INFORMATICA

eiae
UPM

Evaluación

- Primer cuatrimestre (40% de la nota final)
  - FECHAS EXÁMENES PARCIALES:
    - 18-OCTUBRE-2018 HORARIO DE CLASE – AULA E4
    - 13-DICIEMBRE-2018 HORARIO DE CLASE – AULA ¿?
  - ES OBLIGATORIO APROBAR EL PRIMER CUATRIMESTRE PARA CONTINUAR CON LA EVALUACIÓN CONTINUA.

Evaluación

		Teoría	Presentaciones 1	Presentaciones 2
6	Septiembre	Introducción		
13	Septiembre	Datos y Operadores		
20	Septiembre	Control de flujo I		
27	Septiembre	Control de flujo II		
4	Octubre	Arrays I		
11	Octubre	Arrays II	CPU	RAM
18	Octubre	EXAMEN PARCIAL		
25	Octubre	Intrínsecas	DISCO DURO	GPU
8	Noviembre	Formato lectura escritura	FPGA	XEON PHI
15	Noviembre	Programación modular I	CLUSTER	RASPBERRY PI
22	Noviembre	Programación modular II	HISTORIA LENG	FUTURO LENG
29	Noviembre	Tipos derivados		
13	Diciembre	EXAMEN PARCIAL		
20	Diciembre	Programación avanzada	RESTO DE PRESENTACIONES	

¿ALGUIEN SIN TEMA?
¿ALGUIEN CON DOS O MÁS TEMAS?

Clase de hoy

- Repaso de reserva de memoria (estática y dinámica)
- Operaciones con vectores y matrices
- Práctica: multiplicación de una matriz por un vector

Arrays: Vectores y Matrices: Declaración II

- Parámetro

```
program declaracion

integer, parameter :: n = 2
integer, parameter :: m = 3
integer, parameter :: Linf = 0
integer, parameter :: Lsup = 10

integer :: U(n)
real :: V(n*m)
real :: A(n,m)
real :: T(Linf:Lsup, Linf:Lsup)

end program declaracion
```

- Dinámica (luego con más detalle)

```
program declaracion

integer :: n
Integer :: m

integer, allocatable :: U(:)
real,    allocatable :: A(:,:)

end program declaracion
```

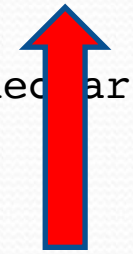
Arrays: Vectores y Matrices: Declaración II

- Parámetro

```
program declaracion

integer, parameter :: n = 2
integer, parameter :: m = 3
integer, parameter :: Linf = 0
integer, parameter :: Lsup = 10

integer :: U(n)
real :: V(n*m)
real :: A(n,m)
real :: T(Linf:Lsup, Linf:Lsup)

end program declaracion
```

- Dinámica (luego con más detalle)

```
program declaracion

integer :: n
Integer :: m

integer, allocatable :: U(:)
real,    allocatable :: A(:,:)

end program declaracion
```



`U(:)` es un vector y `A(:,:)` una matriz de tamaño desconocidos al principio del programa

Arrays: Vectores y Matrices: Asignación II

- Parámetro

```
program declaracion

integer, parameter :: n = 2
integer, parameter :: m = 3

integer :: U(n)
real :: A(n,m)

A = 0.d0
U = 0.d0

end program declaracion
```

- Igual que antes

- Dinámica (luego con más detalle)

```
program declaracion

integer :: n
Integer :: m

integer, allocatable :: U(:)
real,    allocatable :: A(:,:)

! Cuerpo de programa
n = 2
m = 3
allocate(U(n))
allocate(A(n,m))

A = 0.d0
U = 0.d0

end program declaracion
```

Arrays: Vectores y Matrices: Asignación II

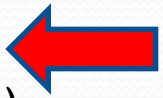
- Parámetro

```
program declaracion

integer, parameter :: n = 2
integer, parameter :: m = 3

integer :: U(n)
real :: A(n,m)

A = 0.d0
U = 0.d0

end program declaracion
```

- Igual que antes

- Dinámica (luego con más detalle)

```
program declaracion

integer :: n
Integer :: m

integer, allocatable :: U(:)
real,    allocatable :: A(:,:)

! Cuerpo de programa
n = 2
m = 3
allocate(U(n))
allocate(A(n,m))

A = 0.d0
U = 0.d0

end program declaracion
```



