ALGORITMIQUE
AVANCE LANGAGE C

TCSRIT LICENCE 1

KONAN HYACINTHE

ESATIC
ECOLE SUPÉRIEURE AFRICAINE DES TIC

CHAPITRE 2 : LES LISTES LINEAIRES CHAINEES

2.1 Les Inconvénients de structure de données TABLEAU

La simulation de phénomènes du monde physique tels que :

- La file d'attente à un guichet ;
- Les urgences d'un hôpital ;
- La distribution de cartes à une table de joueurs ;
- La gestion des dossiers empilés sur un bureau ;

Est mal représentés par la structure en tableaux. En effet cette structure de données présente les contraintes suivantes :

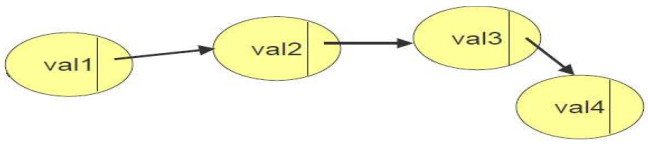
- Nécessité de définir la dimension maximale allouée dès la déclaration, c'est à dire de façon statique
- Lourdeur dans la gestion optimale de l'espace occupé par les données
- Nécessité d'effectuer des contrôles de débordement à chaque étape du traitement

Une représentation moins contrainte devrait :

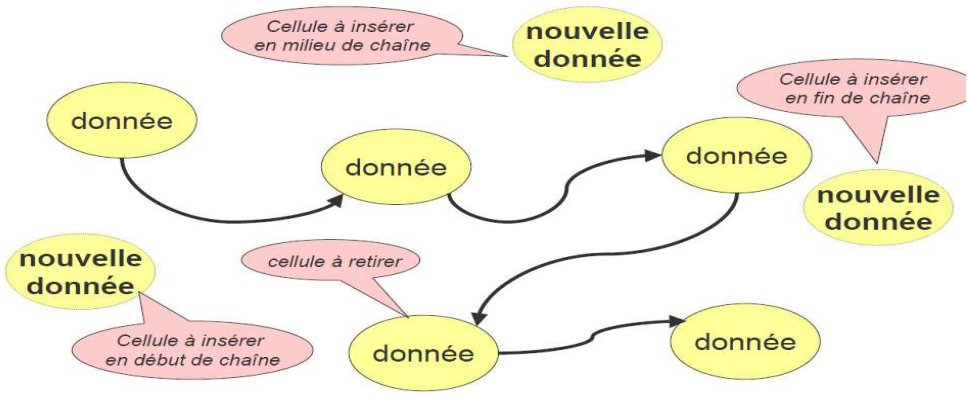
- Permettre l'allocation de mémoire en fonction des besoins, de façon dynamique
- Faciliter la gestion de la mémoire occupée en cas d'insertion ou de suppression de nouvelles données
- Permettre la simulation de phénomènes du monde physique mal représentés par la structure en tableaux

2.2 La notion de liste chaînée

Une **liste linéaire** sur un ensemble E est une suite finie $(e1; \ldots; en)$ d'éléments de E. Chaque élément est une CELLULE composée d'une donnée et d'un pointeur vers la cellule suivante.



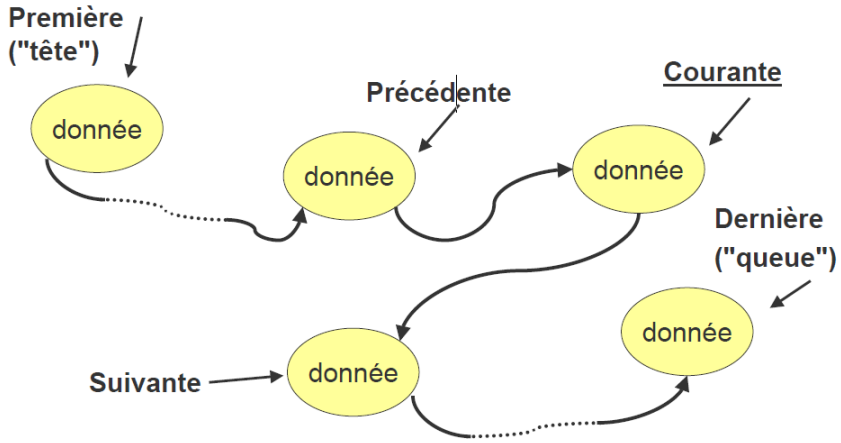
2.2.1 Insertion/suppression d'une donnée



