Présentation C#

C# : Langage orienté objet, mais de plus en plus langage de programmation orientée composant.

* Garbage collector (permet d’optimiser la mémoire utilisée par le programme)
* Gère les exceptions
* Type sécurisé (impossible de lire des variables non initialisées, d’indexer des tableaux au-delà de leurs limites et d’effectuer des transtypages non contrôlés) et unifié (tous les types de bases héritent d’un même type object
* Gestion des versions (modificateurs virtual et override distincts)

Structure du code C# :

* Programmes (se composent d’un ou plusieurs fichiers source)
* Namespaces (Contient un ensemble d’objets ou types)
* Types (les classes ou les interfaces en sont des exemples et contiennent des membres)
* Membres (fonctions ou propriétés contenues dans un type)
* Assemblys (le contenant de l’application une fois compilée, peut être en .exe ou en .dll selon qu’elle implémente une application ou une bibliothèque)

Type-Safe. A type error occurs when the data type of one object is mistakenly assigned to another object, creating unintended side effects. Both C# and Java make a good effort to ensure that illegal casts will be caught at compile time and exceptions will be thrown out at runtime if a cast cannot be cast to a new type.

Garbage Collection. In lower-level languages, memory management can be tedious because you have to remember to properly delete new objects to free up resources. That’s not the case in C# and Java, where built-in garbage collection helps prevent memory leaks by removing objects that are no longer being used by the application. While memory leaks can still occur, the basics of memory management have already been taken care of for you.

Single Inheritance. Both C# and Java support single inheritance—meaning only one path exists from any base class to any of its derived classes. This limits unintended side effects that can occur when multiple paths exist between multiple base classes and derived classes. The diamond pattern is a textbook example of this problem.

Interfaces. An interface is an abstract class where all methods are abstract. An abstract method is one that is declared but does not contain the details of its implementation. The code governing any methods or properties defined by the interface must be supplied by the class that implements it. This helps avoid the ambiguity of the diamond pattern, because it’s always clear which base class is implementing a given derived class during runtime. The result is the clean, linear class hierarchy of single inheritance combined with some of the versatility of multiple inheritance. In fact, using abstract classes is one-way multiple inheritance languages can overcome the diamond pattern.



Windows vs. Open-Source. While open-source implementations exist, C# is mostly used to develop for Microsoft platforms—the .NET Framework’s CLR being the most widely used implementation of the CLI. On the other end of the spectrum, Java has a huge open-source ecosystem and gained a second wind in spite of its age, thanks in part to Google adopting the JVM for Android.

Support for Generics. Generics improve compiler-assisted checking of types largely by removing casts from source code. In Java, generics are implemented using erasures. Generic type parameters are “erased”, and casts are added upon compilation into bytecode. C# takes generics even further by integrating it into the CLI and allowing type information to be available at runtime, yielding a slight performance gain.

Support for Delegates (Pointers). C# has delegates which essentially serve as methods that can be called without knowledge of the target object. To achieve the same functionality in Java, you need to use an interface with a single method or some other workaround that may require a nontrivial amount of additional code, depending on the application.

