Programación en FORTRAN

Nivel Básico - Sesión 2

Martin Josemaría Vuelta Rojas

3 de enero de 2018

SoftButterfly

Contenido

Variables y tipos de datos

Variables y tipos de datos

- Un lenguaje de programación permite identificar los datos que se manipulan y almacenan en grandes cantidades en un ordenador.

Variables y tipos de datos

- Un lenguaje de programación permite identificar los datos que se manipulan y almacenan en grandes cantidades en un ordenador.

Variables

Variables y tipos de datos

- Un lenguaje de programación permite identificar los datos que se manipulan y almacenan en grandes cantidades en un ordenador.

Variables

- Una variable es un objeto que representa un tipo de dato, suceptible de modificarse, nombrado por cadenas de caracteres.

Tipos de datos

Variables y tipos de datos

Tipos de datos

1. **character:** cadena de uno o varios caracteres.

Variables y tipos de datos

Tipos de datos

1. **character:** cadena de uno o varios caracteres.
2. **integer:** números enteros, positivos o negativos.

Variables y tipos de datos

Tipos de datos

1. **character:** cadena de uno o varios caracteres.
2. **integer:** números enteros, positivos o negativos.
3. **logical:** valores lógicos o booleanos, es decir, toman uno de los dos valores, .true. (verdadero) o .false. (falso).

Variables y tipos de datos

Tipos de datos

1. **character:** cadena de uno o varios caracteres.
2. **integer:** números enteros, positivos o negativos.
3. **logical:** valores lógicos o booleanos, es decir, toman uno de los dos valores, .true. (verdadero) o .false. (falso).
4. **real:** números reales, positivos o negativos.

Variables y tipos de datos

Tipos de datos

1. **character:** cadena de uno o varios caracteres.
2. **integer:** números enteros, positivos o negativos.
3. **logical:** valores lógicos o booleanos, es decir, toman uno de los dos valores, .true. (verdadero) o .false. (falso).
4. **real:** números reales, positivos o negativos.
5. **complex:** números complejos, compuestos de una parte real y otra imaginaria, ambas de tipo real.

Variables y constantes

Declaración de variables

- La declaración de una o más variables del mismo tipo está dada por la sintaxis

  <tipo> ,[<atributo(s)>] [::] <variable(s)> [=<valor>]

Variables y constantes

Declaración de variables

- La declaración de una o más variables del mismo tipo está dada por la sintaxis

  <tipo> ,[<atributo(s)>] [::] <variable(s)> [=<valor>]

- Algunos atributos son:
  parameter, save, pointer, target, allocatable, dimension, public, private, external, intrinsic, optional.

Variables y constantes

Declaración de variables

- La declaración de una o más variables del mismo tipo está dada por la sintaxis

  <tipo> ,[<atributo(s)>] [::] <variable(s)> [=<valor>]

- Algunos atributos son:
  parameter, save, pointer, target, allocatable, dimension, public, private, external, intrinsic, optional.

```
1   CHARACTER(len= 4), PARAMETER :: prompt = ">>> "
2   CHARACTER(len= *), PARAMETER :: message = "Ingresa tu primer nombre [máx 20 car]:"
```

Variables y constantes

Declaración de variables

- La declaración de una o más variables del mismo tipo está dada por la sintaxis

  <tipo> ,[<atributo(s)>] [::] <variable(s)> [=<valor>]

- Algunos atributos son:
  parameter, save, pointer, target, allocatable, dimension, public, private, external, intrinsic, optional.

```
1  CHARACTER(len= 4), PARAMETER :: prompt = ">>> "
2  CHARACTER(len= *), PARAMETER :: message = "Ingresa tu primer nombre [máx 20 car]:"
```

Véase strings.f95

Declaración de constantes

- Si se requiere que una variable que tome un valor definido no suceptible de cambio, se utiliza el atributo parameter.

```
1    CHARACTER, PARAMETER :: NewLine = CHAR(10)
```

Variables y constantes

Declaración de constantes

- Si se requiere que una variable que tome un valor definido no suceptible de cambio, se utiliza el atributo parameter.

```
1   CHARACTER, PARAMETER :: NewLine = CHAR(10)
```

Véase strings.f95

Declaración de constantes

- Si se requiere que una variable que tome un valor definido no suceptible de cambio, se utiliza el atributo parameter.

```
1   CHARACTER, PARAMETER :: NewLine = CHAR(10)
```

Véase strings.f95

- Las variables pueden ser definidas en función de constantes mediante el atributo parameter.

