Série n°4 : Les piles et les files
TD N°1 : Les piles

Exercice 1: Ecrire les actions paramétrées suivantes :

a. *remplir(n) : pile ;* qui remplit une pile P avec n entiers données et la retourne,
b. *affiche_pile(P) ;* qui affiche les éléments de la pile P,
c. *recherche(P, val)* ; qui recherche une valeur val donnée dans une pile P,
d. *supprim(P, k) ;* qui supprime de la pile P l'élément qui se trouve à la position k donnée,
e. *insert(P, k, val) ;* qui insère un élément donné dans la pile P à la position k donnée,
f. *fusion(A, B) ;* qui fusionne deux piles d'entiers triées A et B en une pile triée C.

Exercice 2

Il s'agit de vérifier la validité d'une séquence de parenthèses sauvegardée dans une pile P.

a. Écrire la fonction de vérification qui retourne 0 si la séquence est correcte et -1 dans le cas contraire. Les éléments de la pile sont soit '(' soit ')'. Exemple : la séquence ()((()())) est correcte mais la séquence ())()(() est incorrect.
b. Récrire la fonction de vérification où les éléments de la pile sont soit des parenthèses '(' soit ')', soit des crochets '[', ']'. Exemple : [()()]()[(())] est correcte et [(] est incorrecte.

Exercice 3

1. Soient deux mots M1 et M2 de longueur <=30. Ecrire une fonction SUFFIXE qui vérifie si M2 est suffixe (partie à la fin) de M1.

Soit une pile S de mots (chaine de caractères de longueur <=30).

2. Donner la déclaration de la pile S (selon une représentation liste chainée)
3. Ecrire les primitives Empiler, Dépiler et SommetPile pour la pile S
4. On suppose que les mots de la pile sont ordonnés par ordre décroissant de leurs longueurs (le plus long se trouve au sommet). Ecrire une fonction SUPPR qui supprime de la pile tous les mots qui sont suffixes d'un autre mot de la pile. Quelle est la complexité de votre solution. Expliquez.

Exercice 4

a. Définir un type Date exprimée en jour, mois, année et écrire la fonction Comparer (E/D1, D2 : Date) : entier ; qui retourne 1, 0 ou -1 si D1< D2 ou D1=D2 ou D1>D2 respectivement.
b. Définir un type Pile permettant de représenter une pile de dates et écrire une fonction Vérifier (P : Pile) : booléen ; qui vérifie si la pile P est triée ou non par ordre croissant (la plus petite date au sommet de la pile).

Exercice 5

Ecrire une procédure de tri par insertion d'un ensemble de nombres entiers. Les données sont stockées dans une pile A et le programme doit retourner une pile B contenant ces nombres triés avec le minimum au sommet.

PS : L'algorithme proposé est le suivant : on utilise une pile C qui est vide au début. Tant que la pile A n'est pas vide, on considère les deux cas suivants : – si la pile B est vide ou si l'élément au sommet de A est plus petit que celui de B : on retire l'élément au sommet de la pile A pour l'empiler dans la pile B, puis si la pile C n'est pas vide on retire tous les éléments de la pile C pour les empiler dans la pile B. – sinon : on déplace l'élément au sommet de la pile B à la pile C.

TD N°2 : Les files

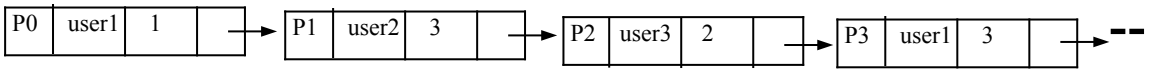
Exercice 6 : Ecrire les actions paramétrées suivantes :

a. *remplir(n) : file ;* qui remplit une file F avec n entiers données et la retourne,
b. *affiche_file(F) ;* qui affiche les éléments de la file F,
c. *recherche(F, val)* ; qui recherche une valeur val donnée dans une file F,
d. *supprim(F, k) ;* qui supprime de la file F l'élément qui se trouve à la position k donnée,
e. *insert(F, k, val) ;* qui insère un élément donné dans la file F à la position k donnée,
f. *fusion(A, B) ;* qui fusionne deux files d'entiers triées A et B en une file triée C.

Exercice 7

*On considère une liste chaînée contenant la description de processus (programmes) dans un système multi-utilisateurs. Chaque processus est décrit par une structure **st1** comportant un **id** (entier), un **username** (chaîne de caractères) et une **priorité** (entier).*

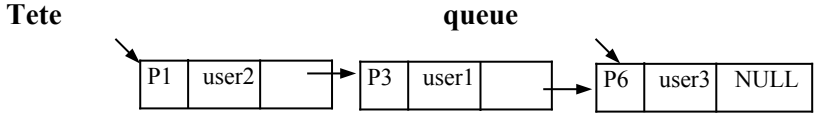
T



1. Donner la déclaration de la liste (on déclare d'abord la structure **st1**).
2. Etant donnée une priorité **pr** de processus, écrire une fonction **SUPPRIM** qui supprime la première occurrence d'un processus de priorité **pr**, d'une liste de point d'entrée **T**. La fonction doit renvoyer en sortie la structure **st1** du processus supprimé s'il existe, elle doit retourner 1 si la suppression est faite et 0 sinon.

