INP TOULOUSE ENSEEIHT

Programmation impérative: Méthode de Programmation

2. Tableaux + Fichiers

Table des matières

Pourquoi un tableau ?

Tableau:

Ensemble ordonné d'éléments de même type désigné par un identificateur unique

Pourquoi un tableau? Parce que ça…

- a une signification physique propre

  Ex.: projection du vecteur sur l'axe i

- a le sens de la numérotation

  Ex.: i-ème valeur

- sert de lien entre les tableaux

  Ex.: a(i,j)*x(j)

Les tableaux simples (de rang 1)

Algorithmique	FORTRAN
	! définition et déclaration de variables
Soit un vecteur de 100 réels	REAL, DIMENSION(100) :: v1
Soit un vecteur de 100 entiers	INTEGER, DIMENSION(100) :: u1
Soit un vecteur de 100 logiques	LOGICAL, DIMENSION(100) :: w1
	! affectation et accès à un champ
Le vecteur v1 est nul	DO i = 1,100 v1(i) = 0. ⟺ v1(:)=0. ⟺ v1=0. (ambigu) END DO
La 50ème composante de v1 reçoit 1	v1(50) = 1.
La 12ème composante de v1 reçoit la 50ème	v1(12) = v1(50)

Manipulation des tableaux

Algorithmique

Soient 2 vecteurs de 100 réels

La 50ème composante de v2 reçoit la 32ème de v1

v2 ← v1

Chaque composante de v2 reçoit le double de la composante de v1 plus 3

Chaque composante de v1 reçoit le produit des composantes de v1 et v2

FORTRAN

```
! définition et déclaration de variables
REAL, DIMENSION(100) ::  v1, v2

! Manipulations d'un tableau
v2(50) = v1(32)

DO i = 1,100
   v2(i) = v1(i)  ⟺  v2(:) = v1(:)  ⟺  v2 = v1
END DO
```

(ambigu)


```
v2=2.*v1+3.

v1=v1*v2
```

(ambigu)

Bornes des indices

Algorithmique	FORTRAN
Créer 1 vecteur comportant 4 éléments de type réels	REAL, DIMENSION(4) :: v1 ! → v(1), v(2), v(3), v(4)
Créer 1 vecteur comportant 4 éléments de type réels dont l'indice commence par 1	REAL, DIMENSION(1:4) :: v1 ! → v(1), v(2), v(3), v(4)
Créer 1 vecteur comportant 4 éléments de type réels dont l'indice commence par 0	REAL, DIMENSION(0:3) :: v1 ! → v(0), v(1), v(2), v(3)
Créer 1 vecteur comportant 4 éléments de type réels dont l'indice commence par -10	REAL, DIMENSION(-10:-7) :: v1 ! → v(-10), v(-9), v(-8), v(-7)

Problème de dépassement d'indices

Algorithmique

Soient 2 vecteur de 10 réels

Que se passe-t-il pour les instructions suivantes?

Attention aux dépassement de tableaux !!!

Toujours vérifier les expressions en début et fin de boucle

FORTRAN

```
REAL, DIMENSION(10) ::  v1, v2

! v(1) v(2) … v(10) existent
! v(0), v(11), v(99), v(-30) n'existent pas…
```

```
v1(15) = 0.
v1(15) = v2(3)
v1(3)   = v2(15)

DO i = 1, 10
   v1(i) = v2(i+1)
END DO
```

Ecriture à l'écran d'un tableau

Algorithmique

Soit u(i)=1.2 avec i=1, …, 100

{ **Ecrire** à l'écran u en colonne }

Ex.: 1.2
1.2
…
1.2

{ **Ecrire** à l'écran u sur une ligne }

Ex.: 1.2 1.2 … 1.2

FORTRAN

```
INTEGER :: i
REAL, DIMENSION (100) :: u
u(:) = 1.2

DO i = 1, 100
    WRITE(*,*) u(i)
END DO
```

retour à la ligne à chaque appel de l'instruction

```
WRITE(*,*) ( u(i), i=1,100 )
```

```
WRITE(*,*) u
```

