

EXERCICES DE PROGRAMMATION C
les instructions de contrôle, les types

Exercice 1:

Écrire un programme qui affiche sur écran la table des codes ASCII des caractères compris entre 23 et 127.

Exercice 2:

Écrire le programme de résolution de l'équation $AX^2 + BX + C = 0$ dans R en étudiant tous les cas possibles.

Exercice 3 :

Le plus grand commun diviseur (PGCD) de deux entiers A et B est le plus grand entier qui divise à la fois A et B

Écrire un programme qui calcule le PGCD de deux entiers :

2.1. en utilisant la méthode $PGCD(A,B) = PGCD(B,A-B)$ si $A > B$
et $PGCD(A,B) = PGCD(A,B-A)$ si $A < B$.

2.2. en utilisant le célèbre algorithme d'EUCLIDE :

$PGCD(A,B) = PGCD(B, A \bmod B)$ si $A \geq B$

$PGCD(A,B) = PGCD(A, B \bmod A)$ si $A < B$

$PGCD(A,0) = A$

où $A \bmod B$ désigne le reste de la division entière entre A et B .

Exercice 4:

Écrire un programme réalisant une calculatrice pour les opérations $+$, $-$, $*$ et $/$:

- demander à l'opérateur de saisir le premier opérande (un flottant)
- demander à l'opérateur de saisir l'opérateur (un caractère)
- demander à l'opérateur de saisir le deuxième opérande (un flottant)
- selon l'opérateur choisi effectuer le calcul (utiliser switch ... case)
- afficher le résultat

On fera un programme simple qui ne fait qu'une seule opération et on testera le cas de la division par zéro pour laquelle on affichera un message d'erreur.

Reprendre l'exercice précédent en effectuant une boucle dans laquelle on demande à l'opérateur s'il veut faire une autre opération.

Exercice 5:

Écrire un programme qui permet de calculer l'exponentielle de x en utilisant la série :

$$\text{exponentielle de } x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

Ajouter une boucle dans laquelle on demande à l'opérateur s'il veut faire une autre opération.

Exercice 6:

Écrire un algorithme qui permet de trouver tous les nombres parfaits inférieurs à une certaine limite

Un nombre entier est parfait s'il est égal à la somme de ses diviseurs (sauf lui-même).

Ex : $6 = 1 + 2 + 3$ est parfait.

EXERCICES DE PROGRAMMATION C
Les fonctions

Exercice 1 :

Ecrire la fonction `NombreChiffre` du type `int` qui obtient une valeur entière N (positive ou négative) du type `long` comme paramètre et qui fournit le nombre de chiffres de N comme résultat.

Ecrire un petit programme qui teste la fonction `NombreChiffre`:

Exemple d'exécution:

Introduire un nombre: 123

Le nombre 123 a 3 chiffres.

Introduire un nombre: 580499

Le nombre 580499 a 6 chiffres.

Exercice 2:

Un nombre entier est parfait s'il est égal à la somme de ses diviseurs (sauf lui-même).

Ex : $6 = 1 + 2 + 3$ est parfait.

Ecrire une fonction `somme_div` qui retourne la somme des diviseurs d'un nombre passé en paramètre.

Ecrire une fonction `parfait` qui teste si un nombre passé en paramètre est parfait et qui retourne 1 s'il l'est et 0 sinon.

Ecrire un programme principal qui affiche tous les nombres parfaits inférieurs à une certaine limite.

Exercice 3:

Deux nombres M et N sont appelés `nombres_amis` si la somme des diviseurs de M est égale à N et la somme des diviseurs de N est égale à M

Ecrire une fonction `amis` qui retourne le nombre `amis` (s'il existe) d'un nombre passé en paramètre, cette fonction utilise la fonction `somme_div` de l'exercice 2.

Ecrire un programme principal qui affiche tous les `nombres_amis` inférieurs à une certaine limite.

