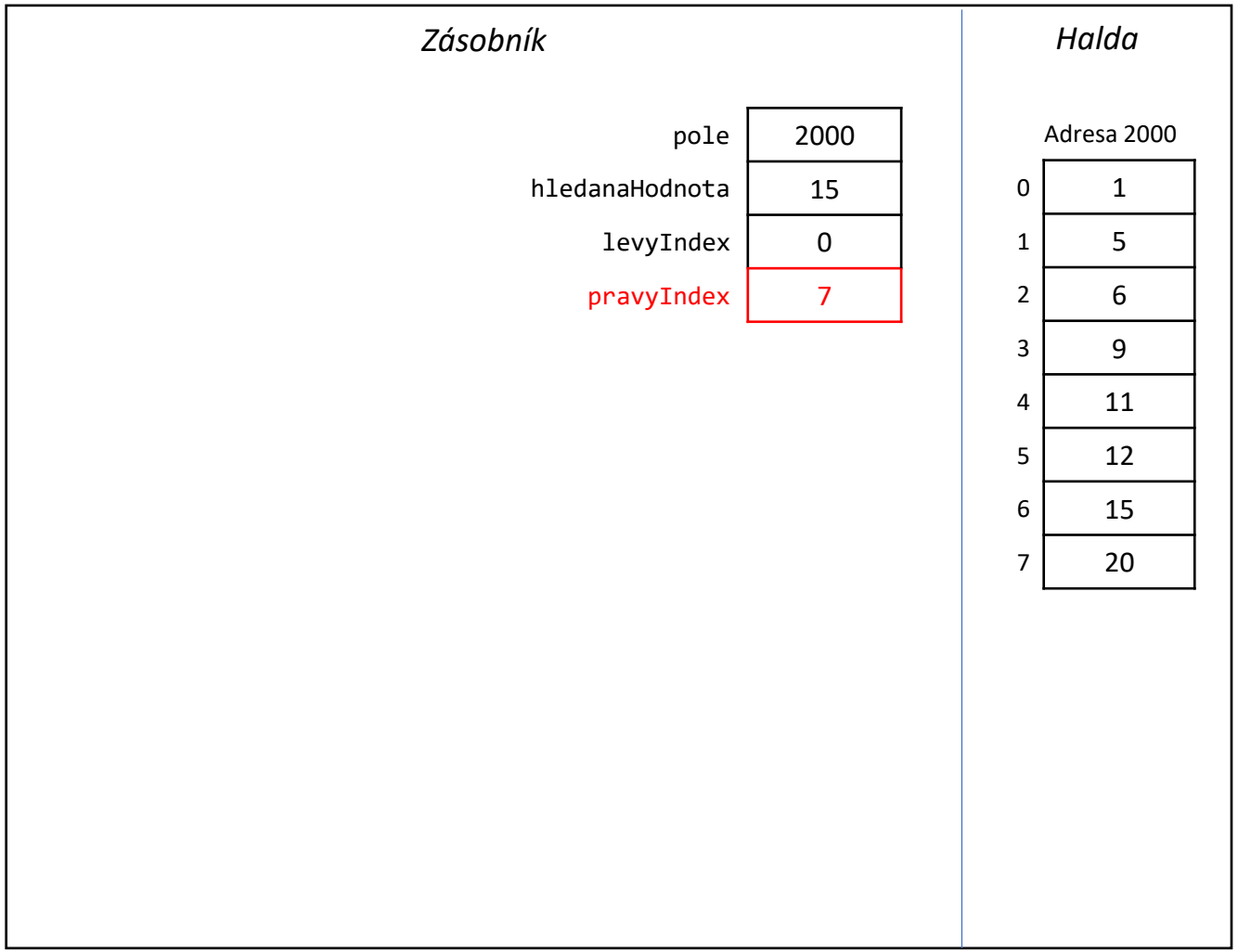
Binary search - rekurzivní

```
static void Main(string[] args)
{
    int[] pole = new int[]
    { 1, 5, 6, 9, 11, 12, 15, 20 };

    int hledanaHodnota = 15;

    int levyIndex = 0;









}
```



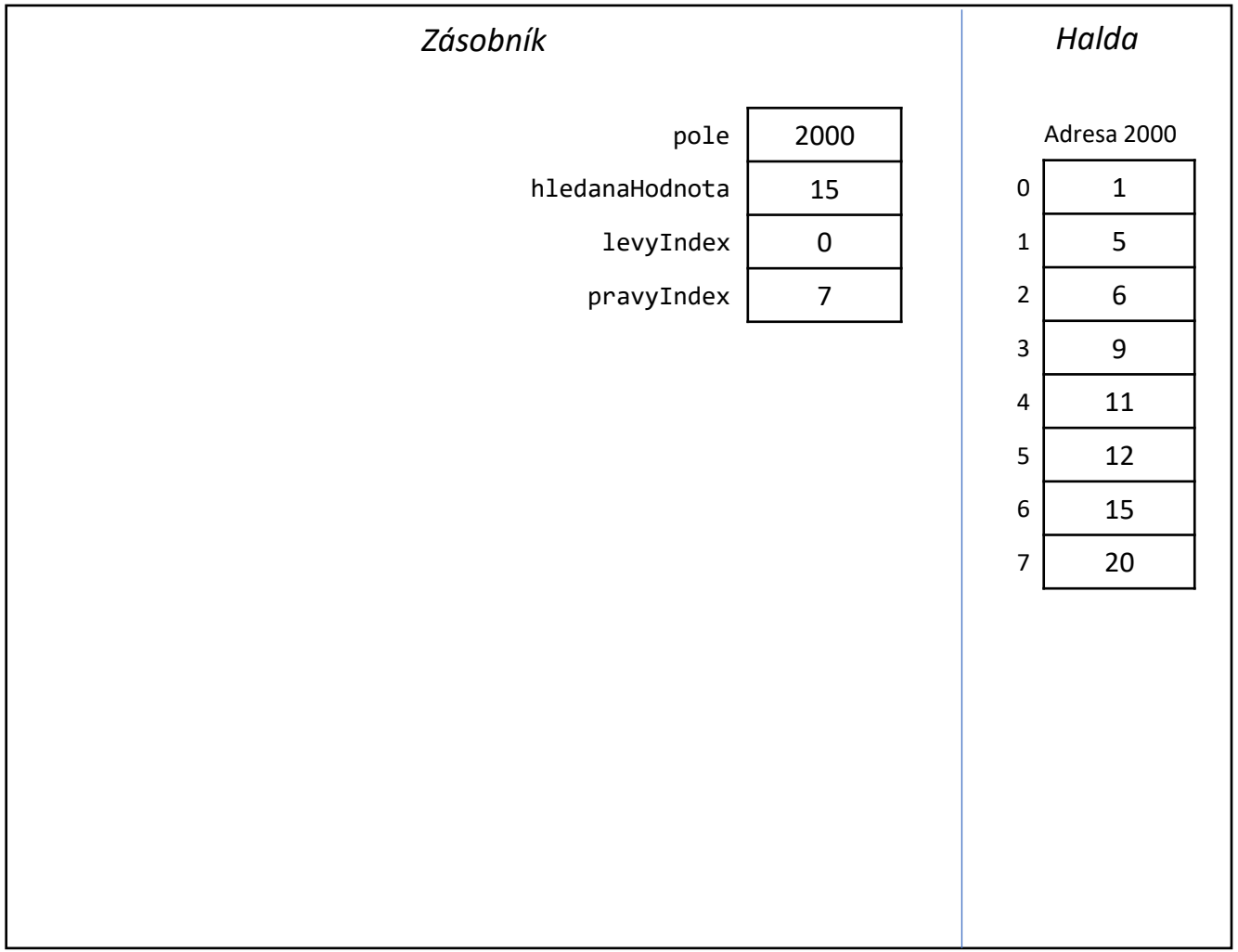
Binary search - rekurzivní

```
static void Main(string[] args)
{
    int[] pole = new int[]
    { 1, 5, 6, 9, 11, 12, 15, 20 };

    int hledanaHodnota = 15;

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









}
```



Binary search - rekurzivní

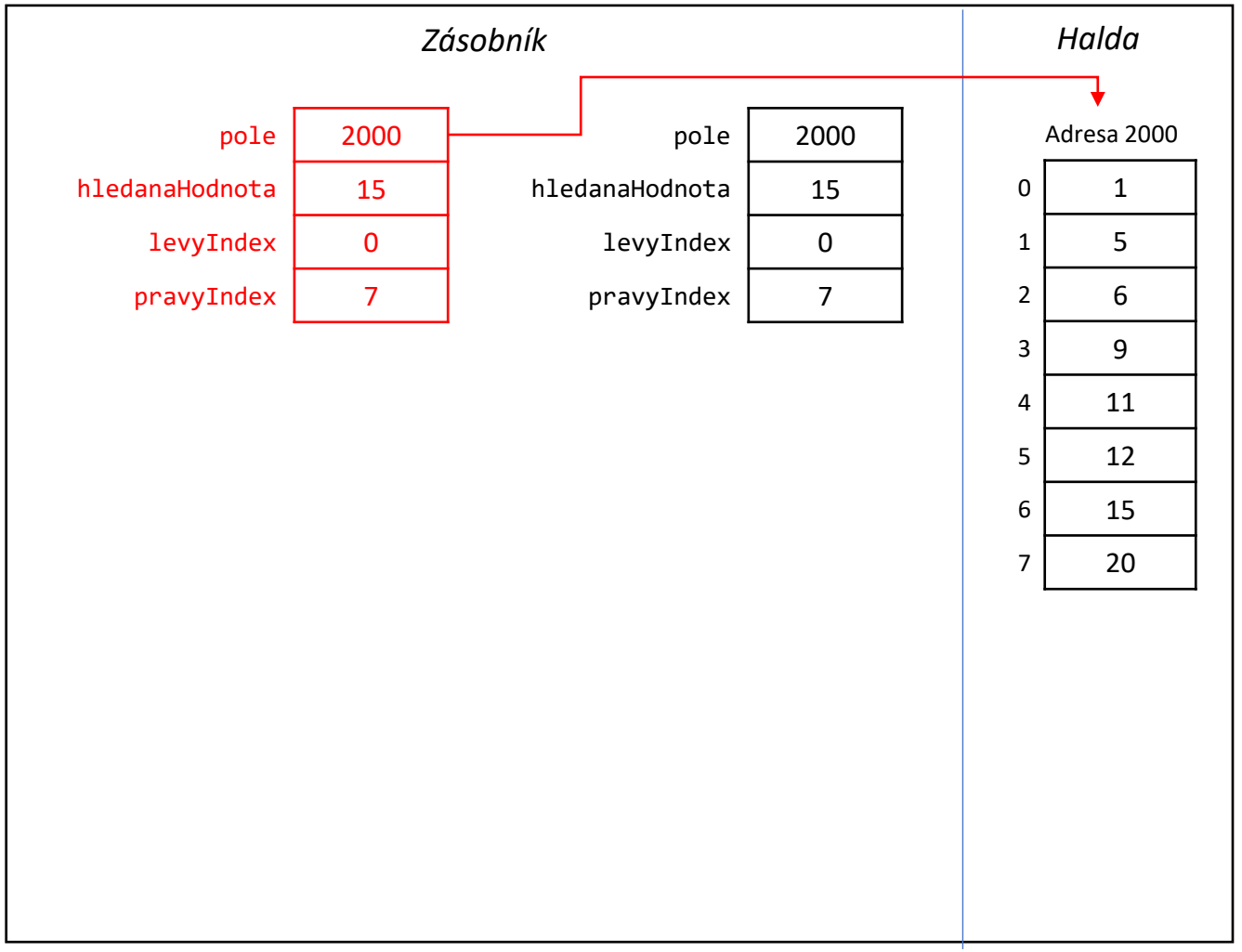
```
static void Main(string[] args)
{
    int[] pole = new int[]
    { 1, 5, 6, 9, 11, 12, 15, 20 };

    int hledanaHodnota = 15;

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

    int nalezenyIndex = BinarySearch(pole,
        hledanaHodnota,
        levyIndex,
        pravyIndex);









}
```



