LIFAPI - CM 8 Examen Blanc

Algorithmique : Longueur d'un chemin dans un tableau

En considérant un tableau de taille MAX (constante définie au préalable) rempli aléatoirement avec tous les entiers de 0 à MAX-1 (sans doublon), nous allons déterminer combien d'itérations sont nécessaires pour arriver sur la case contenant une valeur choisie aléatoirement. Par exemple si nous avons le tableau T de taille MAX = 8 rempli comme suit :

0	1	2	3	4	5	6	7
7	0	3	6	1	2	5	4

On tire aléatoirement une valeur entre 0 et MAX-1, par exemple 3. Comment connaître la longueur du chemin qui nous permettra d'arriver sur la case contenant la valeur 3 choisie en partant de T[3] ?
On commence par regarder le contenu de T[3] ➔ 6. La valeur nous donne l'indice de la case suivante. On regarde ensuite le contenu de T[6] ➔ 5 puis T[5] ➔ 2 puis T[2] ➔ 3. Il nous a donc fallu parcourir 4 cases pour trouver la case contenant la valeur choisie initialement.

1- Ecrire l'algorithme d'une **fonction booléenne** `EstDans` qui retourne vrai si un entier `n` est présent dans un tableau `T` de `MAX` entiers, faux sinon.
2- Ecrire l'algorithme d'un sous-programme `RemplitTab` qui remplit aléatoirement un tableau `T` de taille `MAX` avec tous les entiers de `0` à `MAX-1`. On suppose que le tableau a été initialisé avec des `-1` dans toutes les cases. On utilisera la fonction précédente pour s'assurer que les valeurs ne sont pas déjà présentes dans le tableau. En algorithmique, on dispose d'une fonction `aleatoire()` qui retourne comme en C++ un entier compris entre `0` et une constante `RANDMAX` très grande.
3- Ecrire l'algorithme d'une **fonction** `SuitChemin` qui à partir d'une valeur `n` initiale passée en paramètre calcule et retourne le nombre d'itérations nécessaires pour tomber sur la case du tableau `T` (également passé en paramètre) contenant cette valeur en suivant le chemin comme expliqué précédemment.
4- Ecrire le programme principal qui utilise les 3 sous-programmes précédents.

Langage C/C++ : Une opération dans une chaine de caractères !!!

Nous allons dans cet exercice créer un outil simple qui à partir d'une chaine de caractères contenant une opération d'addition entre deux entiers (par exemple "13+28") nous permettra de calculer et d'afficher le résultat de l'opération mathématique (41). Les chaines de caractères utilisées auront une taille maximale CHMAX fixée à 32 caractères. La constante CHMAX est déjà définie, vous pouvez l'utiliser.

1- Ecrire en langage C/C++ une fonction `ConvertitChaineEntier` qui convertit une chaine de caractères contenant au maximum CHMAX-1 caractères numériques en l'entier correspondant. La fonction devra retourner l'entier obtenu. Exemple si ch = "2351" la fonction renverra 2351. On pourra utiliser la fonction `puissance(x,y)` qui calcule et retourne x^y (équivalente au `pow(x,y)` du C++).
2- Ecrire en langage C/C++ un sous-programme `DecoupeChaine` qui à partir d'une chaine de caractères `chInit` passée en paramètre, extrait et construit **2 sous-chaines** de caractères `Nbre1` et `Nbre2`. `Nbre1` contient tous les caractères numériques entre le début de la chaine `chInit` et le symbole `'+'` et `Nbre2` contient tous les caractères numériques entre le `'+'` et la fin de la chaine. Exemple si `chInit` = `"43+120"` alors `Nbre1` ➔ `"43"` et `Nbre2` ➔ `"120"`.
3- Ecrire en langage C/C++ une **fonction** `Calcule` qui à partir d'une chaine de caractères `ch` passée en paramètre et des sous-programmes précédents, calcule et retourne le résultat du calcul contenu dans la chaine de caractères. On supposera que la chaine `ch` est correctement constituée.
4- Ecrire en langage C/C++ le programme principal qui utilise les 3 sous-programmes précédents.