

Ada Lovelace (1815-1852)



- Inglaterra, século XIX (1815-1852)
- Inventora do primeiro programa de ordenador
- Precursora en máis de 100 anos da informática
- Linguaxe de programación de sistemas de tempo real chamado Ada na súa honra

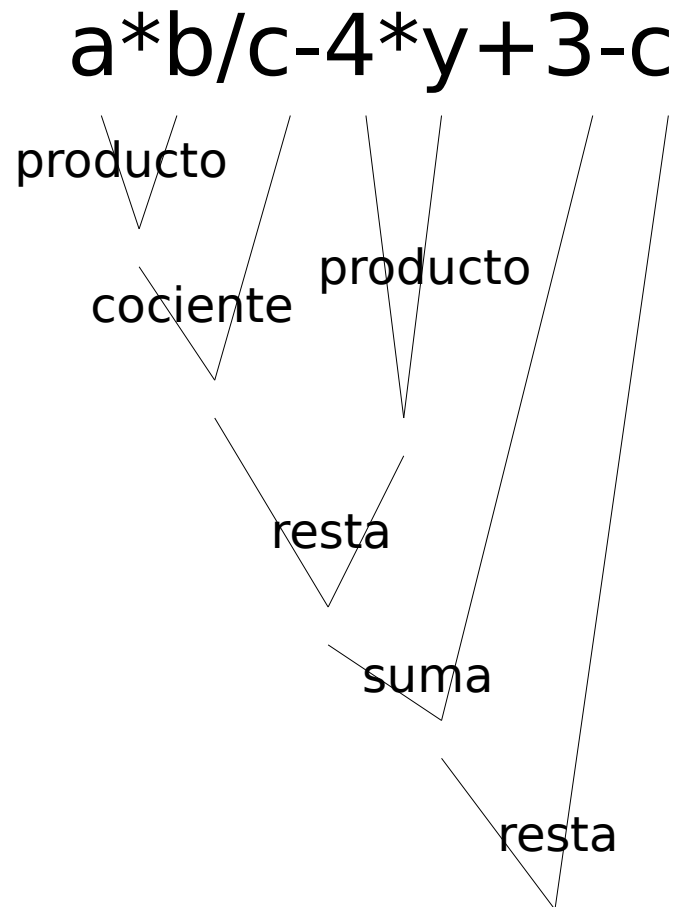
Expresións aritméticas

- Operadores aritméticos: + - * / ** (exponenciación): $x + 3$, $y - z$, $4.3E-2*x$, x/y , $x**2$
- Son binarios (2 operandos). O operador - pode ser unario: *print **, *-x*
- Precedencia (prioridade):
 - 1) **
 - 2) * /
 - 3) - unario
 - 4) + -
- Operadores de igual prioridade: execútanse de esquerda a dereita, agás a exponenciación (**), que se executa de dereita a esquerda

Prioridades na exponenciación

- Exponenciación: $x^{y^z^t} \rightarrow x^{**}y^{**}z^{**}t$
- Execútase de dereita a esquerda:
 - 1) $z^{**}t$
 - 2) y elevado ao resultado de (1)
 - 3) x elevado ao resultado de (2)
 - Poderíamos escribir: $x^{**}(y^{**}(z^{**}t))$, pero non é necesario poñer parénteses porque as prioridades xa fan que se execute nesa orde

Prioridades cos restantes operadores



As operacións combinan constantes e variábeis

Uso de parénteses

- As prioridades pódense modificar usando parénteses en función das nosas necesidades

- Exemplo: $\frac{\frac{x+y}{z-t}+1}{x+\frac{1}{y+z}} \Rightarrow ((x+y)/(z-t)+1)/(x+1/(y+z))$



- Se o puxéramos sen parénteses: Distinto de

$$x + y/z - t + 1/x + 1/y + z \Rightarrow x + \frac{y}{z} - t + \frac{1}{x} + \frac{1}{y} + z$$

- O nº de parénteses de apertura-peche debe coincidir. Se non coinciden, erro de compilación

Tipo (do resultado) dunha expresión

- Operación aritmética: dous operandos que poden ser de tipos distintos (p.ex., real e enteiro)
- Cando un operador actúa sobre datos de distintos tipos, o resultado é do tipo máis alto de ambos (*double* é máis alto que *real*, e *real* é máis alto que *integer*)
- Isto non impide a posíbel perda de información, dependendo dos seus valores.
- Exemplo: na división de enteiros, xa que o resultado será enteiro: perda de parte decimal, se existe: sexan $n=3, m=2$ enteiros: n/m debería ser 1.5, pero como é enteiro, será 1
- Solución: declarar n e/ou m como reais; ou ben usa-la función *real(n)* para converter n (ou m) en *real*

