

Dezvoltarea Aplicatiilor Web utilizand ASP.NET Core MVC

Curs 2

Cuprins:

- [Introducere in C#](#)
- [Structura unui program](#)
- [Variabile si Tipuri de date](#)
- [Conversii de tip](#)
- [Nullable](#)
- [Instructiuni de control](#)
- [Array-uri](#)
- [Conventii de nume - Naming conventions](#)

Introducere in C#

C# este unul dintre cele mai populare limbaje de programare, dezvoltat de Microsoft, care ruleaza pe .NET Framework.

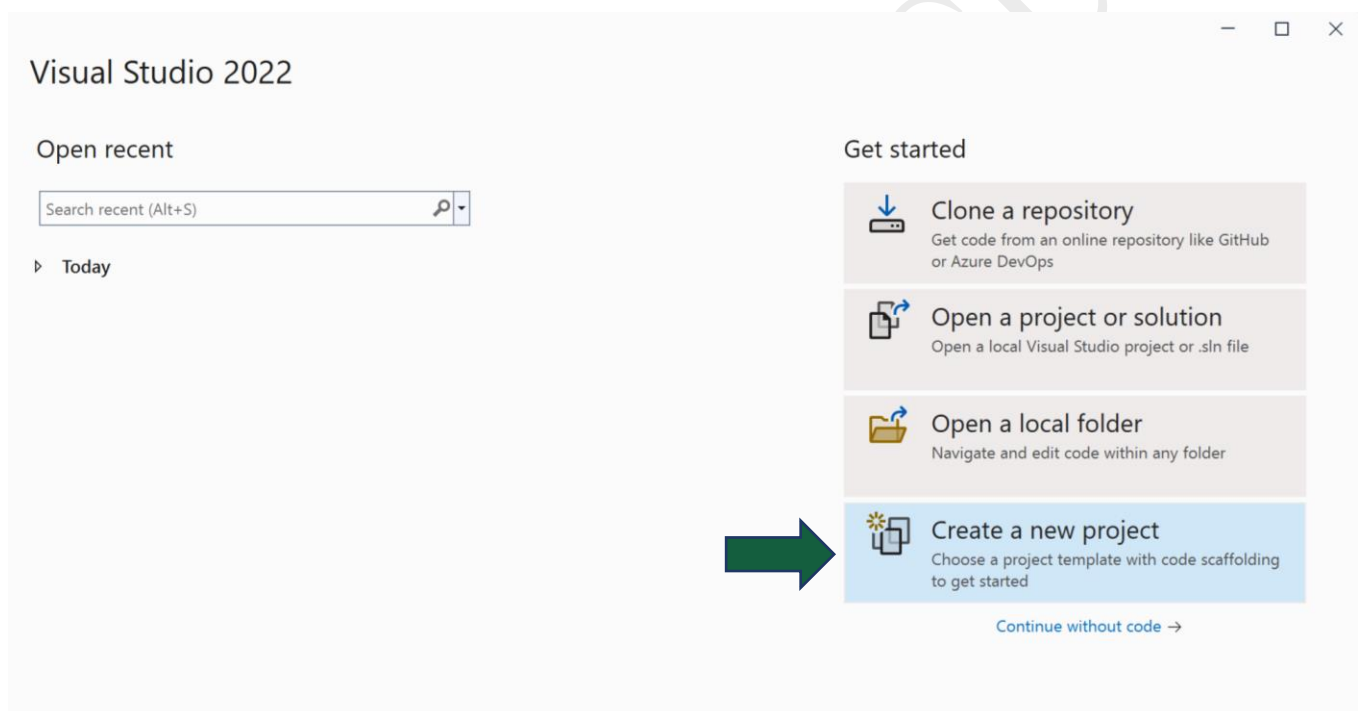
Este folosit pentru a dezvolta o gama variata de aplicatii, de la aplicatii web, mobile, desktop, pana la jocuri, servicii web, VR, etc.

