# Les sous-programmes: exercices corrigés en F
# Corrigé

**Exercice 1 : Livres**

L'objectif de cet exercice est d'écrire un programme pour gérer une bibliothèque personnelle de taille réduite. On ne développera qu'une interface homme/machine minimale.

**1.1** *Définition d'un livre.* Un livre est caractérisé par son titre, son nombre de pages et son genre. Le genre est soit roman, soit BD, soit art, soit technique, etc. Il faudrait bien sûr bien d'autres informations pour décrire complètement un livre.

**1.1.1** Donner le type Livre.

**Solution :** D'après sa caractérisation, un livre peut être représenté par un type énuméré avec des champs titre (une chaîne de caractères), son nombre de page (entier) et son genre (un type énuméré puisqu'il peut prendre une parmi plusieurs valeurs).

```
Constante
    LG_TITRE = 50       -- Longueur maximal pour le titre d'un livre
Type
    Genre = (ROMAN, BD, ART, TECHNIQUE)

    Livre =
        Enregistrement
            titre: Chaîne[LG_TITRE]
            nb_pages: Entier
            genre: Genre
        FinEnregistrement
```

```
INTEGER,PARAMETER,PUBLIC:: LG_TITRE=50        ! Longueur maximal pour le titre d'un li
INTEGER,PARAMETER,PUBLIC :: ROMAN=1,BD=2,ART=3,TECHNIQUE=4

TYPE,PUBLIC:: typ_Livre
    CHARACTER(LEN=LG_TITRE):: titre
    INTEGER:: nb_pages
    INTEGER:: genre
END TYPE typ_Livre
```

**1.1.2** Écrire un sous-programme qui permet d'initialiser un livre à partir de son titre, de son nombre de pages et de son genre.

**Solution :** Ce sous-programme prend en paramètre :

- le livre à initialiser (**out**) ;
- le titre du livre (**in**) ;
- le nombre de pages (**in**)
- et le genre du livre (**in**).

Comme il s'agit d'un sous-programme d'initialisation, nous en faisons une procédure dont voici la spécification.

```
Procédure initialiser_livre(livre: out Livre;
                            titre: in Chaîne;
                            npage: in Entier;
                            genre: in Genre) Est
    -- Initialiser un livre à partir de son titre, son nombre de pages
    -- et son genre.
    --
    -- Nécessite
    --  npage > 0
    --
    -- Assure
    --  livre.titre = titre
    --  livre.nb_pages = npages
    --  livre.genre = genre
```

Le code du sous-programme est alors immédiat puisqu'il s'agit d'initialiser tous les champs de l'enregistrement.

```
Début
    livre.titre <- titre
    livre.nb_pages <- npages
    livre.genre <- genre
Fin
```

```
! Initialiser un livre a partir de son titre, son nombre de pages et
! son genre.
!
! Necessite
!        npages > 0

INTEGER,PARAMETER,PUBLIC::MAX_LIVRES=1000
        ! nombre maximal de livres dans la bibliotheque

TYPE,PUBLIC:: typ_Bibliotheque
    TYPE(typ_Livre),DIMENSION(MAX_LIVRES)::Livres
    INTEGER:: nb_livres
END TYPE typ_Bibliotheque


CONTAINS

SUBROUTINE initialiser_livre(livre,titre,npages,genre)
TYPE(typ_Livre),INTENT(OUT)::livre
CHARACTER(LEN=*),INTENT(IN)::titre
INTEGER,INTENT(IN)::npages
INTEGER,INTENT(IN)::genre

    livre%titre=titre
    livre%nb_pages = npages
    livre%genre = genre
```

```
27
28 END SUBROUTINE initialiser_livre
```

**1.1.3** Écrire un sous-programme qui permet d'afficher un livre.
**Solution :** Ce sous-programme n'a qu'un seul paramètre, le livre qui est en entrée. De plus il fait de l'affichage. Il s'agit donc nécessairement d'une procédure.

```
1 Procédure afficher_livre(livre: in Livre) Est
2     -- Afficher un livre
```

Le code consiste à afficher chacun des champs.

```
1 Début
2     Afficher livre.titre
3     Afficher livre.nb_pages
4     Afficher livre.genre
5 Fin
```

Pour afficher le genre du livre, on peut écrire un nouveau sous-programme.

```
1  ! Afficher le genre d'un livre
2  SUBROUTINE afficher_genre(genre)
3      INTEGER,INTENT(IN):: genre
4      SELECT CASE (genre)
5          CASE (ROMAN)
6              PRINT*,"ROMAN"
7          CASE (BD)
8              PRINT*,"BD"
9          CASE(ART)
10             PRINT*,"ART"
11         CASE(TECHNIQUE)
12             PRINT*,"TECHNIQUE"
13         CASE DEFAULT
14             PRINT*,"ERREUR !!!!"
15     END SELECT
16 END SUBROUTINE afficher_genre
17
18 ! Afficher un livre.
19 SUBROUTINE afficher_livre(livre)
20 TYPE(typ_Livre),INTENT(IN)::livre
21     PRINT*, livre%titre, livre%nb_pages," pages "
22     CALL afficher_genre(livre%genre)
23 END SUBROUTINE afficher_livre
```

**1.2** *La bibliothèque.* Nous nous limitons à une bibliothèque qui ne peut pas contenir plus de 1000 ouvrages qui sont rangés dans l'ordre alphabétique de leur titre.

