LES PILES ET LES FILES

Lilia AYADI ZRIBI
Dalila CHERITI

Application1 : Les Piles

On considère une liste simplement chaînée d'entiers. Vous devez disposez dans votre programmes des fonctions suivantes :

- Fonction Inser_TeteE permettant d'ajouter un élément en tête de la liste,
- Fonction Afficher_rec permettant l'affichage des éléments de la liste dans l'ordre d'insertion.

Pour afficher une liste simplement chaînée d'entiers dans l'ordre inverse (càd du dernier élément au premier), on utilise la fonction récursive suivante :

```
Procédure Afficher_Rec_Inv(E Tete : ↑Element) {
  Si (Tete <> NIL) Alors
     Afficher_Rec_Inv(Tete↑.Suiv)
     Ecrire(Tete↑.Info)
Finsi
Fin Afficher_Rec_Inv
```

Application1 : Les Piles

1- On vous demande d'écrire ***la version itérative*** de cette procédure en utilisant une pile d'adresses pour la sauvegarde des adresses relatives aux éléments visités de la liste suite à un parcours normal.

2- Crée le programme principale qui permet de gérer une liste simplement chaînée d'entiers en utilisant le menu suivant :

- Ajouter autant d'élément en tête de la liste que l'utilisateur désire,
- Afficher les éléments de la liste récursivement dans l'ordre d'insertion,
- Afficher les éléments de la liste récursivement dans l'ordre inverse,
- Afficher les éléments de la liste en utilisant les Piles dans l'ordre inverse,
- Quitter.

En Algorithmique | Traduction en C

```
//Déclaration de la structure
 Type Elem_Liste = Structure
      Info : Entier
      Suiv : ↑Elem_Liste
 Fin Elem_Liste

 Type Elem_Pile = Structure
      AdrEltList : ↑Elem_Liste
      Suiv: ↑Elem_Pile
 Fin Elem_Pile



//Ajouter un élément à la pile
 Procédure Empiler(E/S Sommet : ↑Elem_Pile, E Adr :
 ↑Elem_Liste)
 Var P : ↑Elem_Pile
 Début
      Allouer(P)
      P↑.AdrEltList ← Adr
      P↑. Suiv ← Sommet
      Sommet ← P
 Fin Empiler
```

En Algorithmique

Traduction en C

```
//Supprimer un élément de la pile
Procédure Depiler(E/S Sommet : ↑Elem_Pile)
Var P : ↑Elem_Pile
Début
Si (Sommet = NIL) Alors
      Ecrire("Pile déjà Vide!!!!")
Sinon
   P ← Sommet
   Sommet ← Sommet↑.Suiv
   Libérer (P)
FinSi
Fin Dépiler
```

En Algorithmique

```
//Afficher récursivement les éléments de la liste
dans l'ordre inverse
Procédure Afficher_Rec_Inv(E Tete : ↑Element) {
    Si (Tete <> NIL) Alors
        Afficher_Rec_Inv(Tete↑.Suiv)
        Ecrire(Tete↑.Info)
    Finsi
Fin Afficher_Rec_Inv
```

Traduction en C

En Algorithmique

Traduction en C

```
//Afficher les éléments de la liste dans l'ordre
inverse avec les piles
Procédure Affiche_Liste(Tete :↑Elem_Liste )
Var P : ↑Elem_Liste
      Sommet : ↑Elem_Pile
Début
       P ← Tete
       Sommet ← NIL
       Tant que (P <> NIL) Faire
          Sommet ← Empiler(Sommet,P)
          P ← P↑.Suiv
       FinTQ
       Tant que (Sommet <> NIL) Faire
          Ecrire( Sommet↑.AdrEltList↑.Info)
          Depiler(Sommet)
       FinTQ
Fin Affiche_Liste
```

Application2 : Les Files

- Dans les systèmes d'exploitation, l'ordonnanceur (scheduler) choisit, selon un algorithme bien déterminé (tels que FIFO, SJF, Round-Robin, etc.), l'ordre d'exécution des processus sur le processeur d'un ordinateur.
- Considérons un ordonnanceur qui suit l'algorithme « Round-Robin », appelé aussi « Tourniquet », dont le principe est que les processus voulant être exécutés se partagent le processeur à tour de rôle et pendant un temps fini. Chaque processus dont son laps de temps se termine laisse sa place au processus suivant pour rejoindre la dernière place dans la file d'attente si son temps d'exécution n'est pas encore fini et quitte la file d'attente dans le cas contraire. Plus formellement :
- Tout nouveau processus est ajouté en queue de la file d'attente.
- L'utilisation du processeur ne peut pas dépasser un quantum de temps. Chaque processus dispose du même quantum de temps que les autres.
- Un processus qui vient de finir d'utiliser le processeur (quantum écoulé) rejoint la fin de la file et attend son tour de nouveau jusqu'à épuisement de son temps d'exécution.
- ***Un processus qui termine son travail sort de la file.***

