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

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Programování a algoritmizace

Mocnina a faktoriál

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

Úvod

- V následujících snímcích probereme algoritmy výpočtu mocniny s přirozeným mocnitelem (exponentem).
- Na těchto příkladech si demonstrujeme použití rekurzivní metody.
- Jak si dále ukážeme, použití rekurzivní metody by pro tento algoritmus bylo méně efektivní než iterační varianta.

Mocnina

- Následující algoritmus spočítá hodnotu výrazu x^y
- Využijeme k tomu rekurzivního algoritmu, kdy rekurzivně voláme metodu tak dlouho dokud exponent není roven 0 a potom vrátíme hodnotu 1.

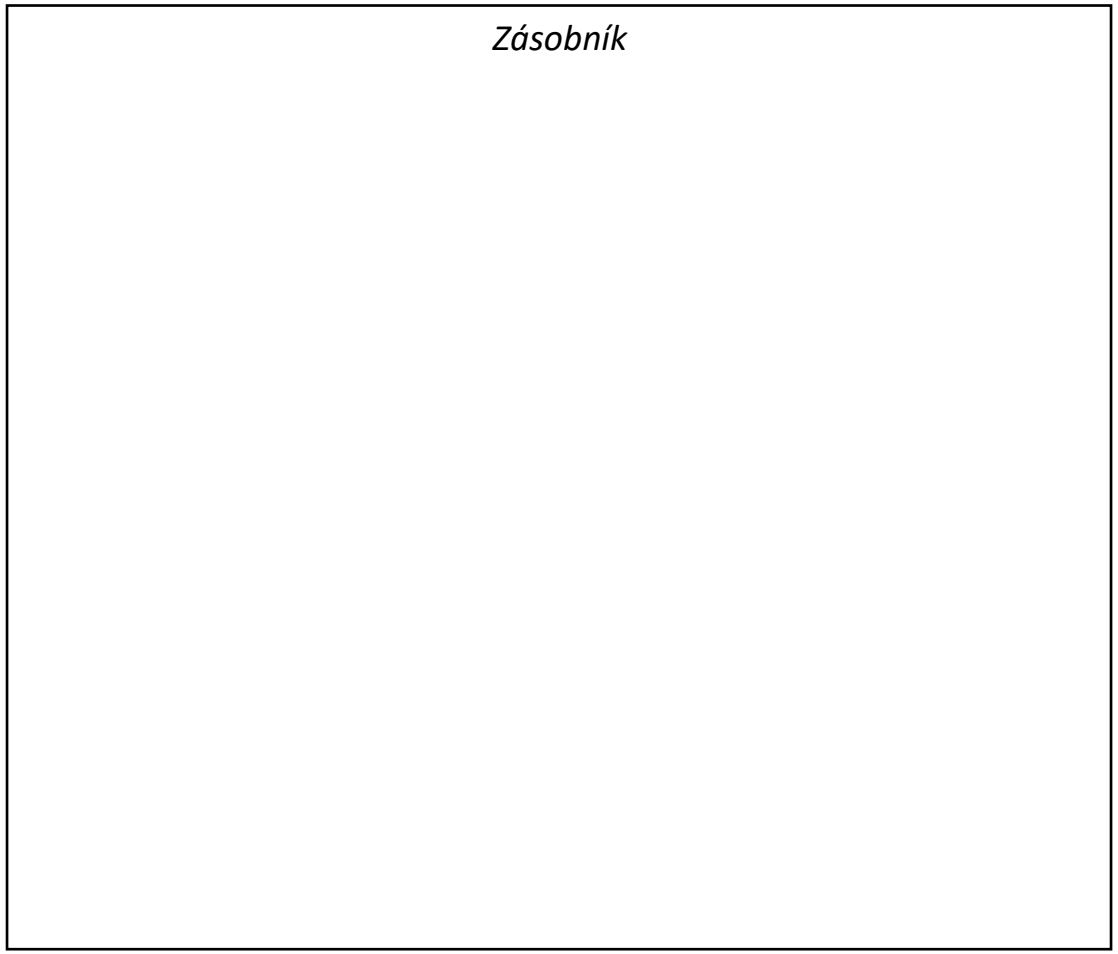
Algoritmus a paměť

- 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].

Mocnina - rekurzivní

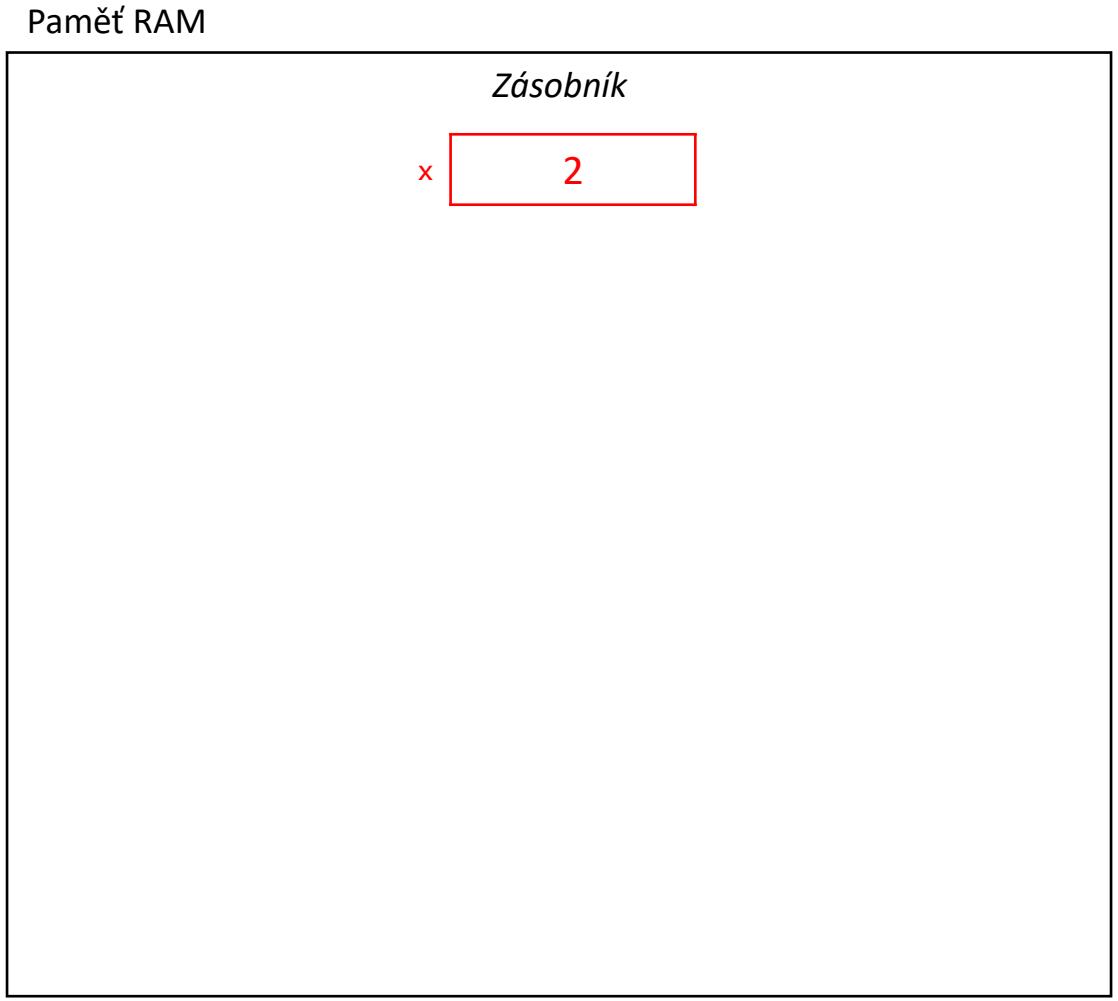
```
static int Mocnina(int x, int y)
{
    if (y == 0)
        return 1;
    else
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);

    Console.WriteLine(mocnina);
}
```

Paměť RAM



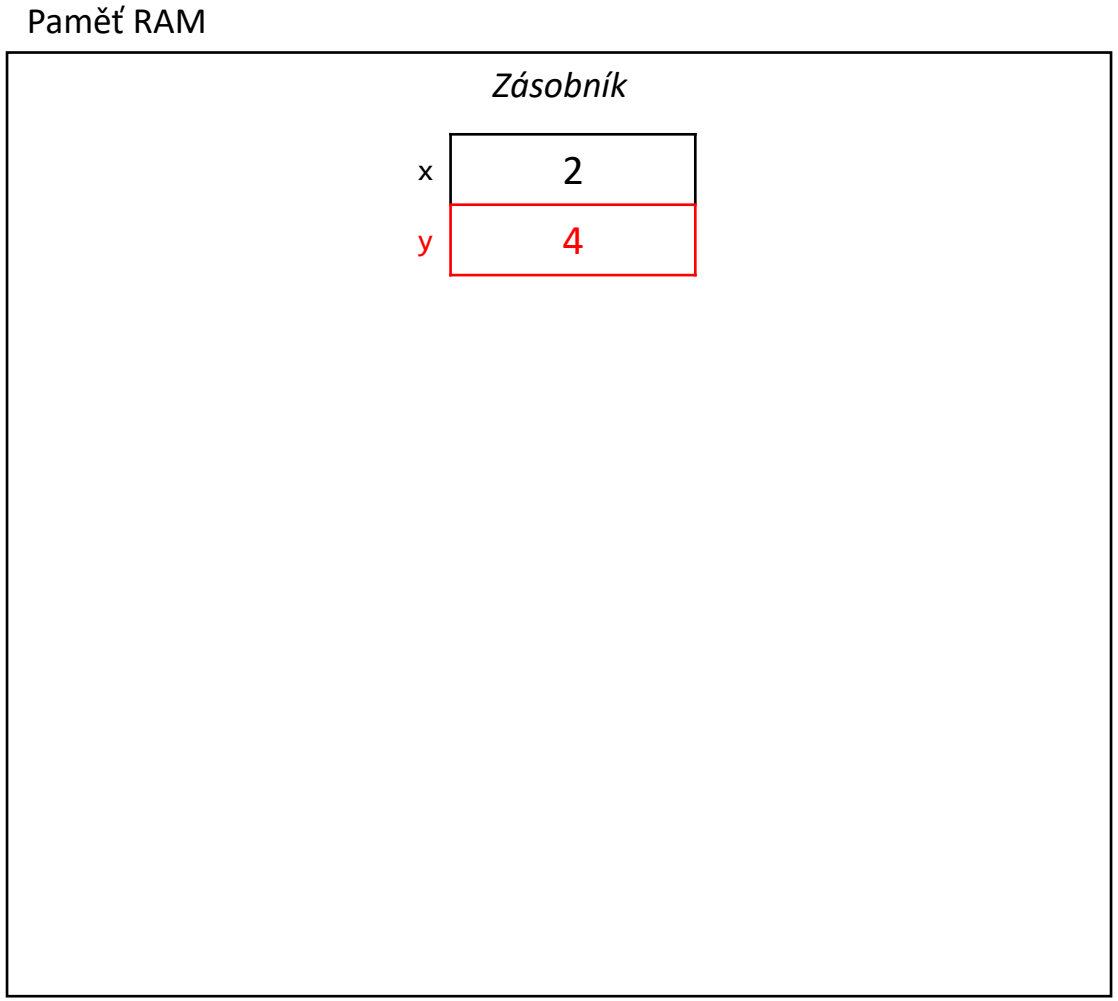
Mocnina - rekurzivní

```
static void Main(string[] args)
{
    int x = 2;



}
```



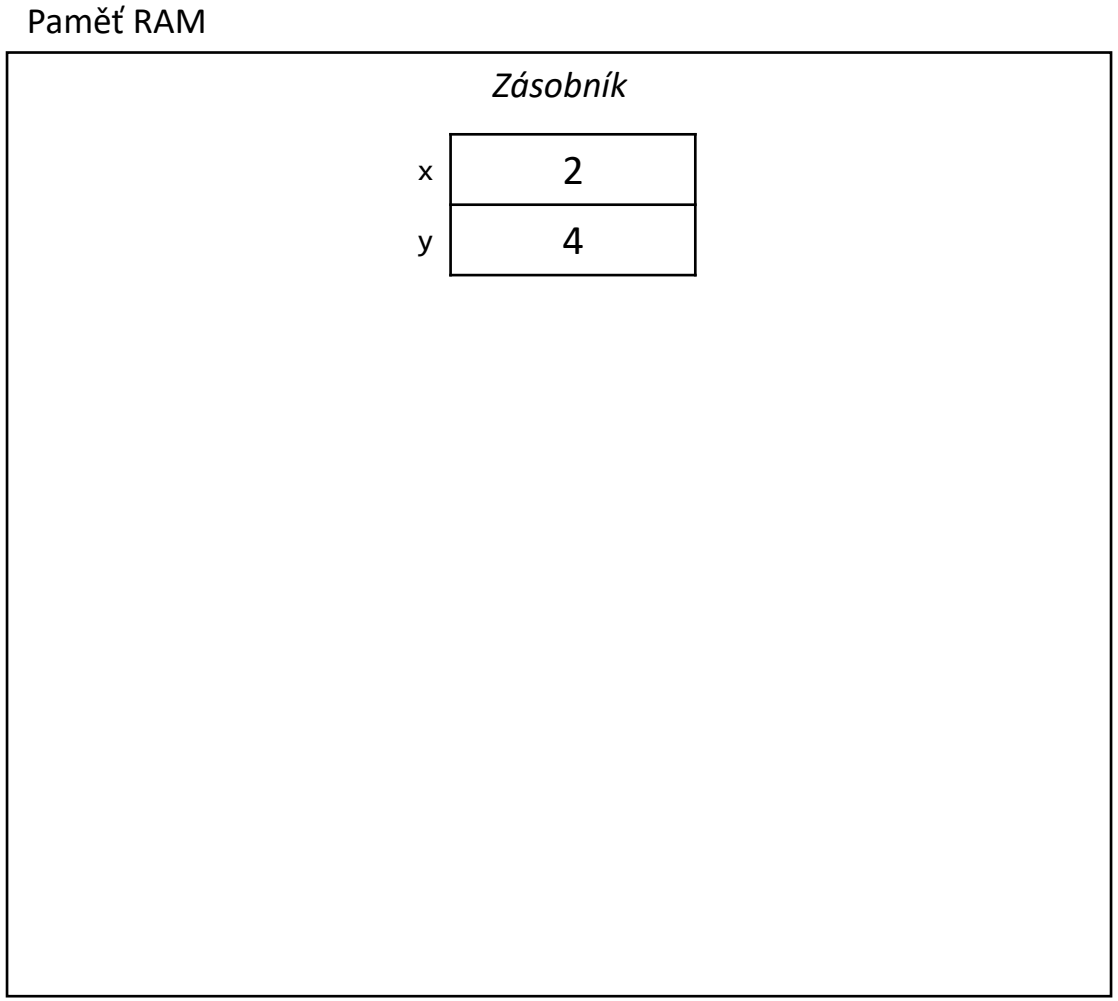
Mocnina - rekurzivní

```
static void Main(string[] args)
{
    int x = 2;
    int y = 4;



}
```



Mocnina - rekurzivní

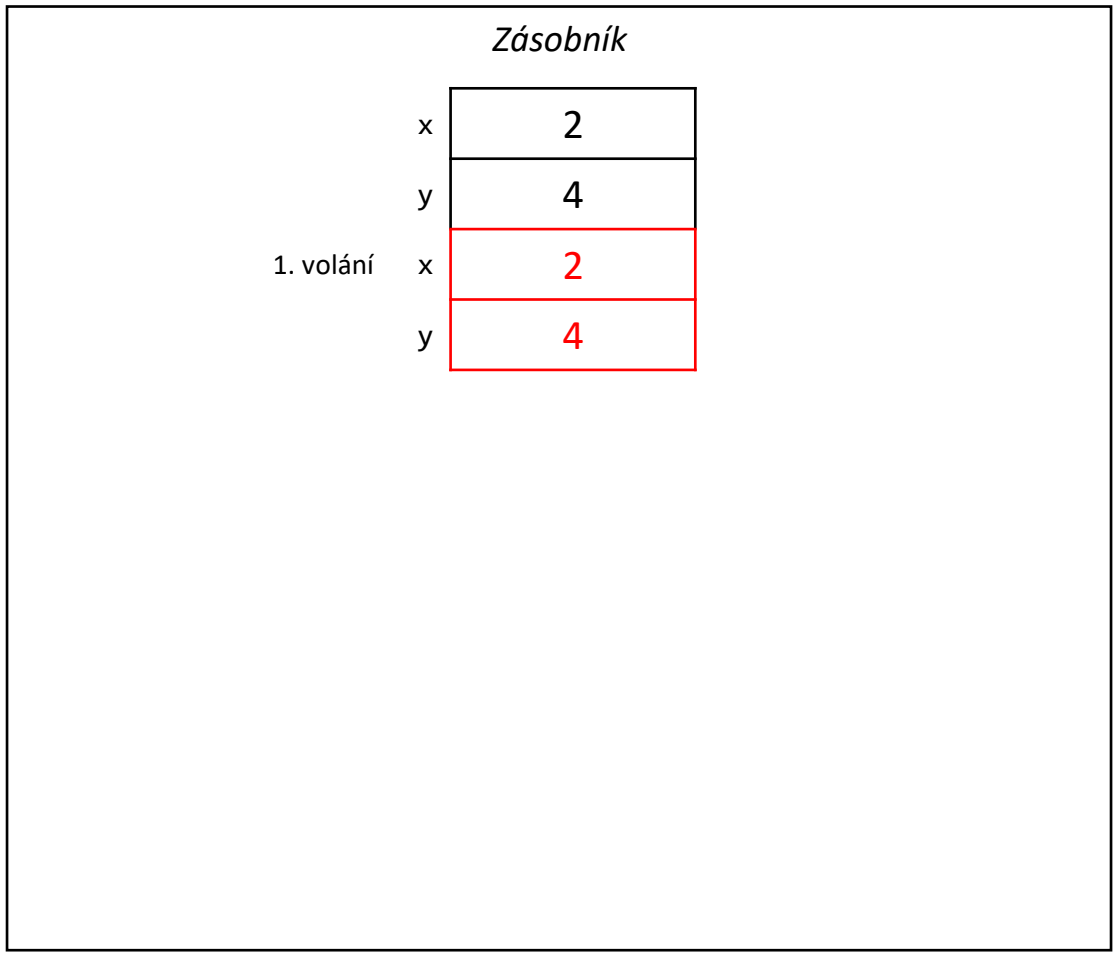
```
static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```



Mocnina - rekurzivní

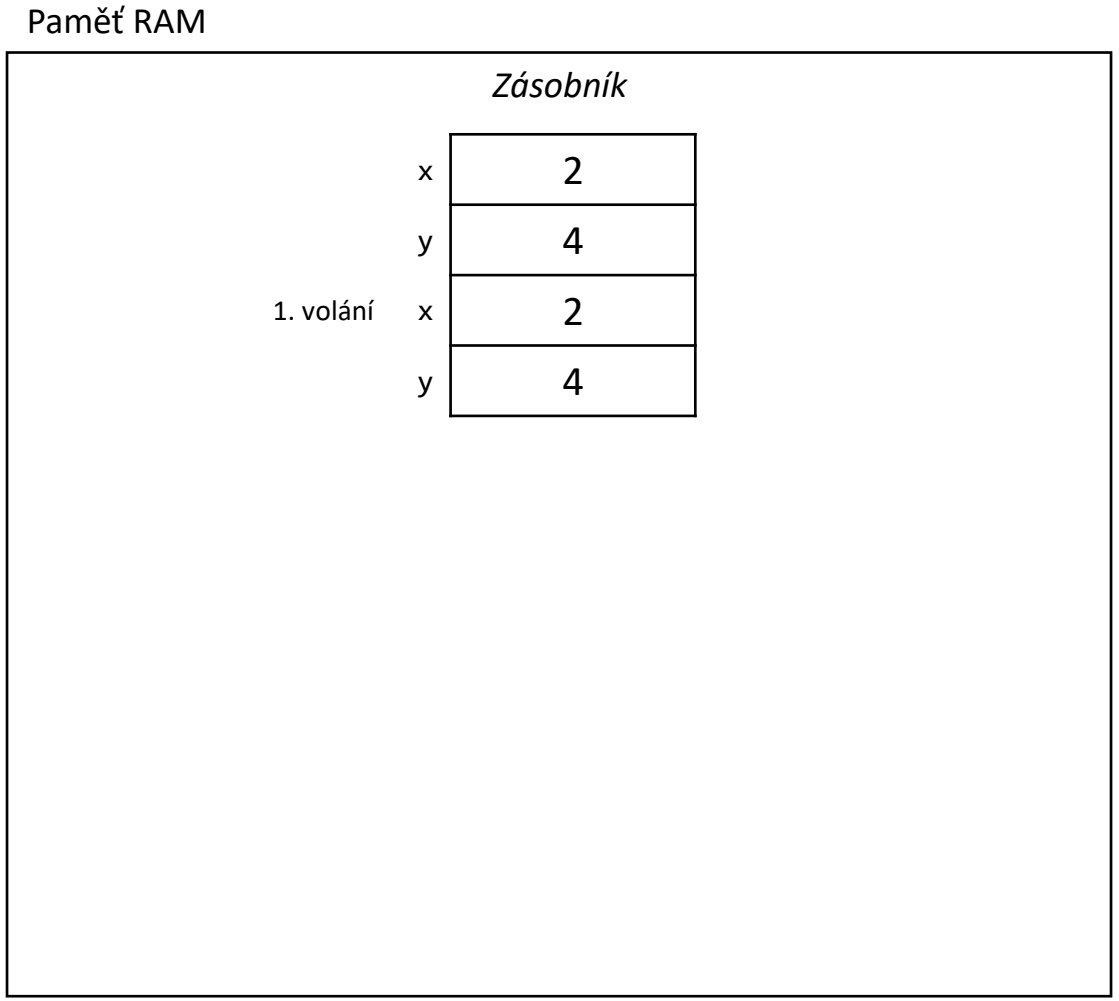
```
static int Mocnina(int x, int y)
{




