Travaux Dirigés – TD 11

Exercice 61. Copie d'une pile/file

Écrire une fonction `void copiePile(Pile *s, Pile *copie)` qui permet de copier la pile `s` dans la pile `copie`. Attention ! l'ordre dans la pile doit être conservé. Faire de même pour la fonction `void copieFile(File *s, File *copie)`.

Exercice 62. File d'attente avec priorité

On désire implémenter une file d'attente avec priorité. La fonction `enfiler` doit recevoir l'élément à enfiler et sa priorité. La fonction `defiler` doit retirer l'élément le plus prioritaire.

— Modifier la structure de donnée `File` et `Element` pour prendre en compte les priorités
— Modifier les fonction `enfiler` et `defiler` pour prendre en compte les priorités

Exercice 63. Permutations circulaires

Ecrire une fonction `pile_circperm(Pile s, int n)` qui reçoit en argument une pile s et un entier n et effectue sur la pile n permutations circulaires successives.

Exemple avec n=2 :

7		98
11		2
98	donnera	103
2		7
103		11

Exercice 64. Expression correctement parenthésée

Dans un logiciel de calcul formel ou, plus généralement dans un éditeur de texte (par exemple utilisé pour écrire des programmes), il y a une gestion dynamique (i.e. « au vol ») du parenthésage : si une parenthèse fermante de trop est ajoutée ou si elle n'est pas de même nature (une accolade au lieu d'un crochet par exemple) que la parenthèse ouvrante associée alors un message d'erreur (visuel, sonore) est envoyé.

Par exemple, les deux expressions suivantes sont erronées :

$$Q = (\sqrt{a^2 + b^2}))^2$$

(deux parenthèses fermantes pour une ouvrante.)

$$Q = [\sqrt{a^2 + b^2})^2$$

(la parenthèse fermante n'est pas de même nature que la « parenthèse » ouvrante.)

Écrire une procédure `verifierParentheses(char* s)` qui reçoit une chaîne de caractères, en analyse la correction du parenthésage et renvoie à l'utilisateur un message adapté.

L'analyse de la chaîne de caractères se fera caractère après caractère. On gèrera une pile contenant les parenthèses ouvrantes et on comparera chaque parenthèse fermante au sommet (éventuel) de la pile.

Exercice 65. Première table de hachage

On dispose d'une table de hachage de taille 9 dont les collisions sont résolues par chaînage, et dont la fonction de hachage est h(k) = k % 9.

Insérer dans la table des valeurs suivantes : 5, 28, 19, 15, 20, 33, 12, 17, 10.

Exercice 66. Double hachage

Le double hachage est l'une des meilleures méthodes connues pour l'adressage ouvert. Il utilise une fonction de hachage de la forme

$$h : \begin{cases} \mathbb{N} \times \mathbb{N} & \to [0, \text{m-1}] \\ (\text{k, i}) & \to (h_1(k) + ih_2(k) \end{cases}$$

où h_1 et h_2 sont des fonctions de hachage auxiliaires "simples" et i un nombre entier entre 0 et 10.

1. Insérez les valeurs de l'exercice précédant dans une table de taille m = 13 en utilisant le double hachage $h(k, i) = h_1(k) + ih_2(k)$ avec $h_1(k) = k\%13$ et $h_2(k) = 1 + (k\%12)$.

2. Que se passe-t-il si $h_2(k)$ et m ne sont pas premiers entre eux ? En quoi est-ce gênant ?