Exemple du tri (2)

Tri

Fonctionnalité : On veut trier par ordre croissant les éléments d'un vecteur contenant n éléments (n<=100)

1° Environnement

Identificateur	Type	Signification
n	Entier	nombre d'éléments à trier
u(1), ... , u(n)	Réel	vecteur de départ
v(1), ... , v(n)	Réel	vecteur « trié »

Exemple du tri (2)

2° Traitement

```
Tri
|
|  Lire (n)
|  Lire (u(1), …, u(n))
|
|  ! Détermination de la valeur « umax »
|  !  des u(1), … u(n)
|  umax ← u(1)
|  Pour i de 2 à n faire
|  |  Si  u(i) >= umax
|  |      Alors
|  |      |  umax ← u(i)
```

```
|  ! Recherche de la plus petite valeur umin
|  !  de u et de son emplacement jmin
|  Pour i de 1 à n faire
|  |  jmin ← 1
|  |  umin ← umax
|  |  Pour j de 1 à n faire
|  |  |  Si  u(j) <= umin
|  |  |      Alors
|  |  |      |  umin ← u(j)
|  |  |      |  jmin ← j
|  |  ! On met la valeur min dans v
|  |  v(i) ← umin
|  |  ! On « écrase » la plus petite valeur de u
|  |  u(jmin) ← umax
|
|  Affiche (v(1),…, v(n))
```

Exemple du tri (2)

```
PROGRAM TRI

   IMPLICIT NONE

   ! Declarations
   INTEGER :: n, i, j, jmin
   REAL :: umax, umin
   REAL, DIMENSION(100) :: u, v

   ! Instructions

   WRITE(*,*)"Donner le nb d'elements a trier"
   READ(*,*) n
   DO i = 1,n
      WRITE(*,*)'Valeur du',i,'eme element?'
      READ(*,*) u(i)
   END DO

   umax = u(1)
   DO i = 2, n
       IF( u(i) >= umax ) umax = u(i)
   END DO

   DO i = 1, n
       jmin = 1
       umin = umax
       DO j = 1, n
            IF ( u(j) <= umin ) THEN
                  umin = u(j)
                  jmin = j
             END IF
        END DO
        v(i) = umin
        u(jmin) = umax
   END DO

   WRITE(*,*) 'Les element tries sont:'
   WRITE(*,*) ( v(i), i = 1, n )

END PROGRAM TRI
```

Les matrices (tableaux de rang 2)

Algorithmique

Soit une matrice de 10×10 réels
Soit une matrice de 10×10 entiers

! Ex. : Création de la matrice Identité

1° création d'une matrice nulle

2° remplissage de la diagonale de 1

FORTRAN

```
! définition et déclaration
REAL, DIMENSION(10,10) ::  m1
INTEGER, DIMENSION(10,10) ::  m2

! affectation et accès à un champ
DO i = 1,10
  DO j = 1, 10
      m1(i,j) = 0.
  END DO
END DO
```