Cadenas de caracteres y valores lógicos

Declaración de cadenas de caracteres

Cadenas de caracteres y valores lógicos

Declaración de cadenas de caracteres

Cadenas de caracteres y valores lógicos

Declaración de cadenas de caracteres

Cadenas de caracteres y valores lógicos

Declaración de cadenas de caracteres Declaración de valores lógicos

- La declaración de una variable de tipo character está dada por la sintaxis
  CHARACTER[(len=<longitud>)],[<atributo(s)>][::]<variable(s)>[=<valo

```
1  CHARACTER(kind=ascii, len=26) :: Alphabet
2  CHARACTER(kind= ucs4, len=30) :: HelloWorld
```

Véase kind_character.f95

-

Cadenas de caracteres y valores lógicos

Declaración de cadenas de caracteres Declaración de valores lógicos

- La declaración de una variable de tipo character está dada por la sintaxis
  CHARACTER[(len=<longitud>)],[<atributo(s)>][::]<variable(s)>[=<valo

```
1   CHARACTER(kind=ascii, len=26) :: Alphabet
2   CHARACTER(kind= ucs4, len=30) :: HelloWorld
```

Véase kind_character.f95

-

- La declaración de una variable lógica está dada por
  LOGICAL <variable(s)>

Enteros

Tipos de enteros

- La representación de valores enteros se declara con INTEGER.

Enteros

Tipos de enteros

- La representación de valores enteros se declara con INTEGER.
- Los valores pueden ser guardados usualmente con presición simple, doble o cuádruple.

Enteros

Tipos de enteros

- La representación de valores enteros se declara con INTEGER.
- Los valores pueden ser guardados usualmente con presición simple, doble o cuádruple.

```
1  INTEGER, PARAMETER :: K02 = SELECTED_INT_KIND(2)
2  INTEGER, PARAMETER :: K04 = SELECTED_INT_KIND(4)
3  INTEGER, PARAMETER :: K08 = SELECTED_INT_KIND(8)
4  INTEGER, PARAMETER :: K16 = SELECTED_INT_KIND(16)
5
6  INTEGER(kind=K02) :: I02
7  INTEGER(kind=K04) :: I04
8  INTEGER(kind=K08) :: I08
9  INTEGER(kind=K16) :: I16
```

Enteros

Tipos de enteros

- La representación de valores enteros se declara con INTEGER.
- Los valores pueden ser guardados usualmente con presición simple, doble o cuádruple.

```
1  INTEGER, PARAMETER :: K02 = SELECTED_INT_KIND(2)
2  INTEGER, PARAMETER :: K04 = SELECTED_INT_KIND(4)
3  INTEGER, PARAMETER :: K08 = SELECTED_INT_KIND(8)
4  INTEGER, PARAMETER :: K16 = SELECTED_INT_KIND(16)
5
6  INTEGER(kind=K02) :: I02
7  INTEGER(kind=K04) :: I04
8  INTEGER(kind=K08) :: I08
9  INTEGER(kind=K16) :: I16
```

Véase kind_integers.f95

Reales

Tipos de reales

Reales

Tipos de reales

- La representación de número reales se declara con REAL y puede ser de presición estándar o simple presición (sp) y de presición superior, doble (dp) o cuádruple (qp) presició en adelante.

Reales

Tipos de reales

- La representación de número reales se declara con REAL y puede ser de presición estándar o simple presición (sp) y de presición superior, doble (dp) o cuádruple (qp) presició en adelante.
- La sintaxis para el tipo real es

real(kind=<np>)

Reales

Tipos de reales

- La representación de número reales se declara con REAL y puede ser de presición estándar o simple presición (sp) y de presición superior, doble (dp) o cuádruple (qp) presició en adelante.
- La sintaxis para el tipo real es

real(kind=<np>)

```
1   REAL(kind=p04) :: X04
2   REAL(kind=p08) :: X08
3   REAL(kind=p16) :: X16
4   REAL(kind=p32) :: X32
```

Reales

Tipos de reales

- La representación de número reales se declara con REAL y puede ser de presición estándar o simple presición (sp) y de presición superior, doble (dp) o cuádruple (qp) presició en adelante.
- La sintaxis para el tipo real es

real(kind=<np>)

```
1   REAL(kind=p04) :: X04
2   REAL(kind=p08) :: X08
3   REAL(kind=p16) :: X16
4   REAL(kind=p32) :: X32
```

Véase kind_real.f95

Reales

- La notación científica para los reales viene dada por los identificadores ”e” (sp), ”d” (dp) y ”q” (qp).

