# FORTAN BY TRAN

A PREPRINT

**Graham S. Walker**
gwalker1@email.arizona.edu

**Akshith Thumma**
akshiththumma@email.arizona.edu

April 4, 2020

**ABSTRACT**

wow how abstract.

***K*eywords** First keyword · Second keyword · More

## 1 Introduction

## 2 History

The history of Fortran is very different from the history of many high-level languages, because FORTRAN was the first high-level language, using the first compiler ever developed. Fortran was developed by a team of programmers at IBM led by John Backus, and was first published in 1957. At first, one might be surprised why anyone would develop a language like Fortran when it reduces the execution efficiency by 20 compared to assembly language, but the programmers at IBM saw a bigger picture which would change the way of writing programs for complex mathematical expressions for a long time in future and is also simple for people to learn. Yes, the big advantage was that unlike assembly, everyone found Fortran very easy to learn and code 500 faster than before. This revolutionary language was one of the big stepping stones because it was not just the first high-level language, but also because it introduced one of the greatest concepts to the field of Computer Science, The Compiler Theory.

Due to its different dialects, Fortran faced a lot of criticism and had to be revised more number of times than you would imagine. Although they released a standardized version in 1966 as FORTRAN ‘66, they had to review it again in 1978 due to new dialects being surfaced, but in 1990 a new fortran came into existence with a lot more features and blended into the new age programming perfectly fine. Today, FORTRAN ‘90 still stands high in the list due to various reasons and one of the most important ones being the knowledge transfer IBM has managed to have. Almost every old age programmer knows Fortan and at a stage even new age programmers become familiar with it due to the usage of the language by many big companies like Intel, Android and HP.It is not just the language and the software but also its impact on the hardware such as the Raspberry Pi, which contributes towards the success of Fortran in the field of Computer Science.

## 3 Control Structures

In many programming languages we know that the flow of the program is controlled by the control structures. In FORTRAN we have a variety of control structures right from the Boolean checks to switch statements to Do Loops. Although FORTRAN is not completely different from many high-level languages, control structures in FORTRAN operate a bit different from Java and C.

In FORTRAN we have three main if conditions we might be able to use: Arithmetic if, Logical if and Block if. Arithmetic If: This if condition was introduced with the very first version of FORTRAN and is a bit different from modern high level programming languages. This type of conditionals use goto statements to execute what the programmer wishes to execute. Although due to the dangers goto statements possess, Fortran 2018 no longer supports this kind of if conditionals. In this format the “if” checks the condition specified by the programmer and the “goto”

executes a jump to the statement associated with the goto and the else case is handled by an extra goto which is executed in case all the if checks return false.

Logical If: Before FORTRAN 77 was introduced FORTRAN IV was one version of the language where logical if conditionals were introduced to make if conditionals more efficient and avoid the use of goto statements. Although developers of the language tried to avoid the goto statements with Logical If statements, it was not completely achieved because after a Logical If condition only one statement declaration was allowed and it was not the best solution with large programs. With Logical If the use of goto statements was not necessary and programmers could control the flow of the program smoothly unlike the Arithmetic conditions. In this format the "if" checks the condition and executes the one line which has been declared by the programmer and continues with the rest of the program without executing the conditions if the condition evaluates to be false.

Block If: One could say a Block If conditional is the perfect if condition in FORTRAN which is now close enough to a lot of popular high level programming languages like Java. This kind of if conditional is very similar to Haskell's if-then-else format of conditions. The main advantage of this type of condition is that one can avoid goto statements and at the same time execute multiple statements unlike the above two conditionals. In this format the "if" checks the conditions an the then does the execution and finally the "else" handles the false case of the condition.

```
if (x .gt. 0) then
        print *, "x is greater than 0"
else
        print *, "x is not greater than 0"
end if
```

Select Switches: According to the previous paragraph about the Block If condition being the more prioritized if condition, a select switch format of conditioning basically acts as a substitute to the Block If statement. In this kind of conditioning the select case statements opens up the list of cases we are about to check and each case statement acts as and if condition checking and comparing the condition in the current case to the base case which is also known as the select case. The command to be executed after a case has been matched will follow the case statement and again gives the programmer the advantage of avoiding goto statements. The "end select" statement indicates that the select switch block has been completed.

```
integer :: n = 3

message: select case (n)
        case (0)
                print *, "not too great"
        case (2)
                print *, "wow, what a treat"
        case (3)
                print *, "omg, like , totes awesome"
        case default
                print *, "no, absolutely not"
end select message
```

Control structures in FORTRAN also include loops like the Do Loop and Do-while Loop to execute a command or commands repetitively until a condition is met. Below are the most useful loops in FORTRAN: Do Loop: A Do Loop executes a list of commands until an "exit" occurs. This is like an uncontrolled loop and if there is no way the pointer reaches the exit statement; we might experience and infinite loop which would not be the most desired solution by a programmer. Below is an example of how Do Loop does recurring execution of commands.

```
do i = 0,20,2  !the 2 is the step size. It is optional
        print *, i
end do
```

Do-while Loop: A Do-while Loop is very similar to a Do Loop, but in this format of the loop, the pointer does not look or wait for an exit statement, but executes the list of commands until a certain condition is met. The main advantage with a Do-while Loop is that it can be a bit more controlled by the programmer and helps avoid fatal infinite loops. Below is an example of how the Do-while loop does recurring execution of commands.

```
integer :: i = 0
```

```
do while (i .lt. 10)
        print *, i
        i = i + 1
end do
```

Both the above loops use incrementation to reach the end statement of the loop i.e. “end do”. When the pointer reaches the end do statement, it is a quick message to the compiler that all the iterations of the loop are completed in order for the program to continue with the flow of the program.

## 4 Data Types

## 5 Subprograms

## 6 Summary

`https://www.ctan.org/pkg/booktabs`

## References

[1] George Kour and Raid Saabne. Real-time segmentation of on-line handwritten arabic script. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 417–422. IEEE, 2014.

[2] George Kour and Raid Saabne. Fast classification of handwritten on-line arabic characters. In *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*, pages 312–318. IEEE, 2014.

[3] Guy Hadash, Einat Kermany, Boaz Carmeli, Ofer Lavi, George Kour, and Alon Jacovi. Estimate and replace: A novel approach to integrating deep neural networks with existing applications. *arXiv preprint arXiv:1804.09028*, 2018.