Evelyn Berezin (1925-2018)

- Inventora (1957) do primeiro ordenador de oficina, gardaba libros e contas e automatizaba un sistema bancario nacional
- Creadora (1971) do primeiro software procesador de texto
- Desenvolvedora (1962) do primeiro sistema software de reservas de pasaxeiros (60 cidades) para a liña aérea United Airlines, o sistema informático máis grande construído ata entón

Formatos de E/S (I)

- Os datos (enteiros, reais, carácter, ...) pódense ler / escribir (teclado/pantalla ou arquivos) de distintas formas (ancho, nº de decimais, modo exponencial ou non, etc.)
- Sentenza *format*:

  print 2, x, y, z   ! escritura en pantalla con formato 2
  read 2, x, y, z   ! lectura por teclado co formato 2
  print '(códigos de formato)', x, y, z
  read '(códigos de formato)', x, y, z
  2 format(códigos de formato)

- Os códigos de formato indican como se le / escribe cada dato, e débense axustar en nº aos datos (constantes / variábeis / expresións) a ler ou escribir.

Formatos de E/S (II)

- Ancho de campo: nº de caracteres máximo que pode ocupar un dato

Códigos de formato

- **Enteiros**: código I: **3i5**: 3 enteiros con 5 caracteres Ex: *print '(i2)', n; read '(i5)', m*

  O mellor: *print '(i0)',n*: escribe co ancho necesario.

- **Reais sen expoñente**: código F: **2f6.1**: 2 reais sen expoñente con 6 cifras e 1 decimal

- **Reais con expoñente**: código E: **4e10.3**: 4 reais con expoñente, con 10 cifras (incluíndo signo, parte enteira e fraccionaria, E do expoñente, signo e expoñente) e 3 decimais

Formatos de E/S (III)

- **Reais de dobre precisión**: código D: **2d10.4**: 2 datos de dobre precisión, igual que con código F
- **Caracteres**: código A: **2a10**: 2 cadeas de caracteres, cada unha con 10 caracteres. Se pos o código **a** sen ancho nun *print*, escribe a cadea co seu ancho
- **Espazos en branco**: código X: **5x**: 5 espazos en branco
- **Supresión de nova liña**: código $: *print* '*(“n? ”,$)*'
- **Tabuladores**: código T: **t10**: tabulador que chega até a columna 10-1=9

print 1, x, y, n, a
1 format(f6.2,e10.1,i7,a20)

1 real de ancho 6 e 2 decimais
1 real con expoñente, ancho 10 e 1 decimal
1 enteiro de ancho 7
1 carácter de ancho 20

Apertura de arquivos: *open* (I)

Sentenza *open*: permite abrir un arquivo e asocialo a unha unidade (representada por un nº enteiro):

open(1,file='datos.dat')

open(1,file='datos.dat',status='old'|'new',err=1)

- *status='old'*: o arquivo xa debe existir (p.ex. para ler): se non existe, *open* da erro para evitar ler dun arquivo baleiro
- *status='new'*: o arquivo non debe existir (p.ex. para crealo e escribir nel): se xa existe, *open* da erro para evitar sobrescribilo

Apertura de arquivos: *open* (II)

- En caso de erro, salta á sentenza con etiqueta 1:

```
open(1, file='datos.dat', status='new', err=1)
...
stop
1 print *,'erro en open abrindo datos.dat'
end program
```

- Neste exemplo, a unidade é a nº 1. Hai unidades reservadas (non se poden usar):
  - Erro estándar (terminal): 0
  - Entrada estándar (teclado): 5, *
  - Saída estándar (terminal): 6, *

Lectura de arquivos: *read* (I)

- Sentenza *close*: desvincula arquivo e unidade:

close(1)

- Lectura dende un arquivo: *read*

```
read (1, *) x, y, z
read (1, fmt=1,end=2) a, b, c
1 format(3f5.2)
read (1,'3f5.2',end=2) a,b,c
...
2 close(1)
```

moi importante: variábeis nas que se almacenan os datos lidos (hai que saber previamente a forma do arquivo): fácil de esquecer

- Le unha liña do arquivo e almacena os datos lidos nas variábeis indicadas
- Unidade (1), formato: * (por defecto), etiqueta ou códigos de formato

Lectura de arquivos: *read* (II)

