Historic Programming Languages

Sandro Lobbene
Toygun Ejder
Tufan Alacapunar
03.05.2023

Abstract

In today's world, there is an overwhelming variety of programming languages, which leads to learning a modernized language. However, understanding how programming languages evolved in the past decades, will lay a great foundation to comprehend certain concepts, which are still relevant to this day in the field of computer science. This allows us to gain insights into how to improve coding in the future. Fortran, Lisp and ALGOL 60 all contributed indispensable features, even though they had limitations.

1. 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][3]
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. [4]
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, since old keyboards did not feature a shift key. Fortran is a statically typed language, which means that the type of a variable is known at compile time.[5]

1.1 Features

1.1.1 Basic structure

A Fortran program consists of a main program and one or more subprograms. The main program is the starting point of the program and declared with the keyword "*program*", followed by the name of the program. In the end, the program is terminated with the keyword "*end program*".
A comment in Fortran starts with an exclamation mark (!) and ends at the end of the line.
Example:

```
program helloworld ! This is a comment
print *, "Hello World!"
end program helloworld
```

1.1.2 Data Types

Fortran's data types can be divided into two categories: intrinsic and derived types. While intrinsic types are provided by the compiler. Derived types are user-defined. Fortran's intrinsic Types are

Integer, *Real*, *Complex*, *Logical* and *Character*. The number of bytes used by these types can be specified with the "*kind*" keyword.
Integer is used to store whole numbers, comparable to the int type in C. It can be declared with the integer keyword. Complex is used to store complex numbers, which consist of a real and an imaginary part. It can be declared with the complex keyword. The real and imaginary part can be accessed with the "*real"* and "*aimag*" functions. Real is used to store floating point numbers, comparable to the float type in C. It can be declared with the real keyword. Logical is used to store boolean values, which can be either true or false. It can be declared with the logical keyword. and initialized with either .true. or .false. Character is used to store a single character or an entire String. It can be declared with the character keyword, while also specifying the length of the string.

1.1.4 Variables

similarly to Java, 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
Variables can also be initialized when they are declared like this: "*integer :: year = 2023*"
Constants are declared with the parameter keyword. Like this:
parameter (pi = 3.1415926535)
Or like this: *real, parameter :: pi = 3.1415926535* [7]

1.1.5 Operators

Fortran supports Arithmetic, Relational and Logical Operators.

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

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

Logical Operators	Logical not	Logical and	Logical or	Logical equivalence	Logical non-equivalence
	.NOT.	.AND.	.OR.	.EQV.	.NEQV.

1.1.6 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, select-case and cycle. [9] The if-then control structure is used to execute a block of code if a condition is true. The if-then-else control structure is used to execute a block of code if a condition is true and a different block of code if the condition is false. The do control structure is comparable to the for loop, the do-while loop matches the while loop in Java.
Furthermore, the select case is an ancestor of the switch statement and the break and continue keyword were influenced by the exit and cycle keyword from Fortran.

1.1.7 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 Fortran's 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.

1.1.8 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.[10]An array in Fortran can have up to 7 dimensions. Arrays make it possible to store multiple values in one variable, which makes it easier to access them.
real, dimension(5) :: Entries
Entries(1) = 3.14
Entries(2) = 2.718

Entries(1)	Entries(2)	Entries(3)	Entries(4)

1.1.9 Strings

Strings are used to store text. They are declared with the character type and can be initialized with a string literal:
character :: name = "John"
Strings can be concatenated with the // operator. *character doublename = name//name*
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.

1.1.10 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. [12]

1.1.11 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 first asterisk (*) is used to indicate the format of the input or output, and the second asterisk is used to indicate the type of the input or output. [13]

1.1.12 Object-Oriented Programming

Fortran is not an object-oriented programming language, but it is possible to use object-oriented programming in Fortran by using modules. Those 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.

1.1.13 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.

1.2 Conclusion

Fortran is an old programming language, since it was the first high-level 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 due to its stability. 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.