}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```

Paměť RAM



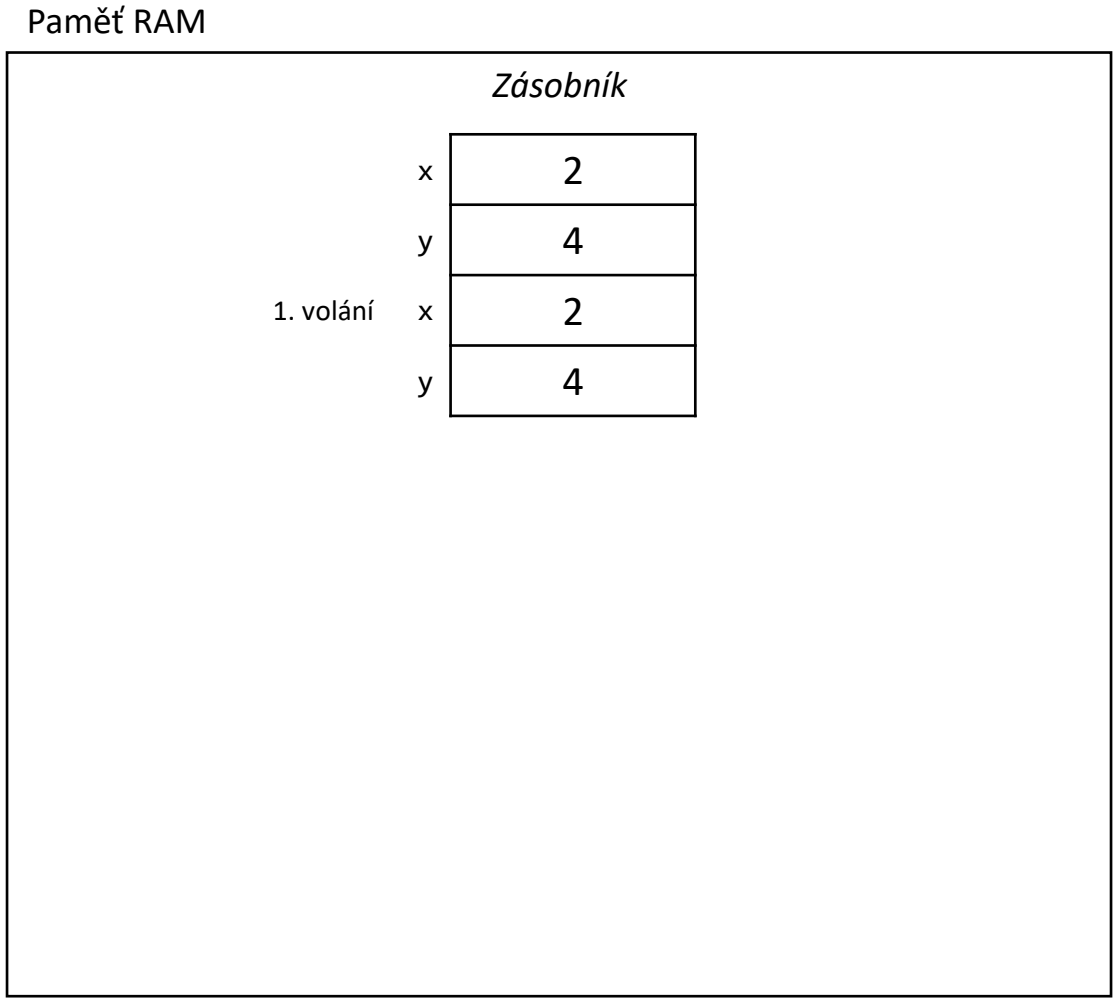
Mocnina - rekurzivní

```
static int Mocnina(int x, int y)
{       4
    if (y == 0)
        return 1;
    else
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```



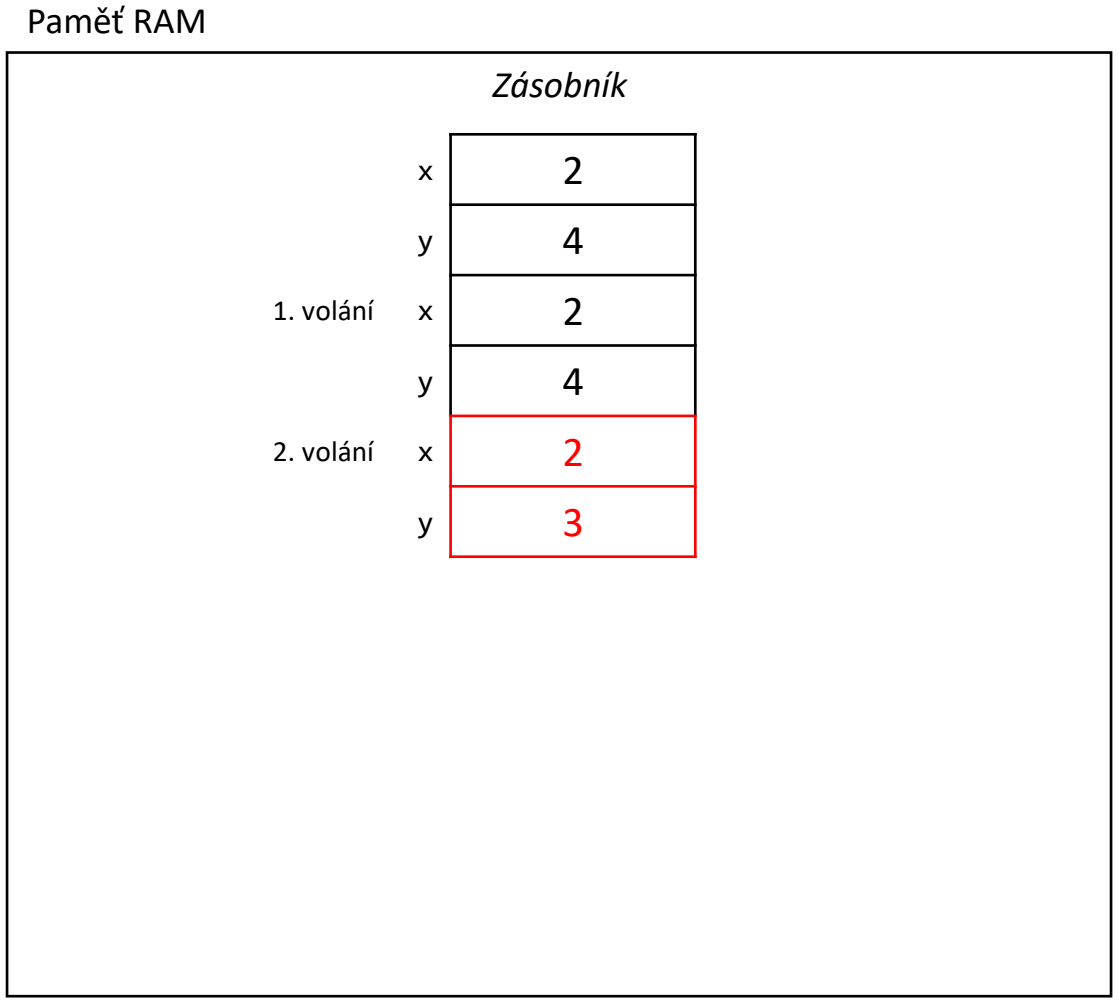
Mocnina - rekurzivní

```
static int Mocnina(int x, int y)
{
    if (y == 0)
        return 1;
    else                        2    3
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```



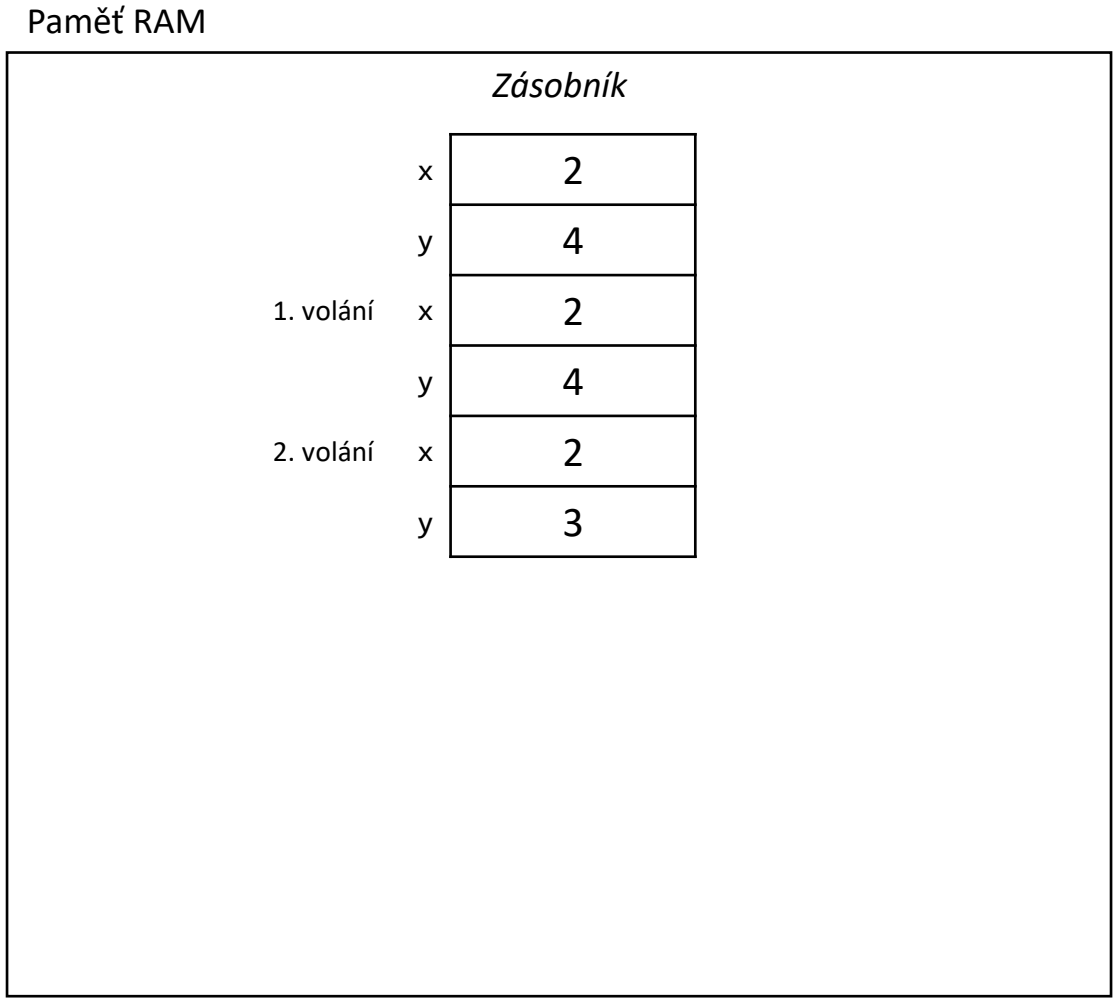
Mocnina - rekurzivní

```
static int Mocnina(int x, int y)
{
    if (y == 0)
        return 1;
    else
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```



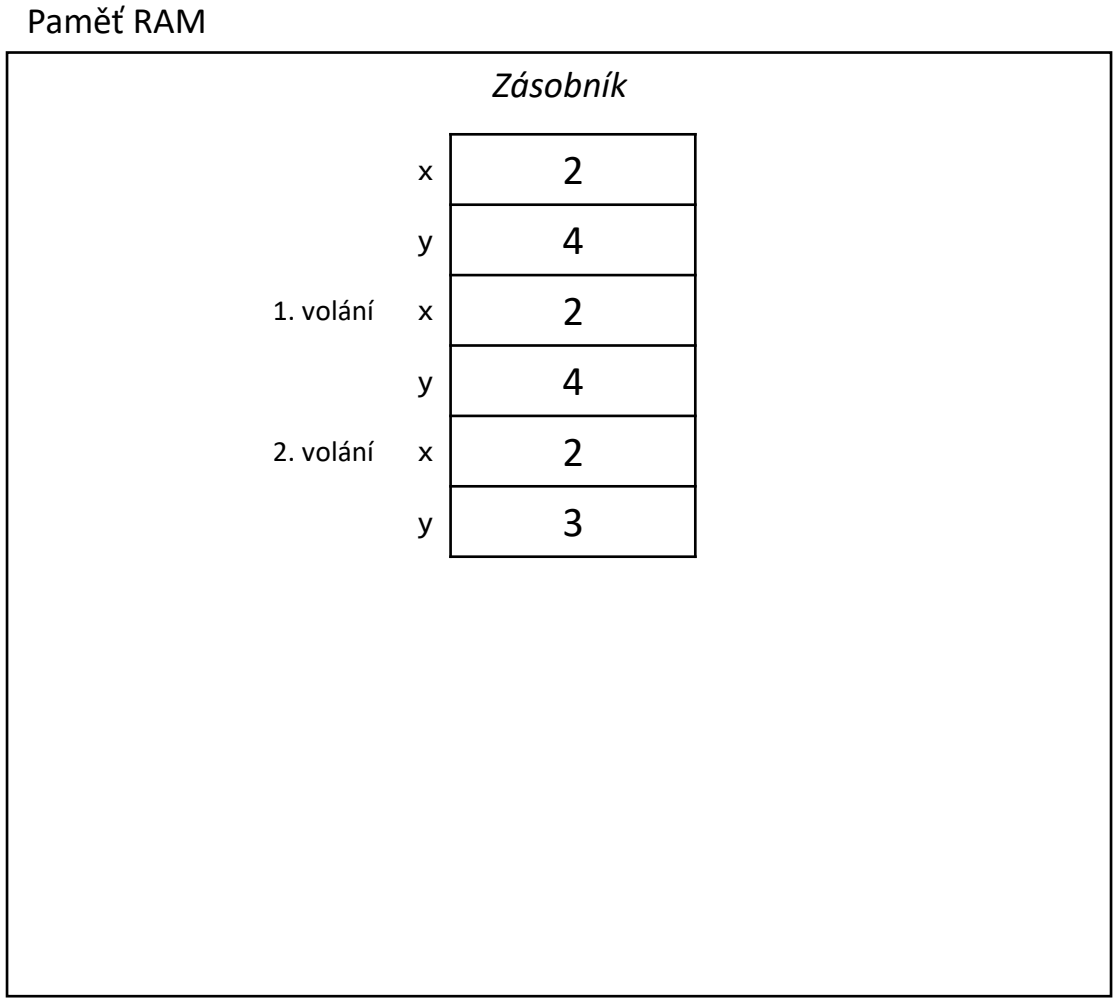
Mocnina - rekurzivní

```
static int Mocnina(int x, int y)
{       3
    if (y == 0)
        return 1;
    else
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```



Mocnina - rekurzivní

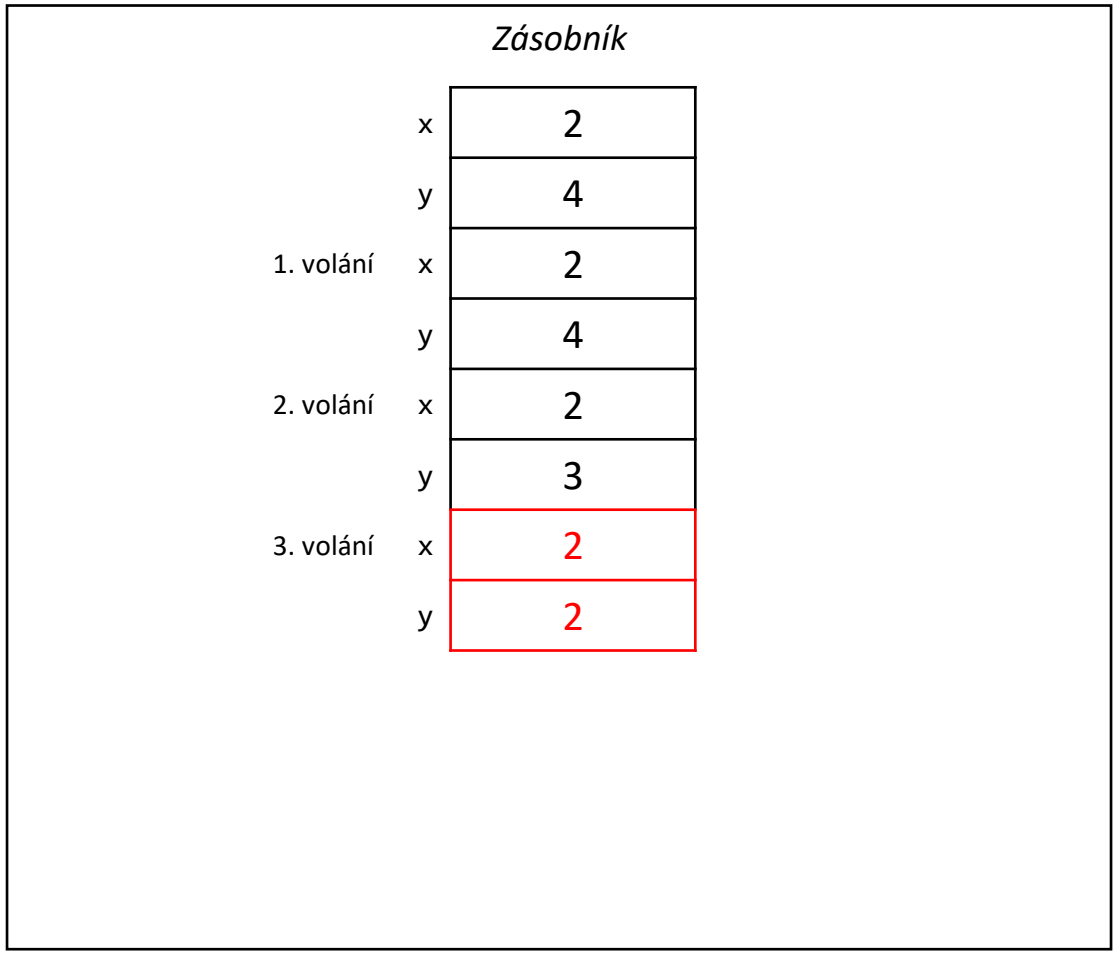
```
static int Mocnina(int x, int y)
{
    if (y == 0)
        return 1;
    else                        2    2
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```



Mocnina - rekurzivní

```
static int Mocnina(int x, int y)
{
    if (y == 0)
        return 1;
    else
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```

