

# Wykład 3

## Tablice i mechanizm indeksowania

**dr inż. Maciej Kusy**

Katedra Podstaw Elektroniki  
Wydział Elektrotechniki i Informatyki  
Politechnika Rzeszowska

Programowanie w języku C#

# Plan wykładu

- Tablice:
  - deklaracja, wartości domyślne elementów tablic
  - inicjalizacja (bezpośrednia, poprzez operator **new**)
  - nowy sposób inicjalizowania tablic jednorodnych (**var**)
  - słowo kluczowe **params**
  - tablice wielowymiarowe prostokątne i nieregularne
- Mechanizm indeksowania (*indeksery*)
  - składnia
  - przykład

# Pojęcie tablicy

Tablica (*ang. array*) to kolekcja, która w sposób zorganizowany przechowuje dane (zbiór obiektów) tego samego typu.

Matematycznym odpowiednikiem tablicy jest ciąg.

Dostęp do poszczególnego elementu tablicy uzyskuje się poprzez odwołanie się do nazwy tablicy i odpowiedniej wartości indeksu ( $0, \dots, n-1$ ) umieszczonego w `[]`.

Nazwa tablicy jest jednocześnie jej adresem w pamięci.

Rozmiar tablicy może być ustalony z góry (statycznie), bądź w trakcie wykonywania programu (dynamicznie).

W C# tablica jest obiektem typu referencyjnego pochodnym po **System.Array**, który implementuje interfejs **Enumerable**:

- tablice przechowują swój rozmiar (właściwość **Length**),
- przez tablice można przechodzić wyk. instrukcję **foreach**.

# Deklaracja tablic

**Typ[] nazwaTablicy;**

Przykład:

**int[] tablicaInt;**

Technicznie jest to deklaracja zmiennej **tablicaInt**

Zmienna (obiekt) przechowująca referencję wskazującą na anonimową tablicę liczb całkowitych

Stworzenie egzemplarza tablicy:

**tablicaInt = new int[5];**

# Wartości domyślne elementów tablicy

```
Pies[] tablicaPsow = new Pies[5];
```

```
int[] tablicaInt = new int[5];
```

Typ	Domyślna wartość
typy liczbowe ( <b>int</b> , <b>long</b> itd.)	<b>0</b>
<b>bool</b>	<b>false</b>
<b>char</b>	<b>'\0'</b>
<b>string</b>	<b>null</b>
obiekty typów wartościowych	domyślna inicjalizacja pól składowych
obiekty typów referencyjnych	<b>null</b>

# Inicjalizacja elementów tablicy

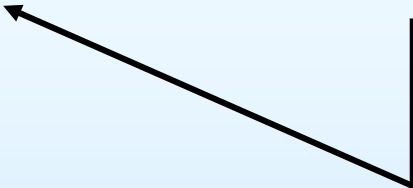
```
int[] tablicaInt = {1, 2, 3};
```



Krótszy i prostszy zapis

```
int[] tablicaInt = new int[] {1, 2, 3};
```

```
int[] tablicaInt = new int[3] {1, 2, 3};
```



Definicja za pomocą inicjalizatora tablicy – jawne użycie operatora **new** oraz podanie długości.

```
Object[] tablicaObj = {1, 'a', "program"};
```

# Nowy sposób inicjalizacji tablic

W języku **C# 3.0** wprowadzono nowy sposób inicjalizacji tablic. Jeżeli tablice są **jednorodne** (mają tego samego typu elementy), to można je zainicjalizować w następujący sposób:

```
var tablicaInt = new [] {1, 2, 3};
```

```
var tablicaString = new []  
    {  
        "element1",  
        "element2",  
        "element3"  
    };
```

## Słowo kluczowe **params**

**params** – pozwala przekazać do metody dowolną liczbę parametrów bez konieczności jawnego tworzenia tablicy; bardzo przydatne przy wyświetlaniu zawartości tablicy za pomocą pętli **foreach**.

```
int[] tablicaInt = new int[3]{1,2,3};
```

```
void PrzeslijElementy(params int[] t)  
{  
}
```

```
obiekt.PrzeslijElementy(tablicaInt);
```

```
obiekt.PrzeslijElementy(1,2,3);
```



# Mechanizm indeksowania

**Mechanizm indeksowania** umożliwia dostęp do elementów kolekcji klasy za pomocą zwykłej składni używanej w tablicach (`[]`).

Mechanizm ten jest właściwością i zawiera akcesory **get** oraz **set**, które pozwalają określić jego działanie.

Interpretacja: przeładowany operator `[]` (C++).

# Składnia mechanizmu indeksowania

```
typ this [typ_argumentu arg] {get; set;
```

**typ** – typ argumentu zwracanego i ustawiającego (**value**)

**this** – referencja wskazująca na obiekt, w którym znajduje się mechanizm indeksowania

**typ\_argumentu** – rodzaj argumentu, który może być używany jako indeks kolekcji zawierającej docelowe obiekty (zwykle **int**, można również **string**)

**arg** – nazwa argumentu

akcesory **get** i **set** muszą zostać zdefiniowane

# Mechanizm indeksowania – przykład

```
class ListaLancuchow
{
    private string[] lancuchy;
    public string this [int index]
    {
        get
        {
            return lancuchy[index];
        }
        set
        {
            lancuchy[index] = value;
        }
    }
}
```

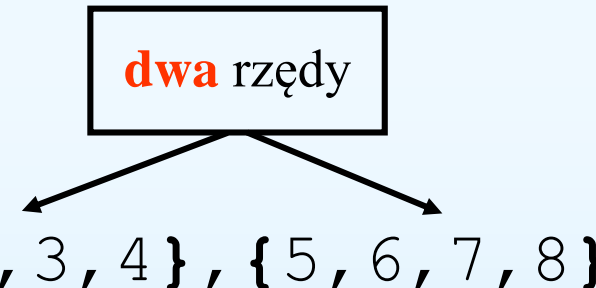
# Tablice wielowymiarowe

Tablica wielowymiarowa – kolekcja posiadająca dwa lub więcej wymiarów: przykład – tablica **prostokątna** W tablicach prostokątnych wszystkie rzędy mają taką samą długość.

Deklaracja tablicy dwuwymiarowej (prostokątnej):

```
int[, ] tablica = new int[2, 3];
```

```
int[, ] tablica = {  
    ↑  
    [2, 4]          {1, 2, 3, 4}, {5, 6, 7, 8}  
    };
```



Deklaracja tablicy wielowymiarowej (3):

```
int[, , ] szescian = new int[3, 3, 3];
```

# Tablice wielowymiarowe (nie)regularne

**Tablica nieregularna** – tablica tablic, z których każda może mieć inną długość: poszczególne rzędy tablicy nie muszą mieć tej samej długości.

Definicja tablicy nieregularnej:

```
int[][] tablica1 = new int[2][];  
tablica1[0] = new int[5];  
tablica1[1] = new int[3];
```



dwa rzędy  
liczba kolumn ?

Definicja tablicy regularnej:

```
int[][] tablica2 = new int[2][];  
tablica2[0] = new int[4];  
tablica2[1] = new int[4];
```