*On désire manipuler une structure de **file** (c'est une liste chaînée de type FIFO avec deux pointeurs particuliers tête et queue) contenant des processus décrits chacun, par une structure **st2** composée de deux champs uniquement : **id** et **username**.*

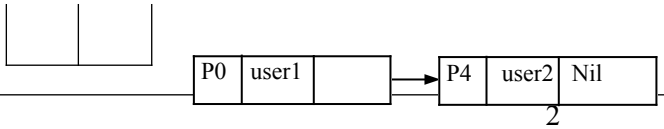
3. Donner la déclaration de la file (on déclare d'abord la structure **st2**).
4. Ecrire la primitive **ENFILER** qui permet de rajouter un élément de type **st2** en queue de file (celle-ci peut être initialement vide).
5. En utilisant les fonctions SUPPRIM et ENFILER, écrire une fonction **CONSTFILE** qui construit à partir de la liste initiale, une file contenant tous les processus d'une priorité **pr** donnée (Il s'agit de supprimer les éléments de la liste et les rajouter à la file).

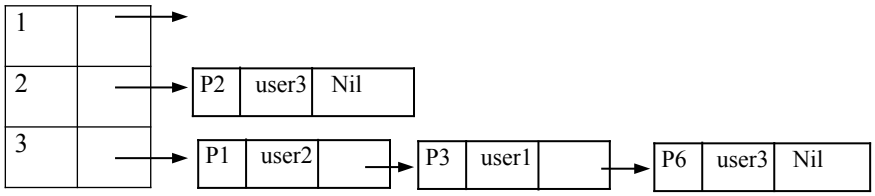


La figure montre tous les processus de priorité 3 extraits de la liste initiale.

On voudrait construire la structure ci-dessous, chaque élément de la pile contient une valeur de priorité et un lien vers la file contenant tous les processus de cette priorité.

- Donner la déclaration de la structure (définition de type)





Pile S

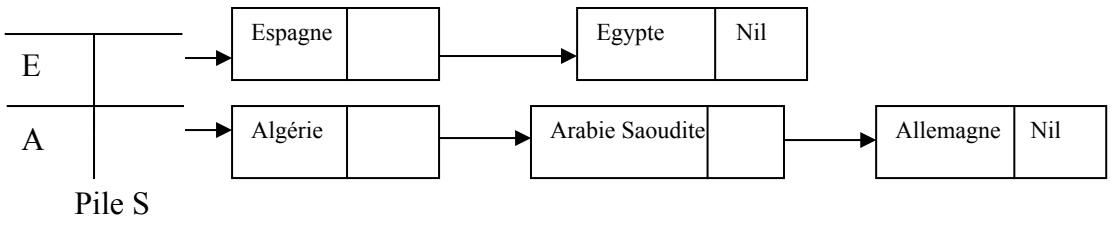
6. Ecrire le programme qui construit la structure ainsi décrite, à partir de la liste initiale supposée existante. On considère que les priorités sont des entiers compris entre 1 et n. La pile doit être triée par ordre croissant des priorités.

Exercices supplémentaires

Exercice 8

On s'intéresse à la manipulation de listes chaînées dont les éléments sont des chaînes de caractères de longueur <=30, correspondant à des noms de pays ("Algérie ", "Espagne", …).

1. Donner la déclaration de la liste
2. Etant donné un nom de pays, écrire une fonction **PREMIER** qui crée le premier élément de la liste (initialement la liste est vide).
3. Ecrire une fonction **AJOUT** qui ajoute un élément à la fin d'une liste de point d'entrée **tete** (Initialement la liste n'est pas vide).
4. Ecrire une fonction **SUPPRIME** qui supprime tous les éléments d'une liste de point d'entrée **tete**.
5. On voudrait ranger ces noms de pays par catégories, on mettra dans une même liste les noms commençant par la même lettre. Les différentes listes seront pointées par une pile S de la manière suivante :



a. Donner la déclaration de la structure (définition de type)

b. En utilisant les fonctions PREMIER et AJOUT, écrire une fonction **CONSTRUIT** qui construit la structure ci-dessus à partir de **n** noms de pays entrés au clavier, l'ordre alphabétique doit être respecté uniquement dans la pile.

c. Ecrire le programme qui construit la structure et supprime ensuite, tous les noms de pays commençant par une lettre alphabétique donnée (s'il en existe).

Exercice 9

Soit une liste circulaire (le dernier élément pointe vers le premier) de point d'entrée tête comportant des éléments décrivant les résultats d'un jeu d'équipe.
Chaque élément de la liste comporte un numéro de joueur (entier) et un score du joueur (entier), les éléments contenus dans la liste représentent les joueurs de la partie de jeu en cours.

- Donner la déclaration de la liste

1. Ecrire une action paramétrée ScoreMin qui détermine le joueur ayant le *score minimal,* l'action doit renvoyer l'adresse de l'élément et l'adresse de l'élément qui le précède. (1,5)

On aimerait remplacer certains joueurs par d'autres qui se trouvent dans une file d'attente F (structure FIFO avec deux points d'entrée tete et queue). Un élément de la file comporte uniquement le numéro du joueur.

- Donner la déclaration de la file selon une représentation dynamique (liste chainée)

2. Ecrire les primitives Enfiler et Défiler pour cette file
3. Ecrire une action paramétrée Remplace qui remplace un joueur de la liste (celui qui a le score minimal) par le joueur en file d'attente (il sera supprimé de la file). Un nouveau joueur rentre avec un score = 0. Le joueur qui sort de la liste sera mis en queue de file.

On aimerait enregistrer le nombre de fois (nombre de parties) où un joueur rentre dans le jeu (sort et rentre de nouveau).

4. Quels changements devra-t-on apporter aux structures de données afin de garder trace du nombre de fois où un joueur rentre en jeu? Donner les nouvelles déclarations de structures.

5. Réécrire la procédure Remplace (de la question 4) avec cette nouvelle structure sachant qu'on doit remplacer le joueur ayant joué le maximum de fois et ayant un score minimal.