Chapitre 12 : LES PILES

Introduction

Une pile est une structure de données complémentaire qui fonctionne avec le principe du LIFO (Last In First Out). Une pile est accessible à travers la variable SOMMET. Les piles sont très utilisées dans la programmation des ressources de l'ordinateur (programmation système) mais également dans la vérification et l'évaluation des expressions arithmétiques et logiques au niveau de l'UAL (Unité Arithmétique et Logique). Les piles sont réalisées soit en utilisant les POINTEURS soit en utilisant les TABLEAUX. La manipulation d'une pile est basée sur l'utilisation des primitives.

I- Les primitives associées à la pile

Les primitives sont les modules suivants :

1- La primitive initPile()

Elle permet d'initialiser les arguments d'une pile.

2- La primitive pileVide()

Elle recoit les arguments d'une pile puis renvoie vrai si la pile est vide et faux dans le cas contraire.

3- La primitive pilePleine()

Elle recoit les arguments d'une pile puis renvoie vrai si la pile est pleine et faux dans le cas contraire.

4- La primitive empiler()

Elle recoit une valeur à ajouter dans la pile et les arguments de la pile puis place cette valeur au sommet de la pile. Pour empiler, il faut que la pile ne soit pas pleine.

5- La primitive depiler()

Elle recoit les paramètres d'une pile puis enlève la valeur située au sommet de la pile pour la sauvegarder dans une variable pour d'éventuels traitements. Pour dépiler, il faut que la pile ne soit pas vide.

II- Réalisation des piles avec les pointeurs

La pile réalisée avec les pointeurs à la même description que la liste monodirectionnelle mais les maillons sont chainés du dernier au premier.

A- Déclaration

a- Syntaxe

Type nomPile = ↑Structure

Debut

info(s) : type(s)

suiv : nomPile

Fin

var Sommet : nomPile

b- Exemple 1

Déclarer une pile de personnes réalisées avec les pointeurs. Personne (nom, prenom, age)

Type PERSONNE = structure

Debut

nom, prenom : chaine

age : entier

Fin

Type PilePersonne = ↑Structure

Debut

info : PERSONNE

suiv : PilePersonne

Fin

var Sommet : PilePersonne

c- Exemple 2

Déclarer une pile d'entiers réalisée avec les pointeurs

Type Pile = ↑Structure

Debut

info : entier

suiv : Pile

Fin

var Sommet : Pile

B- Réalisation des primitives avec les pointeurs

1- La primitive initPile()

Pour initialiser les arguments d'une pile sous forme de pointeur, il faut affecter à Sommet la valeur NIL.

Exercice d'application :

Soit une pile d'entiers réalisée avec les pointeurs, écrire un module qui réalise la primitive initPile()

Type Pile = ↑Structure

Debut

info : entier

suiv : Pile

Fin

Var Sommet : Pile

Procedure initPile (D/R Sommet : Pile)

Debut

Sommet ← NIL

Fin

2- La primitive pileVide()

Une pile est considérée comme vide si Sommet = NIL. Dans ce cas, la primitive renvoie VRAI sinon elle renvoie FAUX.

Exercice d'application :

Soit une pile d'entiers réalisée avec les pointeurs, écrire un module qui réalise la primitive pileVide()

Type Pile = ↑Structure

```
Debut
    info : entier
    suiv : Pile
Fin
Var Sommet : Pile
Fonction pileVide (Donnee Sommet : Pile) : booleen
Debut
    Si (Sommet = NIL) Alors
        Retourner vrai
    Sinon
        Retourner faux
    FinSi
Fin
```

*3- **La primitive pilePleine()***

Une pile est pleine si toutes les ressources mémoires sont allouées. Dans ce cas, la primitive renvoie VRAI sinon elle renvoie FAUX.

Exercice d'application :

Soit une pile d'entiers réalisée avec les pointeurs, écrire un module qui réalise la primitive pilePleine()

```
Type Pile = ↑Structure
Debut
    info : entier
    suiv : Pile
Fin
Var Sommet : Pile
Fonction PilePleine (Donnee Sommet : Pile) : booleen
var p : Pile
```

```
Debut
    Allouer(p)
    Si (p = NIL) Alors
        Retourner Vrai
    Sinon
        Liberer(p)
        Retourner Faux
    FinSi
Fin
```

4- La primitive empiler()

Elle recoit une valeur et les arguments d'une pile puis ajoute la valeur au sommet de la pile si la pile n'est pas pleine.