⟺ `m1(:,:)=0.`

```
DO i = 1,10
    m1(i,i) = 1.
END DO
```

Ecriture à l'écran d'une matrice

Algorithmique

Soit M la matrice identité de 10×10 réels

{ **Ecrire** à l'écran M }

M=
1. 0. 0. 0. …
0. 1. 0. 0. …
0. 0. 1. 0. …
…

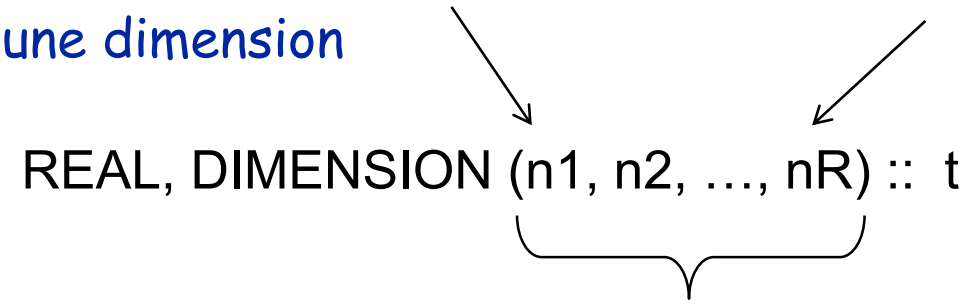
FORTRAN

```
! déclaration
REAL, DIMENSION(10,10) ::  M
INTEGER :: i,j

! instructions
M(:,:)=0.
DO i = 1,10
     M(i,i) = 1.
END DO

PRINT*, 'M='
DO i = 1,10
   PRINT*, (M(i,j), j=1,10)
END DO
```

Rang, étendue et profil des tableaux

Etendue : nb d'éléments suivant une dimension

Rang : nb de dimensions (R≤7 en Fortran 90)



! Exemples

```
INTEGER, DIMENSION (5) :: t                  ! Rang = 1 ;  Profil = (5)  ;  Etendue = 5
INTEGER, DIMENSION (10, 5) :: t1             ! Rang = 2 ;  Profil = (10,5)  ;  Etendue = 10,5
REAL, DIMENSION (-2:7, 0:4) :: t2            ! t2 a le même profil que t1 (même rang et étendue)
LOGICAL, DIMENSION (5, 5, 5, 5) :: test  ! Rang = 4  ;  Profil = (5,5,5,5)  ;  Etendue = 5,5,5,5
```

Manipulation de tableaux

Algorithmique

Exemple 1:
Calcul d'un produit scalaire
x=U.V

FORTRAN

```
REAL, DIMENSION(10) ::  u, v     ! Donnee
REAL ::  x                       ! Resultat
INTEGER :: i

! Produit scalaire
x = 0.
DO i = 1,10
     x = x + u(i)*v(i)
END DO
```

Manipulation de tableaux

Algorithmique

Exemple 2:

Calcul d'un produit matrice vecteur

B=AX

FORTRAN

```
REAL, DIMENSION(10,5) ::  a     ! Donnee
REAL, DIMENSION(5) ::  x        ! Donnee
REAL, DIMENSION(10) ::  b       ! Resultat
INTEGER :: i, j


! Produit matrice-vecteur
DO i = 1,10
  DO j = 1, 5
       b(i) =  a(i,j)*x(j)
  END DO
END DO
```

Suite de Fibonacci

Ecrire un programme en FORTRAN qui détermine l'ensemble des valeurs $u_1, u_2, \ldots, u_n$ ($n \leq 100$ étant fourni en donnée) de la suite définie comme suit:

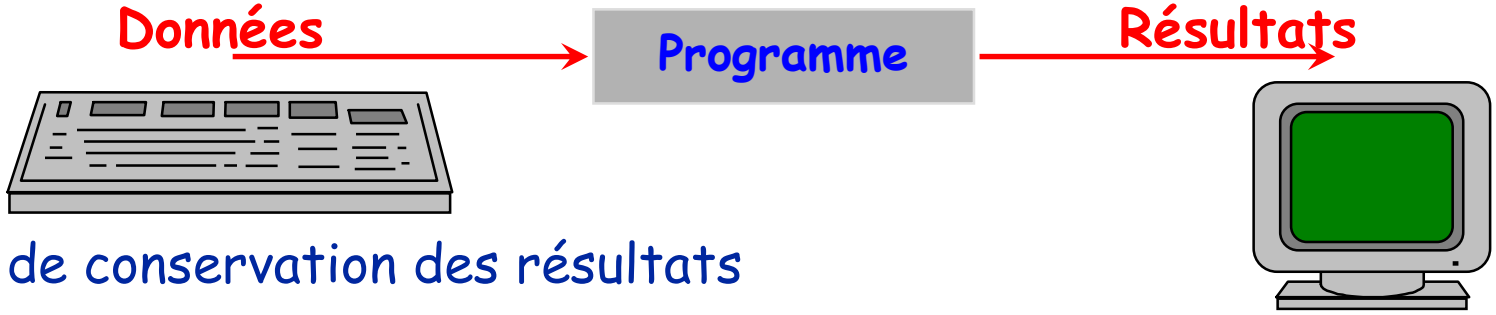
$u_1 = 1$ $\quad u_2 = 1$ $\quad u_n = u_{n-1} + u_{n-2}$ (pour n>2)

