Vježba 7

Programiranje 1 (550) - ak. god. 2024./2025.

Ostale zbirke

ArrayList

Operacije nad nizovima se generalno izvode brže nego operacije nad ostalim zbirkama, kao što i nizovi zauzimaju manje memorijskog prostora od ostalih zbirki, te su tipski sigurni. Međutim nizovi ne mogu mijenjati veličinu jednom kada su deklarirani, dok neki od drugih zbirki, poput `ArrayList` to omogućuju.
Ova zbirka ima svojstva `Capacity` i `Count`. Prvo pokazuje količinu prostora za pohranu podataka, a drugo stvarni broj elemenata koji su pohranjeni u zbirci. Kapacitet je moguće povećati, ako se lista širi, on se udvostručuje.

VB

```
Dim lista As ArrayList = New ArrayList(1)
Console.WriteLine(lista.Capacity) ' 1
lista.Add(1)
lista.Add(2)
Console.WriteLine(lista.Capacity) ' 2
lista.Add(3)
Console.WriteLine(lista.Capacity) ' 4
Console.WriteLine(lista.Count) ' 3
lista.RemoveAt(1) ' Brise element na indeksu 1
Console.WriteLine(lista.Capacity) ' 4
Console.WriteLine(lista.Count) ' 2
```

Stack (stog)

Stog je podatkovna struktura kod koje se podatak koji je zadnji dodan prvi uzima u obradu. Kažemo da se elementi na stog dodaju i brišu po principu LIFO (*last in, first out*). Postoje samo dvije procedure za rukovanje stogom, `Push` i `Pop`. Prva postavlja element na vrh stoga, dok ga druga uklanja i vraća. Postoji još svojstvo `Count` pomoću kojeg možemo saznati koliko je elemenata na stogu.

VB

```
Dim stog As Stack = New Stack()
stog.Push(4)
stog.Push(7)
stog.Push(11)
Console.WriteLine(stog.Count) ' Velicina 3
```

```
Console.WriteLine(stog.Pop()) ' Uklanja i ispisuje 11
Console.WriteLine(stog.Pop()) ' Uklanja i ispisuje 7
Console.WriteLine(stog.Count) ' Velicina 1
```

Queue (red)

Red je podatkovna struktura kod kojeg se elementi dodaju na kraj reda, a uklanjaju s početka, što znači da element koji se prvi doda u red, prvi se i uklanja. Ovaj princip se naziva FIFO (*first in, first out*). Procedura za postavljanje elemenata u red je `Enqueue`, a za uklanjanje iz reda `Dequeue`.

VB

```
Dim q As Queue = New Queue()
q.Enqueue(4)
q.Enqueue(7)
q.Enqueue(11)
Console.WriteLine(q.Count) ' Velicina 3
Console.WriteLine(q.Dequeue()) ' Uklanja i ispisuje 4
Console.WriteLine(q.Dequeue()) ' Uklanja i ispisuje 7
Console.WriteLine(q.Count) ' Velicina 1
```

Nabrajanja

Nabrajanje (engl. *enumeration*) se koristi za definiranje skupa konstanti koje su na neki način povezane. Tekstualnim vrijednostima koje se nalaze u nabrajanju se pridružuje cjelobrojna numerička vrijednost (`Byte`, `Short`, `Integer`, `Long`) počevši od 0, ali ju je moguće i ručno definirati:

VB

```
Enum Size as Integer
        ExtraSmall  ' vrijednost 0
        Small       ' vrijednost 1
        Medium      ' ...
        Large = 40
        ExtraLarge = 50
End Enum
```

Vrijednost nabrajanja je moguće pridružiti nekoj varijabli na sljedeći način:

VB

```
Dim velicina as Size = Size.Medium
```

Strukture

Struktura je *vrijednosni tip* podatka kojeg definira korisnik, a koristi se za grupiranje podataka istih ili različitih tipova koji su povezani nekakvom logikom. Ti podaci se onda pohranjuju u memoriju zajedno.
Strukturu definiramo na sljedeći način:

VB

```
[dostupnost] Structure NazivStrukture
        ' clanovi strukture
End Structure
```

`dostupnost` je modifikator koji određuje kojem dijelu koda će ova struktura biti dostupna. Modifikator `Public` kaže da je struktura dostupna cijelom kodu. Ako se modifikator dostupnosti ne navede, podrazumijeva se `Public`.

Polja strukture

Polja strukture su podatkovni članovi koji se definiraju unutar strukture. Sintaksa je ista kao i za definiranje varijabli:

VB

```
Structure Student
        Dim Ime As String
        Public Prezime As String
End Structure
```

U slučaju definirana polja strukture, modifikatori `Public` i `Dim` su ekvivalentni. Postoji još mogućnost korištenja modifikatora `Private`, što onda polja strukture čini nedostupnim vanjskom kodu, ali ih može mijenjati kod (procedure) koje se nalaze unutar strukture.

> **Važno**
>
> Svaka struktura mora sadržavati minimalno jedno polje (podatkovni član).
>
> Polja strukture mogu biti i vrijednosni i referencni tipovi; nije neobično da jedno ili više polja strukture bude niz ili čak neki još kompleksniji objekt.