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{









}
```

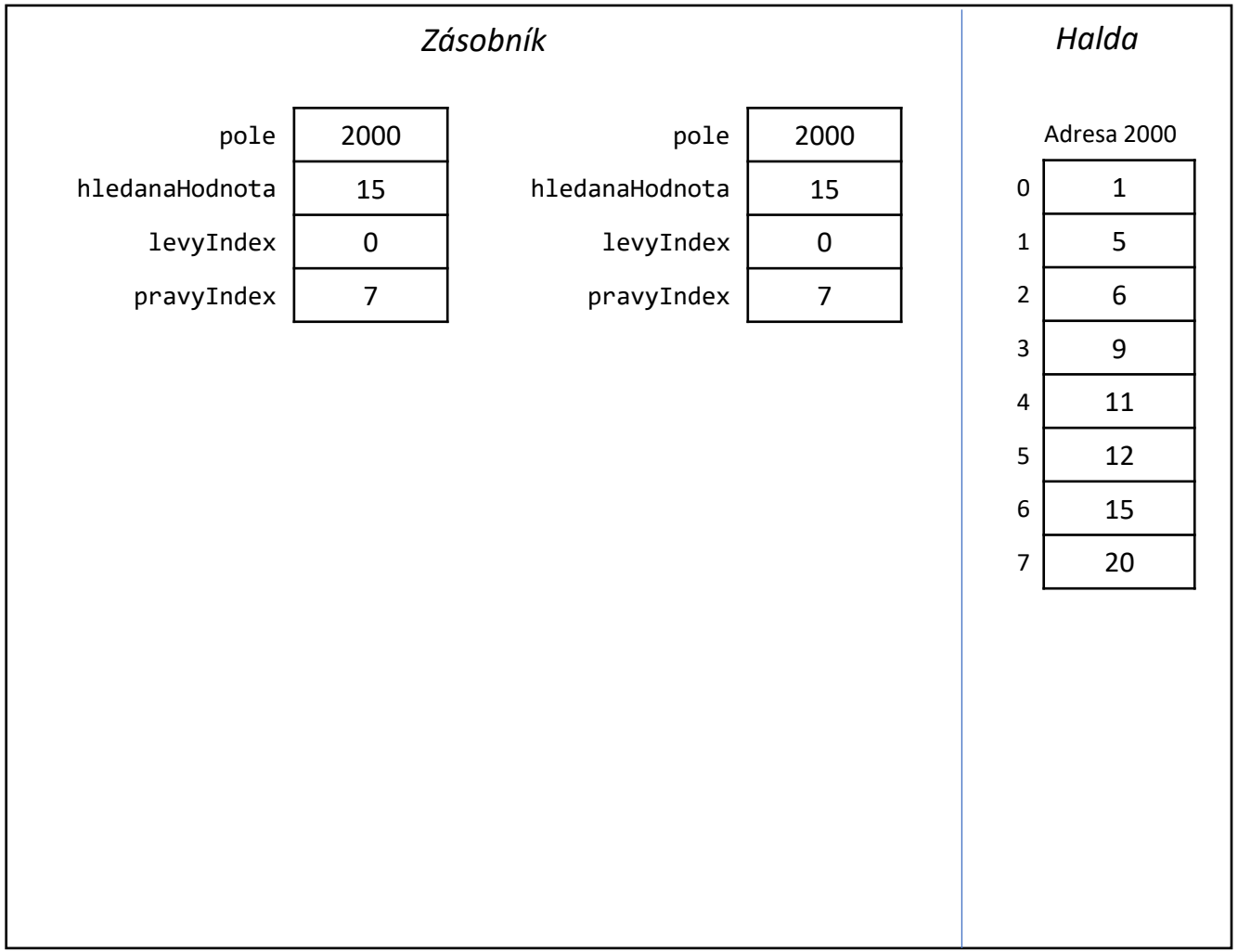
První volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{           0           7
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;









}
```

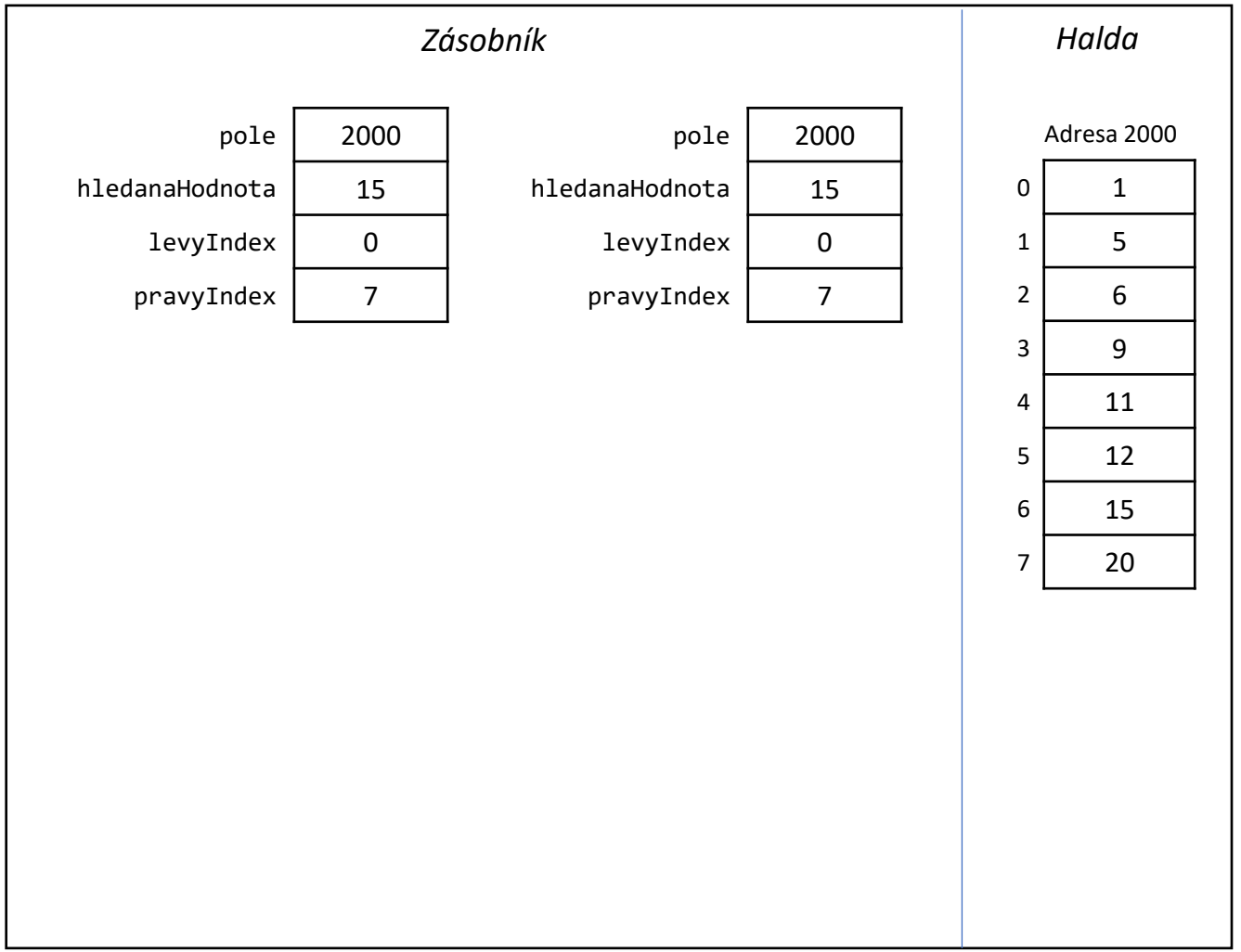
První volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;
                                0           7
    int prostredniIndex = (levyIndex + pravyIndex) / 2;









}
```

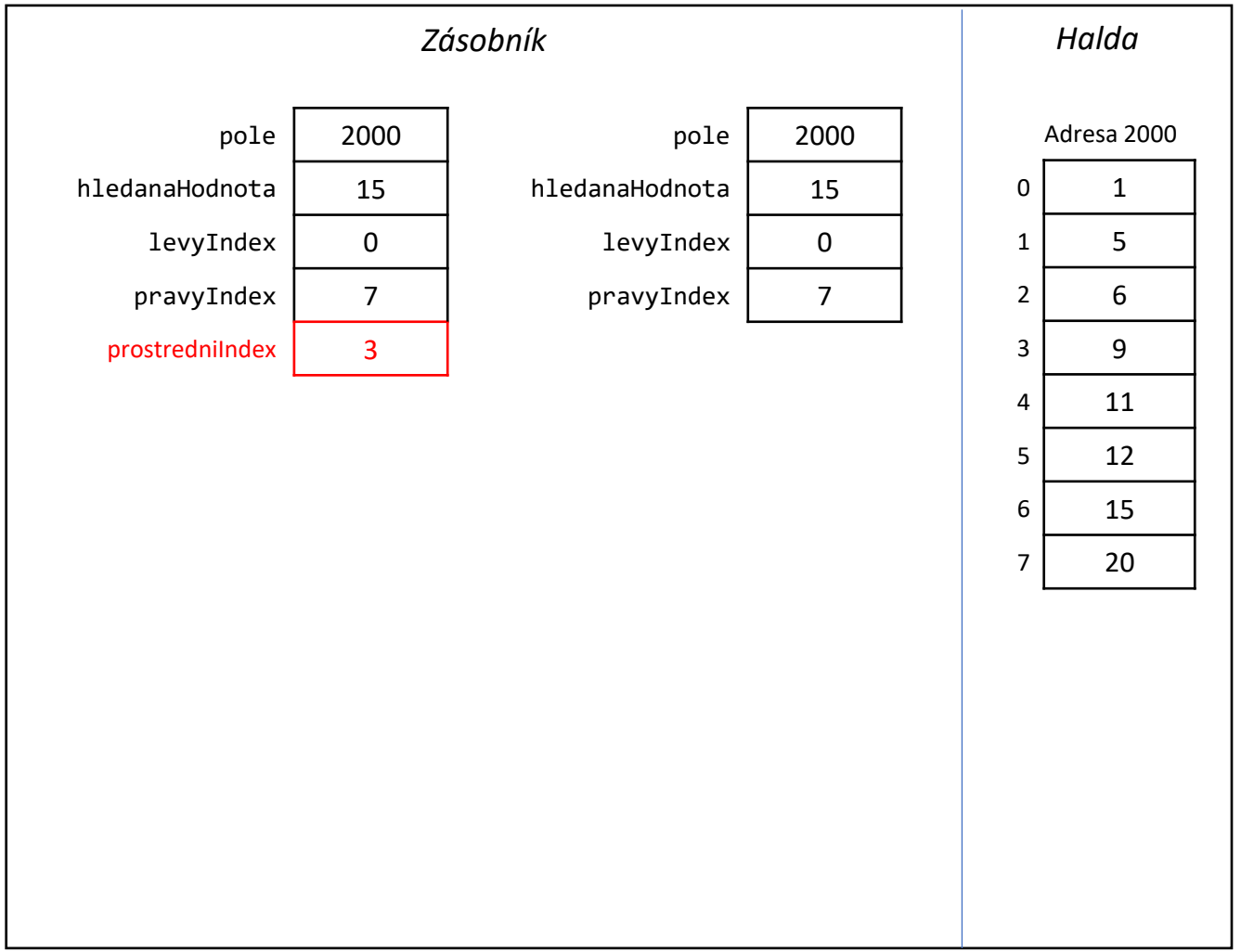
První volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;









}
```

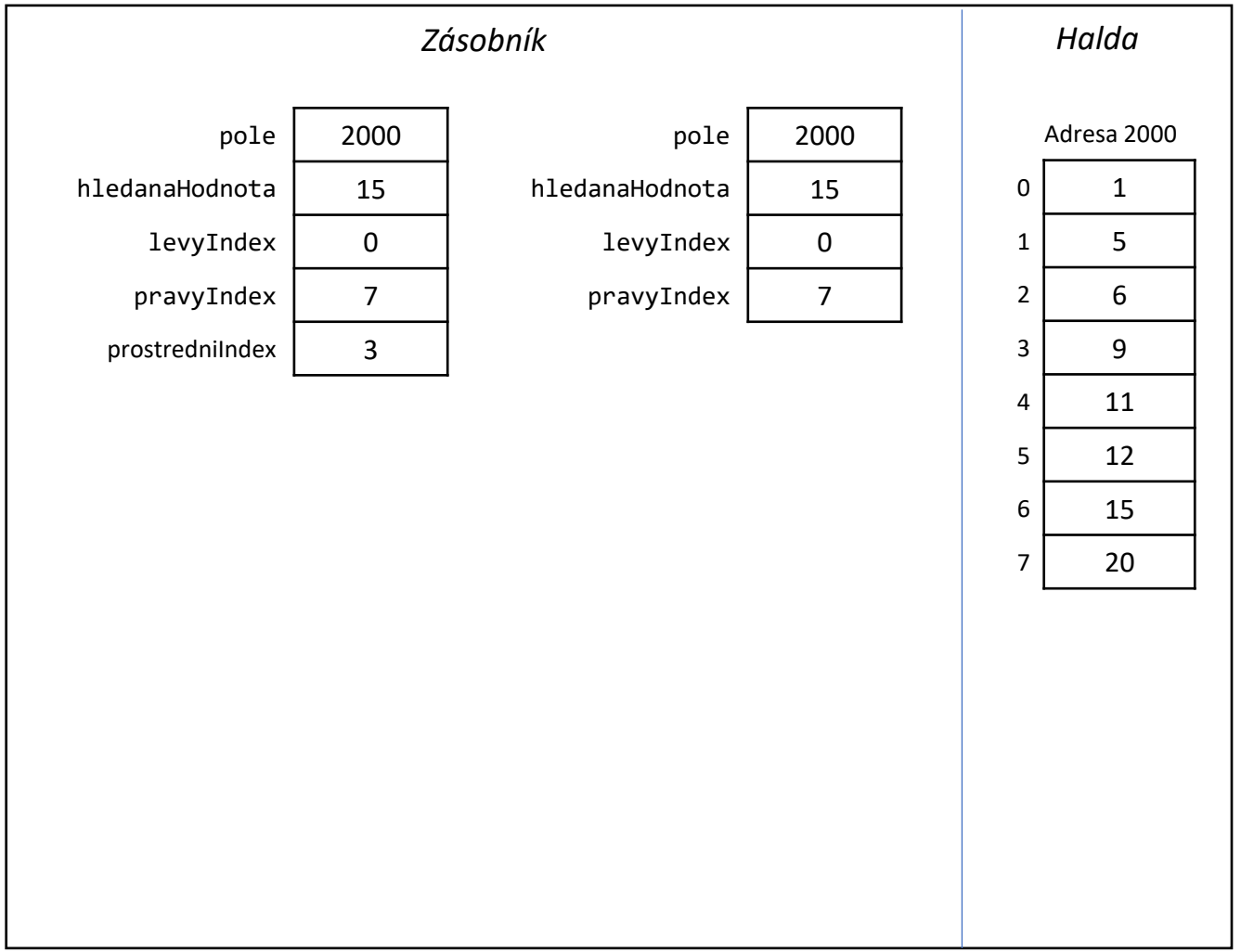
První volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;
                  9                      15
    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

