Chapitre 4 : Organisation des données à l'exécution

Nous nous intéressons dans ce cours à l'organisation des données, aux paramètres, au mode de leur passation et à la structure du programme.

Différents types d'organisation existent :

- Statique.
- Dynamique.
- En liste
- Contrôlé.
- Paramétré.

A/ Organisation statique

Dans l'organisation statique, toutes les données nécessaires à l'exécution du programme sont présentes en mémoire. L'ensemble de ces données est de taille constante. Cette taille peut être calculée lors de l'allocation. Ce type d'organisation des données est celui du langage FORTRAN, HTML, …

Caractéristiques du langage FORTRAN

Ce langage est destiné aux calculs scientifiques (Problèmes d'analyse numérique,…). Ils possèdent les caractéristiques suivantes :

- Déclaration implicite des variables.
- Pas d'objets à structure dynamique : les tableaux sont statiques, leurs bornes sont connues à la compilation.
- Les sous programmes et les fonctions (SUBROUTINE et FUNCTION), ne sont pas récursifs, donc pas de notion de récursivité en FORTRAN.
- Possibilité d'attribuer plusieurs noms à une zone mémoire en utilisant : EQUIVALENCE. *Exemple* : EQUIVALENCE A, B, C ➔ @A =@B=@C
- Possibilité de déclarer des zones de données communes partageables entre programme principal et sous programme (SP). En utilisant COMMON.
  Exemple **PP** COMMON A, B, C, D. **SP** COMMON E, F, G ➔@A=@E et @B=@F,…
- Un SP peut accéder, en plus de ses variables locales aux variables déclarées dans les COMMON et aux paramètres effectifs.
- Un programme FORTRAN a la structure suivante :

```
PP
…
STOP
END

SP1
…
RETURN
END
```

L'appel du PP d'un SP se fait à l'aide de l'instruction CALL (voir ci-dessous).

Définition d'une zone de données

Une zone de données ZD est un bloc de mémoire dans lequel on stocke les valeurs des objets. En FORTRAN, On connait dès la compilation les adresses relatives de toutes les variables du programme et de la dimension de la zone des données. Il existe une ZD pour le PP, pour chaque procédure et pour chaque bloc COMMON. L'adresse de la zone de donnée active (PP ou SP) est stockée dans un registre appelé ACTIVAREA.

Zone de donnée du PP

Dans cette zone, on retrouve les variables du PP non déclarées dans un bloc COMMON, ainsi que les temporaires.

Zone de donnée du SP

Elle est organisée comme suit :

a) Paramètres implicites
   Ce sont des paramètres définissant le contexte à sauvegarder :
   -Adresse de retour dans le programme appelant.
   - Adresse de la zone de donnée du PP.
   -Sauvegarde u contenu de certains registres.

Paramètres effectifs
Paramètres implicites
Variables locales / temporaires

ZD du SP

b) Paramètres effectifs
   Ce sont des paramètres de passage (par valeur, par référence,…) au SP.

c) Variables simples, locales et temporaires.

Appel de SP

Un appel de SP consiste à exécuter le SP en utilisant les paramètres effectifs spécifiés dans l'appel. A la fin de l'exécution du SP, il est nécessaire de se tourner au programme appelant à l'instruction qui suit l'appel.

Un appel ***CALL id (<liste des paramètres effectifs>)*** se traduit par :

- Calcul des paramètres effectifs et stockage de leur valeur (appel par valeur) ou adresse (appel par référence) dans une zone conventionnelle globale utilisée par tous les SP pour réaliser le passage de paramètres.
- Saut au début du code du SP après avoir sauvegardé l'adresse de retour dans un registre.

Le SP est définit par ***SUBROUTINE id (<listes des paramètres formels >)*** se traduit par :

- Recopie des paramètres implicites dans la ZD (@ de retour, @ ZD du programme appelant,….).
- Recopie des paramètres effectifs.

