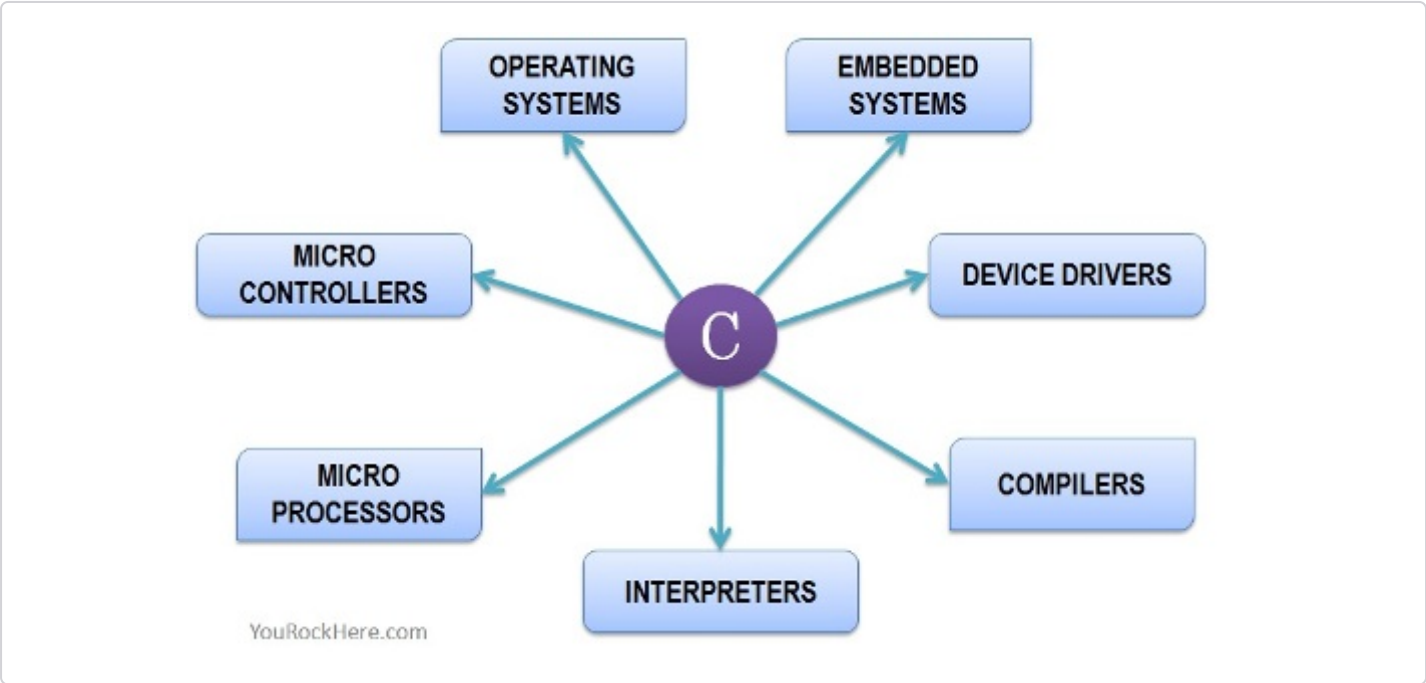
The Bottom Line

In the end, C is a much more efficient language than many of its peers. However, it doesn't work well when prototyping is involved.

My approach is to use **interpreted languages** like Python, R, Octave/Matlab, etc., for prototyping – that is, for exploring small amounts of data, for developing an approach, and algorithms, for analyzing data, and for generating graphics. When I have a computation, or a simulation, or a series of operations that are time-consuming, I think about implementing them in C.

> Interpreted languages should be used for **prototyping**, and C for **performance**.

On the practical side, the nature of C makes it optimal for many processing-based applications:



That’s it for our introduction into this language. There are some handy links in the next lesson. Of course, these are optional, but it never hurts to know more.