Reserva dinámica de memoria para U(:)y A(:,:)

Arrays: Vectores y Matrices: Asignación III

- Asignación por bucle

```
program asignacion

integer, parameter :: n = 10
integer :: U(n)
integer :: i

!--- Fin declaracion -------------------

do i=1,n
        U(i) = i*i
end do

write(*,*) U

do i=1,n
        write(*,*) U(i)
end do

end program asignacion
```

Arrays: Vectores y Matrices: Asignación III

Escribir un programa que defina la matriz A_{nxn} con n = 100

$$a_{ij} = \begin{cases} i + j & Si \ \ i \leq j \\ 0 & Si \ \ i > j \end{cases}$$

$$A = \begin{pmatrix} 2 & 3 & 4 & \cdots & 101 \\ 0 & 4 & 5 & \cdots & 102 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 200 \end{pmatrix}$$

Arrays: Vectores y Matrices: Asignación III

Escribir un programa que defina la matriz A_{nxn} con n = 100

$$a_{ij} = \begin{cases} i+j & Si\ i \le j \\ 0 & Si\ i > j \end{cases} \qquad A = \begin{pmatrix} 2 & 3 & 4 & \cdots & 101 \\ 0 & 4 & 5 & \cdots & 102 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 200 \end{pmatrix}$$

```
program main
integer, parameter :: n = 100
integer :: A(n,n)
integer :: i, j
!--- Fin declaracion ------------------

do i=1,n
        do j=1,n
                if (i <= j) then
                        A(i,j) = i + j
                else
                        A(i,j) = 0
                end if
        end do
end do
end program main
```

Arrays: Vectores y Matrices: Asignación III

Escribir un programa que defina la matriz A_{nxn} con n = 100

$$a_{ij} = \begin{cases} i+j & Si\ i \le j \\ 0 & Si\ i > j \end{cases} \qquad A = \begin{pmatrix} 2 & 3 & 4 & \cdots & 101 \\ 0 & 4 & 5 & \cdots & 102 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 200 \end{pmatrix}$$

```
program main
integer, parameter :: n = 100
integer :: A(n,n)
integer :: i, j
!--- Fin declaracion ------------------

A = 0
do i=1,n
        do j=1,n
                if (i <= j) A(i,j) = i + j
        end do
end do
end program main
```

Arrays: Vectores y Matrices: Asignación III

Escribir un programa que defina la matriz A_{nxn} con n = 100

$$a_{ij} = \begin{cases} i+j & Si\ i \le j \\ 0 & Si\ i > j \end{cases} \qquad A = \begin{pmatrix} 2 & 3 & 4 & \cdots & 101 \\ 0 & 4 & 5 & \cdots & 102 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 200 \end{pmatrix}$$

```
program main
integer, parameter :: n = 100
integer :: A(n,n)
integer :: i, j
!--- Fin declaracion ------------------

A = 0
do i=1,n
        do j=i,n
                A(i,j) = i + j
        end do
end do
end program main
```

Asignación Dinámica

```
program declaracion

integer :: n
Integer :: m

integer, allocatable :: U(:)
real,    allocatable :: A(:,:)

! Cuerpo de programa

allocate(U(n))
allocate(A(n,m))

A = 0.d0
U = 0.d0

end program declaracion
```

- Inconvenientes de la declaración de un *array* con tamaño fijo:
  - Si el tamaño prefijado es mayor que el número de valores que se van a almacenar, estamos *malgastando* memoria.
  - Si el tamaño prefijado es menor que el número de valores que se van a utilizar, el programa dará un error de ejecución.
- Solución en Fortran: ***arrays dinámicos***.

