# Vježba 4

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

---

## Procedure

Procedure služe da bismo izdvojili dio programskog koda u zasebnu cjelinu koja može primati nekakve ulazne podatke, te izvršiti određeni niz naredbi nad tim podacima i eventualno vratiti rezultate izvršavanja tih naredbi kodu koji ih je pozvao. Ovakvim izdvajanjem omogućavamo da taj dio koda možemo koristiti više puta unutar programa bez da ga ponovo pišemo.

U žargonu VB.NET-a najčešće se termin *procedura* koristi za općenito opisivanje dijela koda, a postoji više vrsta procedura: *Sub procedura* može obaviti neki zadatak, ali **ne vraća** nikakvu vrijednost, dok se termin *funkcija* koristi za dio koda koji prima ulaze, te na temelju obrade tih ulaza vraća neke vrijednosti (primijetite analogiju s matematičkim funkcijama). Postoji još termin *metoda*, koji se koristi u objektno orijentiranom programiranju, o čemu će biti riječi u nekim od sljedećih vježbi.

## Sintaksa

**VB**

```vb
[dostupnost] Sub NazivProcedure([listaParametara])
        ' tijelo procedure
End Sub
```

Dostupnost može biti `VB Private`, `VB Friend` ili `VB Public`, a podrazumijevana vrijednost je `VB Public` što znači da su procedure dostupne cijelom kodu programa, pa ako je to ono što nam treba, ključnu riječ `VB Public` možemo izostaviti.

U nastavku je primjer definiranja procedure koja prima dva broja te ispisuje njihov zbroj:

**VB**

```vb
Sub Zbroj(ByVal a As Integer, ByVal b As Integer)
        Console.WriteLine($"Zbroj brojeva {a} i {b} je {a+b}")
End Sub
```

## Pozivanje

Procedure se pozivaju tako da se navede njihovo ime zajedno s listom parametara koji se navode u zagradi iza imena funkcije (ako ih procedura ima). Poslužit ćemo se gornjim primjerom da ilustriramo:

**VB**

```
Sub Main(args As String())
        Dim a, b As Integer
        Console.Write("Unesi prvi broj: ")
        a = Console.ReadLine()
        Console.Write("Unesi drugi broj: ")
        b = Console.ReadLine()
        Zbroj(a, b)
End Sub

Sub Zbroj(ByVal prvi As Integer, ByVal drugi As Integer)
        Dim zbroj As Integer = prvi + drugi
        Console.WriteLine("Zbroj brojeva {0} i {1} je {2}", prvi,
drugi, zbroj)
End Sub
```

Primijetite da se u zadnjoj liniji `Main()` procedure poziva procedura `Zbroj()` s parametrima koji su pohranjeni u varijable `a` i `b`. Kod pozivanja funkcije, vrijednost prvog parametra, u našem slučaju `a` će biti kopirana u varijablu `prvi`, dok će vrijednost drugog parametra `b` biti kopirana u varijablu `drugi`. Varijabla `prvi` i `drugi` su vidljive i mogu se koristiti samo unutar procedure `Zbroj()`. Na osnovu vrijednosti tih varijabli će biti izračunat njihov zbroj i ispisana odgovarajuća poruka.

**Procedure bez parametara**

Postoje i procedure koje ne primaju parametre, pa se one pozivaju s praznim zagradama iza imena procedure. Takve procedure samo izvršavaju kod unutar tijela funkcije, a ne mogu se koristiti za bilo kakvu obradu podataka, budući da ulazni podaci ne postoje.

# Funkcije

Funkcija je jedna od vrsta procedure koja vraća neku vrijednost u program na mjesto odakle je ta funkcija pozvana. Njihova sintaksa je:

**VB**

```
Function NazivProcedure([listaParametara]) As tipPodatka
        ' tijelo funkcije
End Function
```

Svaka funkcija mora imati definiran tip povratne vrijednosti, što predstavlja očekivani tip podataka koji će funkcija vratiti kada završi njeno izvršavanje. Ilustrirat ćemo primjerom sličnim prethodnom, ali će funkcija `Zbroj()` vraćati vrijednost umjesto ispisa.

