Myths

An overview of myths and prejudices about C++ in embedded programming and what are the reasons behind them.

WE'LL COVER THE FOLLOWING

As I started working in an embedded environment, I was astonished that there was so much prejudice against the usage of C++ in embedded programming. Most of them are based on the wrong understanding of the C++ programming language.

Some Common Myths

- Templates cause code bloat
- Objects have to be created on the heap
- Exceptions are expensive
- C++ is slow and needs too much memory
- C++ is too dangerous in safety-critical systems
- You must write object-oriented code in C++
- C++ can only be used for applications
- The `iostream` library is too big; the STL library too slow

In summation:

> *C++ is a cute toy but it cannot handle the challenging tasks.*

Prejudices

Prejudices

The list of prejudice is long consisting partially of *half-truth* and *untruth* statements often stated by experienced C programmers. I will only refer to the untruth statements. The half-truth statements are, to a large extent, questions due to the right usage of C++ and, to a small extent, questions of the implementation of the core and the libraries of C++.

- **Objects must live on the heap**.
    - Objects can be created on the stack or at an arbitrary position with the help of placement new.
- **C++ is too dangerous in safety-critical systems**
    - Of course, it depends on the experience of the developer. But whoever uses C strings instead of C++ strings; uses C arrays instead of C++ arrays; uses macros instead of constant expressions or templates, can not argue that C++ is not well suited for safety-critical systems. Honestly, the contrary holds. C++ has a lot to offer in safety-critical systems.
- **You must program object-oriented in C++**
    - C++ is a multi-paradigm language, meaning you can solve your problem in an object-oriented, structured, functional, generic, or generative style.
- **C++ can only be used for applications**
    - C++ can be used for many different kinds of products for example, fire extinguishers, defibrillators, and cars.

Reasons for Prejudices

What are the reasons for these half-truths? I think, there are more than one reasons.

- **Old C++ compilers**
    - Some programmers have knowledge that is based on old C++

compilers of the last millennium. These compilers implement the C++98 standard, but they have a large potential for optimization.

- **Training deficit**
    - Many embedded programmers have only learned C. Similarly, there is little time for programmers to experiment with new technologies, leading to a deficit in practice or knowledge about C++.
- **Transition from expert to novice**
    - You have to be brave enough to leave your position as a C expert and continue as a C++ novice.
- **Legacy codebase in C**
    - The existing codebase is in C, meaning that most programmers use C to fix a bug or implement a solution. There are a lot of standards that you have to fulfill. The courage to use new technologies seems to be inversely proportional to the pressure of the standards.
- **Many C experts**
    - There are many C experts who continue training the novices in C, making C++ a less respected programming tool.
- **Curse of the monoculture**
    - Embedded world is often a monoculture. I worked for 15 years as a consultant in the automobile area and used about 10 languages. On the contrary, I used only 3 languages in the embedded area.
- **Insufficient knowledge about C++**
    - Many developers do not have sufficient knowledge of the classical C++ and have no knowledge at all of modern C++.

Maybe, we will polarize with this lesson but if it helps to make the great features of modern C++ better known in the embedded world than we want to do it voluntarily. In the next lesson, we compare the myths with the facts.