2.2.2 Repérage des cellules d'une liste chaînée

Le repérage des cellules d'une liste chainée s'effectue à l'aide de pointeur. Les éléments à repérer sont :

- Le premier élément de la liste (la tête de liste)
- L'élément courant
- L'élément précédant
- L'élément suivant
- Le dernier élément (la queue de liste)



2.2.3 Exemples de traitements opérés sur les listes

Traitements relatifs à la composition structurelle de la liste :

- Positionnement sur la première cellule de la structure
- Positionnement sur la dernière cellule de la structure
- Calcul de la longueur d'une liste (nombre de cellules)
- Reconnaissance d'une liste vide
- Déplacement du positionnement courant sur la cellule suivante

Traitements relatifs à l'information enregistrée dans une liste:

- Enregistrement de données jusqu'à épuisement du flot de données
- Visualisation de l'information enregistrée dans une cellule, quelle que soit sa place dans la liste
- Visualisation de l'ensemble des informations enregistrées dans la liste
- Suppression d'une cellule; ajout d'une cellule

2.2.4 Définition de la structure de données listes

Les attributs de la classe Liste doivent permettre :

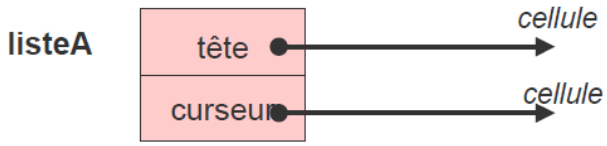
- le positionnement sur les différentes cellules de la liste
- la définition du type d'information enregistrée dans la cellule

Les procédures qui manipuleront cette structure de données doivent permettre tous les traitements décrits précédemment (et plus...)

Type **LISTE** = enregistrement
tête: référence *{référence à la cellule tête de liste}* ;
curseur : référence *{référence à la cellule courante}*;
Fin;

Variable ListeA : **LISTE**

référence: type dont le domaine de définition est l'ensemble des adresses mémoire.
-Valeur de l'attribut tête(attribut curseur) : adresse de la case mémoire où est stockée la cellule de tête de liste (cellule courante)
-Représentation graphique : une flèche

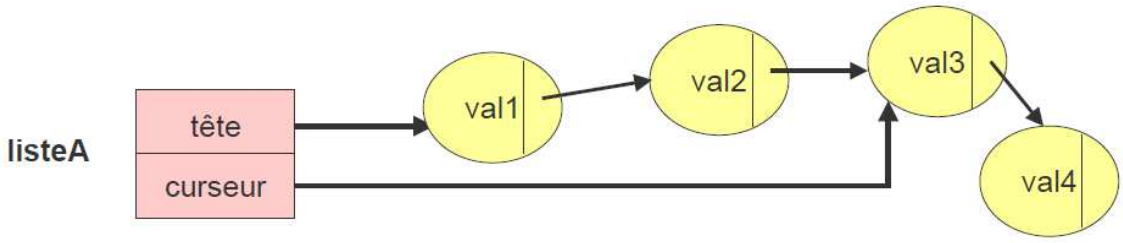


2.2.5 Définition d'une cellule

Type **CELLULE** = enregistrement
Val : info *{information stockée dans une cellule}*;
Suivant : **REFERENCE** *{référence à une autre cellule}*;
Fin;

Type **REFERENCE** = ↑**CELLULE**

Info: le type de l'information stockée; peut-être un type de base (par exemple, un entier) ou bien un type complexe (un agrégat)
Représentation graphique des cellules :



2.3 Les procédures communément utilisées pour manipuler les listes

Procédure suivant(cible : Liste)
{place le curseur sur la cellule qui suit la cellule courante. Si le curseur était sur la dernière cellule, il devient hors liste. Erreur si la liste est vide ou si le curseur est déjà hors liste.}
Paramètre (D/R) cible : Liste

Procédure premier(cible : Liste)
{place le curseur sur la première cellule de la liste. Si la liste est vide, le curseur reste hors liste.}
Paramètre (D/R) cible : Liste

Procédure dernier(cible : Liste)
{place le curseur sur la dernière cellule de la liste. Si la liste est vide, le curseur reste hors liste.}
Paramètre (D/R) cible : Liste

