ESI SALAMA

LUBUMBASHI

Support de Cours

ALGORITHMIQUE AVANCEE

PA ANGOMA Blaise

CHAP IV
STRUCTURE DES DONNEES

I. ENREGISTREMENT (STRUCTURE)

Intérêt :

Rassembler des données hétérogènes caractérisant une entité pour en faire un type utilisateur appelé structure de données ou enregistrement (record).

Contrairement aux tableaux qui sont des structures de données dont tous les éléments sont de même type, les enregistrements sont des structures de données dont **les éléments peuvent être de type différent** et qui se rapportent à la même entité. Les éléments qui composent un enregistrement sont appelés champs.

On peut construire de nouveaux types de données agrégées. Pour cela, on doit préciser :

- Le **nom** du nouveau type
- Le **nom** et la **nature** des données hétérogènes

Structure *nom_type*

nom_champ1: type_champ1

…

nom_champN: type_champN

FinStruct

On peut également utiliser le mot clé **enregistrement** ou **struct** pour la déclaration d'un type structure

Exemple

```
Structure Etudiant
    Chaine nom[20]
    Chaine prenom[20]
    Entier Age
Fin Struct

Etudiant Kalo ;
Etudiant T[20] ;
```

```
Structure Matiere
    Reel math
    Reel anglais
    Reel Info
    Reel Moyenne
FinStruct

Matiere Tab[100];
```

nom	prenom	age						

Manipulation d'un enregistrement

Accès aux champs d'une structure

***Variable_de_type_structure*.champ**

Exemple

Kalo.nom

T[10].prenom

Tab[5].moyenne = (Tab[5].math + Tab[5].anglais + Tab[5].info)/3

Un enregistrement comme champ d'une structure

Structure Date

Entier jour

Entier mois

Entier annee

Fin Struct

Structure Personne

Chaine Nom

Date date_naissance

FinStruct

Personne K

Pour accéder à l'année de naissance d'une personne, il faut utiliser deux fois l'opérateur '.'

K.date_naissance.annee

Exercice

Définir une structure ville qui représente une pays caractérisée par : son nom , sa capitale et sa superficie.

Ecrire un programme qui lit N villes et affiche les villes selon l'ordre croissant de leurs superficies.

II.PILE

Définition

Une pile est une liste linéaire d'éléments (entiers,….) où l'ajout et la suppression d'une élément se font toujours du même coté

Sommet

Si la pile n'est pas vide, l'élément accessible est celui qui se trouve au **sommet de la pile**

Remarque :

L'expression **«Sommet de la pile»** peut désigner l'élément accessible ou sa position.

Les deux opérations principales sont :

- **Empiler** : ajouter un élément au dessus du sommet de la pile
- **Dépiler** : supprimer l'élément au sommet de la pile si elle n'est pas vide.

D'autres opérations possibles :

- **CreerPile** : Crée une pile vide P
- **Sommet** : retourne l'élément au sommet de la pile qui doit être non vide
- PileVide : Détermine si la pile est vide ou non et retourne Vrai ou Faux

Implémentation d'une pile avec un tableau

- La pile est représentée par un enregistrement contenant les champs suivants :
  - L'indice du sommet de la pile : entier
  - Un tableau contenant des éléments de la pile
- Ce tableau peut être statique (taille maximale fixée) ou dynamique
- Il faut faire un contrôle de taille lors de l'empilement pour éviter un débordement.

(Indice du sommet) < (Taille maximale)

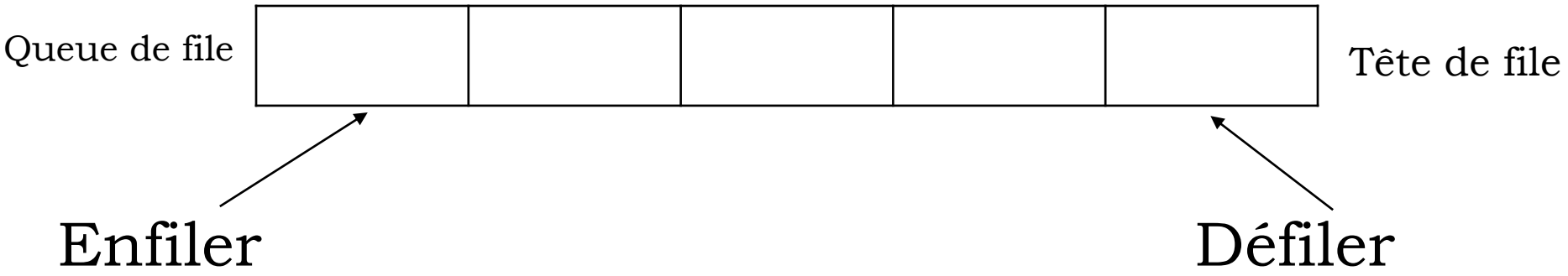
Déclaration d'une Pile avec un tableau statique

```
Structure Pile
    Entier S                    // sommet
    Entier T[100]
Fin Struct
```

