PROGRAMACIÓN ESTRUCTURADA EN FORTRAN

As mulleres na programación e na informática

Proceso de programación en Fortran

- Escribe o programa fonte *programa.f90* no **entorno** *Visual Code*
- Compilación do programa dende a **terminal** do VSCode:

gfortran programa.f90

- Corrección dos erros de compilación
- Creación de programa executábel: *a.exe*
- Execución de programa na terminal: .*a.exe*
- Corrección de erros de execución e lóxicos (edición, compilación, execución)
- Se executas *gfortran programa.f90 -o programa.exe*, o executable chamarase *programa.exe*
- **Coidado**: non poñas *programa.f90* no canto de *programa.exe*: se o fas, sobrescribes o *programa.f90* co executable

Programa básico en Fortran

- Resumo do proceso: *programa.f90* Compilador *gfortran* *a.exe*

```
program proba
print *,'ola!'
end program proba
```

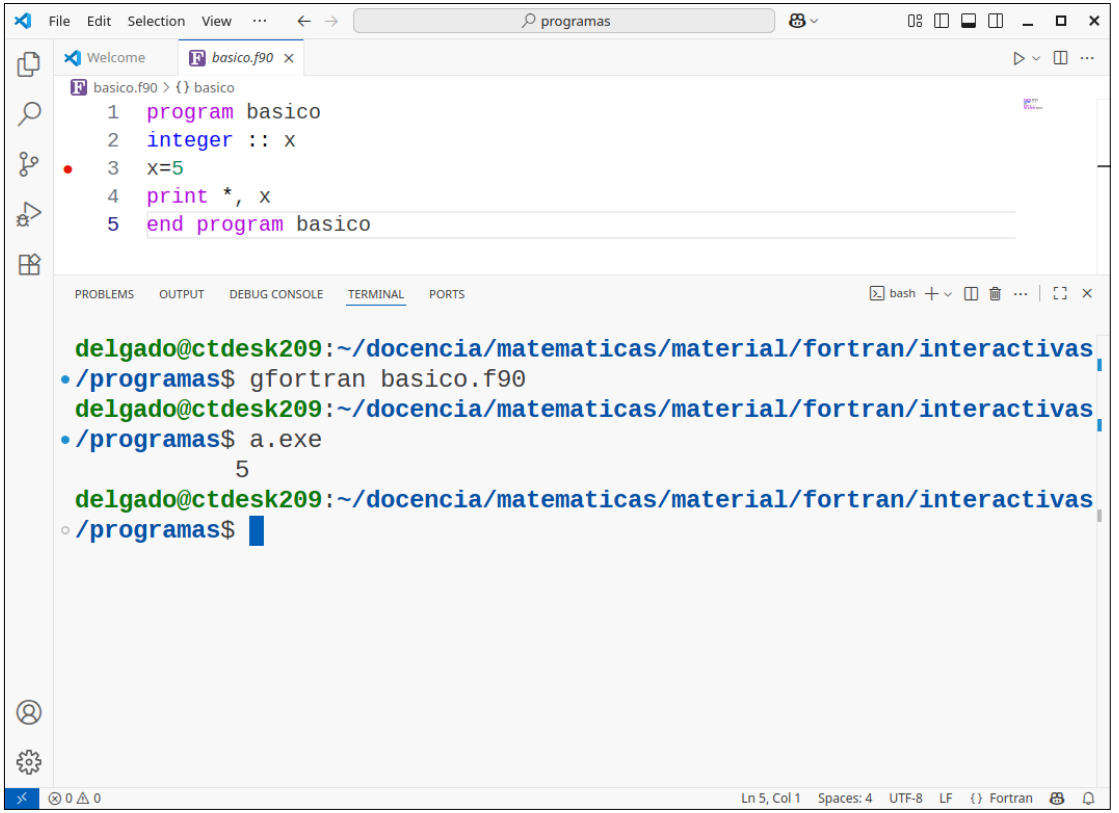
- Comeza con sentenza: "*program nome*"
- Remata con "*end program nome*"
- "*nome*" é o nome do programa: non pode haber variábeis nin subprogramas con ese nome
- Logo de *program*:
  - declaración de variábeis (ao principio)
  - sentenzas executábeis: operacións, entrada / saída, ...

