# OPERATORI



## Operatori

- Fundamentalni za programiranje kao osnovne operacije za matematiku
- Određuju aktivnost nad operandom
- Izvršavanje rezultuje nekom vrednošću
- Vrednost izvršavanja operatora može poslužiti kao operand za drugi operand
- Operatori C# programskog jezika:
  - **aritmetički**
  - **relaciono-logički**
  - **dodeljivanje vrednosti**
  - **ternarni**

# Aritmetički operatori ...

| Unarni operatori | | |
|---|---|---|
| + | operator zadržavanja predznaka operanda | + operand |
| - | operator promene predznaka operanda | - operand |
| ++ | prefiksni operator inkrementiranja | ++ operand |
| ++ | sufiksni operator inkrementiranja | operand ++ |
| -- | prefiksni operator dekrementiranja | -- operand |
| -- | sufiksni operator dekrementiranja | operand -- |



# ... Aritmetički operatori ...

- Primeri unarnih aritmetičkih operatora

| | |
|---|---|
| x++; | x= x + 1 |
| ++x; | x = x + 1 |
| y= x--; | y = x, x = x – 1 |
| y = --x; | x = x – 1, y = x |
| (y + x)--; | nije dozvoljeno |
| ++(y + x); | nije dozvoljeno |
| + -3.14 | -3.14 |

# ... Aritmetički operatori ...

| Binarni operatori | | |
|---|---|---|
| + | operator sabiranja | operand + operand |
| – | operator oduzimanja | operand - operand |
| * | operator množenja | operand * operand |
| / | operator deljenja, tip operanada određuje vrstu deljenja,<br>ako su oba celobrajna, rezultat je celobrajan bez ostatka<br>ako je barem jedan razlomljen, rezultat je razlomljen broj | operand / operand |
| % | operator ostatka celobrojnog deljenja,<br>oba operanda moraju biti celobrojni,<br>predznak ostatka je isti kao predznak prvog operanda | operand % operand |

- Prioritet je isti kao u matematici



# ... Aritmetički operatori

- Primeri binarnih aritmetičkih operatora

| | |
|---|---|
| x + y – z | (x + y) - z |
| x - y * z | x – (y * z) |
| 9 / 2 * 2 | 8 |
| 9.0 / 2 * 2 | 9 |
| -8 % -3 | -2 |

- Zadatak za vežbanje:

  Implementirati program koji za dve celobrojne vrednosti realizuje aritmetičke binarne operacije (+, -, *, /, %) nad njima.
  Rezultat prikazuje korisniku.

# Aritmetički operatori – primer

```
/*****************************************************************************************************
*  Primer jednostavnog programa koji racuna poziciju objeka prilikom slobodnog pada i to prikazuje korisniku
*****************************************************************************************************/
namespace OsnovneOperacije
{
    class Program
    {
        static void Main(string[] args)
        {
            /* Deklaracija promenljivih. */
            double gravitacija = -9.81;
            double inicijalnoUbrzanje = 0.0;
            double vremePadanja = 10.0;
            double inicijalnaPozicija = 0.0;
            double finalnaPozicija = 0.5 * gravitacija * vremePadanja * vremePadanja;
            /* Izracunavanje. */
            finalnaPozicija = finalnaPozicija + inicijalnoUbrzanje * vremePadanja;
            finalnaPozicija = finalnaPozicija + inicijalnaPozicija;
            /* Ispisivanje rezultata. */
            Console.WriteLine("Pozicija objekta nakon ");
            Console.WriteLine(vremePadanja);
            Console.WriteLine(" sekundi je ");
            Console.WriteLine(finalnaPozicija);
            Console.WriteLine("m.");
               Console.ReadLine();
        }
    }
}
```



# Relacioni operatori

- Ispituje relacije između dva operanda
- Rezultat je:
  - **true ako relacija važi**
  - **false ako relacija ne važi**
- Operacije **<**, **<=**, **>**, **>=** su većeg prioriteta od **==**, **!=**

| Binarni operatori | | |
|---|---|---|
| **<** | operator manje | **operand < operand** |
| **<=** | operator manje ili jednako | **operand <= operand** |
| **>** | operator veće | **operand > operand** |
| **>=** | operator veće ili jednako | **operand => operand** |
| **==** | operator jednako | **operand == operand** |
| **!=** | operator nije jednako | **operand != operand** |

