FORTRAN - CHARACTERS

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

Characters could be any symbol taken from the basic character set, i.e., from the letters, the decimal digits, the underscore, and 21 special characters.

A character constant is a fixed valued character string.

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.

Character Declaration

Declaring a character type data is same as other variables:

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

For example,

```
character :: reply, sex
```

you can assign a value like,

```
reply = ‘N’
sex = ‘F’
```

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
```

Concatenation of Characters

The concatenation operator //, concatenates characters.

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
```

Some Character Functions

The following table shows some commonly used character functions along with the description:

Function	Description
len*string*	It returns the length of a character string
index *string, sustring*	It finds the location of a substring in another string, returns 0 if not found.
achar*int*	It converts an integer into a character
iachar*c*	It converts a character into an integer
trim*string*	It returns the string with the trailing blanks removed.
scan *string, chars*	It searches the "string" from left to right $unlessback = .true.$ for the first occurrence of any character contained in "chars". It returns an integer giving the position of that character, or zero if none of the characters in "chars" have been found.
verify *string, chars*	It scans the "string" from left to right $unlessback = .true.$ for the first occurrence of any character not contained in "chars". It returns an integer giving the position of that character, or zero if only the characters in "chars" have been found
adjustl*string*	It left justifies characters contained in the "string"
adjustr*string*	It right justifies characters contained in the "string"
len_trim *string*	It returns an integer equal to the length of "string" $len(string)$ minus the number of trailing blanks
repeat *string, ncopy*	It returns a string with length equal to "ncopy" times the length of "string", and containing "ncopy" concatenated copies of "string"

Example 1

This example shows the use of the **index** function:

```
program testingChars
implicit none
```

```
   character (80) :: text
   integer :: i

   text = 'The intrinsic data type character stores characters and   strings.'
   i=index(text,'character')

   if (i /= 0) then
      print *, ' The word character found at position ',i
      print *, ' in text: ', text
   end if

end program testingChars
```

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

```
The word character found at position 25
in text : The intrinsic data type character stores characters and strings.
```

Example 2

This example demonstrates the use of the **trim** function:

```
program hello
implicit none

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

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

   print *, 'Here is', title, firstname, surname
   print *, 'Here is', trim(title),' ',trim(firstname),' ', trim(surname)

end program hello
```

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

```
Here is Mr. Rowan Atkinson
Here is Mr. Rowan Atkinson
```

Example 3

This example demonstrates the use of **achar** function

```
program testingChars
implicit none

   character:: ch
   integer:: i

   do i=65, 90
      ch = achar(i)
      print*, i, ' ', ch
   end do

end program testingChars
```

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

```
65   A
66   B
67   C
68   D
```

```
69   E
70   F
71   G
72   H
73   I
74   J
75   K
76   L
77   M
78   N
79   O
80   P
81   Q
82   R
83   S
84   T
85   U
86   V
87   W
88   X
89   Y
90   Z
```

Checking Lexical Order of Characters

The following functions determine the lexical sequence of characters:

Function	Description
lle*char, char*	Compares whether the first character is lexically less than or equal to the second
lge*char, char*	Compares whether the first character is lexically greater than or equal to the second
lgt*char, char*	Compares whether the first character is lexically greater than the second
llt*char, char*	Compares whether the first character is lexically less than the second

Example 4

The following function demonstrates the use:

```
program testingChars
implicit none

   character:: a, b, c
   a = 'A'
   b = 'a'
   c = 'B'

   if(lgt(a,b)) then
      print *, 'A is lexically greater than a'
   else
      print *, 'a is lexically greater than A'
   end if

   if(lgt(a,c)) then
      print *, 'A is lexically greater than B'
   else
      print *, 'B is lexically greater than A'
   end if

   if(llt(a,b)) then
      print *, 'A is lexically less than a'
   end if
   if(llt(a,c)) then
      print *, 'A is lexically less than B'
   end if

end program testingChars
```

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

```
a is lexically greater than A
B is lexically greater than A
A is lexically less than a
A is lexically less than B
```