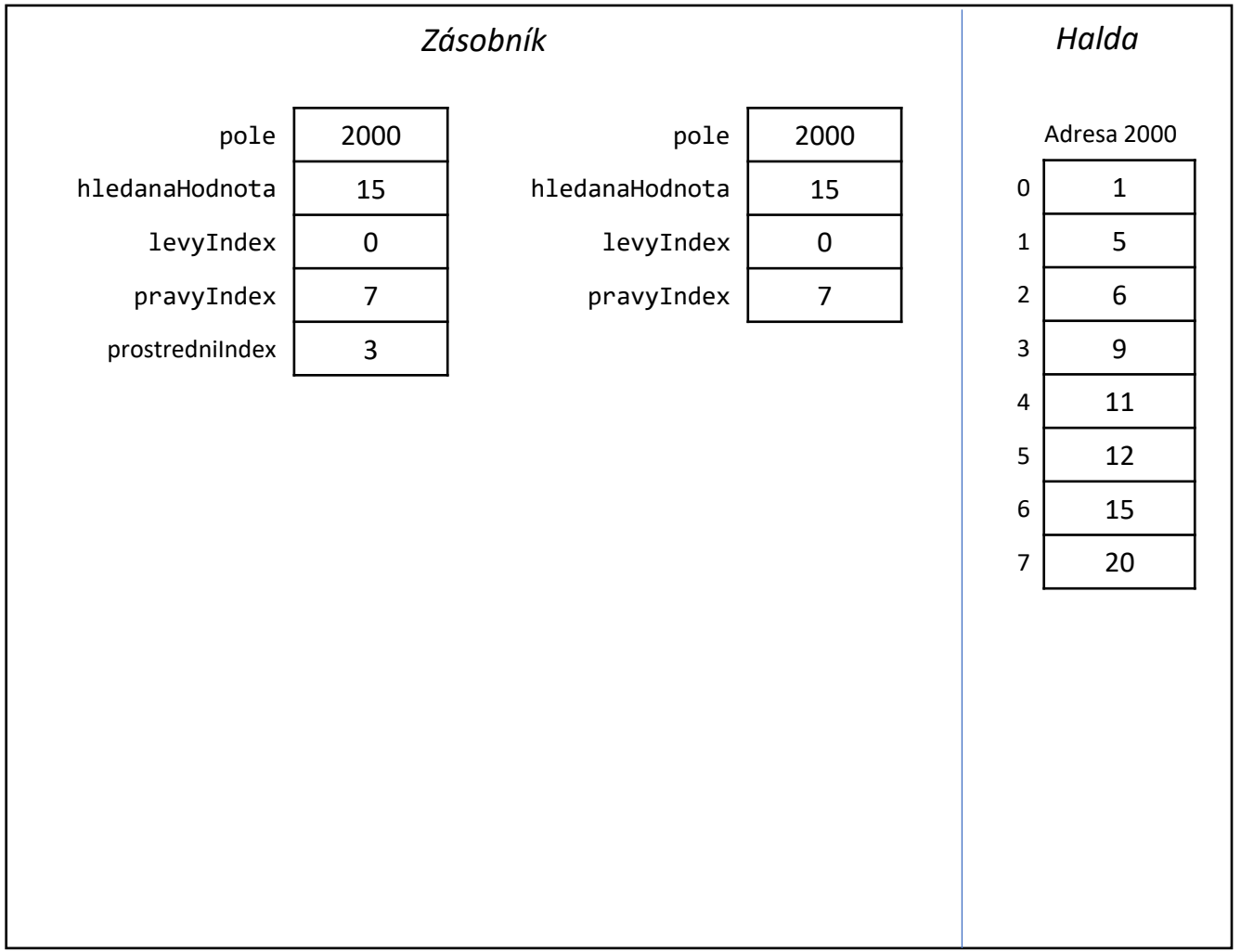
První volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;
                    9                    15
    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

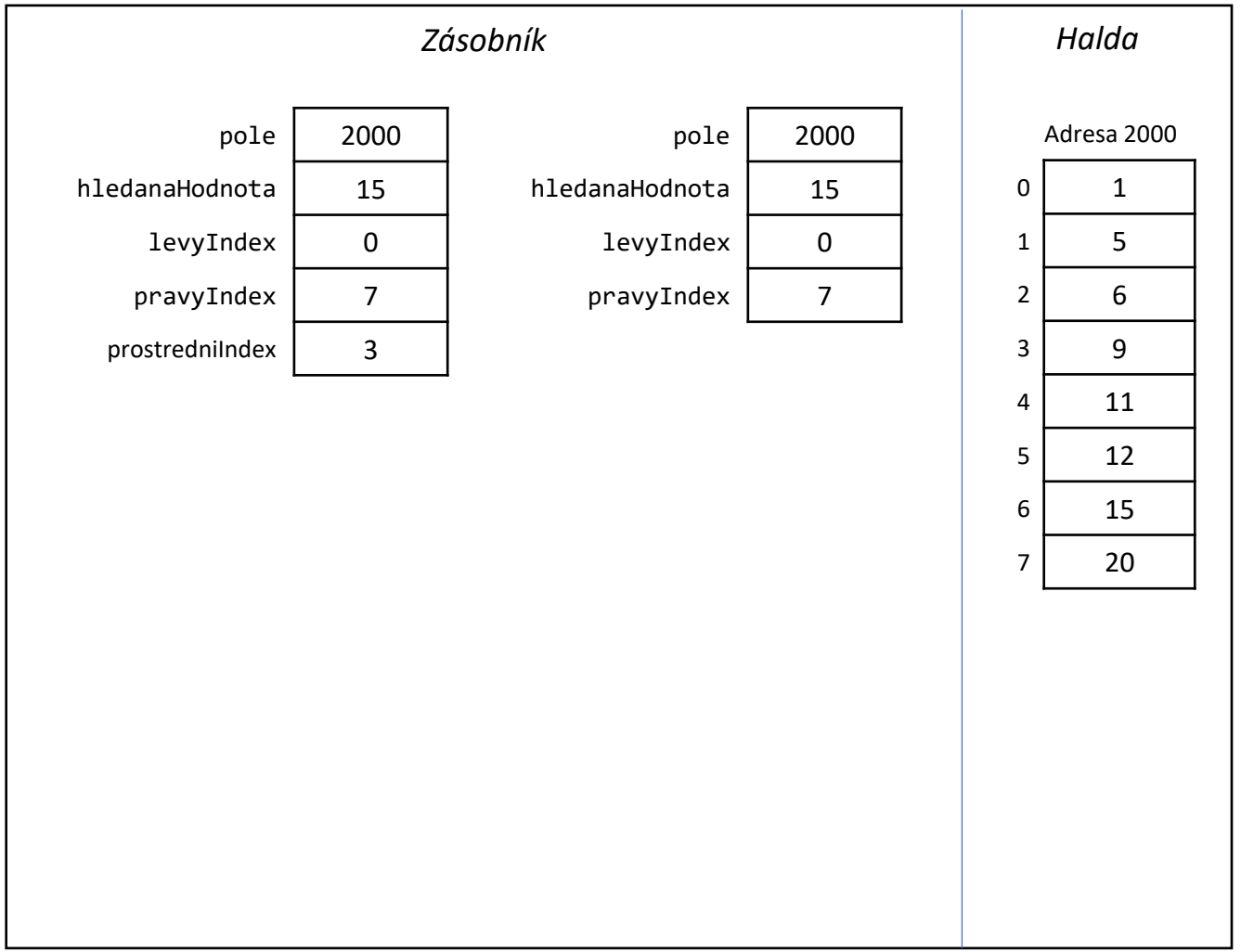
První volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

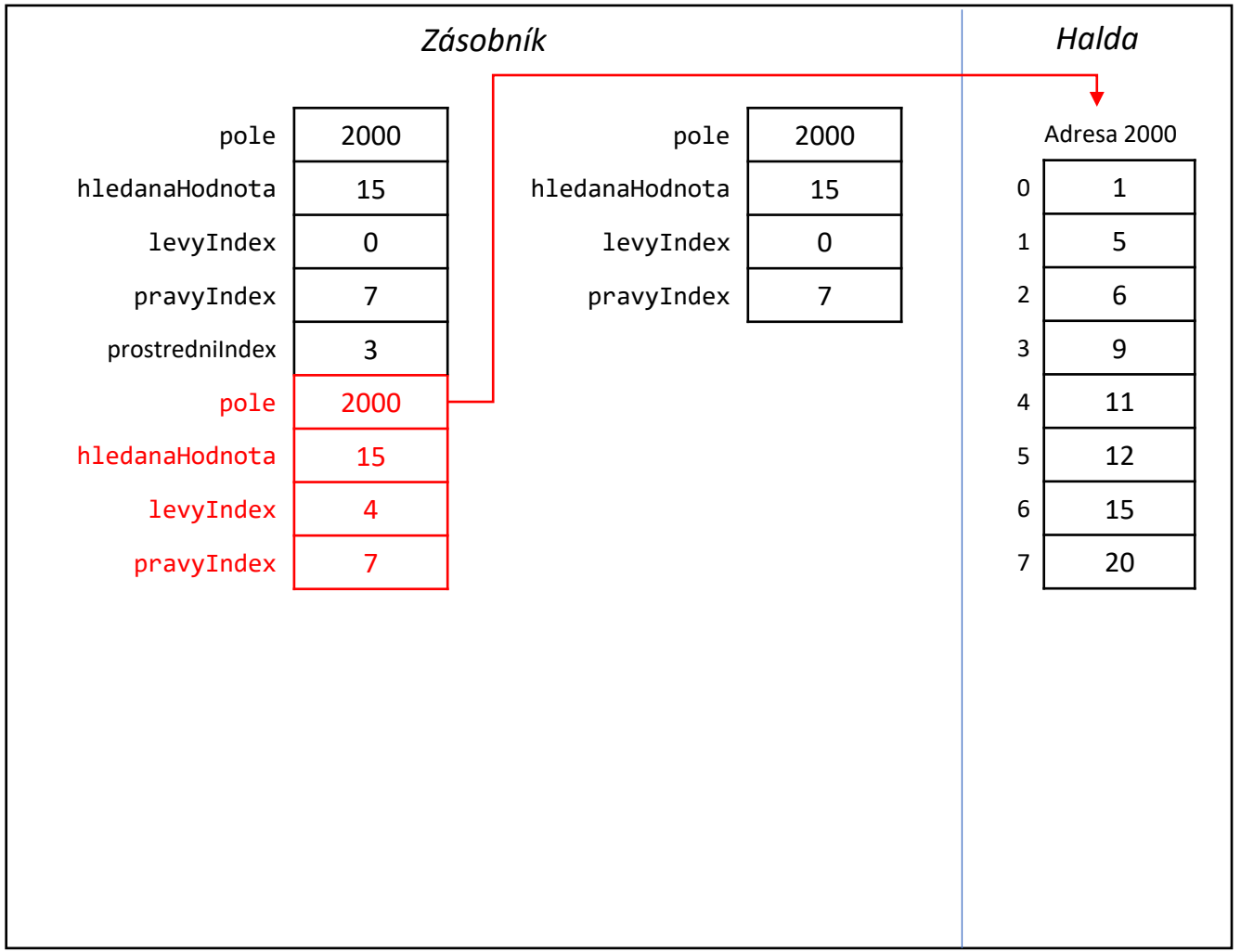
První volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

