Historic Programming Languages

Sandro Lobbene

Toygun Ejder

Tufan Alacapunar

May 1, 2023

1 Motivation

2 Fortran

Fortran is a procedural programming language, which influenced many other programming languages. It was developed by IBM in the 1950s for scientific and engineering applications.[1] Fortran is a general-purpose, imperative programming language and plays a big role in numeric computation as well as scientific computing.
It is known for high-performance computing and is used for programs that benchmark and rank the world's fastest supercomputers.[2][4]
Fortran consists of numerous versions, which have added support for structured programming (FORTRAN 77), array programming, modular programming and generic programming (Fortran 90), high performance (Fortran 95), object-oriented programming since 2003 and concurrent programming in 2008.[1] Influenced languages by Fortran include BASIC, PL/I, and C, as well as Algol and Java.
Fortran is still used today for programming scientific and engineering applications. It is the primary language for some of the most intensive supercomputing tasks, such as weather forecasting, nuclear security and medicine. [3]
FORTRAN is short for "Formula Translation", because it was designed to allow easy translation of math formulas into code.
It is a compiled language, which means that the source code is compiled into a machine code before execution.[6]
Also, it is case-insensitive, which means that the keywords can be written in any case,

even though "FORTRAN" itself was written in uppercase letters, until Fortran 90. Fortran is a statically typed language, which means that the type of a variable is known at compile time.[5]

2.1 Features

2.1.1 Comments

A comment in Fortran starts with an exclamation mark (!) and ends at the end of the line.
Example: ! This is a comment

2.1.2 Data Types

Fortrans data types can be divided into two categories: intrinsic and derived types. Where intrinsic types are provided by the compiler. Derived types are user-defined types. The following intrinsic types are available in Fortran: Intrinsic Types

- Integer
- Real
- Complex
- Logical
- Character

2.1.3 Variables

Variables in Fortran are declared with the object type followed by "::" and the name of the variable. You can initialize a variable by assigning a value to it like this:
integer :: year
year = 2023

2.1.4 Operators

Fortran supports the following operators:

- Arithmetic Operators
- Relational Operators
- Logical Operators

Arithmetic Operators

- + Addition
- - Subtraction
- * Multiplication
- / Division
- ** Exponentiation
- // Concatenation [8]

Relational Operators

- .EQ. Equal to
- .NE. Not equal to
- .LT. Less than
- .LE. Less than or equal to
- .GT. Greater than
- .GE. Greater than or equal to

Logical Operators

- .not. Logical NOT
- .and. Logical AND
- .or. Logical OR
- .eqv. Logical equivalence
- .neqv. Logical non-equivalence

2.1.5 Control Structures

Control structures in Fortran are similar to other programming languages, such as C or Java. They make it possible to control the flow of a program.
Supported control structures are if-then, if-then-else, do, do-while, do-until, select-case and cycle.

2.1.6 Functions and Subroutines

Functions and subroutines are used to divide a program into smaller parts, which makes it easier to read and maintain, hence the name "structured programming" as one of Fortrans paradigms.
Functions and subroutines are similar, but they have some differences, such as that functions can return a value, while subroutines cannot. This is because subroutines are used to perform a task, while functions are used to calculate a value.

2.1.7 Arrays

Arrays are used to store multiple values of the same type. The main difference between arrays in Fortran and other programming languages is that Fortran arrays start at 1, while most other programming languages start at 0.[9]
An array in Fortran can have up to 7 dimensions.
The advantage of arrays is that they make it possible to store multiple values in one variable, which makes it easier to access them.

2.1.8 Strings

Strings are used to store text. They are declared with the character type and can be initialized with a string literal.
Example:
character :: name = "John"
Strings can be concatenated with the // operator.
The length of a string can be determined with the len() function like this:
len(name)
One can also access a single character of a string with the following syntax:
name(1:1)

This would return the first character of the string.
The length of a string can be limited by adding a number after the type declaration like this:
character(10) :: name
This would limit the length of the string to 10 characters.

2.1.9 Input and Output

Input and output in Fortran is done with the read() and write() functions.
The read() function is used to read input from the user and the write() function is used to write output to the console.
Example:
read(*,*) name
write(*,*) "Hello ", name
The asterisk (*) is used to indicate that the input or output is done to the console.
The first asterisk is used to indicate the format of the input or output.
The second asterisk is used to indicate the type of the input or output.
An example where the type is specified would be:
read(*,*) integer :: number
This would read an integer from the console, such as the amount of times a loop should be executed.

2.1.10 File Handling

File handling in Fortran is done with the open(), close(), read() and write() functions.
The names of the functions are self-explanatory.
Example:
open(unit=1, file="file.txt", status="old")
close(unit=1)
read(1,*) name
write(1,*) "Hello ", name
The unit parameter is used to specify the file to be used.
The file parameter is used to specify the name of the file.
The status parameter is used to specify the status of the file (old, new, scratch, replace).
The asterisk (*) is used to indicate the format of the input or output.
The second asterisk is used to indicate the type of the input or output.

2.1.11 Object-Oriented Programming

Fortran is not an object-oriented programming language, but it is possible to use object-oriented programming in Fortran.
This is done by using modules.
Modules are used to group variables, functions and subroutines together.
Example:

```
module myModule
integer :: number
contains
subroutine mySubroutine()
write(*,*) "Hello World"
end subroutine mySubroutine
end module myModule
```

This module can then be used in another program like this:

```
program myProgram
use myModule
call mySubroutine()
end program myProgram
```

This would print "Hello World" to the console.

2.1.12 Parallel Programming

Parallel programming in Fortran is done with the OpenMP API.
OpenMP is a library that makes it possible to write parallel programs in Fortran.
Example:

```
program myProgram
use omp_lib
integer :: i
!$OMP PARALLEL DO
do i = 1, 10
write(*,*) i
end do
!$OMP END PARALLEL DO
end program myProgram
```

This program would print the numbers 1 to 10 to the console.

2.2 Usage

2.3 Advantages and Disadvantages

2.3.1 Advantages

- Fortran is a compiled language, which means that it is faster and more secure than interpreted languages like Python.
- Fortran is a statically typed language, which means that it is faster and more secure than dynamically typed languages like Python.
- Fortran is a structured programming language, which means that it is easier to read and maintain than unstructured programming languages like assembly.
- Fortran is a portable language, which means that it can be used on multiple platforms.
- Fortran is a parallel programming language, which means that it can be used to write parallel programs.

2.3.2 Disadvantages

- Fortran is an old language, which means that it is not as popular as newer languages like Python.
- Fortran is not an object-oriented programming language, which means that it is not as flexible as object-oriented programming languages like Java.
- Fortran is not a functional programming language, which means that it is not as flexible as functional programming languages like Haskell.

2.4 Conclusion

Fortran is an old programming language that is still used today.
It is a compiled, statically typed, structured, portable and parallel programming language, which makes it a good choice for scientific computing. However, it is neither an object-oriented nor a functional programming language, which makes it less flexible

than other programming languages. The readability of Fortran is also not as good as other programming languages, which makes it harder to learn. For beginners, it is recommended to start with a more modern programming language like Python or Java.

3 References