LEÇON 3 : LES POINTEURS

Table des matières

1- Généralité sur les pointeurs

I

1.1- Notion de mémoire

La mémoire centrale utilisée par les programmes, est découpée en octets. Chacun de ces octets est identifié par un numéro séquentiel appelé adresse. Par convention, une adresse est notée en hexadécimal et précédée par 0x.

Déclarer une variable, c'est attribuer un nom à une zone de la mémoire centrale qui se trouve à une adresse bien donnée et la taille que la variable doit occuper.

1.2- Notion de pointeur

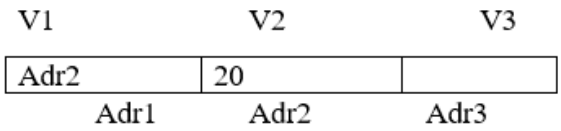
Un *pointeur* est une variable qui contient l'adresse d'une autre variable. L'adresse contenue dans un pointeur est celle d'une variable qu'on appelle *variable pointée*.

On dit que le pointeur pointe sur la variable dont il contient l'adresse. Un pointeur est associé à un type de variable sur lequel il peut pointer.

Remarque :

- Un pointeur sur entier ne peut pointer que sur des variables entières, pareille pour les autres types de données.
- Un pointeur est lui-même une variable qui possède une adresse.
- Il ne faut pas confondre l'adresse de la variable pointeur et l'adresse contenue dans le pointeur.

Exemple :



Explication :

• V2 est placé à l'adresse Adr2

• V1 contient l'adresse de la variable V2 alors on dit V1 pointe sur V2

Application 1 :

II

Exercice

Comment appelle t-on chaque numéro qui identifie un espace en mémoire ?

- Pointeur
- Variable pointée
- Adresse

Exercice

Un pointeur est :

- Un espace qui conserve une valeur
- Un espace mémoire qui conserve une adresse
- Un espace mémoire qui conserve une valeur

Exercice

Un pointeur pointe sur :

- une valeur en mémoire
- une variable entière
- la variable dont il contient l'adresse

2- Traitement sur les pointeurs

III

Syntaxe : 2.1- Déclaration d'un pointeur

variablepointeur : pointeur sur type ou *variablepointeur : ^type*

Exemple :

Nbre : pointeur sur entier ou

Nbre : ^entier

Syntaxe : 2.2- Allocation dynamique d'une zone de mémoire

Lorsque l'on déclare un pointeur Nbre : ^entier, le système réserve un entier pour stocker l'adresse vers laquelle pointe Nbre , il ne réserve pas de place pour la variable pointée par Nbre . Le fait de réserver un espace-mémoire pour stocker la variable pointée par Nbre s'appelle allocation dynamique.

Syntaxe :

allouer(variablepointeur)

NB :

Pour accéder à la valeur d'une variable pointée il faut : *variable*^

Exemple :

Nbre : ^entier *//déclaration du pointeur nbre*

allouer(Nbre) *// réserver un espace-mémoire pour stocker la variable pointée par Nbre*

Nbre^ ← 350 *//Affecter 350 à la variable pointée*

Afficher Nbre^ *//accéder à la valeur de la variable pointée c'est à dire 350*

Syntaxe : 2.3- Désallocation d'une zone de mémoire

Elle permet la libération de la place occupée par une variable et laisse la valeur du pointeur en l'état (n'efface pas l'adresse qui est dans la variable pointeur).

Syntaxe :

desallouer(variablepointeur)

Remarque :

Si on fait appel au pointeur désalloué, il renvoie une information qui n'a aucun sens.

Exemple :

Nbre : ^entier *//déclaration du pointeur nbre*

allouer(Nbre) *// réserver un espace-mémoire pour stocker la variable pointée par Nbre*

Nbre^ ← 350 *//Affecter 350 à la variable pointée*

Afficher Nbre^ *//accéder à la valeur de la variable pointée c'est à dire 350*

desallouer(Nbre) *//on libère la zone mémoire*

2.4- Affectation des pointeurs

Les pointeurs sont des variables particulières, puisque leurs valeurs sont des adresses mémoires. Ils peuvent néanmoins être impliqués dans des affectations au cours desquelles des adresses sont assignées aux pointeurs.

- Pour vider un pointeur, c'est à dire annuler l'adresse qu'il contient, on lui affecte une valeur prédéfinie nommée Nil.