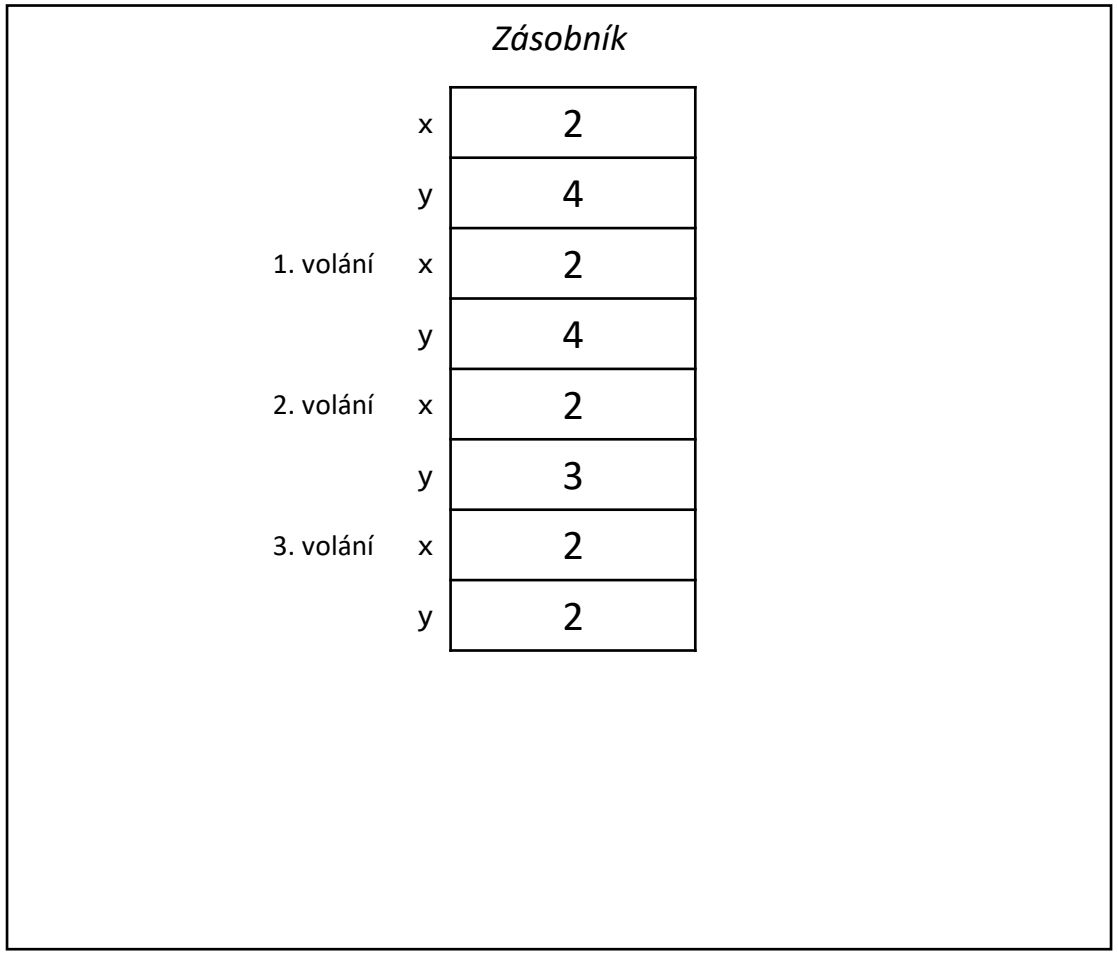
Paměť RAM



Mocnina - rekurzivní

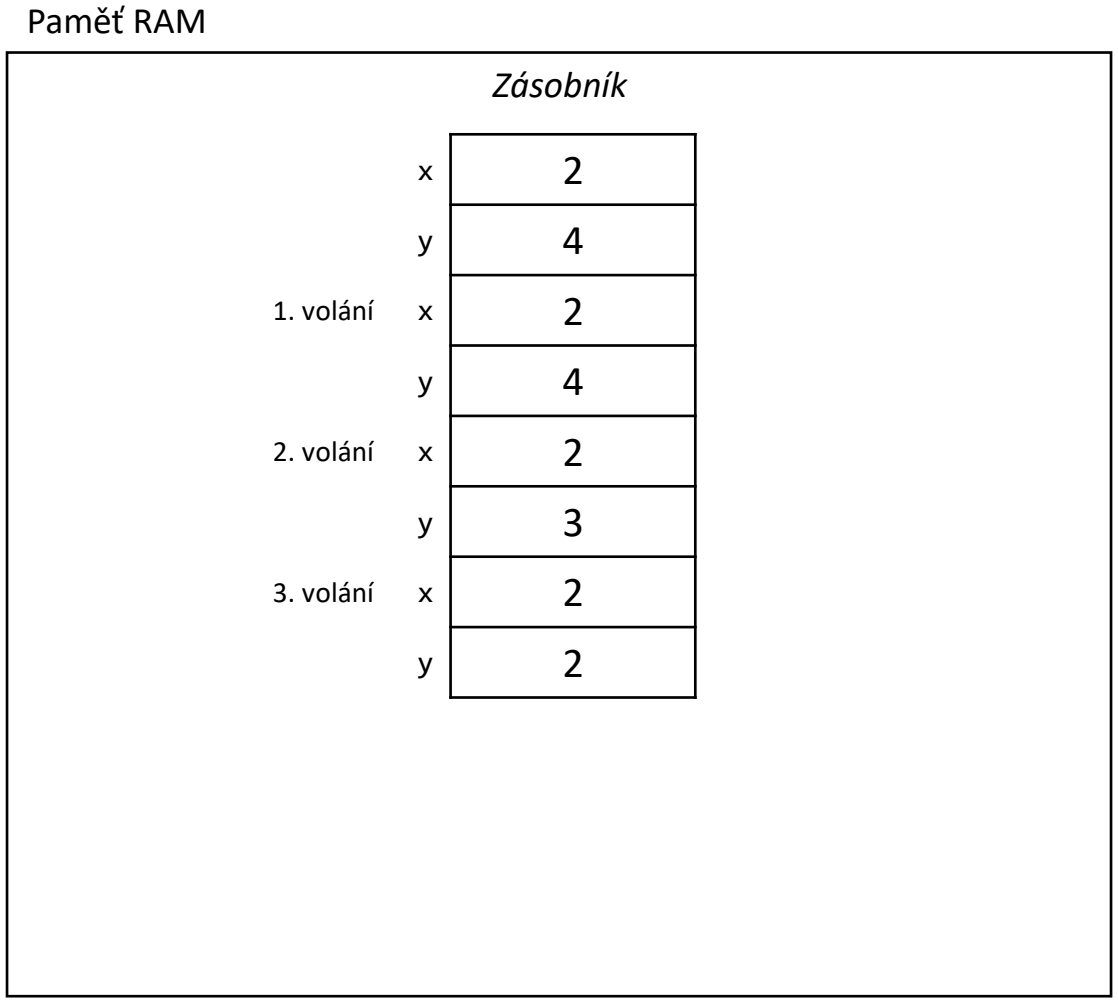
```
static int Mocnina(int x, int y)
{       2
    if (y == 0)
        return 1;
    else
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```

Paměť RAM



Mocnina - rekurzivní

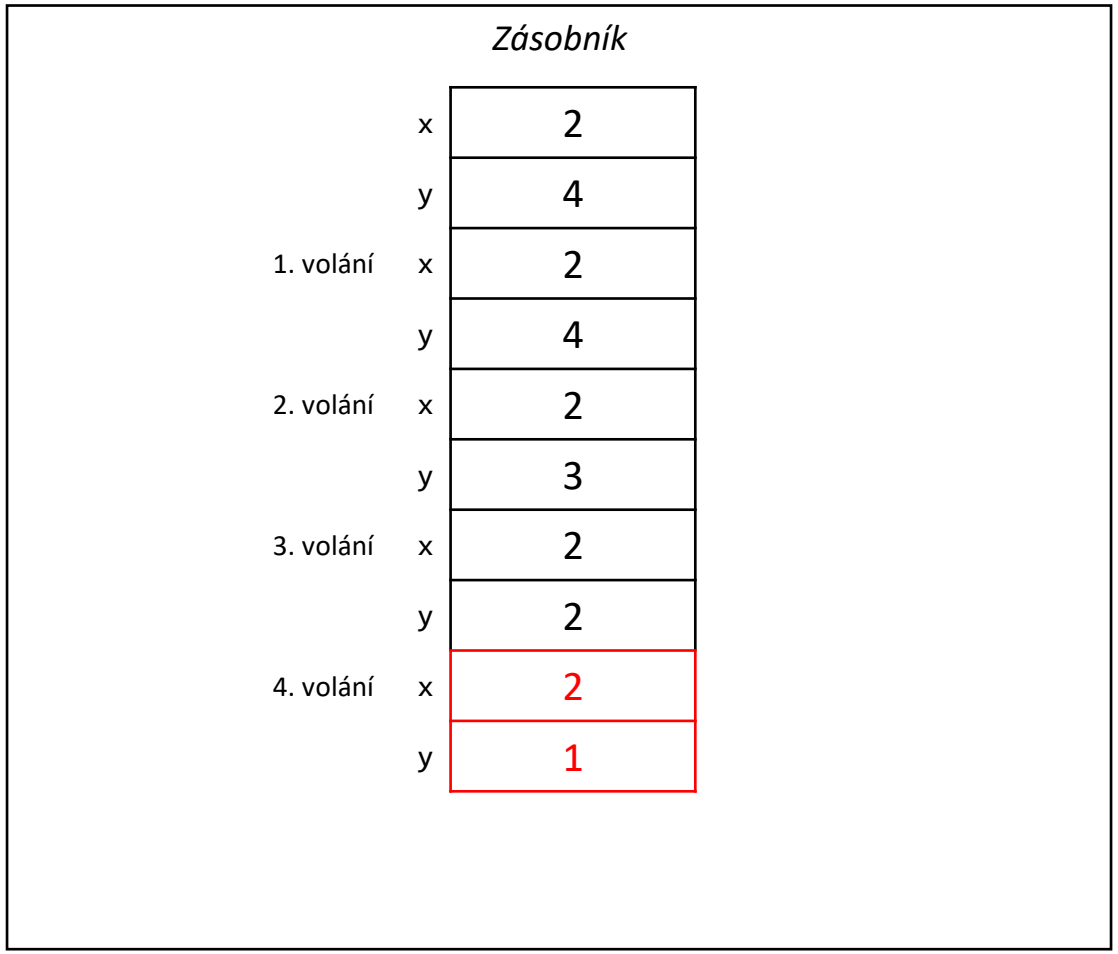
```
static int Mocnina(int x, int y)
{
    if (y == 0)
        return 1;
    else                        2    1
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```



Mocnina - rekurzivní

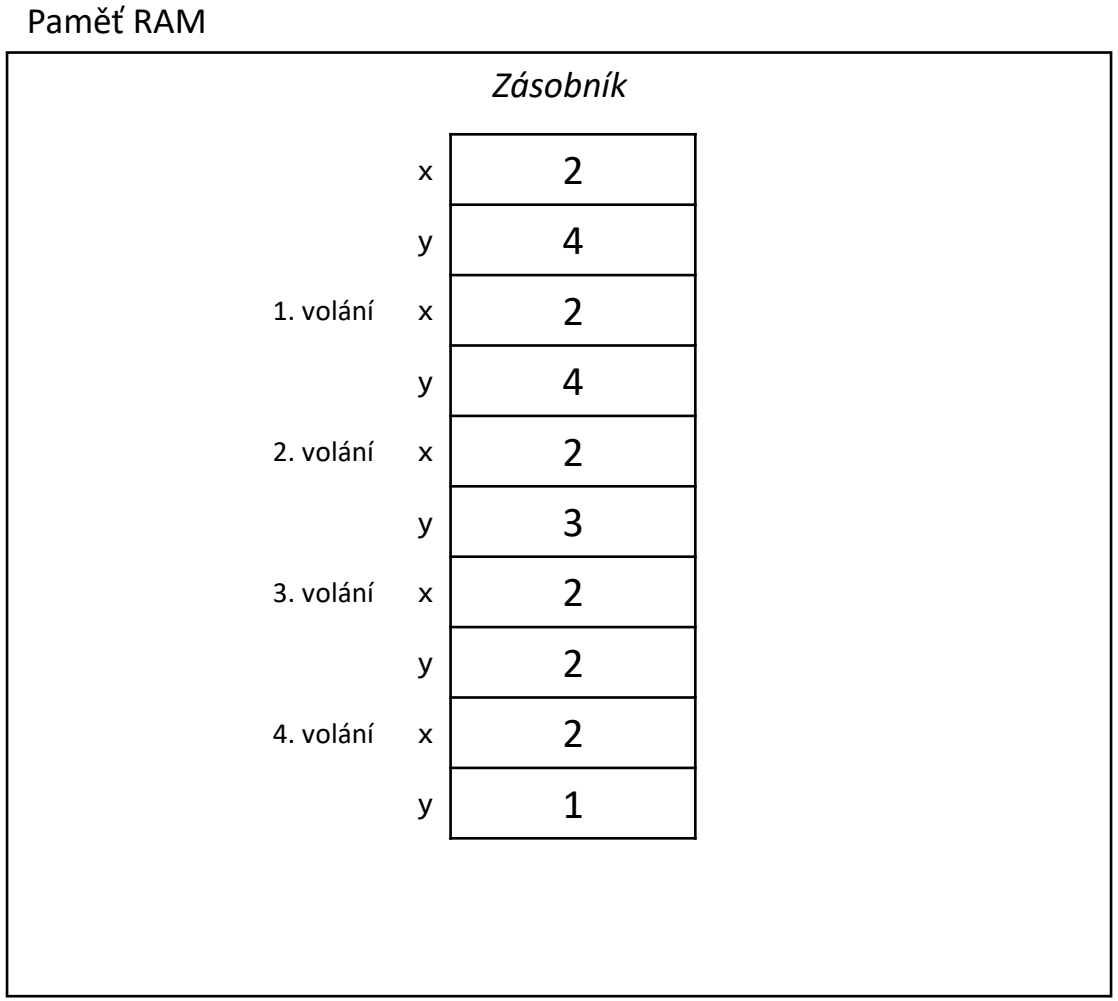
```
static int Mocnina(int x, int y)
{
    if (y == 0)
        return 1;
    else
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```

Paměť RAM



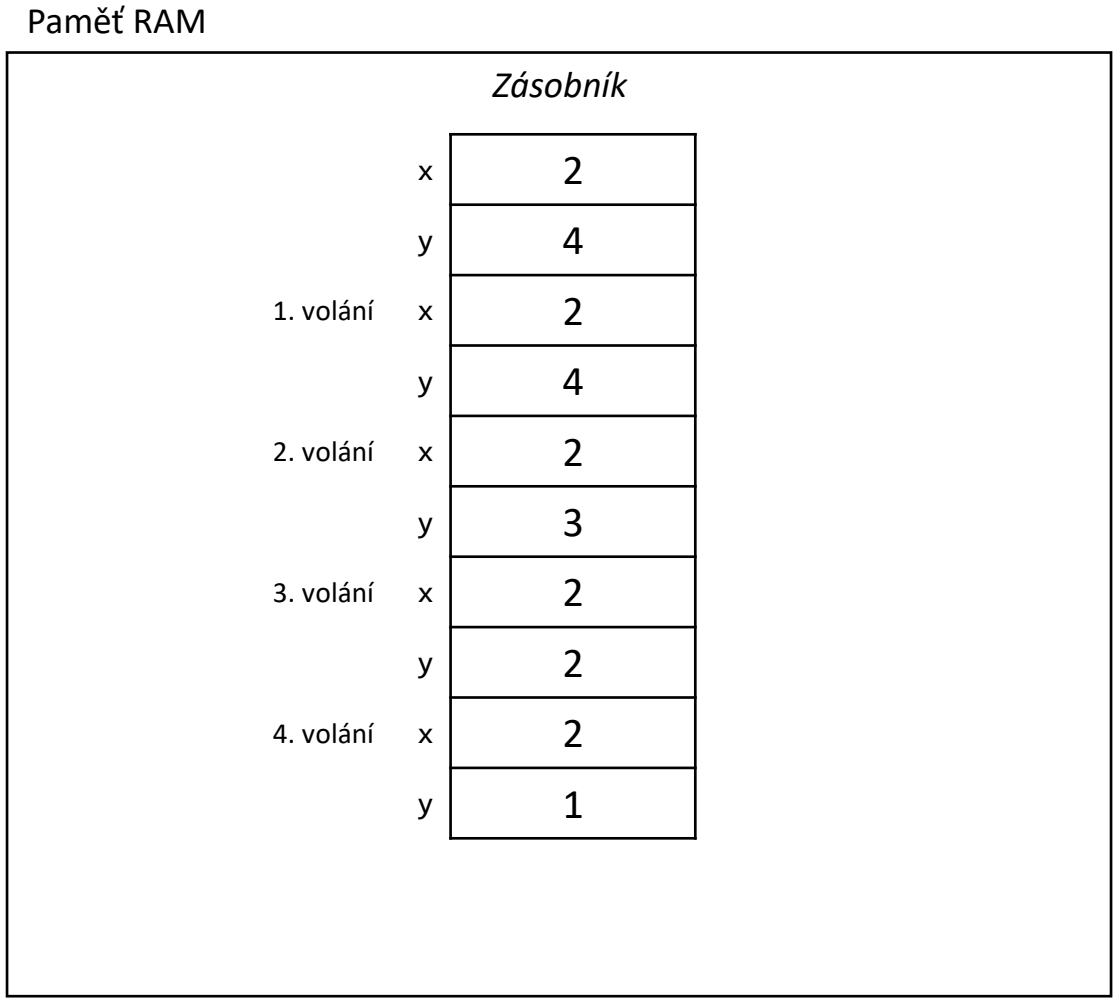
Mocnina - rekurzivní

```
static int Mocnina(int x, int y)
{       1
    if (y == 0)
        return 1;
    else
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```



Mocnina - rekurzivní

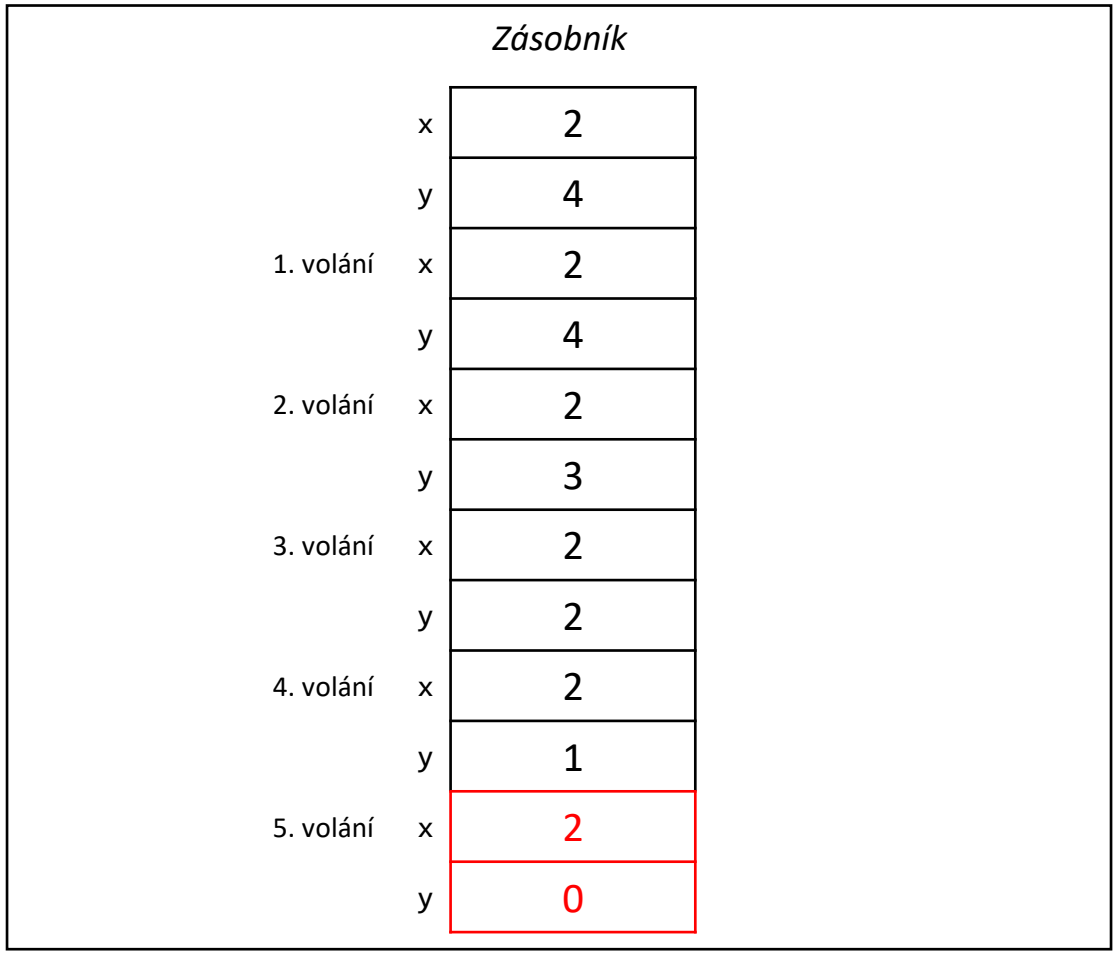
```
static int Mocnina(int x, int y)
{
    if (y == 0)
        return 1;
    else                        2    0
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```



Mocnina - rekurzivní

```
static int Mocnina(int x, int y)
{
    if (y == 0)
        return 1;
    else
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```

Paměť RAM



Mocnina - rekurzivní

```
static int Mocnina(int x, int y)
{       0
    if (y == 0)
        return 1;
    else
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```

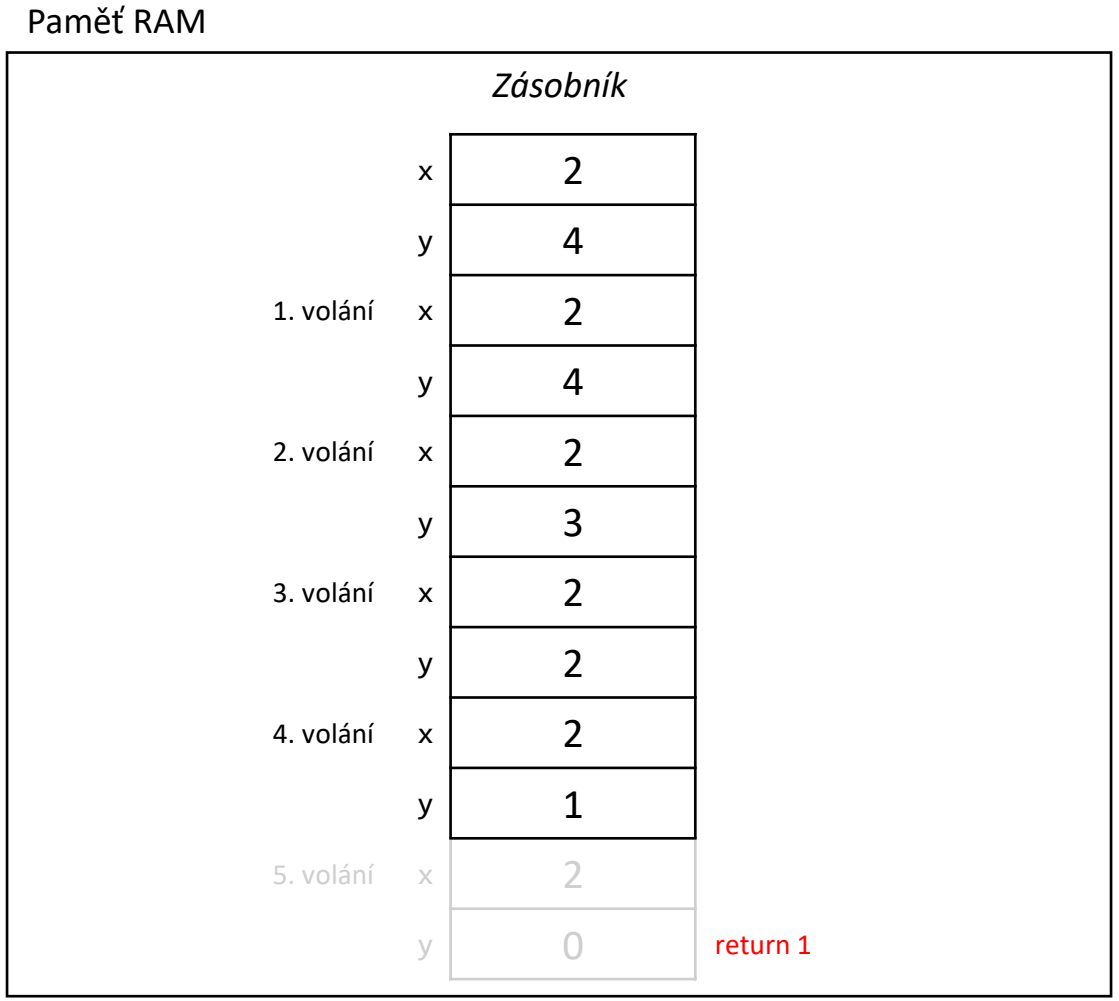
Paměť RAM

Zásobník

	x	2
	y	4
1. volání	x	2
	y	4
2. volání	x	2
	y	3
3. volání	x	2
	y	2
4. volání	x	2
	y	1
5. volání	x	2
	y	0

Mocnina - rekurzivní

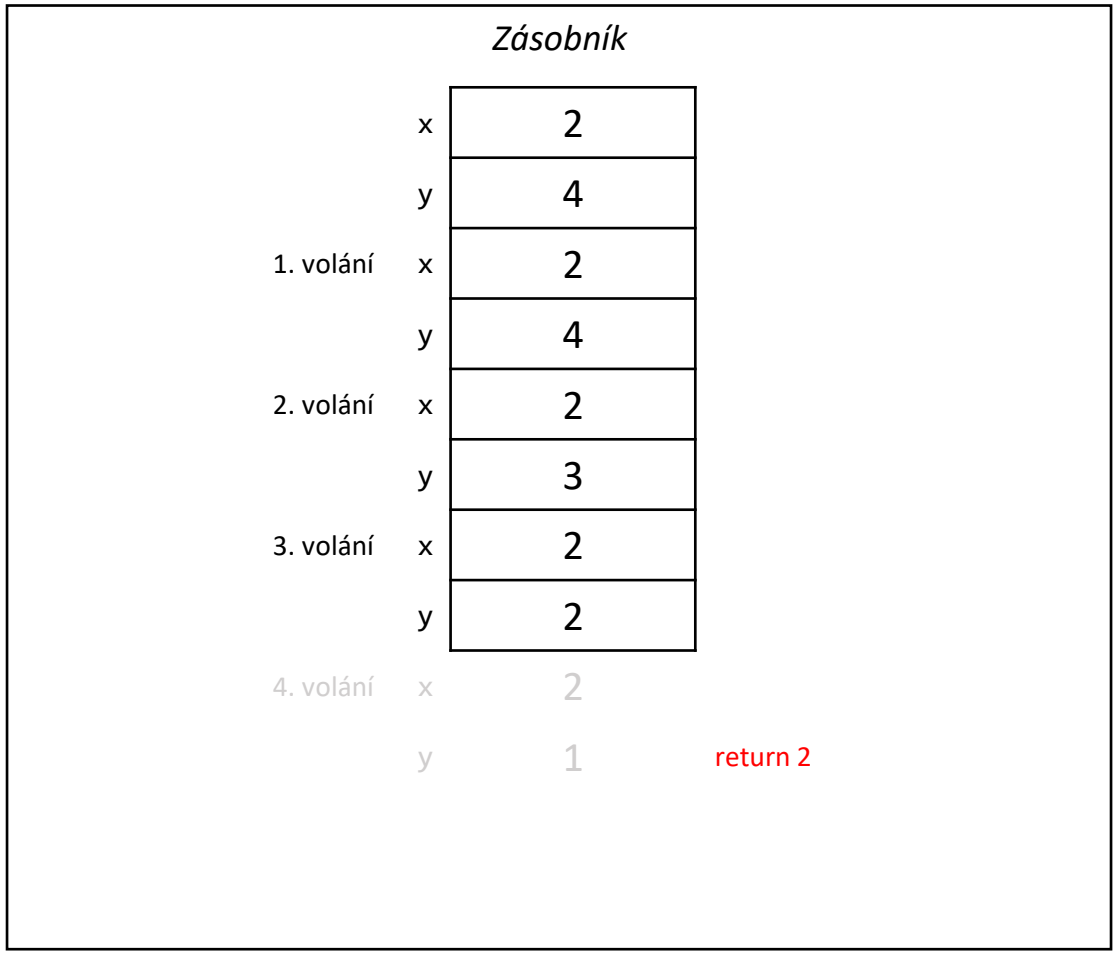
```
static int Mocnina(int x, int y)
{
    if (y == 0)
        return 1;
    else
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```



