

Travaux Dirigés de LO01

TD 1 - Algorithmme

1. Ecrire l'algorithme qui calcule le minimum de cinq nombres.
2. Déterminer un algorithme qui calcule l'heure d'arrivée d'un marathonien à partir de l'heure de départ et de la durée du trajet. Chaque donnée d'entrée sera fournie sous forme de trois nombres précisant les heures, minutes et secondes.
3. Déterminer un algorithme qui calcule la résistance équivalente à 3 résistances montées en parallèle ou en série. Les données fournies en entrée seront le type de montage et la valeur des trois résistances (positives ou nulles).
4. Déterminer un algorithme qui calcule le produit de deux nombres positifs en utilisant uniquement des additions.
5. Ecrire un algorithme qui lit un nombre entier et détermine si celui-ci est un nombre premier ou non. S'il n'est pas premier, on affichera son plus petit diviseur (différent de 1).

Rappel : Un nombre est premier s'il n'est divisible que par 1 et par lui-même.

Exemples :

17 est un nombre premier car il n'existe pas de nombre n appartenant à $[2, 16]$ tel que $17 = n \times \text{quotient}$.

18 n'est pas un nombre premier car $18 = 2 \times 9$ (et $18 = 3 \times 6$). Son plus petit diviseur est 2.

6. Ecrire un algorithme qui lit un nombre octal et le transforme en décimal.

TD 2 - Syntaxe et Expressions

1. Utilisation des diagrammes de Conway

Quels sont les littéraux corrects dans la série:

| | | | | |
|--------|--------|---------|----------|---------|
| 432 | 3,475 | 7.4 | .1475 | '\458' |
| 6E3 | "123" | 0.0001 | '\020' | 0671 |
| 1.0e-4 | 0xffff | 0.1E999 | 432. | 0XA67F |
| '!' | '1' | -6.0 | -12.3e-4 | 11.e4.5 |

Pour chaque cas favorable, préciser le type de cette constante.

2. Vérifier la validité de :

2.1 `int **tab ;` (déclaration variable)

2.2 `for (; ;)` (instructions)

2.3 `for(i=0,*j=0; i<MAX; i++) ;` (instructions)

2.4 `struct {
 char *nom ;
 int age ;
} *18 ;` (déclaration variable)

3 . Calculer les expressions C suivantes :

| | | |
|--------------------------------|---|---|
| <code>4 * 18 - 198 % 10</code> | <code>7 + 3 <= 10</code> | <code>(8+4) > 3 && 3 > 2</code> |
| <code>(36 / 18 * 4) / 3</code> | <code>((3 > 7) (3 <= 7)) && (7 != 10)</code> | <code>8.1 == 9.8</code> |
| <code>10 / 4</code> | <code>10.0 / 4.0</code> | <code>'c' < 'f'</code> |

4 . Que vaut x après la suite d'instructions :

```
y = 3 ;
x = 2 ;
y++ ;
x += 2 ;
x *= 2 + y;
x -= 2 ;
```

TD 3 - Programmes C

1. Ecrire un programme qui affiche le minimum de cinq nombres entrés au clavier.
2. Ecrire un programme qui calcule la surface et le volume d'une sphère à partir du rayon.
3. Ecrire un programme qui calcule l'heure d'arrivée d'un marathonien à partir de l'heure de départ et de la durée du trajet. Chaque donnée d'entrée sera fournie sous forme de trois nombres précisant les heures, minutes et secondes.
4. Ecrire un programme qui calcule la résistance équivalente à 3 résistances montées en parallèle ou en série. Les données fournies en entrée seront le type de montage et la valeur des trois résistances.