- Cada *read* faise nunha liña distinta do arquivo
- Opción *end=2*: salta á sentenza con etiqueta 2 cando atopa o final do arquivo
- Exemplo: le un arquivo até que atopa o final:

```
character(100) :: a
open(1, file='datos.dat', status='old', err=1)
do
   read (1, *, end=2) a
end do
2 close(1)
stop
1 print *,'erro lendo de datos.dat'
```

Exemplo: lectura dun arquivo dato a dato

```
program lectura_arquivo_dato_a_dato
open(1,file='arquivo_dato_a_dato.dat',status='old',err=1)
do
    read (1,'(i2,$)',end=2) n
    print '(i0," ",$)',n
end do
2 close(1)
print *,''
stop
1 stop 'arquivo_dato_a_dato non existe'
end program lectura_arquivo_dato_a_dato
```

Lectura dun enteiro sen pasar á seguinte liña

Remate do bucle ao chegar á fin de arquivo

arquivo_dato_a_dato.dat

```
1 2 3 4 5 6 7 8 9
8 7 6
```

Uso de *read* para extraer variábeis de cadeas de texto formatadas

- Supón que tes unha cadea de texto na cal hai valores numéricos (enteiros/reais) e caracteres: ‘x= 5.32 n=512 s=ola caracola’
- Para extraer os valores 5.32, 512 e ‘ola caracola’ a variábeis real, enteira e carácter debes usar a sentenza *read*:

```
character(100) :: s='x= 5.32 n=512 s=ola caracola'
character(50) :: t,u,v,w
print *,s
read (s,'(a3,f4.2,a3,i3,a3,a)') u,x,v,n,w,t
print *,x,n,t
```

Extrae os valores 5.32. 512 e ‘ola caracola’ ás variábeis *x*,*n* e *t*

Os a3 correspóndense con ‘*x*= ’, ‘*n*= ’ e ‘*s*= ‘

O *read* ten un carácter (*s*) no canto dunha unidade de arquivo

Escritura de arquivos: *write*

write (1,) x, y, z*

- Escribe no arquivo da unidade 1 as variábeis indicadas co formato indicado (neste caso, por defecto, *)
- Para escribir na saída estándar (terminal), poñer unidade=5 ou *; ou *print *, x, y, z*
- Pódese engadir a opción *err=n* (*n*=etiqueta de sentenza á que salta se hai erro na escritura)
- Posíbeis erros: tentar escribir en arquivos nos que non hai permiso de escritura, ou en arquivos de directorios que non existen

Exemplos de formatos de E/S

```
real :: x = -3.2, y = -1.1234
integer :: n = -10, m = 19
write (1, fmt=1) x, n, y, m
1 format(e10.3, t15, i4, f8.2, 6x, i8)
```

```
-0.320E+01    -10   -1.12          19
```

Saída do *write*

```
write(1,fmt=2) n, m, x
2 format(2(i5,4x), e8.1)
```

```
 -10      19   -0.3E+01
```

Saída do *write*

```
write(1,fmt=3) x, y
3 format(2f3.1)
```

non hai ancho de campo dabondo para os datos

```
***  ***
```

Saída do *write*

Exemplo: creación e escritura dun arquivo

```
open(1,file='datos.dat',status='new',err=1)
do i = 1, 10
    write (1, *) i
end do
close(1)
stop
1 print *,'erro en open abrindo datos.dat'
```

- Se non lle poñemos *status='new'*, non nos daría erro en caso de xa existi-lo arquivo. Neste caso, sobrescribiría o arquivo, se este existe, **perdéndose**. Isto é perigoso

Uso de *write* para crear cadeas de texto formatadas a partir de variábeis

- A sentenza *write* tamén permite escribir en cadeas de texto, non so en unidades de arquivo.
- Supoñámos que tes as variábeis *x* (real), *n* (enteira) e *s* (cadea de caracteres).
- Para crear unha cadea de caracteres *t* da forma '*x=X n=N s=S*', onde X, N e S son os valores das variábeis *x* (con 5 cifras decimais), *n* (con 7 cifras) e *s*, tes que facer o seguinte:

```
character(20) :: s='ola caracola'
character(100) :: t
x=3.141592;n=1830
!t debe ter lonxitude dabondo para todo (x,n,s)
write (t,'(a,f9.5,a,i7,a,a)') 'x=',x,' n=',n,' s=',s
print *,t
```

Escribe en *t* a cadea formatada 'x=*X* n=*N* s=*S*'