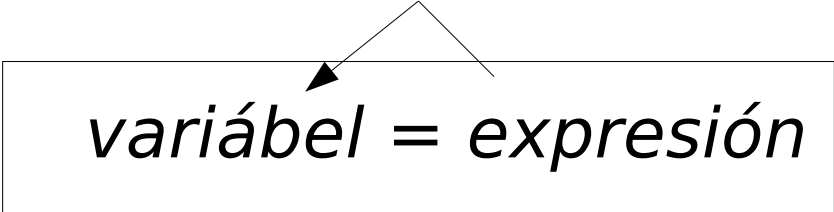
Tipos das expresións

- Sexan $x=1.2, y=1.0$ reais, $n=3, m=2$ enteiros. Na expresión:

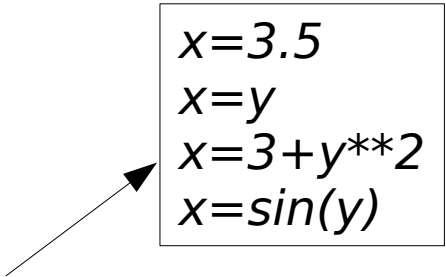
$$x*y + n/m$$

- Erro común: pensar que na división n/m o resultado é real porque x e y son reais.
- Pero como n e m son enteiros, n/m é enteiro. Debería ser 1.5, pero vai ser 1 (enteiro): perda de información
- Hai que ver o tipo do resultado de cada operador individual (de dous operandos)

Sentenza de asignación



variável = expresión



$x=3.5$
 $x=y$
 $x=3+y^{**}2$
 $x=\sin(y)$

- A expresión da dereita pode ser unha constante, variábel, expresión aritmética ou función intrínseca
- Asigna o resultado da **expresión dereita** á **variábel esquerda**
- No lado esquerdo só pode haber unha variábel: non pode haber constante, expresión nin chamada a subprograma: erro de compilación
- Ambos lados poden ser de tipos distintos

Sentenza de asignación

- O valor da dereita convértese ao tipo da esquerda: pode haber perda de información. Exemplo:

$$n = x + 3.4 \quad (n \text{ enteiro, } x \text{ real})$$

A variábel da esquerda da asignación debe ter un tipo non **inferior** ao da expresión da dereita

- Erro común: dúas asignacións consecutivas á mesma variábel. Ex:  *integer < real < double*

$$x = 3.4 - y$$

$$x = z - y \text{ !pérdese o valor anterior de } x$$

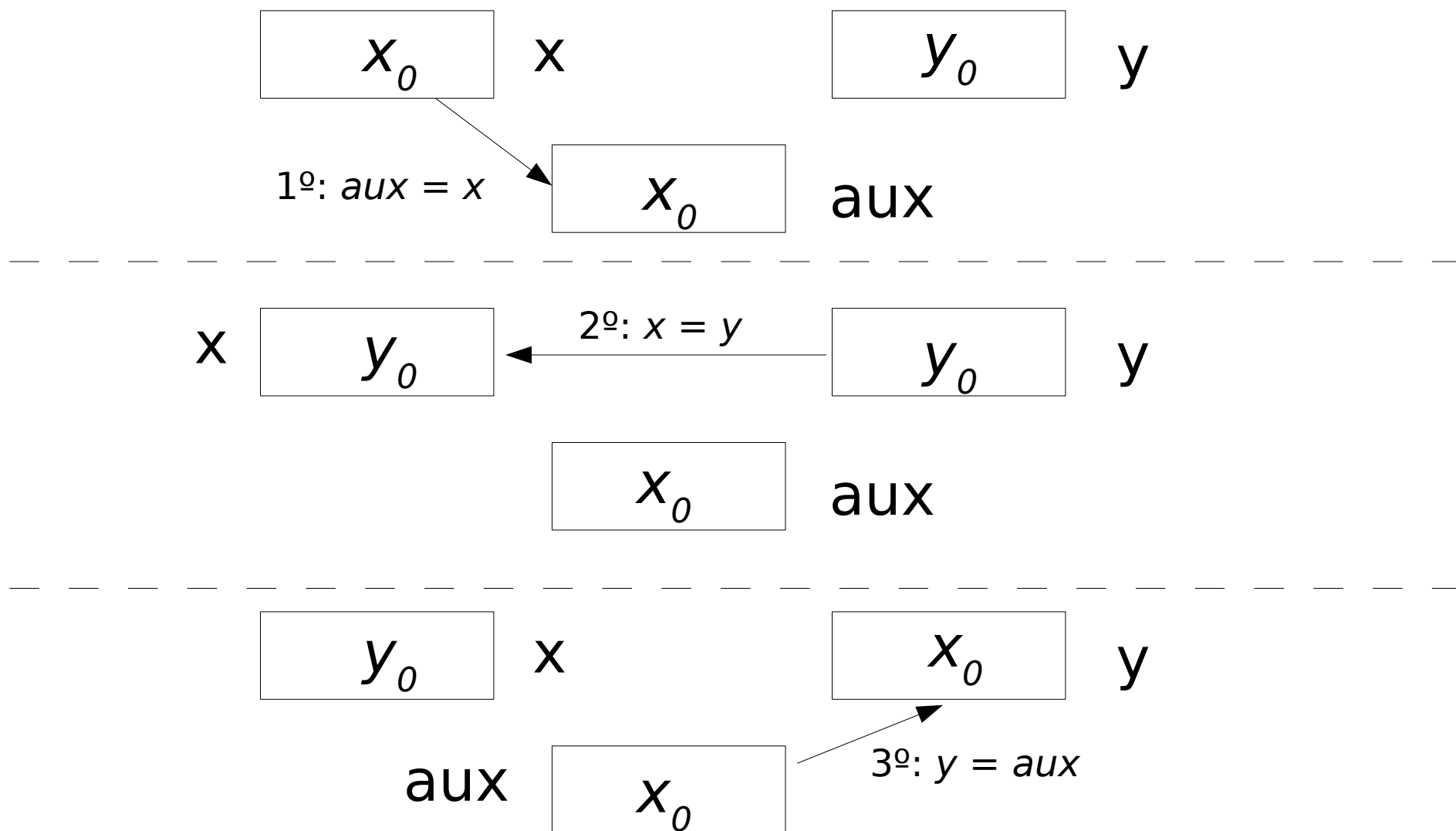
- Pódense encadear varias asignacións na mesma liña con “;”. Ex: $x = y + z; t = 2 * x$