TD 4 - Boucles

1. Ecrire un programme qui lit 10 nombres et affiche le nombre de valeurs négatives
2. Ecrire un programme qui lit 50 nombres et affiche la valeur du plus petit.
3. Ecrire un programme qui lit une suite de caractères et s'arrête lorsqu'il rencontre un point. Le programme affichera le nombre de 'a' ou 'A'.
4. Ecrire un programme qui calcule le produit d'une série de nombres lus au clavier. On supposera que l'on entrera la valeur 0 pour indiquer la fin de la série, cette valeur n'étant pas prise en compte.
5. Ecrire un programme qui lit un nombre entier et détermine si celui-ci est un nombre premier ou non. S'il n'est pas premier, on affichera son plus petit diviseur (différent de 1).
6. Vous devez écrire le programme permettant de réaliser le jeu suivant :

Le programme définit un nombre compris entre 1 et 100 (utiliser la fonction rand()) sans le dévoiler au joueur et le joueur doit deviner ce nombre en un minimum d'essais.

Exemple :

- l'ordinateur "choisit" le nombre 32
- le joueur entre un nombre : 67
- l'ordinateur répond : trop grand
- Voulez-vous continuer : O
- le joueur entre un nombre : 15
- l'ordinateur répond : trop petit
-

A chaque nouvel essai le programme demandera si le joueur veut continuer ou arrêter. La partie s'arrêtera lorsque le joueur a trouvé le nombre ou lorsque le joueur désire arrêter de jouer.

Le message suivant s'affichera :

Bravo vous avez trouvé en x coups (x étant le nombre d'essais)

ou

Vous avez perdu

TD 5 - Problèmes

1. Moyenne olympique

Ecrire un programme qui calcule la moyenne olympique d'une suite de nombres. La moyenne olympique est obtenue en faisant la moyenne des nombres auxquels on a retiré le minimum et le maximum.

2. Cartes de crédit

L'algorithme de Luhn fut développé dans les années 1960 comme une méthode de validation de nombres (entiers). C'est une simple formule de vérification de somme (Checksum) utilisée pour valider une variété de numéros de comptes, comme les numéros de cartes de crédit. Le principe est le suivant :

Tout d'abord, on modifie le nombre de la manière suivante : on multiplie par deux un chiffre sur deux, en commençant par l'avant dernier et en se déplaçant de droite à gauche. Si un chiffre qui est multiplié par deux donne un nombre plus grand que 9 (comme c'est le cas par exemple pour 8 qui devient 16), alors il faut le ramener à un chiffre entre 1 et 9 en additionnant les chiffres qui le composent (pour le chiffre 8 : on obtient d'abord 16 en le multipliant par 2, puis 7 en sommant les chiffres composant le résultat : 1+6). La somme de tous ces chiffres est ensuite effectuée. Enfin, le résultat est divisé par 10. Si le reste de la division est égal à zéro, alors le nombre original est valide.

Exemple

Considérons l'identification du nombre 972 487 086. Les cases colorées dans le tableau suivant indiquent les chiffres doublés. La somme, égale à 50, est divisée par 10 : le reste est 0, donc le nombre est valide.

| | | | | | | | | | |
|--|------------|----|---|---|---|----|---|----|---|
| Nombre initial | 9 | 7 | 2 | 4 | 8 | 7 | 0 | 8 | 6 |
| Doublement d'un chiffre sur 2 | 9 | 14 | 2 | 8 | 8 | 14 | 0 | 16 | 6 |
| Si le double comprend 2 chiffres, on ajoute ces 2 chiffres | 9 | 5 | 2 | 8 | 8 | 5 | 0 | 7 | 6 |
| Somme des chiffres | Somme = 50 | | | | | | | | |

Ecrire un programme qui demande à l'utilisateur de rentrer le numéro de sa carte de crédit (sous forme d'entier) et qui vérifie si celui-ci est valide selon l'algorithme de Luhn. D'autre part le programme devra indiquer si la carte correspond à une carte Visa, une MasterCard ou un autre type. Le numéro d'une carte Visa commence toujours par le chiffre 4 et comporte entre 13 et 16 chiffres. Le numéro d'une MasterCard commence par 54 ou 55 et comporte 16 chiffres.

3. Suite de Fibonacci

Ecrire un programme qui affiche tous les termes d'une suite de Fibonacci inférieurs à 1000.

