Médian AG43 – Automne 2006 – durée 1h30 mardi 7 novembre 2006

Tableaux et nombres

Exercice 1 : Tirage du LOTO (5 points)

Ecrire une procédure nommée `LOTO7` qui procède au tirage aléatoire de 7 chiffres compris entre 1 et 49.
Cette procédure admettra comme unique paramètre le tableau tirage[] (indicé par rapport à 0) comprenant (après appel de la procédure) les 7 éléments (entiers) qui auront été tirés au sort.
Vous disposez de la fonction `rand()` n'admettant aucun argument et retournant un entier aléatoire pouvant être très grand, de la procédure `randomize()` qui initialise le générateur de nombres aléatoires ainsi que de l'opérateur % qui permet d'obtenir le reste de la division entière de 2 entiers.
Votre procédure pourra utiliser un tableau interne de 49 éléments (tableau que vous devrez générer de sorte à ce qu'il contienne initialement les nombres de 1 à 49).
Afin d'éviter que l'on puisse tirer 2 fois le même nombre, dès qu'un nombre est tiré au sort, il est supprimé du tableau (en décalant vers la gauche tous les éléments situés à sa droite).
En fait ce n'est pas le nombre en lui-même que vous tirez au sort mais plus exactement son indice dans le tableau.

Rappels concernant l'opérateur %:

68%49=19

et

x%49 sera toujours compris entre 0 et 48

Exemple de fonctionnement de la procédure :
Le tableau n[] (indicé par rapport à 0) de 49 éléments contient à un moment :

{1, 2, 4, 5, 6, 7, 8, 9,10,…49}.

Notez que le 3 n'y figure plus (ce qui signifie que son indice initial 2 a déjà été tiré et que 3 a été placé dans le tableau tirage).
Si un 7 est maintenant généré de manière aléatoire, le 9 sera à son tour placé dans le tableau tirage et le tableau n[] deviendra :

{1, 2, 4, 5, 6, 7, 8, 10,…49}

puisqu'à l'indice 7 du tableau on trouvait le 9.
Remarque : le 49 pourra apparaître plusieurs fois en fin de tableau après appel de la procédure.

Exercice 2 : Tri du tirage du loto (3 points)

Ecrire une procédure nommée `TRI_LOTO7` qui triera (par ordre croissant) le tableau de 7 entiers généré dans l'exercice précédant tout en laissant le dernier élément tiré au sort à sa place (fin du tableau). Le tableau à trier sera passé en paramètre.

Exercice 3 : FORTRAN (3 points)

Ecrire le code **FORTRAN d'une fonction** nommée `C_to_F` qui admet en paramètre la température en °C et qui retourne la température en Fahrenheit.
Pour rappel, 0°C correspond à 32°F, 100°C correspond à 212°F et la relation est linéaire.

Manipulation de chaînes de caractères

Exercice 4 : nombre d'occurrences des lettres de l'alphabet (4 points)

Ecrire une procédure nommée `alpha_occurrences` qui compte le nombre d'occurrences (nombre de fois où l'on rencontre) de chaque lettre de l'alphabet dans une chaîne de caractères qui sera passée en paramètre (de même que le tableau des occurrences).
On supposera que les chaînes sont délimitées par un caractère NUL.

Vous pourrez utiliser (ou non) une chaîne interne qui contiendra les 26 lettres de l'alphabet dans l'ordre.
Vous disposez de la procédure `strupr(chaine)` qui convertit une chaîne en majuscules et les caractères accentués en caractères non accentués (exemple : é, e, ê deviennent E).
Les tableaux et chaînes sont indicés par rapport à 0.

Exemple :
Si chaine="test" ;
Le tableau des occurrences contiendra [0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,2,0,0,0,0,0,0] après appel de la procédure.

Exercice 5 : Conjugaison des verbes du premier groupe au présent de l'indicatif (5 points)

La conjugaison au présent de l'indicatif de verbes du premier groupe est très simple. Il suffit d'ajouter au radical les terminaisons suivantes : **e, es, e, ons, ez, ent**.
On désire écrire une fonction appelée `Conjugue(verbe)` qui remplit le tableau de six chaînes de caractères `conjugaison[][]` (variable globale) par la conjugaison du verbe contenu dans la variable `verbe`. On dispose du tableau de chaînes de caractères `pronoms[][]` déjà initialisé, du tableau de terminaisons `present1[][]` également initialisé, ainsi que des fonctions et procédures suivantes :

- `strlen(chaine)` qui retourne la longueur d'une chaîne,
- `strncat(chaine1, chaine2, n)` qui ajoute à `chaine1` les n premiers caractères de `chaine2` si sa longueur dépasse n,
- `strcat(chaine1, chaine2)` qui copie `chaine2` au bout de `chaine1`
- `strcpy(chaine1, chaine2)` qui copie `chaine2` dans `chaine1`

Le tableau `pronoms[][]` est initialisé comme suit (inutile de le réinitialiser) :

```
pronoms[] = {"Je ", "Tu ", "Elle/Il ", "Nous ", "Vous ", "Elles/Ils "}
```

Chaque pronom est suivi du caractère espace, ce qui évite de le rajouter individuellement. Le tableau `present1[][]` est initialisé comme suit :

```
present1[] = {"e", "es", "e", "ons", "ez", "ent"}
```

Dans ce problème, on utilise pour désigner les tableaux de chaînes de caractères, le double indiçage référencé par rapport à 0. Ainsi, la quatrième chaîne de caractères du tableau `present1` (qui est "ons") sera désignée par `present1[3]`, `present1[3][1]` représente le caractère 'n', et `present1[3][3]` représente un caractère NUL.
L'argument verbe est une chaîne de caractères contenant l'infinitif du verbe à conjuguer. Exemple :
`verbe = "parler"`

On appelle la fonction de la manière suivante :
`Conjugue(verbe)`

Au retour, le tableau `conjugaison[][]` contient :
`conjugaison[0] = "Je parle"`
…
`conjugaison[3] = "Nous parlons"`
…

Ecrire la procédure `Conjugue()`, qui admet comme argument une chaîne de caractères (verbe), qui remplit le tableau global `conjugaison[][]`.