Fortran

Elizabeth Bastow, David Ludington, Snigdha Upadhyay, Alex Frankel

Introduction

- Originally developed in the 1950s by IBM for scientific computing.
- One of the earliest high-level programming languages.
- Short for formula translation
- Third generation, compiled, imperative programming language that is especially suited to numeric computation and scientific computing.
- A popular language for high-performance computing
- Useful for programs that benchmark and rank the world's fastest supercomputers.

History

- Early Versions
  - Developed as a more practical alternative to assembly language
  - Fortran I (1957)
    - IF (expression) label1, label2, label3
  - Fortran II, III, and IV, evolution
    - Improvements on control structure, allowed for more direct comparison
- Fortran 77
  - A major turning point, as it introduced structured programming elements like IF-THEN-ELSE statements and character string handling.
- Modern Fortran
  - Fortran 90, 95, 2003, 2008, and 2018 introduced features such as:
    - Array programming (Fortran 90).
    - Object-oriented programming (Fortran 2003).
    - Parallel processing (Fortran 2008).

Language Features

Syntax and Structure

Control Structures	Loops, if - else, case structure is available
Array Handling	Simple syntax for handling large matrices and arrays
Parallelism	Newer versions provide parallel processing improving performance
Modules and Subroots	Efficient organization and reuse of code
Memory Management	These can be used in the template, and their size and color can be edited
Statically and Strongly Typed	Statically and strongly typed, which allows the compiler to catch many programming errors early on for you
High Performance	Fortran provides high performance when it comes to computationally intensive applications in science and engineering.

Classification

Determinisitc

Consistent output

Compiled

Converted directly to machine code

Imperative

Steps to completion

Procedural

Functions that run top-down

Principles

Abstraction

Procedures and modules allow for code reuse and organization

Automation

Mechanical and computational tasks

Portability

Designed with the goal of being portable between systems

Localized Costs

Large data set handling efficiently saves time and energy

Simplicity

Straightforward syntax

Preservation of Information

Representation of complex numerical data types

Evaluation

Readability	-Not a part of the C/Java class of languages -Easy to learn for first-time programmers -Older syntax
Writability/Productivity	- Limited nature/number of libraries make it less useful for general-purpose programming, compared to languages like Python. - Modern versions offer support for parallel computing
Reliability	- Offers backwards compatibility - Provides some error-handling features, such as control-structure terminators
Cost	Free
Community/Ecosystem	-Smaller but dedicated community -Narrowly used
Portability	-Not very portable, only offered on fewer platforms than comparable languages
Coolness Factor	Could not be lower

Code Sample

Matrix multiplication
Fortran vs python

```fortran
! Subroutine for matrix multiplication: C = A * B
subroutine matrix_multiply(A, B, C, n)
    real(dp), intent(in) :: A(:,:), B(:,:)
    real(dp), intent(inout) :: C(:,:)
    integer, intent(in) :: n
    integer :: i, j, k

    do i = 1, n
        do j = 1, n
            C(i, j) = 0.0_dp
            do k = 1, n
                C(i, j) = C(i, j) + A(i, k) * B(k, j)
            end do
        end do
    end do
end subroutine matrix_multiply
```

```python
def matrix_multiply(A, B, C, n):
    """Perform matrix multiplication C = A * B m
    for i in range(n):
        for j in range(n):
            C[i, j] = 0.0
            for k in range(n):
                C[i, j] += A[i, k] * B[k, j]
    return C
```

Fortran vs Modern languages

Use

- Fortran has specific use cases
- Many modern languages are more general purpose

Memory Allocation

- Fortran allows for fine control over memory allocation
- Most modern languages handel garbage collection and memory allocation automatically

Syntax

- Strictly typed
- More verbose

Most comparable

Julia

Fortran vs Python

Ease of Use

- Fortran: While modern versions are easier to use than early ones, Fortran is still less intuitive, especially for beginners..
- Python: It has a very simple and readable syntax. Python is considered one of the easiest languages to learn, making it ideal for beginners.