Reales

- La notación científica para los reales viene dada por los identificadores "e" (sp), "d" (dp) y "q" (qp).

```
1   REAL(kind=4)  :: x = 2.e0        !simple presición
2   REAL(kind=8)  :: y = 4.d-6       !doble presición
3   REAL(kind=16) :: z = -8.q-1000   !cuadruple presición
```

Reales

- La notación científica para los reales viene dada por los identificadores "e" (sp), "d" (dp) y "q" (qp).

```
1   REAL(kind=4)  :: x = 2.e0        !simple presición
2   REAL(kind=8)  :: y = 4.d-6       !doble presición
3   REAL(kind=16) :: z = -8.q-1000   !cuadruple presición
```

- Sin embargo, en muchos casos es útil predefinir la clase kind al cambiar de tipo de real variando el valor de np, como por ejemplo

Reales

- La notación científica para los reales viene dada por los identificadores "e" (sp), "d" (dp) y "q" (qp).

```
1   REAL(kind=4)  :: x = 2.e0        !simple presición
2   REAL(kind=8)  :: y = 4.d-6       !doble presición
3   REAL(kind=16) :: z = -8.q-1000   !cuadruple presición
```

- Sin embargo, en muchos casos es útil predefinir la clase kind al cambiar de tipo de real variando el valor de np, como por ejemplo

```
1   INTEGER, PARAMETER :: np=16          !np = 4,8 o 16
2   REAL (kind=np) :: X = 2.e-10_np      !e =  e, d o q
```

Tipos de complejos

Complejos

Tipos de complejos

- La representación de números complejos se declara con COMPLEX, e igualmente que los reales, puede presentar una presición simple, doble o cuádruple.

Complejos

Tipos de complejos

- La representación de números complejos se declara con COMPLEX, e igualmente que los reales, puede presentar una presición simple, doble o cuádruple.

```
1  INTEGER :: re = 25
2  REAL(kind= 4) :: im04 = 3.141592
3  REAL(kind= 8) :: im08 = 3.141592
4  REAL(kind=10) :: im10 = 3.141592
5  REAL(kind=16) :: im16 = 3.141592
6  COMPLEX(kind=16) :: z16 = (25, 3.141592)
```

Complejos

Tipos de complejos

- La representación de números complejos se declara con COMPLEX, e igualmente que los reales, puede presentar una presición simple, doble o cuádruple.

```
1  INTEGER :: re = 25
2  REAL(kind= 4) :: im04 = 3.141592
3  REAL(kind= 8) :: im08 = 3.141592
4  REAL(kind=10) :: im10 = 3.141592
5  REAL(kind=16) :: im16 = 3.141592
6  COMPLEX(kind=16) :: z16 = (25, 3.141592)
```

Véase complex.f95

Tipos de complejos

- La representación de números complejos se declara con COMPLEX, e igualmente que los reales, puede presentar una presición simple, doble o cuádruple.

```
1  INTEGER :: re = 25
2  REAL(kind= 4) :: im04 = 3.141592
3  REAL(kind= 8) :: im08 = 3.141592
4  REAL(kind=10) :: im10 = 3.141592
5  REAL(kind=16) :: im16 = 3.141592
6  COMPLEX(kind=16) :: z16 = (25, 3.141592)
```

Véase complex.f95

- La notación científica y las declaraciones empleando kind siguen las mismas reglas.

La cláusula Implicit

- Si existen variables que no han sido definidas, el tipo de variable depende de la letra inicial. Por lo cual

La cláusula Implicit

- Si existen variables que no han sido definidas, el tipo de variable depende de la letra inicial. Por lo cual
  - i, j, k, l, m, n representan variables enteras.

La cláusula Implicit

- Si existen variables que no han sido definidas, el tipo de variable depende de la letra inicial. Por lo cual
- i, j, k, l, m, n representan variables enteras.
- Las demás letras representan variables reales de presición simple.