**1.2.1** Définir le type bibliothèque.
**Solution :** Nous représentons la bibliothèque par un tableau de capacité 1000. Cependant, pour connaître le nombre d'ouvrages effectivement présents, il faut gérer la taille (entier). La bibliothèque est donc un type enregistrement.

```
1 Constante
2     MAX_LIVRES = 1000   -- nb maximal de livres dans la bibliothèque
```

```
Type
    Bibliothèque =
        Enregistrement
            livres: Tableau [1..MAX_LIVRES] De Livre
            nb_livres: Entier
        FinEnregistrement
```

```
INTEGER,PARAMETER,PUBLIC::MAX_LIVRES=1000
        ! nombre maximal de livres dans la bibliotheque

TYPE,PUBLIC:: typ_Bibliotheque
    TYPE(typ_Livre),DIMENSION(MAX_LIVRES)::Livres
    INTEGER:: nb_livres
END TYPE typ_Bibliotheque
```

**1.2.2** Écrire un sous-programme pour initialiser une bibliothèque à vide.
**Solution :** Ce sous-programme prend en paramètre la bibliothèque (**out**). Il s'agit d'une procédure.

```
Procédure initialiser_bibliothèque(b: out Bibliothèque) Est
    -- Initialiser la bibliothèque à vide
    --
    -- Assure
    --  b.nb_livres = 0 -- bibliothèque vide
```

Le code est donc immédiat.

```
Début
    b.nb_pages <- 0
Fin
```

Notons qu'il n'est pas nécessaire d'initialiser les livres du tableau puisque la champs `nb_pages` mis à zéro indique qu'on ne peut accéder à aucune case du tableau `livres`.

```
! Initialiser une bibliotheque a vide.
!
! Assure
!        b%nb_livres == 0
SUBROUTINE initialiser_bibliotheque(b)
TYPE(typ_Bibliotheque),INTENT(OUT)::b
    b%nb_livres = 0

END SUBROUTINE initialiser_bibliotheque
```

**1.2.3** Écrire un sous-programme pour afficher le contenu de la bibliothèque.
**Solution :** Ce sous-programme prend un seul paramètre, la bibliothèque (en **in**). Il s'agit donc d'une procédure.

```
Procédure afficher_bibliothèque(b: in Bibliothèque) Est
    -- Afficher les livres de la bibliothèque.
```

Pour l'implémentation, nous choisissons d'utiliser une répétition **Pour** puisque nous connaissons le nombre de livres à afficher.

```
Variable
    i: Entier   -- parcourir les livres
Début
    Pour i <- 1 JusquÀ i = b.nb_livres Faire
        afficher_livres(b.libres[i])
    FinPour
Fin
```

```
! Afficher les livres de la bibliotheque.
SUBROUTINE afficher_bibliotheque(b)
    TYPE(typ_Bibliotheque),INTENT(IN)::b
    INTEGER:: i        ! parcourir les livres

    DO i = 1 , b%nb_livres
        CALL afficher_livre(b%livres(i))
        PRINT*
    ENDDO
END SUBROUTINE afficher_bibliotheque
```

**1.2.4** Écrire un sous-programme pour ajouter un ouvrage dans la bibliothèque.

**Solution :** Ce sous-programme prend en paramètre :

- la bibliothèque dans laquel le livre doit être ajouté (**in out**) ;
- le livre à ajouter (**in**).

Il s'agit donc d'une procédure.

```
Procédure ajouter(b: in out Bibliothèque; l: in Livre) Est
    -- ajouter le livre l dans la bibliothèque b
```

Les livres sont rangés dans la bibliothèque dans l'ordre lexicographique des titres. Il s'agit donc de faire un tri par insertion pour trouver la place de du nouveau livre. Nous optons pour une insertion séquentielle dont voici le premier niveau de raffinage.

```
R1 : Raffinage De « ajouter »
  | Déterminer la position du livre     position: out Entier
  | Décaler les livres compris entre position et b.nb_livres
  | Ranger le nouveau livre
```