# Logički operatori ...

- Formiraju logičke izraze
- Rezultat je:
  - **true ako relacija važi**
  - **false ako relacija ne stoji**
- Rade na nivou čitavog operanda

| Logički operatori | | |
|---|---|---|
| ! | prefiksni unarni operator logičke negacije | ! operand |
| && | binarni operator logičke I operacije; ako je prvi operand **false**, drugi operand se ni ne posmatra | operand && operand |
| \|\| | binarni operator logičke ILI operacije; ako je prvi operand **true**, drugi operand se ni ne posmatra | operand \|\| operand |
| & | binarni operator logičke I operacije; drugi operand se uvek posmatra | operand & operand |
| \| | binarni operator logičke ILI operacije; drugi operand se uvek posmatra | operand \| operand |
| ^ | binarni operator logičke ekskluzivno ili operacije (različito) | operand ^ operand |



# ... Logički operatori

- Primeri logičkih operatora:

```
!false == true
!true == false

false && false == false
false && true == false
true && true == true

false || true == true
false || false == false
true || true == true
```

- Koriste se najčešće kao veznici u relacionim izrazima

```
x && y || z && k;
a == !b && c < d ;

(x && y) || (z && k)
(a == (!b)) && (c < d)
```

| A | B | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

AND

| A | B | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

OR

| A | B | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

NAND

| A | B | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

NOR

| A | B | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

EXOR

| A | Z |
|---|---|
| 0 | 1 |
| 1 | 0 |

NOT

# Logički operatori – primer

```
/*********************************************************************************************************
*  Primer kako rade logicki operaori.
*********************************************************************************************************/
namespace LogickiOperatori
{
    class Program
    {
        static void Main(string[] args)
        {
            /* Deklaracija i inicijalizacija promenljivih. */
            bool a = true, b = false;
            /* Negacija. */
            Console.Write("Negacija od true je: "); Console.WriteLine(!a);
            Console.Write("Negacija od false je: "); Console.WriteLine(!b);
            /* I operacija. */
            Console.Write("true I false je: "); Console.WriteLine(a && b);
            Console.Write("true I true je: "); Console.WriteLine(a && a);
            Console.Write("false I false je: "); Console.WriteLine(b && b);
            /* ILI operacija. */
            Console.Write("true ILI false je: "); Console.WriteLine(a || b);
            Console.Write("true ILI true je: "); Console.WriteLine(a || a);
            Console.Write("false ILI false je: "); Console.WriteLine(b || b);
            /* Eksluzivno ILI operacija. */
            Console.Write("true EXILI false je: "); Console.WriteLine(a ^ b);
            Console.Write("true EXILI true je: "); Console.WriteLine(a ^ a);
            Console.Write("false EXILI false je: "); Console.WriteLine(b ^ b);
            Console.ReadLine();
        }
    }
}
```



# Operatori dodele vrednosti ...

- Jedna od osnovnih aktivnosti računara
- Sama po sebi vraća vrednost

| | |
|---|---|
| **x = 12** | promenljiva **x** dobija vrednost **12** |
| **y = x = 3** | i promenljiva **x** i promenljiva **y** dobijaju vrednost **3** |
| **k = (z = x * 10) * y** | promenljiva **z** dobija vrednost **30**<br>promenljiva **k** dobija vrednost **90** |

- Obratiti pažnju na razliku između == i =, iako jedno proverava jednakost a drugo dodeljuje vrednost, obe vraćaju vrednosti pa se mogu koristiti i u logičkim izrazima

# ... Operatori dodele vrednosti ...

| Binarni operatori | | |
|---|---|---|
| = | satandardno dodeljivanje vrednosti | x = y |
| += | sabiranje i dodeljivanje vrednosti | x += y , x = x + y |
| -= | oduzimanje i dodeljivanje vrednosti | x -= y , x = x - y |
| *= | množenje i dodeljivanje vrednosti | x *= y , x = x * y |
| /= | deljenje i dodeljivanje vrednosti | x /= y , x = x / y |
| %= | ostatak celobrojnog deljenja i dodeljivanje vrednosti | x %= y , x = x % y |

- Šta se može promeniti u prethodnom primeru?