Syntaxe :

variablepointeur ← Nil

Exemple :

Nbre : ^entier *//déclaration du pointeur nbre*

allouer(Nbre) *// réserver un espace-mémoire pour stocker la variable pointée par Nbre*

Nbre^ ← 350 *//Affecter 350 à la variable pointée*

Afficher Nbre^ *//accéder à la valeur de la variable pointée c'est à dire 350*

desallouer(Nbre) *//on libère la zone mémoire*

Nbre ← Nil *//efface l'adresse mémoire qui n'a plus d'existence physique après avoir désalloue*

- On peut affecter un pointeur avec tout autre pointeur de même type. Après cette affectation, les deux pointeurs désigneront la même zone de mémoire

Syntaxe :

variablepointeur1 ← variablepointeur2

Exemple :

Nbre1,Nbre2 : ^entier *//déclaration du pointeur nbre*

allouer(Nbre1) *// réserver un espace-mémoire pour stocker la variable pointée par Nbre*

Nbre^ ← 350 *//Affecter 350 à la variable pointée*

Afficher Nbre^ *//accéder à la valeur de la variable pointée c'est à dire 350*

desallouer(Nbre) *//on libère la zone mémoire*

Nbre ← Nil *//efface l'adresse mémoire qui n'a plus d'existence physique après avoir désalloue*

- Pour récupérer l'adresse d'une variable il faut :

Syntaxe :

variablepointeur ← @variable

- Affecter une valeur à la variable pointée :

Syntaxe :

variablepointeur^ ← valeur

Remarque :

- Attention, le fait de mettre Nil dans un pointeur ne libère pas l'emplacement sur lequel il pointait. L'emplacement devient irrécupérable car le lien vers cet emplacement a été coupé par la valeur Nil. Il faut désallouer avant d'affecter le pointeur avec Nil.
- Lorsqu'on désalloue un pointeur, il faut lui affecter la valeur Nil, pour qu'il ne conserve pas une adresse mémoire qui n'a plus d'existence physique.

Application 2 :

IV

Exercice

Partie 1 :

En respectant les instructions,précédent chaque trou, veuillez ajouter les valeurs manquantes

NB : Toutes les réponses doivent être en minuscule et sans espace

Algorithme pointeur

var

//créer trois variables(a,b et c en entier)

//créer deux pointeurs pt1 et pt2 en entier)

Début

a ← 5

// avec le pointeur pt1 réserve de l'espace pour stocker un entier

// stocker la valeur de a dans la zone mémoire réservée ci-dessus et par la suite pt1 doit récupérer l'adresse de a

←

←

b ← 3

c ← 2

//Faites que le pointeur pt2 pointe sur la même zone mémoire que pt1

←

// Modifier la valeur de la zone mémoire pointée par pt2 par la valeur de b

← b

// Donner l'adresse de b à pt2

←

// en utilisant uniquement pt1 augmenter la valeur de la zone mémoire réservée de 3

←

// afficher la valeur de la variable pointée de pt1

// affiche la valeur de la variable pointée de pt2

Fin

Partie 2 : Veuillez remplir le tableau avec les valeurs qui conviennent après l'exécution de l'algorithme

Application 2 :

variable	adresse	valeur
a	adr1	
b	adr2	
c	adr3	
pt1	adr4	
pt2	adr5	

3 - Travaux dirigés

V

Application :

Ce exercice a pour but de comprendre l'utilisation et la manipulation des pointeurs.

Algorithme pointeur

var

nb1,nb2 : entier

p1,p2 : ^entier

Début

nb1 ← 15

nb2 ← 9

allouer(p1) *// on réserve de l'espace pour stocker un entier*

p1^ ← nb1 *// stocke 15 dans la zone mémoire pointée par p1*

Afficher p1^ *// affiche 15*

nb1 ← 12

Afficher p1^ *// affiche 15*

p2 ← p1 *// p2 va pointer la zone mémoire pointée par p1*

Afficher p2^ *// affiche 15*

p2^← nb2 *// on place la valeur de nb2 qui est 9 dans la zone mémoire pointée par p2*

Afficher p2^ *// affiche 9*

Afficher p1^ *// affiche 9*

p1^ ← 14 *// stocke 14 dans la zone mémoire pointée par p1*

p2^ ← p1^+1 *// stocke 15 dans la zone mémoire pointée par p2*

Afficher p2^ *// affiche 15*

desallouer(p1) *// on libère la zone mémoire pintée par p1 et p2*

p1 ← nil // *//efface l'adresse mémoire qui n'a plus d'existence physique après avoir désalloue*

p2 ← nil // *//efface l'adresse mémoire qui n'a plus d'existence physique après avoir désalloue*

Fin