

Programadoras da NASA no programa espacial de viaxe á lúa (1969)

Mary Jackson: enxeñeira da NASA



Katherine Johnson: **matemática** e programadora da NASA
Primeira civil en recibir a **Medalla de Ouro** do Congreso USA



Dorothy Vaughan: programadora en **Fortran** do primeiro ordenador da NASA
Programación estruturada en Fortran



Película
“Figuras ocultas”

Operadores relacionais (I)

- Definen as relacións de equivalencia (igualdade) e de orde numérica (con caracteres, a orde alfabética)
- Operadores: `<` `<=` `>` `>=` `==` `/=`
- Exemplos: `x < 3`, `y >= z`
- Operandos numéricos, resultado lóxico
- `x < y`: *.true.* se `x < y`, *.false.* en caso contrario
- `x > y`: *.true.* se `x > y`, *.false.* en caso contrario
- `x == y`: *.true.* se `x` e `y` son iguais, *.false.* en caso contrario
- `x /= y`: *.true.* se `x` e `y` son distintos, *.false.* en caso contrario

Operadores relacionais (II)

- Permiten definir condicións de igualdade / desigualdade ou maior/menor sobre números ou caracteres (orde alfabética)
- Úsanse en sentenzas de selección para crear distintas “rutas” de execución do programa, dependendo do cumprimento ou non dunha condición

- Sobre vectores e matrices, aplícanse por compoñentes:

```
integer :: x(3)=(/1,2,3/)
print *,x>2 ==> F F T
```

```
integer :: a(2,2)
forall(i=1:2,j=1:2) a(i,j)=i*i+j
print *,mod(a,2)==0
```

- Función *count*: nº de elementos dun vector/matriz que cumpren unha relación: *count(x>5)*, *count(x/=0)*, *count(mod(x,2)==1)*

- Caracteres: *character(10) :: s='ola'*

```
s=='ola': T          s>'pepe': F
```

```
integer :: a(2,2)
forall(i=1:2,j=1:2) a(i,j)=i*i+j
print *,count(mod(a,2)==1)
```

Operadores lógicos (I)

- Combinan unha ou dúas condicións (definidas cos operadores relacionais) mediante relacións de:
 - **Conxunción:** *condición1* e tamén *condición2*
 - **Disxunción:** *condición1* ou *condición2*
 - **Negación:** non é certo *condición1*
 - **Equivalencia lóxica:** *condición1* e *condición2* son iguais (ou ben ambas se cumpren ou ben ningunha se cumpre)
 - **Non equivalencia lóxica:** *condición1* e *condición2* son distintas (se se cumpre unha non se cumpre a outra e ao revés)

Operadores lógicos (II)

- Operadores:

- Conxunción: *.and.*
- Disxunción: *.or.*
- Negación : *.not.*
- Equivalencia lóxica (condicións iguais): *.eqv.*
- Non equivalencia lóxica (condicións distintas): *.neqv.*