Mocnina - rekurzivní

```
static int Mocnina(int x, int y)
{
    if (y == 0)
        return 1;
    else        2      1
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);

    Console.WriteLine(mocnina);
}
```

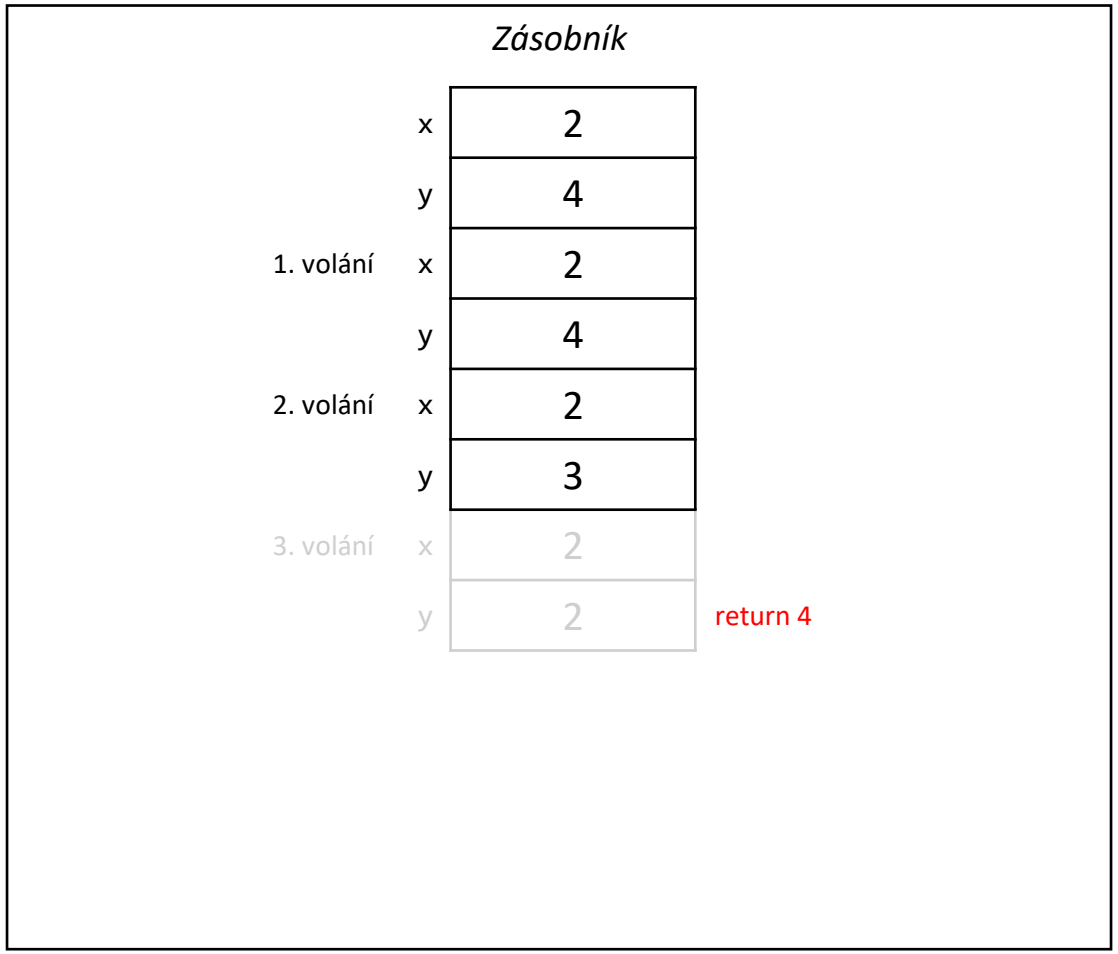
Paměť RAM



Mocnina - rekurzivní

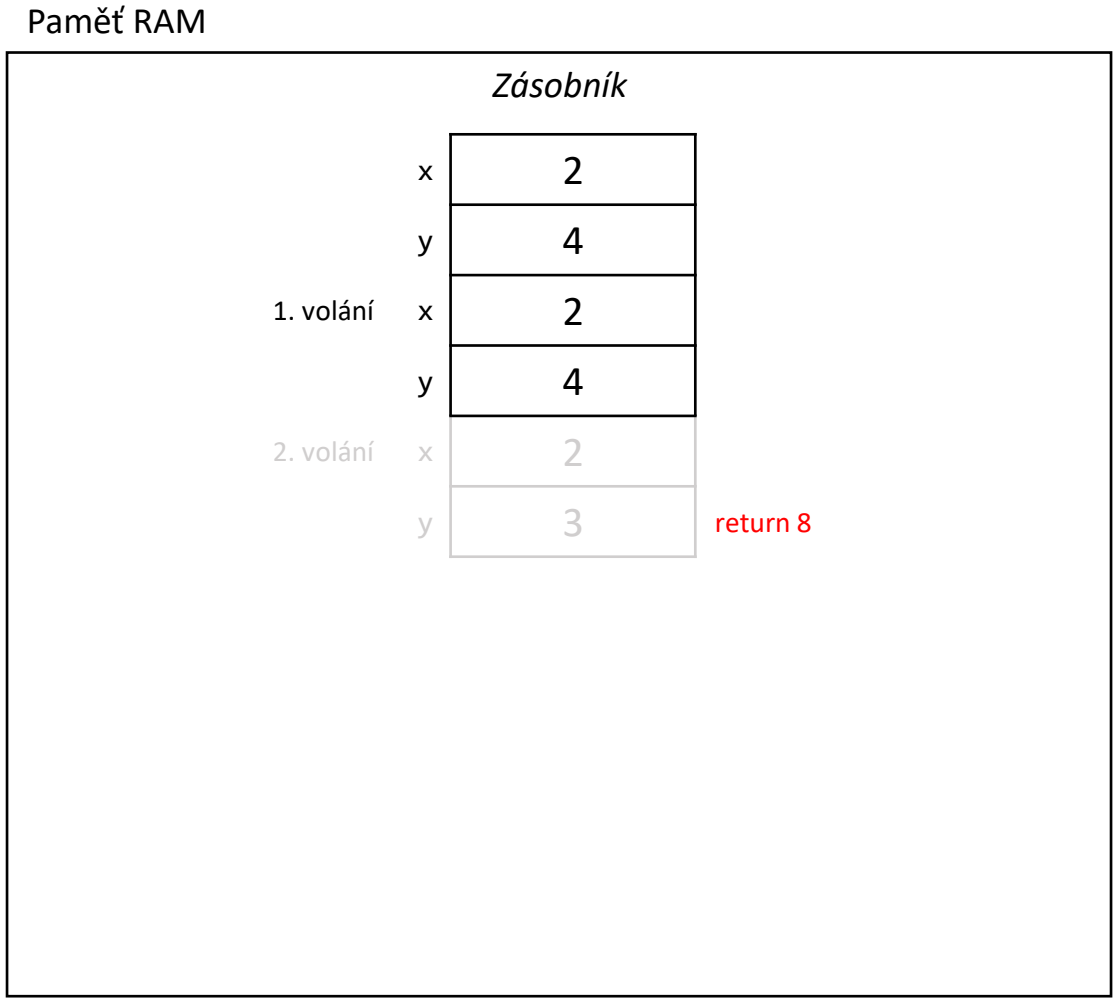
```
static int Mocnina(int x, int y)
{
    if (y == 0)
        return 1;
    else        2      2
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);

    Console.WriteLine(mocnina);
}
```

Paměť RAM



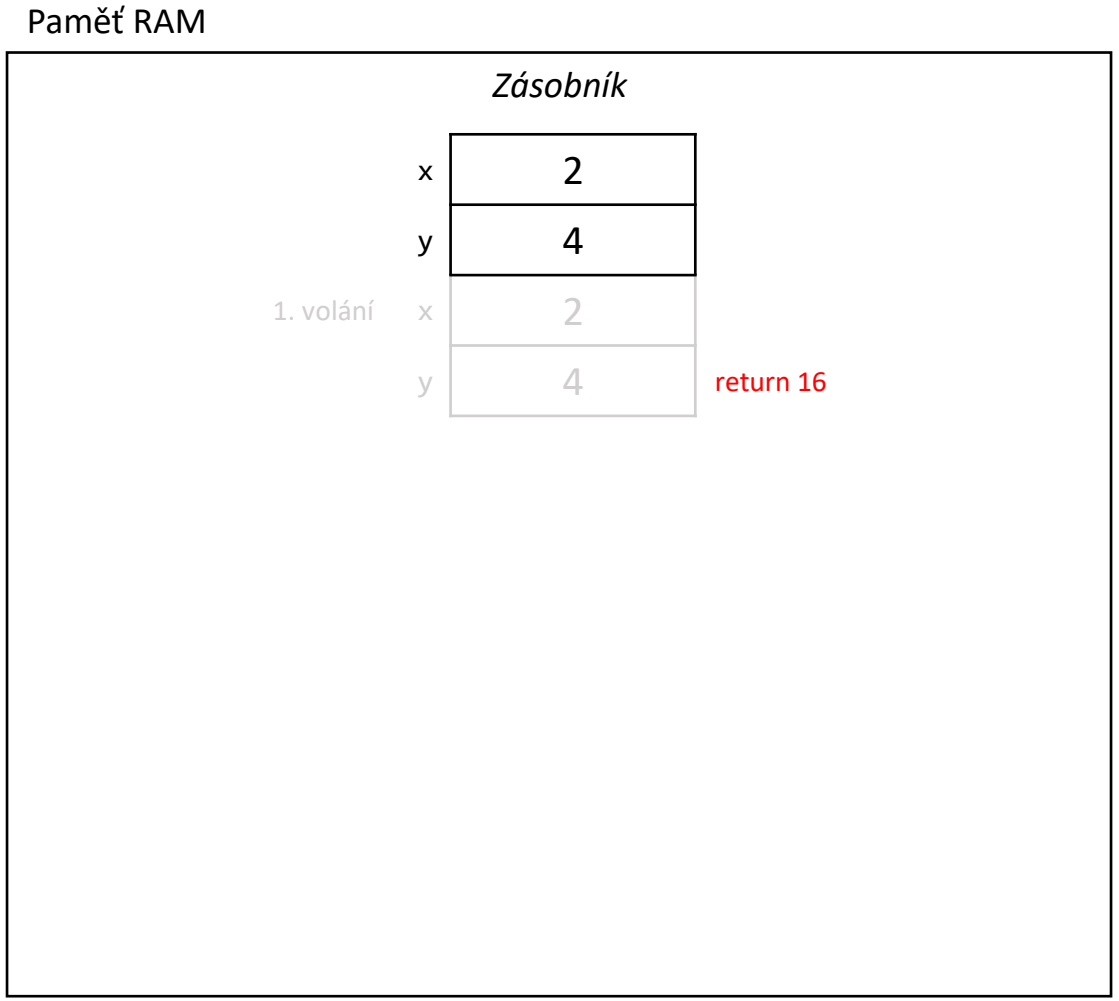
Mocnina - rekurzivní

```
static int Mocnina(int x, int y)
{
    if (y == 0)
        return 1;
    else        2      4
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);

    Console.WriteLine(mocnina);
}
```



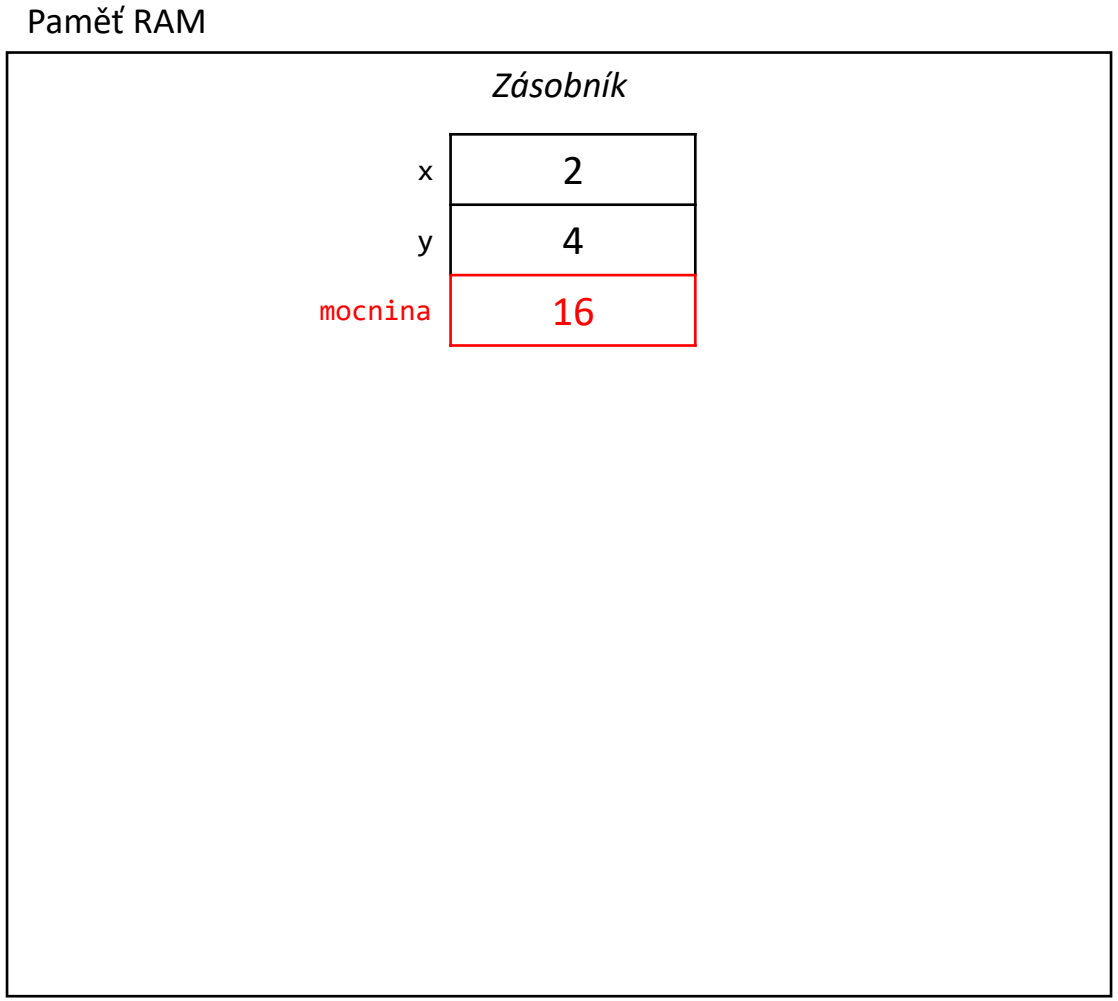
Mocnina - rekurzivní

```
static int Mocnina(int x, int y)
{
    if (y == 0)
        return 1;
    else        2      8
        return x * Mocnina(x, y - 1);
}

static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```



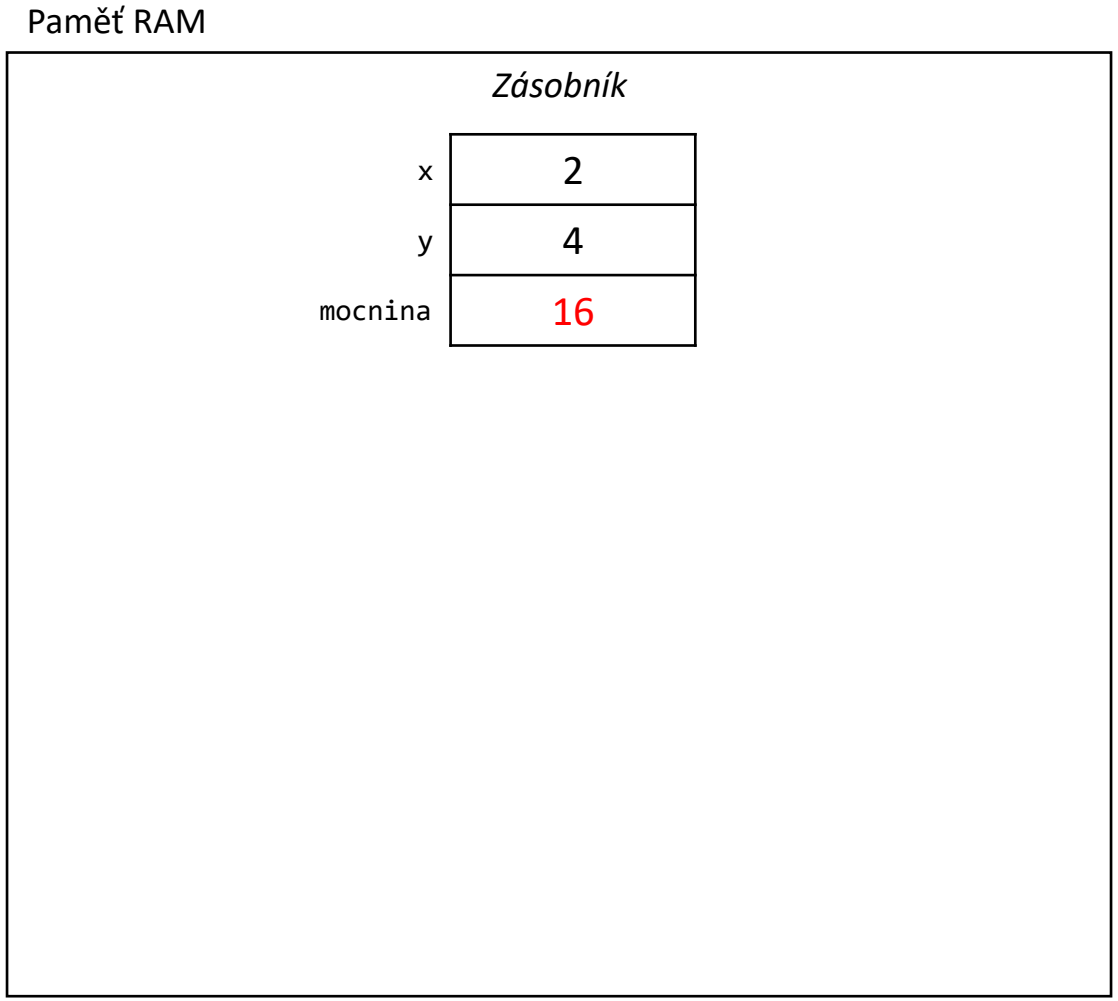
Mocnina - rekurzivní

```
static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);