Escritura de arquivos: pregunta antes de sobrescribir

```
program exemplo
character(5) :: s
open(1,file='datos.txt',status='new',err=1)
2 write(1,*) 'ola'
close(1)
stop
1 print '("datos.txt existe: sobrescribir?(s/n) ",$)';read *,s
if(s/='n') then
   open(1,file='datos.txt');goto 2
end if
stop
end program exemplo
```

Vai a sentenza con etiqueta 2

Escritura ao final dun arquivo (I)

- Tes que empregar a subrutina:

  fseek(unidade, desprazamento, onde)

- Move o cursor polo arquivo, en función do argumento *onde*: *onde=0* (comezo do arquivo), *1* (posición actual no arquivo), *2* (final do arquivo);
- Argumento *desprazamento*: nº de bytes (ou caracteres) logo do comezo / actual / final.
- Exemplos:

  call fseek(1,0,0): sitúa o cursor ao comezo do arquivo

  call fseek(1,10,1): sitúa o cursor 10 bytes logo da posición actual

  call fseek(1,-20,2): sitúa o cursor 20 bytes antes do final do arquivo

Escritura ao final dun arquivo (II)

```
open(1, file='datos.dat', status='old', err=1)
call fseek(1,0,2)  ! sitúase ao final do arquivo
write (1, *) x, y, z  ! engade ao final do arquivo
close(1)
stop
1 print *, "erro en open abrindo datos.dat"
```

- Función *ftell*: retorna a posición (en nº de bytes ou caracteres dende o comezo do arquivo) do cursor:

pos = ftell(unidade)

- Ex: *print *,'pos=', ftell(1)*
- Función *rewind(unidade)*: vai ao comezo do arquivo

Función que le un vector columna dende un arquivo

Hai que reservar memoria para o vector na función e definir a interface axeitada:

```
program proba
interface
    function le_vector(nf) result(x)
        character(len=100),intent(in) :: nf
        real,allocatable :: x(:)
    end function
end interface
character(len=100) :: nf
real,allocatable :: x(:)
print '(a,$)','nf? '
read *,nf  ! usa 'vector.dat'
x=le_vector(nf)
print *,'x=',x
deallocate(x)
stop
end program proba
```

```
function le_vector(nf) result(x)
character(len=100),intent(in) :: nf
real,allocatable :: x(:)
open(1,file=nf,status='old',err=1);n=0
do
    read (1,*,end=2);n=n+1
    print *,n
end do
2 allocate(x(n))
rewind(1)
do i=1,n
    read (1,*) x(i)
end do
return
1 print *,'open: arquivo',nf,'non existe'
stop
end function le_vector
```

Le arquivo e conta datos

Reserva memoria

Le datos e almacena no vector

Arquivo vector.dat:

```
1
2
3
4
5
```

Subrutina que escribe un vector nun arquivo

```
subroutine escribe_vector(v,n)
real, intent(in) :: v(:)
integer, intent(in) :: n
open(1, file = “vector.dat”, status=’new’)
write (1, *) (v(i), i = 1, n)
close(1)
end subroutine escribe_vector
```

```
real, dimension(3) :: v=(/1,2,3/)
call escribe_vector(v,3)
...
```

Inicialízase o vector no programa principal

Función que le unha **matriz cadrada** dende un arquivo

```
1 2 3
4 5 6
7 8 9
```

Arquivo matriz_cadrada.dat

```
program principal
interface
    function le_matriz(f) result(b)
        integer,dimension(:,:),allocatable :: b
        character(100),intent(in) :: f
    end function le_matriz
end interface
integer,dimension(:,:),allocatable :: a
character(100) :: f='matriz_cadrada.dat'
a=le_matriz(f)
print *,'a='
n=size(a,1)
do i=1,n
    print *,(a(i,j),j=1,n)
end do
stop
end program principal
```

```
function le_matriz(f) result(a)
character(100),intent(in) :: f
integer,dimension(:,:),allocatable :: a
character(100) :: s
open(1,file=f,status='old',err=1)
n=0
do
    read (1,*,end=2)
    n=n+1
end do
2 allocate(a(n,n))
rewind(1)
do i=1,n
    read (1,*) (a(i,j),j=1,n)
end do
close(1)
return
1 print *,'erro: arquivo ',f,'non existe'
stop
end function le_matriz
```

Le arquivo e conta liñas

Reserva memoria

Le datos e almacena en matriz

Programa que le unha matriz **non cadrada** dende un arquivo

```
1 2 3
4 5 6
```

