Module développement C#

Antoine Burgaudeau Sup de Vinci 2025

Présentation du C#

Qu’est-ce que C# ?

- Langage de programmation orienté objet créé par Microsoft (2000)
- Fait partie de la plateforme .NET
- Inspiré de C, C++ et Java : syntaxe familière et puissante

Objectifs du langage

- Simplicité, sécurité et productivité
- Favoriser la modularité et la réutilisation du code
- Compatible avec le développement Windows, Web, mobile et cloud

Présentation du C#

Domaines d’utilisation

- Applications Windows (WPF, WinForms)
- Développement Web (ASP.NET Core, Blazor)
- Jeux vidéo avec Unity
- Applications cloud (Azure) et API REST
- Applications multi-plateformes (MAUI, Xamarin)

Outils principaux

Syntaxe et bases du C#

Main()

```
using System;

namespace ConsoleApplication
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("I am being Called from Program.cs");
            Console.ReadLine();
        }
    }
}
```

Type

Type	Description	Exemple
int	Nombre entier (32 bits)	int age = 27;
long	Grand entier (64 bits)	long distance = 15000000L;
float	Nombre décimal	float prix = 19.99f;
double	Nombre décimal	double pi = 3.14159;
decimal	Nombre décimal (précision)	decimal salaire = 2500.75m;
bool	Boolean (vrai/faux)	bool estConnecte = true;
char	Caractère unique	char lettre = ‘A’;
string	Chaine de charactère	string langue = “Français”;
var	Type déduit	var langue = “Français”;
object	Type racine de tous les objets	object x = 42;

Commandes & Méthodes

Commande / Méthode	Description	Exemple
Console.WriteLine()	Affiche un message avec saut de ligne	Console.WriteLine("Hello World!");
Console.Write()	Affiche sans saut de ligne	Console.Write("Entrez votre nom : ");
Console.ReadLine()	Lit une entrée utilisateur (texte)	string nom = Console.ReadLine();
Console.ReadKey()	Attend l'appui d'une touche	Console.ReadKey();
Convertit une chaîne en entier	Convertit une chaîne en entier	int age = Convert.ToInt32(Console.ReadLine());
int.Parse()	Convertit une chaîne en entier (méthode stricte)	int nb = int.Parse("42");
ToString()	Convertit une valeur en texte	int x = 5; Console.WriteLine(x.ToString());
string.Concat()	Concatène plusieurs chaînes	string message = string.Concat("Hello ", "World");
string.Format()	Formate une chaîne avec des variables	Console.WriteLine(string.Format("Bonjour {0}", nom));
$"" (interpolation)	Insère directement des variables dans une chaîne	Console.WriteLine($"Bonjour {nom}, vous avez {age} ans.");

Premier Exercice

Faire un programme en C# qui permet de demander votre nom, prénom date de naissance ainsi que votre adresse découpée en rue, code postal, et ville.

Le programme doit finir en écrivant :

Bonjour Antoine Burgaudeau,

tu as 27 ans et tu habites au 9 rue François Albert 44200 Nantes.

Tableaux & Listes

Type	Description	Exemple
Array	Tableau fixe, rapide à parcourir	int[] notes = { 12, 15, 18 };
Liste<T>	Taille variable, typée	List<String> noms = new List<string>();
Dictionary<K,V>	Paire clef/valeur, accès rapide	Dictionary<string,int> ages = new Dictionary<string,int>();
HashSet<T>	Éléments uniques, non ordonné	HashSet<int> ids = new HashSet<int>();
Queue<T>	Premier entré, premier sorti	Queue<string> file = new Queue<string>();
Stack<T>	Dernier entré, premier sorti	Stack<int> pile = new Stack<int>();

Sources de données

- Yaml
- CSV
- JSON
- XML
- SQL
  - MariaDB
  - PostgresDB
- No-SQL
  - Redis
  - MongoDb
  - SQLlite
- Txt

Deuxième Exercice

Faire un programme en C# qui permet de récupérer les information du CSV et hydrater les objets Person et Detail.