III.FILE

Définition

Une file est une liste linéaire d'éléments où ***les insertions*** *se font d'un côté et* ***les suppressions*** *de l'autre côté.*



- Les noms spécifiques pour ces opérations sont **Enfiler** pour insérer et **Défiler** pour supprimer

- Les opérations sur les files sont syntaxiquement les mêmes que sur les piles; c'est par leur effet qu'elles diffèrent : l'élément supprimé est le premier arrivé dans la file.
- Une file se comporte comme une file d'attente (suit la stratégie **First In First Out(FIFO))**

Les opérations sur les files sont :

- **CréerFile** : crée une file vide
- **Tête** : retourne l'élément en tête de la file qui doit être non vide
- **Enfiler** : Insère un élément à la fin de la file
- **Défiler** : supprime l'élément en tête de la file qui doit être non vide.

Implémentation d'une file avec un tableau

- Simplicité d'implémentation
- Quand le nombre d'insertion est à priori limité
- **Tableau** des éléments
- **Deux indices**, représentant respectivement le début et la fin de la file
- Opération supplémentaire : **FilePleine** qui détermine si une file est pleine.

Déclaration d'une File avec un tableau statique

```
Structure Pile
    Entier T, Q                 // Tête et Queue de la file
    Entier N                    // Nombre maximal d'éléments
    Entier T[100]
Fin Struct
```

Rappel Pointeur

Un pointeur est une variable susceptible de contenir une adresse mémoire

Déclaration

Type_valeur_pointée * pointeur ;

Exemple 1 : Entier * p

&p : adresse de p

*p : contenu de la zone mémoire dont l'adresse est référencée par pointeur

Exemple 2 Entier a, *p

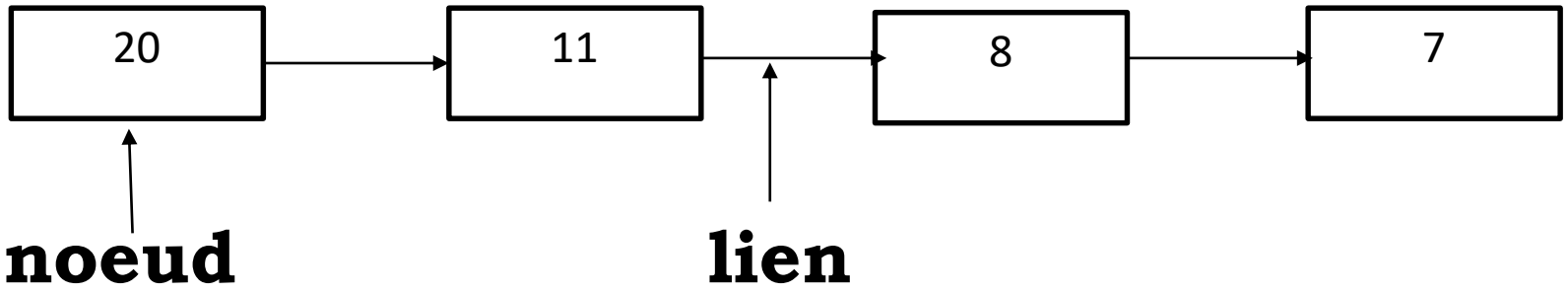
p=&a;

Lorsqu'un pointeur ne contient aucune adresse, il vaut mieux l'initialiser à **NULL**

IV. LES LISTES CHAINEES

Définition

Une liste chainée est un semble d'éléments ***non contigus*** *organisés* ***séquentiellement****; tandis que pour un tableau les éléments en mémoire sont* ***contigus****.*