Structura unui program

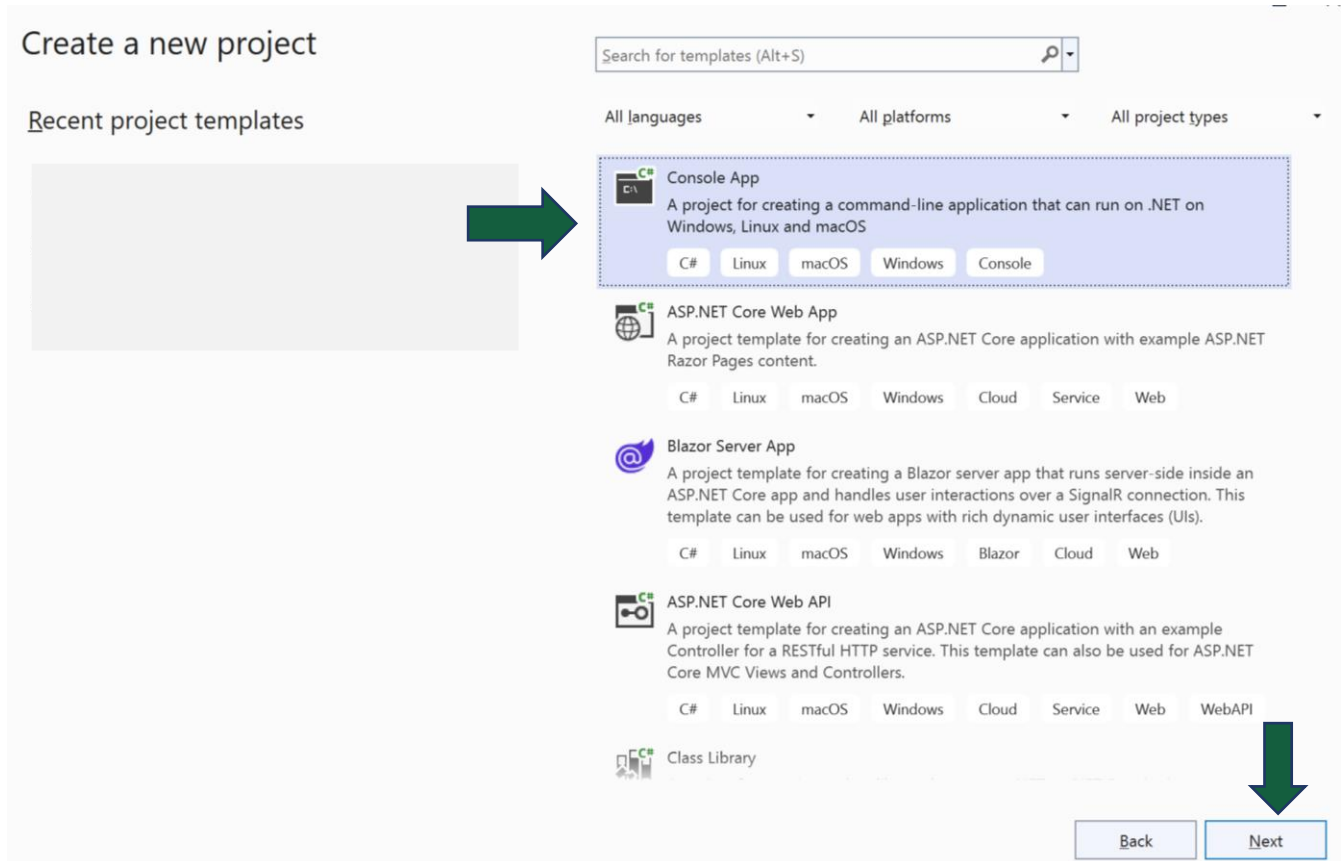
Pentru a dezvolta un program scris in C#, vom utiliza Visual Studio 2022 (VEZI Laborator 1 – instalare VS 2022).

Pentru inceput se creeaza un proiect folosind urmatoarele proprietati:

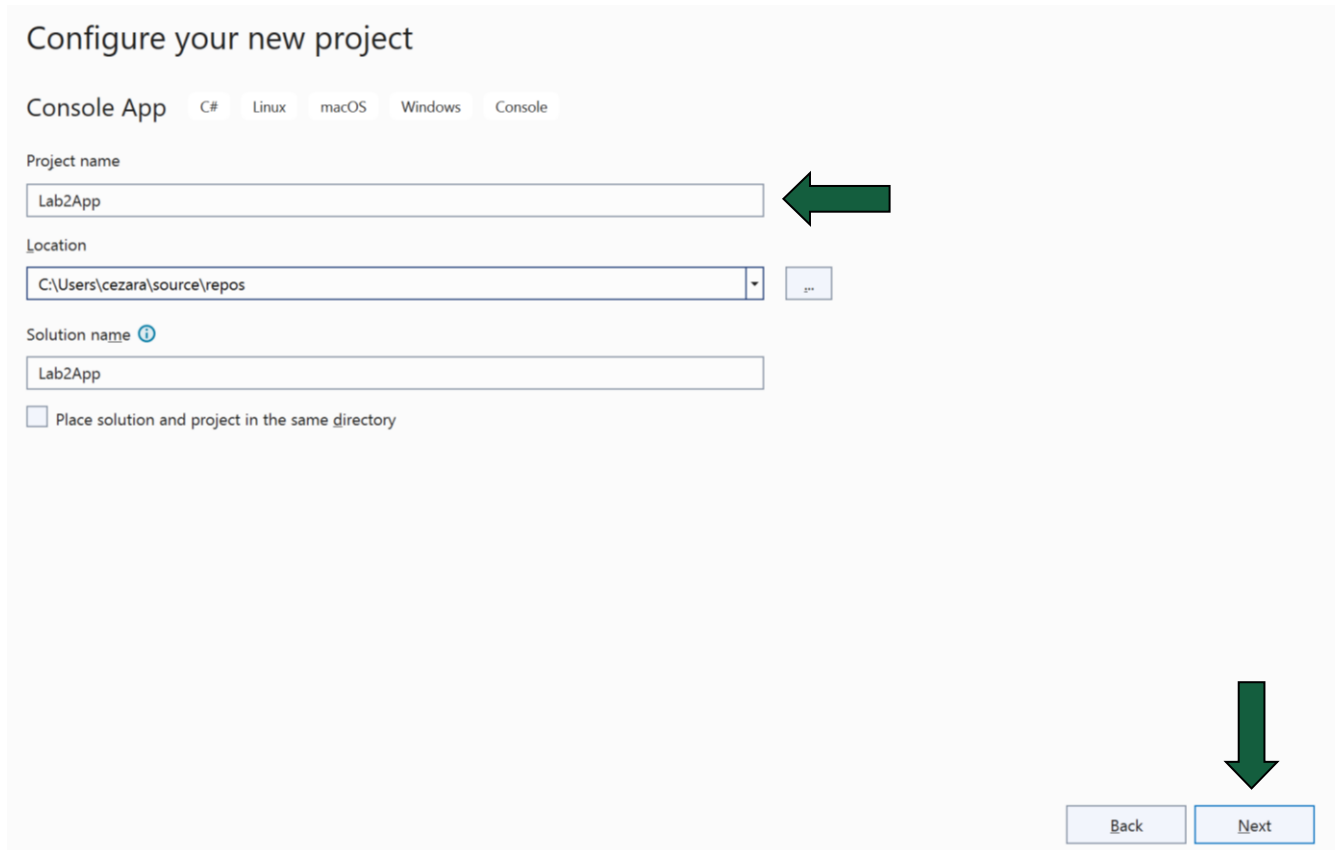
PASUL 1:



PASUL 2:



PASUL 3:



Configure your new project

Console App C# Linux macOS Windows Console

Project name
Lab2App

Location
C:\Users\cezara\source\repos

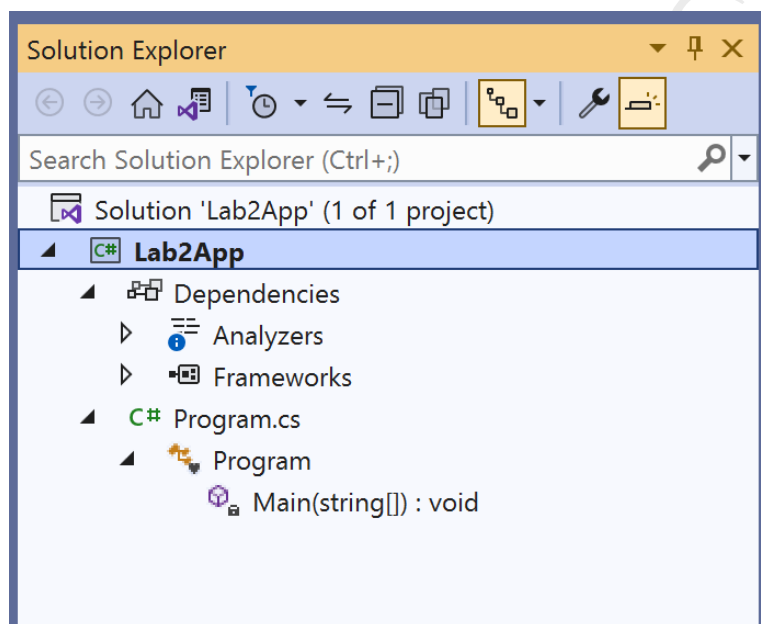
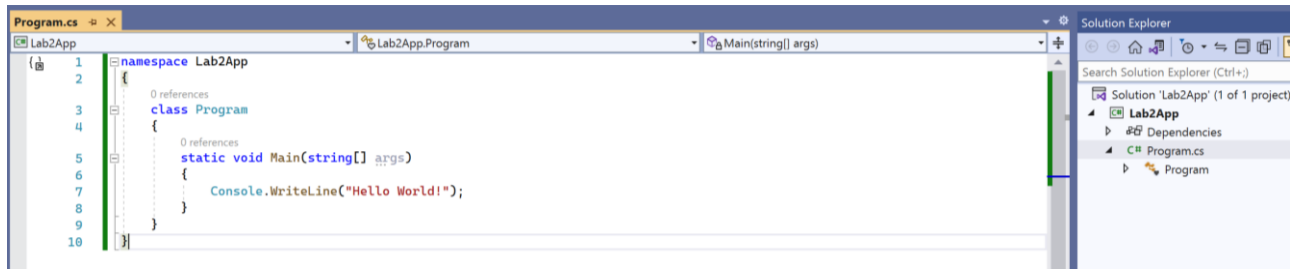
Solution name ⓘ
Lab2App