Fonction vide(cible : Liste) retourne booléen
{retourne vrai si la liste est vide, faux sinon}
Paramètre (D) cible : Liste

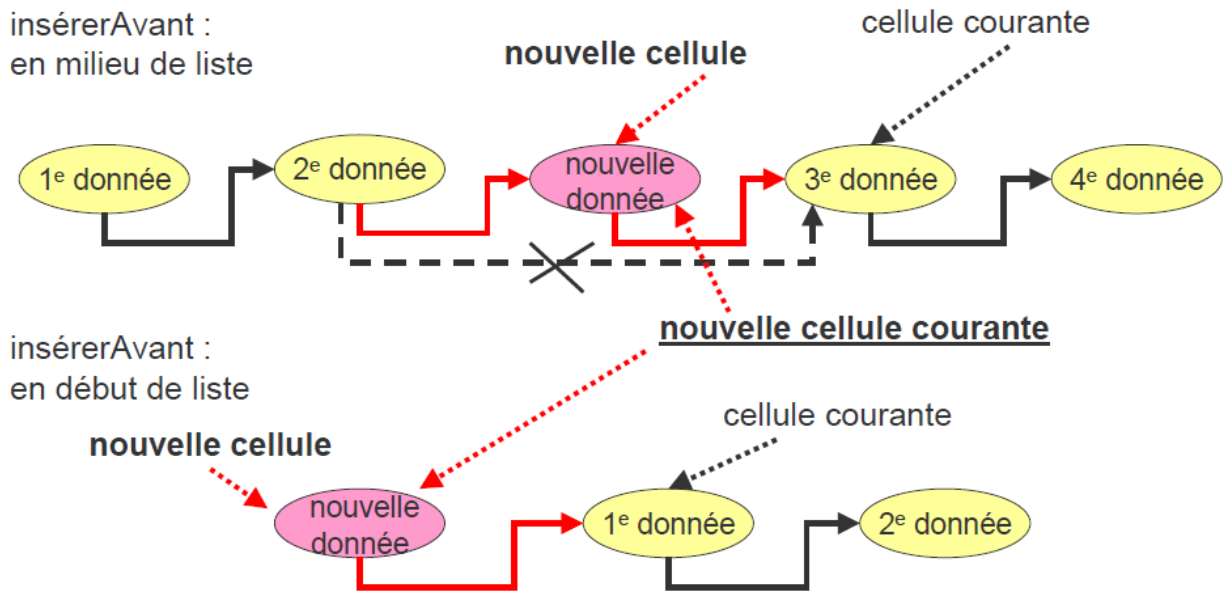
Fonction horsListe(cible : Liste**) retourne booléen**
{retourne vrai si le curseur est placé hors liste ou si la liste est vide, faux sinon.}
Paramètre (D) cible : Liste

Fonction info() retourne Info
{retourne la valeur enregistrée dans la cellule courante. Erreur si la liste est vide ou si le curseur est hors liste.}
Paramètre (D) cible : Liste

Procédure affecter(val)
{affecte la valeur val à la cellule courante Erreur si la liste est vide ou si le curseur est hors liste.}
Paramètres (D/R) cible : Liste
(D) val : Info

