

Frances Allen (1932-2020)



- Pioneira na optimización de compiladores, optimización automática de código e programación paralela
- Creadora de linguaxes de programación e códigos de seguridade para a NSA
- Premio Turing de Informática en 2006

Operadores relacionais

- Operadores relacionais: $>$ (maior) $\geq$ (maior ou igual) $<$ (menor) $\leq$ (menor ou igual) $==$ (teste igualdade), $\sim =$ (teste desigualdade)
- Se comparamos escalares, o resultado é 1 (certo) se se cumpre o teste ou 0 (falso) se non se cumpre
- Se comparamos matrices (deben ser da mesma dimensión en filas e columnas), a comparación faise elemento a elemento
- O resultado é unha matriz de 0s (nos elementos onde falla o teste) e 1s (nos elementos onde se cumpre) coa mesma dimension cas orixinais
- Precedencia: todos teñen a mesma, e avalíanse de esquerda a dereita

Operadores lógicos

- NOT Lógico: $\sim$: P. ex: $\sim x$ da 1 se x é 0, e 0 se x é distinto de 0
- AND lógico: $\&$ (para vectores/matrices), $\&\&$ (para escalares)
- OR lógico: $|$ (para vectores/matrices), $||$ (para escalares)
- Operandos numéricos (0 é falso, $\neq 0$ é certo)
- Con escalares, dan 0 ou 1; con matrices, operan elemento a elemento e dan unha matriz da mesma orde
- Se actúan cun escalar e unha matriz, cada elemento da matriz opérase co escalar

Función lóxica *all*

- *all(x)*: retorna 1 se tódolos elementos do vector *x* son non nulos, 0 se algún elemento de *x* é nulo
- *all(a)*, *all(a,1)*: vector de lonxitude *size(a,2)*, ou sexa, nº de columnas de *a*, con valores 1 nas columnas de *a* con tódolos elementos non nulos e 0 nas restantes
- *all(a,2)*: vector de lonxitude *size(a,1)*, nº de filas de *a*, con valores 1 nas filas de *a* con tódolos elementos non nulos e 0 nas restantes
- *all(all(a))* ou *all(a(:))*: retorna un número, que é 1 se tódolos elementos de *a* son non nulos, ou 0 se *a* ten algún elemento nulo
- *all* pode aplicarse a unha expresión: *all(rem(a,2)==0)* vale 1 se tódolos elementos de *a* son pares

Funcións lóxicas *xor* e *any*

- *xor(a,b)*: retorna 1 se un dos operandos é 0 e o outro non, ou viceversa; con vectores/matrices opera elemento a elemento
- *any(x)*: retorna 1 se algún elemento do vector *x* é non nulo
- *any(a)*, *any(a,1)*: vector de lonxitude *size(a,2)* con valores 1 nas columnas de *a* con alomenos un elemento non nulo e 0 nas restantes
- *any(a,2)*: vector de lonxitude *size(a,1)* con valores 1 nas filas de *a* con alomenos un elemento non nulo e 0 nas restantes
- *any(any(a))* ou *any(a(:))*: actúa para toda a matriz *a*
- *xor* ou *any* poden aplicarse sobre expresións: *any(a>2)*

Función *find*

- *find(v)*: retorna os índices dos elementos non nulos do vector *v*
- *find(v>0 & v<5)*: índices dos elementos no intervalo $[0,5]$
- *v(v>0)* ou *v(find(v))* => elementos non nulos
- *[i,j]=find(rem(a,2)==0)*: índices de fila (*i*) e columna (*j*) dos elementos pares da matriz *a*
- *find(v>0,1,'first')*: índice do primeiro elemento positivo de vector *v*
- *find(isprime(v),3,'last')*: índices dos 3 últimos elementos primos de *v*

Sentenzas de selección (I)

As condicións elabóranse con operadores relacionais e lóxicos

- Sentenza IF:

```
if condición  
    sentenzas;  
end
```

- Cunha soa sentenza:
if condición; sentenza; end

- Sentenza IF-ELSE

```
if condición  
    sentenzas1;  
else  
    sentenzas2;  
end
```

- Sentenza IF-ELSE IF:

```
if condición1  
    sentenzas1;  
elseif condición2  
    sentenzas2;  
...  
else  
    sentenzasN;  
end
```

```
if condición; sentenza1; else; sentenza2; end
```

Sentenzas de selección (II)

- Sentenza *switch*:
- Compara a expresión cos distintos valores *val1...* e executa as sentenzas asociadas ao valor co cal coindice
- Se non coincide con ningún valor, execútanse *sentenzasN* (isto é opcional, pero permite aforrar un caso)

```
switch expresión  
case val1  
    sentenzas1;  
case val2  
    sentenzas2;  
  
...  
otherwise  
    sentenzasN;  
end
```

Sentenza de iteración definida (I)