☐ Place solution and project in the same directory

Back Next

PASUL 4:

In fisierul **Program.cs** vom scrie cel mai simplu program in C#. Se va afisa in consola mesajul "Hello World!"

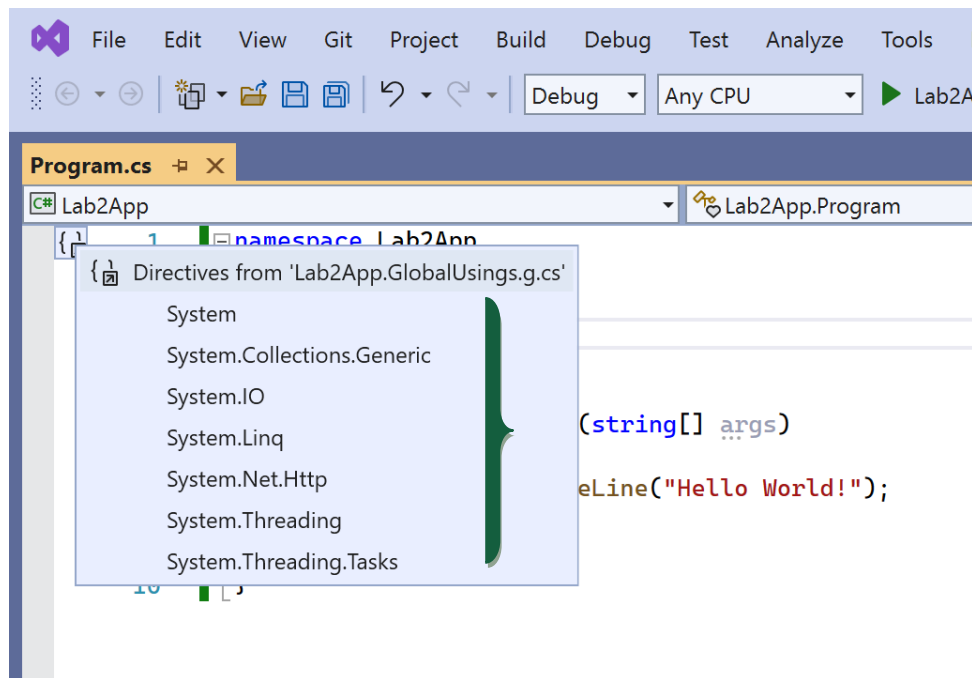


In momentul crearii proiectului, acesta a fost numit **Lab2App**. Denumirea proiectului devine automat **namespace**.

