Université Assane SECK de Ziguinchor


Unité de Formation et de Recherche des Sciences et Technologies

Département d'Informatique

# Pointeurs

Licence 1 en Ingénierie Informatique

Octobre 2021



**Papa-alioune.cisse@univ-zig.sn**

**Résumé :** La mémoire centrale d'un ordinateur est composée d'un très grand nombre d'octets (*cellules mémoires*) pour contenir les données à stocker. Chacune de ces cellules est numérotée par ce qui est appelé « une adresse mémoire ». Un pointeur est une variable qui prend comme valeurs les « adresse » d'une mémoire, c'est-à-dire, un pointeur permet de référencer une « cellule mémoire ». Dans ce sens, l'utilisation des pointeurs dans un programme permet d'optimiser la mémoire centrale et permet aussi une meilleur organisation et gestion des données d'un programme.

# 1 - Notion d'adresse

Toute variable possède trois caractéristiques : un ***nom*** et un ***type***, qui ne peuvent être modifiés au cours du programme, car fixés au moment de la définition de la variable ; et une ***valeur*** qui au contraire évolue au cours du programme. En réalité, toute variable possède également une autre caractéristique fondamentale, utilisée en interne par la machine : une ***adresse***. En effet, afin de mémoriser la valeur d'une variable, l'ordinateur doit la stocker au sein de sa mémoire, mémoire dont les éléments ou cellules sont repérés par des numéros (de manière analogue à ce qui se passe dans un tableau en réalité). Le nom de la variable n'est en fait pas utile à la machine, qui se contente de son adresse ; c'est pour le confort du programmeur que le choix du nom est rendu possible. Ainsi, il existe une dualité entre le nom de la variable (côté programmeur) et son adresse (côté machine). L'adresse d'une variable n'est autre que le numéro de la case ou cellule mémoire que celle-ci occupe au sein de la machine. Au même titre que son nom, l'adresse d'une variable ne peut pas varier au cours d'un programme. Il n'est même pas possible de choisir l'adresse d'une variable, cette adresse est attribuée automatiquement par l'ordinateur. Chose curieuse, ces adresses de variables seront manipulées sans même connaître leur valeur exacte. Il suffira de savoir les nommer pour les utiliser. Pour ce faire, un nouvel opérateur est nécessaire : l'opérateur &. Il se lit tout simplement « adresse de » et précède uniquement un nom de variable.

**Exemple**. Soit la définition de variable suivante : ***VAR x : réel***. Cette simple définition attribue à la variable : un nom (x), un type (réel), une valeur (inconnue pour le moment), et également une adresse (de manière automatique). La notation ***&x*** signifie donc : « adresse de x ». La valeur précise de cette adresse ne nous est en réalité d'aucun intérêt, mais il sera parfois nécessaire de la manipuler.

La mémoire centrale d'un ordinateur est composée d'un très grand nombre d'octets. Chaque octet est repéré par un numéro appelé adresse de l'octet.

| adresses | bits | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| octet 0 | | | | | | | | | NULL |
| octet 1 | | | | | | | | | |
| octet 2 | | | | | | | | | |
| etc. | | | | | | | | | |
| octet $2^n - 2$ | | | | | | | | | |
| octet $2^n - 1$ | | | | | | | | | |

Chaque variable dans la mémoire occupe des octets contigus, c'est-à-dire des octets qui se suivent. Par exemple, un réel occupe 4 octets qui se suivent. L'adresse de la variable est l'adresse de son premier octet.

# 2 - Variables de type Pointeur

## 2.1 - Définition de pointeur

L'adresse d'une variable peut elle-même être mémorisée dans une variable. Les variables dont les valeurs sont des adresses s'appellent des pointeurs. Un pointeur est donc une variable qui permet de stocker l'adresse mémoire d'une autre variable.

Un pointeur est donc une variable un peu particulière, car elle ne stocke ni un entier, ni un réel, ni un caractère, mais une adresse. Il est tentant de penser qu'une adresse étant un numéro de cellule dans la mémoire, elle est comparable à un entier. Cependant, une adresse ne s'utilise pas comme un entier, puisqu'une adresse ne peut pas être négative, et ne sert pas à faire des calculs. Une adresse est donc en ce sens un nouveau type.

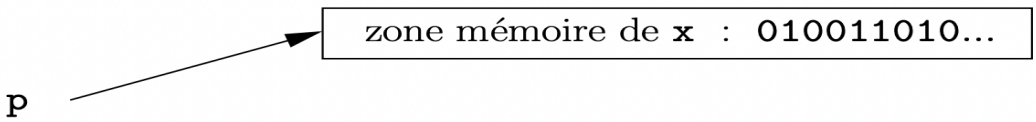
## 2.2 - Notion de contenu d'un pointeur

Il faut noter qu'un pointeur est aussi une variable et donc possède un nom, une valeur (qui est une adresse) et un type. Un pointeur possède en plus une autre caractéristique, qui est son contenu. Une adresse est le numéro d'une cellule mémoire, et dans cette cellule, se trouve une valeur. Cette valeur est appelée le contenu du pointeur, et ne doit pas être confondue avec sa valeur. Le contenu d'un pointeur est la valeur de la cellule mémoire dont ce pointeur stocke l'adresse. Puisque ces deux notions sont différentes, il existe une nouvelle notation pour indiquer l'accès à un contenu : cette notion est l'opérateur ^ (lire chapeau). Voici une règle très simple et très utile pour l'utilisation de cet opérateur ^: il se lit toujours comme « contenu de », et non pas « pointeur ». Cette règle évitera de nombreuses confusions par la suite.