Le programme doit finir en écrivant :

Bonjour Antoine Burgaudeau,

tu as 27 ans et tu habites au 9 rue François Albert 44200 Nantes.

Méthodes Linq

Méthode	Description	Exemple
Where()	Filtrer les éléments	List<Person> majeurs = ages.Where(a => a.Value >= 18);
Select()	Transformer les données	List<String> noms = users.Select(u => u.Name);
OrderBy()	Trier Croissant	List<Person> tri = notes.OrderBy(n => n);
OrderByDescending()	Trier Décroissant	List<Person> tri = notes.OrderByDescending(n => n);
First()/FirstOrDefault()	Premier élément	Person premier = noms.First();
Count()	Compte les éléments	int nb = persons.Count();
Sum()/Average()	Calculs rapides	float moyenne = person.taille.Average();
Distinct()	Supprimer les doublons	List<Person> uniques = persons.Distinct();

Troisième Exercice

Faire un programme en C# qui permet de nous dire combien de personne sont plus grandes que la moyenne de la classe.

Le programme doit finir en écrivant :

Il y a 15 personnes qui sont plus grandes que la moyenne de la classe qui est de 1,75 mètre.

Troisième Exercice point cinq

Faire un programme en C# qui permet d'ajouter la liste des Personne à un Objet Classe qui comporte une liste d'élève, un Nom, Une école, un Niveau.

Quatrième Exercice

Faire un programme en C# qui permet de nous écrire le nom des personnes les plus grandes de la moyenne de la Classe et de nous écrire leur Prénom par ordre de taille. Il faut aussi que ces personnes soient uniquement de Nantes.

Le programme doit finir en écrivant :

1 - Romain - 1,90

2-Clara - 1.89…

Notation

- Qualité/Optimisation du Code
- Commentaires/région
- Résultat
- Mise en place des éléments vu en cours

Introduction à Entity Framework

- ORM = *Object Relational Mapper*
- Permet d’interagir avec une base de données **sans écrire de SQL brut**
- Traduit automatiquement les opérations C# en requêtes SQL
- Fait partie intégrante de l’écosystème **.NET / ASP.NET Core**

Les approches possibles

Approche	Description
Code First	Les classes C# génèrent la base de données (via migrations)
Database First	La base existe déjà, EF crée les classes
Model First	Modèle visuel → génération code + base

Principe de fonctionnement

- Les **classes C#** représentent les **tables**
- Les **propriétés** représentent les **colonnes**
- Le **contexte** (DbContext) gère la connexion, les entités et les transactions

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

public class Classe
{
    [Key]
    public Guid Id { get; set; } = Guid.NewGuid();

    [Required]
    public string Name { get; set; }

    public string Level { get; set; }

    // Relation 1..n : une classe contient plusieurs personnes
    public ICollection<Person> Persons { get; set; } = new List<Person>();
}
```

```cs
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

public class Person
{
    [Key]
    public Guid Id { get; set; } = Guid.NewGuid();

    [Required]
    public string Firstname { get; set; }

    public string Lastname { get; set; }

    public DateTime Birthdate { get; set; }

    public double Size { get; set; }

    // Clé étrangère vers Classe
    [ForeignKey("Classe")]
    public Guid ClasseId { get; set; }

    // Navigation property
    public Classe Classe { get; set; }
}
```

Class Person avec EF

Mise en place

On ajoute le package: Microsoft.EntityFrameworkCore.Design

Puis on lance la migration

```
dotnet ef migrations add InitialCreate
dotnet ef database update
dotnet ef dbcontext list
```

Exemple d’utilisation

```
using var context = new SchoolContext();
var persons = context.Persons
                     .Where(p => p.Size > 1.80)
                     .ToList();
```

```
SELECT * FROM person WHERE size > 1.80;
```