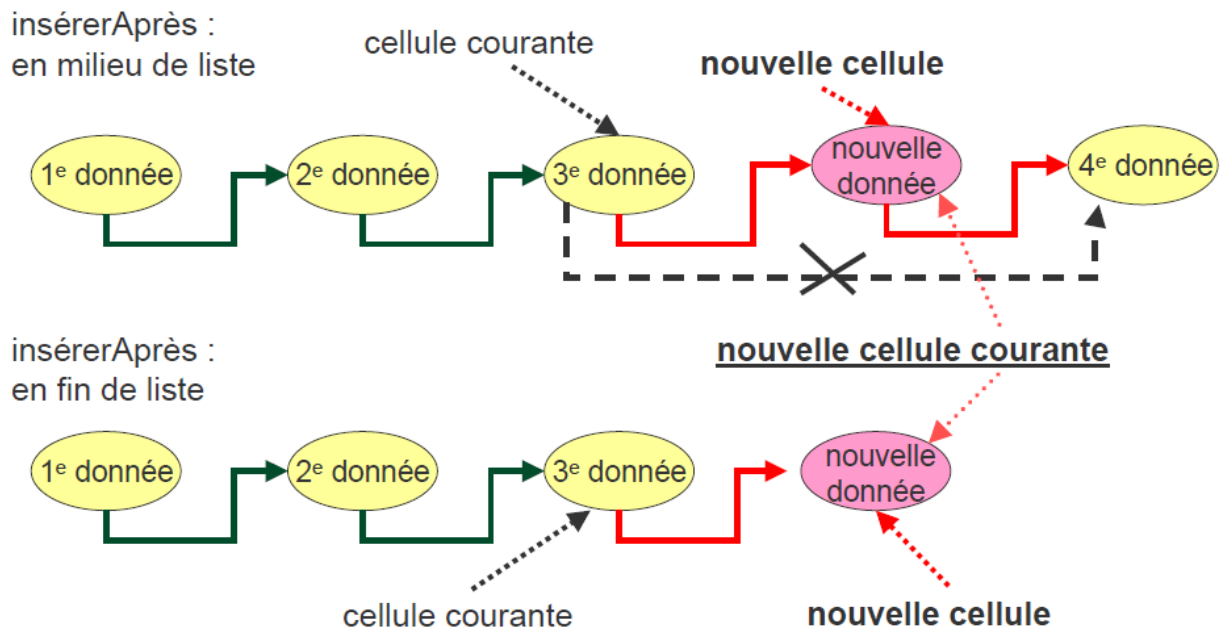
Procédure insérerAvant(val)
{crée une nouvelle cellule, y affecte la valeur val, et l'insèreavantla cellule courante. Le curseur est alors placé sur cette nouvelle cellule qui devient ainsi la nouvelle cellule courante. Si le curseur était placé sur la tête, la nouvelle cellule devient la nouvelletête. Si la liste était vide, elle contient maintenant l'unique cellule qui vient d'être créée.
Erreur si la liste était non vide et le curseur hors liste.}
Paramètres (D/R) cible : Liste
(D) val : Info

Insertion d'une cellule : insérerAvant



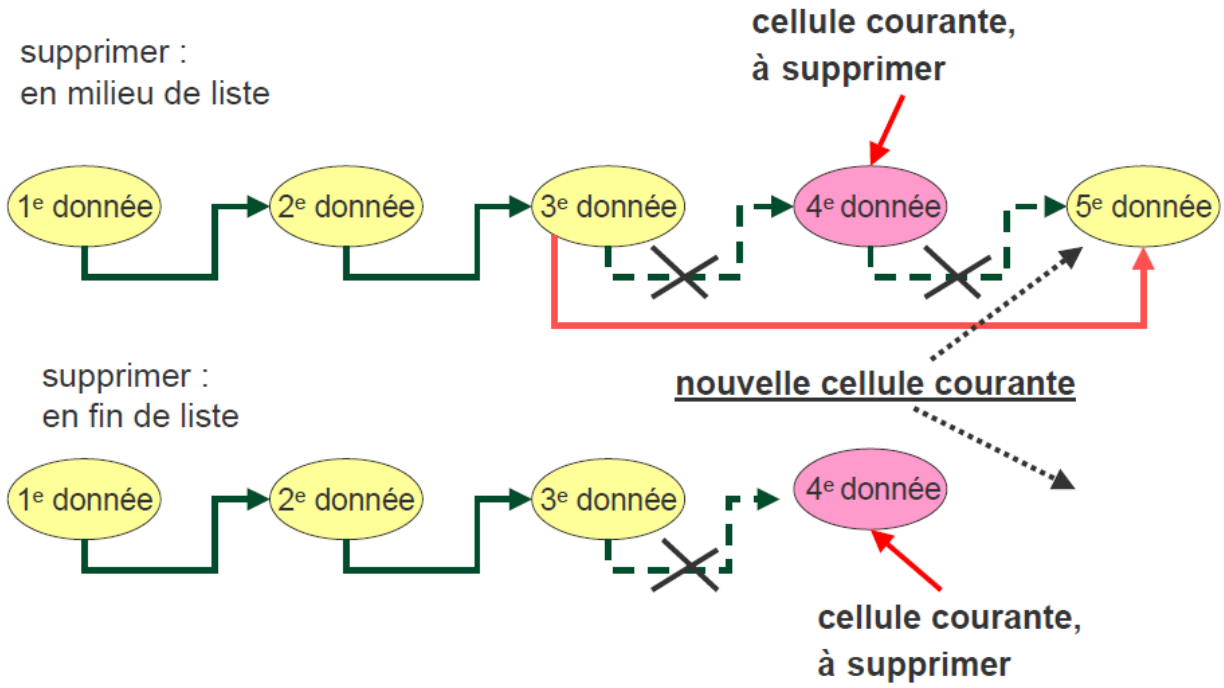
Procédure insérerAprès(cible : Liste ; val : Info)
{crée une nouvelle cellule, y affecte la valeur val, et l'insère après la cellule courante. Le curseur est alors placé sur cette nouvelle cellule qui devient ainsi la nouvelle cellule courante. Si liste était vide, elle contient maintenant l'unique cellule qui vient d'être créée. Erreur si la liste était non vide et le curseur hors liste.}
Paramètres (D/R) cible : Liste
(D) val : Info