- Se x é un vector fila, a é unha matriz:

```
for i=x  
    sentenzas  
end
```

```
for i=a(:)'  
    sentenzas  
end
```

Poño $a(:)'$ para que sexa un vector fila, porque $a(:)$ é un vector columna

- Nas sentenzas, i percorre os elementos do vector x ou da matriz a (por columnas)

```
for i=x  
    disp(i)  
end
```

```
for i=a(:)'  
    disp(i)  
end
```

- Para percorrer a por filas:

```
b=a';  
for i=b(:)'  
    disp(i)  
end
```

Sentenza de iteración definida (II)

- O vector x pode ser da forma *ini:paso:fin*

```
for var = ini:paso:fin  
    sentenzas;  
end
```

Cando so é unha sentenza

```
for var=ini:paso:fin; sentenza; end
```

- Pódense usar variábeis i , j (por defecto son a unidade imaxinaria: $i^2=-1$)
- Inicializa $var = ini$
- En cada iteración executa as sentenzas
- Se $var + paso > fin$ rematan as iteracións.
- En caso contrario, executa $var=var+paso$ e continúa coa seguinte iteración

Sentenza de iteración definida (III)

- Se $paso > 0$, entón debe ser $ini \leq fin$ para que haxa iteracións; se $paso < 0$, debe ser $ini \geq fin$
- A *var* pódenselle asignar valores específicos. Ex:

```
for k = [1 3 6 -4]
    fprintf('k=%i\n', k);
end
```
- Non se lle debe cambia-lo valor a *var* dentro do bucle *for* (Matlab non o detecta)
- Exemplo: suma da serie $\sum_{k=0}^n \frac{(-1)^k x^{2k+1}}{(2k+1)!}$

```
x=input('x?');n=input('nº de elementos? '); suma=0;
for k=0:n
    suma=suma+(-1)^k*x^(2*k+1)/factorial(2*k+1);
end
```

Bucle *for* con varios índices

- O *for* pódese poñer con dous ou tres índices (deben ser vectores fila)

```
i=1:3; j=4:6;  
for k=[i;j]  
    disp(k)  
end
```

Iter. 1: k=[1;4]
Iter. 2: k=[2;5]
Iter. 3: k=[3;6]

```
i=1:3; j=4:6;  
for k=[i;j]  
    fprintf('%i %i\n',k(1),k(2))  
end
```

1 4
2 5
3 6

- Con tres índices (caracteres):

```
x='abc'; y='def'; z='ghi';  
for k=[x;y;z]  
    fprintf('%c %c %c\n',k(1),k(2),k(3))  
end
```

a d g
b e h
c f i

Exemplos de estruturas iterativas definidas

- Mostrar por pantalla un vector na mesma liña:

```
v=randi([vmax vmin],1,n);  
fprintf('%7.3f ',v);fprintf('\n')
```

Estrutura iterativa implícita (vectorizada)

- Mostrar por pantalla unha matriz (unha fila en cada liña):

```
a=randi([vmin vmax],n,m);  
for i=1:n  
    fprintf('%10.2f ',a(i,:)); fprintf('\n')  
end
```

- Cálculo do máximo dunha serie de números (para o mínimo debes inicializar *m=inf*):

```
m=-inf;  
for i=1:n  
    x=input('x? ');m=max(m,x);  
end
```

Se os números están almacenados nun vector *x*: *min(x)*, *max(x)*

Sentenza de iteración indefinida

- Sentenza *while* (iteración indefinida): ten unha condición para continuar coa iteración

```
while condición  
    sentenzas;  
end
```

```
while condición;sentenza; end
```

- A condición debe ter alomenos unha variábel (se é nula, a condición é falsa, e certa en caso contrario)
- As variábeis da condición deben inicializarse antes (en caso contrario, erro de execución)
- Dentro das sentenzas débese modifica-lo valor de alomenos unha das variábeis da condición: en caso contrario, entrará nun bucle infinito

Sentenza de iteración indefinida

- As modificacións nestas variábeis deben garantir que a iteración acade un final: en caso contrario, iteración infinita (isto non o detecta Matlab)
- Non poñer condicións de igualdade estricta entre variábeis con valores reais, xa que o redondeo poden levar a que nunca se cumpran
- Ex: suma de serie $\sum_{n=0}^{\infty} \frac{x^n}{n!}$ con sumandos > 0.0001

```
suma = 0;sumando = 1;n = 0;x=1;  
while sumando > 0.0001  
    sumando = x^n/factorial(n);  
    suma = suma + sumando; n = n + 1;  
end
```

```
n=1:1000;  
sum(x.^n./factorial(n))
```

