CHAPITRE : LES PILES

INTRODUCTION

Une PILE est une structure de données complémentaires 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 de ces 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 pile Pleine():

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

4- La primitive Empiler ():

Elle reçoit 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 Dépiler ():

Elle reçoit les paramètres d'une pile puis enlevé la valeur située au sommet de la pile pour la sauvegarder dans une variable pour d'éventuel traitement. Pour dépiler il faut que la pile ne soit pas vide.

Mme Ndacke GUEYE
+221 77 485 88 92
ndackeg@gmail.com

II- Réalisation des piles avec les pointeurs

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

A- Déclaration

Syntaxe

```
Type nomPile = ↑Structure
Debut
        info(s):type(s)
        SUIV:nomPile
Fin
Var Sommet:nomPile
```

Exemple 1: Déclarer une pile de personnes réalisées avec les pointeurs. Personne (nom, prénom, âge)

```
Type PERSONNE = structure
Debut
        nom,prenom:chaine
        age:entier
Fin
Type PilePersonne = ↑Structure
Debut
        Info : PERSONNE
        SUIV:PilePersonne
Fin
var Sommet:PilePersonne
```

Exemple 2: Déclarer une pile d'entiers réalisée avec les pointeurs

```
Type Pile = ↑Structure
Début
```

Mme Ndacke GUEYE
+221 77 485 88 92
ndackeg@gmail.com

```
        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()

Mme Ndacke GUEYE
+221 77 485 88 92
ndackeg@gmail.com

```
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, écrire un module qui réalise la primitive pile Pleine()

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

Mme Ndacke GUEYE
+221 77 485 88 92
ndackeg@gmail.com

```
        Si p = Nil Alors
                Retourner Vrai
        Sinon
                Retourner Faux
        FinSi
        Liberer(p)
Fin
```

4- La primitive Empiler()

Elle reçoit 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
Début
        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
```

Mme Ndacke GUEYE
+221 77 485 88 92
ndackeg@gmail.com

```
                Si pileVide(Sommet)  = Vrai Alors
                        Sommet <- pc
                Sinon
                        pc↑.SUIV <- sommet // chainage des 2 Dernier
                        sommet <- pc //Marquage du dernier element
                FinSi
        FinSi
Fin
```

5- La primitive Depiler()

Elle reçoit les arguments d'une pile puis enlevé la valeur située au sommet de la pile pour la sauvegarder dans une variable pour d'éventuel traitement.

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
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
```

Mme Ndacke GUEYE
+221 77 485 88 92
ndackeg@gmail.com

```
                    initPile(sommet)
                Sinon
                    sommet <- sommet↑.SUIV
                FinSi
                Liberer(pval)
        FinSi
Fin
```

Exercice d'application : Soit un fichier de produits a 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 pointeur.

Produit (code, nom, catégorie, prix unitaire, quantité)

Mme Ndacke GUEYE
+221 77 485 88 92
ndackeg@gmail.com

Exercice d'application 2: soit une pile d'entiers, écrire un module qui détermine le nombre de nombre premiers ou parfait de la pile

Mme Ndacke GUEYE
+221 77 485 88 92
ndackeg@gmail.com

III- REALISATION DES PILES AVEC LE TABLEAUX

Une pile réalisée sous forme de tableau hérite des contraintes du tableau. La taille de la pile est égale à celle du tableau. La variable sommet est de type entier et elle contient la position de la dernière cellule remplie.

A- Déclaration de la pile

```
Const Taille=valeur
Type nomPile = Tableau[1..Taille] typeValeur
var nomVariablePile:nomPile
var sommet:entier
```

Exemple 1: pile d'entiers réalisée sous la forme d'un tableau d'un tableau de 125 cellules.

```
Const N=125
Type Pile=Tableau[1..N]Entier
```

Mme Ndacke GUEYE
+221 77 485 88 92
ndackeg@gmail.com

```
var vpile:Pile
var sommet:entier
```

Exemple 2: pile de produits réalisée sous la forme d'un tableau de 150 produits.