**VB**

```
Sub Main(args As String())
        Dim a, b As Integer
        Console.Write("Unesi prvi broj: ")
        a = Console.ReadLine()
        Console.Write("Unesi drugi broj: ")
        b = Console.ReadLine()
        Dim c as Integer = Zbroj(a, b)
        Console.WriteLine("Zbroj brojeva {0} i {1} je {2}", a, b, c)
End Sub

Function Zbroj(ByVal prvi As Integer, ByVal drugi As Integer) As
Integer
        Return prvi + drugi
End Function
```

Promotrite gornji primjer. Kako se ovdje radi o *funkciji* Zbroj(), ona će zbrojiti dva parametra koja joj se nađu na ulazu, te tu vrijednost vratiti. Da bismo sačuvali vrijednost koju funkcija vrati, nju je potrebno zapisati u varijablu, te tu varijablu potom iskoristiti pri ispisu rezultata.

## Opcionalni parametri

Procedure mogu imati i opcionalne parametre, oni moraju biti označeni ključnom riječju `Optional` prije navođenja parametra, te moraju imati označenu vrijednost koja će se koristiti ako se vrijednost parametra *ne* unese. Npr.

**VB**

```
Function Korijen(broj As Integer, Optional stupanj As Integer = 2) As
Integer
        Return Math.Pow(broj, 1 / stupanj)
End Function
```

Dana funkcija računa $n$-ti korijen zadanog broja, a ako stupanj korijena nije zadan, uzima se vrijednost 2, pa se računa kvadratni korijen. Npr. ako navedenu funkciju pozovemo sa sljedećim parametrima:

**VB**

```
Korijen(256, 4)
Korijen(256)
```

u prvom slučaju dobit ćemo $\sqrt[4]{256}$, a u drugom $\sqrt{256}$.

# Prijenos podataka

Podaci se mogu u funkcije i procedure proslijediti na dva načina: po *vrijednosti* i po *referenci*. Kod prosljeđivanja parametara po vrijednosti za taj parametar se stvara privremena kopija tog podatka na novo memorijsko područje. Na ovaj način osiguravamo da se, ako se unutar funkcije/procedure dogodi promjena vrijednosti te varijable, ona utječe samo na tu kopiju, ali ne i na originalnu vrijednost. Ovakav tip parametra se naziva *vrijednosni parametar* i u VB.NET-u se obilježava s ključnom riječi `ByVal` prije navođenja njegovog imena i tipa podataka.

Kod prosljeđivanja po referenci (ključna riječ `ByRef`), ne stvara se nikakva kopija već se prosljeđuje memorijska adresa gdje se nalaze originalni podaci, što za posljedicu ima da eventualna promjena u tim podacima rezultira i promjenom u originalnim podacima.

> **Vrijednosni i referencni**
>
> VB.NET pretpostavlja da je parametar vrijednosni ako nije naveden tip parametra u prototipu funkcije/procedure. Dakle, ako želimo proslijediti parametar po *vrijednosti*, **može se, ali ne mora** navesti ključna riječ `ByVal`, dok ako želimo proslijediti parametar po *referenci*, **obavezno** moramo navesti ključnu riječ `ByRef`.

Primjer:

**VB**

```vb
Sub Main(args As String())
    Dim a As Integer = 3
    Dim b As Integer = 6

    Console.WriteLine($"a = {a}, b = {b}") ' a = 3, b = 6

    TestnaProcedura(a, b)

    Console.WriteLine($"a = {a}, b = {b}") ' a = 3, b = 8
End Sub

Sub TestnaProcedura(ByVal prvi As Integer, ByRef drugi As Integer)
    prvi = 5
    drugi = 8
End Sub
```

Pozivanjem navedene procedure, u varijablu `prvi` se kopira **vrijednost** varijable `a`, dok se u varijablu `drugi` postavlja referenca na memorijsku adresu gdje se nalaze podaci

varijable `b`. Iz tog razloga, nakon izvršavanja naredbi u proceduri, varijabla `b` će imati promijenjenu vrijednost u odnosu na prije poziva procedure.