# ... Operatori dodele vrednosti

```
finalnaPozicija = finalnaPozicija + inicijalnoUbrzanje * vremePadanja;
finalnaPozicija = finalnaPozicija + inicijalnaPozicija;
```

ili

```
finalnaPozicija += inicijalnoUbrzanje * vremePadanja;
finalnaPozicija += inicijalnaPozicija;
```

# Prioriteti izvršavanja operatora

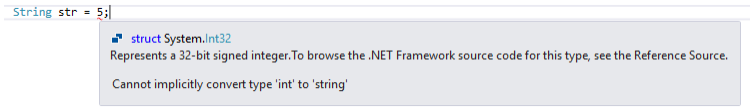
- Čest izvor grešaka, razmišljati o prioritetu operatora
- Može se regulisati pomoću zagrada, ali se unose dodatne operacije

1. **() [ ] -> .**
2. **- + -- ++ ! ~ & sizeof ()**
3. *** / % + -**
4. **<< >>**
5. **< <= > >= == !=**
6. **& ^ | && ||**
7. **? :**
8. **= += -= *= /= %= &= |= ^= >>= <<=**
9. **,**



# Nepodudaranje operatora

- C# prepoznaje grešku nepodudaranja tipa (type mismatch)
- Npr. `String str = 5;`




- Greška! - jer C# ne može da konvertuje implicitno broj u tekst

- Npr. `int a = 51.4;`

- Greška! - jer C# ne može da konvertuje realni broj (implicitno) u celobrojni

# Konverzija tipova

- C# podržava dve vrste konverzija prostih tipova:
  - Implicitnu – kada kompajler prepozna da postoji potreba za konverzijom, pa manji tip konvertuje u veći

```
int a = 2;    // a = 2
float a = 2; // a = 2.0 (implicitna konverzija)
```

  - Eksplicitna – kada programer eksplicitno navede u koji tip želi da se napravi konverzija (voditi računa, ne moraju biti podržane konverzije između svih tipova)

```
int a = 51.4; // Greška!
int a = (int) 51.4;    // a = 51 (eksplicitna)
float b = 2/3; // b = 0.0 (implicitna), ali zašto 0.0?
float b = (float)2/3; // b = 0.6666 (eksplicitna)
```



# Ternarni operator

- Uslovni operator
- Efikasan, iako nije intuitivan kao osnovni operatori
- U zavisnosti od vrednosti uslova izvršava prvi ili drugi izraz:

**(uslov) ? izraz_za_tacan_uslov : izraz_za_netacan_uslov;**

- Npr.

```
maks = (a < b) ? b : a;
y = (z > 0) ? x / z : 0;
```

# Operatori – zadaci

- Napisati program koji izračunava zapreminu kupe sa poluprečnikom r i visinom H (V=1/3*r*r*H*PI)
- Napisati program koji izračunava otpornost bakarnog provodnika dužine l, prečnika d. Specifična otpornost bakra iznosi 1.588e-8 (R=RCu*4*l/(d*d*3.14))
- Napisati program koji određuje pritisak jednog mola (količina gasa - n) idealnog gasa p na osnovu zapremine gasa V i temperature T. Pritisak idealnog gasa se određuje prema sledećoj formuli: p=n*R*T/V, gde je R - univerzalna gasna konstanta i iznosi 8.314472 J/(mol·K)
- Napisati program koji izračunava površinu valjka sa poluprečnikom r i visinom H (A=2*r*(r+H)*PI)



# OSNOVNO O KLASAMA, OBJEKTIMA, METODAMA

# Osnovno o potprogramima ...

- Potprogrami (funkcije, metode) omogućuju **modularnost** i ponovno **korišćenje koda**
- Implementiraju **semantički zatvoren posao** tako da se mogu koristiti u drugom rešenju
- Identifikuju se **nazivom**
- Potrebne podatke za rad dobijaju putem **ulaznih** i/ili **ulazno/izlaznih** parametara
- Rezultate rada prosleđuju **nazivom** i/ili **izlaznim** i/ili **ulazno/izlaznih** parametrima



