Algorithmique

Chapitre 3: Types Enumérés Et Algébriques

Dr. N'golo KONATE

konatengolo@ufhb.edu.ci

Types Enumérés Et Algébrique

1. Type intervalle
2. Types énumération
3. Types tableaux
4. Les pointeurs

Type intervalle

Type intervalle

Un type intervalle est un type dont les objets prennent leur valeur dans une portion continue de l'intervalle des valeurs d'un autre type (entier, énuméré ou caractère).

Type intervalle

Un type intervalle est un type dont les objets prennent leur valeur dans une portion de l'intervalle des valeurs d'un autre type (entier, énuméré ou caractère).

Le type intervalle est crée selon l’ensemble de définition concerné:

Type intervalle

Un type intervalle est un type dont les objets prennent leur valeur dans une portion de l'intervalle des valeurs d'un autre type (entier, énuméré ou caractère).

Le type intervalle est crée selon l'ensemble de définition concerné:

- Type entier

Type intervalle

Un type intervalle est un type dont les objets prennent leur valeur dans une portion de l'intervalle des valeurs d'un autre type (entier, énuméré ou caractère).

Le type intervalle est crée selon l'ensemble de définition concerné:

- Type entier
- Type chaine de caractère

Type intervalle

Un type intervalle est un type dont les objets prennent leur valeur dans une portion de l'intervalle des valeurs d'un autre type (entier, énuméré ou caractère).

Le type intervalle est crée selon l'ensemble de définition concerné:

- Type entier
- Type chaine de caractère

La syntaxe de déclaration est la suivante:

Type < nom du type intervalle > = [borne_inférieure, borne_supérieure]

Type intervalle

Un type intervalle est un type dont les objets prennent leur valeur dans une portion de l'intervalle des valeurs d'un autre type (entier, énuméré ou caractère).

Le type intervalle est crée selon l'ensemble de définition concerné:

- Type entier
- Type chaine de caractère

La syntaxe de déclaration est la suivante:

Type < nom du type intervalle > = [borne_inférieure, borne_supérieure]

La boucle pour est la mieux apte a traiter un type enuméré

Type intervalle

Des fonctions prédéfinies peuvent être utilisées sur les intervalles :

Type intervalle

Des fonctions prédéfinies peuvent être utilisées sur les intervalles :

- Incrémenter (x, n) : sert à incrémenter de n la variable x passée en paramètre

Type intervalle

Des fonctions prédéfinies peuvent être utilisées sur les intervalles :

- Incrémenter (x, n) : sert à incrémenter de n la variable x passée en paramètre
- Décrémenter (x, n) : sert à décrémenter de n la variable x passée en paramètre

Type intervalle

Des fonctions prédéfinies peuvent être utilisées sur les intervalles :

- Incrémenter (x, n) : sert à incrémenter de n la variable x passée en paramètre
- Décrémenter (x, n) : sert à décrémenter de n la variable x passée en paramètre
- Succéder (x) : permet de retourner le successeur de la variable passée en paramètre

Type intervalle

Des fonctions prédéfinies peuvent être utilisées sur les intervalles :

- Incrémenter (x, n) : sert à incrémenter de n la variable x passée en paramètre
- Décrémenter (x, n) : sert à décrémenter de n la variable x passée en paramètre
- Succéder (x) : permet de retourner le successeur de la variable passée en paramètre
- Précéder (x) : permet de retourner le prédécesseur de la variable passée en paramètre

Type intervalle

Des fonctions prédéfinies peuvent être utilisées sur les intervalles :

- Incrémenter (x, n) : sert à incrémenter de n la variable x passée en paramètre
- Décrémenter (x, n) : sert à décrémenter de n la variable x passée en paramètre
- Succéder (x) : permet de retourner le successeur de la variable passée en paramètre
- Précéder (x) : permet de retourner le prédécesseur de la variable passée en paramètre
- Ordonner (x) : sert à retourner le rang de la variable dans l'intervalle

Type intervalle

Des fonctions prédéfinies peuvent être utilisées sur les intervalles :

- Incrémenter (x, n) : sert à incrémenter de n la variable x passée en paramètre
- Décrémenter (x, n) : sert à décrémenter de n la variable x passée en paramètre
- Succéder (x) : permet de retourner le successeur de la variable passée en paramètre
- Précéder (x) : permet de retourner le prédécesseur de la variable passée en paramètre
- Ordonner (x) : sert à retourner le rang de la variable dans l'intervalle

Les fonctions succéder et précéder ne sont respectivement pas définies pour le dernier élément de la liste et pour le premier élément de la liste.