RETURN se traduit par :

- Chargement des paramètres implicites.
- Saut à l'adresse de retour.

EQUIVALENCE et COMMON

Ces instructions sont des ordres non exécutables qui régissent l'implantation des variables en mémoire et permettant le partage d'une même zone par plusieurs programmes et sous programmes.

Cas du COMMON

L'utilisation essentielle de cet ordre est de permettre le partage de certaines variables par un ensemble « programme principal-SP » destiné à être utilisé en même temps.

Les variables listées dans le COMMON sont implantées en mémoire dans l'ordre de leurs apparitions dans la liste. Donc pour les variables déclarées dans le COMMON, on procède à une allocation séquentielle. Cet ordre indique que l'on doit trouver dans le bloc de données désigné par le nom COMMON, les objets dans l'ordre dans lequel ils apparaissent.

Exemple

COMMON |Compil1| A(5,5),C B(10)

Remarques

-La taille d'un bloc COMMON est déterminée après avoir compilé le PP et tous les SP et fonctions.
-L'attribution des adresses pour les variables qui apparaissent dans le COMMON se fait de la manière suivante :
COMMON V1, V2,…., Vn
@V1=@début du COMMON
@V2=@V1+taille (V1)
…
@Vi=@Vi-1+taille (Vi-1)

Cas d'EQUIVALENCE

Le but de cet ordre est d'assigner la même adresse en mémoire à chacune des variables situées dans la même liste.

Exemple

1-EQUIVALENCE (A, B(1,2), C). A , B, C sont des appellations équivalentes de la même variable
2- DIMENSION C(10), R(3,4)
EQUIVALENCE (A, C(4), R(2,3))
A, C(4) et R(2,3) sont des appellations d'une même case mémoire.

Les variables apparaissant dans cet ordre ne sont pas forcément du même type et ne sont pas égales.

L'EQUIVALENCE présente un intérêt, elle permet de désigner une même variable sous plusieurs noms différents, pouvant être de types différents.

Cet ordre peut également être utilisé pour économiser de l'espace mémoire en affectant des noms différents aux mêmes cases mémoires, dans la mesure où les différentes variables sont utilisées dans des parties différentes du programme. Supposons par exemple que dans un programme, nous calculons la moyenne de n nombres NMOY, le PGCD de ces nombres NPGCD et le plus petit nombre dans une liste NMi. Si ces trois variables ne sont jamais utilisés en même temps, il est possible de les mettre en EQUIVALENCE, et de les adresser à la même case mémoire. Ceci est d'autant plus intéressant lorsque l'on travaille avec les tableaux.

A/ Organisation dynamique

Dans cette organisation, les objets sont créés et supprimés en fonction des besoins du programme. De façon précise, à une unité particulière du programme (bloc ou procédure) est associée un ensemble de données. Ainsi, lorsque l'on débute l'exécution du bloc ou de la procédure, on dispose de l'ensemble de données nécessaire à l'exécution de cette unité de programme. Lorsque l'exécution du bloc ou de la procédure est achevée, on libère le place mémoire occupée par ces données devenues inutiles.

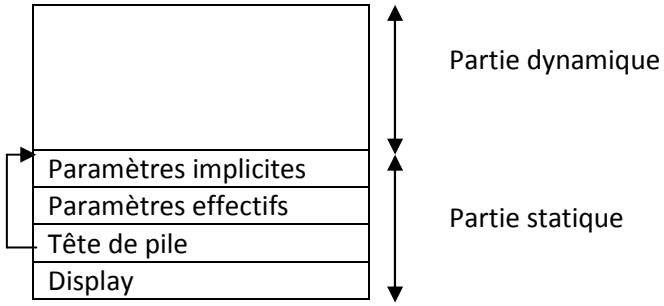
Dans ce type d'organisation de données, la zone de mémoire utilisée est gérée comme une ***pile***, c'est le cas des majorités des langages actuels (exemple : langage ALGOL).

