Les exercices 1 à 3 seront rendus sur une copie différente des exercices 4 et 5.

Calculs horaires

Exercice 1 : Durée entre 2 horaires (3 points)

Soient t1, t2 deux horaires (t2 postérieur à t1) donnés sous forme de tableaux de 3 entiers (h, m, s). Ecrire la procédure `duree(t1,t2,t3)` permettant de calculer le temps écoulé entre t1 et t2. Le résultat sera placé dans le tableau t3.
t2 sera toujours considéré comme postérieur à t1. Si tel n'est pas le cas, alors t2 sera considéré comme étant un horaire du jour suivant. De ce fait le résultat sera toujours compris entre 0 et 24h.

Tableaux et chaînes de caractères

Exercice 2 : Terminaison d'un mot (3 points)

Ecrire la procédure `terminaison(mot,n,termin)` qui copie les n derniers caractères de la chaîne `mot[]` dans la chaîne `termin[]`.
Exemple :
Si mot = "administration", après appel de :

`terminaison(mot,4,termin)`

`termin` vaudra "tion".
Les chaînes se terminent par un caractère nul. L'indiçage est effectué par rapport à zéro. Vous n'utiliserez pas la fonction `strlen`. Si le nombre de caractères de la chaîne est inférieur ou égal à n alors la terminaison sera égale au mot.

Exercice 3 : Pluriel d'un mot

En français, la règle générale pour la formation du pluriel d'un nom est de rajouter un *-s* à la fin du mot.
Les exceptions concernent les mots finissant par : *-s, -x, -z, -al, -ail, -ou, -eau, -au, -eu.*
Les mots se terminant par *-s, -x* ou *-z* sont invariables.
Les mots de terminant par *-al* forment leur pluriel en *-aux* sauf exceptions.
Les mots se terminant par *-ail* prennent un *-s* au pluriel (règle normale) sauf exceptions.
Les mots se terminant par *-ou* prennent un *-s* au pluriel sauf exceptions.
Les mots se terminant par *–eau, -au, -eu* prennent un *-x* sauf exceptions.
Tous les tableaux seront indicés par rapport à 0.

3-1) Recherche d'un mot dans une liste (3 points)

Ecrire la fonction `est_exception(mot)` qui retourne l'indice du mot passé en argument dans un tableau d'exceptions de 200 mots maxi.
Le tableau `exceptions[200][15]` est une variable globale.
`exception[13]` représente la 14ème exception dans la liste.
`exception[13][2]` représente le 3ème caractère de cette 14ème exception.
S'il ne s'agit pas d'une exception (mot ne figurant pas dans la liste des exceptions), alors la fonction retournera -1.
Vous disposez de la fonction `strcmp(string1,string2)` qui retourne 0 si la chaîne string1 est identique à la chaîne string2.

3-2) procédure pluriel (5 points)

Ecrire la procédure `pluriel(mot)` qui retourne le pluriel du mot passé en argument.
S'il s'agit d'une exception, le pluriel sera lu dans le tableau `pluriels[200][30]` de structure identique à celle du tableau des exceptions et contenant (dans le même ordre) le pluriel de chaque exception.
Si le mot se termine par *-s, -x* ou *-z*, le mot est invariable.
Si la terminaison vaut *-al* alors il faudra remplacer *-al* par *-aux* pour construire le pluriel.
Si la terminaison vaut *-eau, -au* ou *-eu*, le mot prendra un *-x*.
Dans le cas général, il suffira d'ajouter un *-s* au mot d'origine pour construire le pluriel.
Le programme principal aura préalablement appelé la procédure `strupr(mot)` qui aura permis de transformer `mot` en majuscules.
Vous disposez de la procédure `strcpy(dest,orig)` qui copie la chaîne `orig` dans la chaîne `dest` et de la procédure `terminaison(mot,n,termin)` préalablement écrite.

