Université de M'hamad Bougara de Boumerdès
Faculté des Sciences
Département de Mathématiques
Deuxième Année Licence
Responsable du Module:
Recherche Opérationnelle
Mr. M. BEZOUI

Durée: 1H30'

Semestre 03

Corrigé de la série de TD N°03 de Structures de Données

Exercice $n°1$: Fonctions sur les listes chaînées

Soit L une liste chaînée, des villes de Boumerdès, Dont le pointeur vers la tête est T, ayant dans chaque noeud, le nom de la ville, la superficie de la ville et le nombre d'habitants dans la ville.

Écrire des fonctions ou procédures qui permettent de:

1. Récupérer toutes les villes ayant plus (strictement) de 10.000 habitants.
2. Compter le nombre de villes
3. Ajouter une ville
4. Supprimer la première ville.
5. Supprimer la dernière ville.
6. Supprimer toutes les villes ayant plus de 10.000 habitants.

Exercice $n°02$: Piles

Soit P une Pile représentée par une liste chaînée, des villes de Boumerdès, Dont le pointeur vers la tête est P, ayant dans chaque noeud, le nom de la ville, la superficie de la ville et le nombre d'habitants dans la ville.

Écrire des fonctions ou procédures qui permettent de:

1. Ajouter la ville de Corso.
2. Supprimer la troisième ville.
3. Supprimer toutes les villes dont le nom commence par B.

Exercice $n°03$: Files

Soit F une File représentée par une liste chaînée, des villes de Boumerdès, Dont le pointeur vers la tête est F, ayant dans chaque noeud, le nom de la ville, la superficie de la ville et le nombre d'habitants dans la ville.

☛ Même questions de l'exercice numéro 2.

Correction de l'exercice $n°4$

Soit P une pile d'entiers. Écrire les fonctions pour determiner:

a/ Le nombre d'éléments.

b/ La valeur maximale.

c/ La valeur minimale.

Algorithme 1 nbr_element

```
fonction nbr_element(P:Pile)
var P1:Pile;N,val:entier;
début
N ⟵ 0
CréerPile(P1);
tantque Pile_vide(P)=faux faire
   dépiler(P,val);
   empiler(P1,val);
   N ⟵ N + 1
fin tantque
tantque Pile_vide(P1)=faux faire
   dépiler(P1,val);
   empiler(P,val);
fin tantque
renvoyer(N);
fin;
```

Algorithme 2 max

```
fonction maximum(P:Pile)
var P1:Pile;M,val:entier;
début
CréerPile(P1);
si Pile_vide(P)=faux alors
   dépiler(P,val);
   empiler(P1,val);
finsi
M ⟵ val
tantque Pile_vide(P)=faux faire
   dépiler(P,val);
   empiler(P1,val);
   si val > M alors
      M ⟵ val
   finsi
fin tantque
tantque Pile_vide(P1)=faux faire
   dépiler(P1,val);
   empiler(P,val);
fin tantque
renvoyer(M);
fin;
```

Correction de l'exercice $n°5$

On dispose d'une pile de nombres entiers, ordonnés suivant l'ordre décroissant des valeurs.

Écrire une procédure qui insère une valeur val à la place qu'il faut (pour garder l'ordre décroissant) dans les deux cas suivants:

a/ Si elle n'existe pas;

b/ Même si elle existe, si la même chose avec le premier cas, mais, cette fois, au lieu de faire "si $x > Val$", vous faites "si $x \geq Val$"

Algorithme 3 min

```
fonction minimum(P:Pile)
var P1:Pile;MN,val:entier;
début
CréerPile(P1);
si Pile_vide(P)=faux alors
   dépiler(P,val);
   empiler(P1,val);
   MN ⟵ val
   tantque Pile_vide(P)=faux faire
      dépiler(P,val);
      empiler(P1,val);
      si val < MN alors
         MN ⟵ val
      finsi
   fin tantque
   tantque Pile_vide(P1)=faux faire
      dépiler(P1,val);
      empiler(P,val);
   fin tantque
finsi
renvoyer(MN);
fin;
```

Exercice $n°6$

Soit P une pile d'entiers. Écrire une procédure qui permet de trier ses éléments selon un ordre croissant.

Avec les files maintenant!

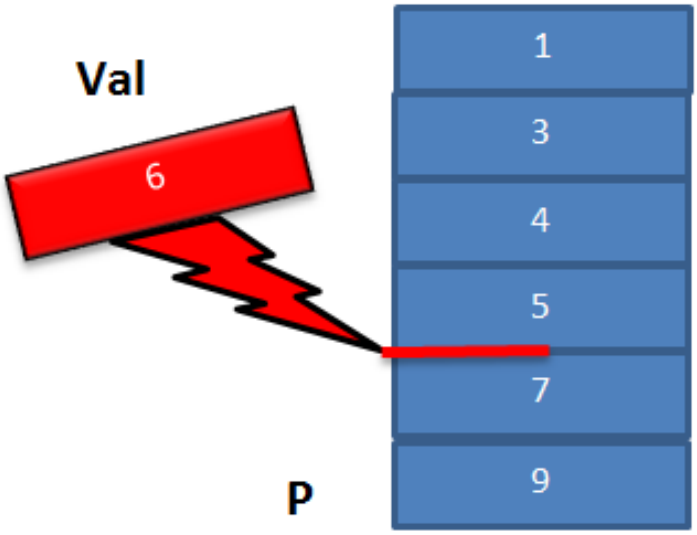

Figure 1: Représentation graphique d'un exemple de l'exo 5

Correction de l'exercice $n°4$

Soit F une File d'entiers. Écrire les fonctions pour determiner:

a/ Le nombre d'éléments.

b/ La valeur maximale.

c/ La valeur minimale.