}
```



Mocnina - rekurzivní

```
static void Main(string[] args)
{
    int x = 2;
    int y = 4;

    int mocnina = Mocnina(x, y);

    Console.WriteLine(mocnina);
}
```



Faktoriál rekurzivní

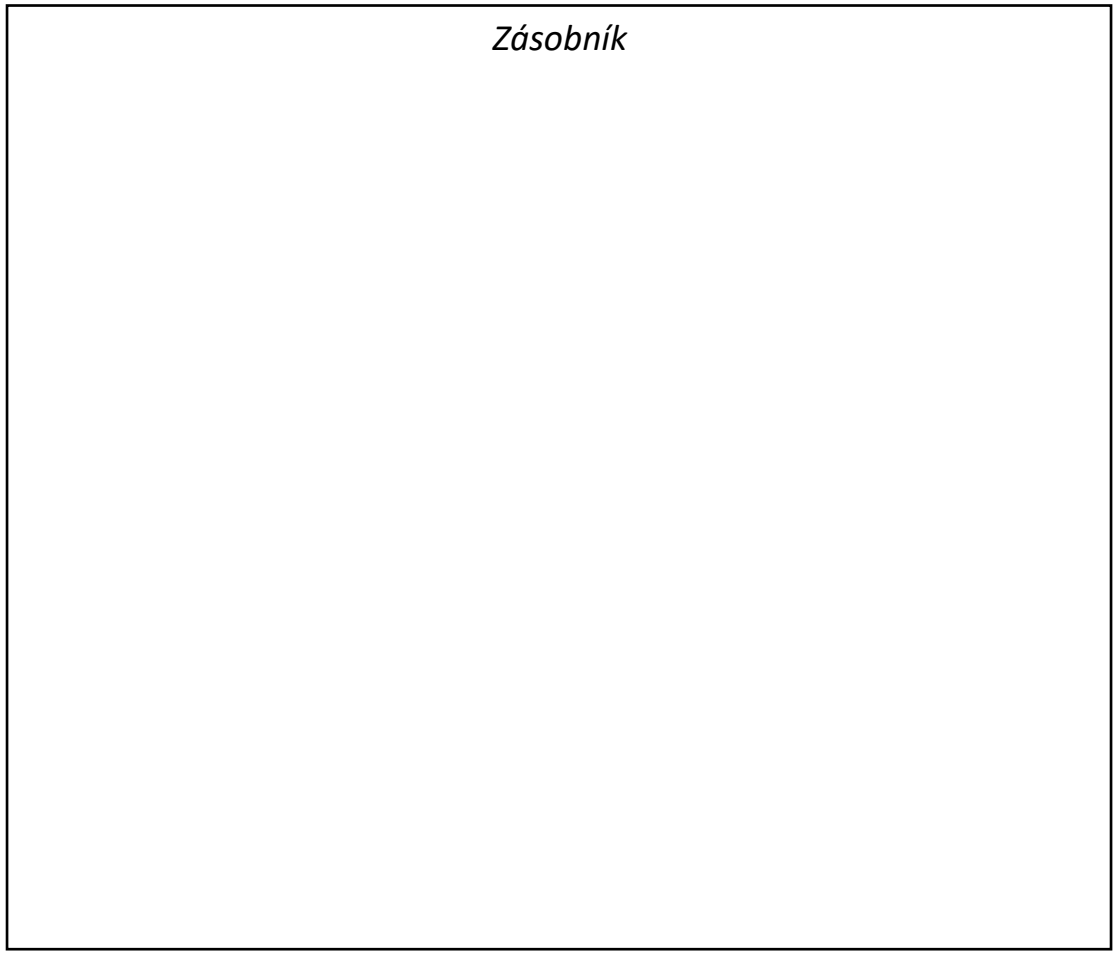
- Následující algoritmus spočítá hodnotu výrazu $n!$
- Nejprve si nadefinujeme proměnnou *n* a potom zavoláme metodu *Faktorial* s parametrem.
- Metoda *Faktorial* vrátí hodnotu *1* pokud bude mít parametr *n* hodnotu *1* a jinak vrátí hodnotu *n* krát výsledek rekurzivního volání *Faktorial* s argumentem *n-1*.

$$n! = \begin{cases} 1, & n = 0 \\ n(n-1), & n \neq 0 \end{cases}$$

Faktoriál rekurzivní

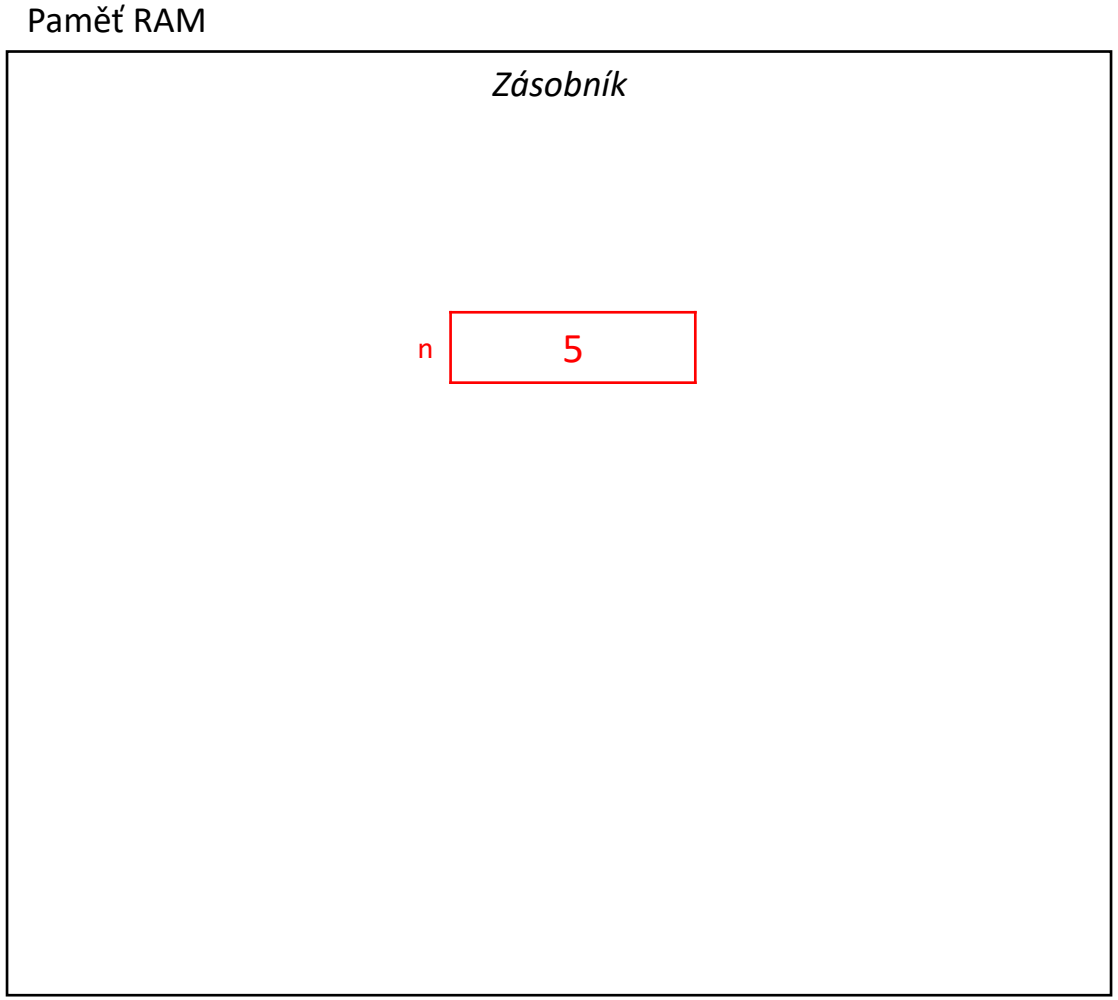
```
static int Faktorial(int n)
{
    if (n == 1)
        return 1;
    else
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);

    Console.WriteLine(faktorial);
}
```

Paměť RAM



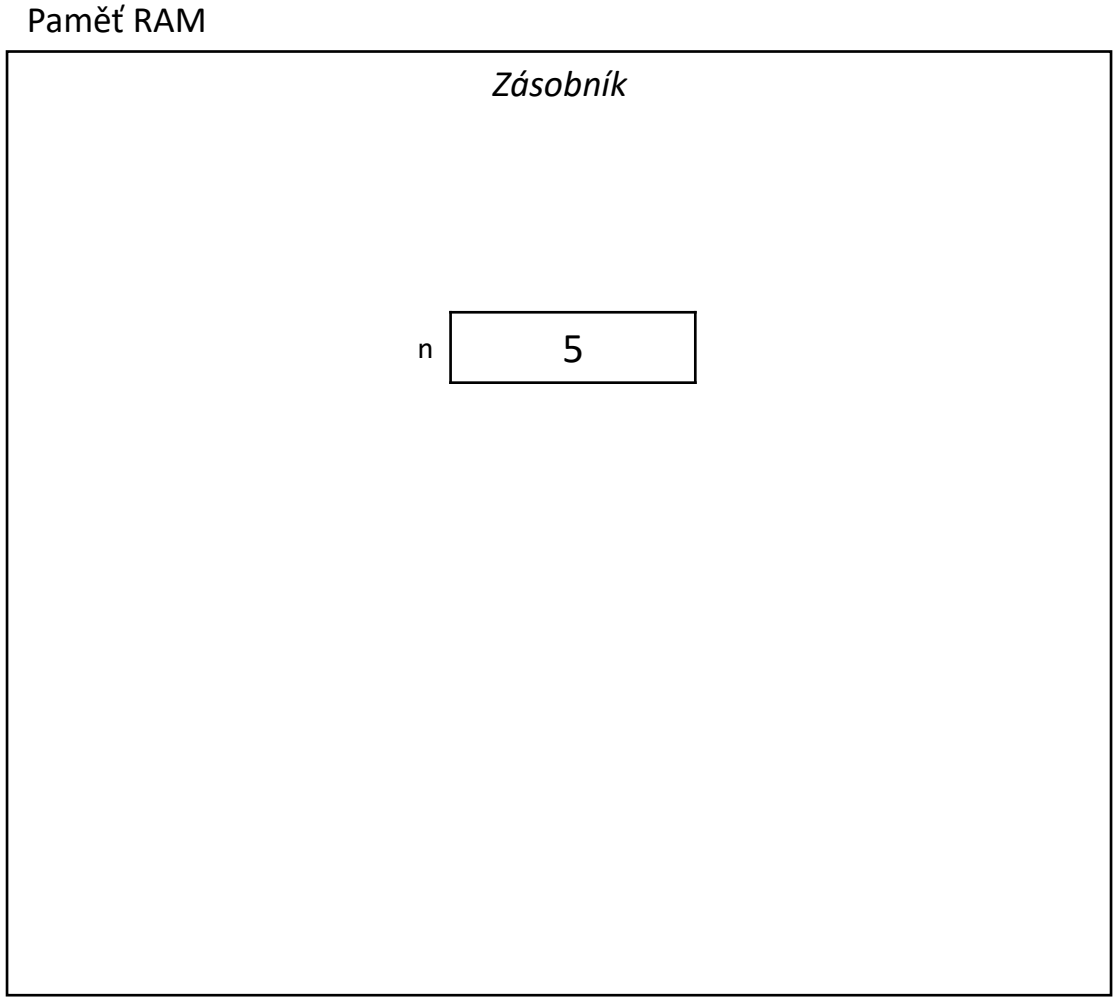
Faktoriál rekurzivní

```
static void Main(string[] args)
{
    int n = 5;



}
```



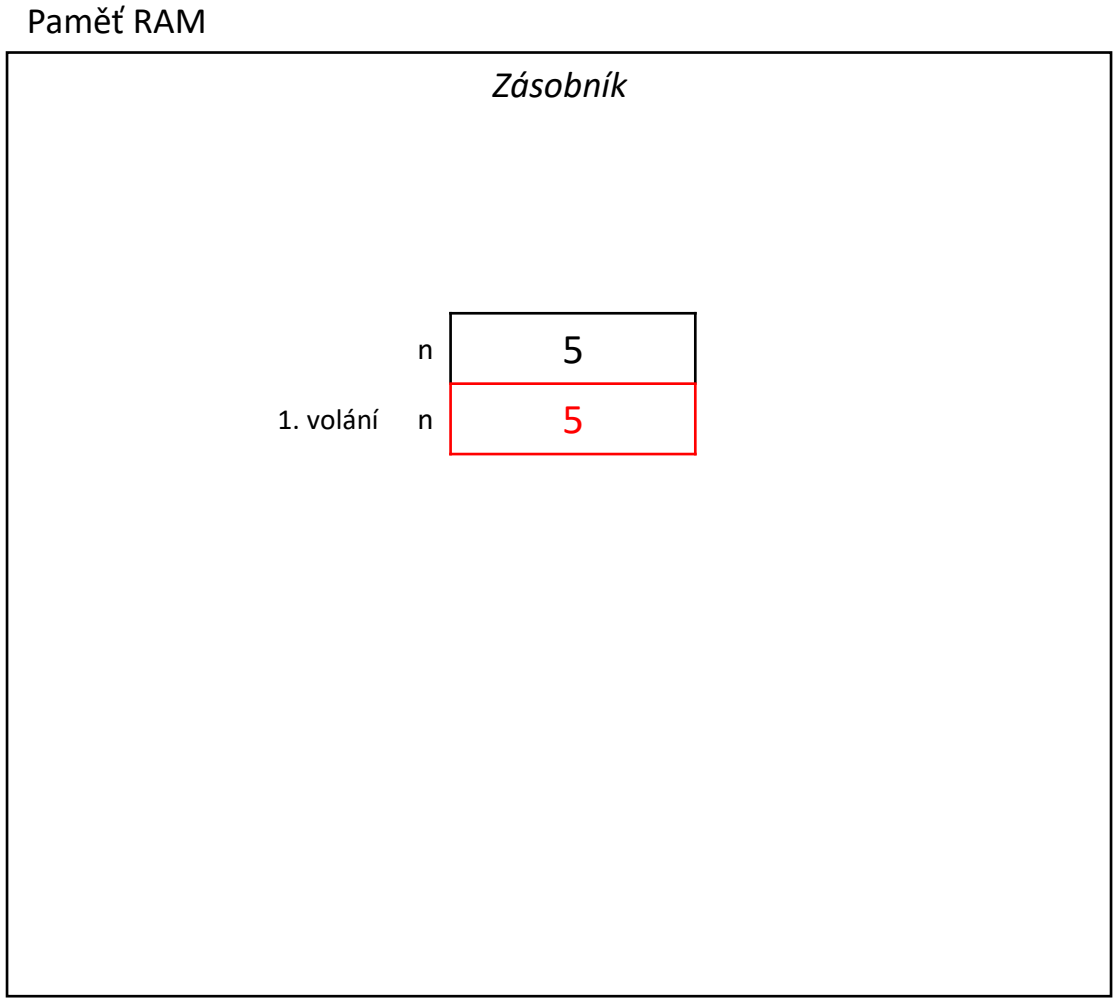
Faktoriál rekurzivní

```
static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



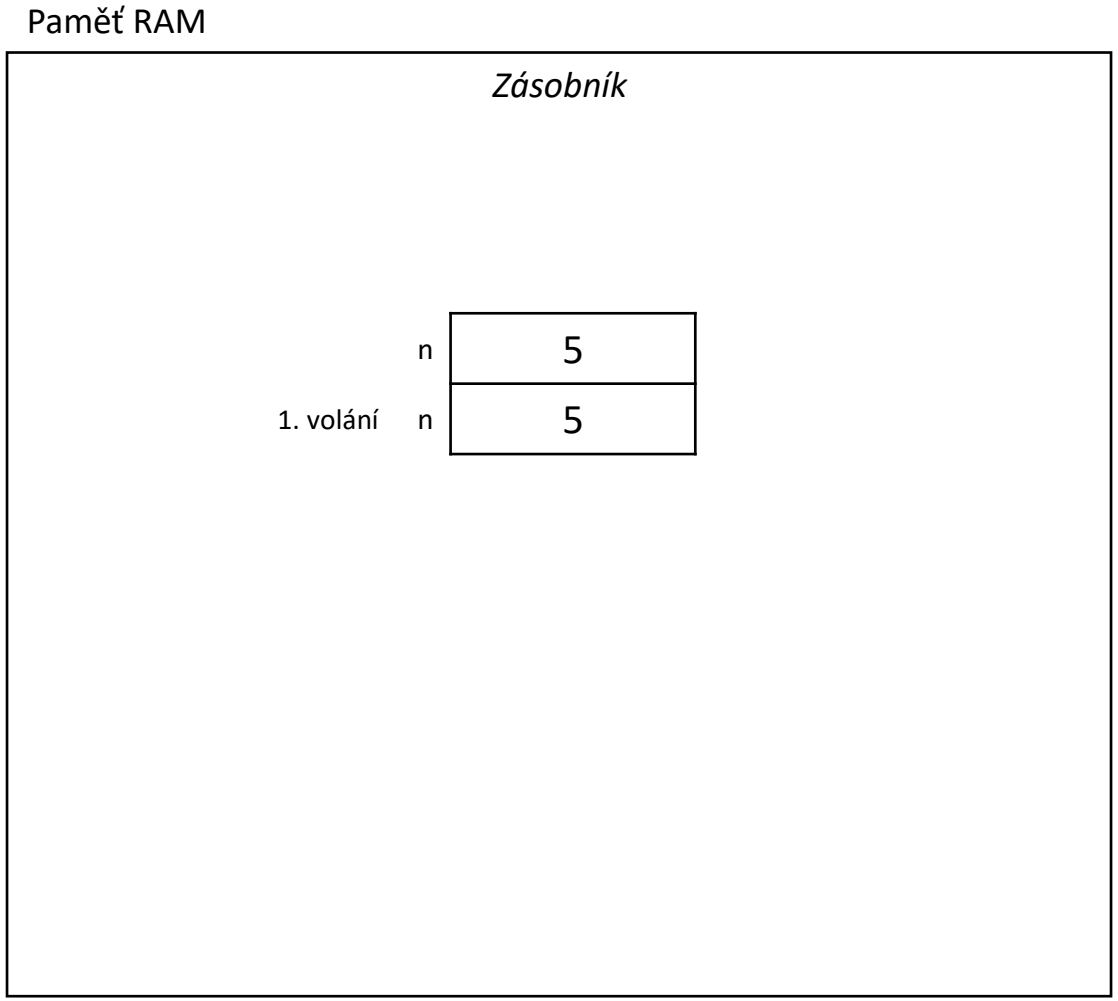
Faktoriál rekurzivní

```
static int Faktorial(int n)
{




}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



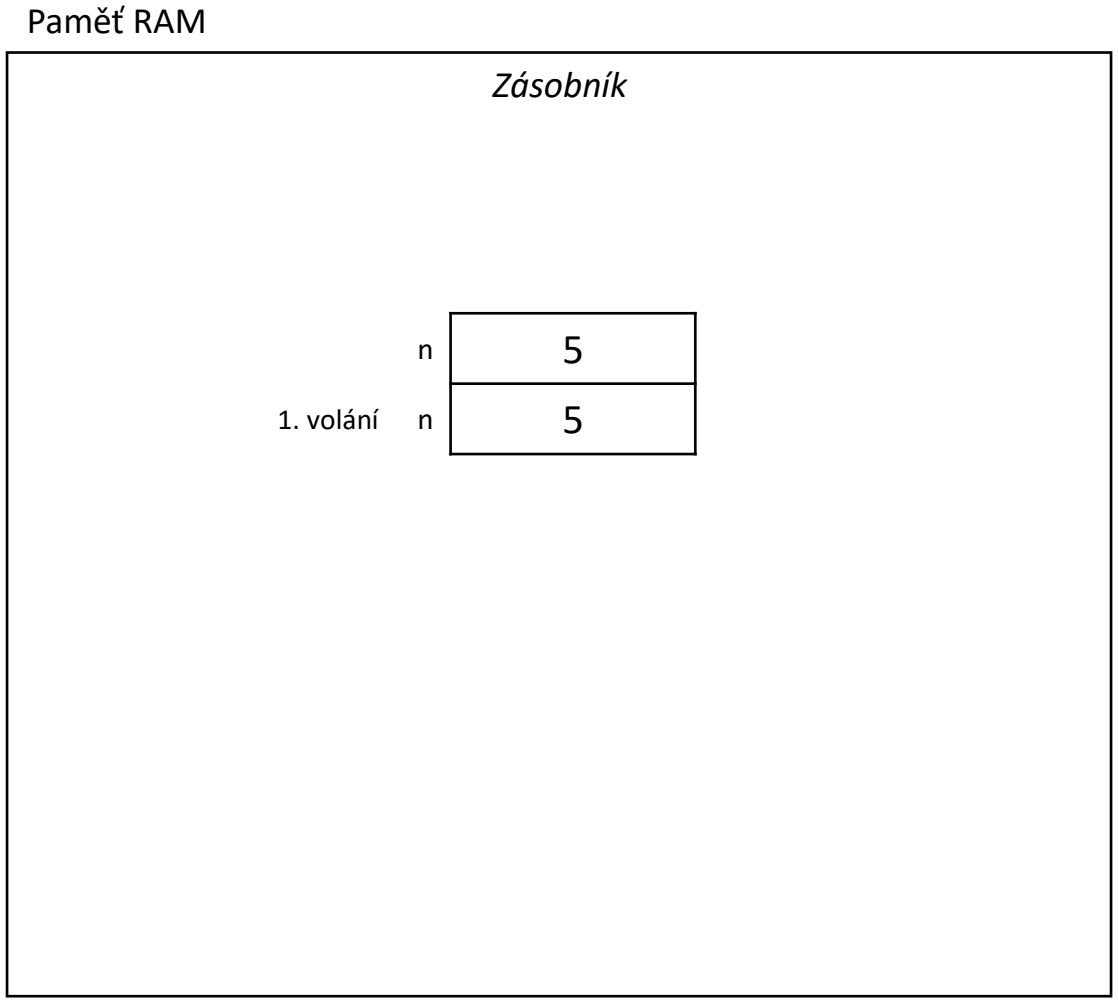
Faktoriál rekurzivní