Les fichiers

Notion de fichier

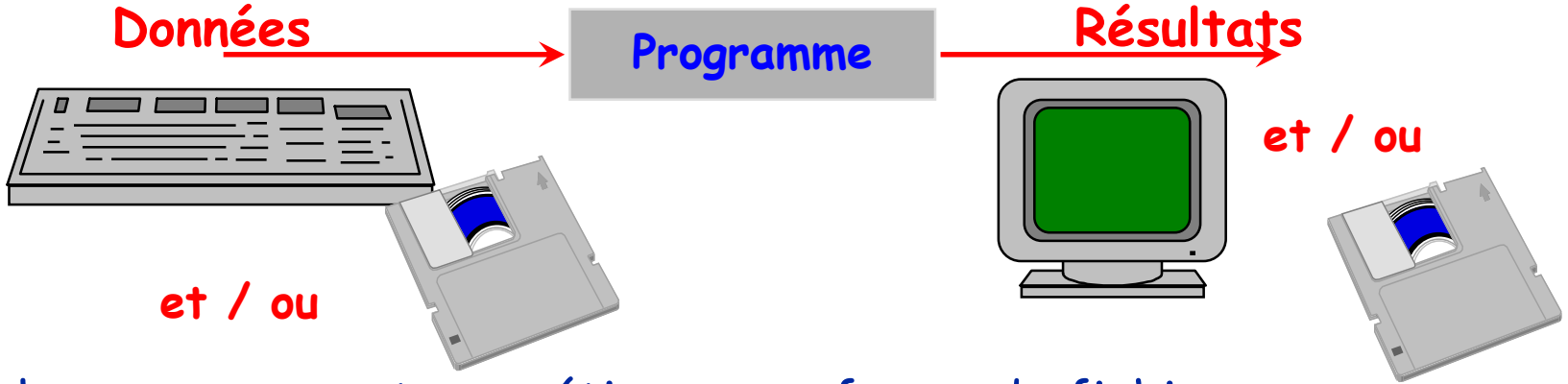
Organisation

PROBLEMATIQUE



Pas de conservation des résultats

Entrées des données fastidieuses



stockage sur support magnétique sous forme de fichiers

Organisation d'un fichier

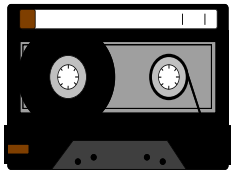
Un fichier est une suite d'articles (ou enregistrements) en général de même nature

Exemples :

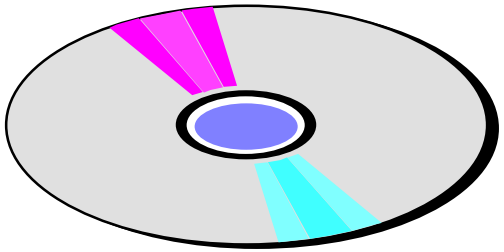
- fichier d'adresses
- fichier de composants électroniques
- fichier de bouteilles de vin

Exception :
les fichiers textes

Accès séquentiel



Accès direct



Pour relire un fichier que l'on a constitué,
il faut connaître la forme de son organisation

Fichiers formatés / non-formatés

2 façons de représenter l'information:

Fichiers formatés (ou « fichiers texte »)

- transfert « mémoire → fichier » avec modification de l'info
  (codage binaire -> codage en base 10 -> chaine de caractère)
- volume des infos dans fichiers > volume des infos dans mémoire ☹
- fichiers lisibles avec éditeurs de texte + portabilité entre machines ☺