Définitions :

$$U_{n+1} = U_n + U_{n-1} \quad U_0 \text{ et } U_1 \quad \text{entiers positifs entrés par l'utilisateur.}$$

Affichage :

$$U_0 = 5$$

$$U_1 = 3$$

$$U_2 = 8$$

$$U_3 = 11 \dots$$

TD 7 - Fonctions (1)

1. Factorielle

Ecrire une fonction *fact(x)* qui retourne la factorielle de x .

2. Puissance

Ecrire une fonction *puissance()* qui calcule la valeur de x^n pour x réel et n entier relatif.

3. Couicable

- 3.1 Ecrire une fonction *nbDeChiffre(x)* qui retourne le nombre de chiffres composant l'entier x .
- 3.2 Une fonction *nbChestPaire(x)* qui retourne 1 si le nombre de chiffres composant x est pair, 0 sinon.
- 3.3 Une fonction *extraitNombre(x,n,lg)* qui extrait de x , à partir du $n^{\text{ième}}$ chiffre en partant de la droite, le nombre composé des lg chiffres.
Exemple *extraitNombre(45863047,5,3)* -> 586.
- 3.4 Une fonction *sommeDesChiffres(x)* qui retourne la somme des chiffres composant le nombre x .
- 3.5 Un nombre x est dit *couicable* si la somme des chiffres de sa partie droite égale la somme des chiffres de sa partie gauche. Exemple 256823 est *couicable* (le nombre de chiffres le composant doit être pair). Ecrire le programme correspondant.

TD 8 - Fonctions (2)

1. Scope lexical

Indiquer la valeur des variables i, j, k aux différents *printf* du programme.

```
#include <stdio.h>
void plus(int i,int j,int k)
{
    k = i + j ;
    printf(" point 2 : i = %d, j = %d, k = %d \n ",i,j,k) ;
}
void fact(int j, int *k)
{
    int i ;
    for (i=1, *k = 1 ; i<=j ; i++)
        *k = *k * i ;
    printf(" point 4 : i = %d, j = %d, k = %d \n ",i,j,k) ;
}
void mult( int *i,int *j,int *k)
{
    int l ;
    for ( l=1 , *i = 0 ; l<=*j ; l++)
        *i = *i + *k ;
    printf(" point 6 : i = %d, j = %d, k = %d \n ",*i,*j,*k) ;
    { int i = 0 ;
        for( l = 1 ; l <= *k ; l++)
            i = i + *k ;
        printf(" point 7 : i = %d, j = %d, k = %d \n ",i,*j,*k);
    }
    printf(" point 8 : i = %d, j = %d, k = %d \n ",*i,*j,*k) ;
}

void main()
{
    int i,j,k;
    i = 2;
    j = 5;
    k = 10;

    printf(" point 1 : i = %d, j = %d, k = %d \n ",i,j,k) ;
    plus(i,j,k) ;
    printf(" point 3 : i = %d, j = %d, k = %d \n ",i,j,k) ;
    fact(j,&k) ;
    printf(" point 5 : i = %d, j = %d, k = %d \n ",i,j,k) ;
    mult(&i,&j,&k) ;
    printf(" point 9 : i = %d, j = %d, k = %d \n ",i,j,k) ;
}
```

2. Permutations

Ecrire une fonction *ordre(a,b,c)* qui range par ordre croissant les valeurs de ses trois paramètres. Après l'appel *ordre(a,b,c)*, les valeurs de ces variables doivent vérifier :

$$a \leq b \leq c$$

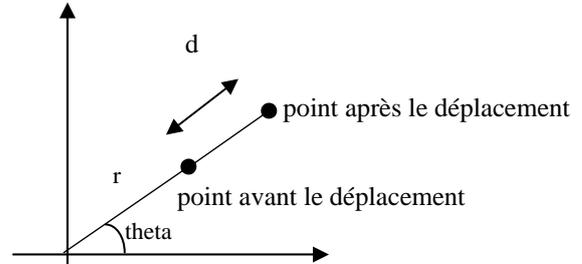
3. Transformations de coordonnées.