```
static int Faktorial(int n)
{       5
    if (n == 1)
        return 1;
    else
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



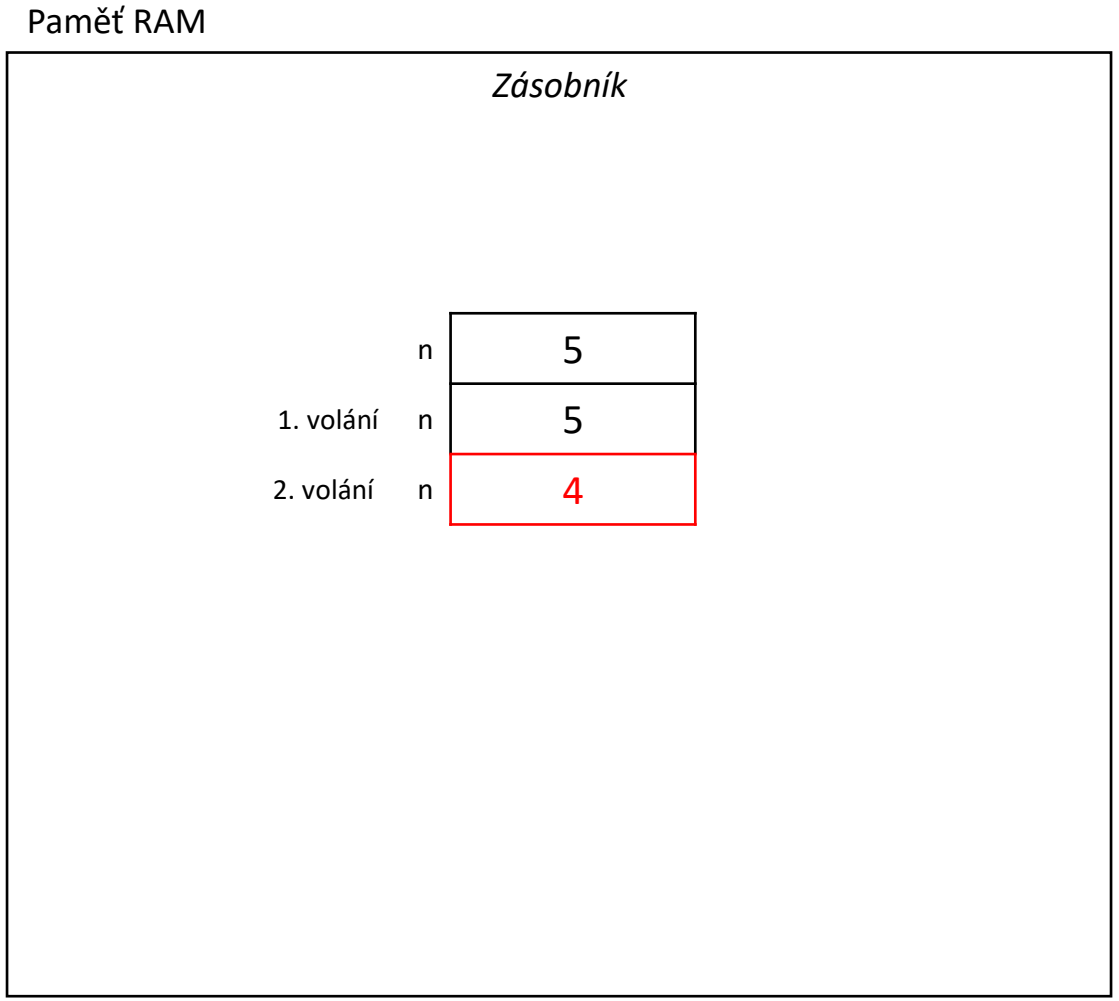
Faktoriál rekurzivní

```
static int Faktorial(int n)
{
    if (n == 1)
        return 1;
    else                          4
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



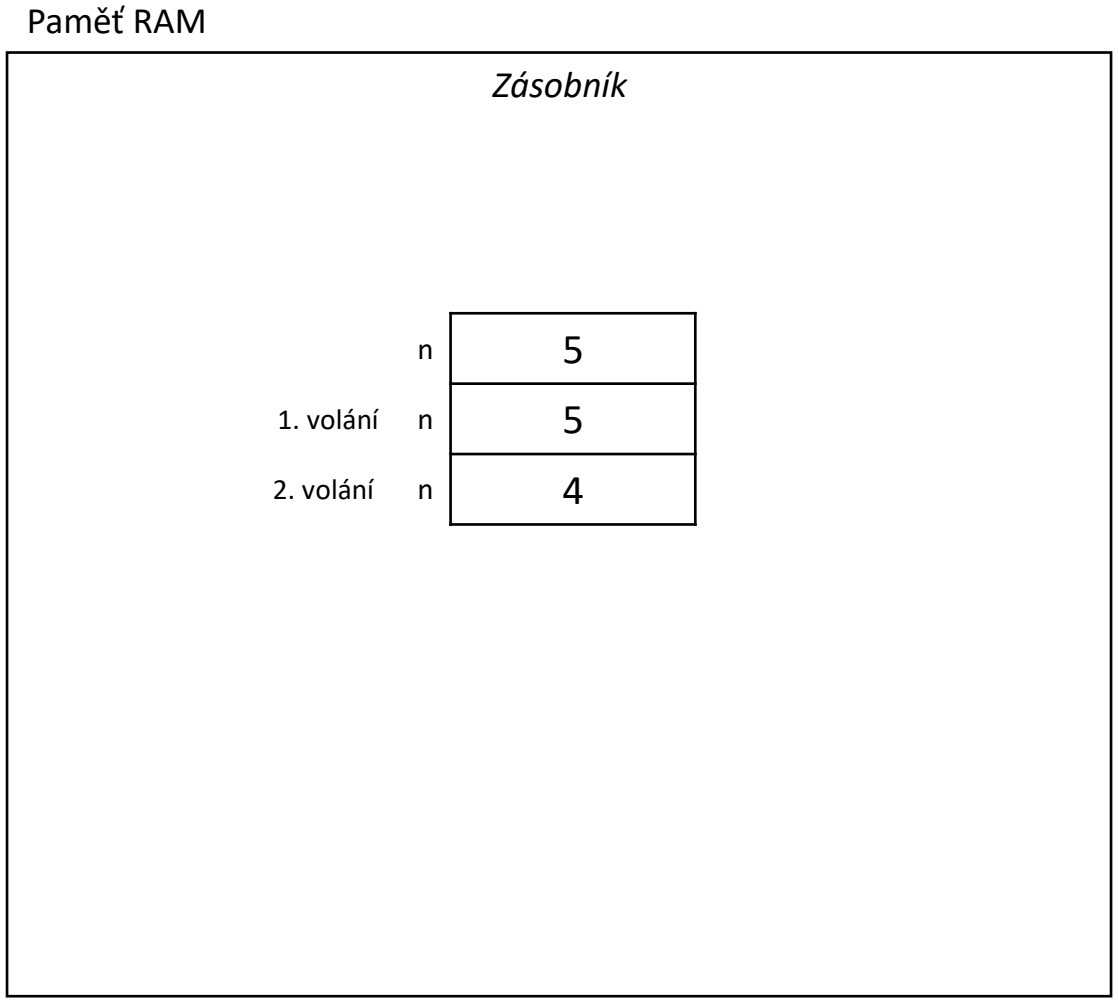
Faktoriál rekurzivní

```
static int Faktorial(int n)
{
    if (n == 1)
        return 1;
    else
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



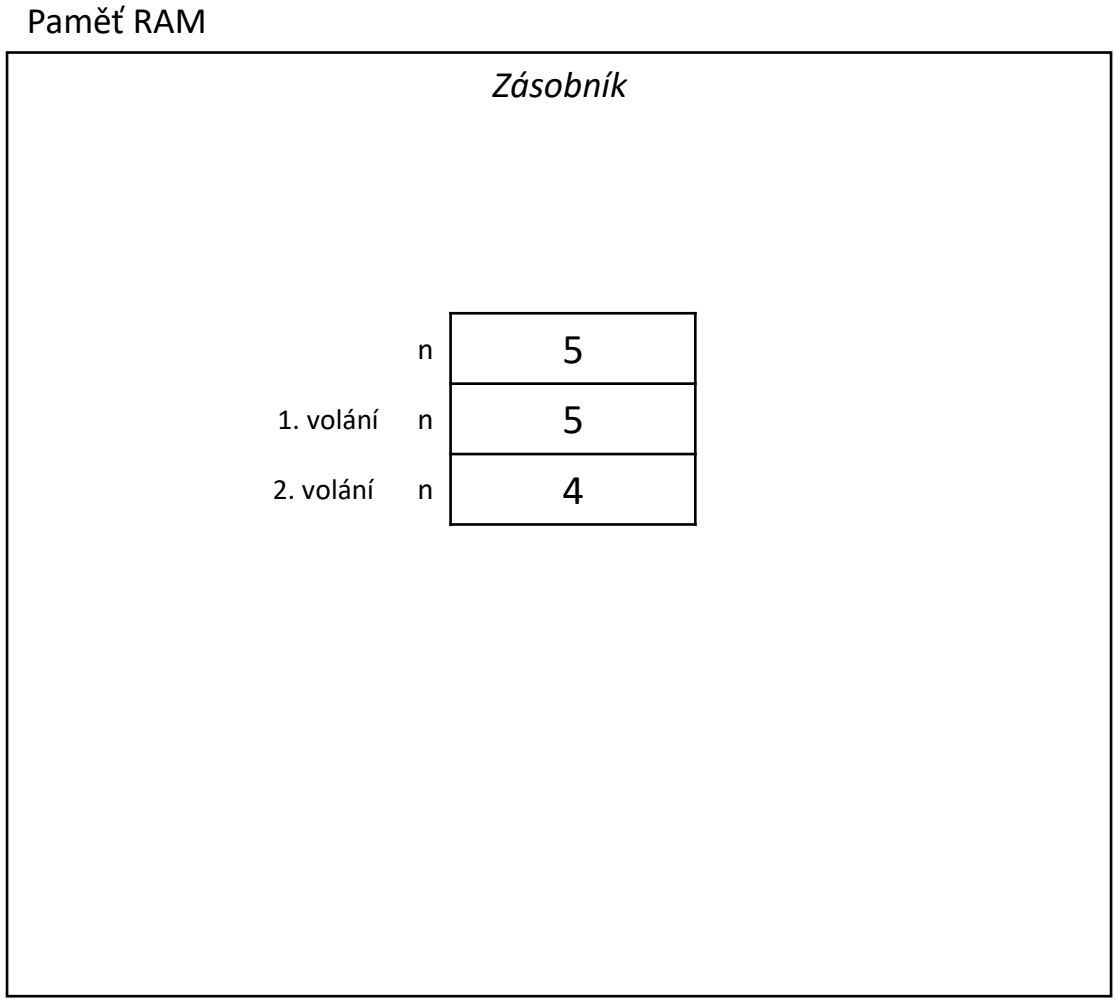
Faktoriál rekurzivní

```
static int Faktorial(int n)
{       4
    if (n == 1)
        return 1;
    else
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



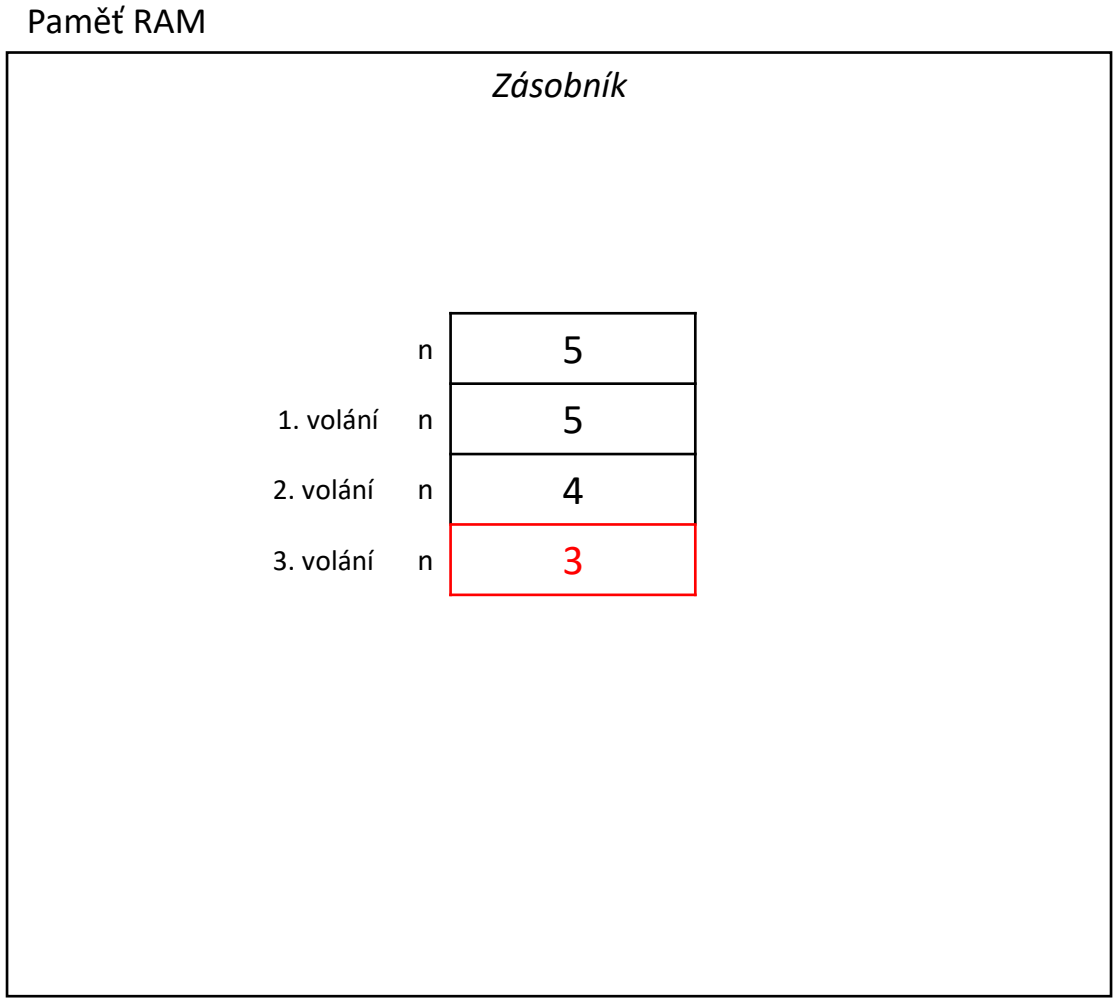
Faktoriál rekurzivní

```
static int Faktorial(int n)
{
    if (n == 1)
        return 1;
    else                          3
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



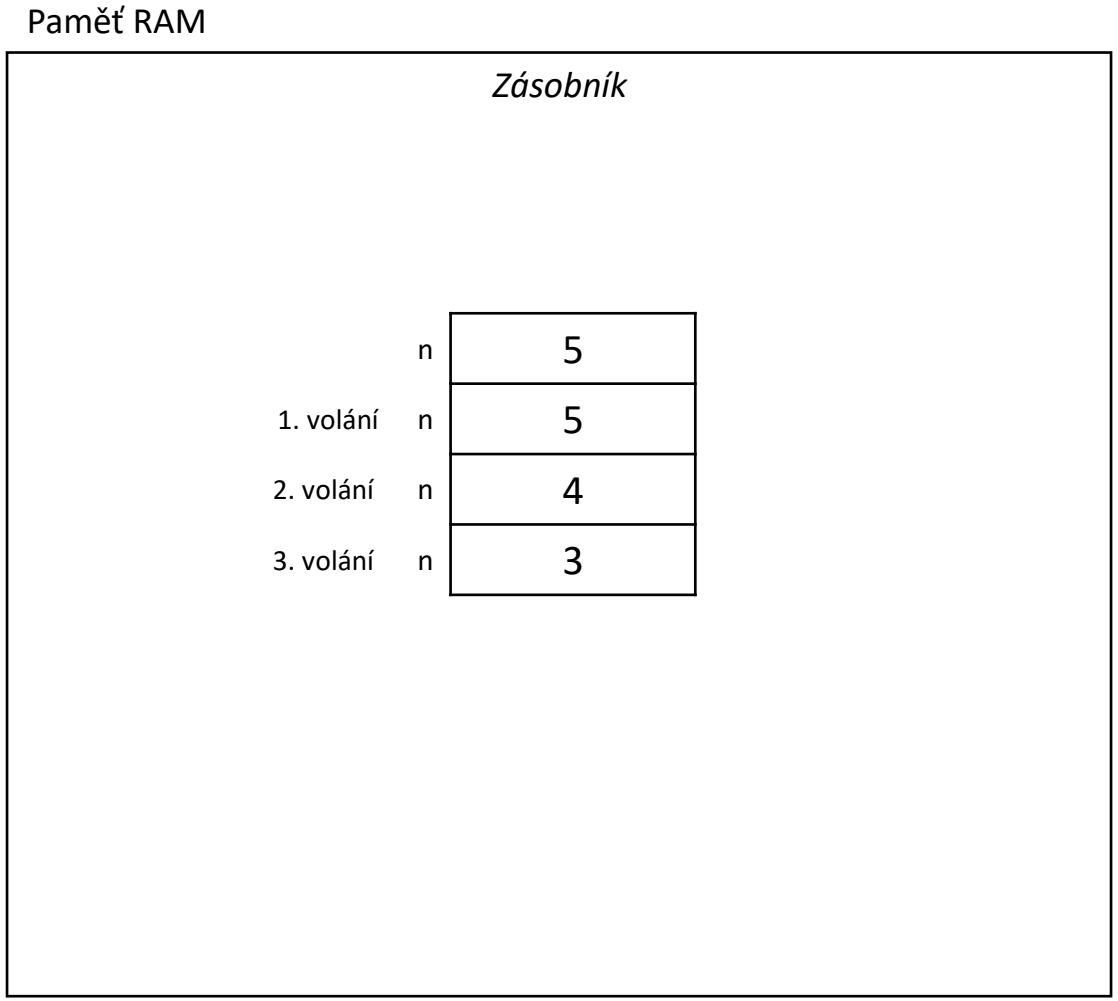
Faktoriál rekurzivní

```
static int Faktorial(int n)
{
    if (n == 1)
        return 1;
    else
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



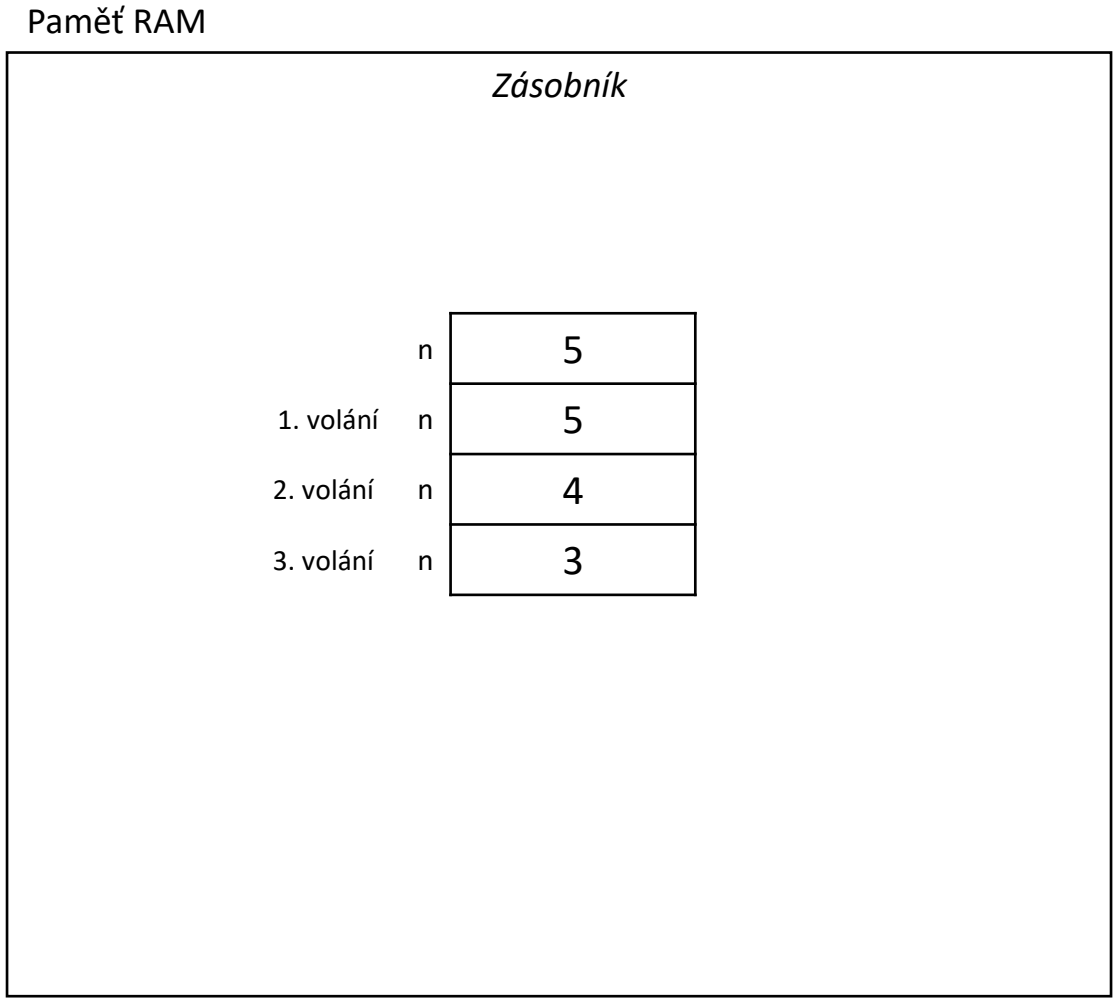
Faktoriál rekurzivní