Fichiers non-formatés (ou « fichier binaires »)

- transfert « mémoire → fichier » sans modification de l'info
- volume des infos dans fichiers = volume des infos dans mémoire (codage binaire) ☺
- vitesse de transfert plus rapide que pour les fichiers formatés ☺
- fichiers non-lisibles avec éditeurs de texte + pb de portabilité possibles ☹

Exemples de fichiers formatés / non-formatés ouvert avec un éditeur de texte

Fichier formaté

62 62 3
0.0000000000000000 0.0000000000000000
0.0000000000000000 0.0000000000000000
0.1000000000000000 0.1500000000000000
0.2500000000000000 0.3000000000000000
0.4000000000000000 0.4500000000000000
0.5500000000000000 0.6000000000000001

Fichier non-formaté

?¹ZÊ…ý/á? É
àä?úR<…
Öè?ÈeyGê?~2à8Jñ?RÏ„ûð?òãL!óãó?Qæ´
LPE÷?ÔßO8çö?¹úÛbø?ªÿ;Dh¶ü?y®Bd?@
¹- DXi?@¼ð¦=Æ¥?@eÖ¼ZX@ÁÝ
Ýª@gnB @¥¤ísÛ8 @“ÝÞ«ãÂ
@D1YÆs@¶¼SÕ @Q`4ð©
@ÇÕ%[

Quel types de fichiers pour quels type d'accès?

Entrée - Sortie	Formatée	Non Formatée
Séquentiel	X On sait comment on a écrit dans le fichier	
Direct		X On sait où on a écrit dans le fichier

Remarque:

- Lire/écrire en direct dans des fichiers formatés est possible (mais moins utilisé)
- Lire/écrire en séquentiel dans des fichiers non-formatés est possible (mais moins utilisé)

Lecture/écriture dans un fichier (séquentiel / formaté)

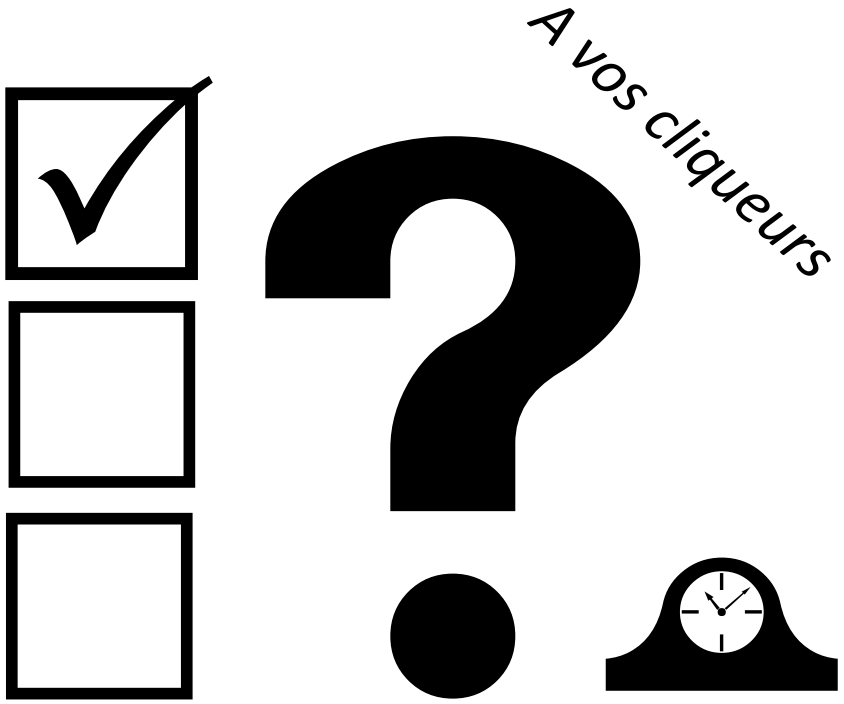
A chaque fichier est associé un numéro (numéro d'unité logique) qui devient le « nom » du fichier pour le programme