Exercice d'application :

Soit une pile d'entiers réalisée avec les pointeurs, écrire un module qui réalise la primitive empiler()

```
Type Pile = ↑Structure
Debut
    info : entier
    suiv : Pile
Fin
Var Sommet : Pile
Procedure empiler(Donnee val : entier
                  Donnee/Resultat Sommet : Pile)
var pval : Pile
Debut
    Si (pilePleine (Sommet) = vrai) Alors
        Ecrire "Impossible d'ajouter car la pile est pleine"
    Sinon
```

```
        Allouer(pc)
        pc↑.info ← val
        pc↑.suiv ← NIL
        Si (pileVide(Sommet)  = vrai) Alors
                Sommet ← pc
        Sinon
                pc↑.suiv ← Sommet // Chainage
                Sommet ← pc // Marquage
        FinSi
    FinSi
Fin
```

5- La primitive depiler()

Elle recoit les arguments d'une pile puis enlève la valeur située au sommet de la pile pour la sauvegarder dans une variable pour d'éventuels traitements.

Exercice d'application :

Soit une pile d'entiers réalisée avec les pointeurs, écrire un module qui réalise la primitive depiler().

```
Type pile = ↑Structure
Debut
    info : entier
    suiv : Pile
Fin
Var Sommet : Pile
Procedure depiler (Donnee/Resultat Sommet : Pile
                   Resultat val : entier)
var pval : Pile
Debut
```

```
    Si (pileVide(Sommet) = vrai) Alors
        Ecrire "Impossible d'enlever car la pile est vide"
    Sinon
        val ← Sommet↑.info
        pval ← Sommet
        Si (Sommet↑.suiv = NIL) Alors
            initPile(Sommet)
        Sinon
            Sommet ← Sommet↑.suiv
        FinSi
        Liberer(pval)
    FinSi
Fin
```

Exercice d'application 1 :

Soit un fichier de produits à organisation séquentielle, écrire un module qui transfère les produits dont le nom commence par une voyelle et finit également par une lettre autre qu'une voyelle dans une pile réalisée sous forme de pointeurs.

Produit (code, nom, categorie, prix unitaire, quantite)

```
Type PRODUIT = structure
Debut
    code, nom, categorie : chaine
    prixUnitaire : entier
    quantite : reel
Fin
Type FProduits = Fichier PRODUIT
Organisation : sequentielle
```

```
Type PileProduits = ↑structure
Debut
        info : PRODUIT
        suiv : PileProduits
Fin
var Sommet : PileProduits
    FP : FProduits
Procedure Application1 (Donnee FP : FProduits
                        Resultat Sommet : PileProduits)
var p : PRODUIT
    voyelles : chaine
Debut
   Ouvrir (FP) Lecture
   initPile(Sommet)
   voyelles ← "AOIUEYaoiuey"
   Repeter
         Lire(FP,p)
         Si (EOF(FP) = faux) Alors
               Si (Rang(voyelles, sschaine(p.nom, 1, 1))!=0 et
                   Rang(voyelles, sschaine(p.nom, long(p.nom), 1))=0) ALors
                       empiler(p, Sommet)
               FinSi
         FinSi
   Jusqua(EOF(FP)=vrai)
   Fermer(FP)
Fin
```

Exercice d'application 2 :

Soit une pile d'entiers, écrire un module qui détermine le nombre de nombre premiers ou parfaits de la pile.

```
Type Pile = ↑Structure
Debut
        info : entier
        suiv : Pile
Fin
var Sommet : Pile
Fonction Application2 (Donnee Sommet : Pile) : entier
var val, i, cpt, som, nb : entier
Debut
        nb ← 0
        TantQue (pileVide(Sommet) = faux) Faire
                depiler(Sommet, val)
                cpt ← 1
                som ← 0
                Pour i ← 1 à val div 2 Faire
                        Si (val mod i = 0) Alors
                                cpt ← cpt + 1
                                som ← som + i
                        FinSi
                FinPour
                Si ((cpt = 2 ou s = val) et val >= 2) Alors
                        nb ← nb + 1
                FinSi
        FinTantQue
        Retourner nb
FIN
```