Exercice 4 :

Ecrire un programme qui construit et affiche le triangle de Pascal en calculant les coefficients binomiaux:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
...
```

On n'utilisera pas de tableau, c.-à-d. il faudra calculer les coefficients d'après la formule:

$$C_p^q = \frac{p!}{q!(p-q)!}$$

Exercice 5:

On désire calculer l'exponentielle de x en utilisant la série :

$$\text{exponentielle de } x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

Ecrire une fonction `serie_expon` qui développe la série jusqu'à un indice n ; elle reçoit en paramètre les valeurs de x et de n .

Ecrire une fonction `expon` qui retourne la valeur de l'exponentielle d'un nombre x passé en paramètre en développant jusqu'à l'indice 20.

Ecrire un programme principal qui saisit un nombre flottant et affiche son exponentielle.

Exercice 6:

Il s'agit de reprendre l'exercice de la calculatrice en le découpant en fonctions.

Ecrire les fonctions suivantes :

- *saisir_operande : elle demande de saisir un opérande (un flottant), elle effectue la saisie et elle retourne la valeur saisie*
- *saisir_operateur : elle demande de saisir un operateur (un caractère), elle effectue la saisie et elle retourne l'opérateur saisi*
- *afficher_resultat : elle effectue le calcul et elle affiche le résultat; elle a en paramètres les deux opérandes et l'opérateur; elle affiche un message d'erreur dans le cas de la division par zéro ou si l'opérateur est inconnu*
- *continuer : elle demande si on veut faire une nouvelle opération, elle saisit la réponse et elle retourne 1 si oui et 0 si non.*

Ecrire un programme principal qui réalise la calculatrice en utilisant ces fonctions.

EXERCICES DE PROGRAMMATION C
Les pointeurs

Exercice 1: Addition de deux nombres complexes

Ecrire une fonction **somme** qui permet de faire l'addition de deux nombres complexes
Ecrire le programme principal qui
- saisie les parties réelles et les parties imaginaires de deux nombres complexes,
- calcule la somme de deux nombres complexes (fait appel à la fonction **somme**),
- affiche le résultat de la somme

Modification du programme:

Ecrire une fonction qui permet de saisir la partie réelle et la partie imaginaire d'un nombre complexe.
Ecrire une fonction qui affiche un nombre complexe.
Ecrire le programme principal qui fait appel à ces fonctions.

Exercice 2: Résolution de l'équation $AX^2 + BX + C = 0$ dans l'ensemble R

Ecrire une fonction *saisie* qui permet de saisir A, B et C.

Ecrire une fonction *resoudre* qui permet de **résoudre l'équation $AX^2 + BX + C = 0$ ET retourner**
0 si l'équation n'a pas de solution
1 si l'équation n'a qu'une solution
2 si l'équation possède deux solutions
3 si R est l'ensemble de solutions

Ecrire une fonction *affiche* qui permet d'afficher
pas de solution réelle si la valeur retournée par la fonction *resoudre* est 0
la valeur **X₁** si la valeur retournée par la fonction *resoudre* est 1
les valeurs **X₁** et **X₂** si la valeur retournée par la fonction *resoudre* est 2
l'ensemble de solution est R si la valeur retournée par la fonction *resoudre* est 3
Ecrire le programme principal qui fait appel à ces fonctions

Exercice 3: jeux de cailloux

Le jeu des cailloux : un tas de N cailloux se trouve entre deux joueurs ; à tour de rôle chacun prend 1, 2 ou 3 cailloux. Celui qui est obligé de prendre le dernier caillou a perdu.

Ce jeu possède une stratégie gagnante : le joueur qui réussit, à laisser un nombre de cailloux égal à un multiple de 4 plus 1, à chaque fois qu'il doit jouer, gagne à coup sûr.

Le but est d'écrire un programme qui simule le jeu des cailloux (entre vous et la machine) :

Ecrire une fonction *initialiser* qui demande le nombre de cailloux et quel joueur commence le premier.

Ecrire une fonction *utilisateur_joue* qui demande à l'utilisateur le nombre de cailloux à prendre et qui met à jour le nombre de cailloux restants.

Ecrire une fonction *machine_joue* qui permet à la machine de prendre un nombre de cailloux et qui met à jour le nombre de cailloux restants.

Ecrire le programme principal.

Les prototypes :