Dans le but de gérer le processeur en question, nous proposons de gérer une variable constante Quantum = 5 et la structure d'un élément de la file d'attente suivante :

```
Global Quantum = 5
Type Processus = Structure
   NumProc : Entier
   NomProc : Chaîne[30]
   DureeExec : Entier
   Svt : ↑Processus
Fin Processus
```

Travail à faire

- Une fonction ***CréerFile*** permettant de créer une file d'attente vide.
- Une fonction ***CréerElement*** qui crée un nouvel élément représentant un processus de la file.
- Une fonction ***Enfiler*** qui ajoute un nouveau processus à la file d'attente.
- Une procédure ***Defiler*** qui élimine le processus totalement servi de la file d'attente.
- Une fonction ***Attarder*** qui défile un processus si son temps d'exécution est écoulé. Dans le cas contraire, elle le rattache à la fin de la file pour continuer son exécution. Dans ce cas, son temps d'exécution doit être décrémenté du quantum de temps écoulé.

En Algorithmique	Traduction en C

```
//Déclaration de la structure
Global Quantum = 5
Type Processus = Structure
        NumProc : Entier
        NomProc : Chaîne[30]
        DureeExec : Entier
        Svt : ↑Processus
Fin Processus


Type File = Structure
        Tete : ↑Elt_File
        Queue : ↑Elt_File
Fin File
```

En Algorithmique

Traduction en C

```
//Créer une file vide
Procédure CréerFile (S F:File)
Début
   F.Tête ← Nil
   F.Queue ← Nil
Fin CréerFile


//Créer un élément pour l'ajouter dans la file
Procédure CréerElément (S Pr : ↑Processus, E Num :
Entier, Nom : Chaîne[30], Durée : Entier )
Début
   Allouer(Pr)
   Pr↑.NumProc ← Num
   Pr↑.NomProc ← Nom
   Pr↑.DuréeExécution ← Durée
   Pr↑.Svt ← Nil
Fin CréerElément
```

En Algorithmique

```
//Ajouter un élément dans la File
Procédure Enfiler(E/S F : File, E Pr : ↑Processus)
Début
  Si (F.Tête = Nil) Alors
      F.Tête ← Pr
      F.Queue ← Pr
  Sinon
      F.Queue↑.Svt ← Pr
      F.Queue ← Pr
  Fin Si
Fin Enfiler

//Supprimer un élément de la file
Procédure Défiler (E/S F:File)
Var P : ↑ Processus
Début
        P ←F.Tête
        Si (F.Tête = Nil et F.Queue=Nil) Alors
            Ecire("File Vide!!!!")
        Fin Si
        F.Tête  ← F.Tête↑.Svt
        Free(P)
Fin Défiler
```

Traduction en C

En Algorithmique

```
//Ajouter un élément dans la File
Procédure Attarder (E/S F:File)
    Var P : ↑ Processus
Début
  Si (F.Tête↑.DuréeExécution - Quantum <= 0) Alors
      Défiler (F)
  Sinon
      F.Tête↑.DuréeExécution ← F.Tête↑.DuréeExécution
– Quantum
      P  ← F.Tête
      F.Tête  ← F.Tête↑.Svt
      F.Queue↑.Svt  ← P
      P↑.Svt  ← Nil
      F.Queue  ← P
  Fin Si
Fin Attarder
```

Traduction en C

En Algorithmique

Traduction en C

```
//Afficher les éléments d'une File
Procédure Afficher(E F : File)
Var P : ↑ Processus
Début
        P ← F.Tete
        Tant que(P <> NIL) Faire
                Ecrire(P↑. NumProc, " , ", P↑. NomProc, " , ",
        P->DureeExec)
                P ← P ↑.Svt
        FinTQ
Fin Afficher
```

En Algorithmique

```
//Programme principal
Algorithme Principal
Var F : File
P1,P2,P3 : ↑Processus
Début
        CreerFile(F);
        CreerElement(P1,1,"Processus1",11);
        Enfiler(F,P1);
        CreerElement(P2,2,"Processus2",21);
        Enfiler(F,P2);
        CreerElement(P3,3,"Processus3",4);
        Enfiler(F,P3);
        Tant que(F.Tete<>Nil) Faire
                Afficher(F);
                Attarder(F);
        Fin TQ
Fin
```

Traduction en C