Algorithmique

Ouvrir le fichier toto.txt
Si le fichier existe: il l'ouvre
Si le fichier n'existe pas: il le crée
Lire la valeur de x et y dans toto.txt
Ecrire la valeur de z dans toto.txt
Fermer le fichier toto.txt

FORTRAN

```
PROGRAM lecture_fichier
IMPLICIT NONE
REAL :: x,y,z

OPEN (10, file = 'toto.txt')
Z=99.9
READ (10, *) x, y
WRITE (10, *) z
CLOSE (10)
END PROGRAM lecture_fichier
```


A vos cliqueurs

Que se passe-t-il à l'exécution du programme avec ce fichier toto.txt ?

Quiz n°1

```
PROGRAM lecture_fichier
IMPLICIT NONE
REAL :: x,y,z

 OPEN(10,file='toto.txt')
 Z=99.9
 READ(10,*) x, y
 WRITE(10,*) z
 CLOSE(10)

END PROGRAM lecture_fichier
```

Contenu du fichier toto.txt :

```
10.2    21.2
```

A. Erreur à l'exécution

B. Exécution OK et fichier modifié

C. Exécution OK et fichier non-modifié

Que se passe-t-il à l'exécution du programme avec ce fichier toto.txt ?

Quiz n°2

```
PROGRAM lecture_fichier
IMPLICIT NONE
REAL :: x,y,z

 OPEN(10,file='toto.txt')
 Z=99.9
 READ(10,*) x, y
 WRITE(10,*) z
 CLOSE(10)

END PROGRAM lecture_fichier
```

Contenu du fichier toto.txt :

```
10.2    21.2    32.3
```

A. Erreur à l'exécution

B. Exécution OK et fichier modifié

C. Exécution OK et fichier non-modifié

Que se passe-t-il à l'exécution du programme avec ce fichier toto.txt ?

Quiz n°3

```
PROGRAM lecture_fichier
IMPLICIT NONE
REAL :: x,y,z

 OPEN(10,file='toto.txt')
 Z=99.9
 READ(10,*) x, y
 WRITE(10,*) z
 CLOSE(10)

END PROGRAM lecture_fichier
```

Contenu du fichier toto.txt :

```
10.2
21.2
```

A. Erreur à l'exécution

B. Exécution OK et fichier modifié

C. Exécution OK et fichier non-modifié

Que se passe-t-il à l'exécution du programme avec ce fichier toto.txt ?

Quiz n° 4

```
PROGRAM lecture_fichier
IMPLICIT NONE
REAL :: x,y,z

 OPEN(10,file='toto.txt')
 Z=99.9
 READ(10,*) x, y
 WRITE(10,*) z
 CLOSE(10)

END PROGRAM lecture_fichier
```

Contenu du fichier toto.txt :

```
10.2    21.2
32.3
```

A. Erreur à l'exécution

B. Exécution OK et fichier modifié

C. Exécution OK et fichier non-modifié

Que se passe-t-il à l'exécution du programme avec ce fichier toto.txt ?

Quiz n°5

```
PROGRAM lecture_fichier
IMPLICIT NONE
REAL :: x,y,z

 OPEN(10,file='toto.txt')
 Z=99.9
 READ(10,*) x, y
 WRITE(10,*) z
 CLOSE(10)

END PROGRAM lecture_fichier
```

Contenu du fichier toto.txt :

```
10.2    21.2    ! Valeurs de x et y
```

A. Erreur à l'exécution

B. Exécution OK et fichier modifié

C. Exécution OK et fichier non-modifié

Que font ces deux programmes ?