Soient les fonctions suivantes, supposées connues:

- void `cartesienEnPolaire(float x, float y, float *r, float *theta)`
qui calcule les coordonnées polaires (r, θ) d'un point décrit par ses coordonnées cartésiennes (x, y)

- void `polaireEnCartésien(float r, float theta, float *x, float *y)`
qui calcule les coordonnées cartésiennes (x, y) d'un point décrit par ses coordonnées polaires (r, θ)

Ecrire la fonction `deplace(x, y, d)` qui éloigne (ou rapproche) le point de coordonnées cartésiennes (x, y) du centre du repère d'une distance d . Vous utiliserez les fonctions précédemment définies, le déplacement se calculant de façon très simple en coordonnées polaires.



TD 9 - Tableaux (1)

1. Saisie et parcours dans un tableau.

Ecrire un programme qui lit 100 nombres, en fait la moyenne et affiche les nombres supérieurs à cette moyenne.

2. Recherche dans un tableau.

2.1 Ecrire un programme qui remplit un tableau de 100 valeurs avec des entiers compris entre 0 et 1000 à l'aide de la fonction *rand()*.

2.1 Ajouter à ce programme la recherche de la première occurrence d'une valeur x donnée par l'utilisateur. Le programme affichera la position de x ou -1 si ce nombre n'est pas dans le tableau.

4. Recherche du minimum.

Ecrire une fonction qui retourne la valeur minimum d'un tableau de réels. Le tableau peut être partiellement rempli. L'occupation réelle de ce tableau (nombre d'éléments) sera transmise en paramètre.

5. Dichotomie.

On voudrait connaître la position d'un élément dans un tableau d'entiers déjà trié. Pour cela on écrira une fonction *dicho(x,tab,nb)* x est l'élément que l'on recherche, *tab* le tableau et *nb* le nombre d'éléments de *tab*. La fonction retournera la position de l'élément dans le tableau (-1 si inexistant). On utilisera une méthode dichotomique.

6. Crible d'Eratosthène.

On désigne sous le nom de crible d'Eratosthène une méthode de recherche des nombres premiers plus petits qu'un entier naturel n donné en éliminant tous les multiples successifs. Ecrire la fonction *crible(n)* qui utilisera la fonction *malloc()*.

TD 10 - Tableaux (2)

1. Algorithme de tri

On dispose de n valeurs numériques stockées dans un tableau. Il s'agit de ranger par ordre croissant ces n valeurs dans le même tableau. Par exemple, si on dispose au départ des 6 valeurs 7, 3, 1, 2, 5, 4, elles devront être rangés à la fin dans l'ordre 1, 2, 3, 4, 5, 7. Voici un algorithme simple, appelé **tri par sélection**, réalisant ce tri : on cherche le plus petit nombre et on le permute avec le nombre placé en première position, on cherche le plus petit des nombres restants et on le permute avec le nombre placé en seconde position. Avec notre exemple cela donnerait :

| | | | | | |
|---|---|---|---|---|---|
| 7 | 3 | 1 | 2 | 5 | 4 |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| 1 | 3 | 7 | 2 | 5 | 4 |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 7 | 3 | 5 | 4 |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 7 | 5 | 4 |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 7 |
|---|---|---|---|---|---|

Le programme demandé aura la structure suivante : saisie d'un tableau d'entiers (*initTab*), tri de ce tableau (*triTab*) et affichage des nombres ainsi triés (*afficheTab*).

2. Tableaux de caractères

On veut construire l'adresse électronique d'un étudiant de l'UTC à partir de son prénom et de son nom. Par exemple l'étudiant Jean Dupont aura l'adresse Jean.Dupont@etu.utc.fr. Le nom et le prénom sont stockés dans 2 chaînes de 20 caractères, l'adresse électronique sera stockée dans une chaîne de 40 caractères.