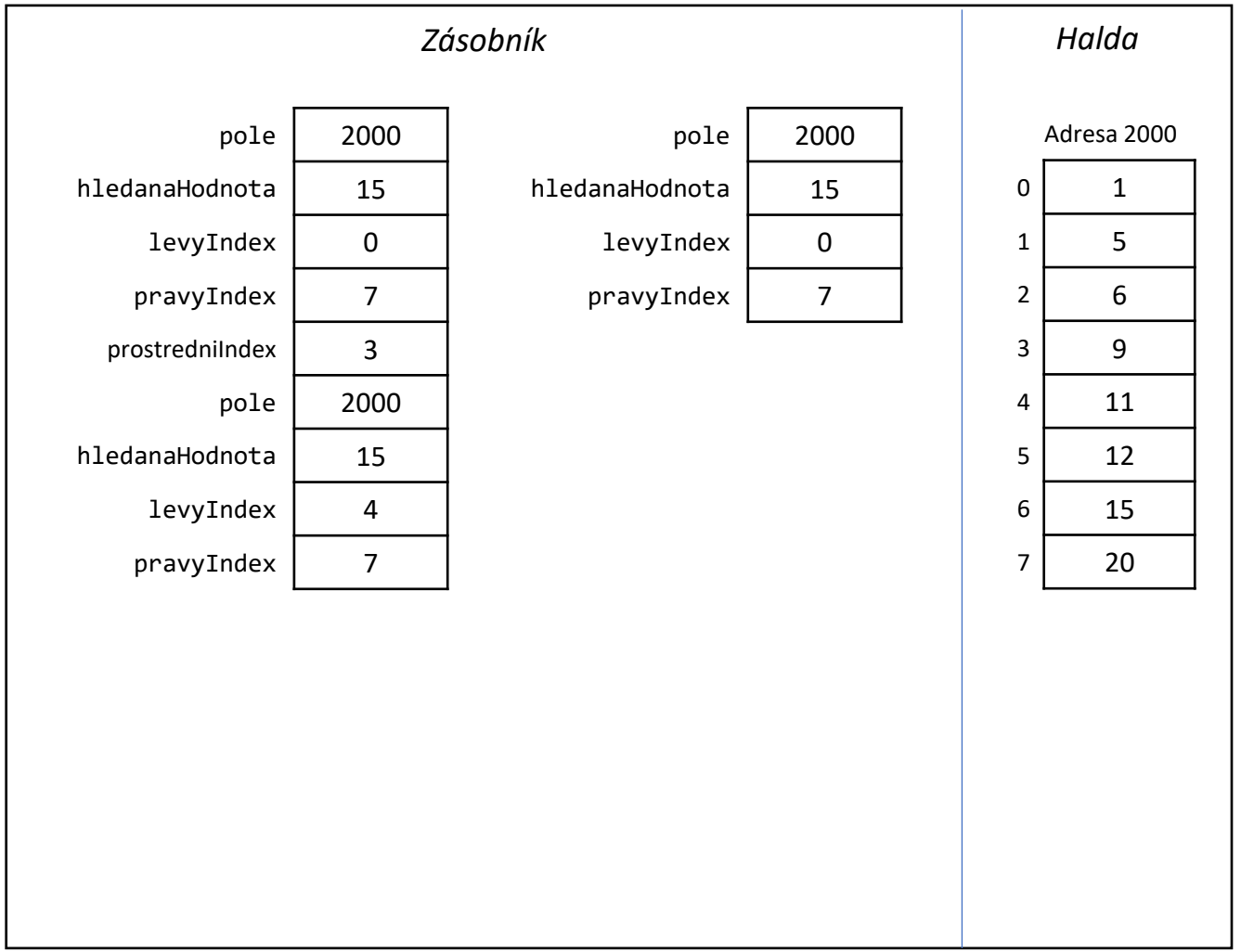
Druhé volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{           4           7
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

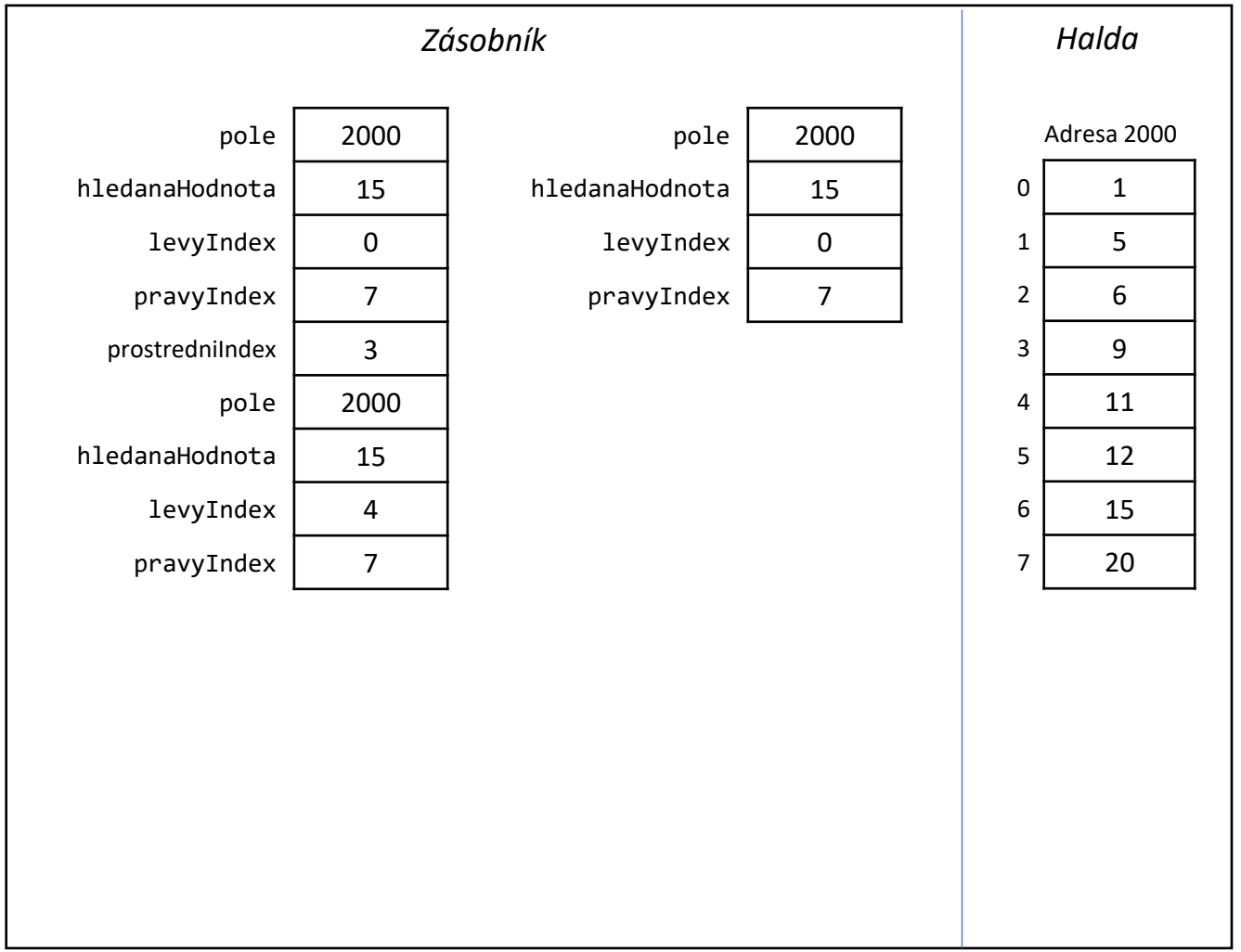
Druhé volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;
                                4           7
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

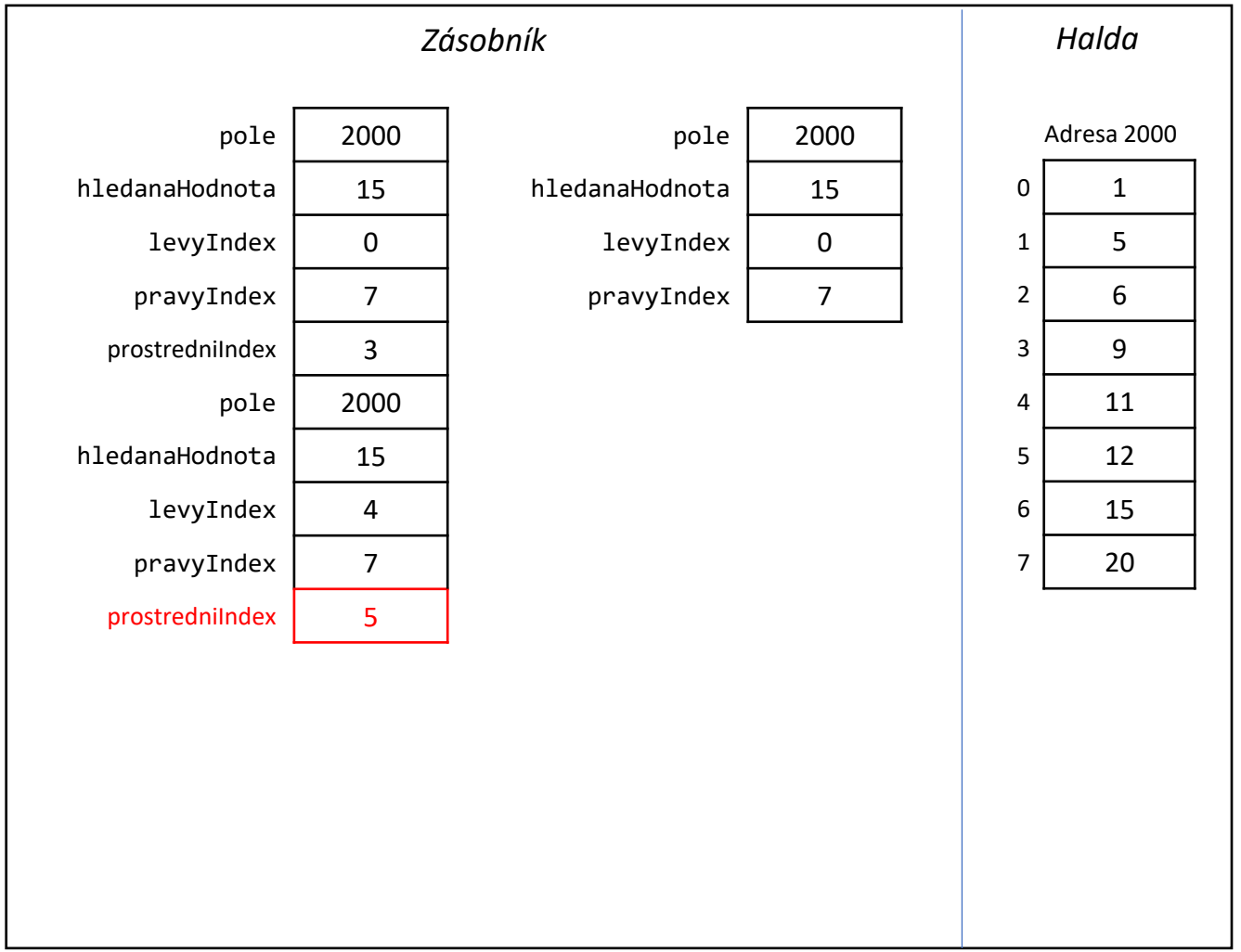
Druhé volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

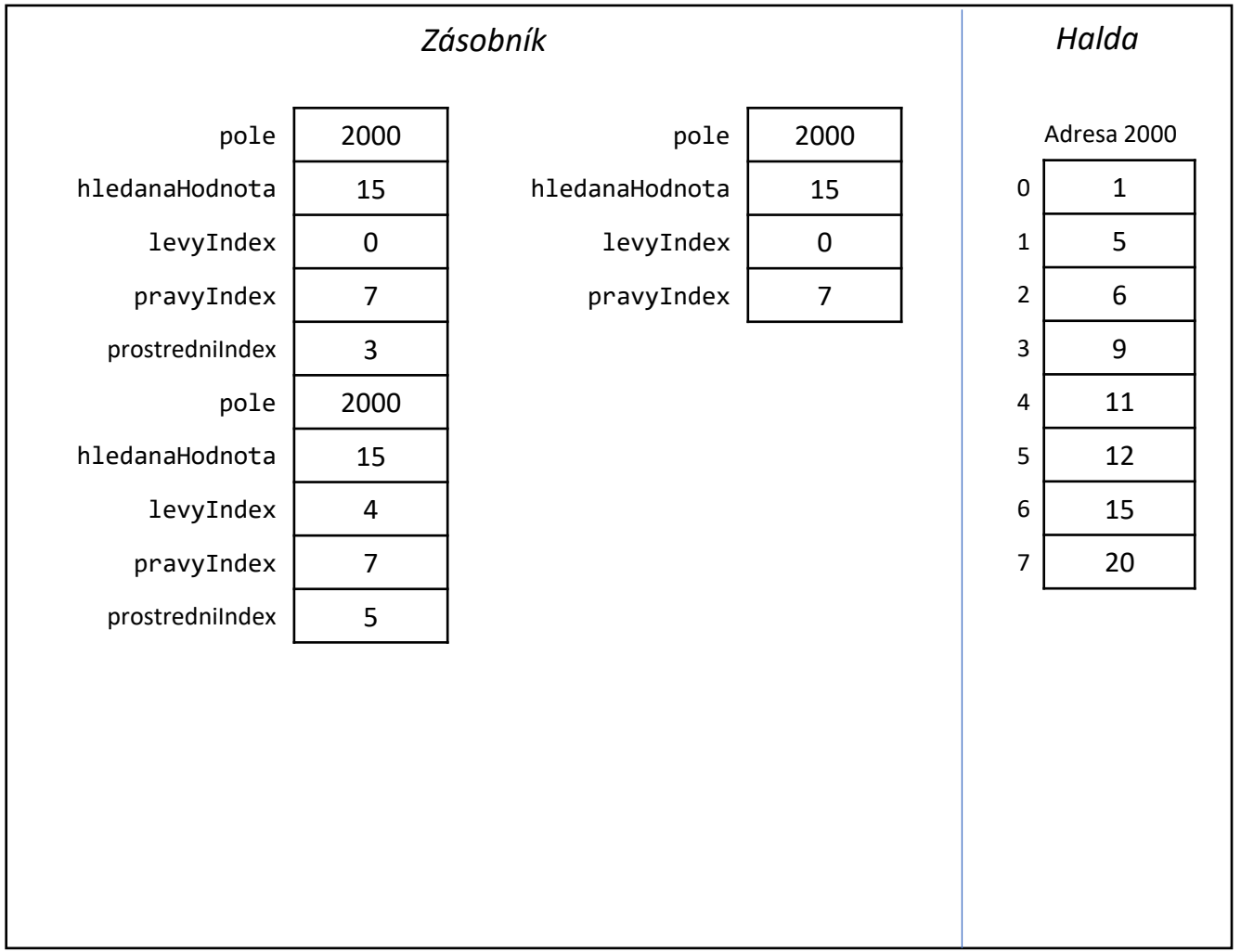
Druhé volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;
                  12                    15
    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