La cláusula Implicit

- Si existen variables que no han sido definidas, el tipo de variable depende de la letra inicial. Por lo cual
  - i, j, k, l, m, n representan variables enteras.
  - Las demás letras representan variables reales de presición simple.
- El carácter implicito puede ser modificado empleando la instrucción IMPLICIT bajo la siguiente sintaxis:

La cláusula Implicit

- Si existen variables que no han sido definidas, el tipo de variable depende de la letra inicial. Por lo cual
- i, j, k, l, m, n representan variables enteras.
- Las demás letras representan variables reales de presición simple.
- El carácter implicito puede ser modificado empleando la instrucción IMPLICIT bajo la siguiente sintaxis:
- Debido a que el compilador puede reconocer variables por defecto, se recomienda emplear la instrucción IMPLICIT NONE, especificando todas las variables y evitando errores con el código fuente.

Operaciones elementales

Reglas generales

Operaciones elementales

- Las operaciones siguen el orden tradicional de las operaciones matemáticas, es decir, primero los términos entre paréntesis, exponentes, multiplicación y adición.

Operaciones elementales

- Las operaciones siguen el orden tradicional de las operaciones matemáticas, es decir, primero los términos entre paréntesis, exponentes, multiplicación y adición.
- El uso del símbolo $=$ en el lenguaje Fortran tiene el sentido de asignación mientras que en el uso matemático tiene sentido de igualdad.

Operaciones elementales

- Las operaciones siguen el orden tradicional de las operaciones matemáticas, es decir, primero los términos entre paréntesis, exponentes, multiplicación y adición.
- El uso del símbolo $=$ en el lenguaje Fortran tiene el sentido de asignación mientras que en el uso matemático tiene sentido de igualdad.
- La asignación de una variable tiene la sintaxis

$$<\text{variable}> = <\text{expresión}>$$

Operaciones aritméticas

- Estan permitidas las operaciones entre valores tipo INTEGER, REAL y COMPLEX.

Operaciones aritméticas

- Estan permitidas las operaciones entre valores tipo INTEGER, REAL y COMPLEX.
- Las operaciones están dadas por adición (+), sustracción (-), multiplicación (*), división (/) y potenciación (**).

Operaciones aritméticas

- Estan permitidas las operaciones entre valores tipo INTEGER, REAL y COMPLEX.
- Las operaciones están dadas por adición (+), sustracción (-), multiplicación (*), división (/) y potenciación (**).
- Los operandos del mismo tipo y clase resultan en otro del mismo tipo y clase.

Operaciones aritméticas

Operaciones de tipo INTEGER

- Las operaciones de tipo INTEGER manejan números enteros dentro de un rango en $\mathbb{Z}$

Operaciones aritméticas

Operaciones de tipo INTEGER

- Las operaciones de tipo INTEGER manejan números enteros dentro de un rango en $\mathbb{Z}$
- La división se obtiene con un resto; es decir

$$\frac{x}{y} = z \Longleftrightarrow |x| = |z| \cdot |y| + \mathit{resto}$$

Operaciones aritméticas

Operaciones de tipo INTEGER

- Las operaciones de tipo INTEGER manejan números enteros dentro de un rango en $\mathbb{Z}$
- La división se obtiene con un resto; es decir

$$\frac{x}{y} = z \Longleftrightarrow |x| = |z| \cdot |y| + \textit{resto}$$

- La potenciación depende del tipo de variable del exponente.

$$x**n = \begin{cases} \underbrace{x * x * \cdots * x}_{n \text{ veces}} & si \quad n > 0 \\ \frac{1}{x**(-n)} & si \quad n < 0 \\ 1 & si \quad x = 0 \end{cases}$$

Operaciones aritméticas

Conversión de tipos

- Los enteros son convertidos en reales o complejos.

Operaciones aritméticas

Conversión de tipos

- Los enteros son convertidos en reales o complejos.
- Los reales son convertidos en complejos.

Operaciones aritméticas

Conversión de tipos

- Los enteros son convertidos en reales o complejos.
- Los reales son convertidos en complejos.
- Los reales o complejos son convertidos en la clase (kind) más alta.

Operaciones aritméticas

Conversión de tipos

- Los enteros son convertidos en reales o complejos.
- Los reales son convertidos en complejos.
- Los reales o complejos son convertidos en la clase (kind) más alta.
- Al asignar valores ($=$), la parte derecha se evalúa en el tipo y clase correspondiente, luego es convertida al del tipo y clase de la variable al lado izquierdo.

Operaciones aritméticas

Conversión de tipos

- Los enteros son convertidos en reales o complejos.
- Los reales son convertidos en complejos.
- Los reales o complejos son convertidos en la clase (kind) más alta.
- Al asignar valores (=), la parte derecha se evalúa en el tipo y clase correspondiente, luego es convertida al del tipo y clase de la variable al lado izquierdo.