Quiz n°6

```
PROGRAM ecriture_fichier
IMPLICIT NONE
REAL, DIMENSION(5) :: x
INTEGER :: i

 x(:)=12.34
 OPEN(10,file='res.dat')
 WRITE(10,*) (x(i),i=1,5)
 CLOSE(10)

END PROGRAM ecriture_fichier
```

```
PROGRAM ecriture_fichier
IMPLICIT NONE
REAL, DIMENSION(5) :: x
INTEGER :: i

 x(:)=12.34
 OPEN(10,file='res.dat')
 DO i=1,5
   WRITE(10,*) x(i)
 END DO
 CLOSE(10)

END PROGRAM ecriture_fichier
```

Que font ces deux programmes ?

Quiz n°6

```
PROGRAM ecriture_fichier
IMPLICIT NONE
REAL, DIMENSION(5) :: x
INTEGER :: i

 x(:)=12.34
 OPEN(10,file='res.dat')
 WRITE(10,*) (x(i),i=1,5)
 CLOSE(10)

END PROGRAM ecriture_fichier
```

Contenu du fichier res.dat :

```
12.34  12.34  12.34  12.34  12.34
```

```
PROGRAM ecriture_fichier
IMPLICIT NONE
REAL, DIMENSION(5) :: x
INTEGER :: i

 x(:)=12.34
 OPEN(10,file='res.dat')
 DO i=1,5
   WRITE(10,*) x(i)
 END DO
 CLOSE(10)

END PROGRAM ecriture_fichier
```

Contenu du fichier res.dat :

```
12.34
12.34
12.34
12.34
12.34
```

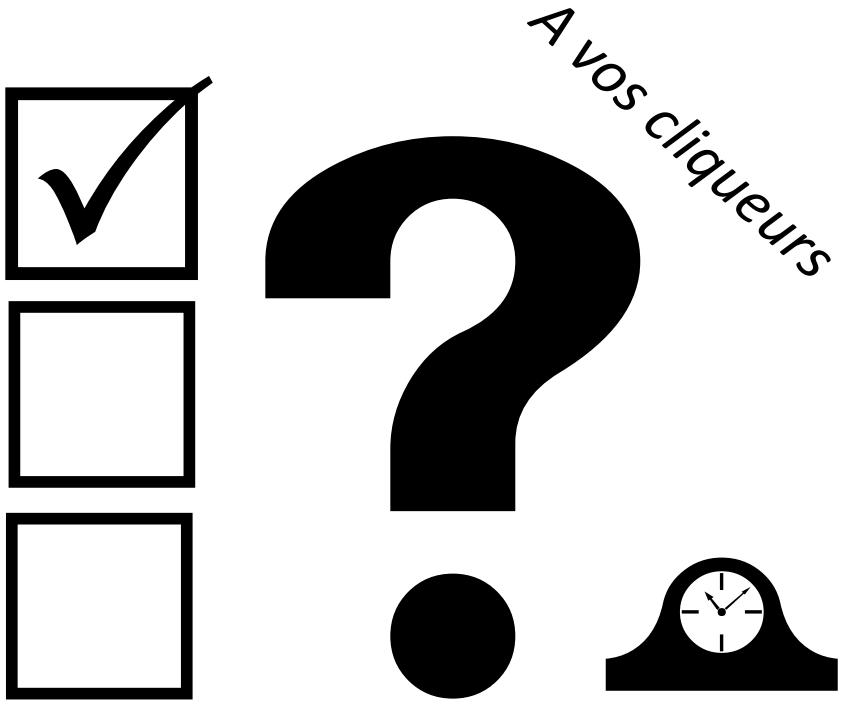
Options d'ouverture d'un fichier

FORTRAN

```
! Version courte
OPEN (10, file = 'toto.txt')
! Version moyenne
OPEN (unit = 10, file = 'toto.txt', form = 'formatted', status = 'new')
```

n° d'unité — nom du fichier — type de représentation — statut du fichier

```
! Version très longue…
OPEN ( unit = 10, file = 'toto.txt',                &
       form = 'formatted'/'unformatted',            &  ! fichier formaté ou non
       status = 'old'/'new',                        &  ! fichier existant ou nouveau
       access = 'sequential'/'direct',              &  ! accès séquentiel ou direct
       position = 'rewind'/'append',                &  ! Position du 'curseur' en début ou en fin
       advance = 'yes'/'no',                        &  ! Revient à la ligne ou non
       iostat = err )                                  ! err=0 si tout s'est bien passé, ≠ 0 sinon
```