Les règles de construction d'adresse sont les suivantes :

- Une adresse est composée du prénom suivi d'un point (.) suivi du nom et enfin de l'extension @etu.utc.fr
- On n'accorde pas d'importance aux majuscules et minuscules.
- Dans le cas d'un prénom et d'un nom composés d'un grand nombre de caractères, c'est le prénom qui est tronqué. Par exemple l'étudiante Marie-Sebastienne Dumoulinabord aura l'adresse Marie-Sebastie.Dumoulinabord@etu.utc.fr
- Le caractère d'espace n'étant pas autorisé dans les adresses électroniques, il sera automatiquement omis. Par exemple l'étudiant Pierre De La Tourdenface aura l'adresse Pierre.DeLaTourdenface@etu.utc.fr

2.1 Ecrire la fonction *construitAdresseMail(prenom, nom, adresse)* qui construit l'adresse électronique d'un étudiant à partir de *prenom* et *nom*.

2.2 Ecrire le programme qui demande à l'utilisateur son nom et son prénom, construit l'adresse électronique et l'affiche

3. La bataille navale

Ecrire un programme qui simule une bataille navale. La grille sera stockée dans un tableau à deux dimensions (5 sur 5). L'ordinateur placera 5 bateaux à l'intérieur de la grille, pour cela on utilisera la fonction *rand()* pour calculer les indices *i* et *j* correspondant à la ligne et la colonne du bateau à placer. Si un bateau se trouve déjà sur l'emplacement l'ordinateur redonnera une autre position. L'utilisateur devra « deviner » l'emplacement des bateaux. Le programme s'arrêtera lorsque tous les bateaux auront été trouvés, on affichera alors le nombre de coups qu'il aura été nécessaire pour trouver les bateaux.

TD 11 - Structures

1. Demain est un autre jour...

Définir le type *uneDate* basé sur une structure permettant de représenter une date comportant le numéro du jour, le numéro du mois et l'année.

Ex : 14 7 1792 pour le 14 juillet 1792
25 12 2007 pour le 25 décembre 2007
29 2 2008 pour le 29 février 2008

1.1 Ecrire un programme qui détermine la date du lendemain d'un jour donné.

1.2 Transformez le programme en la fonction *jourDuLendemain()*.

Remarque : On supposera connue la fonction *int estBissextile(int a)* qui retourne 1 si a est un entier correspondant à une année bissextile, 0 sinon.

2. Gestion d'un garage

2.1 Définir le type *uneVoiture* basé sur une structure composée des champs suivants : modèle, nombre de chevaux et longueur.

2.2 Ecrire les fonctions *initVoiture()* et *compareVoiture()*. On utilisera la fonction *strcmp()* qui compare deux chaînes et renvoie 0 si les chaînes sont égales.

2.3 On désire stocker l'ensemble des voitures composant un garage dans un tableau (maximum 100 voitures). Ecrire le programme permettant de remplir le tableau, demande à un client la voiture qu'il recherche et affiche le nombre de voitures correspondant à la demande du client.

3. Station Météo

Dans une station météo, on relève chaque jour la température minimum et la température maximum. A la fin du mois, on saisit les données dans une structure de type *unReleveMensuel*.

Le relevé mensuel contiendra les informations suivantes :

- nom de la station (chaînes de caractères de 20 caractères maximum)
- coordonnées GPS (latitude, longitude, altitude)
- mois
- tableau des températures min en degré (taille max 31)
- tableau des températures max en degré (taille max 31)

D'autre part, on supposera connue la fonction *int nbDeJours(int mois)* qui retourne le nombre de jours que contient le mois passé en paramètre.

3.1 Ecrire le type *uneCoordGPS* comprenant trois réels, en degré pour la latitude et la longitude et en mètre pour l'altitude. Ecrire le type *unReleveMensuel* comprenant les champs suivants : le nom de la station, les coordonnées GPS (*uneCoordGPS*), le mois concerné (sous forme d'entier compris entre 1 et 12), le tableau des températures journalières minimum en degrés et le tableau des températures journalières maximum en degrés.

3.2 Ecrire la fonction *initUnReleveMensuel()* qui initialise un relevé à partir d'un dialogue avec l'opérateur, le relevé sera passé en argument de sortie de la fonction et le mois en argument d'entrée.

