Les piles

Les piles

Définition

Une pile est une structure de données telle que :

- l'ajout d'un élément se fait au sommet de la structure,

- la suppression d'un élément se fait également au sommet de la structure.

La structure de données est appelée LIFO : "last in, first out" soit "dernier entré, premier sorti".

Les piles

Utilisation du module de gestion des listes

À l'aide d'une liste, les opérations sur une pile peuvent être réalisées comme suit :

initialiserPile	initialiser une liste vide
pileVide	vrai si la liste est vide
empiler	ajouter un élément en tête de la liste
dépiler	enlever un élément en tête de la liste

Les piles

Utilisation du module de gestion des listes

Le module de gestion de piles peut facilement se réaliser avec **le module de gestion des listes** présenté précédemment en utilisant :

- insererEnTeteDeListe()

- extraireEnTeteDeListe()

Les déclarations et les opérations sur la pile sont données ci-dessous.

Les piles

Le fichier d'en-tête des piles (pile.h)

```
#include "liste.h"

typedef  Liste Pile;

Pile*     creerPile     ();
booleen   pileVide      (Pile* p);
void      empiler       (Pile* p, Objet* objet);
Objet*    depiler       (Pile* p);
void      listerPile    (Pile* p, void (*f) (Objet*));
void      detruirePile  (Pile* p);
```

Les piles

Le module des piles

```
Pile* creerPile () {
  return creerListe ();
}


booleen pileVide (Pile* p) {
  return listeVide (p);
}


void empiler (Pile* p, Objet* objet) {
  insererEnTeteDeListe (p, objet);
}
```

Les piles

Le module des piles

```
// fournir l'adresse de l'objet en sommet de pile,
// ou NULL si la pile est vide
Objet* depiler (Pile* p) {
  if (pileVide (p)) {
    return NULL;
  } else {
    return extraireEnTeteDeListe (p);
  }
}

void listerPile (Pile* p, void (*f) (Objet*)) {
  listerListe (p, f);
}

void detruirePile (Pile* p) {
  detruireListe (p);
}
```

Utilisation du module de gestion de piles

Exercice rapide :

Ecrivez un programme principale (*main*) qui utilise une pile pour :

1) empiler trois personnes
2) afficher le contenue de la pile
3) dépiler un élément

Les piles

Utilisation du module de gestion de piles

```
void main () {

  printf ("GESTION D'UNE PILE DE PERSONNES\n");

  Pile*   pile1 = creerPile();

  empiler (pile1, creerPersonne("Alaoui",   "Karim"));
  empiler (pile1, creerPersonne("Alaoui",   "Ali"));
  empiler (pile1, creerPersonne("Alaoui",   "Sarah"));

  printf ("Valeurs dans la pile : du sommet vers la base\n");
  listerPile (pile1, ecrirePersonne);

  printf ("\nValeur dépilée : ");
  Personne* p = (Personne*) depiler (pile1);
  if (p!=NULL) ecrirePersonne (p);

}
```

Les files

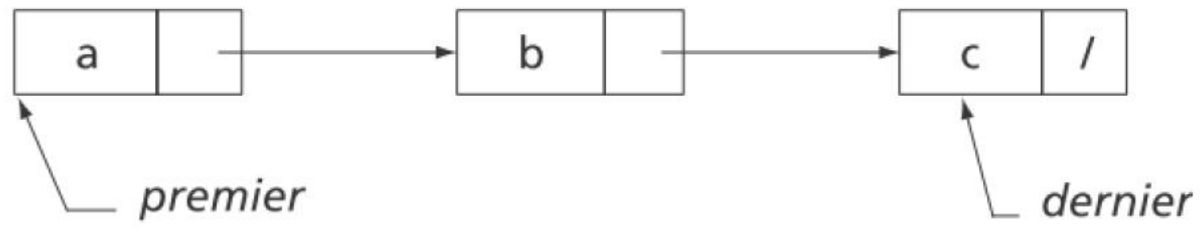
Les files

Définition

Une file d'attente est une structure de données telle que :

- l'ajout d'un élément se fait en fin de file d'attente,
- la suppression d'un élément se fait en début de file d'attente.

La structure de données est appelée FIFO : « first in, first out » soit « premier entré, premier sorti ».



Les files

Exercice :

En vous inspirant des programmes précédents pour liste, pile et du fichier d'en-tête *file.h* ci-dessus, écrire *file.c*

```
#include "liste.h"

typedef Liste File;

File*     creerFile     ();
booleen   fileVide      (File* file);
void      enFiler       (File* file, Objet* objet);
Objet*    deFiler       (File* file);
void      listerFile    (File* file, void (*f) (Objet*));
void      detruireFile  (File* file);
```

Les files

```
File* creerFile () {
  return creerListe ();
}

booleen fileVide (File* file) {
  return listeVide (file);
}

void enFiler (File* file, Objet* objet) {
  insererEnFinDeListe (file, objet);
}

Objet* deFiler (File* file) {
  if (fileVide (file)) {
    return NULL;
  } else {
    return extraireEnTeteDeListe (file);
  }
}

void listerFile (File* file, void (*f) (Objet*)) {
  listerListe (file, f);
}

void detruireFile (File* file) {
  detruireListe (file);
}
```