FORTRAN - STRINGS

The Fortran language can treat characters as single character or contiguous strings.

A character string may be only one character in length, or it could even be of zero length. In Fortran, character constants are given between a pair of double or single quotes.

The intrinsic data type **character** stores characters and strings. The length of the string can be specified by **len specifier**. If no length is specified, it is 1. You can refer individual characters within a string referring by position; the left most character is at position 1.

String Declaration

Declaring a string is same as other variables:

```
type-specifier :: variable_name
```

For example,

```
Character(len=20) :: firstname, surname
```

you can assign a value like,

```
character (len=40) :: name
name = “Zara Ali”
```

The following example demonstrates declaration and use of character data type:

```
program hello
implicit none

   character(len=15) :: surname, firstname
   character(len=6) :: title
   character(len=25)::greetings

   title = 'Mr.'
   firstname = 'Rowan'
   surname = 'Atkinson'
   greetings = 'A big hello from Mr. Beans'

   print *, 'Here is', title, firstname, surname
   print *, greetings

end program hello
```

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

```
Here is Mr. Rowan Atkinson
A big hello from Mr. Bean
```

String Concatenation

The concatenation operator //, concatenates strings.

The following example demonstrates this:

```
program hello
implicit none

   character(len=15) :: surname, firstname
   character(len=6) :: title
   character(len=40):: name
   character(len=25)::greetings

   title = 'Mr.'
   firstname = 'Rowan'
   surname = 'Atkinson'

   name = title//firstname//surname
   greetings = 'A big hello from Mr. Beans'

   print *, 'Here is', name
   print *, greetings

end program hello
```

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

```
Here is Mr. Rowan Atkinson
A big hello from Mr. Bean
```

Extracting Substrings

In Fortran, you can extract a substring from a string by indexing the string, giving the start and the end index of the substring in a pair of brackets. This is called extent specifier.

The following example shows how to extract the substring ‘world’ from the string ‘hello world’:

```
program subString

   character(len=11)::hello
   hello = "Hello World"
   print*, hello(7:11)

end program subString
```

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

```
World
```

Example

The following example uses the **date_and_time** function to give the date and time string. We use extent specifiers to extract the year, date, month, hour, minutes and second information separately.

```
program  datetime
implicit none

   character(len = 8) :: dateinfo ! ccyymmdd
   character(len = 4) :: year, month*2, day*2

   character(len = 10) :: timeinfo ! hhmmss.sss
   character(len = 2)  :: hour, minute, second*6

   call  date_and_time(dateinfo, timeinfo)

   !  let’s break dateinfo into year, month and day.
   !  dateinfo has a form of ccyymmdd, where cc = century, yy = year
   !  mm  = month and dd = day

   year  = dateinfo(1:4)
   month = dateinfo(5:6)
   day   = dateinfo(7:8)

   print*, 'Date String:', dateinfo
   print*, 'Year:', year
   print *,'Month:', month
   print *,'Day:', day

   !  let’s break timeinfo into hour, minute and second.
   !  timeinfo has a form of hhmmss.sss, where h = hour, m = minute
   !  and s = second

   hour   = timeinfo(1:2)
   minute = timeinfo(3:4)
   second = timeinfo(5:10)

   print*, 'Time String:', timeinfo
   print*, 'Hour:', hour
   print*, 'Minute:', minute
   print*, 'Second:', second

end program  datetime
```

When you compile and execute the above program, it gives the detailed date and time information:

```
Date String: 20140803
   Year: 2014
   Month: 08
   Day: 03
   Time String: 075835.466
   Hour: 07
   Minute: 58
   Second: 35.466
```

Trimming Strings

The **trim** function takes a string, and returns the input string after removing all trailing blanks.

Example

```
program trimString
implicit none

   character (len=*), parameter :: fname="Susanne", sname="Rizwan"
   character (len=20) :: fullname

   fullname=fname//" "//sname !concatenating the strings

   print*,fullname,", the beautiful dancer from the east!"
   print*,trim(fullname),", the beautiful dancer from the east!"

end program trimString
```

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

```
Susanne Rizwan, the beautiful dancer from the east!
Susanne Rizwan, the beautiful dancer from the east!
```

Left and Right Adjustment of Strings

The function **adjustl** takes a string and returns it by removing the leading blanks and appending them as trailing blanks.

The function **adjustr** takes a string and returns it by removing the trailing blanks and appending them as leading blanks.

Example

```
program hello
implicit none
```

```
   character(len=15) :: surname, firstname
   character(len=6) :: title
   character(len=40):: name
   character(len=25):: greetings

   title = 'Mr. '
   firstname = 'Rowan'
   surname = 'Atkinson'
   greetings = 'A big hello from Mr. Beans'

   name = adjustl(title)//adjustl(firstname)//adjustl(surname)
   print *, 'Here is', name
   print *, greetings

   name = adjustr(title)//adjustr(firstname)//adjustr(surname)
   print *, 'Here is', name
   print *, greetings

   name = trim(title)//trim(firstname)//trim(surname)
   print *, 'Here is', name
   print *, greetings

end program hello
```

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

```
Here is Mr. Rowan  Atkinson
A big hello from Mr. Bean
Here is Mr. Rowan Atkinson
A big hello from Mr. Bean
Here is Mr.RowanAtkinson
A big hello from Mr. Bean
```

Searching for a Substring in a String

The index function takes two strings and checks if the second string is a substring of the first string. If the second argument is a substring of the first argument, then it returns an integer which is the starting index of the second string in the first string, else it returns zero.

Example

```
program hello
implicit none

   character(len=30) :: myString
   character(len=10) :: testString

   myString = 'This is a test'
   testString = 'test'

   if(index(myString, testString) == 0)then
      print *, 'test is not found'
   else
      print *, 'test is found at index: ', index(myString, testString)
   end if

end program hello
```

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

```
test is found at index: 11
```