Programa básico en Fortran

- Sentenza “*stop*”: remata a execución:
- *stop ‘mensaxe’*: remata e imprime a mensaxe
- Exemplo de programa básico:

```
program basico
integer :: x
x=5
print *, x
end program basico
```

Imprime o nº 5 na terminal de comandos



- Liñas de comentarios: con ! ao comezo da liña

Entrada e saída estándar

- Entrada de datos estándar: le datos dende o teclado e almacénaos en variábeis
  - Sentenza *read*: *read *, x*
  - Pódese producir un erro se a variábel é numérica e introducimos un carácter (p.ex.)
- Saída de datos estándar: visualiza na terminal (pantalla)
  - Sentenza *print*: *print *, 'resultado=', 2*x*
  - Con formato: *print '("n=",i0)',n*
    print '("x=",f10.6)',x

Ancho do nº enteiro

Ancho e nº de decimais

Algoritmo e diagrama de fluxo

Algoritmo: método para a resolución dun problema, detallado completamente en tódolos seus pasos.

- **Entrada** de datos: dende teclado ou arquivos
- **Procesamento**: operacións cos datos
- **Saída** de resultados: por pantalla ou a arquivos

Lectura de datos por teclado ou escritura de datos na terminal:

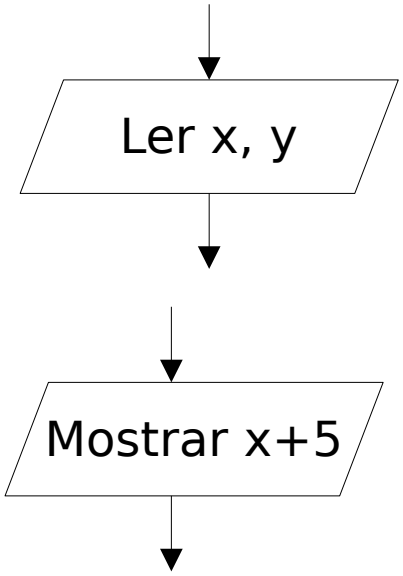
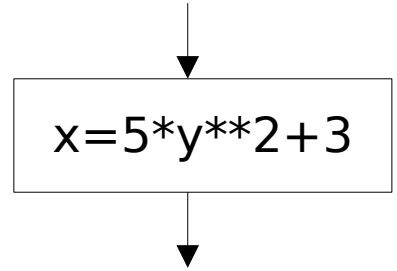


Diagrama de fluxo:

- Forma de representar gráficamente un algoritmo como unha secuencia de operacións
- Diagrama de caixas (operacións) e flechas que as conectan (orden de execución).

Operación aritmética ou asignación:



Estrutura de selección básica

- IF/ELSE: avalía unha serie de condicións e executa unhas sentenzas ou outras dependendo de que condicións se cumpren

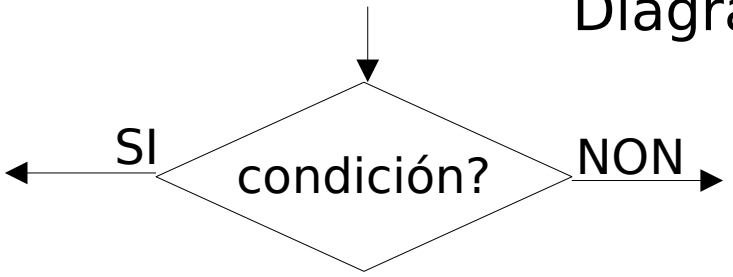
Sintaxe:

