Fortran Programming

By

Dr.Vinod Kumar Singh

Associate Professor Physics K .S. Saket P G College Ayodhya

Email: vks423272@gmail.com

FORTRAN, in full **Formula Translation**, computer-programming language created in 1957 by John Backus that shortened the process of programming and made computer programming more accessible.

The creation of FORTRAN, which debuted in 1957, marked a significant stage in the development of computer-programming languages. Previous programming was written in machine (first-generation) language or assembly (second-generation) language, which required the programmer to write instructions in binary or hexadecimal arithmetic. Frustration with the arduous nature of such programming led Backus to search for a simpler, more accessible way to communicate with computers. During the three-year development stage, Backus led an eclectic team of 10 International Business Machines (IBM) employees to create a language that combined a form of English shorthand with algebraic equations.

FORTRAN enabled the rapid writing of computer programs that ran nearly as efficiently as programs that had been laboriously hand coded in machine language. As computers were rare and extremely expensive, inefficient programs were a greater financial problem than the lengthy and painstaking development of machine-language programs. With the creation of an efficient higher-level (or natural) language, also known as a third-generation language, computer programming moved beyond a small

coterie to include engineers and scientists, who were instrumental in expanding the use of computers.

By allowing the creation of natural-language programs that ran as efficiently as hand-coded ones, FORTRAN became the programming language of choice in the late 1950s. It was updated a number of times in the 1950s and 1960s in order to remain competitive with more contemporary programming languages. FORTRAN 77 was released in 1978, followed by FORTRAN 90 in 1991 and further updates in 1996 and 2004. However, fourth- and fifth-generation languages largely supplanted FORTRAN outside academic circles beginning in the 1970s.

A Fortran program is made of a collection of program units like a main program, modules, and external subprograms or procedures.

Each program contains one main program and may or may not contain other program units. The syntax of the main program is as follows −

```
program program_name

implicit none

! type declaration statements

! executable statements

end program program_name
```

A Simple Program in Fortran

Let's write a program that adds two numbers and prints the result −

Live Demo

```
program addNumbers
```

```
! This simple program adds two numbers

 implicit none

! Type declarations

 real :: a, b, result

! Executable statements

 a = 12.0

 b = 15.0

 result = a + b

 print *, 'The total is ', result

end program addNumbers
```

When you compile and execute the above program, it produces the following result −

The total is 27.0000000

Please note that −

- All Fortran programs start with the keyword **program** and end with the keyword **end program,** followed by the name of the program.

- The **implicit none** statement allows the compiler to check that all your variable types are declared properly. You must always use **implicit none** at the start of every program.

- Comments in Fortran are started with the exclamation mark (!), as all characters after this (except in a character string) are ignored by the compiler.

- The **print *** command displays data on the screen.
- Indentation of code lines is a good practice for keeping a program readable.
- Fortran allows both uppercase and lowercase letters. Fortran is case-insensitive, except for string literals.

Basics

The **basic character set** of Fortran contains −

- the letters A ... Z and a ... z
- the digits 0 ... 9
- the underscore (_) character
- the special characters = : + blank - * / ( ) [ ] , . $ ' ! " % & ; <> ?

Tokens are made of characters in the basic character set. A token could be a keyword, an identifier, a constant, a string literal, or a symbol.

Program statements are made of tokens.

Identifier

An identifier is a name used to identify a variable, procedure, or any other user-defined item. A name in Fortran must follow the following rules −

- It cannot be longer than 31 characters.
- It must be composed of alphanumeric characters (all the letters of the alphabet, and the digits 0 to 9) and underscores (_).
- First character of a name must be a letter.

- Names are case-insensitive

Keywords

Keywords are special words, reserved for the language. These reserved words cannot be used as identifiers or names.

The following table, lists the Fortran keywords −

The non-I/O keywords				
allocatable	allocate	assign	assignment	block data
call	case	character	common	complex
contains	continue	cycle	data	deallocate
default	do	double precision	else	else if
elsewhere	end block data	end do	end function	end if
end interface	end module	end program	end select	end subroutine

end type	end where	entry	equivalence	exit
external	function	go to	if	implicit
in	inout	integer	intent	interface
intrinsic	kind	len	logical	module
namelist	nullify	only	operator	optional
out	parameter	pause	pointer	private
program	public	real	recursive	result
return	save	select case	stop	subroutine
target	then	type	type()	use
Where	While			
The I/O related keywords				

backspace	close	endfile	format	inquire
open	print	read	rewind	Write

FORTRAN as a Programming Language

The FORTRAN programming language was conceived in the early 1950s the name produced from the two words FORmula TRANslation. In 1966 the language was standardized and FORTRAN IV was born.

Revision of the language led to FORTRAN 77, the language we use today. The standard for FORTRAN 90 is now available although not yet in widespread use. F77 is a subset of F90.

FORTRAN was designed for scientists and engineers, and has dominated this field. For the past 30 years FORTRAN has been used for such projects as the design of bridges and aeroplane structures, it is used for factory automation control, for storm drainage design, analysis of scientific data and so on.

Throughout the life of this language, groups of users have written libraries of useful standard FORTRAN programs.

These programs can be borrowed and used by other people who wish to take advantage of the expertise and experience of the authors, in a similar way in which a book is borrowed from a library.

High performance

Fortran has been designed from the ground-up for computationally intensive applications in science and engineering. Mature and battle-tested compilers and libraries allow you to write code that runs close to the metal, fast.

Statically and strongly typed

Fortran is statically and strongly typed, which allows the compiler to catch many programming errors early on for you. This also allows the compiler to generate efficient binary code.

Easy to learn and use

Fortran is a relatively small language that is surprisingly easy to learn and use. Expressing most mathematical and arithmetic operations over large arrays is as simple as you'd write them as equations on a whiteboard.

Versatile

Fortran allows you to write code in a style that best fits your problem: Imperative, procedural, array-oriented, object-oriented, or functional.

Natively parallel

Fortran is a natively parallel programming language with intuitive array-like syntax to communicate data between CPUs. You can run almost the same code on a single CPU, on a shared-memory multicore system, or on a distributed-memory HPC or cloud-based system. Coarrays, teams, events, and collective subroutines allow you to express different parallel programming patterns that best fit your problem at hand.