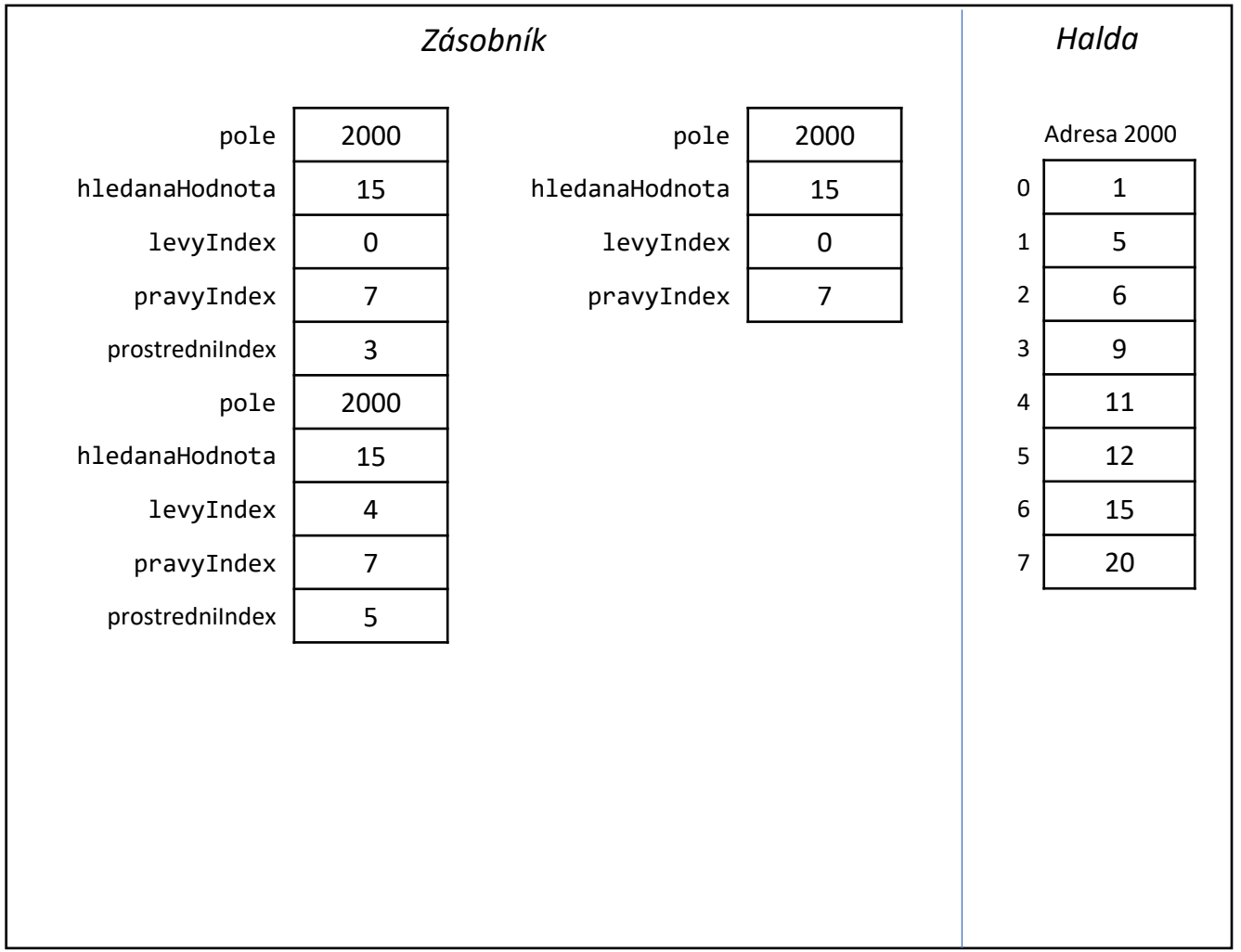
Druhé volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;
                    12                    15
    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

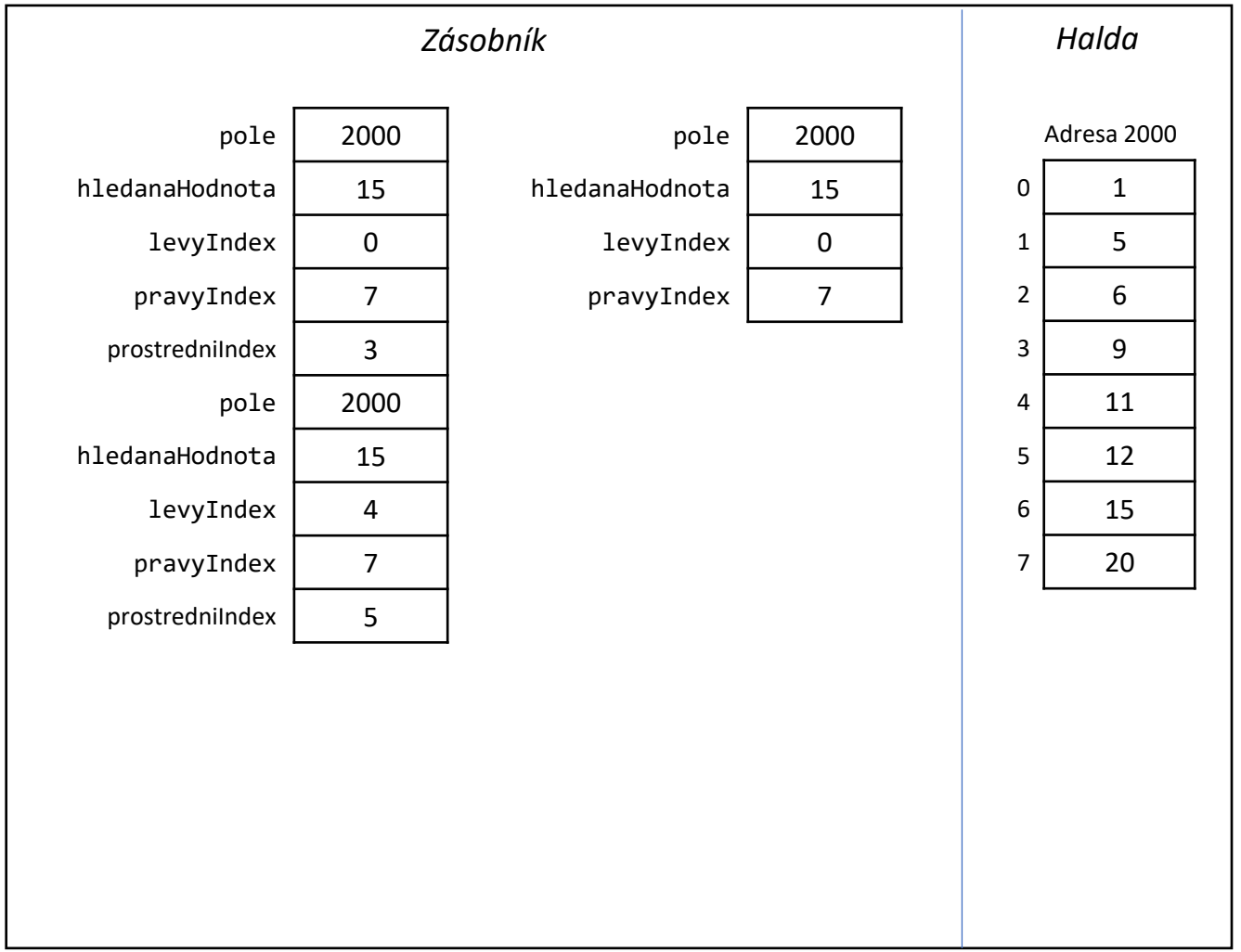
Druhé volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

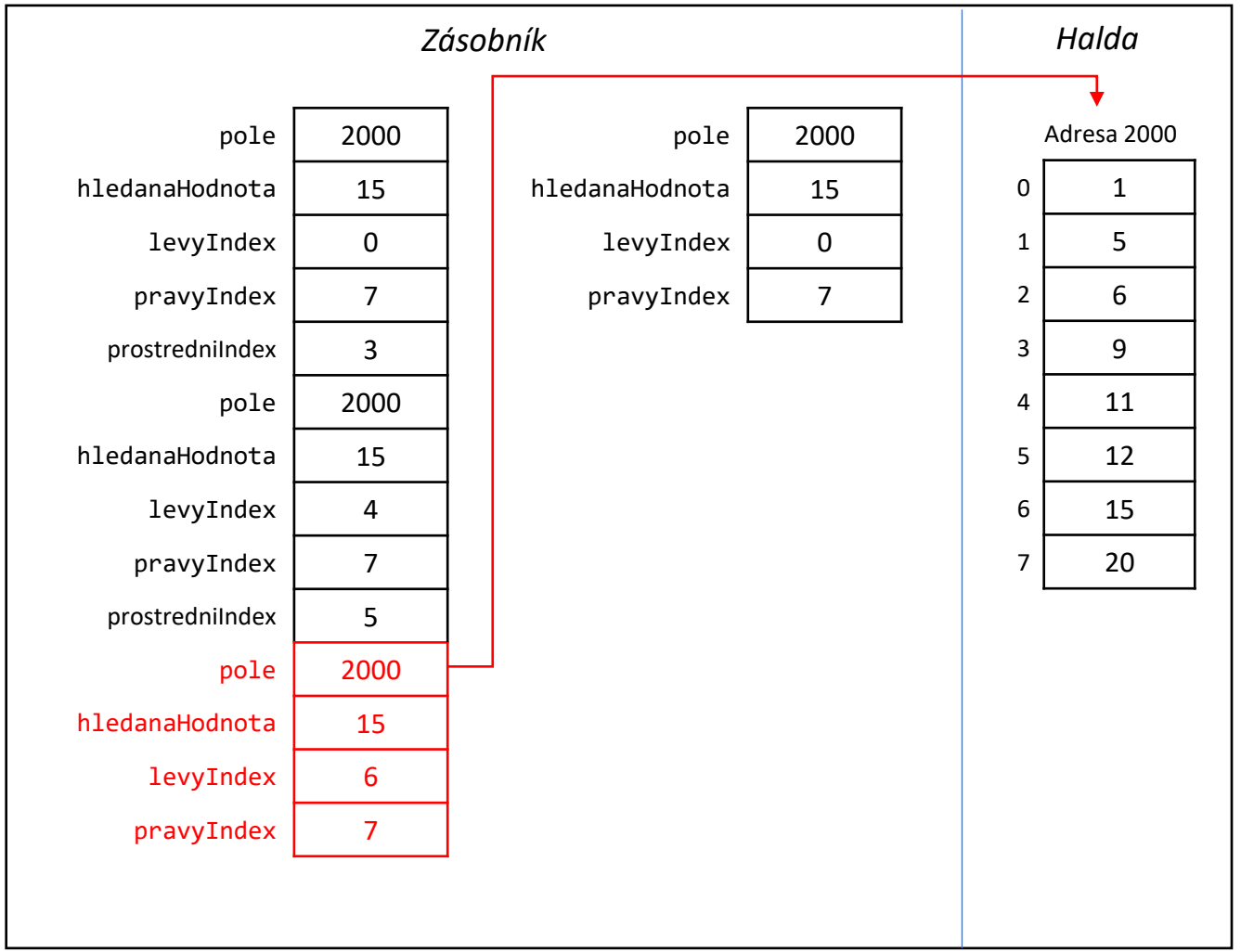
Druhé volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