Format d'une ZD d'une procédure

La ZD est constituée d'une partie statique dont la taille est déterminée à la compilation et d'une ZD dynamique construite à l'exécution.

ZD statique : elle est composée de :

- *Display* : contient les adresses des ZD auxquels la procédure peut accéder.
- *Tête de pile (stack top)* : est un pointeur vers le dernier mot de la zone statique de la procédure.
- *Paramètres implicites :* servent à sauvegarder le contexte de la procédure appelante (l'adresse de retour, sommet de pile avant l'allocation, …)
- Paramètres effectifs : zone réservée pour les valeurs et les adresses des paramètres de passage.
- Pour chacun des blocs de la procédure, une variable *tête de pile* qui contiendra l'adresse du dernier mot de la ZD libre.
- Les variables simples et les temporaires de ce bloc.



ZD d'une procédure

A l'exécution, un certain nombre de procédures peuvent figurer dans l'état d'activation du programme : ces procédures ont été appelées mais leur exécution n'est pas terminée. De toutes ces procédures, une seule est en cours d'exécution à un instant donné, c'est la procédure « active ».

Soit i, le niveau d'imbrication de la procédure active, le niveau d'imbrication du programme principal est 0 implicitement. La procédure active i peut accéder aux variables associées aux niveaux 0, 1, 2,…, i-1 et i. le display actif (le display associé à la procédure active).

- Une procédure peut définir une autre.
- Une procédure de niveau i, peut accéder aux variables déclarées dans les procédures de niveau inférieur.
- Deux procédures de même niveau ne peuvent s'appeler, exemple : A ne peut accéder aux variables de B et inversement.
  PP(niveau 0) A (niveau 1) B (niveau 1) .

@ZD de i
@ZDde 1
@ZD dePP

ZD de procédure i

Dans certaines réalisations, on utilise deux piles distinctes l'une pour les displays et l'autre pour les données.

Avant l'exécution d'un bloc (à la compilation), on alloue de l'espace aux variables et aux vecteurs de renseignements, des tableaux dans le bloc, l'espace est alors libérer dés que l'on sort du bloc.

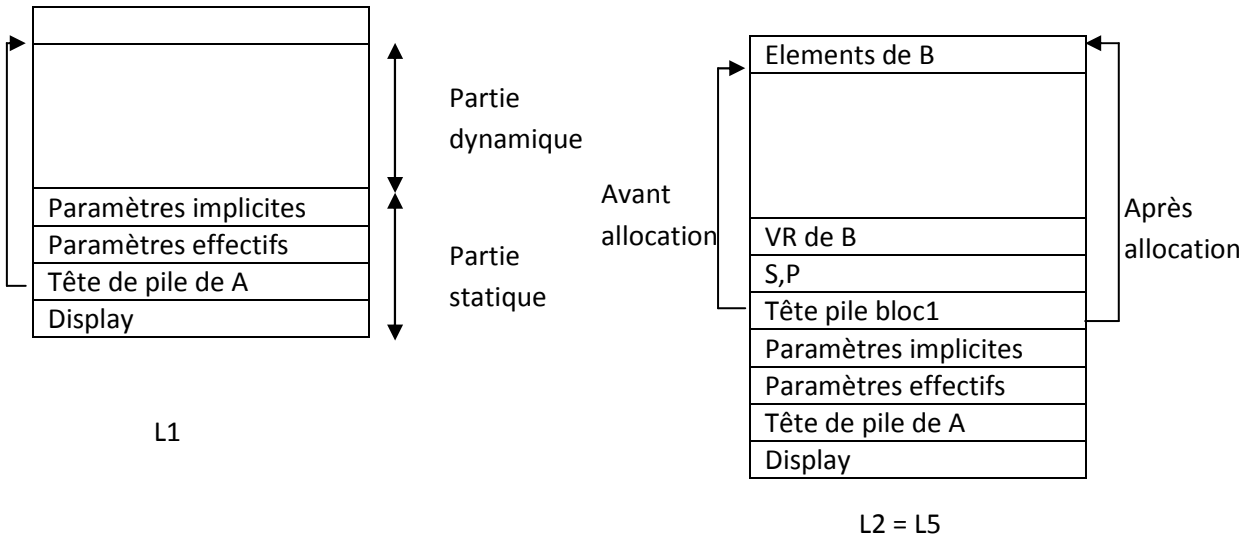
ZD d'un PP elle contient :