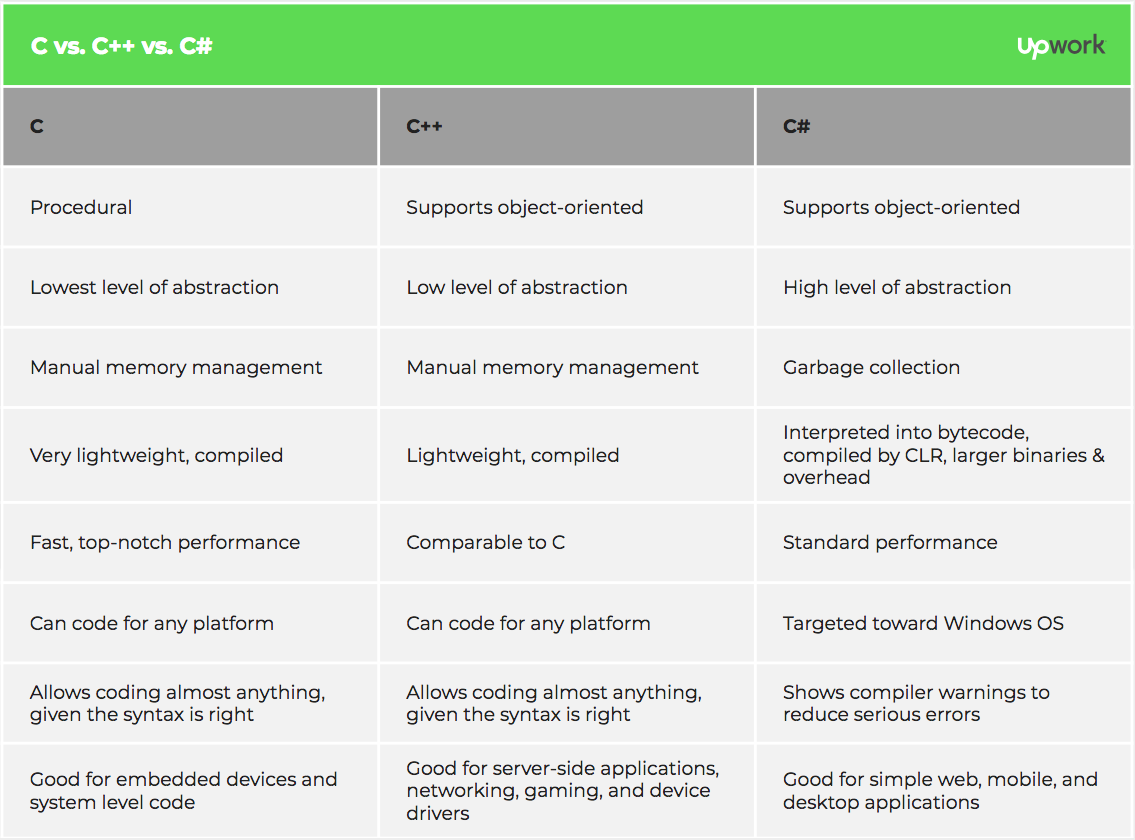
Checked Exceptions. Java distinguishes between two types of exceptions—checked and unchecked. C# chose a more minimalist approach by only having one type of exception. While the ability to catch exceptions can be useful, it can also have an adverse effect on scalability and version control.

Polymorphism: C# and Java take very different approaches to polymorphism. While Java enables polymorphism by default, C# must invoke the “virtual” keyword in a base class and the “override” keyword in a derived class.

Enumerations (Enums): In C#, enums are simple lists of named constants where the underlying type must be integral. Java takes the enum further by treating it as a named instance of a type, making it easier to add custom behavior to individual enums.

Object-oriented: Although the syntax is slightly different, the concept of classes, inheritance and polymorphism.

Compiled languages: Unlike Java which is an interpreted language, both C# and C++ are compiled languages. This means that before an application is launched on a PC or the server, the code must be converted to binaries. An executable EXE file is an example of a compiled file that could be written in C++ or C#.



Size of binaries: We mentioned that the two languages are compiled languages that turn your code into binary files. C# has a lot of overhead and libraries included before it will compile. C++ is much more lightweight. Therefore, C# binaries are much larger after it compiles compared to C++.

Performance: C++ is widely used when higher level languages are not efficient. C++ code is much faster than C# code, which makes it a better solution for applications where performance is important. For instance, your network analysis software might need some C++ code, but performance is probably not a huge issue for a standard word processing application coded in C#.

Garbage collection: With C#, you don’t have to worry much about garbage collection. With C++, you have no automatic garbage collection and must allocate and deallocate memory for your objects.

Platform target: C# programs are usually targeted towards the Windows operating system, although Microsoft is working towards cross-platform support for C# programs. With C++, you can code for any platform including Mac, Windows and Linux.

Types of projects: C++ programmers generally focus on applications that work directly with hardware or that need better performance than other languages can offer. C++ programs include server-side applications, networking, gaming, and even device drivers for your PC. C# is generally used for web, mobile and desktop applications.

Compiler warnings: C++ will let you do almost anything provided the syntax is right. It’s a flexible language, but you can cause some real damage to the operating system. C# is much more protected and gives you compiler errors and warnings without allowing you to make some serious errors that C++ will allow.