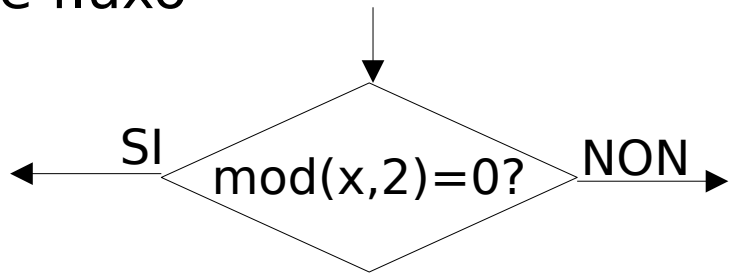
```
if(condicion) then
    sentenzas1
else if(condición2) then
    sentenzas2
else
    sentenzas3
endif
```

Exemplo:

```
if (x > 3) then
    print *, 'alto'
else
    print *, 'baixo'
end if
```

Diagrama de fluxo





Estrutura iterativa *do* definida

- Permite repetir unha ou varias sentenzas un certo número de veces

Sintaxe:

```
do var=ini, fin, paso
    sentenzas
end do
```

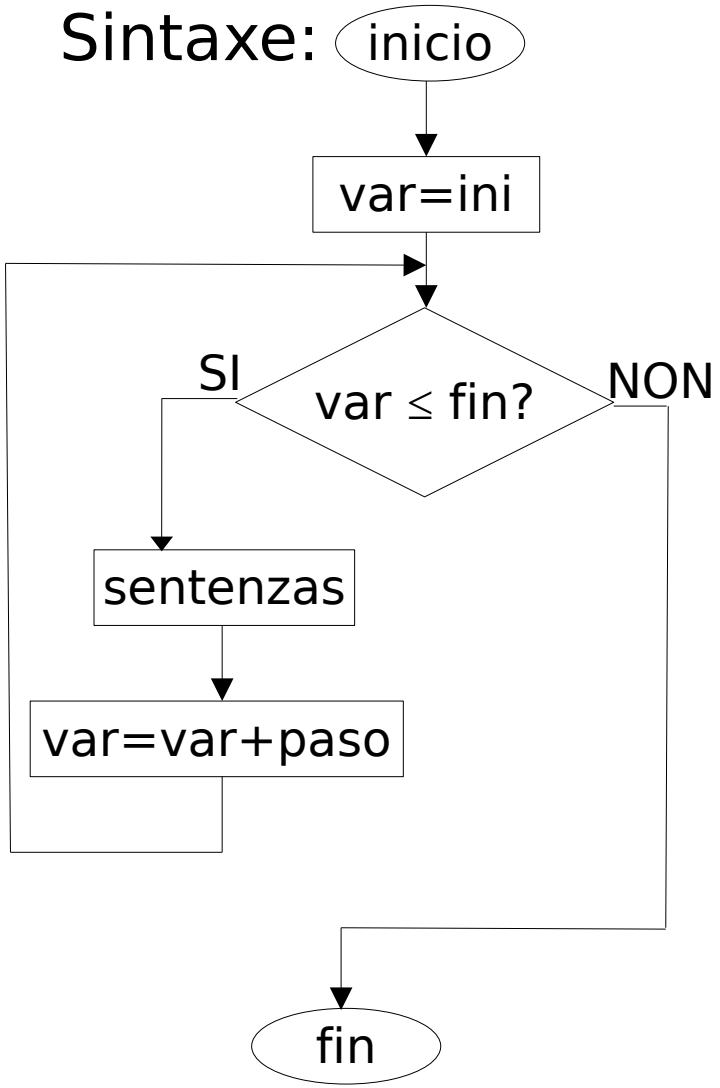
Exemplo:

```
do i=1, 10
    print *,i+2
end do
```