Déclaration de la structure d'une liste chainée

Structure Elem

Entier info ;

Elem * next ;

FinStruct

- Le champ info contient la valeur de la case tandis que le pointeur **next** pointe sur la case du prochain élément
- Si le prochain élément n'existe pas , le pointeur **next** pointe sur NULL (NIL)

Opération sur les listes chaînées

- Ces opérations sont basées sur la manipulation des liens.
- Un nœud est une structure contenant
  - Les données utilisateur(entier, réel,…..) (les éléments de la liste)
  - Un pointeur(lien) vers le nœud suivant
- Le dernier élément a un lien **NULL**
- On mémorise l'adresse du premier nœud dans un pointeur appelé l'entête de la liste chainée
- **Opérations**
  - Liste vide
  - Insérer un élément au début ou à la fin de la liste
  - Supprimer un élément au début ou à la fin de la liste
  - Afficher la liste

les listes chaînées circulaires

- Sont des listes chainées dont les derniers éléments pointent sur les premiers éléments.

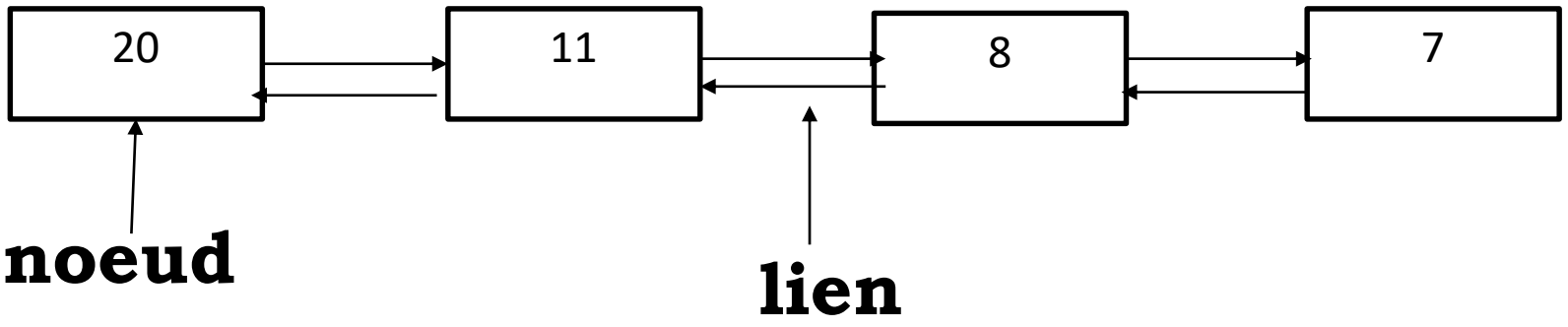
Exercice

Ecrire un programme principal (main) permettant à l'utilisateur de saisir plusieurs entiers jusqu'à ce qu'il veuille s'arrêter. Chaque valeur saisie doit être insérer au début d'une liste chainée.

IV. LES LISTES DOUBLEMENT CHAINEES

Définition

Une liste doublement chainée une autre extension possible des listes simplement liées dans laquelle on enregistre dans chaque élément de la liste l'adresse de l'élément qui le précède, en plus de l'élément suivant.



Déclaration de la structure d'une liste doublement chainée

Structure Elem

Entier info ;

Elem * prec ;

Elem * next ;

FinStruct

Comme dans le cas des listes simplement liées, le dernier élément de la liste aura son champ next à **NIL**.
De façon symétrique, le premier élément de la liste aura son champ prec à **NIL** également, puisqu'aucun élément ne le précède.

Opération sur les listes doublement chaînées

- Liste vide
- Insérer un élément au début ou à la fin de la liste
- Insérer un élément à la $n^{ième}$ position
- Supprimer un élément au début ou à la fin de la liste
- Supprimer un élément à la $n^{ième}$ position
- Afficher la liste
- Trier la liste
- Vider la liste

Conclusion

Structure	Dimension	Position d'une information	Accès à une information
Tableau	Fixe	Par son indice	Directement par l'indice
Liste chaînée	Evolue selon les actions	Par son adresse	Séquentiellement par le pointeur de chaque élément

Structures	Ajout	Suppression	Type de Liste
PILE	Tête	Tête	LIFO (Last In First Out)
FILE	Queue	Tête	FIFO (First In First Out)