## Preopterećene procedure

Preopterećene procedure (*overloaded procedures*) se koriste za izvođenje istih ili sličnih operacija nad različitim tipovima ili broju podataka ili imaju različit tip povratne vrijednosti. Sljedeći primjeri su primjeri preopterećenih procedura i funkcija:

**VB**

```vb
' Primjeri preopterećenih Sub procedura
Sub X()
Sub X(ByVal a As Integer)
Sub X(ByVal a As Integer, ByVal b as Long)

' Primjeri preopterećenih funkcija
Function X(ByVal a As Integer, ByVal b as Integer) As Integer
Function X(ByVal a As Long, ByVal b as Long) As Long
Function X(ByVal a As Long, ByVal b as Long) As Integer
Function X(ByVal a As Long, ByVal b as Long) As Short
```

Svaka od ovih procedura ili funkcija ima svoj kod koji se izvršava. Programski prevodilac će odlučiti koju od njih će pozvati tako da usporedi parametarske liste ili tip povratne vrijednosti.

## Rekurzivne funkcije

Rekurzivne funkcije su funkcije koje unutar svog tijela pozivaju same sebe. One obično znaju kako riješiti jednostavne oblike problema kojeg rješavamo, a ako se pojavi složeniji oblik problema ona ga dijeli na bazni/jednostavni dio i složeniji, te složeniji rješava rekurzivnim pozivima samoj sebi čime problem pojednostavnjuje. Najlakše je navedeno opisati primjerom funkcije koja računa faktorijelu nekog prirodnog broja. Faktorijela je definirana s

$$n! = \prod_{i=1}^{n} i = n \cdot (n-1)!$$

uz uvjet $1! = 1$.

**VB**

```
Function Factorial(ByVal n As Integer)
        If (n = 1) Then
                Return 1
        End If

        Return n * Factorial(n - 1)
End Function
```

Kada pozovemo ovu funkciju, ona rekurzivno poziva samu sebe $n-1$ puta. Nakon inicijalnog poziva s parametrom `n`, u svakoj sljedećoj iteraciji se poziva se ista funkcija s parametrom `n-1`, itd. sve dok ne dođemo do poziva funkcije s parametrom 1 za što znamo rješenje koje iznosi 1. Ta funkcija stoga vraća 1, funkcija koja ju je pozvala vraća 2×1, sljedeća vraća 3×2×1, itd. sve dok se ne vratimo kompletno nazad do prve funkcije koju smo pozvali iz `Main()` procedure.

# Zadaci

1. Kreirajte jednostavnu konzolnu aplikaciju koja se sastoji od dva modula: Start i Display. Unutar modula Display napišite tri procedure od kojih jedna ispisuje pozdrav „Dobro jutro“, druga „Dobar dan“ i treća „Dobra večer“ . Unutar modula Start izvršite pozivanje navedenih procedura.
2. Napišite funkciju koja radi okretanje znamenki broja. Prihvaća samo cjelobrojnu vrijednost za unos. Na primjer broj 12345 vraća kao 54321. Napiši drugu funkciju koja vraća broj znamenki zadanog broja.
3. Napišite funkciju koja će za uneseni broj n, provjeriti je li palindrom. Funkcija treba vraćati true ako je broj palindrom inače false.
4. Napišite proceduru koja provjerava i ispisuje je li testirani broj prost. Unutar procedure Main izvedite ispisivanje svih prostih brojeva od 1 do 1000.
5. Napišite proceduru za izračun potencije koja kao parametar prima broj i potenciju, potencija treba biti opcionalna, te ukoliko korisnik ne unese potenciju, ona treba iznositi 2. Na poziv procedure uneseni broj se treba potencirati, te rezultat ispisati u konzoli.
6. Napisati proceduru koja u konzoli ispisuje N slučajno generiranih vrijednosti u rasponu između A i B. Brojeve A, B i N unosi korisnik i u proceduru se prosljeđuju putem parametara. Rezultat se ispisuje u konzoli.
7. Napišite funkciju koja vraća vrijednost n-tog Fibonacci-jevog broja. Fibonacci-jevi brojevi su definirani rekurzivno.
8. Napravite funkciju koja računa najveći zajednički djelitelj. Najveći zajednički djelitelj dva cijela broja je najveći cijeli broj koji ih dijeli bez ostatka. Procedura prihvaća dvije cjelobrojne vrijednosti te vraća njihov zajednički djelitelj.
9. Napišite rekurzivnu funkciju za sumu svih parnih brojeva, u rasponu od 0 do N vrijednosti koje korisnik unosi preko konzole.

10. Napišite funkciju koja će zadani prirodni broj pretvoriti u rimski (rimski brojevi su definirani za brojeve od 1 do 3999).