```
void initialiser(int *nbCa, int *jo) ;  
void utilisateur_joue(int *nbCa) ;  
void machine_joue(int *nbCa) ;
```

nbCa : le nombre de cailloux

jo : joueur (1 : l'utilisateur et 2 : la machine)

EXERCICES DE PROGRAMMATION C
Les tableaux

Exercice 1: Moyenne, minimum et maximum

Ecrire une fonction **saisir** qui permet saisir un tableau de réels

Ecrire une fonction **afficher** qui permet d'afficher les éléments du tableau

Ecrire une fonction **calculer_moyenne** qui permet de calculer la moyenne des éléments du tableau

Ecrire une fonction **trouver_minmax** qui permet de trouver le minimum et le maximum des éléments du tableau.

Ecrire le programme principal

Exercice 2: Inversion d'un tableaux

Ecrire une fonction **SaisieTableau** qui un tableau Tab d'entier de dimension N.

Ecrire une fonction **AfficheTableau** qui affiche le tableau Tab.

Ecrire le programme principal.

Ajouter au programme une fonction **InverseTableau** qui inverse le tableau Tab sans utiliser de tableau d'aide (la fonction **InverseTableau** doit échanger les éléments du tableau à l'aide de deux indices qui parcourent le tableau en commençant respectivement au début et à la fin du tableau et qui se rencontrent en son milieu).

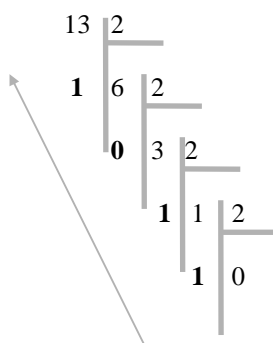
Exercice 3: Conversion en binaire

On veut convertir un nombre décimal en binaire selon la méthode suivante:

1. Tant que le nombre décimal est différent de zéro, on le divise par deux et on met le reste de la division dans un tableau.

2. On inverse les élément du tableau.

Exemple: le nombre 13 vaut **1101** en binaire



1. Ecrire une fonction **inverser** qui permet d'inverser les éléments d'un tableau.

2. Ecrire une fonction **convertir** qui permet de convertir un nombre décimal en binaire.

3. Ecrire une fonction **affiche** qui affiche les éléments d'un tableau.

4. Ecrire le programme principal.

les prototypes des fonctions:

void inverser(int *tab, int nb);

void convertir (int dec, int *bin, int *nb);

void affiche (int *tab, int nb);

Exercice 4: Matrices

Ecrire une fonction **saisir** qui permet de saisir une matrice carrée

Ecrire une fonction **afficher** qui permet d'afficher les éléments d'une matrice

Ecrire une fonction **additionner** qui permet d'additionner deux matrices

Ecrire une fonction **multiplier** qui permet de multiplier deux matrices

Ecrire une fonction **menu** qui demande à l'utilisateur quelle est l'opération à effectuer

Ecrire le programme principal

Exercice 5: Recherche des Points-Clos MaxLignes

Un Points-Clos MaxLignes est un élément d'une matrice qui est à la fois:

un maximum sur la ligne

et

un minimum sur la colonne

Exemples:

$$\begin{vmatrix} \mathbf{3} & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{vmatrix}$$