Varijabla strukture

Varijabla strukture se deklarira na isti način kao što smo do sada deklarirali varijable različitih tipova, a za tip se koristi naziv strukture. Ako koristimo strukturu iz gornjeg primjera, varijablu te strukture možemo definirati sa niže navedenim kodom, te potom pridijeliti vrijednosti poljima te strukture s operatorom `.` (točka) i korištenjem imena polja iza točke:

```vb
Dim s As Student
s.Ime = "Mate"
s.Prezime = "Matić"
```

Kako je struktura vrijednosni tip, kopiranje varijable strukture je *plitko*, što znači da se kopiraju stvarni podaci, a ne upućivanje. Za vrijednosna polja strukture se kopira vrijednost, a za referencna polja strukture se kopira referenca.
Ako se objekt strukture prosljeđuje kao parametar u funkciju, stvara se lokalna kopija, osim ako se prosljeđuje po referenci.

Procedure u strukturama

Strukture mogu imati i procedure unutar sebe, koje se uobičajeno definiraju s ciljem da naprave neku operaciju nad podacima koji se nalaze u poljima strukture. Npr.

```vb
Structure Student
        Public Ime As String
        Public Prezime As String

        Public Function PunoIme() As String
                Return String.Concat(Ime, " ", Prezime)
        End Function
End Structure

Sub Main(args As String())
        Dim s As Student
        s.Ime = "Mate"
        s.Prezime = "Matić"
        Console.WriteLine(s.PunoIme())
End Sub
```

Konstruktor

Konstruktor je specijalna procedura koja se uobičajeno koristi za postavljanje početnih vrijednosti na polja strukture ili klase. Dok je kod struktura njegovo korištenje opcionalno (tj. događa se samo od sebe u pozadini ako ga ne koristi korisnik), kod klasa je obavezno (više o tome u sljedećoj vježbi). Procedura konstruktora u pozadini postavlja sva polja na njihove pretpostavljene vrijednosti prema tipu tog člana.
Konstruktor nam može biti vrlo koristan ako npr. želimo omogućiti da pri deklaraciji varijable tipa naše strukture možemo odmah postaviti početne vrijednosti na njena polja. Koristimo prethodni primjer:

```vb
```

```
Structure Student
        Public Ime As String
        Public Prezime As String

        Public Sub New(ime As String, prezime As String)
                Me.Ime = ime
                Me.Prezime = prezime
        End Sub

        Public Function PunoIme() As String
                Return String.Concat(Ime, " ", Prezime)
        End Function
End Structure
```

Ovaj kod omogućava nam da deklariramo varijablu tipa `Student` i da postavimo početne vrijednosti, sve u jednoj liniji. Procedura `New()` je konstruktor, koji, u ovom slučaju prima dva parametra i njihove vrijednosti postavlja u polja strukture:

VB

```
Dim s As Student = New Student("Pero", "Perić")
```

> **Važno**
>
> Konstruktor mora imati ime `New()` (to je rezervirano ime).

Primijetite unutar konstruktora ključnu riječ `Me`. Ta ključna riječ se koristi da bismo se referirali na određenu instancu strukture (a u sljedećim vježbama i klase) koja se trenutno izvršava. Obavezno se koristi kada imamo konflikt u imenima varijabli strukture i lokalnih varijabli u funkcijama, a preporučeno je koristiti **uvijek**, radi lakše distinkcije radi li se negdje o lokalnim varijablama/funkcijama ili o članovima strukture (ili klase).

Zadaci

1. Implementirajte program u kojem će korisnik unositi znakove, pa ih na kraju ispišite obrnutim redoslijedom nego što su uneseni. Koristite podatkovnu strukturu Stack (stog).
2. Implementirajte funkciju koja će primati niz znakova (string) kao ulaz i koja će provjeriti jesu li zagrade u ulaznom stringu balansirane (tj. Je li svaka lijeva zagrada ima i desnu i jesu li po pravom redoslijedu). Funkcija vraća True/False. Za implementaciju koristite strukturu Stack. Za testiranje funkcije možete koristiti sljedeće ulaze:
   - `[()]{}{[()()]()}` – očekivani izlaz je true
   - `[(])` – očekivani izlaz je false
3. Implementirajte strukturu *Point* koja će se koristiti za opisivanje točke u prostoru). Struktura treba imati tri člana koji predstavljaju x, y i z koordinate točke i trebaju biti

realni brojevi. Implementirajte člansku funkciju *ToString()* tako da ispisuje koordinate u konzoli kao (x,y,z) – zajedno sa zagradama.

4. U strukturi iz prethodnog zadatka kreirati člansku funkciju *DistanceTo()* koja će primati jedan parametar tipa *Point* koji će vraćati euklidsku udaljenost između te dvije točke. Testirajte funkciju tako da u *Main()* proceduri kreirate dvije točke te ispišete njihovu udaljenost korištenjem ove metode.
5. Implementirajte strukturu kojom ćete opisati predmet na fakultetu (sadrži naziv predmeta, broj ECTS bodova i ocjenu). Potom implementirajte funkciju kojoj će se kao parametar proslijediti **niz** objekata opisanih implementiranom strukturom, a koja vraća težinski prosjek ocjena. (**NAPOMENA:** Za računanje težinskog prosjeka možete koristiti implementaciju koju ste napravili na prošlim vježbama, ali je trebate prilagoditi tako da prima parametar kako je navedeno u zadatku).