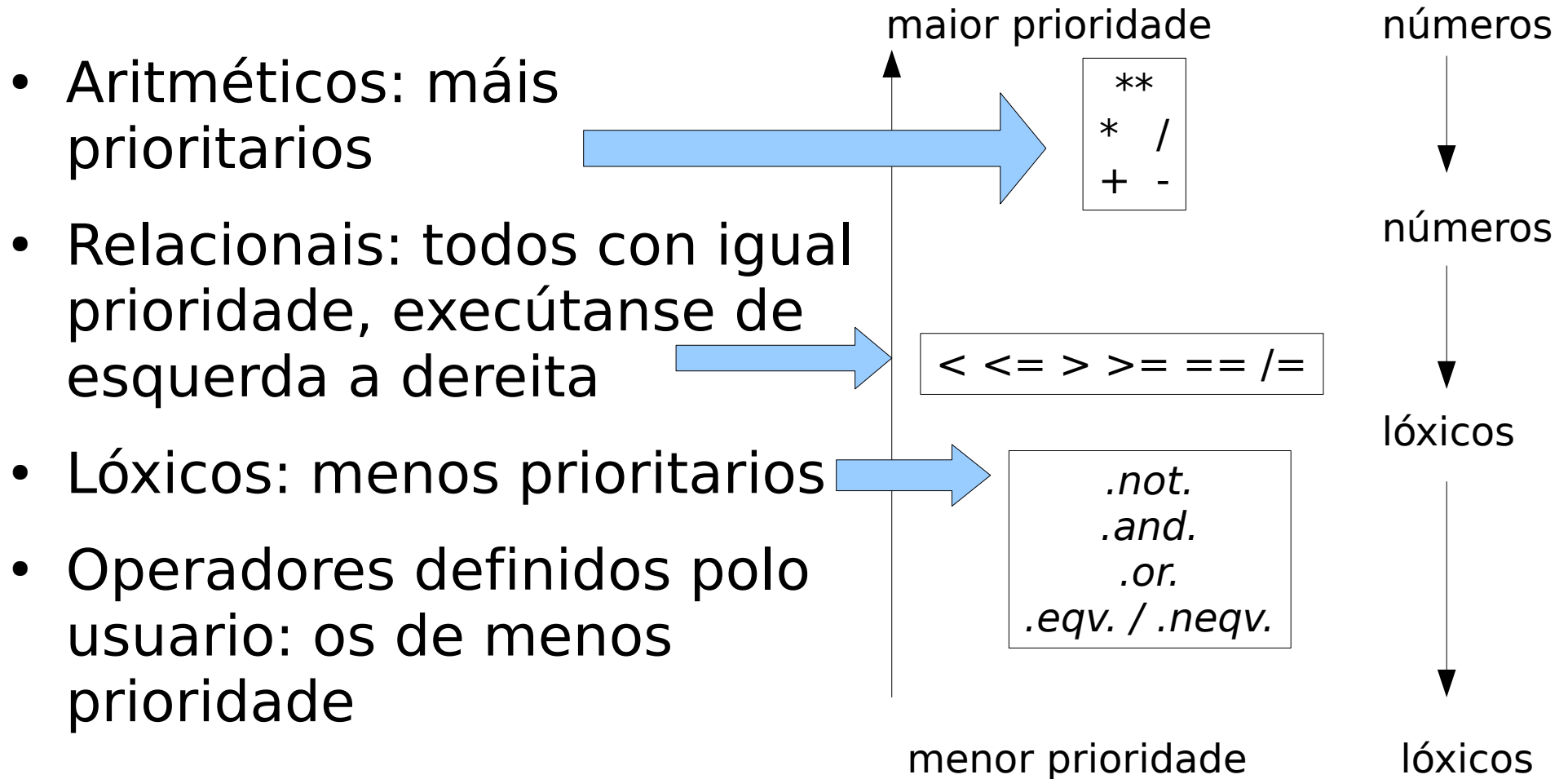
<i>a</i>	<i>.true.</i>	<i>.false.</i>	<i>.true.</i>	<i>.false.</i>
<i>b</i>	<i>.true.</i>	<i>.true.</i>	<i>.false.</i>	<i>.false.</i>
<i>a.and.b</i>	<i>.true.</i>	<i>.false.</i>	<i>.false.</i>	<i>.false.</i>
<i>a.or.b</i>	<i>.true.</i>	<i>.true.</i>	<i>.true.</i>	<i>.false.</i>
<i>.not.a</i>	<i>.false.</i>	<i>.true.</i>		
<i>a.eqv.b</i>	<i>.true.</i>	<i>.false.</i>	<i>.false.</i>	<i>.true.</i>
<i>a.neqv.b</i>	<i>.false.</i>	<i>.true.</i>	<i>.true.</i>	<i>.false.</i>

- Operandos lógicos (resultado de operadores relacionais ou lógicos). Resultado lóxico