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

```
Const N=150
Type produit = Structure
Debut
code,nom,categorie:chaine
prixUnitaire,quantite:entier
Fin
Type PileProduit=Tableau[1..N]produit
var vpileProd:PileProduit
var sommet:entier
```

B- Réalisation des primitives avec les tableaux

1- La primitive initpile()

La pile sous forme de tableau est vide si sommet = 0.

Exercice d'application : soit une pile d'entiers réalisée sous la forme d'un tableau de 750 cellules, écrire un module qui réalise la primitive initPile().

```
Const N=750
Type pile=Tableau[1..N]Entier
var vpile:pile
var sommet:entier
Procedure initPile(D/R sommet:entier)
Debut
sommet <- 0
Fin
```

Mme Ndacke GUEYE
+221 77 485 88 92
ndackeg@gmail.com

2- la primitive pilevide()

La pile sous forme de tableau est vide si sommet = 0. Dans ce cas, la primitive renvoie VRAI sinon elle renvoie FAUX

Exercice d'application : soit une pile d'entiers réalisée sous la forme d'un tableau de 750 cellules, écrire un module qui réalise la primitive pileVide().

```
Const N=750
Type pile=Tableau[1..N]Entier
var vpile:pile
var sommet:entier
Fonction pileVide(Donnee sommet:entier):Booleen
Debut
Si sommet = 0 Alors
retourner Vrai
Sinon
retourner Faux
FinSi
Fin
```

3- la primitive pilePleine()

la pile est pleine quant sommet = N qui est la taille de la pile. Dans ce cas, la primitive renvoie Vrai sinon elle renvoie Faux.

Exercice d'application : soit une pile d'entiers réalisée sous la forme d'un tableau de 750 cellules, écrire un module qui réalise la primitive pilePleine()

```
Const N=750
Type pile=Tableau[1..N]Entier
var vpile:pile
var sommet:entier
Fonction pilePleine(Donnee sommet,N:ENtier):Booleen
Debut
```

Mme Ndacke GUEYE
+221 77 485 88 92
ndackeg@gmail.com

```
Si sommet = N Alors
retourner Vrai
Sinon
retourner Faux
FinSi
```

4- la primitive Empiler ()

Elle reçoit une valeur et les paramètres de la pile puis place la valeur au sommet de la pile si la pile n'est pas pleine.

Exercice d'application : soit une pile d'entiers réalisée sous la forme d'un tableau de 750 cellules, écrire un module qui réalise la primitive empiler().

```
Const N=750
Type pile=Tableau[1..N]Entier
var vpile:pile
var sommet:entier
Procedure empiler(Donnee val:Entier
Donnees/Resultats vpile:Pile, N,sommet:Entier)
Debut
Si pilePleine(sommet,N)=Vrai Alors
Ecrire "Impossible d'ajouter car la pile est pleine"
Sinon
Si pileVide(sommet)=vrai Alors
sommet <- 1
vpile[sommet] <- val
Sinom
sommet <- sommet + 1
vpile[sommet] <- val
FinSi
FinSi
Fin
```

Mme Ndacke GUEYE
+221 77 485 88 92
ndackeg@gmail.com

5- la primitive depiler ()

Elle reçoit une pile puis enlevé la valeur qui est au sommet de la pile pour la sauvegarder dans une variable. La pile devient vide si elle était composée d'un élément.

Exercice d'application : soit une pile d'entiers réalisée sous la forme d'un tableau de 750 cellules, écrire un module qui réalise la primitive depiler().

```
Const N=750
Type pile=Tableau[1..N]Entier
var vpile:pile
var sommet:entier
Procedure depiler(D/R vpile:pile, N,sommet:Entier
Resultats val:entier)
Debut
si pilevide(sommet)=vrai alors
ecrire "impossible d'enlever car la pile est vide"
Sinon
val <- vpile[sommet]
Si sommet = 1 Alors
initpile(sommet)
sinon
sommet <- sommet - 1
FinnSi
FinSi
Fin
```

Mme Ndacke GUEYE
+221 77 485 88 92
ndackeg@gmail.com