Insertion d'une cellule : insérerAprès



Procédure supprimer(cible : Liste)
{supprime la cellule courante. Le curseur est alors placé sur la cellule suivante qui devient ainsi la nouvelle cellule courante. Si la cellule à supprimer est la dernière cellule, le curseur devient hors liste. Si la cellule à supprimer est la tête, la cellule suivante devient alors la nouvelle tête. Si la liste ne contenait qu'une seule cellule, la liste devient vide. Erreur si la liste est vide ou si le curseur est hors liste.}
Paramètres (D/R) cible : Liste

Suppression d'une cellule



Procédure saisirListe(cible : Liste)
{Saisit des valeurs (de type Info), jusqu'à une valeur d'arrêt, et crée au fur et à mesure autant de cellules que nécessaire, en y affectant les valeurs saisies. Le curseur est ensuite replacé en tête de liste.}
paramètre(R) cible : Liste

Procédure afficherListe(cible : Liste)
{Affiche toutes les valeurs contenues dans la liste cible.}
paramètre(D) cible : Liste

Procédure supprimerTout(cible : Liste)
{supprime toutes les cellules de la liste (qui peut être vide); la liste devient vide et le curseur devient hors liste.}
paramètre(D/R) cible : Liste

Saisie d'une liste

Procédure saisirListe(cible : Liste)
{Saisit des valeurs (de type Info), jusqu'à une valeur d'arrêt (constante définie dans l'algorithme appelant), et crée au fur et à mesure autant de cellules que nécessaire, en y affectant les valeurs saisies. Le curseur est ensuite replacé en tête de liste.}

```
        Paramètre    (R) cible : Liste
        Variables    uneVal : Info
                     cpt : entier
début
        saisir(uneVal) ; cpt←0
        tant que uneVal≠valStop faire
                cpt←cpt + 1
                insérerAprès(cible ; uneVal)
                saisir(uneVal)
        ftq
        premier(cible){replace le curseur en tête}
        afficher(”La nouvelle liste contient ”, cpt, “ cellules.”)
fin
```

Saisie d'une liste : simulation

Procédure afficherListe(cible : Liste)
{Affiche toutes les valeurs contenues dans la liste cible.}

```
        paramètre(D) cible : Liste
        variable copieCible : Liste
début
        copieCible ← cible{copie de la cible, pour permettre modification du curseur}
        premier(copieCible){place le curseur en tête}
        tant que non horsListe(copieCible) faire
                {arrêt quand curseur hors liste}
                afficher(info(copieCible)){récupère la valeur de la cellule courante et l'affiche}
                copieCible.suivant() {place le curseur sur la cellule suivante}
        ftq
fin
```

Exemple d'algorithme

Algorithme ManipListes1

{Saisie et affichage d'une liste.}
constante (VALSTOP : entier) ← 0
variable listeA: Liste

début

saisirListe(listeA)
afficher("Liste saisie : ")
afficher(listeA)

fin

Exemple de fonction utilisant la liste.