Exemplo: intercambio dos valores de dúas variábeis

- Temos dúas variábeis x e y : x ten o valor x_0 , e y ten o valor y_0 : queremos intercambiar os seus valores
- Distinguir entre o valor que se almacena nunha variábel e a variábel (que é o contedor do valor)
- Necesitamos unha variábel para almacenar temporalmente un dos dous valores: aux
- 1º paso: $aux = x$ (almacena x_0 en aux)
- 2º paso: $x = y$ (copia y_0 de y á variábel x)
- 3º paso: $y = aux$ (copia x_0 , que está almacenado na variábel aux , á variábel y)

Ollo: escribe na
mesma variábel
que salvou
antes en aux

Intercambio de 2 variáveis



Intercambio de 2 variábeis

- Resumindo:

$aux = x$

$x = y$ ←

$y = aux$

Sempre escribe nunha
variábel que se salvou
na sentenza anterior
a outra variábel

- No intercambio, a variábel na dereita dunha sentenza de asignación debe aparecer na esquerda da seguinte sentenza de asignación
- Non pode estar a mesma variábel na esquerda de dúas sentenzas consecutivas de asignación

Algunhas funcións intrínsecas útiles de Fortran

- Valor absoluto: *abs(x)*. Argumento real/dobre.
- Raíz cadrada: *sqrt(x)*, *csqrt(z)* para números complexos
- Resto da división enteira: *mod(m,n)* resto de *m/n*
- Trigonómicas: *sin, cos, tan, asin, acos, atan*
- Hiperbólicas: *sinh, cosh, tanh, asinh, acosh, atanh*
- Exponencial/logarítmica: *exp(x), log(x)*
- Números aleatorios ($0 < x < 1$): *call random_number(x), x=rand()*
- Arrai aleatorio: *real a(3,3); call random_number(a)*
- Truncamento a enteiro: *int, floor, ceiling, nint*
- Executar comando de Windows: *call system('comando')*. Ex: *call system('dir')*: mostra o contido do directorio actual

Funcións de sentenza

- Se queres definir unha función matemática, p.ex. $f(x,y)=x*y$ que se poda calcular nunha única sentenza, podemos definir e chamar a unha **función de sentenza** (ou **de liña**) como a calquera función intrínseca
- Así non tes que poñer a expresión varias veces
- Hai que definila antes de ser chamada
- Os valores que se pasan como argumentos deben ter o mesmo tipo que o definido implícitamente polos argumentos da función de liña.
- Permite definir unha función como se fose intrínseca
- Só se pode chamar dende o programa principal no que se define.
- Verémolas de novo no tema de subprogramas

```
program exemplo  
  f(x,y)=x*y  
  print *,f(3.,2.5)  
  stop  
end program exemplo
```