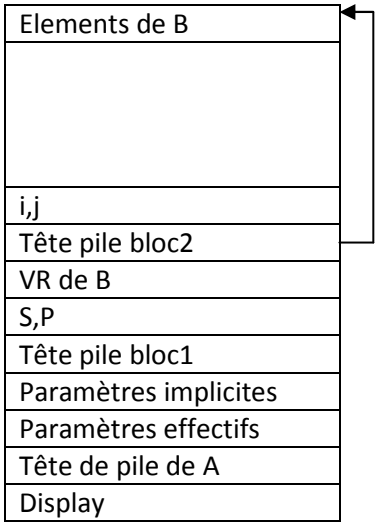
- Display qui contient l'@ du PP.
- Tête de pile du PP.
- Variables.
- Vecteurs de renseignements.
- Temporaires de PP.

Pour chaque bloc du PP, mêmes opérations que les procédures.

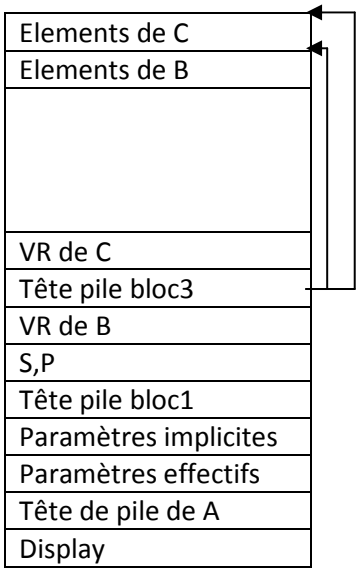
Exemple

```
Procedure A(x,y) :integer x,y;
        L1: begin
              Real S,P; Array B[x:y];
              L2: begin integer i,j;
                2  L3: For i:=x to y step 1
        1             Do s:=s+B[i];
                 L4: End;
              L5: begin array C [1:x];
                  L6 : begin real E ;
                3    4      L7 :
                       End ;
                  End ;
            End ;
```

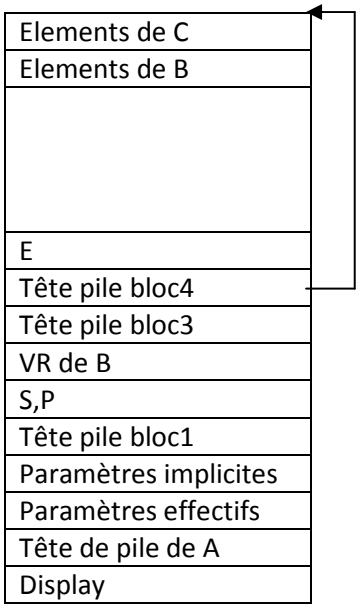




L3 = L4



L6



L7

<u>Calcul d'adresses des variables</u>

Pour calculer les adresses absolues des variables, on utilise une zone mémoire appelée ActiveArea qui pointe sur le début de la ZD active.

ActiveArea : identifie le display actif et la ZD associée à la procédure active (pointe vers le début de la ZD active).

1. <u>Adresse des variables déclarées dans le PP</u>
   Adresse absolue := ActiveArea + déplacement de la variable du début de la zone du PP.

2. Variables déclarées dans une procédure
   Adresse absolue := ActiveArea + déplacement de la variable du début de la zone de SP.

3. Variables non locales déclarées dans une procédure de niveau k, référencée par une procédure de niveau i :
   Adresse absolue := Contenu (ActiveArea+k) + déplacement de la variable du début de la zone de la procédure k.