Asignación Dinámica

```
program declaracion

integer :: n,info
integer, allocatable :: U(:)
! Cuerpo de programa

n = 5

allocate(U(n), stat=info)

if (info > 0) stop '** No hay memoria &
                  & suficiente para U**'

U = 0.d0


deallocate(U, stat=info)

if (info > 0) stop '** U no tenía&
                  & memoria reservada**'


end program declaracion
```

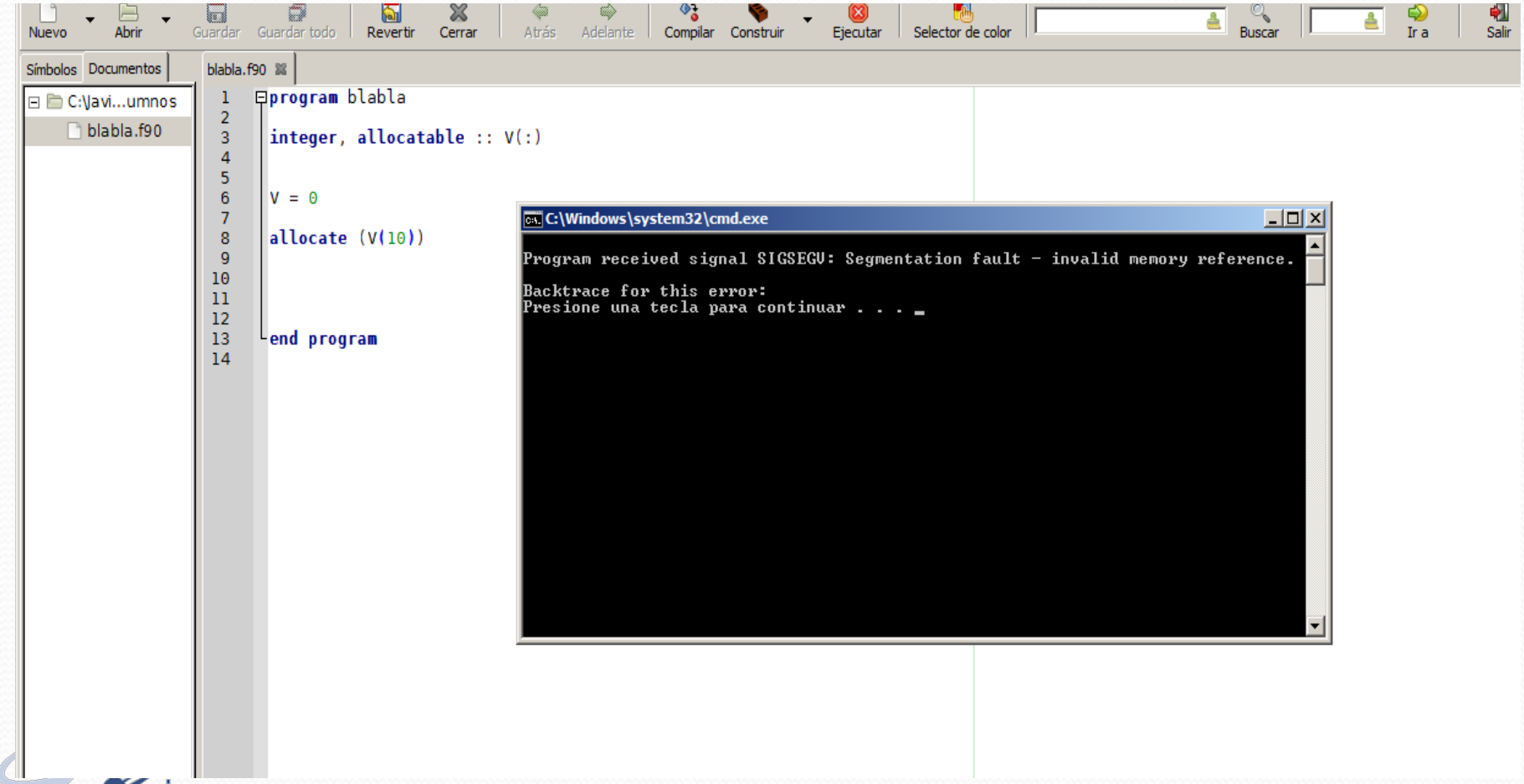
Declaración de U como array dinámico

Reserva de memoria, con control de error
Si la asignación es correcta stat = 0.
En caso contrario stat > 0