Fonction valMin(uneListe) retourne réel

{retourne la plus petite valeur contenue dans une liste de réelssupposée non vide}
Paramètre (D) uneListe : ListeRéel

Fonction inverse(uneListe) retourne Liste

{crée une nouvelle liste, en y affectant les valeurs de la listeuneListemais dans l'ordre inverse. La liste uneListeest supposée non vide}
Paramètre (D) uneListe : Liste

Procédure supprimerGlobal(uneVal, uneListe)

{supprime toutes les cellules de la liste uneListequi contiennent la valeur uneVal. Replace le curseur en tête.}
Paramètre (D/R) uneListe: Liste
(D) uneVal: Info

Algorithme ManipListes2
{Exemple d'algorithme manipulant les listes.}
constante (VALSTOP : entier) ← 0
variables listeA, listeB, listeC : Liste
début
saisir(listeA)
afficher("listeA: ")
afficher(listeA)
si nonlisteA.vide() **alors**
listeB ← **inverse**(listeA)
afficher("listeBcontient les éléments de lalisteAen ordreinverse : ")
afficher(listeB)
supprimerGlobal(0,listeA)
afficher("ListeA sans zéros: ")
afficher(listeA)
afficher("Plus petite valeur delisteA:" , valMin(listeA))
fsi
fin

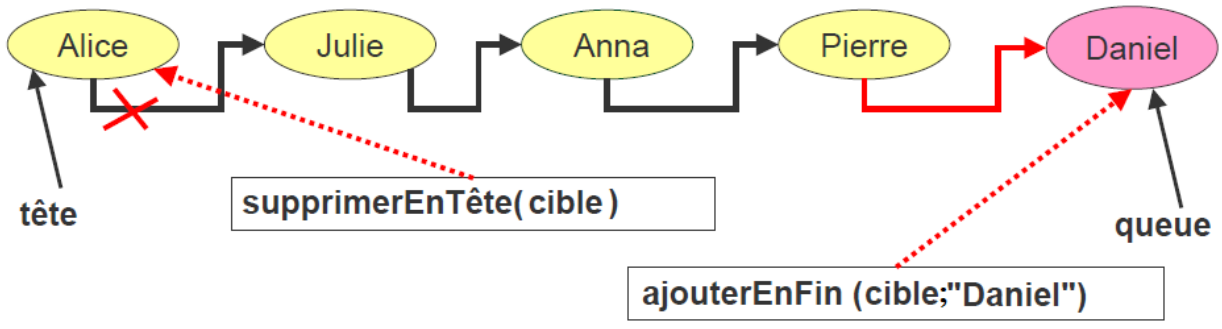
2.4 Files et Piles

Dans beaucoup d'applications, on peut se contenter de modes d'accès très restreints à la structure de données.

Avantages:

- le programme n'a pas à se préoccuper de détails de gestion (des références, par exemple)
- traitements plus simples et moins rigides (moins d'opérations)

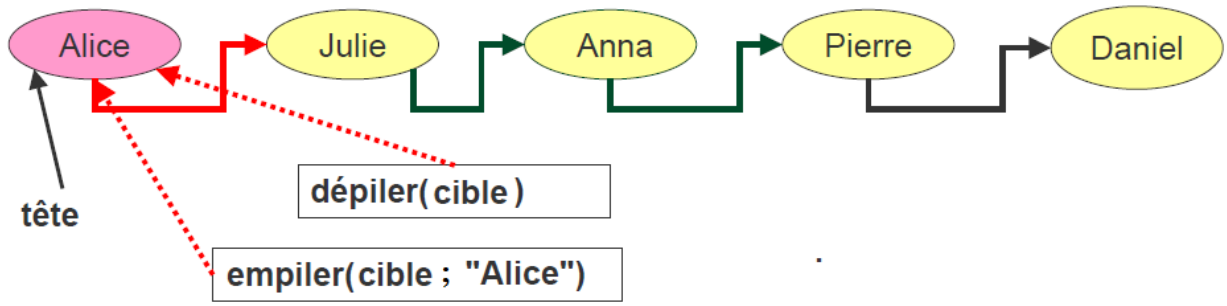
2.4.1 Représentation d'une file par une liste chaînée



- les ajouts se font en fin de file, les suppressions en tête de file
- seule l'information de la tête est accessible et traitable

→file d'attente à un guichet "premier rentré, premier sorti" (FIFO : first in first out)

2.4.2 Représentation d'une pile par une liste chaînée



- les ajouts comme les suppressions se font en tête de pile
- seule l'information de la tête est accessible et traitable

→ pile d'assiettes "dernier rentré, premier sorti" (LIFO : last in first out)

2.4.1 La structure de donnée file

Attributs :

- la tête et la queue, mais pas de curseur

Les procédures :

- infoTête(): retourne la valeur de l'information en tête de file
- vide(): indique si la file est vide
- ajouterEnFin(val) : ajoute une information en fin de file
- supprimerEnTête() : supprime (et retourne) l'information en tête de file
- saisirFile()
- afficherFile()