Type intervalle

Activités

1. Créer un type intervalle pour les lettres de l'alphabet français
2. Créer un type intervalle pour la base Octale
3. Créer un type intervalle pour la base hexadécimale
4. créer un type intervalle pour les voyelles

Type intervalle
Solution

Solution

Type alphabet = ['a', 'z']

On peut également utiliser des constantes de déclarations:

Const min ← 'a', max ← 'z'

Type alphabet = [min, max]

Type intervalle
Solution

Solution

Type alphabet = ['a', 'z']

On peut également utiliser des constantes de déclarations:

Const min ← 'a', max ← 'z'

Type alphabet = [min, max]

Application aux fonctions

Instructions	Résultats
incrémenter (d, 1)	`e`
décrémenter (d, 1)	`c`
succéder (a)	`b`
précéder (a)	la fonction retourne un message d'erreur
ordonner (f)	6

Type intervalle
Solution

Type alphabet = ['a', 'z']

On peut également utiliser des constantes de déclarations:

Const min ← 'a', max ← 'z'

Type alphabet = [min, max]

Application aux fonctions

Type énumération

Type énumération

Un type énuméré (ou énumération) est un type dont les valeurs sont données in extenso par le programmeur.
Un type énuméré permet de définir des valeurs n'existant pas dans les types fournis par le langage.
Un type énuméré s'utilise comme n'importe quel type pour typer des variables ou des paramètres.

Type énumération

La création d'un type énumération consiste à déclarer la liste des valeurs que peut prendre une variable de ce type.

Type énumération

La création d'un type énumération consiste à déclarer la liste des valeurs que peut prendre une variable de ce type.

Type < nom du type énumération > = (< liste des valeurs >)

Type énumération

La création d'un type énumération consiste à déclarer la liste des valeurs que peut prendre une variable de ce type.

Type < nom du type énumération > = (< liste des valeurs >)

La mise en œuvre du type énuméré respecte certains concepts clés:

Type énumération

La création d'un type énumération consiste à déclarer la liste des valeurs que peut prendre une variable de ce type.

Type < nom du type énumération > = (< liste des valeurs >)

La mise en œuvre du type énuméré respecte certains concepts clés:

1. Chaque élément de la liste est séparé de son prédécesseur ou de son suivant par une virgule.

Type énumération

La création d'un type énumération consiste à déclarer la liste des valeurs que peut prendre une variable de ce type.

Type < nom du type énumération > = (< liste des valeurs >)

La mise en œuvre du type énuméré respecte certains concepts clés:

1. Chaque élément de la liste est séparé de son prédécesseur ou de son suivant par une virgule.
2. chaque élément est repéré par sa position.

Type énumération

La création d'un type énumération consiste à déclarer la liste des valeurs que peut prendre une variable de ce type.

Type < nom du type énumération > = (< liste des valeurs >)

La mise en œuvre du type énuméré respecte certains concepts clés:

1. Chaque élément de la liste est séparé de son prédécesseur ou de son suivant par une virgule.
2. chaque élément est repéré par sa position.
3. Le premier élément de la liste prend le numéro 1, le deuxième élément prend le numéro 2, et ainsi de suite. L'ordre porte sur la place des éléments et non sur l'ordre des éléments

Type énumération

La création d'un type énumération consiste à déclarer la liste des valeurs que peut prendre une variable de ce type.

Type < nom du type énumération > = (< liste des valeurs >)

La mise en œuvre du type énuméré respecte certains concepts clés:

1. Chaque élément de la liste est séparé de son prédécesseur ou de son suivant par une virgule.
2. chaque élément est repéré par sa position.
3. Le premier élément de la liste prend le numéro 1, le deuxième élément prend le numéro 2, et ainsi de suite. L'ordre porte sur la place des éléments et non sur l'ordre des éléments
4. La liste est ainsi mise en correspondance avec l'intervalle [1, n].