$$\begin{vmatrix} 1 & 7 & 9 & 8 \\ 4 & \mathbf{6} & 0 & \mathbf{6} \end{vmatrix}$$

$$\begin{vmatrix} 0 & \mathbf{7} \\ 8 & 9 \\ 6 & \mathbf{7} \\ 8 & 8 \end{vmatrix}$$

Ecrire une fonction *SaisirMatrice* qui permet de saisir

- le nombre de lignes

- le nombre de colonnes

- les éléments d'une matrice MAT

Ecrire une fonction *AfficherMatrice* qui affiche les éléments d'une matrice.

Ecrire une fonction *MaxLigne* qui crée une nouvelle matrice Maxlig à partir de la matrice saisie MAT tel que:

$$\text{MaxLig}[i,j] = \begin{cases} 1 & \text{si Mat}[i,j] \text{ est un maximum sur la ligne } i \\ 0 & \text{sinon} \end{cases}$$

Ecrire une fonction *MinColonne* qui crée une nouvelle matrice MinCol à partir de la matrice saisie MAT tel que:

$$\text{MinCol}[i,j] = \begin{cases} 1 & \text{si Mat}[i,j] \text{ est un minimum sur la colonne } j \\ 0 & \text{sinon} \end{cases}$$

Ecrire une fonction *TrouvePointsClos* permet trouver et afficher tous les Points-Clos MaxLigne d'une matrice Mat (c'est à dire: cette fonction affiche les valeurs des éléments ainsi que leurs numéros de lignes et de colonnes)

EXERCICES DE PROGRAMMATION C
Les chaînes de caractères

Exercice 1: Longueur d'une chaîne

Ecrire une fonction **longueur_chaine** qui calcule la longueur d'une chaîne de caractères.
Ecrire le programme principal.

Exercice 2. Concaténation de deux chaînes

Ecrire la fonction **concatener** dont le résultat est la concaténation de deux chaînes de caractères. Cette fonction accepte trois paramètres : les deux chaînes à concatener, et la chaîne résultante.
Ecrire un programme permettant de tester cette fonction.

Exercice 3 : palindrome

Un palindrome est une chaîne de caractères qui est égale à son image miroir.
Exemples ABE, ELLE et LAVAL sont des palindromes.
Ecrire une fonction **palindrome** qui vérifie qu'une chaîne de caractères est un palindrome. Elle accepte un tableau de caractères et retourne la valeur 1 si son paramètre contient un palindrome et 0 dans le cas contraire.
Ecrire un programme permettant de tester cette fonction.

Exercice 4 :

On désire écrire un programme qui permet de convertir une chaîne de caractères minuscules en majuscules et calcule le nombre de voyelles, consonnes, espaces et calcule le total des caractères.
Ecrire une fonction **convertir_caractere** qui permet de convertir minuscule en majuscule
Ecrire une fonction **convertir_chaine** qui permet de convertir minuscule en majuscule
Ecrire une fonction **compte_chaine** qui compte le nombre de voyelles, de consonne et retourne le total de tous les caractères.
Ecrire une fonction qui affiche la chaîne en majuscule, le nombre de voyelles, de consonne et le total de tous les caractères.

Les prototypes des fonctions utilisées :