2.4.1.1 Définition de la structure de donnée file

Les Attributs :

tête: référence {référence à la tête de file}
queue: référence {référence à la queue de file}

Les procédures :

Fonction infoTête() retourne Info
{retourne la valeur enregistrée dans la cellule de tête. Erreur si la file est vide}
paramètre(D) cible : File

fonction vide() retourne booléen
{retourne vrai si la file est vide, faux sinon}
paramètre(D) cible : File

procédure ajouterEnFin(val)
{Crée une nouvelle cellule, y affecte la valeur val, et l'insère après la dernière cellule.
Si file était vide, elle contient maintenant l'unique cellule qui vient d'être créée.}
Paramètre (D/R) cible : File ; (D) val : Info

fonction supprimerEnTête() retourne Info
{Supprime la première cellule de la file et retourne la valeur qu'elle contient. Si la file ne contenant qu'une seule cellule, la file devient vide. Erreur si la file est vide.}
Paramètre (D/R) cible : File

Procédure saisirFile(cible : File**)**
{Saisit des valeurs (de type Info), jusqu'à une valeur d'arrêt (constante définie dans l'algorithme appelant), et crée au fur et à mesure autant de cellules que nécessaire, en y affectant les valeurs saisies.}

```
        paramètre       (R) cible : File
        variables uneVal : Info, cpt : entier
début
        saisir(uneVal) ; cpt←0
        tant que uneVal≠VALSTOP faire
                cpt←cpt + 1
                ajouterEnFin(cible ; uneVal)
                saisir(uneVal)
        ftq
        afficher("La nouvelle file contient", cpt, "cellules. ")
fin
```

Procédure afficherFile(cible : File**)**
{Affiche toutes les valeurs contenues dans la file cible.}

```
        paramètre(D) cible : File
        variablesuneVal: Info
        copieCible : File
début
        copieCible ← cible
        tant que non vide(copieCible) faire
                uneVal ← supprimerEnTête(copieCible)
                afficher(uneVal)
        ftq
fin
```

2.4.2 La structure de donnée pile

Les Attributs :

- la tête mais pas de curseur

Les procédures :

- infoTête() : retourne la valeur de l'information en tête de pile
- vide() : indique si la pile est vide
- empiler(val) : ajoute une information en tête de pile
- dépiler() : supprime (et retourne) l'information en tête de pile
- saisirPile()
- afficherPile()

2.4.2.1 Définition de la structure de donnée pile

Attributs :

tête: référence {référence à la tête de pile}

Les procédures :

Fonction infoTête() retourne Info
{retourne la valeur enregistrée dans la cellule de tête. Erreur si la pile est vide}
Paramètre (D) cible : Pile

Fonction vide() retourne booléen
{retourne vrai si la pile est vide, faux sinon}
Paramètre (D) cible : Pile

procédure empiler(val)
{Crée une nouvelle cellule, y affecte la valeur val, et l'insère en tête de pile. Si pile était vide, elle contient maintenant l'unique cellule qui vient d'être créée.}
paramètre (D/R) cible : Pile ; (D) val : Info

fonction dépiler() retourne Info
{Supprime la première cellule de la pile et retourne la valeur qu'elle contient. Si la pile ne contenant qu'une seule cellule, la pile devient vide. Erreur si la pile est vide.}
paramètre (D/R) cible : Pile

Procédure saisirPile()

{Saisit des valeurs (de type Info), jusqu'à une valeur d'arrêt (constante définie dans l'algorithme appelant), et crée au fur et à mesure autant de cellules que nécessaire, en y affectant les valeurs saisies}

```
        Paramètre       (R) cible : Pile
        Variables uneVal: Info
        cpt : entier
début
        saisir(uneVal) ; cpt←0
        tant que uneVal≠VALSTOP faire
                cpt←cpt + 1
                empiler(cible; uneVal)
                saisir(uneVal)
        ftq
        afficher("La nouvelle pile contient", cpt, "cellules.")
fin
```

Procédure afficherPile(cible : Pile)

{Affiche toutes les valeurs contenues dans la pile cible.}

```
        Paramètre       (D) cible : Pile
        variables       uneVal : Info
                        copieCible : Pile
début
        copieCible ← cible
        tant que non vide(copieCible) faire
                uneVal ← dépiler(copieCible)
                afficher(uneVal)
        ftq
fin
```