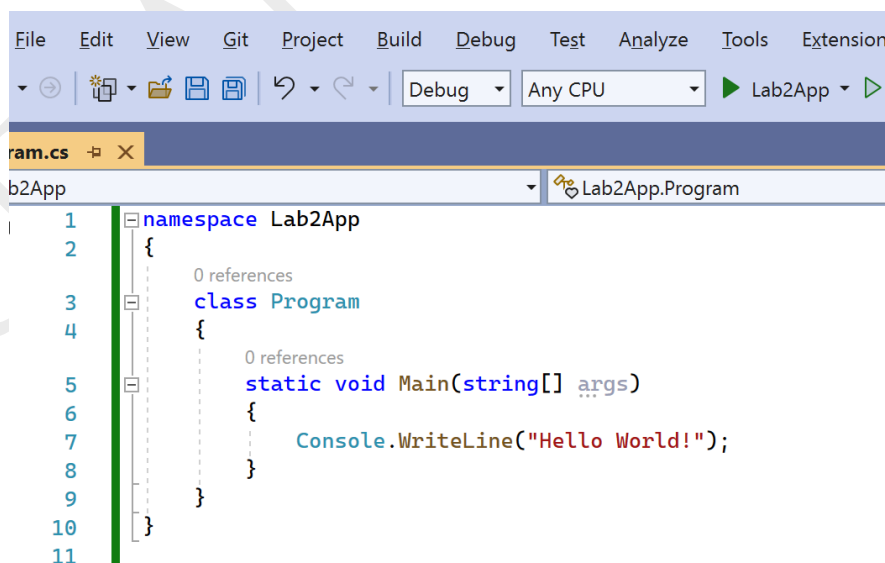
Un **namespace** – este o **colectie de clase**. Namespace-ul Lab2App contine clasa **Program**. Atunci cand scriem cod intr-un namespace, avem acces la toate clasele definite in el.

Daca se doreste utilizarea unei clase dintr-un alt namespace, atunci acel namespace trebuie importat.

Pentru import se utilizeaza **using**. Visual Studio are incluse cateva astfel de directive.



Directiva **System** – ofera acces la toate clasele de baza. In cazul de fata, System ofera acces la clasa Console care la randul ei contine metoda **WriteLine**.



Clasa **Program** – **defineste metoda Main**. Comparativ cu Java, unde clasa care contine metoda Main trebuie sa se numeasca tot Main, in cazul lui C# clasa poate avea orice denumire.

static void Main (string[] args) – este metoda principala a programului, metoda care se apeleaza prima.

!/! OBS:

- C# este case sensitive (**VEZI** mai jos – [naming conventions](#))
- Orice declaratie sau expresie se incheie cu ;
- Executia programelor incepe intotdeauna cu metoda Main
- Comentariul pe o singura linie se include folosind //
- Comentariile pe mai multe linii se includ folosind /**/

Variabile si Tipuri de date

O **variabila** – este un nume pe care il ia o zona de memorie. Ulterior acel spatiu de memorie poate fi manipulat prin intermediul denumirii variabilei.

Fiecare variabila are un anumit **tip de date**, care determina dimensiunea zonei de memorie.

Tipuri de date in C#

Cele mai utilizate tipuri de date sunt:

- int – numere intregi fara virgula
- double – numere intregi cu virgula
- char – stocheaza cate un caracter
- string – stocheaza text
- bool – stocheaza valori true sau false
- object – tipul obiect este clasa de baza pentru toate tipurile de date in C#. Tipurilor obiecte li se pot atribui valori de orice tip.

```
// TIPURILE DE DATE
```

```
int nr = 100;
string str = "Acesta este un text";
double d = 12.35;
char c = 'a';
bool b = true;
object obj = 100;

Console.WriteLine("Numarul este " + nr);
Console.WriteLine("Stringul este: " + str);
Console.WriteLine("Numarul in virugla mobila este: " + d);
Console.WriteLine("Caracterul este: " + c);
Console.WriteLine("Valoarea de adevar este: " + b);
Console.WriteLine("Obiectul este: " + obj);
```

Conversii de tip

Conversiile de tip se impart in doua categorii:

- **implicite** – compilatorul C# convertește automat un tip de date în alt tip de date. În general tipuri de date care ocupa o zona mai mica de memorie, cum este tipul int, sunt convertite automat în tipuri de date care ocupa o zona mai mare de memorie.

```
// CONVERSII IMPLICITE
```

```
int nrInt = 10;

// Metoda GetType() preia tipul de date
Type tipNrInt = nrInt.GetType();

// Conversie implicita
double nrDouble = nrInt;

// Se preia tipul
Type tipNrDouble = nrDouble.GetType();

// Afisare valori inainte de conversie
Console.WriteLine("nrInt value: " + nrInt);
Console.WriteLine("nrInt Type: " + tipNrInt);

// Afisare valori dupa conversia implicita
Console.WriteLine("nrDouble value: " + nrDouble);
Console.WriteLine("nrDouble Type: " + tipNrDouble);
```


Compilatorul a convertit implicit tipul int in double, fara pierdere de informatie.