Arquivo matriz.dat

```
program proba
character(100) :: s
integer,allocatable :: a(:)
open(1,file='matriz.dat',status='old',err=1)
read (1,'(a)') s
nf=1;nc=1
do i=1,len_trim(s)
   if(s(i,i)==' ') nc=nc+1
end do
do
   read (1,*,end=2)
   nf=nf+1
end do
2 rewind(1)
allocate(a(nf,nc))
do i=1,nf
   read (1,*) (a(i,j),j=1,nc)
end do
print *,'a='
do i=1,nf
   do j=1,nc
      print '(i5,$)',a(i,j)
   end do
   print *,''
end do
close(1);deallocate(a)
stop
1 print *,'erro: matriz.dat non existe'
end program proba
```

Conta as columnas da matriz

Conta as filas da matriz

Le a matriz

Mostra a matriz por pantalla

O nº de columnas é o nº de espazos da columna máis 1

len_trim(*s*): lonxitude de *s*

Subrutina que escribe unha matriz nun arquivo

```
program principal
real, dimension(3,4) :: a
forall(i=1:3,j=1:4) a(i,j)=i*j+j/2
call escribe_matriz(a,3,4)
end program principal
```

Inicialízase a matriz no programa principal

```
subroutine escribe_matriz(a,nf,nc)
real, intent(in) :: a(nf,nc),nf,nc
open(1, file='matriz.dat',status='new',err=1)
do i = 1, nf
    write (1, *) a(i, :)
end do
close(1)
return
1 print *,'erro escribindo matriz.dat'
stop
end subroutine escribe_matriz
```

Exemplo: lectura de arquivo en formato *write.table* de R

```
program le_arquivo
character(10),allocatable :: entrada(:),saida(:)  !vectores de strings
real,allocatable :: a(:,:)   !matriz cos datos
nf=2;ne=4
allocate(entrada(ne),a(nf,ne),saida(nf))
open(1,file='arquivo.dat',status='old',err=1)
read (1,*) (entrada(i),i=1,ne)  ! descarta a última cadea de caracteres ('Saida')
do i=1,nf
      read (1,*) j,(a(i,j),j=1,ne),saida(i)
end do
close(1)
print *,'a='
do i=1,nf
      print *,(a(i,j),j=1,ne)
end do
print *,'saida=',(saida(i),i=1,nf)
deallocate(entrada,a,saida)
stop
1 print *,'erro en open abrindo arquivo.dat'
end program le_arquivo
```

Pre-define o nº de filas e de columnas

arquivo.dat: columnas separadas por tabuladores

	E1	E2	E3	E4	Saida
1	0.25	0.33	1.23	-0.51	Branco
2	-0.34	1.3E5	0.22	4.3	Negro

Le arquivo en formato R sen pre-definir o nº de filas e columnas

```
program le_arquivo_formato_R
character(100) :: fich='arquivo.dat'
real,allocatable :: x(:,:)
character(100),allocatable :: y(:),none(:)
call conta_filas_columnas(fich,nf,nc)
allocate(x(nf,nc),y(nf),name(nc))
open(1,file=fich,status='old',err=1)
read (1,*) (nome(i),i=1,nc)
do i=1,nf
      read (1,*) t,(x(i,j),j=1,nc),y(i)
end do
close(1)
do i=1,nc
      print '(a10,$)',nome(i)
end do
print *,'saída'
do i=1,nf
      do j=1,nf
            print '(f10.2,$)',x(i,j)
      end do
      print *,y(i)
end do
deallocate(x,y)
stop
1 print *,fich,'non atopado'
end program le_arquivo_formato_R
```

Le datos do arquivo

Mostra datos por pantalla

```
subroutine conta_filas_columnas(fich,nf,nc)
character(100),intent(in) :: fich
integer,intent(out) :: nf,nc
character(100) :: s
open(1,file=fich,status='old',err=1)
read (1,'(a)') s
nf=0;nc=0
do i=2,len_trim(s)
      j=i-1;n=ichar(s(j:j));m=ichar(s(i:i))
      if(n/=69.and.m==69) nc=nc+1
end do
do
      read (1,*,end=2);nf=nf+1
end do
2 close(1)
return
1 print *,fich,'non atopado';stop
end subroutine conta_filas_columnas
```

Conta columnas na cabeceira

Conta as filas de datos

69=código do carácter tabulador