A vos cliqueurs

Ces programmes sont-ils équivalents? Quiz

```
! Version n°1
 PROGRAM COUCOU
   IMPLICIT NONE
   OPEN(33, file='coucou.txt')
   WRITE(33,*) 'Coucou'
   CLOSE(33)
 END PROGRAM COUCOU
```

```
! Version n°2
PROGRAM COUCOU
  IMPLICIT NONE
  OPEN(22, file='coucou.txt')
  WRITE(*,*) 'Coucou'
  CLOSE(22)
END PROGRAM COUCOU
```

```
! Version n°3
PROGRAM COUCOU
  IMPLICIT NONE
  INTEGER :: num_fich
  CHARACTER (len=20) :: nom_fich
  num_fich = 44
  nom_fich = 'coucou.txt'
  OPEN (unit=num_fich, file = nom_fich, &
            form = 'formatted', status = 'new')
  WRITE(num_fich,*) 'Coucou'
  CLOSE(num_fich)
END PROGRAM COUCOU
```

A. oui

B. Version n°1 $\Leftrightarrow$ Version n°2 $\neq$ Version n°3

C. Version n°1 $\Leftrightarrow$ Version n°3 $\neq$ Version n°2

D. Version n°1 $\neq$ Version n°2 $\Leftrightarrow$ Version n°3

E. ils sont tous différents

Exemple du tri (3)

Tri

Fonctionnalité : On veut trier par ordre croissant les éléments d'un vecteur contenant n éléments (n<=100), stockés dans un fichier texte nommé don.dat contenant les informations sous la forme suivante:

« n
u(1)
...
u(n) »

On veut écrire le résultats dans un fichier texte appelé res.dat ayant la même forme que don.dat (mais avec les éléments rangés dans le bon ordre…).

Exemple du tri (3)

```
PROGRAM TRI
  IMPLICIT NONE

  ! Declarations
  INTEGER :: n, i, j, jmin
  REAL :: umax, umin
  REAL, DIMENSION(100) :: u, v

  ! Instructions
  ! Récupération des donnees du fichier don.dat
  OPEN(10, file='don.dat')
  READ(10,*) n
  DO i = 1, n
    READ(10,*) u(i)
  END DO
  CLOSE(10)
  umax = u(1)
  DO i = 2, n                ! Determination du max de u
     IF( u(i) >= umax ) umax = u(i)
  END DO
  DO i = 1, n                ! Recherche du min + indice de u
       jmin = 1
       umin = umax
       DO j = 1, n
          IF ( u(j) <= umin ) THEN
               umin = u(j)
               jmin = j
           END IF
       END DO
       v(i) = umin           ! Met la valeur min dans v
       u(jmin) = umax        ! Ecrase la valeur min de u
  END DO
  OPEN(20, file='res.dat') ! Ecriture dans res.dat
  WRITE(20,*) n
  DO i = 1, n
    WRITE(20,*) v(i)
  END DO
  CLOSE(20)
END PROGRAM TRI
```

Exercices

Suite de Fibonacci

Ecrire un programme qui détermine l'ensemble des valeurs u_1, u_2, …, u_n ($n \leq 100$ étant fourni en donnée) de la suite définie comme suit:

$u_1 = 1$ $\quad u_2 = 1$ $\quad u_n = u_{n-1} + u_{n-2}$ (pour n>2)

Les valeurs n, u_1 et u_2 sont stockées dans le fichier formaté don_entree.dat sous la forme: « n u_1 u_2 ». Les résultats seront écrits dans le fichier texte don_sortie.dat sous la forme:

« Les 'n' premiers termes de la suite de Fibonacci sont:
u(1)
…
u(n) »