Cela veut dire qu'on peut donc accéder à une variable de 2 façons :

- grâce à son nom
- grâce à l'adresse du premier bloc alloué à la variable, c'est-à-dire un pointeur vers cette adresse.

Voici un exemple utilisant un pointeur p qui prend pour valeur l'adresse de x (on dit que p pointe sur x).




Un autre exemple : Soit p un pointeur. Supposons que la valeur de ce pointeur soit 25040 (rappelons que cette valeur est une adresse mémoire et est tout à fait arbitraire). Supposons également que dans la mémoire, à l'emplacement 25040 se trouve la valeur 17. Dans ce cas, la valeur de p, notée p, est 25040. Le contenu de p, noté p^, est la valeur de la cellule mémoire numéro 25040, c'est-à-dire 17. P^ vaut donc 17.

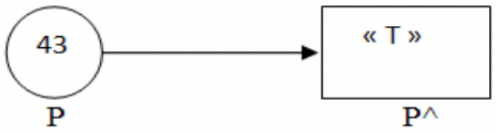
### 2.3 - Intérêt des pointeurs

Les pointeurs ont un grand nombre d'intérêts :

- Ils permettent de manipuler de façon simple des données importantes (au lieu de passer à une fonction un élément très grand en taille, on pourra par exemple lui fournir un pointeur vers cet élément).
- Les tableaux ne permettent de stocker qu'un nombre fixé d'éléments de même type. En stockant des pointeurs dans les cases d'un tableau, il sera possible de stocker des éléments de taille diverse, et même de rajouter des éléments au tableau en cours d'utilisation (c'est la notion de tableau dynamique qui est très étroitement liée à celle de pointeur).
- Il est possible de créer des structures chaînées.

## 3 - Déclaration d'un pointeur

Le type de la variable pointée est appelé type de base. La déclaration d'une variable de type pointeur a pour effet la réservation d'une case mémoire qui va contenir l'adresse de la variable pointée. Exemple :



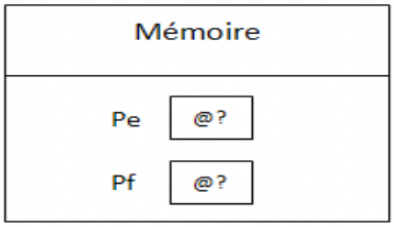
Le pointeur P a pour valeur l'adresse « 43 ». Il pointe sur l'espace mémoire à l'adresse 43 dont le contenu (désigné par « P^ » est le caractère « T ». On déclare un pointeur en précédant son type de base par le caractère « ^ ». Par exemple pour déclarer un pointeur appelé P sur une variable de type caractère on écrit : P : ^caractère.

### 3.1 - Pointeur vers un seul élément

La déclaration d'une variable de type pointeur a pour effet la réservation d'une case mémoire qui va contenir l'adresse de la variable pointée. Par exemple si on déclare 2 pointeurs Pe qui pointe vers une variable de type entier et Pf qui pointe vers une variable de type réel, cela implique la création de deux variables qui contiennent respectivement l'adresse d'un entier et l'adresse d'un réel comme suit :

Pe : ^entier

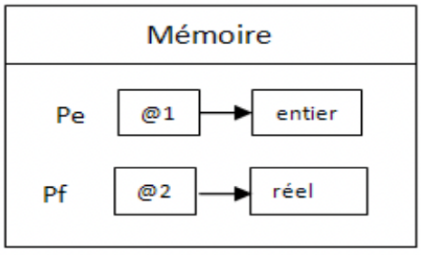
Pf : ^reel




Pour utiliser un pointeur, il faut d'abord :

- Soit, lui donner un espace mémoire déjà identifié avec un autre pointeur,
- Soit, lui donner l'adresse mémoire d'une variable déjà déclaré,
- Soit, lui réserver un bloc en mémoire capable de stocker une valeur de son type de base : on appelle cela « allocation mémoire ». L'instruction permettant une allocation mémoire pour un pointeur P est : « ***Allouer (P)*** ».

Par exemple pour l'allocation mémoire de Pe, on écrit : ***Allouer (Pe)***, et pour celle de Pf, on écrit : ***Allouer (Pf)***.




L'instruction ***Allouer (P)*** :

- Réserve un bloc mémoire de la taille adéquate pour contenir la valeur de la variable pointée par P.
- Récupère l'adresse de ce bloc et la met dans la variable P.

