

# Pole a kolekce

## 5. cvičení

Jiří Zacpal

KMI/ZP3CS – Základy programování 3 (C#)

# Deklarace pole

- syntaxe:

```
typ[] identifikátor;
```

- vytvoření instance pole:

```
identifikátor = new typ[velikost];
```

- velikost pole nemusí být stanovena konstantou

- inicializace pole:

```
typ[] identifikátor = new  
typ[velikost]{hodnota1, ..., hodnota  
n};
```

# Implicitně typovaná pole pole

- pokud pole ihned inicializujeme, nemusíme uvádět typ

- příklad:

```
var jmena=new[] { „Jan“, „Václav“ } ;
```

- všechny hodnoty musí být stejného typu
- inicializace pole:

```
identifikátor new  
typ[velikost]{hodnota1,...,hodnota  
n} ;
```

# Procházení polem

- kromě klasického cyklu for lze použít i foreach

```
int[] císla=new int {1,2,3,4};  
foreach(int c in císla)  
{  
    //zpracovani pole  
}
```

# Kopírování pole

- kromě klasického je možné použít i metodu CopyTo ze třídy System.Array

```
int[] císla=new int[]{1,2,3,4};  
int[] kopie=new int[císla.Length];  
císla.CopyTo(kopie,0);
```

- nebo

```
Array.Copy(císla,kopie,kopie.Length);
```

- nebo

```
int[] kopie=(int[])císla.Clone();
```

# Vícerozměrná pole

- vytvoření:

```
typ[, ] identifikátor=new  
int[vel1,vel2];
```

# Příklad 1



# Kolekce

- datové struktury, kde se pro přístup k prvkům používá celočíselný index
- kolekce uchovávají a vracejí typ object (pole hodnotový typ)



# Třída ArrayList

- dynamické pole
- vytvoření:

```
ArrayList id = new ArrayList();
```

- metody:
  - Add – přidá prvek na konec kolekce  
`id.Add(hodnota)`
  - Insert – přidá prvek na libovolné místo  
`id.Insert(index, hodnota)`
  - Remove – odebere prvek s hodnotou  
`id.Remove(hodnota)`
  - RemoveAt – odebere prvek na místě indexu  
`id.RemoveAt(index)`
  - Count – vlastnost s počtem prvků  
`id.Count`

# Třída Queue

- fronta
- vytvoření:

```
Queue id = new Queue();
```

- metody:
  - Enqueue – přidá prvek do fronty  
`id.Enqueue(hodnota)`
  - Dequeue – odebere prvek z fronty a tento prvek vrátí  
`id.Dequeue()`
  - Count – vlastnost s počtem prvků  
`id.Count`

# Třída Stack

- zásobník
- vytvoření:

```
Stack id = new Stack();
```

- metody:
  - Push – přidá prvek do zásobníku  
`id.Push(hodnota)`
  - Pop – odebere prvek ze zásobníku a tento prvek vrátí  
`id.Pop()`
  - Count – vlastnost s počtem prvků  
`id.Count`

# Třída Hashtable

- hashovací tabulka
- klíče v tabulce musí být unikátní
- vytvoření:

```
Hashtable id = new Hashtable();
```

- práce se strukturou:
  - vložení prvku  
`id[Klíč]=hodnota`
  - Pomocí metody `ContainsKey` lze otestovat, zda klíč v tabulce je
  - Pro průchod tabulkou se používá objekt typu `DictionaryEntry` (má vlastnosti `Key` a `Value`)

# Příklad 2