Libera la memoria previamente reservada, con control de error

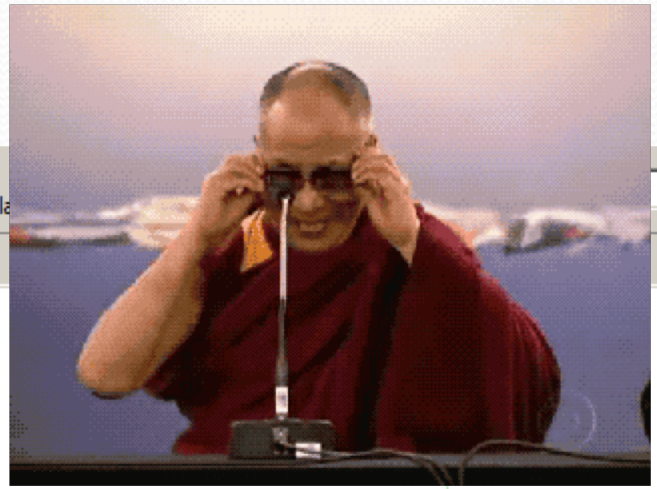
Estilo de programación:
Leyendo los mensajes de error

Reservar espacio para variables antes de usarlas



Estilo de programación:
Leyendo los mensajes de error

Respetad las dimensiones.



Guardar | Guardar todo | Revertir | Cerrar | Atrás | Adelante | Compila | Buscar

blabla.f90

```
1  program blabla
2
3  integer, allocatable :: V(:)
4  integer              :: n,m,i
5
6  n = 5
7  m = 10
8  allocate (V(n))
9
10 do i = 1, m
11
12     V(i) = i
13
14 enddo
15
16 write(*,*) V
17
18 end program
19
```

C:\Windows\system32\cmd.exe

```
At line 12 of file blabla.f90
Fortran runtime error: Index '6' of dimension 1 of array 'v' above upper bound o
f 5
Presione una tecla para continuar . . .
```

Operaciones con vectores y matrices

```
program vectores

integer, allocatable   :: u(:),v(:)
integer :: i

allocate (u(5),v(5))

do i = 1, 5
        u(i)= i
enddo
v = 1

u = u+v

write(*,*) u

end program
```

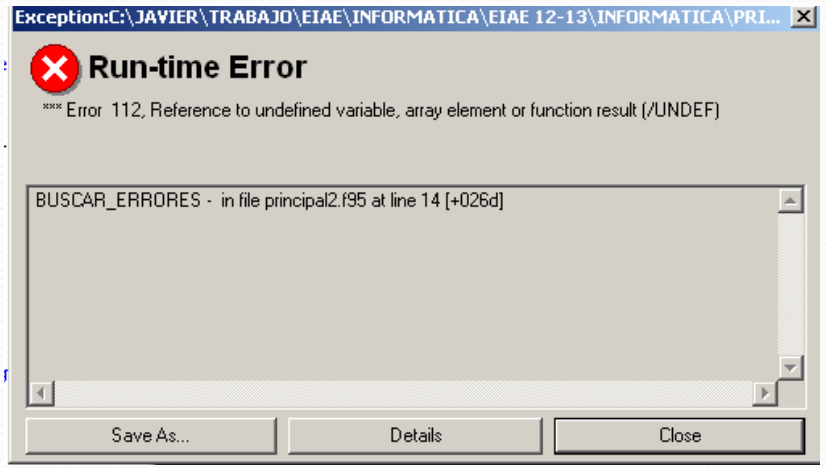
Operaciones con vectores y matrices

```
program vectores

integer, allocatable   :: u(:),v(:)
integer :: i

allocate (u(5),v(5))

do i = 1, 5
        u(i)= i
enddo
v = 1

u = u+v

write(*,*) u

end program
```

```
2  3  4  5  6
```

Operaciones con vectores y matrices

```
program vectores

integer, allocatable   :: u(:),v(:)
integer :: i

allocate (u(5),v(3))

do i = 1, 5
        u(i)= i
enddo
v = 1

u = u+v

write(*,*) u

end program
```



Operaciones con vectores y matrices

```
program vectores

integer, allocatable   :: u(:)
integer :: i,k

allocate (u(5))

do i = 1, 5
        u(i)= i
enddo
k = 2

u = k*u

write(*,*) u

end program
```

```
2  4  6  8  10
```

Operaciones con vectores y matrices

```
program suma_matrices_I

integer, allocatable      :: A(:,:),B(:,:),C(:,:)
integer  :: i,j

allocate (A(3,5),B(3,5),C(3,5))

A = 0
B = 0
C = 0
do i = 1, 3
        A(i,i)= 1
        do j = 1, 5
                B(i,j)=i+j
        enddo
enddo

C = A + B
do i = 1,3
        write(*,*) C(i,:)
enddo

end program
```

```
3 3 4 5 6
3 5 5 6 7
4 5 7 7 8
```