Performance

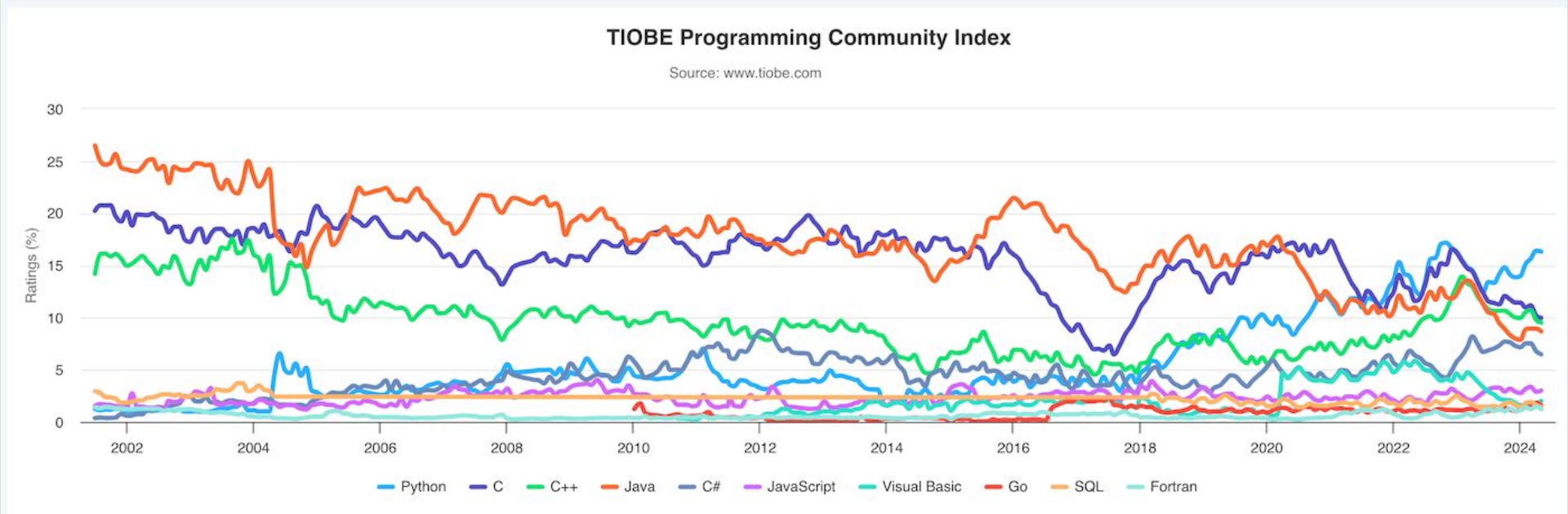
- Fortran: Excellent performance, especially for numerical and scientific computing. It’s highly optimized for heavy numerical tasks, and compilers can generate very efficient code.
- Python: Slower in execution compared to Fortran. However, performance can be significantly improved using libraries like NumPy, which are implemented in C or Fortran under the hood.

. Interoperability

- Fortran: Can be used with other languages (e.g., C) through interfaces, but the process is more complex than in Python.
- Python: Easily integrates with other languages (e.g., C, C++, Java) and tools, making it highly versatile.

Modern Usage

- High-Performance Computing:Fluid dynamics, astrophysics ertc.
- Legacy Systems: Many older but critical systems, like energy grids and defense simulations, still use FORTRAN.
- Financial and Scientific Analysis: Some financial modeling and statistical analysis software.



Fortran relevance today and in future

- Fortran is surprisingly relevant today because of its efficiency in computing.
- Provides High-Performance Computing (HPC) and Scientific Computing.
- Highly efficient in numeric calculations and parallel programming.
- Heavily used in computational extensive fields(weather forecasting, climate situation, huge physics and astro-physics calculation).
- Some of the instruments aboard NASA's Voyager spacecraft were programmed with Fortran. In 2015, NASA posted a job listing for the Voyager team that specified that hires had to have an expertise in Fortran.
- Interoperability makes it easier to have Fortran's speed while enjoying modern languages like Python's ease of use for data manipulation and visualization.
- Fortran's community and ecosystem is growing rapidly making it a language still very much in use.
- It is likely to remain a key language in specialized, computation-heavy domains for the foreseeable future, especially in academia and research.

Conclusion

- One of the earliest high-level programming languages.
- Modern Fortran provides control structure,array handling, parallelism and memory management
- Best for high-performance numerical computation, scientific research, and legacy systems in scientific domains.
- It might become more niche, focusing on applications where performance and precision are critical.
- Language’s evolution and interoperability with languages like Python will help Fortran maintain its place in scientific computing,
- It may face increasing competition from newer languages tailored for similar tasks.

Sources

https://en.wikipedia.org/wiki/Fortran

https://fortran-lang.org/

https://www.ibm.com/history/fortran

https://gcc.gnu.org/wiki/GFortranStandards

https://www.spiceworks.com/tech/tech-general/articles/what-is-fortran/