Type énumération
Activité

Créer un type énuméré pour:

1. La base hexadécimale
2. Les 10 premiers nombres premiers

Type énumération
Solution

La base hexadécimale comporte les nombres: [0-9,A-F]

On écrit: type hexadécimal=(0,1,2,3,4,5,6,7,8,9,'A','B','C','D','E','F')

Instructions	Résultats
Incrémenter(0,2)	
Décrémenter('A',3)	
Succéder(5)	
Précéder(0)	
Ordonner('A)	

Type tableau

Type tableaux

La mise en œuvre de certaines taches nous oblige a utiliser une collection d'objet homogènes suivant un agencement définit par le programmeur.

Type tableaux

La mise en œuvre de certaines taches nous oblige a utiliser une collection d'objet homogènes suivant un agencement définit par le programmeur.

Saisir les noms de plusieurs étudiants

Saisir des notes et calculer la moyenne

Type tableaux

La mise en œuvre de certaines taches nous oblige a utiliser une collection d'objet homogènes suivant un agencement définit par le programmeur.

Saisir les noms de plusieurs étudiants

Saisir des notes et calculer la moyenne

Ces travaux nécessitent la mise en œuvre d'un tableau d'une dimension

Type tableaux

La mise en œuvre de certaines taches nous oblige a utiliser une collection d'objet homogènes suivant un agencement définit par le programmeur.

Saisir les noms de plusieurs étudiants

Saisir des notes et calculer la moyenne

Ces travaux nécessitent la mise en œuvre d'un tableau d'une dimension

D'autres part, on peut enregistrer les noms des étudiants et associés les notes.

Ce cas nécessite la mise en œuvre d'un tableau a deux dimension

Type tableaux

La mise en œuvre de certaines taches nous oblige a utiliser une collection d'objet homogènes suivant un agencement définit par le programmeur.

Saisir les noms de plusieurs étudiants

Saisir des notes et calculer la moyenne

Ces travaux nécessitent la mise en œuvre d'un tableau d'une dimension

D'autres part, on peut enregistrer les noms des étudiants et associés les notes.

Ce cas nécessite la mise en œuvre d'un tableau a deux dimensions

On peut aller a un tableau a plusieurs dimensions

Type tableaux

Un tableau est une structure de données linéaire qui permet de stocker des données de même type. Chaque donnée est repérée par un ou plusieurs indices qui indiquent la position de la donnée dans le tableau.

Type tableaux

Un tableau est une structure de données linéaire qui permet de stocker des données de même type. Chaque donnée est repérée par un ou plusieurs indices qui indiquent la position de la donnée dans le tableau.
Les données d'un tableau doivent être toutes du même type. C'est ce type commun qui est attribué au tableau. Donc le type d'un tableau est le type des données qu'il contient.

Type tableaux
Tableaux à une dimension

Un tableau est une variable particulière. À sa déclaration, il possède :

- un identificateur ;
- un type ;
- un ou plusieurs indice(s).

L'identificateur d'un tableau est le nom que l'on donne au tableau.

La syntaxe :

< nom du tableau > [1 .. indice maximal] : tableau de < type de donnée>

Type tableaux
Tableaux à une dimension

La saisie peut s'effectuer dans un ordre quelconque :

T [i] ← < valeur > // T[i] désigne la cellule d'indice i, et i varie de 1 à indice maximal

Type tableaux
Tableaux à une dimension

La saisie peut s'effectuer dans un ordre quelconque :

T [i] ← < valeur > // T[i] désigne la cellule d'indice i, et i varie de 1 à indice maximal

ou de manière séquentielle:

Pour (i ← 1 à < indice maximal > **pas de** 1) **Faire**

< instructions >

lire (T [i])

< instructions >

FinPour

Type tableaux
Tableaux à une dimension

L'affichage peut s'effectuer sur une cellule:

Ecrire(T [i])

ou sur tout le contenu:

Pour (i ← 1 à < indice maximal >) **Faire**

< instructions >

ecrire (T [i])

< instructions >

FinPour

Type tableaux
Tableaux à une dimension

Activité

1. Ecrire un algo permettant de générer un tableau composés des 50 premiers nombre impair
2. Ecrire un algorithme permettant de compter le nombre d'occurrence d'une valeur donnée par l'utilisateur en entré
3. Ecrire un algorithme permettant de calculer la moyenne et le minimum dans un tableau

Type tableaux
Tableaux à deux dimension

Les tableaux a deux dimensions correspondent a une matrice composée de lignes et de colonnes.

Type tableaux
Tableaux à deux dimension

Les tableaux a deux dimensions correspondent a une matrice composée de lignes et de colonnes.

Sa déclaration permet de préciser l'identificateur, le nombre maximal de lignes, le nombre maximal de colonnes et le type du tableau:

< nom du tableau > [1 .. indice ligne maximal, 1 .. indice colonne maximal] :

tableau de < type de donnée >

Exemple: Créer un tableau Etudiants de 10 ligne et 5 colonnes: **Etudiant[1:10,1:5]**

Type tableaux
Tableaux à deux dimension

La saisie d'un tableau a deux dimensions obéit aux mêmes règles qu'un tableau a une dimension.

Type tableaux
Tableaux à deux dimension

La saisie d'un tableau a deux dimensions obéit aux mêmes règles qu'un tableau a une dimension.

- On saisit le contenu d'une cellule:

Etudiant[i,j]← valeur

Type tableaux
Tableaux à deux dimension

- Soit séquentiellement. Dans ce cas, la boucle principale (boucle extérieure) parcourt le tableau en ligne tandis que la boucle intérieure parcourt le tableau en colonne.

```
Pour (i ← 1 à 10 pas de 1) Faire
   Pour (j ← 1 à 5 pas de 1) Faire
            < instructions >
            lire (Etudiant [i, j])
           < instructions >
  FinPour j
FinPour i
```

Type tableaux
Tableaux à deux dimension

Soit séquentiellement. Dans ce cas, la boucle principale (boucle extérieure) parcourt le tableau en ligne tandis que la boucle intérieure parcourt le tableau en colonne.

```
Pour (i ← 1 à 10 pas de 1) Faire
   Pour (j ← 1 à 5 pas de 1) Faire
            < instructions >
            lire (Etudiant [i, j])
           < instructions >
  FinPour j
FinPour i
```

En règle général on omettra le pas lorsqu'il est de 1

Type tableaux
Tableaux à deux dimension

L'affichage est également similaire a un tableau a une dimension:

- On pourra afficher le contenu d'une cellule

écrire (Etudiant[i, j])

Type tableaux
Tableaux à deux dimension

- Soit sur tout le tableau:

Pour (i ← 1 à 10 **pas de** 1)**Faire**

    Pour (j ← 1 à 5 **pas de 1**) **Faire**

        < instructions >

        écrire (**Etudiant** [i, j])

        < instructions >

    FinPour j

FinPour i

Type tableaux
Tableaux à deux dimension

Activité

1. Ecrire un tableau permettant de remplir une matrice carré d'ordre 5

Type tableaux
Tableaux à deux dimension

```
Algorithme Matrice
Var T[1:5,1:5] : tableau de réel
 i, j : entier
Début
écrire ("Saisie et affichage d'une matrice carrée d'ordre 5")
écrire ("Saisie des éléments…")
Pour (i ← 1 à 5) Faire
    Pour (j ← 1 à 5) Faire
            écrire ("Donnez l'élément de la ligne ", i, " colonne ", j, " : ")
            lire (T [i,j])
    FinPour j
FinPour i
écrire ("Saisie terminée !")
écrire ("Vous avez saisi…")
  Pour (i ← 1 à 3) Faire
      Pour (j ← 1 à 3) Faire
          écrire (T [i,j])
      FinPour j
FinPour i
Fin
```

