comparison

programming languages c and rust.

The comparison between Rust and C is a trending topic for developers. The languages are a lot alike, and developers often wonder which of the two is better. Let’s start with a definition of these 2 languages before comparing them.

C.

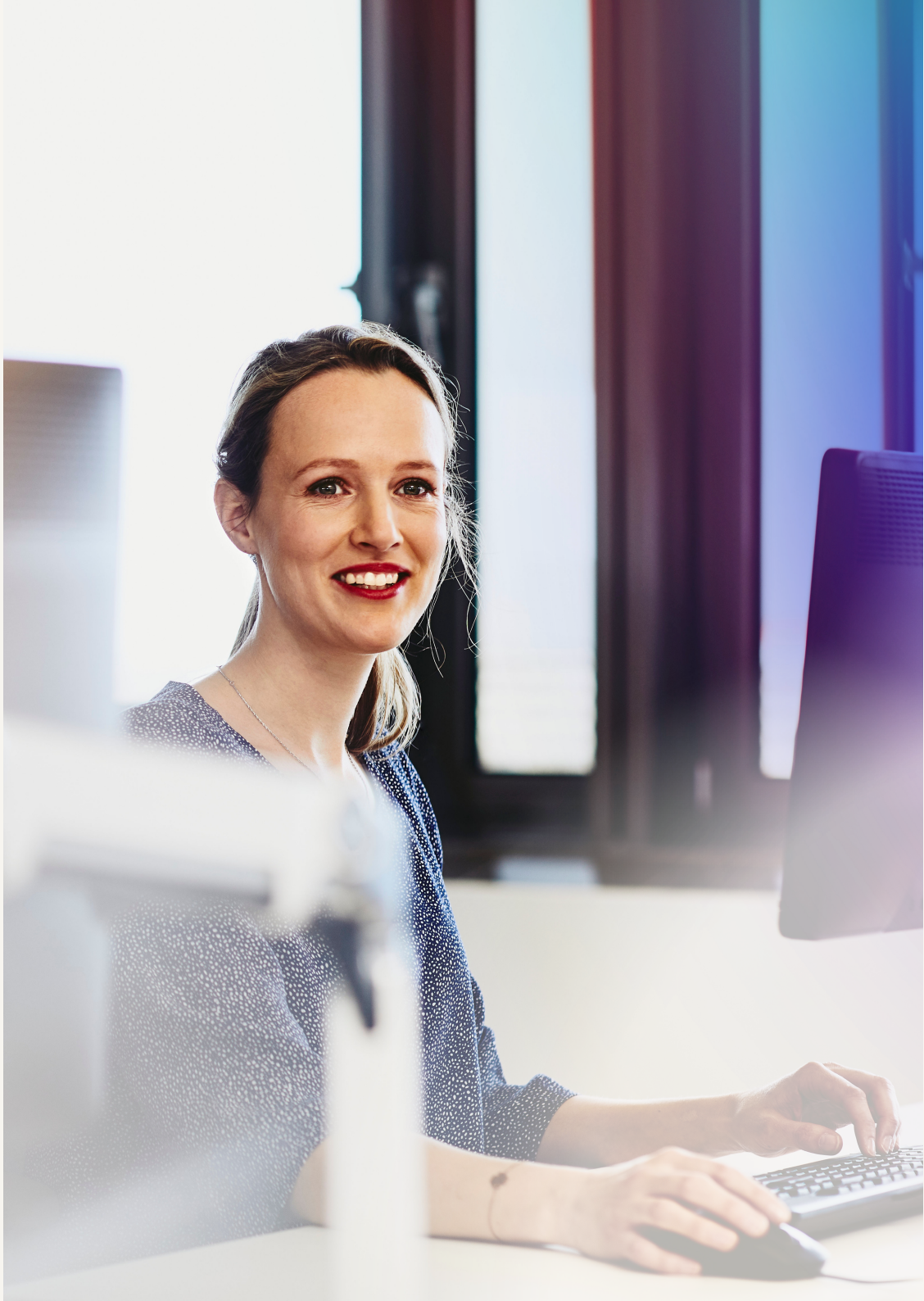
C is an imperative programming language, based on the language B. C is procedural: all code is part of a function. Even though C has typed data, the language still allows for conversion. Conversion is possible through casts: the data types don’t have to have the same size (although this is recommended).

C is close to the hardware. This has many advantages, but also some disadvantages. C can be used for many purposes, and is relatively fast. On the other hand, this programming language isn’t very forgiving when it comes to errors, and the compiler doesn’t do many checks.

rust.

Rust is an expression programming language, which was initially created as a project of Mozilla. This language is inspired by C and C++, but does differ quite a bit from these programming languages.

Rust focuses on speed, safety, reliability and productivity. The biggest advantage of Rust is its safety: the language prevents errors in memory use without the overhead of a garbage collector and provides concurrency without race conditions.



rust vs c.

So what are the main differences between these programming languages? We'll gladly list them for you.