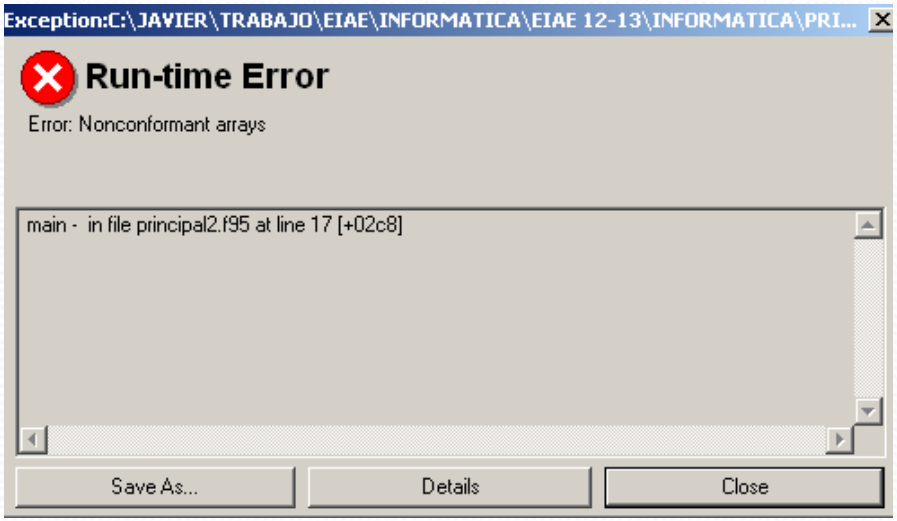
Operaciones con vectores y matrices

```
program suma_matrices_I

integer, allocatable      :: A(:,:),B(:,:)
integer  :: i,j

allocate (A(3,5),B(3,5))

A = 0
B = 0
do i = 1, 3
         A(i,i)= 1
         do j = 1, 5
                  B(i,j)=i+j
         enddo
enddo

write(*,*) A + B


end program
```

```
3  3  4  3  5  5  4  5  7
  5  6  7  6  7  8
```

Operaciones con vectores y matrices

```
program suma_matrices_I

integer, allocatable      :: A(:,:),B(:,:) ,C(:,:)
integer  :: i,j

allocate (A(5,3),B(3,5),C(5,5))

A = 0
B = 0
do i = 1, 3
        A(i,i)= 1
        do j = 1, 5
                B(i,j)=i+j
        enddo
enddo

C = A + B


end program
```



Operaciones con vectores y matrices

```
program producto_matriz

integer, allocatable      :: A(:,:)
integer  :: i,k

allocate (A(3,5))

A = 0
k = 7
do i = 1, 3
        A(i,i)= 1
enddo

A = k*A

do i = 1,3
        write(*,*) A(i,:)
enddo

end program
```

```
7 0 0 0 0
0 7 0 0 0
0 0 7 0 0
```

Funciones intrínsecas.

- Ejercicio 1. crear un programa que:
  - Defina tres variables A(3,3), X(3) y b(3) que sean una matriz y un vector de reales respectivamente.
  - La reserva de memoria se realizará de forma estática.
  - Asigne los valores de A y X:

$$A=\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad X=\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

  - El programa debe calcular el producto de la matriz A por el vector X y almacenarlo en un vector b.

Funciones intrínsecas.

- Ejercicio 2. crear un programa que:
  - Defina tres variables A(N,N), X(N) y b(N) que sean una matriz y un vector de enteros respectivamente.
  - La reserva de memoria se realizará de forma dinámica (N=100).
  - Asigne los valores de A y X:

$$a_{ij}=\begin{cases} i+j & Si\ \ i \le j \\ 0 & Si\ \ i > j \end{cases} \qquad A=\begin{pmatrix} 2 & 3 & 4 & \cdots & 101 \\ 0 & 4 & 5 & \cdots & 102 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 200 \end{pmatrix} \qquad X = i^2$$

  - El programa debe calcular el producto de la matriz A por el vector X y almacenarlo en un vector b.