```
char convertir_caractere(char car);  
void convertir_chaine(char *chaine);  
int compte_chaine(char *chaine,int *pv,int *pc, int *pes);  
void afficher(char *chaine,int v,int c, int e, int t);
```

(pv : nombre de voyelles pc : nombre de consonnes pe : nombre d'espaces)

Exercice 5: Conjugaison

On désire écrire un programme qui conjugue un verbe régulier du premier groupe.
Ecrire une fonction qui saisie un verbe et vérifie qu'il est du premier groupe et retourne sa longueur.
Ecrire une fonction qui conjugue le verbe.
Ecrire le programme principal.

Exemple d'exécution :

```
je chante  
tu chantes  
il chante  
nous chantons  
vous chantez  
ils chantent
```

EXERCICES DE PROGRAMMATION C
Les structures

Exercice 1.

Pour représenter un nombre complexe définir un type de structure qui contient deux flottants, la partie réelle et la partie imaginaire.

1. Ecrire une fonction permettant de saisir un complexe passé en paramètre.
2. Ecrire une fonction permettant d'afficher un complexe passé en paramètre.
3. Ecrire une fonction d'addition qui reçoit en paramètre deux complexes, qui fait la somme des deux et range le résultat dans la première structure.
4. Ecrire un programme principal qui :
 - saisit deux complexes c1 et c2
 - ajoute c2 à c1
 - affiche c1 après ajout

Exercice 2.

On veut gérer un répertoire téléphonique (contenant les noms et numéros de téléphones).

1. Définir un type de structure PERS qui contient deux champs: le nom d'une personne et son numéro de téléphone.
2. Ecrire une fonction **saisir_personne** qui permet de saisir une personne (une structure de type PERS).
3. Ecrire une fonction **saisir_repertoire** qui permet de saisir un tableau de personnes.
4. Ecrire une fonction **afficher_repertoire** qui permet d'afficher le contenu du répertoire.
5. Ecrire une fonction **chercher_personne** qui permet de chercher un numéro d'une personne donné dans le répertoire.
6. Ecrire un programme principal qui saisie un répertoire téléphonique et propose le dialogue suivant:
 - 6.1 afficher le contenu du répertoire
 - 6.2 chercher le numéro de téléphone d'une personne
 - 6.3 quitter le Programme

EXERCICES DE PROGRAMMATION C
Les fichiers

Exercice 1: répertoire téléphonique.

On veut réaliser un répertoire téléphonique. Chaque personne est représentée dans le répertoire par un nom, prénom et le numéro de téléphone.

Pour représenter une personne, définir un type de structure **personne** qui contient deux chaînes de caractères et un entier.

1. Ecrire une fonction **void ajout_personne (FILE *rep)** qui saisit une personne et l'ajoute à la fin du fichier passé en paramètre.
2. Ecrire une fonction **void affiche_ensemble_personne (FILE *rep)** qui affiche le contenu du fichier.
3. Ecrire une fonction **void trouve_numéro_personne (FILE *rep, char *nom)** qui permet de trouver le numéro de téléphone d'une personne donnée en paramètre.
4. Ecrire une fonction **void changer_numéro_personne (FILE *rep, char *nom)** qui permet de changer le numéro de téléphone d'une personne donnée en paramètre.
5. Ecrire une fonction **void menu(FILE *rep)** qui demande l'opération à exécuter: ajouter une personne, afficher le contenu du répertoire, trouver un numéro ou quitter le programme.
6. Ecrire le programme principal.

Exercice 2: Gestion de comptes bancaire.

1. Définir un type structure DATE qui contient trois membres d'entiers : jour, mois et année
2. Définir un type structure CLIENT qui contient les membres suivants :
 - numero_cmpt : entier (numéro de compte)
 - nom : chaîne de caractères (nom d'un client)
 - der_operation : caractère (R : Retrait, V : Virement)
 - anc_solde : réel (ancien solde)
 - nouv_solde : réel (nouveau solde)
 - date : DATE (jj mm aa)
3. Ecrire une fonction *ouvrir* qui ouvre un fichier existant ou le crée sinon
4. Ecrire une fonction *fermer* qui ferme le fichier
5. Ecrire le programme principal qui fait appel SEULEMENT aux fonctions *ouvrir*, *fermer* et la fonction *menu*. Le début de la fonction menu est donné.
Avant de continuer VERIFIEZ que le fichier est bien créé
6. Ecrire une fonction *ajout* qui ajoute un client
7. Ecrire une fonction *affiche* qui affiche le compte d'un client. Cette fonction doit être, capable de chercher un client soit par son nom soit par son numéro de compte
TESTEZ votre programme avant de continuer
8. Ecrire une fonction *lister* qui affiche tous les comptes des clients
TESTEZ votre programme avant de continuer
9. Ecrire une fonction *operation* qui réalise les retraits, les virements et les mises à jour des comptes.

```
void menu(FILE *fic)
{
    char choix;
    do
    {
        printf("\n\nAjouter d'un nouveau client.....: A\n");
        printf("Consultation d'un compte client.....: C\n");
        printf("Lister tous les comptes de clients.....: L\n");
        printf("Opération sur un compte client.....: O\n");
        printf("Quitter.....: Q\n");
        printf("          votre choix: ");
        rewind(stdin);
        scanf("%c",&choix);
        switch(choix)
        {
            case 'a':
            case 'A': ajout( ... ..
                .....
                .....
                .....
                .....
                .....
                .....
            }
        }while (choix != 'q' && choix != 'Q');
    }
```