
Rozdział 3

Laboratorium 3

3.1. Tablice

Tablica jest strukturą danych zawierającą zmienne tego samego typu. CLR środowiska .NET Framework wspiera tworzenie tablic jedno oraz wiele wymiarowych.

3.1.1. Tablice w VB.NET

Każda tablica VB.NET jest obiektem dziedziczącym z klasy `System.Array`. Tablice deklaruje się w następujący sposób:

```
Dim <identyfikator>(<rozmiar tablicy>) As <typ>
```

Deklaracja tablicy do przechowania 10 wartości typu `Integer`:

```
Dim tablicaLiczbaCalkowitych(9) As Integer
```

Powyższy fragment kodu deklaruje tablicę mogącą przechować 10 wartości typu `Integer`, elementy tablicy indeksowane są od 0 do 9. W tym przypadku rozmiar tablicy jest stały i został podany w nawiasach przy deklaracji zmiennej typu tablicowego.

Możemy również zadeklarować zmienną tablicową bez podawania jej rozmiaru, rozmiar zostanie ustalony podczas inicjalizacji tablicy:

```
Dim tab() As Integer  
tab = New Integer() {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

Powyższy kod deklaruje najpierw zmienną `tab` jako tablicę liczb typu `Integer`, następnie w nawiasach klamrowych występują oddzielone przecinkami kolejne elementy tablicy.

Kiedy deklaracja i inicjalizacja tablicy są rozdzielone, inicjalizację trzeba wykonać podając w nawiasach klamrowych elementy tablicy.

Odwołanie się do elementu tablicy odbywa się poprzez operator indeksowania (`i`) gdzie `i` oznacza indeks elementu tablicy do którego chcemy się odwołać. Pamiętajmy, że indeks pierwszego elementu to 0

Przykładowy program korzystający z tablicy:

```
Sub Main()  
    Dim i As Integer = 0  
    Dim tab() As Integer  
    tab = New Integer() {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}  
    For i = 0 To 9  
        Console.WriteLine(tab(i))  
    Next  
End Sub
```

Powyższy program jest dość prosty, ale musieliśmy na sztywno umieścić rozmiar tablicy w pętli `For`. Wcześniej wspomnieliśmy, że wszystkie klasy `.NET Framework` dziedziczą z klasy `System.Array`, w której znajduje się wiele przydatnych metod i właściwości. jedną z bardzo przydatnych właściwości klasy `System.Array` jest `Length` gdzie przechowywana jest informacja o rozmiarze tablicy. Możemy napisać kod dla powyższego przykładu zastępując sztywny rozmiar tablicy użyty w pętli `For` odwołaniem do właściwości `Length`

```
Sub Main()  
    Dim i As Integer = 0  
    Dim tab() As Integer  
    tab = New Integer() {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}  
    For i = 0 To tab.Length - 1  
        Console.WriteLine(tab(i))  
    Next  
End Sub
```

Ponieważ właściwość `Length` zawiera rozmiar tablicy a elementy indeksowane są od 0, w pętli `for` pojawia się `Length - 1`

Możemy teraz zaprezentować inną formę funkcji `Main` od której rozpoczyna się wykonanie aplikacji konsolowej napisanej w języku `VB.NET`.

```
Sub Main(ByVal args As String())  
  
End Sub
```

Zauważmy, że procedura `Main` posiada parametr o nazwie `args` (nazwa zmiennej może być dowolna) typu tablica wartości typu `String`. Dodatkowo pojawiło się słowo kluczowe `ByVal`, które zostanie omówione później.

Zmienna `args` będzie zawierała parametry przekazane naszej aplikacji z linii poleceń. Możemy napisać program, który wyświetli wszystkie parametry z jakimi została wywołana nasza aplikacja:

```
Sub Main(ByVal args As String())
    Dim i As Integer
    For i = 0 To args.Length - 1
        Console.WriteLine(args(i))
    Next
End Sub
```

Powyższy przykład wykorzystuje pętlę typu `For`. Język `VB.NET` posiada jednak wygodniejszą konstrukcję do iteracji poprzez elementy tablicy, mianowicie konstrukcję `For Each`

```
For Each <identyfikator> In <tablica lub kolekcja>
    <blok instrukcji>
Next
```

Zamieńmy kod wyświetlający elementy tablicy tak, aby wykorzystywał konstrukcję `For Each`:

```
Sub Main()
    Dim tab As Integer()
    tab = New Integer() {1, 2, 3, 4, 5}
    Dim i As Integer
    For Each i In tab
        Console.WriteLine(i)
    Next
End Sub
```

Zmienna `i` w każdej iteracji zawiera wartość kolejnego elementu tablicy `tab`. Zmienna `i` musi być tego samego typu co elementy tablicy (w naszym przypadku `Integer`).

- Zmienna `i` może być wykorzystana tylko do odczytania wartości elementu tablicy, nie można przy jej pomocy zmienić zawartości tablicy. Jeżeli zachodzi konieczność modyfikacji elementów tablicy podczas iteracji trzeba skorzystać z konstrukcji `For...Next`,
- Konstrukcja `For Each` służy do iteracji po elementach tablicy jak i kolekcji. Pojęcie kolekcji zostanie omówione później,
- Klasa `String` jest także kolekcją znaków.

Możemy zastosować konstrukcję `For Each` do iteracji poprzez kolejne znaki wchodzące w skład łańcucha znaków. Poniższy program wypisuje poszczególne litery wchodzące w skład napisu:

```

Sub Main()
    Dim napis As String = "Jakiś napis"
    Dim c As Char
    For Each c In napis
        Console.WriteLine(c)
    Next
End Sub

```

Poniższy kod pokazuje zastosowanie tablicy tablic, do wyświetlenia kalendarza.

```

1  Sub Main()
2      Dim i As Integer
3      Dim Miesiace(), Dni() As String
4      Dim Kalendarz()() As Integer = New Integer(11)() {}
5      Dim Miesiac, Dzień As Integer
6      Miesiace = New String(11) {"Styczeń", "Luty", "Marzec", -
7                                "Kwiecień", "Maj", "Czerwiec", -
8                                "Lipiec", "Sierpień", "Wrzesień", -
9                                "Październik", "Listopad", "Grudzień"}
10     Dni = New String(6) {"Pn", "Wt", "Śr", "Cz", -
11                          "Pt", "So", "Nie"}
12     For Miesiac = 0 To Miesiace.Length - 1
13         Dzień = DateTime.DaysInMonth(Year(Now), Miesiac + 1)
14         Kalendarz(Miesiac) = New Integer(Dzień - 1) {}
15         For i = 0 To Dzień - 1
16             Kalendarz(Miesiac)(i) = i + 1
17         Next i
18     Next Miesiac
19     Dim d As DateTime
20     For Miesiac = 0 To Miesiace.Length - 1
21         Console.WriteLine(Miesiace(Miesiac))
22         d = New DateTime(Year(Now), Miesiac + 1, 1)
23         i = d.DayOfWeek      ' 0 - nd, 1 - pn ...
24         If (i = 0) Then
25             i = 6
26         Else
27             i = i - 1
28         End If
29         For Dzień = 0 To Dni.Length - 1
30             Console.Write("{0} ", Dni(Dzień))
31         Next
32         Console.WriteLine()
33         For Dzień = 0 To i - 1
34             Console.Write(" ".PadRight(3))
35         Next
36         For Dzień = 0 To Kalendarz(Miesiac).Length - 1
37             Console.Write("{0:00} ", Kalendarz(Miesiac)(Dzień))
38             If (i + 1) Mod 7 = 0 Then
39                 Console.WriteLine()
40             End If
41             i = i + 1
42         Next Dzień
43         Console.WriteLine()
44         Console.WriteLine()
45     Next Miesiac
46 End Sub

```

3.1.2. Tablice w C#

Każda tablica C# jest obiektem dziedziczącym z klasy `System.Array`. Tablice deklaruje się w następujący sposób:

```
<typ>[<rozmiar tablicy >] <identyfikator >;
```

Deklaracja tablicy do przechowania 10 wartości typu `Integer`:

```
int [10] tablicaLiczbCalkowitych;
```

Powyższy fragment kodu deklaruje tablicę mogącą przechować 10 wartości typu `Integer`, elementy tablicy indeksowane są od 0 do 9. W tym przypadku rozmiar tablicy jest stały i został podany w nawiasach przy deklaracji zmiennej typu tablicowego.

Podawanie rozmiaru w deklaracji tablicy różni się w języku C# w stosunku do języka VB.NET. Deklarując tablicę w VB.NET podawaliśmy maksymalny indeks tablicy, natomiast w C# podajemy ilość elementów.

Tak jak w przypadku VB.NET możemy zadeklarować tablicę bez podawania jej rozmiaru, a następnie dokonać jej inicjalizacji:

```
int [] tab;
tab = new int [10] {1,2,3,4,5,6,7,8,9,10};
```

Możemy również utworzyć tablicę dynamicznie nie inicjalizując jej:

```
int [] tab;
tab = new int [10];
```

Operatorem indeksowania w przypadku C# jest operator `[]`. przykładowy program deklaruje zmienną `tab` jako tablicę liczb typu `int`, następnie tworzy nowy obiekt reprezentujący tablicę i wypełnia elementy tablicy kolejnymi liczbami by na końcu używając pętli `for` wyświetlić zawartość tablicy:

```
static void Main(string [] args)
{
    int [] tab;
    tab = new int [10];
    for (int i=0; i<tab.Length; i++)
        tab[i]=i+1;
    for (int i = 0; i < tab.Length; i++)
        Console.WriteLine(tab[i]);
}
```

Tak samo jak w przypadku VB.NET w języku C# istnieje konstrukcja `foreach` służąca do wykonania operacji iteracji na elementach tablicy lub kolekcji.

```
foreach (<typ> <identyfikator> in <tablica lub kolekcja >)
    <blok instrukcji >
```

Analogiczny kod jak w przypadku VB.NET używający konstrukcji `foreach`

```
static void Main(string [] args)
{
    string s = "Jakiś napis";

    foreach (char c in s)
        Console.WriteLine(c);
}
```

Zauważmy, że deklaracja funkcji `Main` języka C# od której rozpoczyna się wykonanie aplikacji konsolowej, zawiera argument o nazwie `args` typu `string []`.

Tak jak w przypadku VB.NET jest to tablica łańcuchów znaków zawierająca parametry przekazane w linii poleceń podczas uruchamiania programu.

Przykładowy kod wykorzystuje pętlę `foreach` do wyświetlenia parametrów przekazanych z linii poleceń:

```
static void Main(string [] args)
{
    if (args.Length < 1)
        Console.WriteLine("Brak parametró wywołania!");
    else
    {
        Console.WriteLine("Przekazana parametry:");
        foreach (string c in args)
        {
            Console.WriteLine(c);
        } //foreach
    } //else
} //Main
```

Tablice wielowymiarowe deklaruje się podobnie do tablic jednowymiarowych:

```
int [,] tablica2Wymiarowa;
tablica2Wymiarowa = new int [10,10];
```

Kod wykorzystujący tablice tablic:

```

1  static void Main(string[] args)
2  {
3      int Dzień, Miesiąc, i;
4      string[] Miesiące, Dni;
5      int[][] Kalendarz = new int[12][];
6      Miesiące = new string[12] {"Styczeń", "Luty", "Marzec", "Kwiecień",
7                                "Maj", "Czerwiec", "Lipiec", "Sierpień",
8                                "Wrzesień", "Październik", "Listopad",
9                                "Grudzień"};
10     Dni = new string[7] {"Pn", "Wt", "Śr", "Cz", "Pt", "So", "Nie"};
11     for ( Miesiąc = 0; Miesiąc < Miesiące.Length; Miesiąc++)
12     {
13         Dzień = DateTime.DaysInMonth(DateTime.Now.Year, Miesiąc + 1);
14         Kalendarz[Miesiąc] = new int[Dzień];
15         for (i = 0; i < Dzień; i++)
16             Kalendarz[Miesiąc][i] = i + 1;
17     } // for Miesiąc
18     for (Miesiąc = 0; Miesiąc < Miesiące.Length ; Miesiąc++)
19     {
20         Console.WriteLine(Miesiące[Miesiąc]);
21         DateTime d = new System.DateTime(DateTime.Now.Year, Miesiąc + 1, 1);
22         i = (int)d.DayOfWeek;
23         i = i == 0 ? 6 : i - 1;
24         // 0 - nd, 1 - pn ...
25         for (Dzień = 0; Dzień < Dni.Length; Dzień++)
26             Console.Write("{0} ", Dni[Dzień]);
27         Console.WriteLine();
28         Console.Write("".PadRight(3*i));
29         for (Dzień = 0; Dzień < Kalendarz[Miesiąc].Length; Dzień++)
30         {
31             Console.Write("{0:00} ", Kalendarz[Miesiąc][Dzień]);
32             if ((i++ + 1) % 7 == 0)
33                 Console.WriteLine();
34         }
35         Console.WriteLine();
36         Console.WriteLine();
37     } // for Miesiąc
38 } // Main

```

```

C:\WINDOWS\system33...
Styczeń
Pn Wt Śr Cz Pt So Nie
    01
02 03 04 05 06 07 08
09 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31

Luty
Pn Wt Śr Cz Pt So Nie
    01 02 03 04 05
06 07 08 09 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28

```

Rys. 3.1.1. Wynik działania programu