# ... Osnovno o potprogramima

- U C# potprogrami su deo neke klase – bilo da se pozivaju u duhu objektonog programiranja ili ne
- Metode ugrađene u c# jezik se često koriste, jer mnoge implementiraju stvari koje se ubičajeno koriste u programima
- Mogu se koristiti i ako se ne zna šta rade, kao neka vrsta crne kutije

```
Console.Write("Negacija od true je: ");
Console.WriteLine(!a);
```

- Potprograme WriteLine i Write smo do sada koristili za ispis rezultata – umesto da svaki put programiramo svoju metodu za ispis

# Osnovno o klasama i objektima

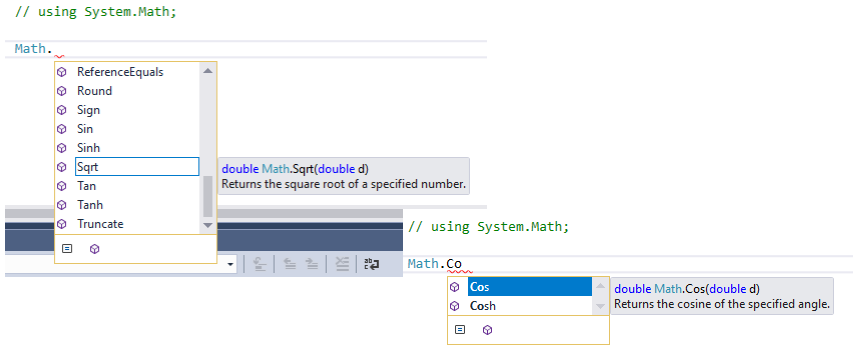
- Klase predstavljaju kontejnere za promenljive i metode
- Reprezentuju entitet (neku pojavu) iz stvarnog sveta
- Koriste se za stvaranje objekata (promenljiva tipa klase)
- Objekat je jedna promenljiva koja sadrži sve atribute (promenljive) i metode jedne klase
- Dostupnim atributima i metodama objekta pristupa se preko operatora . (tačka)

```
Console.WriteLine(!a); // klasa, atribut i metoda
Math.Sqrt(x); // klasa i njena metoda
```

- Neke klase su ugrađene u C#, poput System i Math



- Ako se zna nazive klase, ali ne i metode (i generalno za auto kompletiranje izraza) – vršite ili izbor iz ponuđenog menija ili pišete prvo slovo pa TAB




- Ukoliko želite pregled svih metoda koje vam se nude u ugrađenim (ili kasnije i vašim) klasama, pozicionirajte se mišem iznad poziva neke metode i kliknite F12

# Ugrađene metode i objekti primer

```
/*********************************************************************************************************
* Program koji obavlja određene matematicke proracune pomocu Math klase i ispisuje rezultat. U primeru se koristi
* i konstanta PI. Takođe koristi se ugrađena i metoda DateTime.Now.Millisecond kako bi se preuzelo trenutno vreme
*********************************************************************************************************/
namespace RacunanjeVremena{
    class Program {
        static void Main(string[] args) {
            long vremePocetka; // Vreme pocetka rada programa u milisekundama.
            long vremeKraja;   // Vreme kraja rada programa u milisekundama.
            double vreme;    // Ukupno vreme trajanja programa.
            double sirina = 42.0, visina = 17.0, hipotenuza;  // strane trougla
            vremePocetka = DateTime.Now.Millisecond;
            hipotenuza = Math.Sqrt(sirina * sirina + visina * visina);
            Console.Write("Pravougli trougao cije su stranice 42 i 17 ima hipotenuzu: ");
            Console.WriteLine(hipotenuza);
            Console.Write("sin(1)*sin(1) + cos(1)*cos(1) - 1 je ");
            Console.WriteLine(Math.Sin(1) * Math.Sin(1) + Math.Cos(1) * Math.Cos(1) - 1);
            Random rnd = new Random();
            double nasumicniBroj = rnd.NextDouble();
            Console.Write("Slucajno generisana vrednost:  ");
            Console.WriteLine(nasumicniBroj);
            Console.Write("Broj PI ima vrednost: ");
            Console.WriteLine(Math.PI);
            vremeKraja = DateTime.Now.Millisecond;
            vreme = (vremeKraja - vremePocetka) / 1e3;
            Console.Write("Ukupno trajanje programa u sekundama je:  ");
            Console.WriteLine(vreme);
            Console.Read();
        }
    }
}
```