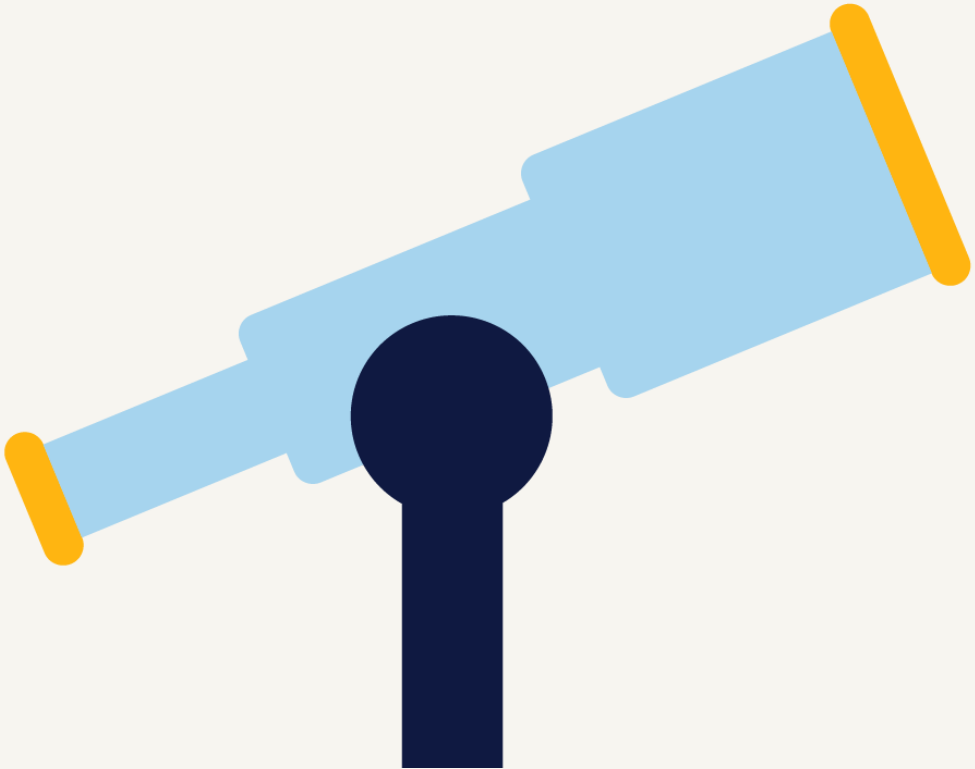
	c	rust
age	C was developed in 1972 and is 49 years old.	Mozilla developed Rust in 2010, so with 11 years on the calendar, this programming language is very young - but that doesn't mean the technology is still in its infancy.
syntax	C's syntax is, unlike other languages such as C# and Python, somewhat more difficult to read. Think of function pointers, pointer mathematics for arrays ... It's possible to write cryptic code in C that still compiles.	In its foundational syntax, Rust strongly resembles C. By applying zero-cost abstraction, the code becomes a lot more readable, and it can even be compared to for example C#.

	c	rust
safety	As mentioned earlier, the C compiler doesn't do many checks. A C program that has been deemed correct by the C compiler, doesn't necessarily work. That's where the saying 'C provides all the rope you need to hang yourself' comes from: you can do a lot with C, but only if you know what you're doing.	Rust is famous for being a safe programming language. Rust works with ownership: the language allows programming without the overhead of a garbage collector, and without fear of segmentation errors. Rust catches errors while compiling the code, and gives programmers the chance to fix problems before the bugged code can be executed.
speed	Large parts of popular operating systems such as Windows, UNIX and Linux are written in C. While the language is relatively old, it doesn't have too many competitors when it comes to speed and performance.	Rust is at least as quick and high-performing as many of the traditional programming languages. Rust may even be quicker than C or C++. We can conclude that the speed of both programming languages is very similar.

	c	rust
(raw) pointers	Memory location, addressing and manipulation is done through pointers. The programmer has to liberate the allocated memory to avoid memory leaks.	Rust's raw pointers are the equivalent of C's pointers. Rust mainly uses references or fat pointers. Memory is liberated automatically when it's no longer in use.
compiler	At this point in time, compiling code in C is quicker than compiling code in Rust. The C compiler doesn't check whether the memory is liberated after use.	The biggest advantage of Rust as compared to C is the borrow checker: the part of the compiler that makes sure references don't continue to exist after the data to which they refer is gone, which helps to eliminate bugs caused by memory issues.

	c	rust
dependencies	You'll have to integrate the project dependencies yourself. What's more, possible new versions have to be checked manually.	Rust has its own package manager, Cargo, which will introduce project dependencies automatically.
user friendliness	While it might be easier to learn to work with C, it is also less user-friendly. C executes fewer automatic checks in the compiler, which means it's easier to make mistakes in the code and there's a higher risk of bugs.	Rust's learning curve is quite a bit steeper than C's, but this programming language, with its automatic checks and coding without the overhead of a garbage collector, is easier to use and more user-friendly than C. Once you master Rust, working with this language is easy peasy lemon squeezy. But be careful: C and Rust both require a strong basic programming knowledge. You definitely have to know what's under the hood before you can use them.

	c	rust
what about the future?	C's latest update was in June 2018. This update didn't give us any new features, only technical corrections. The next version is not expected before 2022.	Rust only works with editions and quicker version updates. Each edition brings with it such big changes, that for example code written for edition 2018 is not guaranteed to compile when using the 2021 edition of the compiler/language. Editions are chosen when big changes are made that will improve the language on the long term.



ausy
by randstad.

our values.

entrepreneurship.

We aim to blow everyone away with our courage, flexibility and creativity.

trust.

By giving and receiving trust, we establish bonds that have a positive impact on everyone.

teamspirit.

We achieve even the most crazy goals by sharing knowledge and helping each other out. And by appreciating our colleagues and supporting their successes.

find a job on www.ausy.be