```
static int Faktorial(int n)
{       3
    if (n == 1)
        return 1;
    else
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



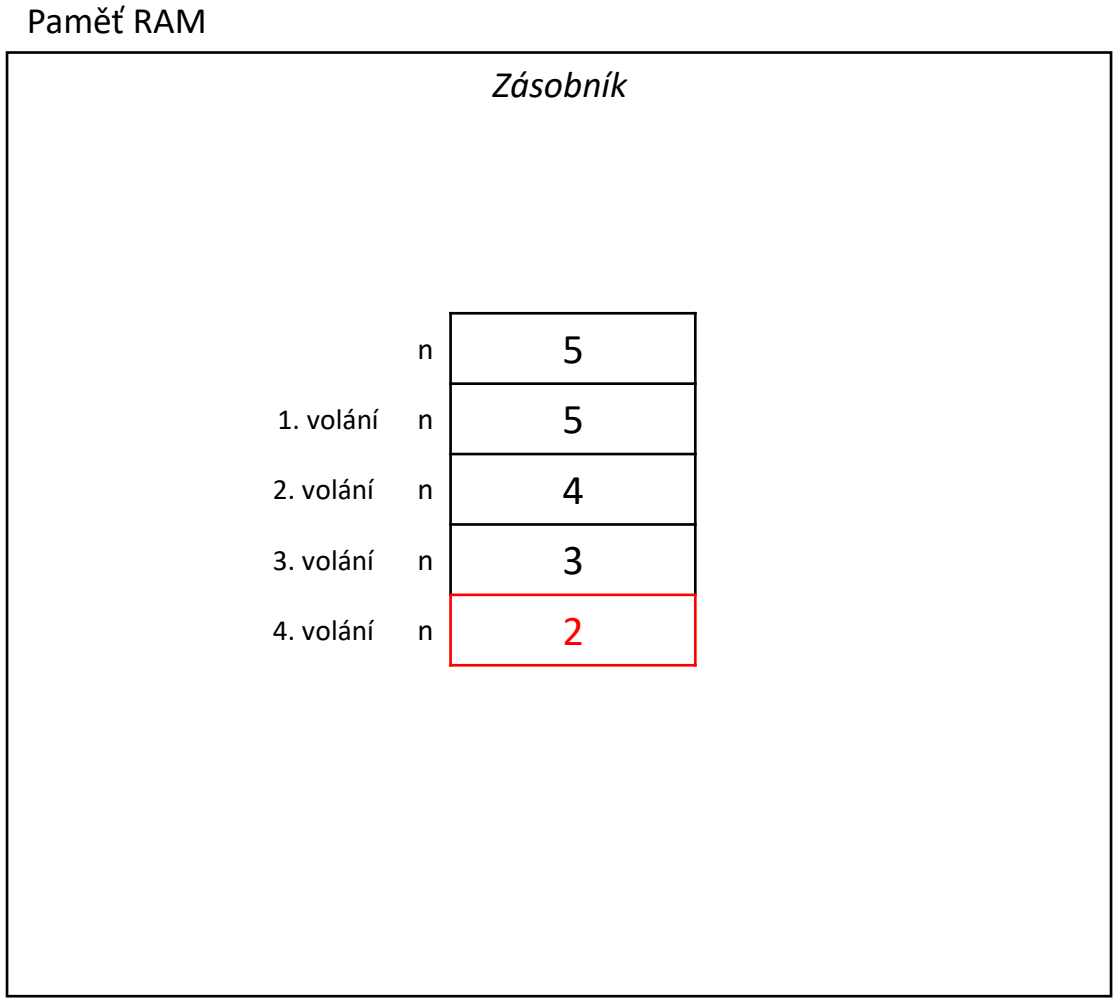
Faktoriál rekurzivní

```
static int Faktorial(int n)
{
    if (n == 1)
        return 1;
    else                          2
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



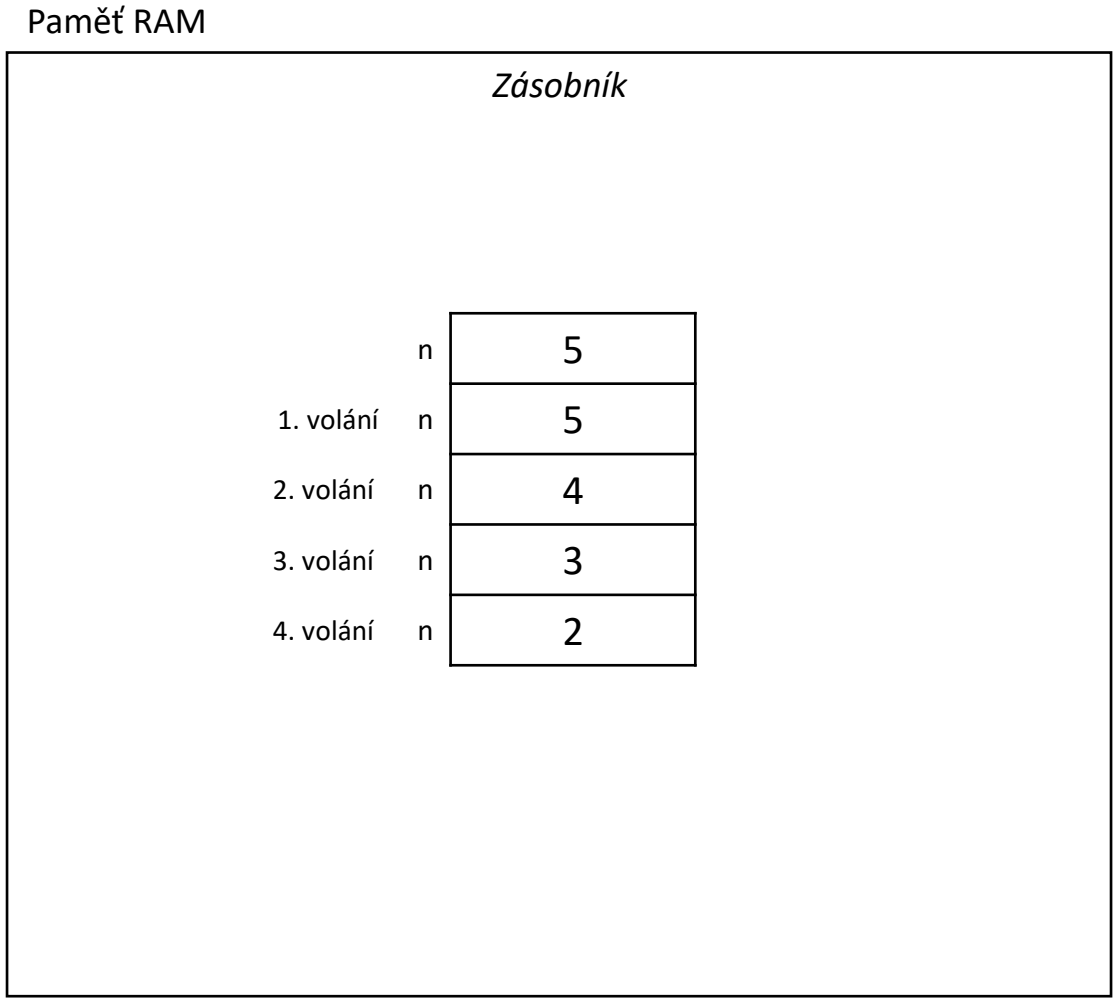
Faktoriál rekurzivní

```
static int Faktorial(int n)
{
    if (n == 1)
        return 1;
    else
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



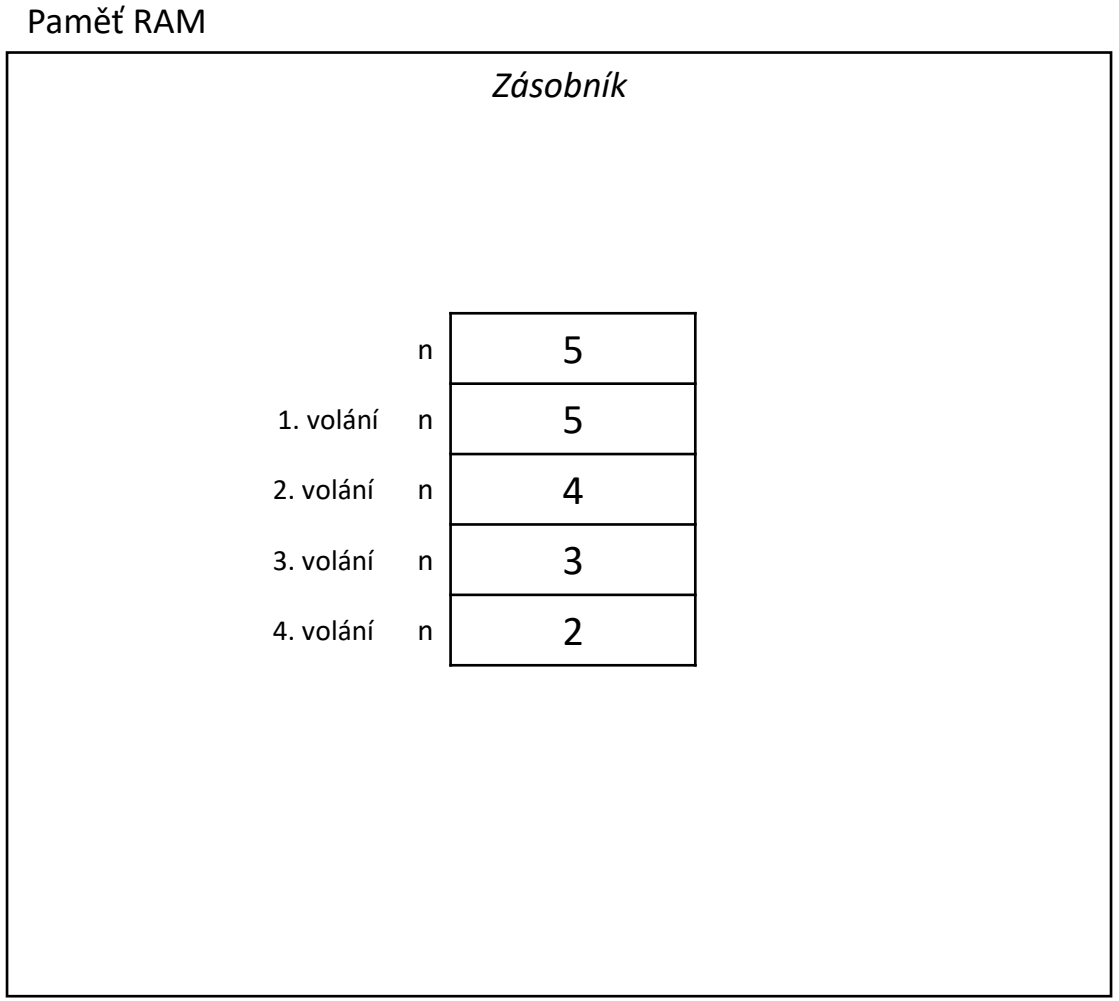
Faktoriál rekurzivní

```
static int Faktorial(int n)
{       2
    if (n == 1)
        return 1;
    else
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



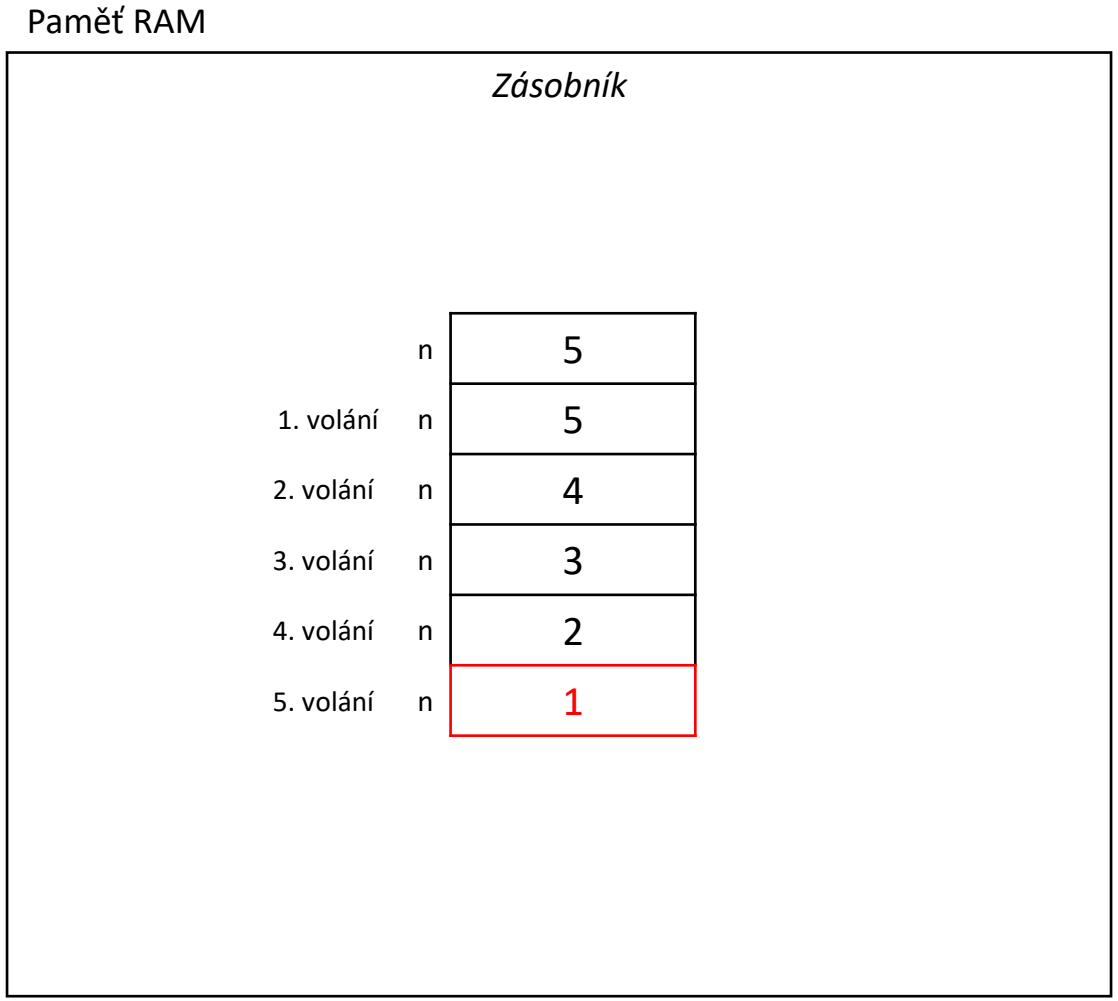
Faktoriál rekurzivní

```
static int Faktorial(int n)
{
    if (n == 1)
        return 1;
    else                          1
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



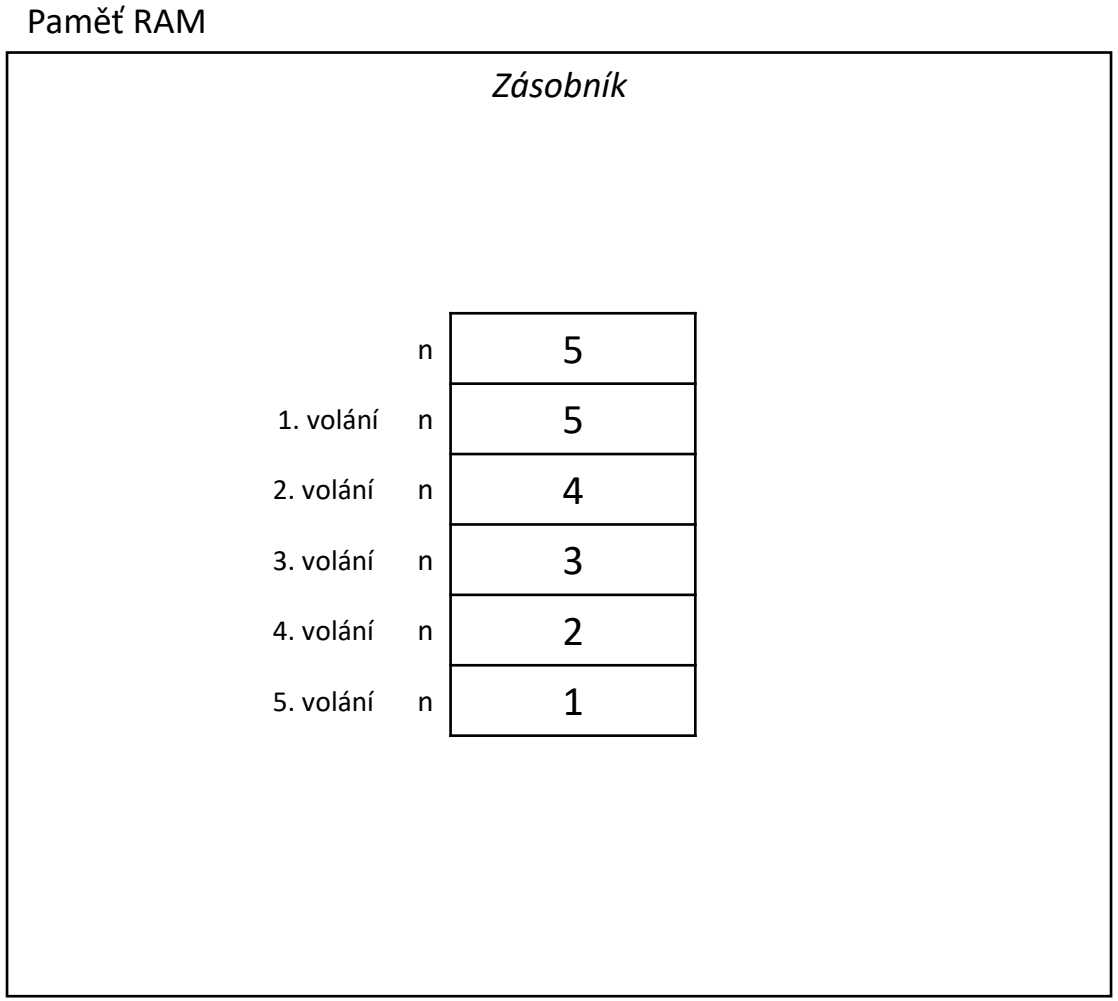
Faktoriál rekurzivní

```
static int Faktorial(int n)
{
    if (n == 1)
        return 1;
    else
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



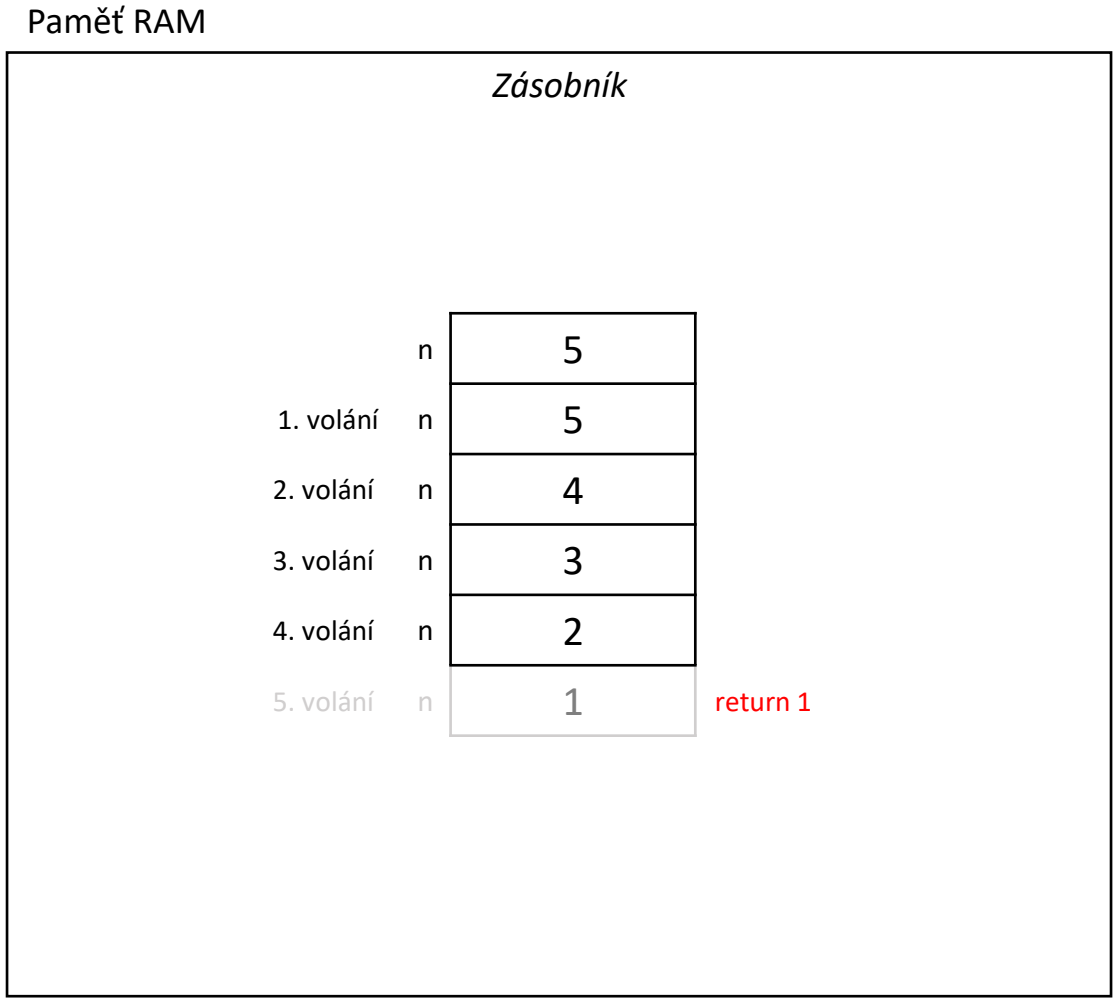
Faktoriál rekurzivní

```
static int Faktorial(int n)
{       1
    if (n == 1)
        return 1;
    else
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