  Por ejemplo

```
1  INTEGER           :: n, m
2  REAL              :: a, b
3  REAL(kind=8)      :: x, y
4  COMPLEX           :: c
5  COMPLEX(kind=8)   :: z
6  :
7  a = (x*(n**c))/z
8  :
```

Operaciones aritméticas

Conversiones de tipo más significativas

Conversión	Mecanismo de Conversión
x = n	x = n
x = a	x = a
n = x	$n = \begin{cases} m \quad si \quad m \leq x \leq m+1 \quad y \quad x \geq 0 \\ -m \quad si \quad m \leq -x \leq m+1 \quad y \quad x < 0 \end{cases}$
a = x	a = round(x)
a = c	a = $\mathbb{R}$(z)
z = x	z = (x, 0)

Operaciones de comparación

- La operación de comparación se expresa de la forma

  <expresión_1> <operador> <expresión_2>

Operaciones de comparación

- La operación de comparación se expresa de la forma

  <expresión_1> <operador> <expresión_2>

- Para variables de tipo COMPLEX solo son válidos los operadores == y /= .

Operaciones de comparación

- La operación de comparación se expresa de la forma

  <expresión_1> <operador> <expresión_2>

- Para variables de tipo COMPLEX solo son válidos los operadores == y /= .
- Los operadores de comparación son variables de tipo LOGICAL.

Operaciones de comparación

- La operación de comparación se expresa de la forma

  <expresión_1> <operador> <expresión_2>

- Para variables de tipo COMPLEX solo son válidos los operadores == y /= .
- Los operadores de comparación son variables de tipo LOGICAL.

Fortran 90	Fortran 77	Significado
==	.eq.	es igual a
/=	.ne.	no es igual a
>	.gt.	es estrictamente mayor a
>=	.ge.	es mayor o igual a
<	.lt.	es estrictamente menor a
<=	.le.	es menor o igual a

Operadores de comparación

Operaciones lógicas

- La operación lógica se expresa de la forma

  <expresión_1> <operador> <expresión_2>

Operaciones lógicas

- La operación lógica se expresa de la forma

  <expresión_1> <operador> <expresión_2>

- Las operaciones lógicas se evaluan luego de las operaciones de comparación, de izquierda a derecha.

Operaciones lógicas

- La operación lógica se expresa de la forma

  <expresión_1> <operador> <expresión_2>

- Las operaciones lógicas se evaluan luego de las operaciones de comparación, de izquierda a derecha.

Operador	Significado
.not.	No
.and.	y
.or.	o
.eqv.	equivalente
.neqv.	no equivalente

Operadores lógicos

Funciones intrínsecas *built in functions*

- Funciones predefinidas independientes del compilador.

Funciones intrínsecas *built in functions*

- Funciones predefinidas independientes del compilador.
- Una función intrínseca monoargumental se expresa de la forma

<función>(<argumento>)

Funciones intrínsecas *built in functions*

- Funciones predefinidas independientes del compilador.
- Una función intrínseca monoargumental se expresa de la forma

  <función>(<argumento>)

- Las funciones intrínsecas pueden no tener argumento o tener varios.

Funciones intrínsecas *built in functions*

- Funciones predefinidas independientes del compilador.
- Una función intrínseca monoargumental se expresa de la forma

  <función>(<argumento>)

- Las funciones intrínsecas pueden no tener argumento o tener varios.
- Las funciones biargumentales más destacadas son la función cmplx y la función mod.

Funciones intrínsecas *built in functions*

- Funciones predefinidas independientes del compilador.
- Una función intrínseca monoargumental se expresa de la forma

  <función>(<argumento>)

- Las funciones intrínsecas pueden no tener argumento o tener varios.
- Las funciones biargumentales más destacadas son la función cmplx y la función mod.
  - cmplx asigna un valor complejo a partir de valores reales.

```
1  !Sea z = (x,y) una variable compleja
2  cmplx (<expres_1>, <expres_2>) !expres son de tipo REAL
```

Funciones intrínsecas *built in functions*

- Funciones predefinidas independientes del compilador.
- Una función intrínseca monoargumental se expresa de la forma

  <función>(<argumento>)

- Las funciones intrínsecas pueden no tener argumento o tener varios.
- Las funciones biargumentales más destacadas son la función cmplx y la función mod.

