

Chapitre 3: Les Pointeurs

I. Introduction

Jusqu'ici, nous avons utilisé des variables dites « **statiques** ».

Une **variable statique** est caractérisée par les propriétés suivantes :

- elle est déclarée en tête du bloc où elle est utilisée
- elle occupe un espace mémoire dont la taille est fixée dès le début pour qu'on y place ses valeurs
- l'accès à la valeur se fait par le nom de la variable.

Au contraire, une **variable dynamique** est caractérisée par les propriétés suivantes :

- elle peut être créée et détruite au cours de l'exécution du bloc dans lequel elle est déclarée
- l'espace mémoire rendu libre peut être récupéré
- l'accès à la valeur se fait par un **pointeur**.

II. Pointeurs

Un **pointeur** **P** est une variable statique dont les valeurs sont des adresses.

Une variable dynamique pointée par **P** sera notée **P[^]**. Cette variable dynamique est appelée **le contenu de la variable pointée par le pointeur P**

Selon le type de donnée contenu à l'adresse en question, on aura un **pointeur d'entier, de réel, de caractère**, ou de tout autre type. En accédant à cette adresse, on peut accéder indirectement à la variable et donc la modifier.

II.1 Déclaration de pointeur

Pour déclarer un variable pointeur on utilise l'opérateur **^** placé avant le type du pointeur.

Un pointeur sur entier est déclaré par **p : ^ entier**

Un pointeur sur réel **p : ^ réel**

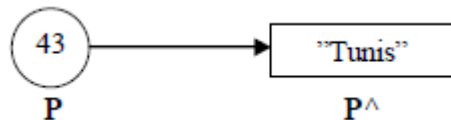
Un pointeur sur caractère **p : ^ caractère**

Pointeur sur pointeur de type double **tab : ^^ double**

La variable **tab** est un pointeur pointant sur un pointeur qui pointe sur un flottant double !

On peut déclarer un type de pointeur.

Exemple :

<p>Algorithme Pteur</p> <p>Types</p> <p>Pointeur = ^Chaîne</p> <p>Variables</p> <p>P : Pointeur</p> <p>Début</p> <p>Allouer(P) (* création du pointeur P *)</p> <p>P[^] ← "Tunis"</p> <p>Fin.</p>	 <pre> graph LR P((43)) --> P^["Tunis"] subgraph Labels P_label[P] P_label_car[P^] end </pre>
---	--

La variable pointeur **P** a pour valeur **43**. Elle pointe sur l'espace mémoire **P[^]** d'adresse **43** et dont le contenu est la chaîne **"Tunis"**.

II.2 Opérations sur les pointeurs

Il ne faut pas confondre les opérations sur les pointeurs avec les opérations sur les variables pointées

Algorithme Pteur Types Pointeur = ^Chaîne Variables P : Pointeur Q : Pointeur Début	
Allouer(P) (* création du pointeur P *) $P^{\wedge} \leftarrow \text{"Tunis"}$ (*la variable pointée par P reçoit la valeur "Tunis" *) Allouer(Q) (* création du pointeur Q *) $Q^{\wedge} \leftarrow \text{"Sousse"}$ (*la variable pointée par Q reçoit la valeur "Sousse" *)	<pre> graph LR P((30 P)) --> P_ptr[30 P^] P_ptr --> Tunis[Tunis] Q((55 Q)) --> Q_ptr[55 Q^] Q_ptr --> Sousse[Sousse] </pre>
$P^{\wedge} \leftarrow Q^{\wedge}$ (* le contenu de la variable pointée par P (noté P [^]) reçoit le contenu de la variable pointée par Q *)	<pre> graph LR P((30 P)) --> P_ptr[30 P^] P_ptr --> Sousse2[Sousse] Q((55 Q)) --> Q_ptr[55 Q^] Q_ptr --> Sousse[Sousse] </pre>
$Q^{\wedge} \leftarrow \text{"Gabes"}$ (* le contenu de la variable pointée par Q reçoit la chaîne "Gabes" *) (*Attention : le (* le contenu de la variable pointée par P ne change pas*)	<pre> graph LR P((30 P)) --> P_ptr[30 P^] P_ptr --> Sousse[Sousse] Q((55 Q)) --> Q_ptr[55 Q^] Q_ptr --> Gabes[Gabes] </pre>
$P \leftarrow Q$ (* P sera égale à Q donc *) (* P et Q pointent sur la même @ mémoire *) (*la variable pointée par P est celle pointée par Q*)	<pre> graph LR P((55 P)) --> P_ptr[55 P^] Q((55 Q)) --> Q_ptr[55 Q^] P_ptr --> Gabes[Gabes] Q_ptr --> Gabes[Gabes] </pre>
$P^{\wedge} \leftarrow \text{"Sfax"}$ (*équivalent à $Q^{\wedge} \leftarrow \text{"Sfax"}$ *) (* le contenu de la variable pointée par Q (noté Q [^]) sera Sfax car P et Q pointent sur la même variable *)	<pre> graph LR P((55 P)) --> P_ptr[55 P^] Q((55 Q)) --> Q_ptr[55 Q^] P_ptr --> Sfax[Sfax] Q_ptr --> Sfax[Sfax] </pre>

III. Pointeurs et arguments de fonctions

En raison de l'appel par valeur, une fonction ne peut pas modifier ses arguments. L'utilisation de pointeurs permet de tourner la difficulté.

Soit par exemple la fonction `echange (x,y)` qui est **censée** échanger ses arguments :

Procédure `echange (x : entier, y :entier) /* INCORRECT */`

Var

aux :entier

debut

aux \leftarrow y

y \leftarrow x

x \leftarrow aux

fin proc

Programme Exemple1

Var

i :entier

j :entier

début

i \leftarrow 3

j \leftarrow 4

echange(i,j)

écrire ("i=",i) /*affichei= 3*/

écrire ("j=",j)/*affiche j=4*/

Fin

L'appel de `echange (i, j)` avec i et j entiers n'aura aucun effet car ces paramètres sont passés par valeur et ne sont pas modifiés en dehors de `echange ()`.

Procédure `echange (x : ^entier, y :^entier) /* CORRECT */`

Var

aux :entier

début

aux \leftarrow y[^]

y[^] \leftarrow x[^]

x[^] \leftarrow aux

fin proc

Programme Exemple2

Var i :^entier

j :^entier

début

Allouer(i) , Allouer(j)

i[^] \leftarrow 3

j[^] \leftarrow 4

echange (i,j)

écrire ("i=",i[^]) /*affichei= 4*/

écrire ("j=",j[^])/*affiche j=3*/

Fin

Maintenant l'appel de `echange (i, j)` modifie les variables i[^] et j[^] car les adresses des paramètres sont passées à la fonction (on parlera de passage par VARIABLE ou par ADRESSE).