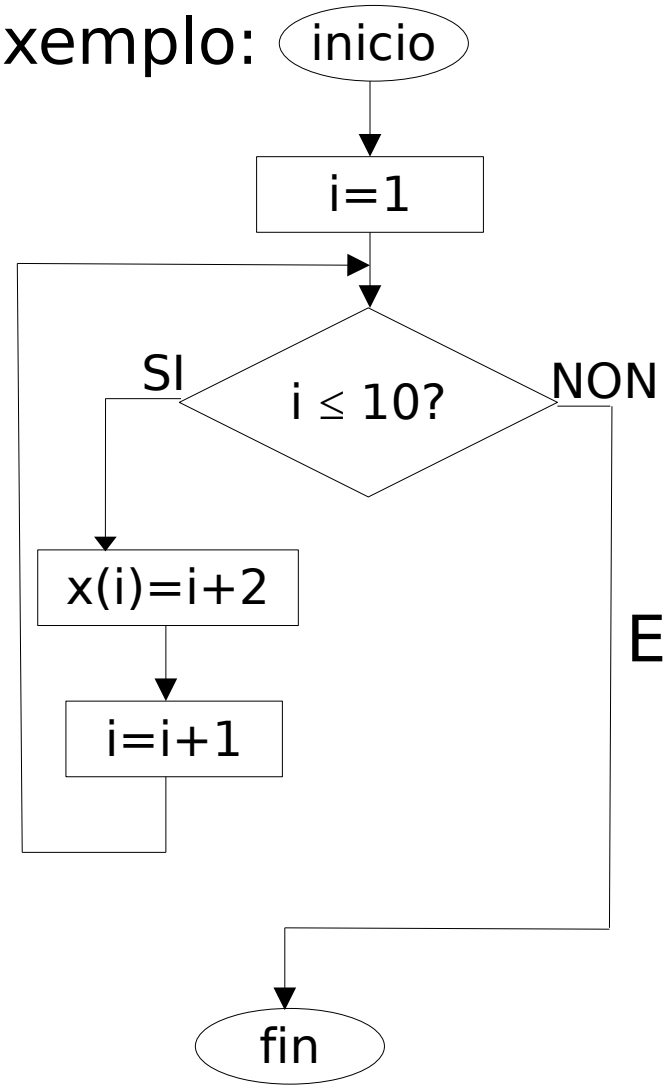
- Repite as sentenzas dende que a variábel *var* adopta o valor *ini*, ata que adopta o valor *fin*.
- En cada repetición, *var* increméntase en *paso*
- O incremento *paso* vale 1 por defecto

Diagrama de fluxo *do* definido

Sintaxe:

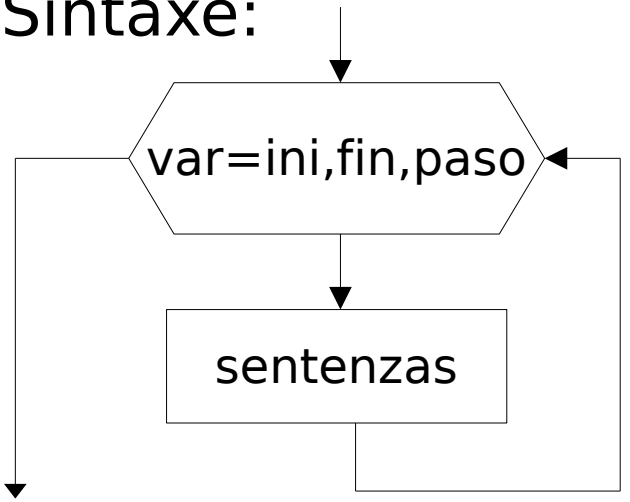


Exemplo:

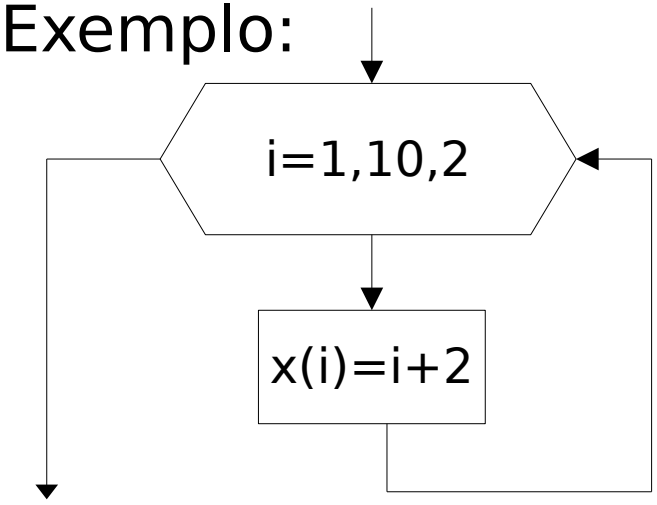


Versión simplificada:

Sintaxe:



Exemplo:



Acumulador

- Variable que aparece na esquerda e na dereita dunha asignación dentro dun bucle *do*: Exemplo: *x=x+i*; *p=p*j*
- Emprégase en operacións cun número variábel de operandos, que require un bucle *do*.
- O acumulador debe inicializarse co elemento neutro da operación.
- Acumulador de suma e de producto dun número variábel (*n*) de valores:

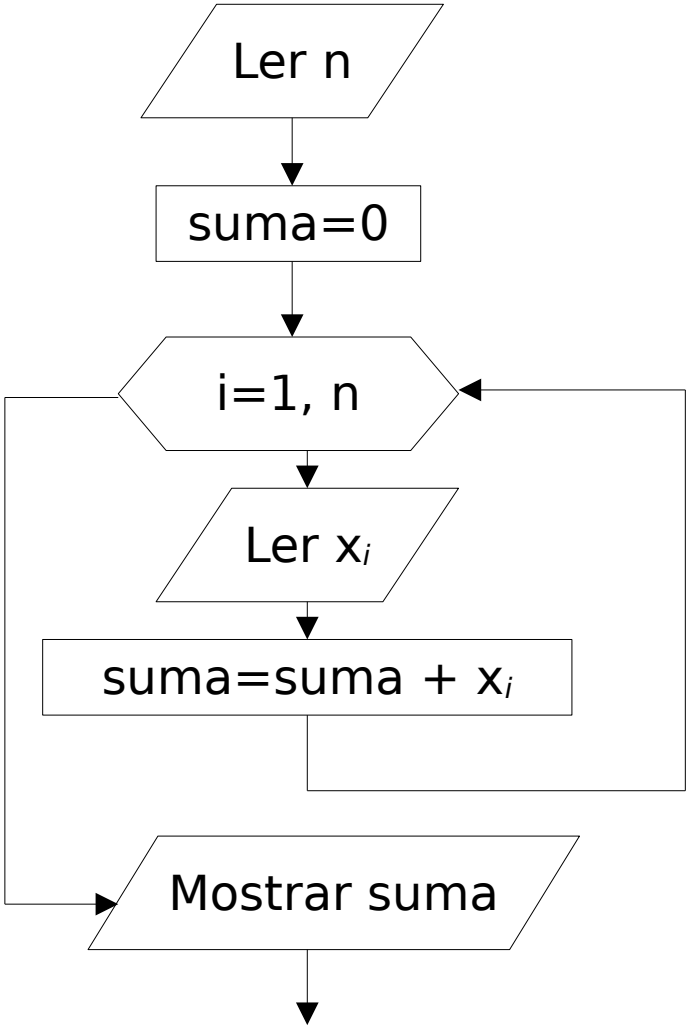
$$\sum_{i=1}^{n} x_i$$

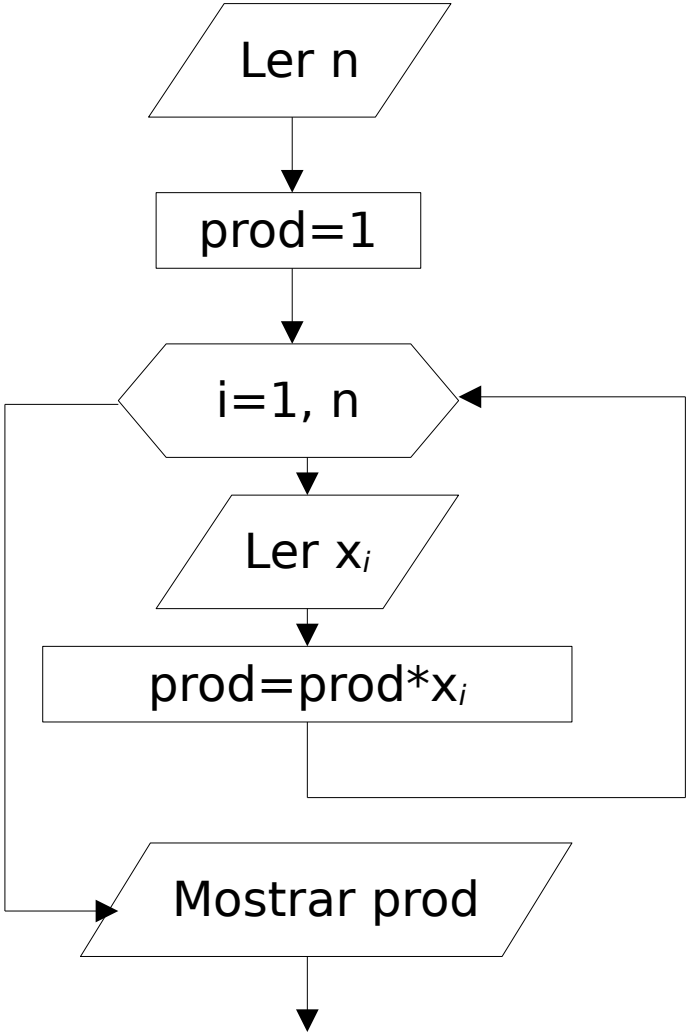
```
suma=0
do i=1,n
    print *,'x? '
    read *,x
    suma=suma+x
end do
```

$$\prod_{i=1}^{n} x_i$$

```
prod=1
do i=1,n
    print *,'x? '