Exemplos de operadores lóxicos

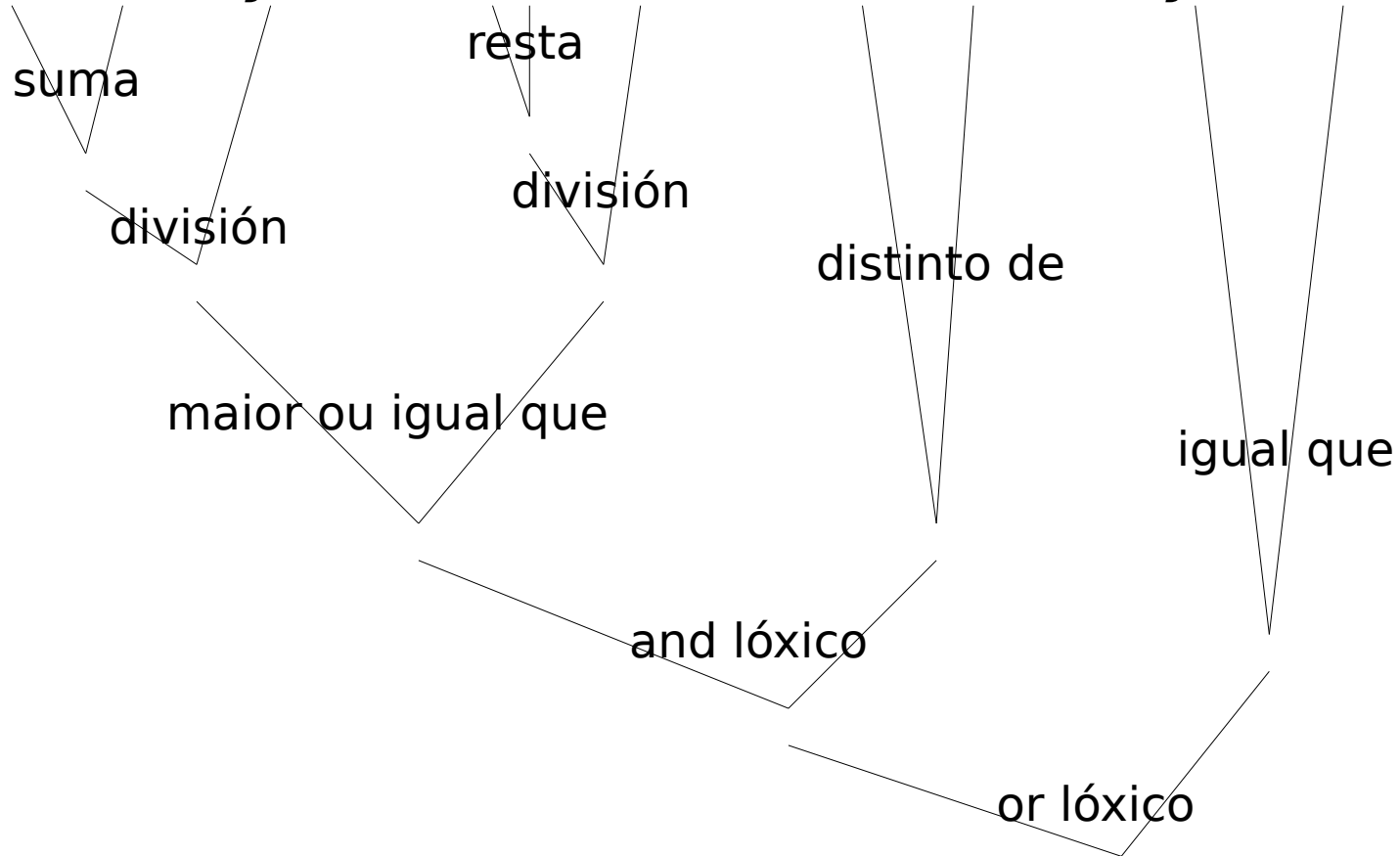
- AND: $x > 5 .and. x \leq 7$: x debe estar en $(5, 7]$ para que se cumpra a condición
- OR: $x == 2 .or. x == 3$: x debe ser 2 ou 3 para que a condición sexa `.true.`
- NOT: $.not.(x > 3)$ é certa so se $x \leq 3$
- EQV: $mod(x, 2) == 0 .eqv. mod(x, 3) == 0$: é certa se x é múltiplo de 2 e de 3, e tamén se x non é múltiplo de 2 nin de 3
- NEQV: $mod(x, 2) == 0 .neqv. mod(x, 3) == 0$: é certa se x é múltiplo de 2 pero non de 3, e tamén se x non é múltiplo de 2 pero si de 3

Precedencia de operadores



Exemplo de prioridades de execução

$(z+3.4)/y \geq (z-t)/x.\text{and.}(x \neq 0).\text{or.}(y==z)$



Funcións lóxicas *all* e *any* para vectores/matrices

- *all(v/=0)*: *true*. se tódolos elementos do vector *v* son non nulos.
- *all(a>0)*: *true* se tódolos elementos da matriz *a* son positivos
- *all(a,1)*: executa o *all* por columnas (*a* debe ser *logical*)
- *all(a,2)*: executa o *all* por filas
- *any(a>5)*: *true* se *a* ten algún elemento maior que 5
- *any(mod(a,2)==0,1)*: *true* para columnas con elemento par.
- *any(a==3,2)*: *true* para as filas cun 3

```
integer :: a(2,3)=reshape((/1,-1,3,1,5,-2/),shape(a))
```

```
print *,all(a>0) → False
```

```
print *,all(a>0,1) → False True False
```

```
print *,any(a==3,2) → True False
```

```
1 3 5  
-1 1 -2
```