  - cmplx asigna un valor complejo a partir de valores reales.

```
1  !Sea z = (x,y) una variable compleja
2  cmplx (<expres_1>, <expres_2>) !expres son de tipo REAL
```

  - mod es el valor que corresponde al resto de una división, es decir

$$n = n/m + mod(n, m)$$

```
1  !Sea n de tipo INTEGER o REAL y m del mismo tipo y distinto de 0
2  mod (<expres_1>, <expres_2>) !expres son de tipo INTEGER o REAL
```

Funciones intrínsecas *built in functions*

Función	Argumento	Resultado	Descripción
abs	real, complex, integer	real, integer	$\|x\|$, $\|z\|$, $\|n\|$
sqrt	real, complex	real, complex	$(\sqrt{x},\ x \geq 0)$, $(\sqrt{x},\ z \in \mathbb{C})$
int	real	integer	Parte entera de un real x
fraccion	real	real	Parte fraccional de un real x
real	complex	real	$\mathbb{R}e(z),\ z \in \mathbb{C}$
aimag	complex	real	$\mathbb{I}m(z),\ z \in \mathbb{C}$
conjg	complex	complex	$\bar{z},\ z \in \mathbb{C}$
cos	real complex	real complex	$(\cos x, x \in \mathbb{R})$, $(\cos z, z \in \mathbb{C})$
sin	real complex	real complex	$(\sin x, x \in \mathbb{R})$, $(\cos z, z \in \mathbb{C})$
tan	real complex	real complex	$(\tan x, x \in \mathbb{R})$, $(\tan z, z \in \mathbb{C})$
acos	real complex	real complex	$(\arccos x, x \in \mathbb{R})$, $(\arccos z, z \in \mathbb{C})$
asin	real complex	real complex	$(\arcsin x, x \in \mathbb{R})$, $(\arcsin z, z \in \mathbb{C})$
atan	real complex	real complex	$(\arctan x, x \in \mathbb{R})$, $(\arctan z, z \in \mathbb{C})$
exp	real complex	real complex	$(\exp x, x \in \mathbb{R})$, $(\exp z, z \in \mathbb{C})$
log	real complex	real complex	$(\log x, x > 0)$, $(\log z, z \in \mathbb{C}, z \neq 0)$
log10	real	real	$(\log_{10} x, x > 0$

Funciones intrínsecas más relavantes

Operaciones con caracteres

Operaciones con caracteres

Operaciones con caracteres

Operaciones con caracteres

Operadores binarios

- La asignación del tipo CHARACTER es de la forma

  $$<\text{variable}> = <\text{expresión}>$$

  donde <variable> y <expresión> son de tipo CHARACTER y pueden tener longitud $len = n$ o $len = m$ respectivamente; $n, m \in \mathbb{Z}^+$

- Si $n \leq m$, se asigna a la <variable> los n primeros caracteres de la <expresión> de izquierda a derecha, eliminando la diferencia $m - n$.
- Si $n > m$, se asigna a la <variable> de izquierda a derecha la cadena de caracteres de <expresión>, completando los últimos $n - m$ caracteres de la derecha con espacios.

Operaciones con caracteres

Operadores binarios

- La asignación del tipo CHARACTER es de la forma

  $$<\text{variable}> = <\text{expresión}>$$

  donde <variable> y <expresión> son de tipo CHARACTER y pueden tener longitud $len = n$ o $len = m$ respectivamente; $n, m \in \mathbb{Z}^+$

  - Si $n \leq m$, se asigna a la <variable> los n primeros caracteres de la <expresión> de izquierda a derecha, eliminando la diferencia $m - n$.
  - Si $n > m$, se asigna a la <variable> de izquierda a derecha la cadena de caracteres de <expresión>, completando los últimos $n - m$ caracteres de la derecha con espacios.

- asd

Operaciones con caracteres

Operadores binarios

- La asignación del tipo CHARACTER es de la forma

  $$<\text{variable}> = <\text{expresión}>$$

  donde <variable> y <expresión> son de tipo CHARACTER y pueden tener longitud $len = n$ o $len = m$ respectivamente; $n, m \in \mathbb{Z}^+$

  - Si $n \leq m$, se asigna a la <variable> los n primeros caracteres de la <expresión> de izquierda a derecha, eliminando la diferencia $m - n$.
  - Si $n > m$, se asigna a la <variable> de izquierda a derecha la cadena de caracteres de <expresión>, completando los últimos $n - m$ caracteres de la derecha con espacios.
- asd
- asd