### 3.2 - Pointeur vers un tableau d'éléments

On utilise l'instruction ***Allouer (T, N)*** pour créer un tableau dynamique de N cases. T doit être déclaré comme un pointeur sur le type des éléments du tableau. Dans le cas particulier où les éléments du tableau sont des caractères on dit qu'on a fait une allocation dynamique pour une chaîne de caractères.

L'instruction Allouer (T, N) :

- Réserve un bloc mémoire pour contenir un tableau de N cases.
- Récupère l'adresse de la 1ère case de ce tableau et la met dans la variable T

Exemple :

```
Algorithme Pointeur_tab
            Var T : ^Entier
            Début
              Allouer (T, 5) {tableau de 5 entiers}
            Fin
```

## 4 - UTILISATION D'UN POINTEUR

### 4.1 - Accès à une variable pointée

Pour accéder à la variable pointée par un pointeur appelé P, on utilise l'opérateur ^. Donc « P^ » désigne la variable pointée par P. Exemple : l'algorithme suivant stocke la valeur 10 dans l'espace pointé par le pointeur P.

```
Algorithme exemple
            Var P : ^Entier
            Début
              Allouer (P)
              P^ ← 10
            Fin
```

### 4.2 - Libération de l'espace mémoire d'un pointeur

Lorsqu'un pointeur n'a plus d'utilité, il est possible de le supprimer et de rendre disponible l'espace mémoire qu'il occupe. L'instruction qui réalise cette tâche est « Libérer ». Si on a déclaré un pointeur appelé P, l'instruction : « Libérer (P) » supprime la variable pointé par P et libère l'espace mémoire occupé par cette variable. Par contre la variable pointeur « P » n'est pas supprimée mais son contenu n'est plus l'adresse de la variable pointé mais une adresse non significative. Exemple :l'algorithme suivant montre l'effet de l'instruction « libérer » sur la mémoire.

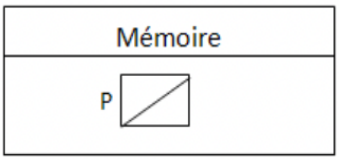
```
Algorithme exemple_libérer
            Var P : ^Entier
            Début
              Allouer (P)
              P^ ← 10
              Libérer (P)
              P^← 4 {Erreur}
            Fin
```

### 4.3 - Affectation d'une valeur à pointeur

Il y'a 3 façons d'affecter une valeur à un pointeur :

➢ **Initialisation avec la constante « Nil »**

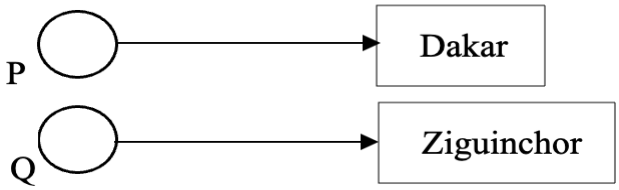
Pour initialiser un pointeur on peut lui affecte une valeur constante appelé « ***Nil*** », cette constante n'est pas une valeur d'une adresse mémoire, mais elle signifie que le pointeur ne pointe nulle part (sur rien). Ainsi, si on déclare un pointeur P vers un entier il sera initialisé comme suit : ***P ← Nil***. Et on le représente par le schéma suivant :



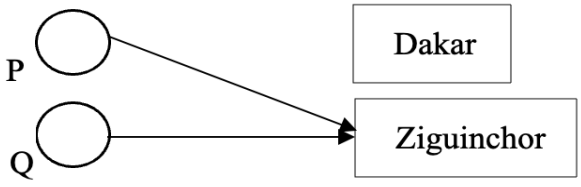

➢ **Affecter une valeur à un pointeur à partir d'un autre pointeur**

Une valeur peut être affectée à un pointeur à partir d'un autre pointeur comme le montre l'exemple suivant :

Supposons qu'on soit dans la situation initiale illustrée par la figure suivante :




En faisant l'opération P ← Q, on se retrouve dans la situation suivante où l'espace contenant « Dakar » n'est plus pointé par P qui désormais pointe vers le même espace que Q, c'est-à-dire, vers Ziguinchor :




➢ **Affectation avec l'opérateur d'adresse « & »**

L'opérateur d'adresse (noté « & ») permet de récupérer l'adresse mémoire d'une variable. Comme un pointeur contient l'adresse d'une variable, on peut utiliser l'opérateur d'adresse pour affecter à un pointeur l'adresse d'une autre variable. La syntaxe est la suivante :

« pointeur ← & variable ».

Exemple :

```
Algorithme Adresse
            Var P : ^Entier
                A : Entier
            Début
              Allouer (P)
              A ← 10
              P ← &A
              Écrire (A) {10}
              Écrire (P^) {10}
              P^← 4
              Écrire (A) {4}
              Écrire (P^) {4}
              Libérer (P)
              Écrire (A) {4}
              Écrire (P^) {Erreur}
            Fin
```

### 4.4 - Comparaison de pointeurs

On peut comparer 2 pointeurs entre eux avec l'opérateur = ou <> à condition qu'ils aient le même type de base. Et on peut comparer la valeur Nil à n'importe quel pointeur quel que soit son type de base. On peut aussi comparer les valeurs des variables pointées par 2 pointeurs.