ALGORITMIQUE
AVANCE LANGAGE C

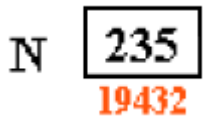
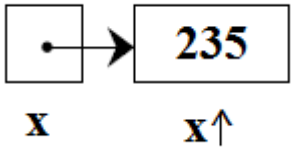
TCSRIT LICENCE 1

ESATIC
ECOLE SUPÉRIEURE AFRICAINE DES TIC

CHAPITRE 1 : LES POINTEURS

1.1 DEFINITION

Prenons par exemple une variable numérique **N** de type **ENTIER** d'adresse en mémoire centrale **19432** et contenant le nombre entier **235**. Nous appelons **variable de type POINTEUR *x*** vers cette variable **N**, une **variable dynamique** contenant l'adresse de la variable N.
Nous dirons aussi que ***x*** « pointe » vers la variable **N** et que le contenu de ***x*** ↑ est **235**.
L'information **N** peut être simple ou structurée.

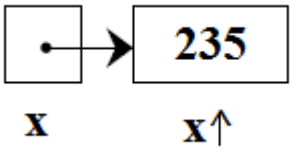
	

1.2 DECLARATION D'UNE VARIABLE DYNAMIQUE

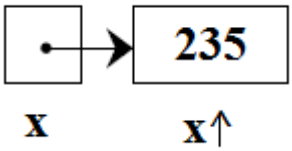
Une variable dynamique se déclare comme une variable classique mais le type est précédé du symbole «↑», elle est typée (le type de la donnée vers laquelle elle pointe), mais sa gestion est entièrement à la charge du programmeur à travers les procédures d'allocation et de désallocation mémoire respectivement appelées **new** et **dispose**.

1.3 UTILISATION PRATIQUE DES VARIABLES DYNAMIQUES

Le Contenu d'une variable dynamique « ***x*** » déjà allouée : il est noté « ***x*** ↑ »
Dans l'exemple précédent :

	***x*** ↑ **vaut 235** (contenu de la variable dynamique) ***x*** **vaut 19432** (adresse de la variable dynamique)

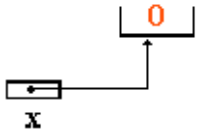
Détaillons pas à pas un programme d'utilisation de pointeur

Soit l'exemple précédent :  Le programme de droite écrit sur l'écran le contenu de la variable ***x*** ↑ (contenu de la cellule pointée par x)	Voici le programme à analyser : **Algorithme** VariableDynamique**;** **variable** x **: ↑Entier;** **Debut** **new**(x)**;** ***x*** ↑ ← 235**;** **Ecrire**("la variable d'adresse x vaut: ", ***x*** ↑)**;** **dispose**(x)**;** **Fin**

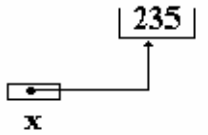
Déclaration d'une variable dynamique « *x* » de type entier :

Soit l'instruction : **variable *x* : ↑ENTIER** ;	**Résultat produit :**  ***x*** est créée (mais x ne pointe vers rien encore) **x vaut nil**

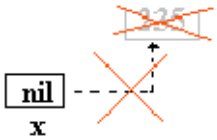
Allocation d'une variable dynamique « x » déjà déclarée :

Soit l'instruction : **new (*x*)** ;	**Résultat produit :**  une cellule mémoire de type **integer** est crée, *x* pointe vers la cellule créée. (*x* vaut la valeur de l'adresse de la cellule)

Affectation du contenu d'une variable dynamique « *x* » déjà déclarée :

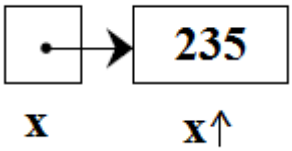
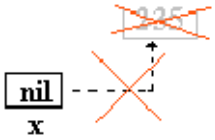
Soit l'instruction : ***x*** **↑ ← 235 ;**	**Résultat produit :**  **La cellule mémoire pointée par x contient 235.**

Désallocation d'une variable dynamique « *x* » déjà allouée :

Soit l'instruction : **dispose (*x*) ;**	Résultat produit :  La cellule mémoire qui contenait 235 n'existe plus, elle est rendue au système (ont dit qu'elle a été désallouée)

Attention

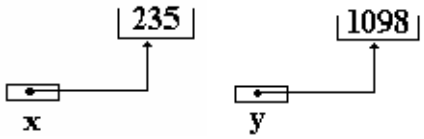
Ne pas confondre **l'effacement de l'adresse d'une variable dynamique** et sa **désallocation**.
Effacement de l'adresse d'une variable dynamique : mot clef **« NIL »**
Désallocation d'une variable dynamique : procédure **DISPOSE(…)**

Soit l'exemple précédent :	**Résultat produit par** x ← nil **:**
	 • ***x*** ↑ n'existe plus (x ne pointe vers plus rien) • **x vaut nil** • La cellule mémoire qui contient 235 **existe toujours**, mais n'est plus accessible ! **Résultat produit par** dispose (x) **:**  • ***x*** ↑ n'existe plus (x ne pointe vers plus rien) • **x vaut nil** • La cellule mémoire qui contenait 235 **n'existe plus** !

C'est en particulier cette dernière remarque qui pose le plus de soucis de maintenance aux développeurs utilisant les pointeurs (par ex : problème de la référence folle).

Affectation de variables dynamiques entre elles :

On suppose que deux variables dynamiques « x et y » de type ↑**ENTIER** ont été déclarées et créées par la procédure **new**, nous figurons ci-après l'incidence de l'affectation **x** ← **y** sur ces variables :

Soient les instructions : *x* ↑ ← 235 ; *y* ↑ ← 1098 ;	**Résultat produit :** 

Soient l'affectation : x ← y ; x et y pointent vers la même cellule mémoire	**Résultat produit :**