Prekvalifikacije za IT

# STRING

# String ...

- String je klasa, a svaka promenljiva tipa String je objekat
- Svaka promenljiva tipa String sadrži veliki broj veoma korisnih metoda za rad sa stringovima
- Npr. String sadrži metodu (tačnije Property) ***Length*** koja će vratiti ukupan broj karaktera unutar stringa, ali i već pomenutu String.Format() :

```
String str = "Dragan Jovanovic";
Console.Write("Broj karaktera u ");
Console.Write("stringu  \"Dragan Jovanovic\" je: ");
Console.WriteLine(str.Length);
Console.WriteLine(String.Format("Broj karaktera u stringu
\"{0}\" je: {1}", str, str.Length));
```



# ... String ...

| Neke funkcije za manipulaciju stringovima | |
|---|---|
| **s1.Equals(s2);** | Vraća **true** ako su **s1** i **s2** identični, u suprotnom **false;** |
| **s1.Equals(s2, StringComparison.OrdinalIgnoreCase);** | Vraća **true** ako su **s1** i **s2** identični, u suprotnom **false,** ne razlikujući velika i mala slova**;** |
| **s1.Length** | Vraća dužinu stringa **s1** kao **int** |
| **s1.StartsWith(s2)** | Vraća true ukoliko **s1** započinje sa **s2;** slično je EndsWith |
| **s1.substring(N,M)** | Vraća podstring stringa **s1** počev od karaktera na poziciji N i završno sa karakterom na poziciji M |
| **s1.indexOf(s2)** | Ako je **s2** podstring stringa **s1**, ova metoda vraća indeks (kao **int**) pozicije od koje započinje podstring |
| **s1.ToUpper()** | Vraća string koji je isti kao **s1**, ali čiji su svi karakteri velika slova |
| **s1.ToLower()** | Vraća string koji je isti kao **s1**, ali čiji su svi karakteri velika slova |
| **s1.Trim()** | Vraća string koji je isti kao **s1**, ali čiji su svi karakteri koji se ne mogu odštampati izbačeni (kao npr. \n, \t); slično je TrimEnd, TrimStart |

- Pregled ostalih metoda Stringa moguć je npr. klikom na F12 kada je miš fokusiran na klasi

# ... String

- Spajanje dva stringa (konkatenacija) se realizuje pomoću + operatora
- Npr. ***"Pera" + " Dobrnjac"*** rezultuje sa ***"Pera Dobrnjac"***.
- Operator spajanja stringova podržava spajanje većine osnovnih tipova i stringa
- Npr. umesto da se koristi:
  - `Console.Write("Broj PI ima vrednost: ");`
  - `Console.WriteLine(Math.PI);`
- može da se koristi
  - `Console.WriteLine("Broj PI ima vrednost: " + Math.PI);`



# String – primer

```
/*********************************************************************************************************
* Program koji demonstrira operacije sa stringovima.
*********************************************************************************************************/
namespace StringPrimer{
    class Program{
        static void Main(string[] args)
        {
            String s1 = "Program";
            String s2 = "iranje";
            String s3 = " i programski jezici";
            String s4 = s3;
            Console.WriteLine("String s1: " + s1);
            s1 = s1 + s2;
            Console.WriteLine("String s1 nakon prve konkatacije: " + s1);
            s1 = s1 + s3;
            Console.WriteLine("String s1 nakon druge konkatacije: " + s1);
            Console.WriteLine("String s1 konvertovan u mala slova: " + s1.ToLower());
            Console.WriteLine("String s1 konvertovan u velika slova: " + s1.ToUpper());
            Console.WriteLine("Karakter na 3.oj poziciji u stringu s1: " + s1[2]);
            Console.WriteLine("Rezultat poredjenja stringova s1 i s3: " + s1.Equals(s3));
            Console.WriteLine("Rezultat poredjenja da li su stringovi s4 i s3 jednaki: " + s4.Equals(s3));
            Console.WriteLine("Podstring stringa s1 izmedju 4 i 10 karaktare je: " + s1.Substring(3,9));
            Console.Read();
        }
    }
}
```