    read *,x
    prod=prod*x
end do
```

Diagrama de fluxo de suma e producto





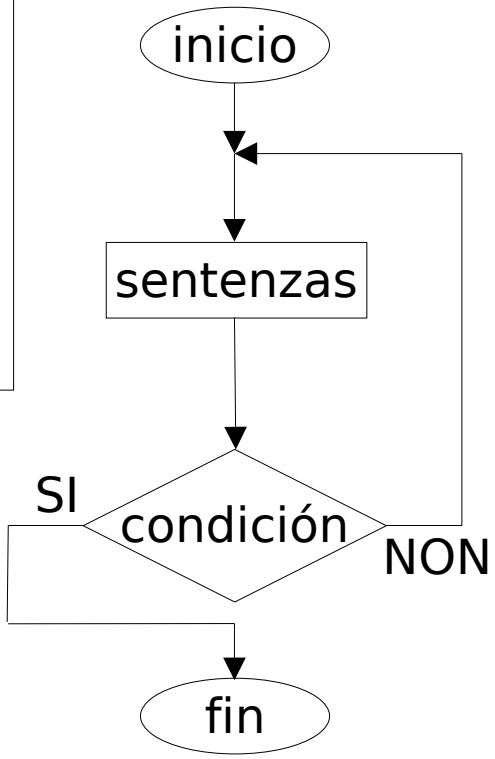
Estruturas iterativas *do* indefinidas

- Bucle *do/exit*: repite as sentenzas e remata cando se cumpre unha condición.

```
do
   sentenzas
   if(cond) exit
   sentenzas
end do
```

```
x=0;dx=0.1
do
   x=x+dx
   print *,x,x*x
   if(x>1) exit
end do
```

```
x=0;dx=0.1
do
   if(x>1) exit
   x=x+dx
   print *,x,sin(x)
end do
```



- Bucle *do while*: repite mentres se cumpre a condición.

```
do while(condición)
   sentenzas
end do
```

```
x=0;dx=0.1
do while(x<1)
   print *,x,x*x
   x=x+dx
end do
```