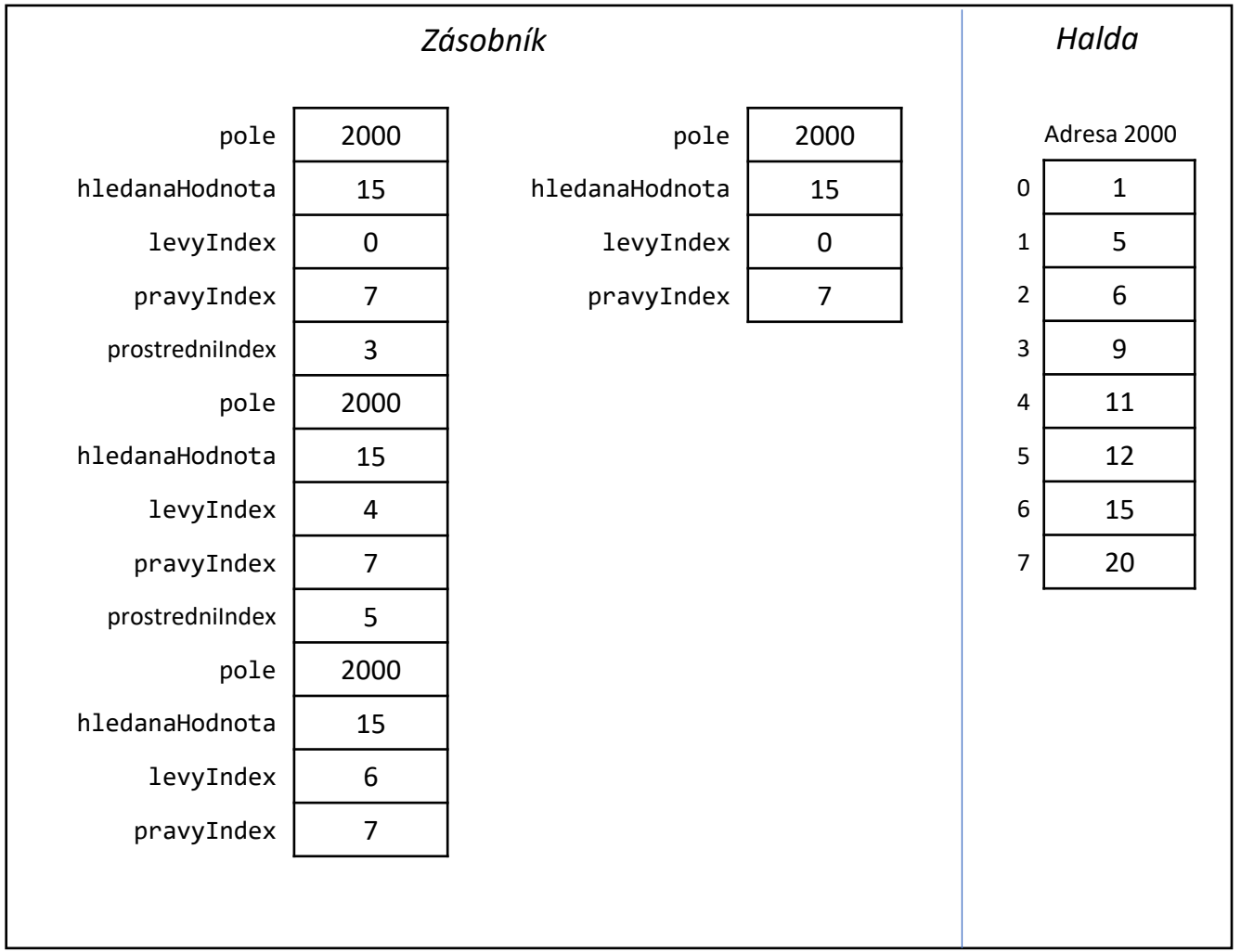
Třetí volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{           6           7
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

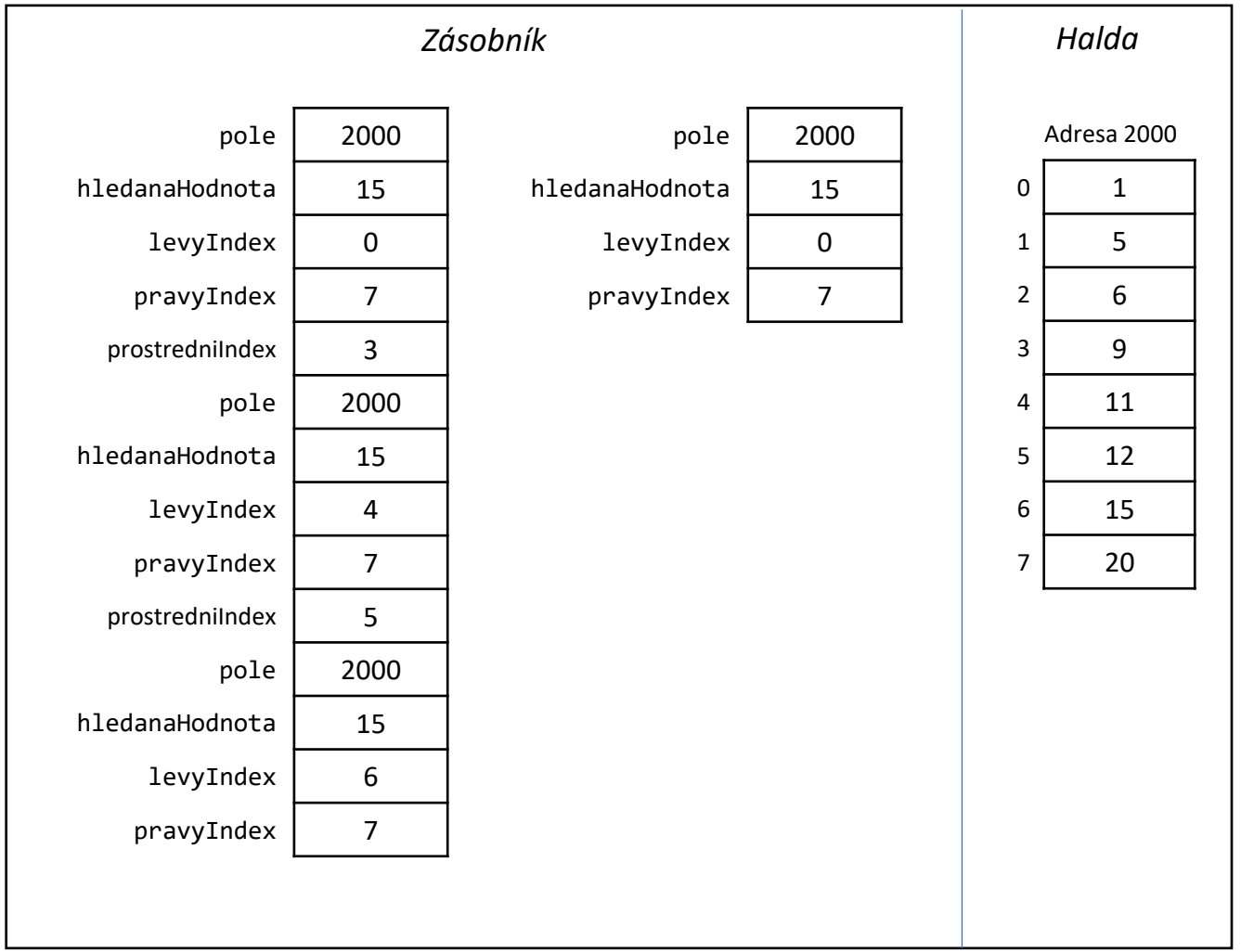
Třetí volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;
                                6           7
    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

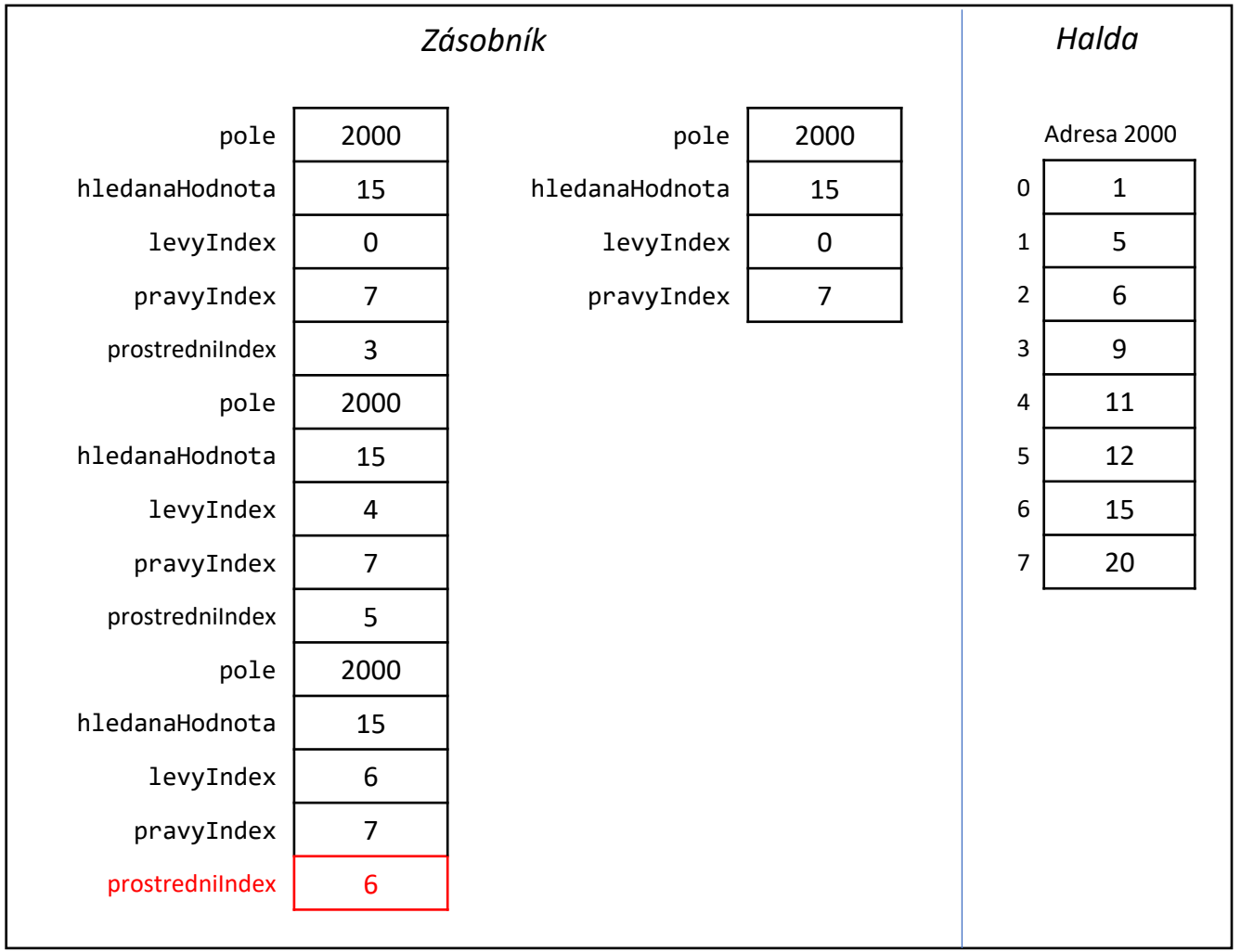
Třetí volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

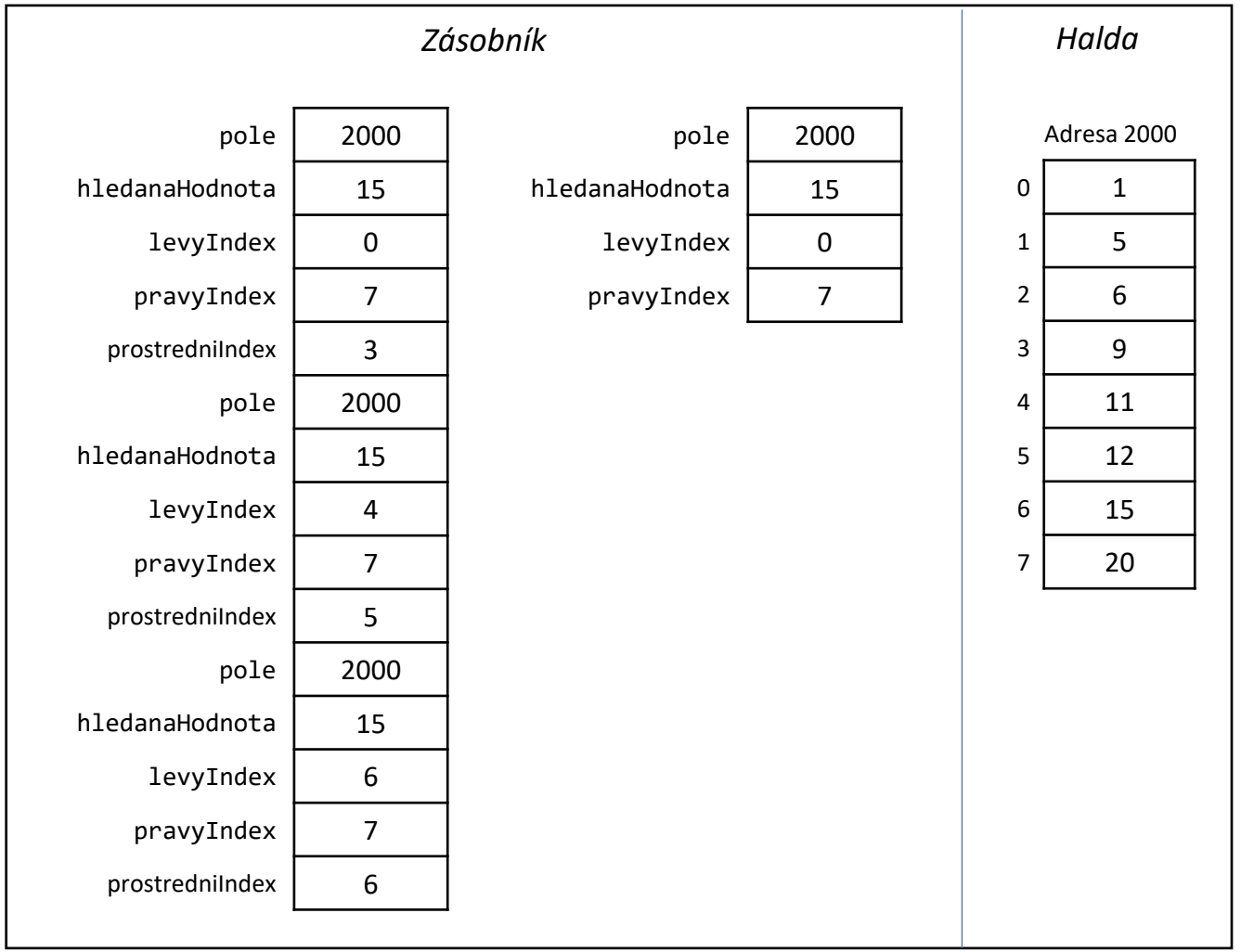
Třetí volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;
                  15                    15
    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