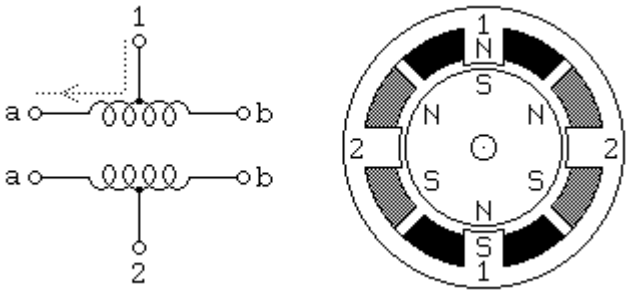
Automatisme

Exercice 4 : Générateur de nombres aléatoires (3 points)

On désire réaliser un générateur de nombres aléatoires. Pour cela, on utilise une technique classique consistant à boucler un registre à décalage sur lui même au travers d'un OU EXCLUSIF. Pour simplifier, on utilise un tableau d'octets appelé TAB[] (entiers codés sur 8 bits), dont la valeur pourra être -1 ou +1.
Le tableau comporte 16 éléments dont la valeur sera +/-1. Il est indicé par rapport à 0. L'opération OU EXCLUSIF se fera entre le 4ème et le dernier élément. S'ils sont identiques, le résultat sera -1, et s'ils sont différents, le résultat sera 1. On peut utiliser soit des tests ou des opérations arithmétiques (multiplication) pour obtenir le résultat.
Après l'opération OU EXCLUSIF, on procède au décalage à droite de tous les éléments du tableau. Le dernier élément (TAB[15]) sera perdu, et prendra la valeur de l'avant dernier, etc. Le premier élément prendra la valeur du résultat du OU EXCLUSIF.
Ecrire une procédure appelée `Alea()` qui effectue cette opération sur le tableau TAB[] passé en argument.

Exercice 5 : Commande d'un moteur pas à pas uni polaire (3 points)

On désire commander des moteurs pas à pas unipolaires à l'aide d'un microcontrôleur. Le schéma du moteur est le suivant :



On alimente successivement les points a et b de chaque bobine, les points 1 et 2 étant reliés à un potentiel commun (masse). L'ordre d'alimentation des points a et b des enroulements va dépendre du mode de fonctionnement choisi. Pour simplifier, on adopte la séquence suivante (la valeur 1 signifie que la bobine est alimentée) :

Bobine 1		**Bobine 2**	
a	**b**	**a**	**b**
1	0	0	0
0	0	1	0
0	1	0	0
0	0	0	1

Dans ce schéma, on alimente successivement les 4 bobines dans un ordre bien précis. Si l'on applique sur les 4 points d'alimentation a et b la séquence indiquée dans le tableau, on avance de 4 pas.
Chaque point a et b des 2 bobines est connecté à un bit d'un port d'entrée-sortie du microcontrôleur, par l'intermédiaire d'un amplificateur. Il faut donc 4 bits pour piloter le moteur pas à pas. Sur les 8 bits disponibles sur un port d'entrée sortie nous n'utiliserons que les quatre bits successifs de poids faible. Dans la configuration mode sortie (un port peut fonctionner en entrée ou en sortie), l'état des bits du port est la copie conforme d'un octet donné de la mémoire du microcontrôleur. Nous appellerons cet octet R (nous le considèrerons comme étant la variable R, octet).
Pour faire avancer le moteur de 6 pas dans le sens horaire, nous écrirons successivement dans la mémoire R les valeurs binaires suivantes :
1000 ; 0010 ; 0100 ; 0001 ; 1000 ; 0010
Entre chaque écriture, il faut marquer un temps d'attente, qui permet au rotor d'atteindre sa nouvelle position et de se stabiliser. On dispose pour cela d'une fonction `delai()` qui admet en argument un entier, indiquant la durée d'attente en millisecondes. Elle permet, par la même occasion, de jouer sur la vitesse et assure une durée minimale d'alimentation d'une bobine. Après l'avance complète, on n'oubliera pas de couper l'alimentation du moteur (il suffit de mettre R à zéro).
On remarque que la séquence décrite dans le tableau va se répéter en fonction du nombre de pas nécessaires. Si l'on veut tourner dans le sens inverse, il suffit de présenter la séquence dans l'ordre inverse.
Les 4 états de base sont placés dans un tableau nommé `etat[]`, variable globale déjà initialisée, comportant 4 octets indicés par rapport à 0 . Une variable globale entière notée `position` va compter (ou décompter) le nombre de pas. On l'utilisera pour sélectionner l'état à présenter, après avoir utilisé l'opération modulo pour rétablir l'indiçage correct, compris entre 0 et 3. La variable position permet donc de mémoriser la position angulaire du moteur entre deux appels de la fonction de rotation du moteur. Elle ne devra donc pas être réinitialisée.

Ecrire la fonction `Avance(n,t)` qui fait effectuer au moteur une rotation de n pas, avec un délai de t millisecondes entre deux pas. Attention, il ne faudra pas s'étonner de la simplicité de cette fonction.