FORTRAN 95

Connor Richardson
cbrichardson6@email.arizona.edu

Sean Callahan
seanpcallahan@email.arizona.adu

April 14, 2020

ABSTRACT

1 Introduction

2 History

Fortran was initially developed by IBM to provide developers working on the IBM 704 Mainframe Computer with an alternative to assembly language. Because of hardware limitations at the time (1952-1956) Fortran prioritized performance with its the first Fortran compiler being an optimizing compiler, which brought its performance significantly closer to programs written with assembly language. Fortran's primary focus is scientific computing, primarily in use in scientific research and engineering.

Today, Fortran, although significantly less popular than in previous decades, is still a supported language, with revisions as recent as 2018. Its reach, however, has spread to encompass much more than its own development, but has also built a foundation for a number of dialects and entirely new languages, including BASIC.

3 Control Structures

Because of Fortran's age, the controls structures have shifted to accurately match common tropes in programming during each era. While the first versions supported what is known as the arithmetic if statement, which allows programmers to jump to one of three labels, with a conditional that determined which label by the sign of the operand. Modern versions of Fortran, don't encourage its use, with it finally losing full support in Fortran 2018. [2]

Fortran, and more specifically Fortran 95, supports various forms of the if conditional, first gaining support of the arithmetic if statement and in FORTRAN IV and Fortran 77, gaining an if conditional that is most similar to what we see in many other programming languages today. The if statement block works much like an if block in C or Java, with a conditional followed by code, but instead of the common curly braces separating the code from the conditional, this is done with the word "then" to signify the start of an if or else, and "end if" to end the contained code block. Closely related to the if statement, Fortran 95 also supports the switch case in a way very similar to that of Java. [1]

The primary control structure used in Fortran 95 is the do Loop, which is effectively a "while True" loop. This form of loop continuously repeats the loop until the exit operation is used, acting as a break. Curiously, however, the do loop also behaves very similarly to the for loop, allowing the creation of an iteration variable:

```
do i = 0, 100, 1
    print *, "Hello World"
end do
```

Much like list comprehensions, the do can be used to populate arrays by having what is essentially the beginning of a do loop contained in the array:

```
integer :: nums(5) =  [(i, i = 0,4)]
```

Perhaps the most interesting control structure in Fortran 95 is the where statement. Primarily used for modifying specific elements in an array, masking out elements that we don't wish to change. This simultaneously acts as a loop, while also acting as a conditional for each element in the array being iterated over. An example of this would be a snippet of code that puts a min and max value on a set of numbers:

```
integer :: grades(10) = [90, -2, 100, 101, 85]
where (grades > 100)
    grades = 100
else where (grades < 0)
    grades = 0
end where
```

4 Data Types

For most of Fortran's data types they closely resemble that of C. The CHARACTER type in Fortran works as one would expect in many C variant languages, where a character is a single byte, and can be strung together into arrays in order to create strings. One key difference, however, is that the user can specify what kind of encoding is desired for the CHARACTER or string of characters. This could be ASCII, UNICODE, etc., although some compilers may not support all languages and encodings.

Much like the CHARACTER type, Fortran also supports LOGICALs, which are roughly equivalent to Booleans in Java, holding a true or false value and are declared as:

```
.TRUE.
```

Fortran also has support for data types like INTEGER and REAL, which, once again, function similarly to ints and floats that we already know. Much like C, we can specify how many bytes we want an integer to be by using the kind parameter. This also allows the programmer to specify a variable to be of double precision instead of the standard floating point precision. In order to do this, you add the statement:

```
kind = 8 /*where 8 would be the number of bytes you desire*/
```

So, to specify to initialize a variable with a C equivalent of a long long int on a 32 bit machine, we can simply say

```
integer(kind=8):: c_long_long_int
```

Where Fortran really starts to differ from C and Java is in its support of complex numbers. Storing the complex number in what is essentially a tuple, such that (a, b) is equivalent to the expression a - bi.

Fortran, since it is a mathematically focused language, does support arrays, in up to seven dimensions. These can be declared using the attribute dimension directly following the type of the array.

```
integer, dimension(10,10) :: grades_and_ids
```

5 Subprograms

6 Summary

References

[1] John W. Backus. The history of fortran i, ii, and III. *IEEE Annals of the History of Computing*, 20(4):68–78, 1998.

[2] Ian D. Chivers and Jane Sleightholme. *Introduction to Programming with Fortran - With Coverage of Fortran 90, 95, 2003, 2008 and 77, Third Edition*. Springer, 2015.