Les pointeurs

Les pointeurs

Les variables « classiques », déclarées avant l'exécution d'un programme, pour ranger les valeurs nécessaires à ce programme, sont des variables statiques, c'est à dire que la place qui leur est réservée en mémoire est figée durant toute l'exécution du programme. Ceci a deux conséquences :

Les pointeurs

Les variables « classiques », déclarées avant l'exécution d'un programme, pour ranger les valeurs nécessaires à ce programme, sont des variables statiques, c'est à dire que la place qui leur est réservée en mémoire est figée durant toute l'exécution du programme. Ceci a deux conséquences :

- Risque de manquer de place si la place réservée (par exemple le nombre d'éléments d'un tableau) est trop petite. Il faut alors que le programme prenne en charge le contrôle du débordement.

Les pointeurs

Les variables « classiques », déclarées avant l'exécution d'un programme, pour ranger les valeurs nécessaires à ce programme, sont des variables statiques, c'est à dire que la place qui leur est réservée en mémoire est figée durant toute l'exécution du programme. Ceci a deux conséquences :

- Risque de manquer de place si la place réservée (par exemple le nombre d'éléments d'un tableau) est trop petite. Il faut alors que le programme prenne en charge le contrôle du débordement.

- Risque de risque de gaspiller de la place si la place réservée est beaucoup plus grande que celle qui est effectivement utilisée par le programme

Les pointeurs

Les variables dynamiques vont permettre :

- De prendre la place en mémoire au fur et à mesure des besoins, celle-ci étant bien sûr limitée par la taille de la mémoire.

- De libérer de la place lorsqu'on n'en a plus besoin.

Les pointeurs

Un pointeur est une adresse mémoire: il permet de désigner directement une zone de la mémoire et donc l'objet dont la valeur est rangée à cet endroit.

Les pointeurs

Un pointeur est une adresse mémoire: il permet de désigner directement une zone de la mémoire et donc l'objet dont la valeur est rangée à cet endroit.

À partir d'un type t quelconque, on peut définir un type pointeur_sur_t.

Les variables du type pointeur_sur_t contiendront, sous forme d'une adresse mémoire, un mode d'accès au contenu de la variable de type t.

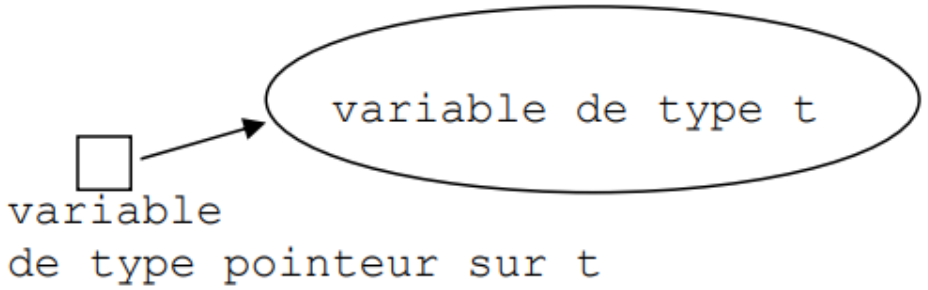
Les pointeurs

Un pointeur est une adresse mémoire: il permet de désigner directement une zone de la mémoire et donc l'objet dont la valeur est rangée à cet endroit.

À partir d'un type t quelconque, on peut définir un type pointeur_sur_t.

Les variables du type pointeur_sur_t contiendront, sous forme d'une adresse mémoire, un mode d'accès au contenu de la variable de type t.

Celle-ci n'aura pas d'identificateur, mais sera accessible uniquement par l'intermédiaire de son pointeur.



Les pointeurs
Declaration

On a défini un type t.

On déclare : ptr : pointeur_sur_t ou ^t.

Les pointeurs
Allocation

Création d'une variable dynamique de type t

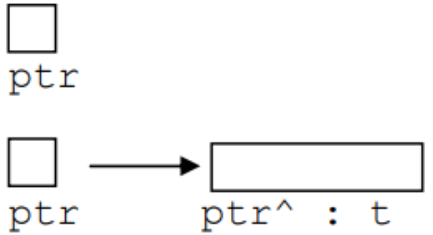
allouer(ptr)

- réserve un emplacement mémoire de la taille correspondant au type t,
- met dans la variable ptr l'adresse de la zone mémoire qui a été réservée.