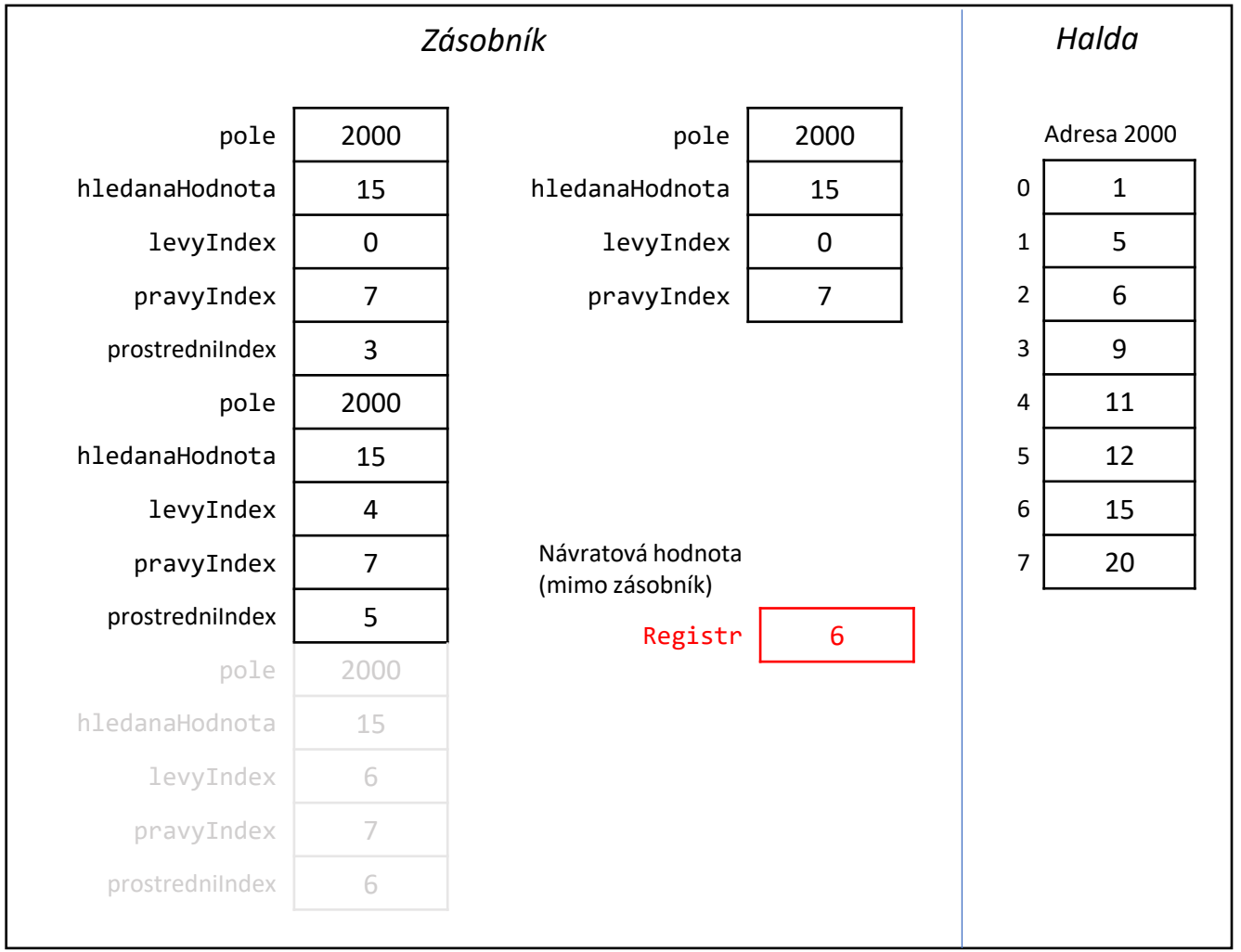
Třetí volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

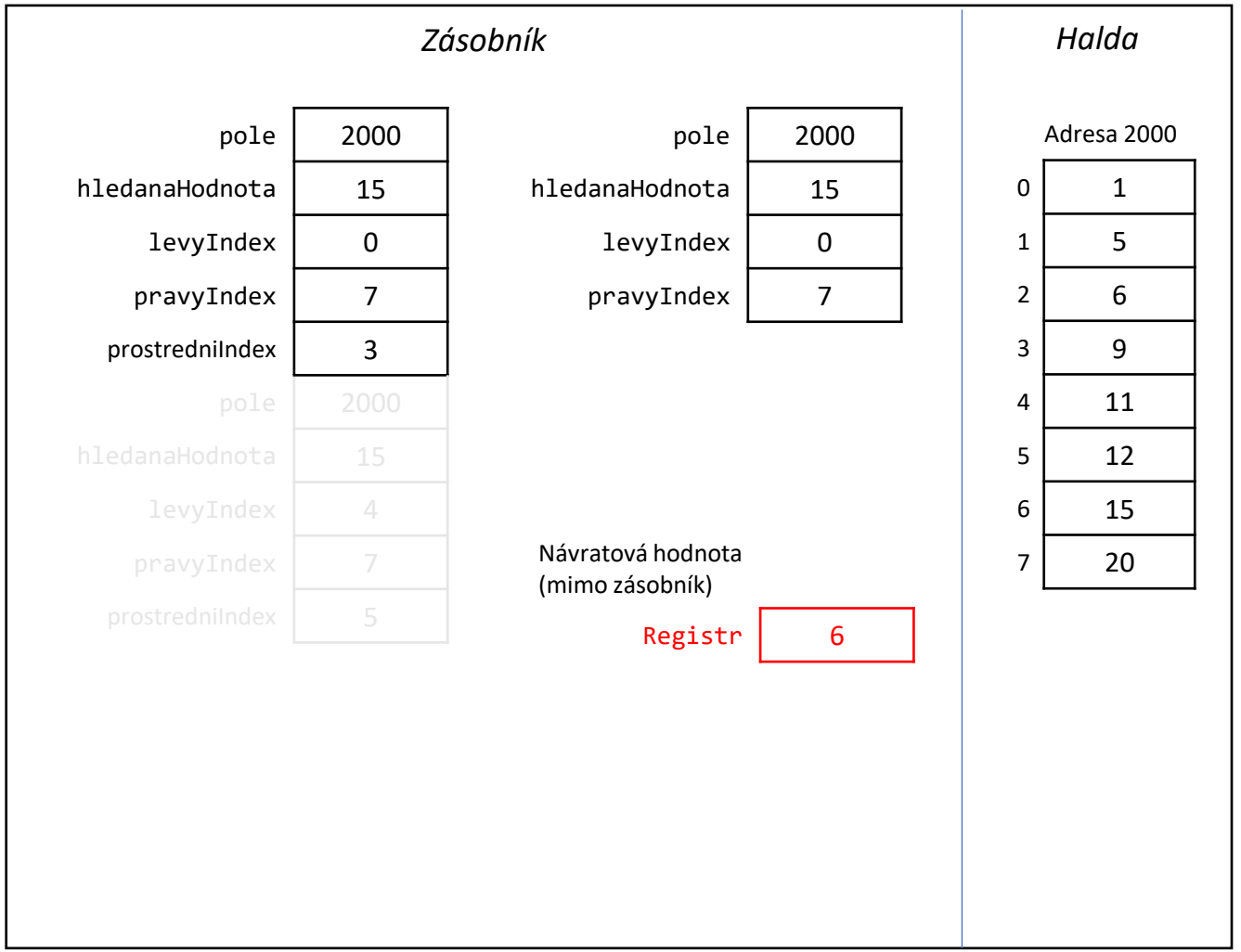
Třetí volání metody



Binary search - rekurzivní

```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

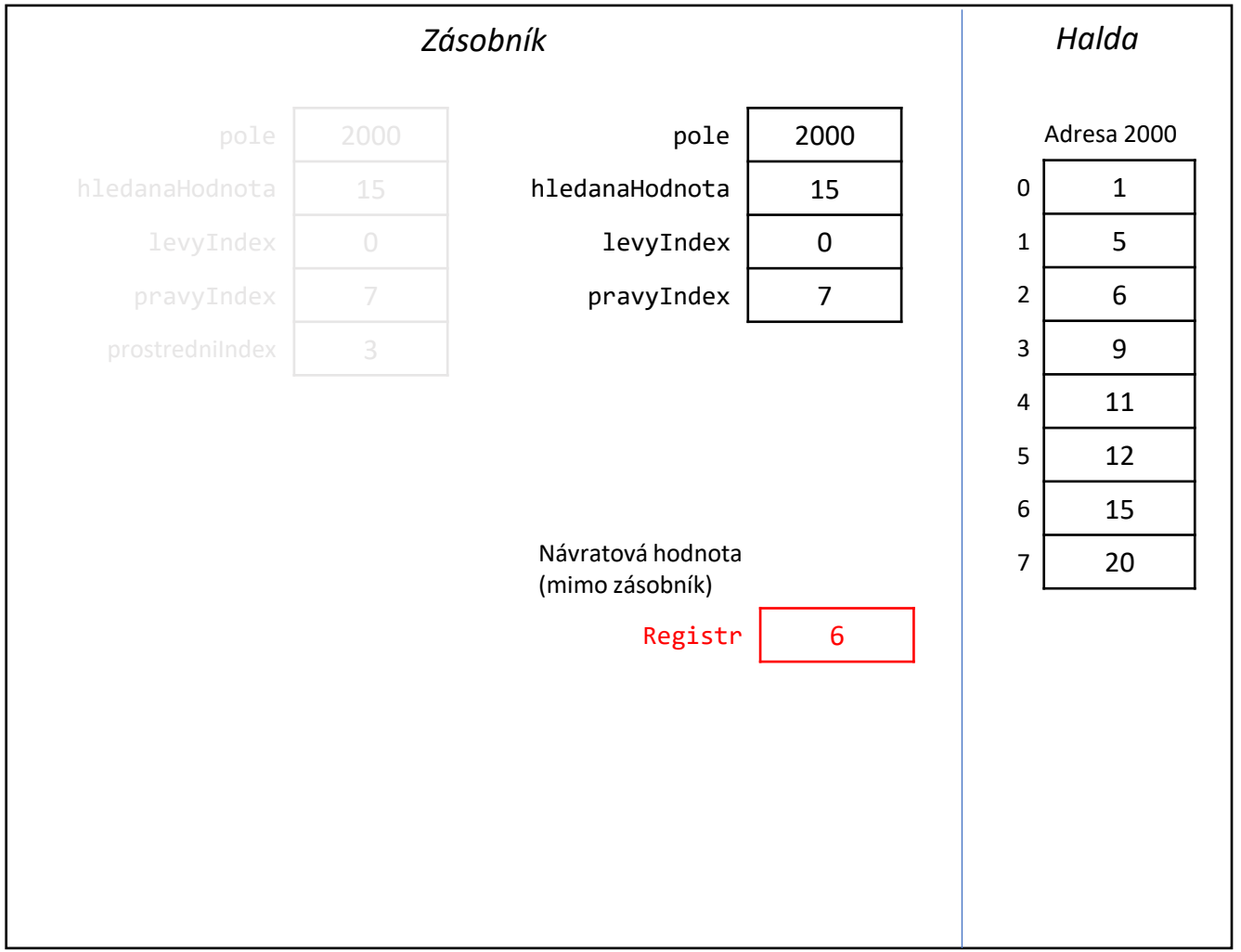
Druhé volání metody



Binary search - rekurzivní

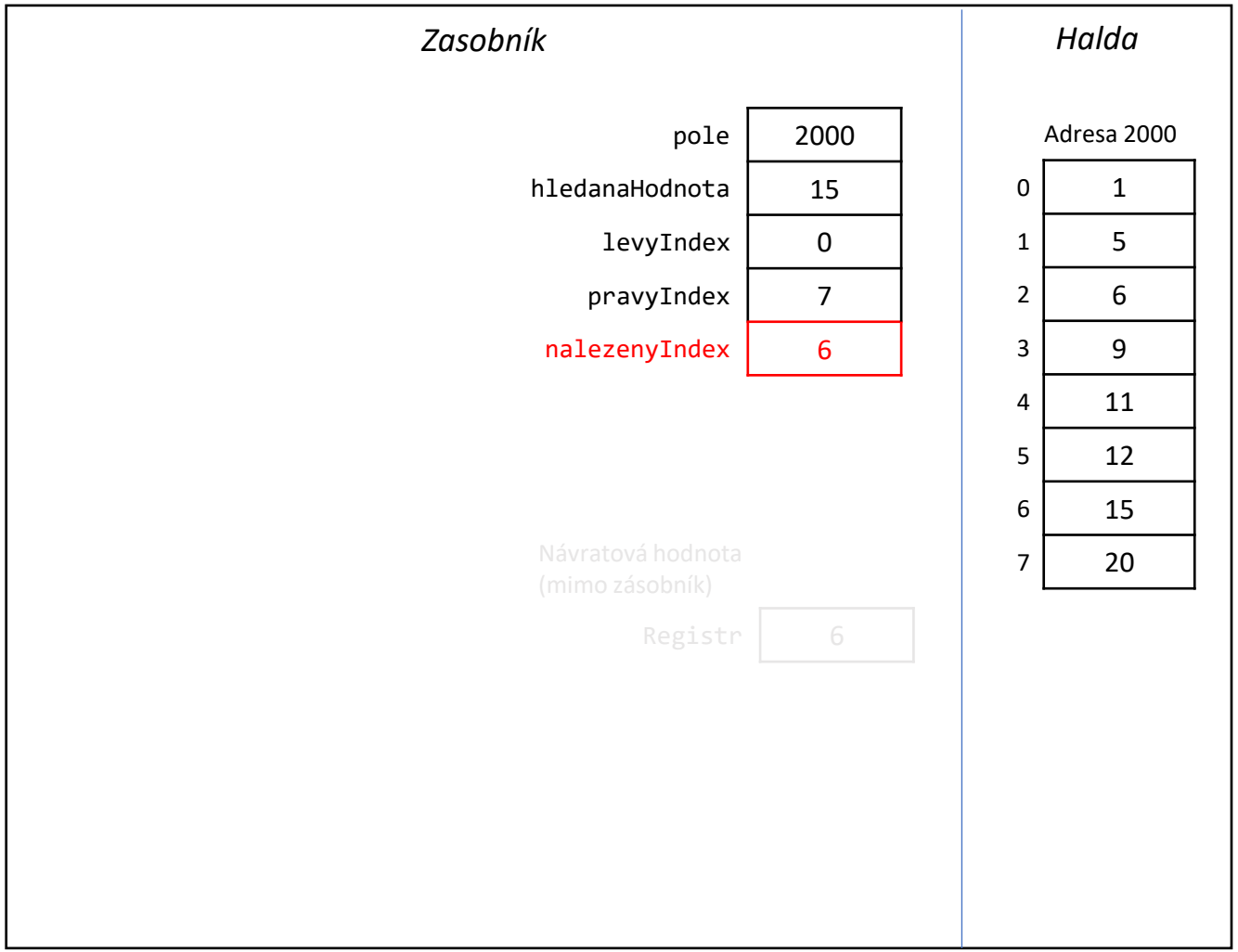
```
static int BinarySearch(int[] pole,
    int hledanaHodnota,
    int levyIndex,
    int pravyIndex)
{
    if (levyIndex == pravyIndex
        && pole[levyIndex] != hledanaHodnota)
        return -1;