- **explicite** – in cazul in care se doreste conversia unui tip care ocupa o zona mai mare de memorie, intr-un tip care are o dimensiune mai mica a memoriei

```
// CONVERSII EXPLICITE

double nDouble = 25.123;

// Conversie explicita
int nInt = (int)nDouble;

// Afisarea valorii inainte de conversie
Console.WriteLine("Valoarea inainte de conversie a fost: "
+ nDouble);

// Afisarea valorii dupa conversie
Console.WriteLine("Valoarea dupa conversie este: " +
nInt);
```

- Se poate utiliza si conversia folosind **Parse()**. Aceasta se utilizeaza cu precadere in cazurile in care se convertesc tipuri de date care nu sunt compatibile. De exemplu, int si string

Sintaxa: int.Parse(parametru);

```
// CONVERSIE UTILIZAND PARSE()

string st = "100";

// tipul de date
Type tip1 = st.GetType();

// Se convertește tipul string în int
int x = int.Parse(st);
Type tip2 = x.GetType();

Console.WriteLine("Valoarea initiala a fost: " + st);
Console.WriteLine("A avut tipul: " + tip1);

Console.WriteLine("Noua valoare dupa conversie este: " + x);
Console.WriteLine("Valoarea dupa conversie are tipul: " + tip2);
```

- **Conversii folosind clasa Convert** – clasa pune la dispoziție numeroase metode pentru a converti orice tip de date într-un alt tip de date -> **ToBoolean(), ToChar(), ToDouble(), ToInt16(), ToString()**

```
// CONVERSII FOLOSIND CLASA CONVERT

int num = 25;
Console.WriteLine("Valoare de tip int: " + num);

// Se convertește valoarea int în stringul "25"
string strConvert = Convert.ToString(num);
Console.WriteLine("Valoarea dupa conversie " +
strConvert);
Console.WriteLine("Tipul dupa conversie: " +
strConvert.GetType());

// Conversie în Double
Double doubleConvert = Convert.ToDouble(num);
Console.WriteLine("Valoarea dupa conversie " +
doubleConvert);
Console.WriteLine("Tipul dupa conversie: " +
doubleConvert.GetType());
```

Nullable

Nullable reprezintă în C# mai multe tipuri de date. Aceste tipuri de date pot lua valori dintr-un interval de valori sau pot fi null.

Sintaxa pentru declararea unui tip de date nullable este următoarea:

<tipul_de_date> ? <nume_variabila> = null;

Exemplu:

```
// NULLABLE
int? num1 = null;
int? num2 = 45;

Console.WriteLine("Valorile sunt: {0}, {1}", num1, num2);
```

Instructiuni de control

```
// INSTRUCȚIUNI DE CONTROL

// if
if (Conditie)
{
    // se executa daca conditia este true
}

// if else
if (Conditie)
{
    // se executa daca conditia este true
}
else
{
    // se executa daca conditia este false
}
```

```
//if else --- else
if (Conditie1)
{
    // se executa daca conditia1 este true
}
else if (Conditie2)
{
    // se executa daca conditia2 este true
}
else if (Conditie3)
{
    // se executa daca conditia3 este true
}
else
{
    // se executa daca nicio conditie din cele anterioare
    nu se indeplineste
}
```

```
// if imbricat (nested)
if (Conditie1)
{
    // se executa daca conditia1 este true
    if (Conditie2)
    {
        // se executa daca conditia2 este true
    }
}
```

```
// while loop
while (Conditie)
{
    statement(s);
}
```

```
// for loop
for (init; conditie; increment)
{
    statement(s);
}
```

```
// do...while loop
do
{
    statement(s);
} while(Conditie);
```

Array-uri

Un **array** este utilizat pentru a stoca o colectie de date de acelasi tip. Fiecare element din array este accesat prin indexul sau.

Sintaxa de declarare a unui array:

datatype[] numeArray;

```
// ARRAY
```

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

for (int i = 0; i < 10; i++)
{
    n[i] = i + 1;
}

for (int i = 0; i < 10; i++)
{
    Console.WriteLine("Element[{0}] = {1}", i, n[i]);
}
```

Conventii de nume - Naming conventions

- Se utilizeaza **PascalCase** pentru:
 - Clase
 - Constructor
 - Metode

- Se utilizeaza **camelCase** pentru:
 - Variabile
 - Argumentele metodelor