Ces étapes correspondant à des algorithmes classiques sur les tableaux, il ne sont pas détaillés en algorithmique.

```
! Ajouter le livre l dans la bibliotheque b.
!
! Necessite
!        b%nb_livres < MAX_LIVRES        -- non pleine
SUBROUTINE ajouter(b,l)
TYPE(typ_Bibliotheque),INTENT(INOUT)::b
TYPE(typ_Livre),INTENT(IN)::l
    INTEGER:: position        ! position où doit être range le livre l
    INTEGER:: i               ! parcourir les livres

    ! Determiner la position du livre
    position = 1
    DO
```

```
14      IF (position >= b%nb_livres .OR. l%titre >= b%livres(position)%titre) EXIT
15          position=position+1
16      END DO
17
18      ! Decaler les livres
19      DO i = b%nb_livres , position,-1
20          b%livres(i+1) = b%livres(i)
21      END DO
22
23      ! Ranger le livre
24      b%livres(position) = l
25      b%nb_livres=b%nb_livres+1
26  END SUBROUTINE ajouter
```

**1.2.5** Écrire un sous-programme pour indiquer combien il y a d'ouvrages d'un genre donné.
**Solution :** Ce sous-programme a pour paramètres :

- la bibliothèque dans laquelle on veut recenser les ouvrages (**in**) ;
- le genre cherché (**in**) ;
- le nombre d'ouvrages du genre cherché (**out**).

Un seul paramètre en sortie, deux en entrée. On peut donc en faire une fonction.

```
1  Fonction nb_livres_genre(b: in Bibliothèque; genre: in Genre): Entier Est
2      -- Nombre de livres de la bibliothèque b qui sont du genre spécifié.
```

Concernant l'implantation, il faut parcourir tous les ouvrages (boucle **Pour**) et comptabiliser ceux qui sont du genre demandé.

```
1  Variable
2      i: Entier   -- parcourir les livres
3  Début
4      Résultat <- 0
5      Pour i <- 1 JusquÀ i = b.nb_livres Faire
6          Si b.livres[i].genre = genre Alors
7              Résultat <- Résultat + 1
8          FinSi
9      FinPour
10 Fin
```

```
1  ! Nombre de livres de la bibliotheque b qui sont du genre specifie.
2  FUNCTION nb_livres_genre(b, genre) RESULT(resultat)
3      TYPE(typ_Bibliotheque),INTENT(IN)::b
4      INTEGER,INTENT(IN)::genre
5      INTEGER:: resultat
6      INTEGER:: i        ! parcourir les livres
7
8      resultat = 0
9      DO i = 0 , b%nb_livres
10         IF (b%livres(i)%genre == genre)  resultat=resultat+1
11     END DO
12 END FUNCTION nb_livres_genre
```

**1.3** *Programme de test.* Écrire un programme de test des sous-programmes écrits ci-dessus.

**Solution :** Voici un programme de test minimal qui ajoute quelques livres dans une bibliothèque et affiche le nombre de livres des différents genres.

```
! Programme de test minimal
END MODULE Bibliotheque_mod
PROGRAM main
    USE Bibliotheque_mod
    TYPE(typ_Bibliotheque):: b
    TYPE(typ_Livre):: l

    CALL initialiser_bibliotheque(b)
    CALL afficher_bibliotheque(b)

    PRINT*,"Ajouter le livre :"
    CALL initialiser_livre(l, "FFF", 54, BD)
    CALL afficher_livre(l)
    PRINT*,"La bibliotheque contient :"
    CALL ajouter(b, l)
    CALL afficher_bibliotheque(b)

    PRINT*,"Ajouter le livre :"
    CALL initialiser_livre(l, "XXX", 120, ART)
    CALL afficher_livre(l)
    PRINT*,"La bibliotheque contient :"
    CALL ajouter(b, l)
    CALL afficher_bibliotheque(b)

    PRINT*,"Ajouter le livre :"
    CALL initialiser_livre(l, "CCC", 540, TECHNIQUE)
    CALL afficher_livre(l)
    PRINT*,"La bibliotheque contient :"
    CALL ajouter(b, l)
    CALL afficher_bibliotheque(b)

    PRINT*,"Ajouter le livre :"
    CALL initialiser_livre(l, "DDD", 140, TECHNIQUE)
    CALL afficher_livre(l)
    PRINT*,"La bibliotheque contient :"
    CALL ajouter(b, l)
    CALL afficher_bibliotheque(b)

    PRINT*,"Nb livres ART :", nb_livres_genre(b, ART)
    PRINT*,"Nb livres BD  :", nb_livres_genre(b, BD)
    PRINT*,"Nb livres ROMAN  :", nb_livres_genre(b, ROMAN)
    PRINT*,"Nb livres TECHNIQUE  :", nb_livres_genre(b, TECHNIQUE)

END PROGRAM main
```