    int prostredniIndex = (levyIndex + pravyIndex) / 2;

    if (pole[prostredniIndex] == hledanaHodnota)
        return prostredniIndex;

    else if (pole[prostredniIndex] > hledanaHodnota)
        return BinarySearch(pole,
            hledanaHodnota,
            levyIndex,
            prostredniIndex - 1,);

    else
        return BinarySearch(pole,
            hledanaHodnota,
            prostredniIndex + 1,
            pravyIndex);
}
```

První volání metody



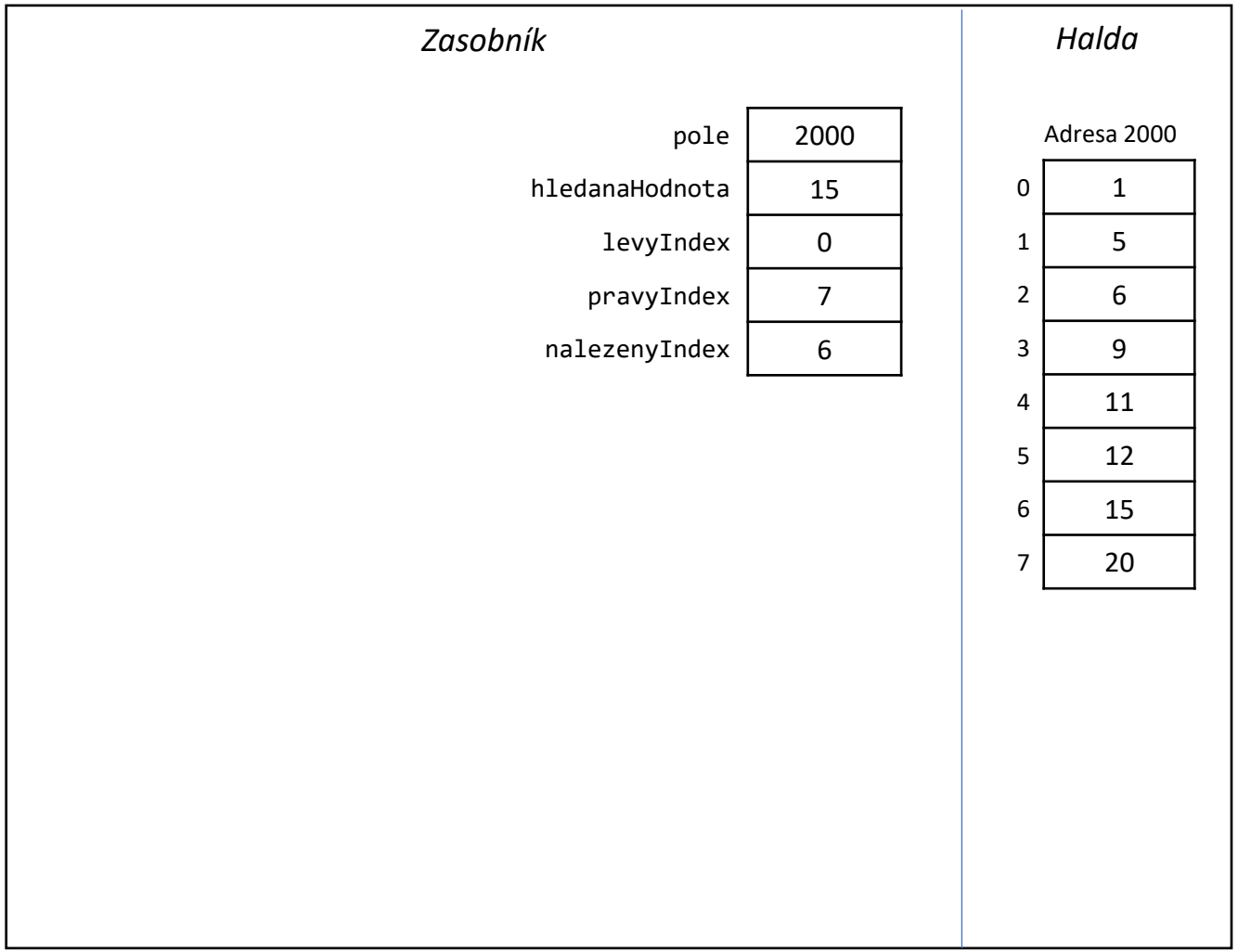
Binary search - rekurzivní

```
static void Main(string[] args)
{
    int[] pole = new int[]
    { 1, 5, 6, 9, 11, 12, 15, 20 };

    int hledanaHodnota = 15;

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

    int nalezenyIndex = BinarySearch(pole,
        hledanaHodnota,
        levyIndex,
        pravyIndex);




}
```



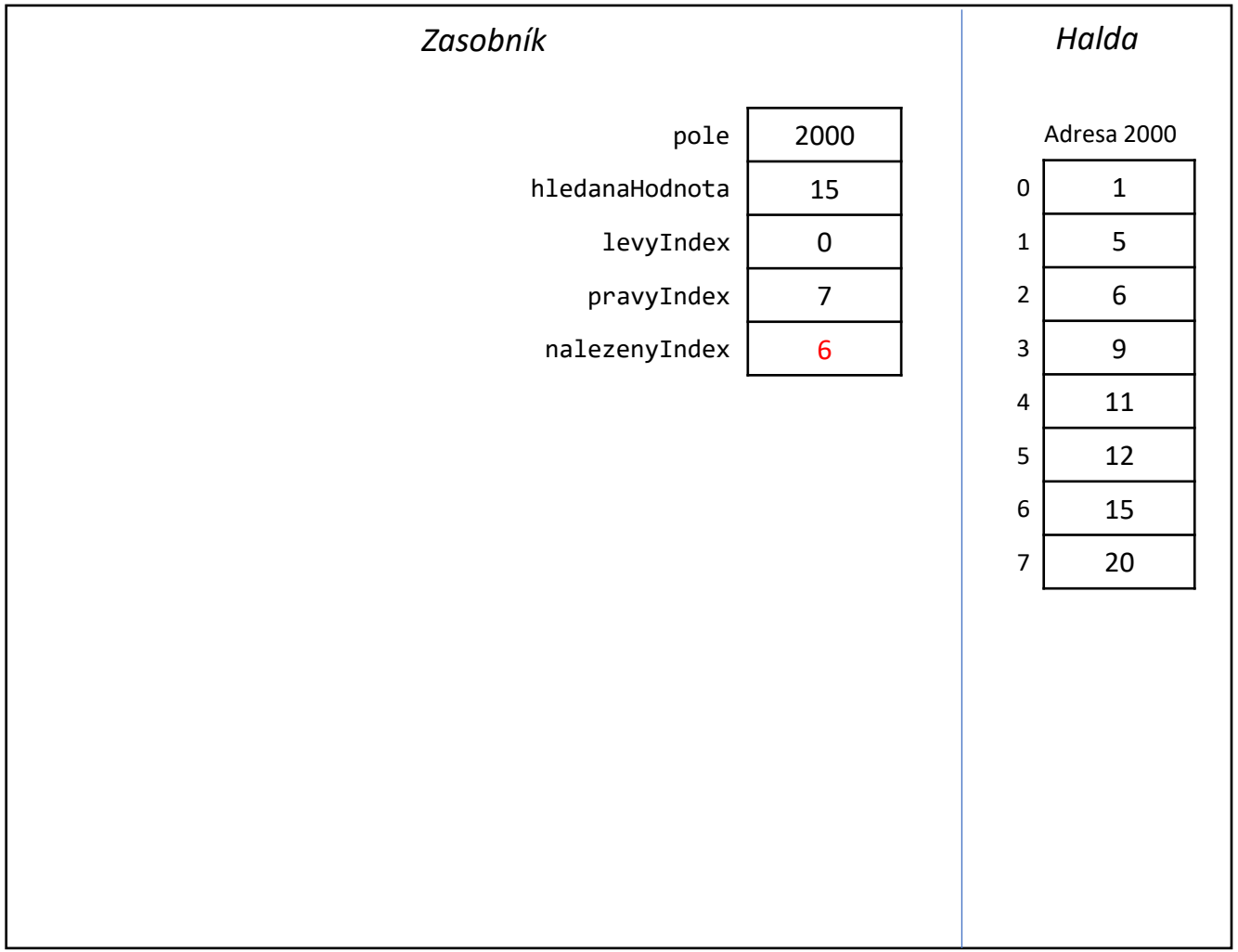
Binary search - rekurzivní

```
static void Main(string[] args)
{
    int[] pole = new int[]
    { 1, 5, 6, 9, 11, 12, 15, 20 };

    int hledanaHodnota = 15;

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

    int nalezenyIndex = BinarySearch(pole,
        hledanaHodnota,
        levyIndex,
        pravyIndex);
              6
    if (nalezenyIndex < 0)
    {
        Console.WriteLine($"Hodnota nenalezena");
    }
    else
    {
        Console.WriteLine(nalezenyIndex);
    }
}
```



Binary search - rekurzivní

```
static void Main(string[] args)
{
    int[] pole = new int[]
    { 1, 5, 6, 9, 11, 12, 15, 20 };

    int hledanaHodnota = 15;

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

    int nalezenyIndex = BinarySearch(pole,
        hledanaHodnota,
        levyIndex,
        pravyIndex);

    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

[2] Stanford Encyclopedia of Philosophy [online]. Copyright © 2020 [cit. 24.02.2021]. Dostupné z: https://plato.stanford.edu/entries/recursive-functions/

[3] x64 calling convention | Microsoft Docs. [online]. Copyright © Microsoft 2021 [cit. 23.02.2021]. Dostupné z: https://docs.microsoft.com/en-us/cpp/build/x64-calling-convention?view=msvc-160#return-values

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