Algorithme 4 inserer_a

```
procédure inser_a(P:Pile,Val:entier)
{Regardez la figure 1, pour une représentation graphique de la pile!}
var P1:Pile;R,x:entier;
début
R ⟵ 0
CréerPile(P1);
tantque (Pile_vide(P)=faux)et(R=0) faire
   dépiler(P,x);
   si x > Val alors
      empiler(P,x);
      empiler(P,Val);
      R ⟵ 1
   sinon
      empiler(P1,x);
   finsi
fin tantque
tantque Pile_vide(P1)=faux faire
   dépiler(P1,x);
   empiler(P,x);
fin tantque
fin;
```

Correction de l'exercice $n°5$

On dispose d'une file de nombres entiers, ordonnés suivant l'ordre décroissant des valeurs.

Écrire une procédure qui insère une valeur val à la place qu'il faut (pour garder l'ordre décroissant) dans les deux cas suivants:

a/ Si elle n'existe pas;

b/ Même si elle existe, si la même chose avec le premier cas, mais, cette fois, au lieu de faire "si $x > Val$", vous faites "si $x \geq Val$"

Algorithme 5 Tri1

```
procédure Tri(P:Pile)
var P1:Pile; R,i,K:entier; Tab[1,..,100]: tableau d'entier;
début
{Extraction des élément de la pile dans un tableau!}
K ⟵ 0
tantque (Pile_vide(P)=faux) faire
   dépiler(P,x);
   K ⟵ K + 1;
   Tab[i] ⟵ x;
fin tantque
pour i allant de 1 à K faire
   pour j allant de i+1 à K faire
      si Tab[i] > Tab[j] alors
         tmp ⟵ Tab[i]
         Tab[i] ⟵ Tab[j]
         Tab[j] ⟵ tmp
      finsi
   fin pour
fin pour
pour i allant de 1 à K faire
   empiler(P,Tab[i]);
fin pour
fin;
```

Algorithme 6 nbr_element

```
fonction nbr_element(F:File)
var F1:File;N,val:entier;
début
CréerFile(F1);
N ⟵ 0
tantque File_vide(F)=faux faire
   défiler(F,val);
   enfiler(F1,val);
   N ⟵ N + 1
fin tantque
tantque File_vide(F1)=faux faire
   défiler(F1,val);
   enfiler(F,val);
fin tantque
renvoyer(N);
fin;
```

Exercice $n°6$

Soit F une File d'entiers. Écrire une procédure qui permet de trier ses éléments selon un ordre croissant.

Algorithme 7 max

```
fonction maximum(F:File)
var F1:File;i,K,M,val:entier;
début
si File_vide(F)=faux alors
   défiler(F,val);
   enfiler(F,val);
   M ⟵ val
   K ⟵ nbr_element(F)
   pour i allant de 2 à K faire
      défiler(F,val);
      enfiler(F,val);
      si val > M alors
         M ⟵ val
      finsi
   fin pour
sinon
   écrire("j peux pas!!!");
finsi
renvoyer(M);
fin;
```

Algorithme 8 min

```
fonction minimum(F:File)
var F1:File;i,K,M,val:entier;
début
si File_vide(F)=faux alors
   défiler(F,val);
   enfiler(F,val);
   M ⟵ val
   K ⟵ nbr_element(F)
   pour i allant de 2 à K faire
      défiler(F,val);
      enfiler(F,val);
      si val < M alors
         M ⟵ val
      finsi
   fin pour
sinon
   écrire("j peux pas!!!");
finsi
renvoyer(MN);
fin;
```

Pour le reste, ... Je vous laisses vous débrouiller seuls ... Bon courage. Faites partager avec vos camarades S'il vous plait.

Mr. BeZoui

Algorithme 9 inserer_a

```
procédure inser_a(F:File,Val:entier)
var F1:Pile;R,i,x:entier;
début
R ⟵ 0
i ⟵ 0
K ⟵ nbr_element(F)
pour i allant de 1 à K faire
   défiler(F,x);
   si x < Val et (R=0) alors
      enfiler(F,Val);
      enfiler(F,x);
      R ⟵ 1
   sinon
      enfiler(F,x);
   finsi
   i ⟵ i + 1;
fin pour
fin;
```

Algorithme 10 exo6files

```
procédure Tri(F:File)
var R,i,K:entier; Tab[1,..,100]: tableau d'entier;
début
{Extraction des élément de la file dans un tableau!}
K ⟵ 0
tantque (File_vide(F)=faux) faire
   défiler(F,x);
   K ⟵ K + 1;
   Tab[i] ⟵ x;
fin tantque
pour i allant de 1 à K faire
   pour j allant de i+1 à K faire
      si Tab[i] > Tab[j] alors
         tmp ⟵ Tab[i]
         Tab[i] ⟵ Tab[j]
         Tab[j] ⟵ tmp
      finsi
   fin pour
fin pour
pour i allant de 1 à K faire
   enfiler(F,Tab[i]);
fin pour
fin;
```