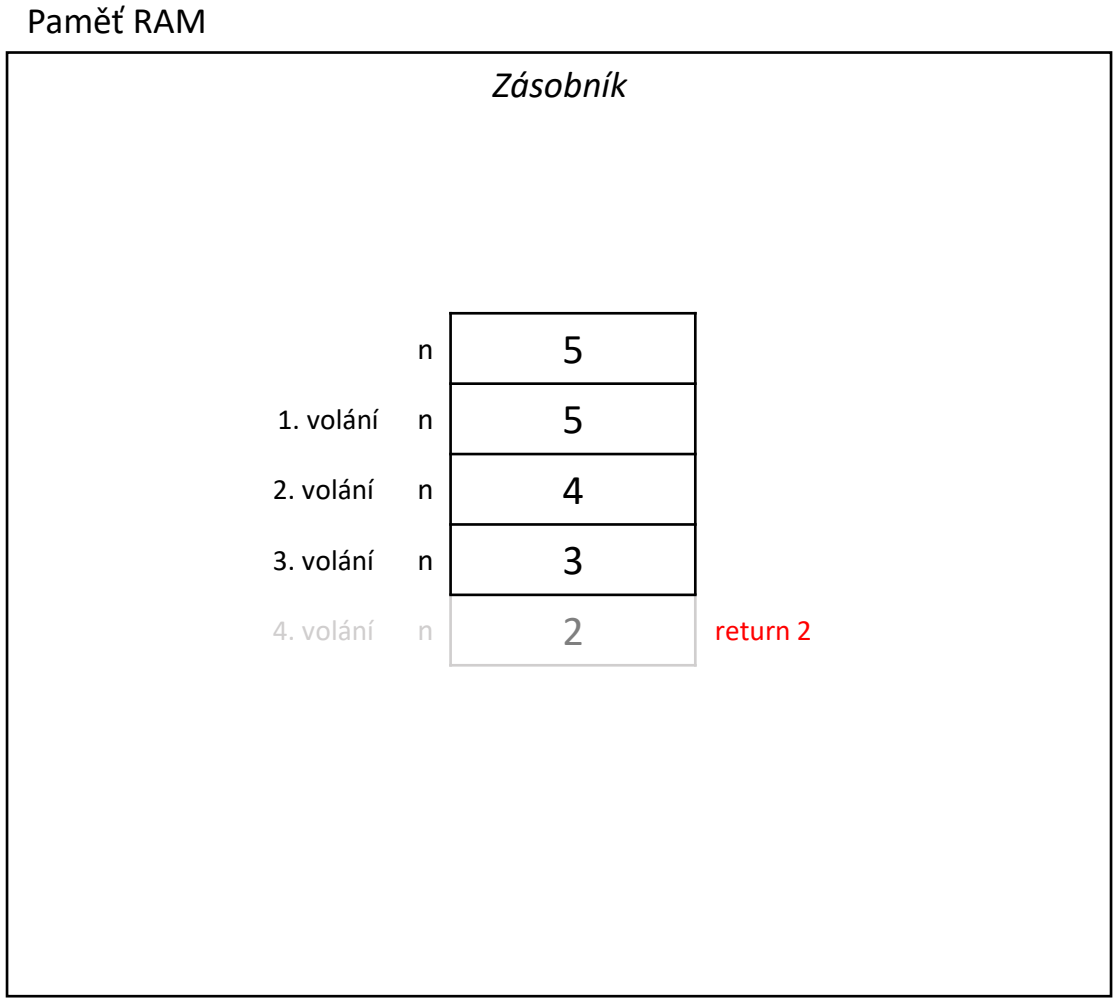
Faktoriál rekurzivní

```
static int Faktorial(int n)
{
    if (n == 1)
        return 1;
    else
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



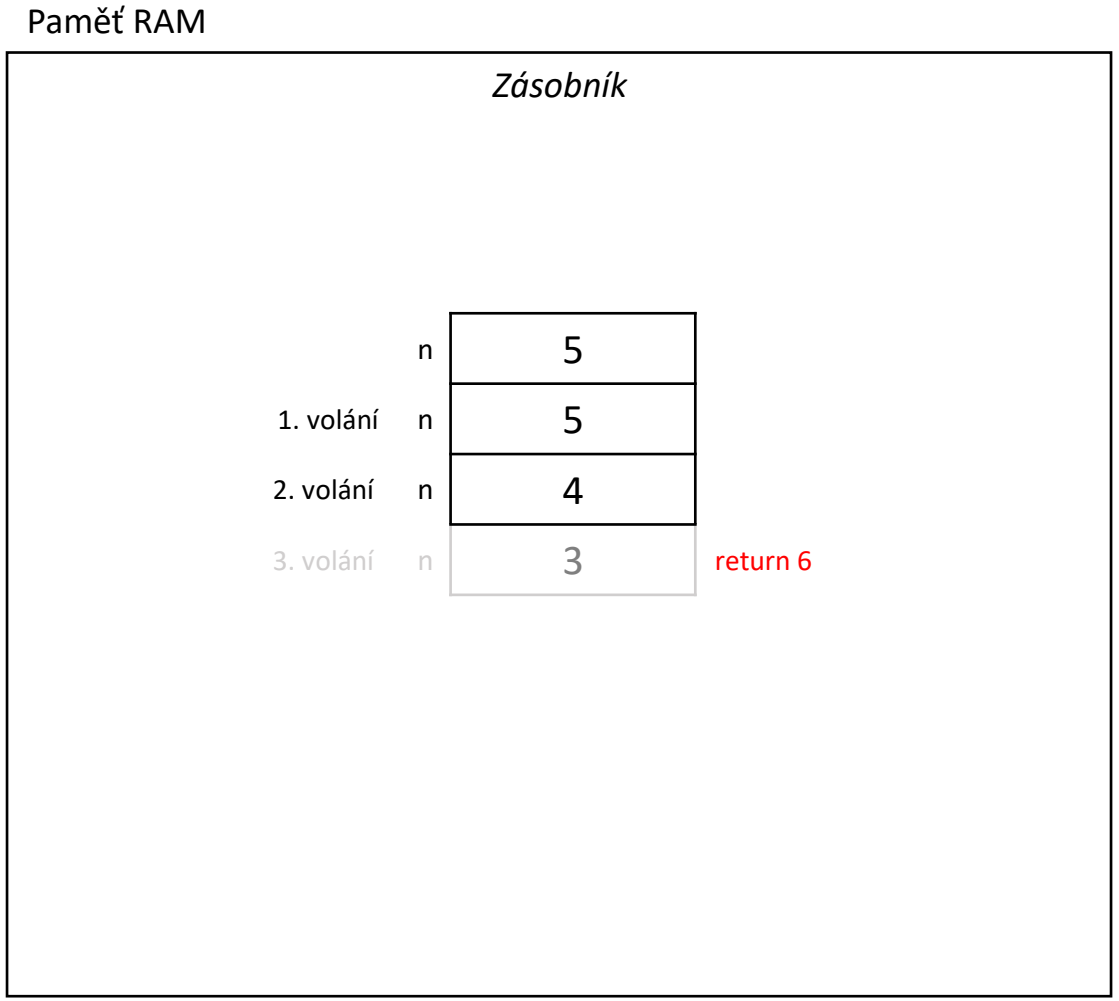
Faktoriál rekurzivní

```
static int Faktorial(int n)
{
    if (n == 1)
        return 1;
    else        2       1
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



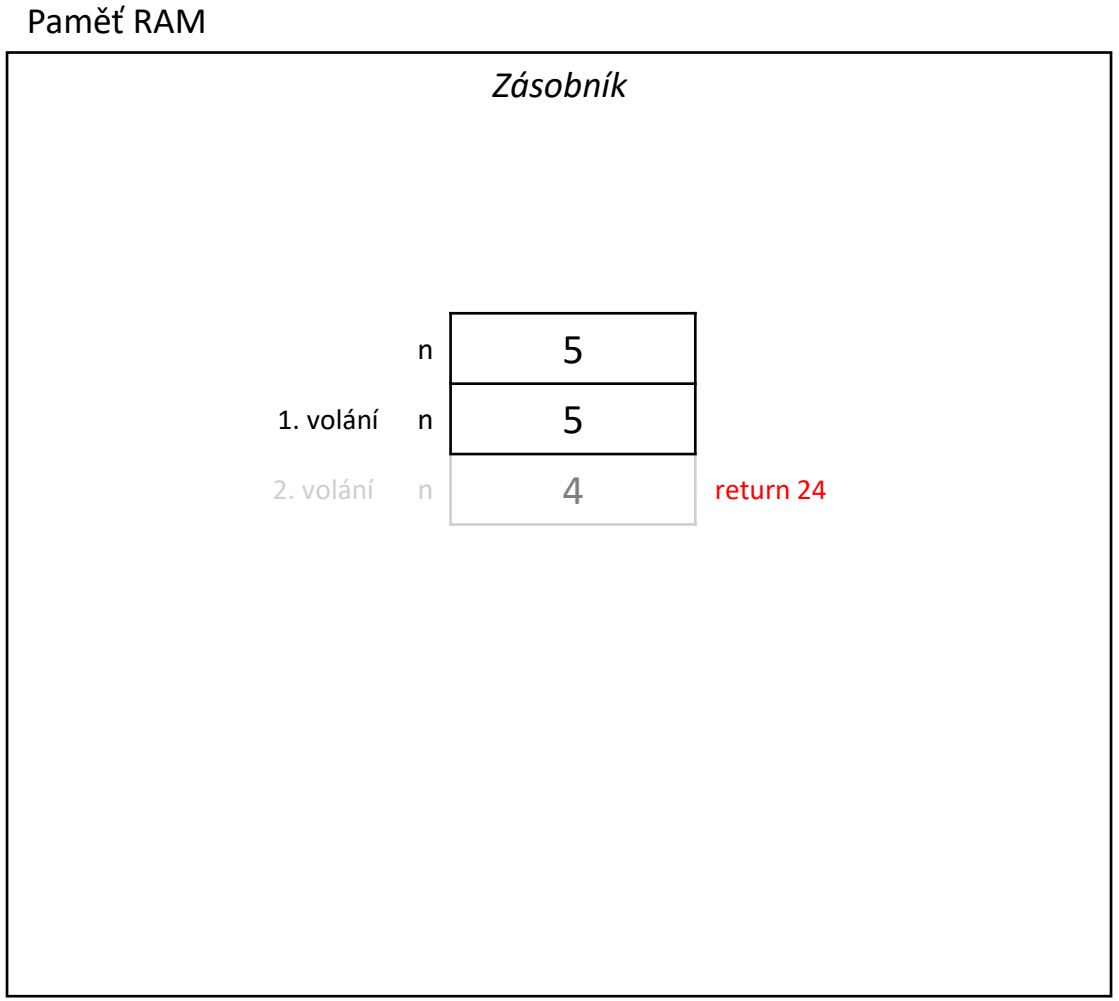
Faktoriál rekurzivní

```
static int Faktorial(int n)
{
    if (n == 1)
        return 1;
    else        3       2
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



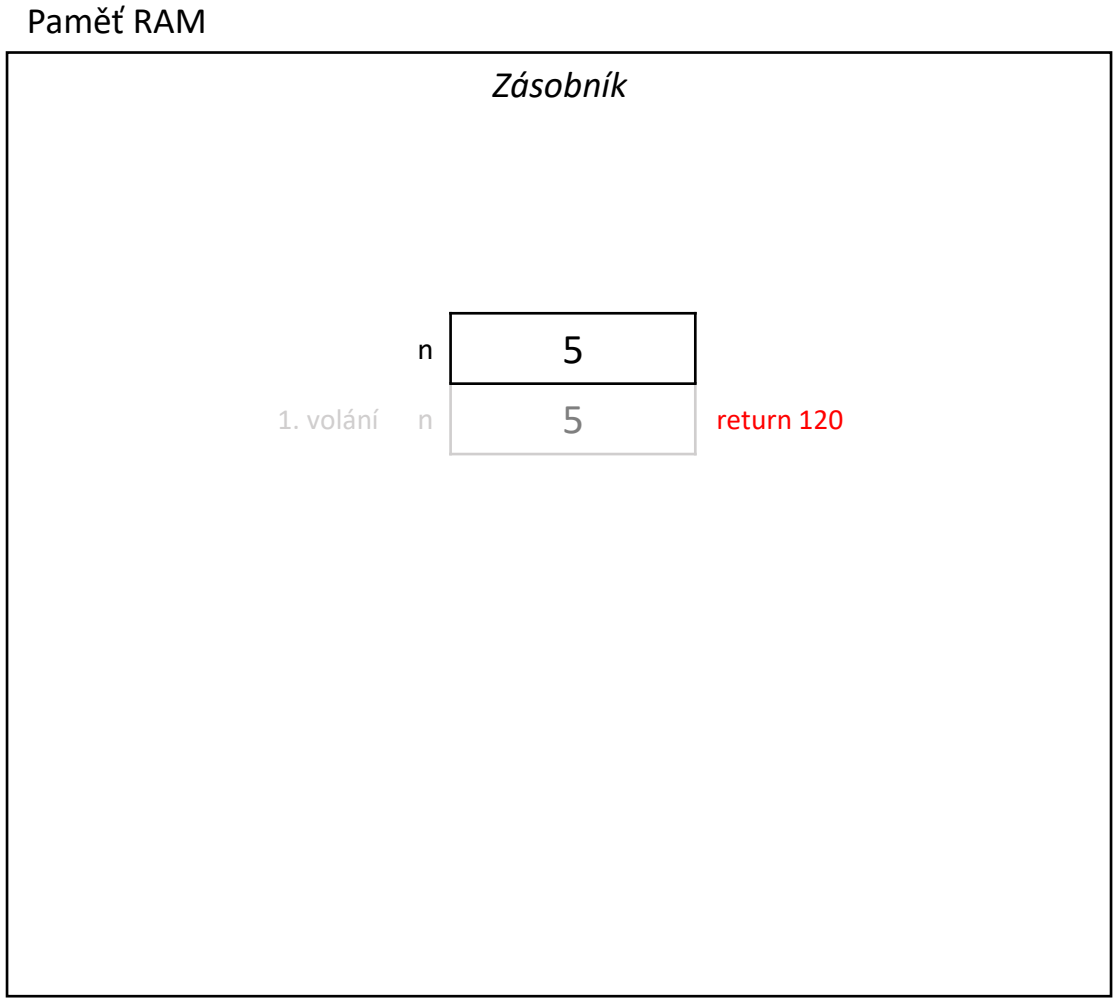
Faktoriál rekurzivní

```
static int Faktorial(int n)
{
    if (n == 1)
        return 1;
    else        4       6
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



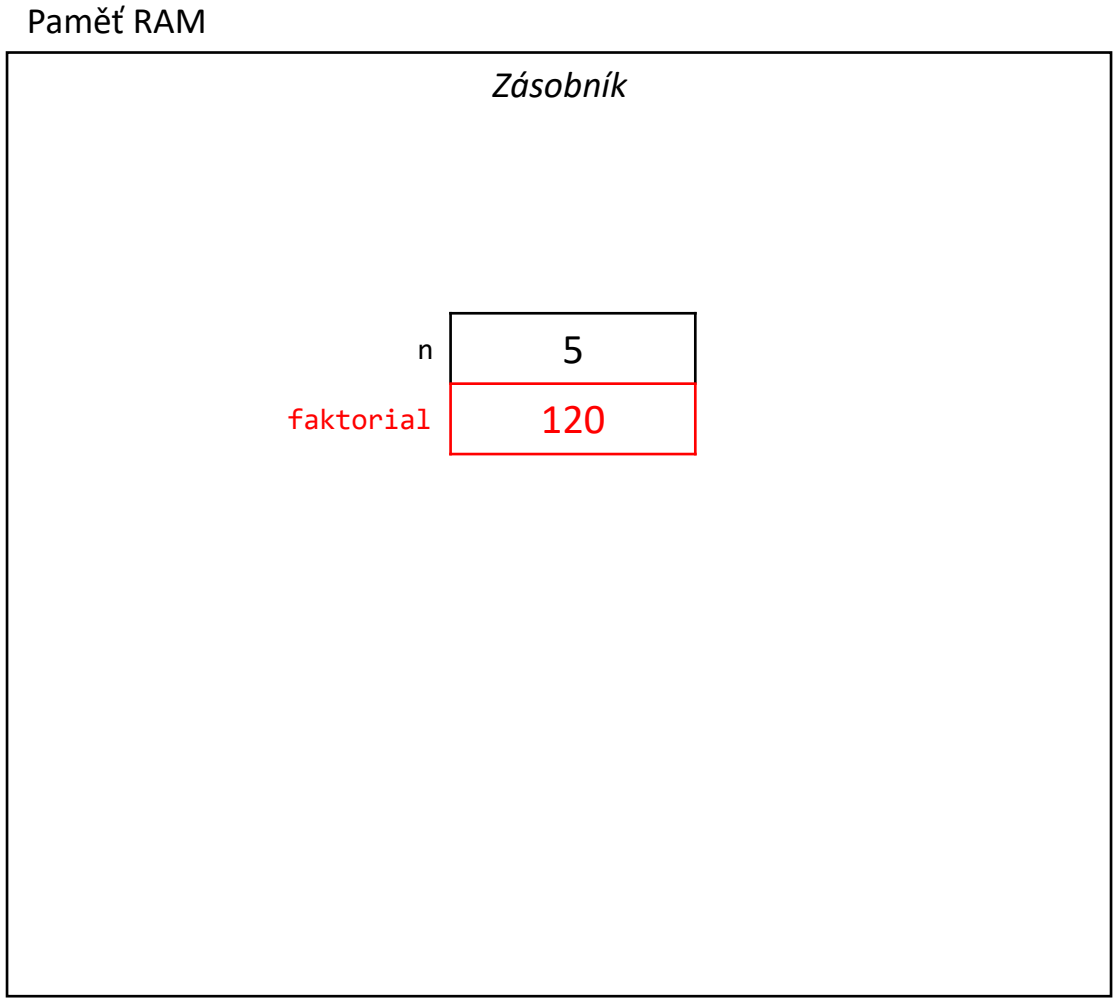
Faktoriál rekurzivní

```
static int Faktorial(int n)
{
    if (n == 1)
        return 1;
    else        5       24
        return n * Faktorial(n - 1);
}

static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



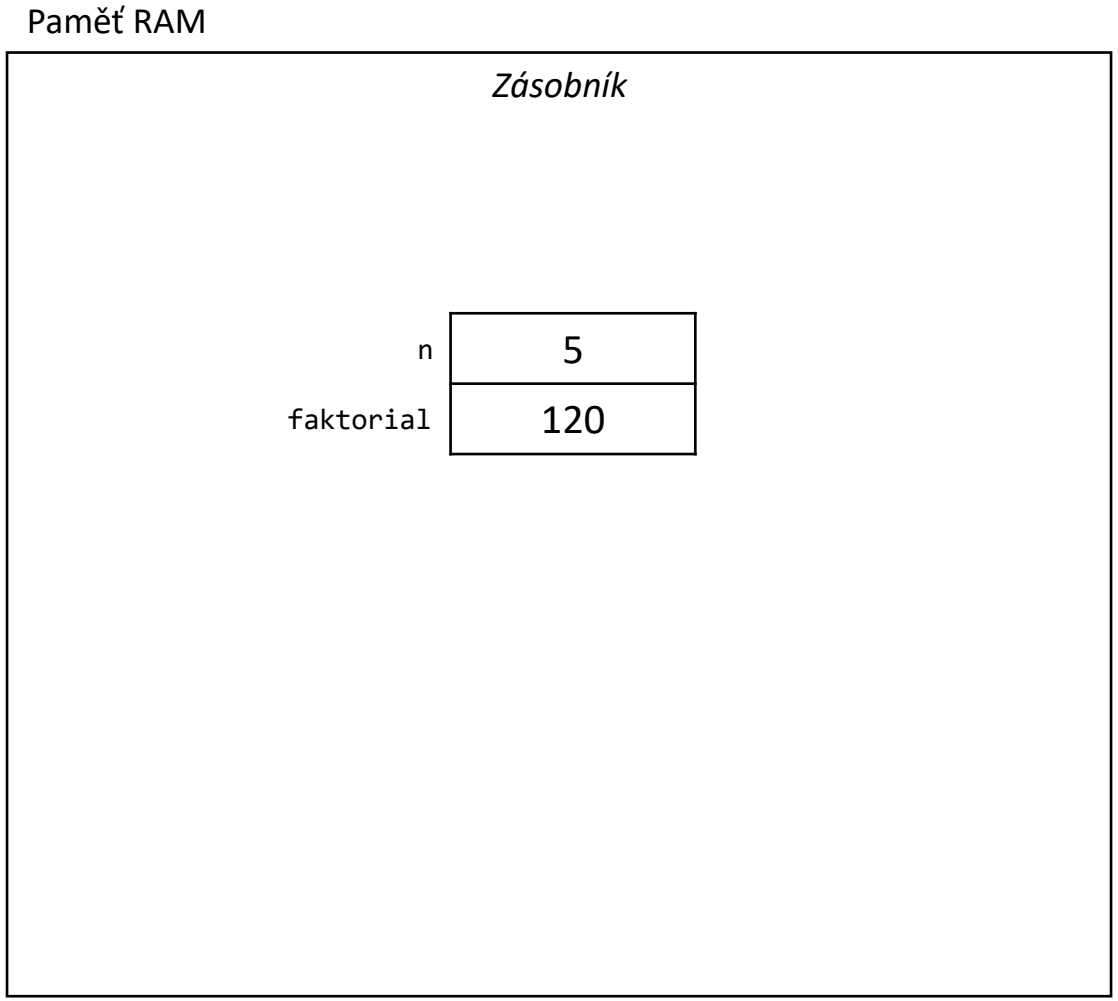
Faktoriál rekurzivní

```
static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);


}
```



Faktoriál rekurzivní

```
static void Main(string[] args)
{
    int n = 5;

    int faktorial = Faktorial(n);

    Console.WriteLine(faktorial);
}
```



Použité zdroje

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

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