3.3 Ecrire une fonction *tempMoyMensuelle()* qui retourne la température moyenne d'un relevé mensuel.

3.4 Ecrire une fonction *memeRegion()* qui retourne 1 si les deux coordonnées GPS transmises en paramètres sont proches et 0 sinon. On suppose que 2 positions GPS sont proches si l'écart entre les latitudes et l'écart entre les longitudes n'excèdent pas 5 (degrés).

3.5 Ecrire une fonction *etudeRechauffement()* qui calcule deux valeurs :

- *nbRtot* : le nombre de relevé contenus dans le tableau ayant le même mois que le relevé courant et appartenant à la même région.
- *nbRinf* : Le nombre de relevés qui, pour un même mois et dans une même région, présentent un signe de réchauffement : moyenne mensuelle inférieure et altitude de la station plus faible par rapport au relevé courant.

On transmet à cette fonction le relevé courant, le tableau de relevés et le nombre de relevé dans le tableau.

TD 12 – Fichiers

1. Initiation aux fichiers.

- 1.1 Ecrire un programme qui lit une série de nombre au clavier et la stocke dans un fichier structuré dont le nom est donné par l'utilisateur.
- 1.2 Ecrire un programme qui affiche la moyenne des nombres contenus dans le fichier précédent.

2. Mondial de football 2014.

Dans un fichier structuré centralisé au comité de la fédération française de football sont référencés tous les joueurs licenciés susceptibles d'être sélectionnés pour le mondial 2014. Ce fichier *joueurs.dat* contient pour chacun les informations suivantes :

- nom
- prénom
- âge
- club

Le sélectionneur Didier Deschamps a créé chez lui un petit fichier texte *selectionneur.txt* dans lequel figure uniquement le nom des joueurs sélectionnés.

Ribery
Benzema
Lloris
Matuidi
Sakho
...

On voudrait à partir de ces deux fichiers, créer le fichier structuré *selectionnes.dat* contenant les informations du fichier *joueurs.dat* mais uniquement pour les joueurs sélectionnés.

- 2.1 Définir le type *unJoueur*
- 2.2 Ecrire la fonction *creerFichierSelectionnes()* qui crée le fichier *selectionnes.dat*. Les paramètres de la fonction seront trois chaînes de caractères qui contiendront les noms des fichiers.
- 2.3 Ecrire la fonction *ageMoyen()* qui renvoie l'âge moyen des joueurs contenus dans un fichier dont le nom est passé en paramètre.
- 2.4 Ecrire le programme principal qui crée le fichier *selectionnes.dat* et affiche l'âge moyen des joueurs sélectionnés.

On suppose que les noms écrits dans le fichier texte sont écrits de la même manière que dans le fichier de joueurs.

TD 13 - Récursivité

1. Ecrire une fonction récursive du PGCD de deux entiers.
2. On suppose que les seules fonctions arithmétiques disponibles sont *Plus_1()* et *Moins_1()* qui retourne respectivement la valeur de leur paramètre augmenté ou diminué de 1. Ecrire une fonction qui effectue la somme de deux nombres positifs ou nuls en utilisant ces fonctions.
3. Ecrire la fonction exposant *Exp(m,n)*
$$\text{Exp}(m,n) = 1 \quad \text{si} \quad n = 0$$
$$\text{Exp}(m,n) = (m^2)^{n/2} \quad \text{si} \quad n \text{ est pair}$$
$$\text{Exp}(m,n) = m \cdot (m^2)^{(n-1)/2} \quad \text{si} \quad n \text{ est impair}$$
$$\text{Exp}(m,n) = 1/m^{-n} \text{ si} \quad n \text{ est négatif}$$
4. Ecrire une fonction récursive de recherche dichotomique d'un élément à l'intérieur d'un tableau trié.
5. Déterminer récursivement *sin (x)* en utilisant les formules :
$$\sin 2x = 2 \sin x \cdot \cos x$$
$$\cos 2x = \cos^2 x - \sin^2 x$$
et $\sin x = x$ et $\cos x = 1$ pour $x < 0.00001$
6. Ecrire la fonction récursive *palindrome* qui affiche une chaîne de caractères suivie de son miroir.

Exemple : *palindrome(veronique,...)* -> veroniqueeuqinorev