L'emplacement pointé par ptr sera accessible par ptr^.

```
ptr : ^t

allouer(ptr)
```



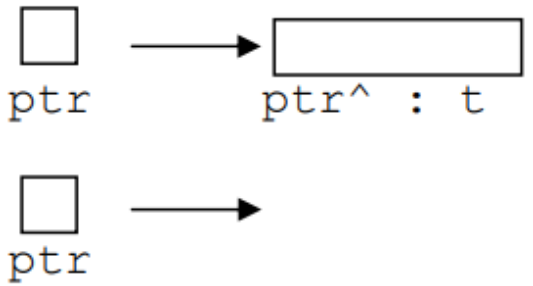
Les pointeurs
desallocation

Libération de la place occupée par une variable dynamique

desallouer(ptr)

- libère la place de la zone mémoire dont l'adresse est dans ptr (et la rend disponible pour l'allocation d'autres variables)
- laisse la valeur du pointeur en l'état (n'efface pas l'adresse qui est dans la variable pointeur).

```
desallouer(ptr)
```



Les pointeurs

Affectation entre 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.

Les pointeurs

Affectation entre 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 (ou Null). 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.

Ptr← Nil

Les pointeurs

Affectation entre pointeurs

Règle de bonne programmation : dès qu'on a désalloué 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.

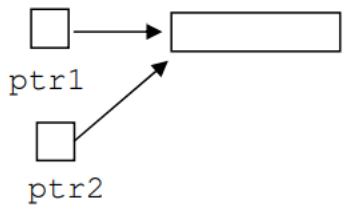
Les pointeurs
Affectation entre pointeurs

Règle de bonne programmation : dès qu'on a désalloué 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.

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

```
ptr1, ptr2 : pointeur sur t
allouer(ptr1)

ptr2  ← ptr1
```

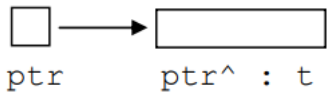


Pour libérer la zone mémoire on désalloue ptr1 ou (exclusif) ptr2, puis on met la valeur Nil dans ptr1 et dans ptr2.

Les pointeurs

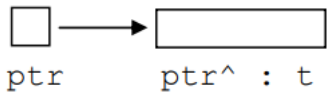
Accès à la variable pointée

Une fois la variable ptr déclarée et allouée, l'accès en écriture ou lecture à la variable pointée par ptr se fait avec l'identificateur ptr ^. On peut appliquer à ptr^ les même instructions qu'à une variable simple



Les pointeurs
Accès à la variable pointée

Une fois la variable ptr déclarée et allouée, l'accès en écriture ou lecture à la variable pointée par ptr se fait avec l'identificateur ptr ^. On peut appliquer à ptr^ les même instructions qu'à une variable simple



Affecter une valeur à la variable pointée : ptr^ ← <expression de type t>

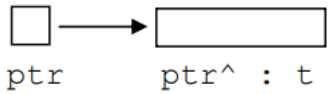
Lire une valeur de type t et la mettre dans la

variable pointée :

lire(ptr^)

Les pointeurs
Accès à la variable pointée

Une fois la variable ptr déclarée et allouée, l'accès en écriture ou lecture à la variable pointée par ptr se fait avec l'identificateur ptr ^. On peut appliquer à ptr^ les même instructions qu'à une variable simple



Affecter une valeur à la variable pointée : ptr^ ← <expression de type t>

Lire une valeur de type t et la mettre dans la

variable pointée :

lire(ptr^)

Afficher la valeur présente dans la zone pointée : écrire(ptr^)

Les pointeurs

Accès à la variable pointée

Exemple: L'algorithme suivant n'a comme seul but que de faire comprendre la manipulation des pointeurs. À droite les schémas montrent l'état de la mémoire en plus de ce qui s'affiche à l'écran.

Les pointeurs

Accès à la variable pointée

Variables
ptc : pointeur sur chaîne de caractères
ptx1, ptx2 : pointeur sur entier