# String vs. string

- Koja je razlika između String i string?
- Nema je – string je samo alias za String.

Preporuka je koristiti string kad god se instanciraju objekti:

```
string mesto = "world";
```

- Odnosno String, kada se koriste metode same klase:

```
string pozdrav = String.Format("Hello {0}!", mesto );
```



# Još neki ugrađeni aliasi

```
object:  System.Object
string:  System.String
bool:    System.Boolean
byte:    System.Byte
sbyte:   System.SByte
short:   System.Int16
ushort:  System.UInt16
int:     System.Int32
uint:    System.UInt32
long:    System.Int64
ulong:   System.UInt64
float:   System.Single
double:  System.Double
decimal: System.Decimal
char:    System.Char
```

# ENUM



## enum ...

- Stvaranje novih tipova u C# se esencijalno svodi na pravljenje novih klasa (nije moguće da programski jezik predvidi sve tipove koji će trebati programerima)
- Posebna vrsta klase jeste nabrojivi tip (enumerated types)
- Ona služi da se napravi fiksna lista nekih vrednosti
- Deklariše se na sledeći način:

```
enum naziv_enum_tipa { lista_dozvoljenih_vrednosti }
```

- Na primer:

```
enum Dani { pon, uto, sre, cet, pet, sub, ned }
Dani slobodanDan = Dani.uto;
```

# ... enum

- Svaki element enumeracija zauzima tačno određenu poziciju u enumeraciji (koja je tipa int)
- Tako će ***Dani.pon*** vratiti ***0***, ***Dani.uto*** vratiti ***1*** i tako redom (osim ako nije drugačije navedeno)
- Enumeracija omogućava veću čitljivost koda, jer često postoji potreba da se vrednost koju može da uzme neka promenljiva ograniči na jedan skup mogućih vrednosti
- Omogućuje da kompajler uoči ako se ne upotrebi neka od očekivanih vrednosti (jer su samo enum vrednosti legalne) čime se mogu smenjiti semantičke greške
- enum se deklariše van ***main()*** metode



# enum – primer

```
/*************************************************************************************************
* Program koji koristi enumeraciju za dana u nedelji i mesece u godini. I ispisuje podatke o njima.
*************************************************************************************************/
namespace Enumeracija{
    enum Dan { PONEDELJAK, UTORAK, SREDA, CETVRTAK, PETAK, SUBOTA, NEDELJA }
    enum Mesec { JANUAR, FEBRUAR, MART, APRIL, MAJ, JUN, JUL, AUGUST, SEPTEMBAR, OKTOBAR, NOVEMBAR, DECEMBAR }

    class Program {
        static void Main(string[] args) {
            Dan danRodjenja;     // promenljiva tipa Dan.
            Mesec mesecRodjenja; // promenljiva tipa Mesec.
            danRodjenja = Dan.SREDA;    // dodela vrednosti promenljivoj tipa Dan.
            mesecRodjenja = Mesec.JANUAR;    // dodela vrednosti promenljivoj tipa Mesec.
            Console.WriteLine("Moj znak je vodolija, jer sam ja rodjen u " + mesecRodjenja + "u.");
            Console.WriteLine("To je " + ((int)mesecRodjenja + 1) + ". mesec u godini."); // zasto +1?
            Console.WriteLine("Dan kada sam se rodio je " + danRodjenja + ".");
            Console.WriteLine(danRodjenja + " je " + ((int)danRodjenja + 1) + ". dan nedelje."); //zasto
zagrade?
            Console.Read();
        }
    }
}
```

- Koje biste vi enumeracije koristili?