

# C# Podsetnik

<b>Vaš prvi c# program Hello World program</b>	<pre>class HelloWorld {     Static void Main()     {         System.Console.WriteLine("Hello World");     } }</pre>
<b>Komentari</b>	<pre>// - u jednoj liniji /* */ - u više linija</pre>
<b>Namespaces</b>	<pre>using namespace; /*na početku programa označavaju koje će klase biti korišćene */</pre>
<b>Tipovi podataka</b>	<pre>byte, char, bool, sbyte, short, ushort, int, uint, float, double, decimal, long, ulong.</pre>
<b>Varijable</b>	Na primer: <pre>int x = 1; // Varijable moraju biti inicijalizovane pre upotrebe.</pre>
<b>Konstante</b>	Na primer: <pre>const int x =1;</pre>
<b>Enumeracioni tip</b>	Na primer: <pre>enum Days {Sat, Sun, Mon, Tue, Wed, Thu, Fri};</pre> <p>Ovde je Sat is 0, Sun is 1, Mon is 2, itd. Koriste se kao Days.Sat na primer.</p>
<b>Stringovi</b>	Na primer: <pre>string s = "ABC";</pre>
<b>if naredba ( selekcija)</b>	<pre>if (expr) { } else { }</pre>
<b>Switch naredba (višestruka selekcija)</b>	<pre>switch switch (expr) {     case expr: statement; break or goto;     default: statement }</pre>
<b>Repeticija (iteracija, petlja, ciklus)</b>	<pre>while (expr is true) { } Ili do { } while (expr is true) Ili for (int i=start;i&lt;end;i++) { } Ili foreach (obj x in coll) { x.blah(); }</pre>

<b>break/continue</b> naredbe	continue u repeticiji označava nastavak ciklusa break prekida izvršavanje repeticije (ciklusa)
<b>Operatori</b>	Assignment (=) Arithmetic (+, -, *, /, %(modulus)) Increment (++), +=, +* etc), decrement (--) Relational (==, !=, >, >=, <, <=) Conditional(&&,   , !) Logical (&, ^,  ) Ternary (cond-expr ? expr1 : expr2)
<b>Klase</b>	Definisanje klase: <pre>public class A:base-class { }</pre> Kreiranje objekta iz date klase: <pre>A a = new A();</pre>
<b>Prava pristupa</b> ( <b>Access modifiers</b> )	public : bez restrikcija private : može se pristupati samo iz klase protected : može se pristupati samo iz klase ili subklase internal : pristupačno svim klasama iz skupa (assembly) protected internal : == protected ili internal
<b>Definisanje metoda</b>	<pre>return-type Name(params) { }</pre>
<b>Konstruktori</b>	To su metode kojima se kreiraju objekti date klase. Imaju isto ime kao i klasa, ali nemaju return-type. Može postojati više konstruktora sa različitim parametrima.  Primer: <pre>ClassName () =;</pre>
<b>Destruktori</b>	Koriste se samo u slučaju objekata kojima se ne upravlja automatski. Imaju isto ime kao i klase, sa znakom ~ (tilda) ispred imena.  Primer: <pre>~ClassName()</pre>
<b>Reference unutar klase</b>	<b>this</b> je referenca na tekući objekat <b>base</b> je referenca na objekat iz superklase
<b>Static members</b> ( <b>statički članovi</b> )	Pripadaju i mogu se koristiti samo unutar klase Ne mogu se koristiti kao objekti.

<b>Parametri (u metodama)</b>	<p>Pretostavlja se prenos po vrednosti. Ako se želi prenos po referenci koristi se reč ref ispred parametra. Na primer:</p> <pre>MethodA(ref int i); /* varijabla i se prenosi kao adresa varijable a ne kao vrednsot. */</pre> <p>Postoji i out referenca za rezultate pozivanja metoda. Na primer:</p> <pre>MethodB(out double[] iznos); /* oznacava da ce niz iznos biti rezultat rada metoda */</pre> <p>Moguće je prenti i promenljiv niz parametara kao u pimeru:</p> <pre>Void MyMethod(param int[] intVals) // Definicija metoda</pre> <pre>MyMethod(1,2,3); ili MyMethod(myIntArray); // Pozivanje metoda</pre>
<b>Properties</b>	<p>Omogućavaju da varijable ostanu privatne, i da im se pristupa samo preko properties. Na primer:</p> <pre>public int Xyz { get { return Xyz; }   set { Xyz = value } }</pre> <p>omogućava da se varijabli Xyz pristupa kao da je obična varijabla, npr. a.Xyz++</p>
<b>Upravljanje izuzecima (Exception Handling)</b>	<p>Generisanje izuzetka:</p> <pre>throw new System.Exception("...");</pre> <p>Upravljanje izuzecima:</p> <pre>try { } catch { } // or catch (Exception type) { } finally { }</pre>