↑
Versión vectorizada

Sentenzas *break* e *continue*

- Sentenza *break*:
 - Provoca o remate inmediato da estrutura iterativa, se está dentro dun bucle *for* ou *while*
 - Se non está dentro dun bucle, remátase o programa (p.ex., se se introducen datos inválidos)
- Sentenza *continue*: provoca o paso inmediato á seguinte iteración, saltando a execución do que resta da iteración actual
- O *continue* debe estar dentro dunha estrutura de selección (para que so se execute cando se cumpra a condición)

Exemplo de *break* en bucle doble con matrices

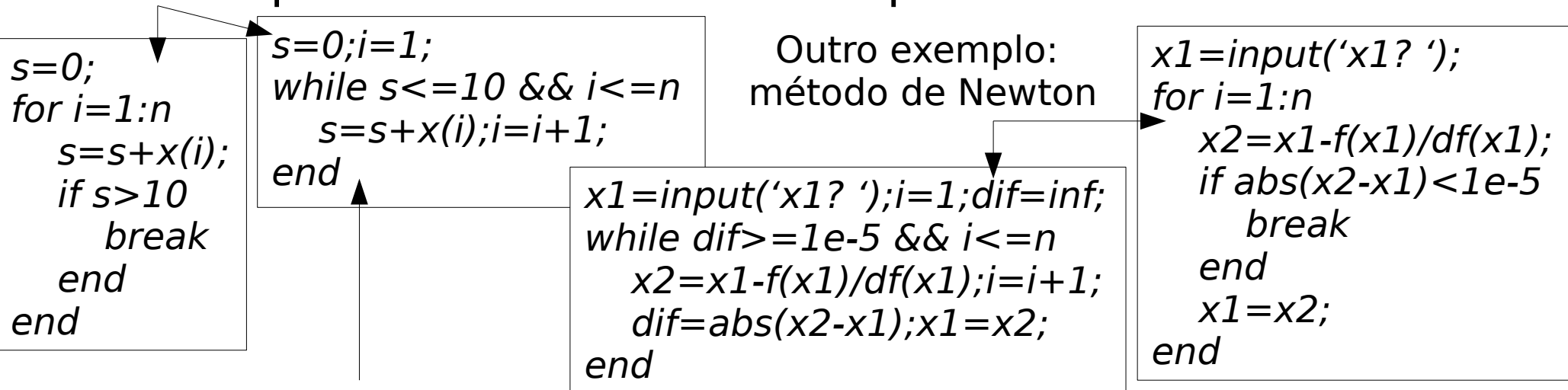
- ¿Cómo rematar un bucle doble?
- Exemplo: busca un elemento impar nunha matriz:

```
clear all
% a=matriz nxn enteira aleatoria en {1..10}
n=5;a=randi(10,n,n);impar=0;
for i=1:n
    for j=1:n
        if rem(a(i,j),2)==1
            impar=1; break
        end
    end
    if impar; break end
end
fprintf('elemento (%i,%i) con valor %g é impar\n',i,j,a(i,j));
```

Podes vectorizalo: `any(rem(a(:),2)==1)`
`disp(find(rem(a(:),2)==1,1,'first'))`

Sentenzas de iteración híbrida

- Teñen un número máximo de iteracións (parte definida) pero poden rematar antes se se cumpre unha condición (parte indefinida).
- Pódese facer cun *for+break* ou cun *while+and* lóxico.
- Exemplo: executa n iteracións pero remata cando $s > 10$:



- O *while+and* é útil para controlar os índices dun vector ou matriz, evitando superar os rangos dos seus índices.

Exemplo: Conversión dun vector por filas/columnas a unha matriz

- Manualmente (conversión por filas): vector v de n elementos, matriz a de orde $m \times m$, con $m = \text{ceil}(\text{sqrt}(n))$

```
clear all
n=10;v=randi(10,1,n);m=ceil(sqrt(n));a=zeros(m);k=1;
for i=1:m
    for j=1:m
        a(i,j)=v(k);k=k+1;
        if k>n; break; end % para evitar sairse do vector
    end
    if k>n; break; end
end
disp(v);disp(a)
```

- Para convertir por columnas: $a(j,i)=v(k)$
- Automáticamente: `reshape([1 2 3 4],2,2)` por columnas; `reshape([1 2 3 4],2,2)'` por filas. O vector debe ter exactamente o mesmo nº de elementos que a matriz

Exemplo: Conversión dunha matriz nun vector

- Manualmente: matriz a de orde $n \times n$ a vector v de $m = n^2$ elementos (conversión por filas):

```
clear all
n=5;a=randi(10,n,n);m=n^2;v=zeros(1,m);k=1;
for i=1:n
    for j=1:n
        v(k)=a(i,j);k=k+1;
    end
end
disp(a);disp(v)
```

- Para convertir por columnas: $v(k) = a(j,i)$
- Automáticamente: $a(:,i)$, como vector columna; $a(:,i)'$, como vector fila; $\text{reshape}(a,1,m)$ por columnas; $\text{reshape}(a',1,m)$ por filas.