UVHC
UNIVERSITE DE VALENCIENNES
ET DU HAINAUT-CAMBRESIS

Licence 1ère année
2018-2019

TP PROGRAMMATION LANGAGE C

Intervenants : N. CHOUCHANI, A. EL IDRISSI, M. A. LABIOD, R. TODOSIJEVIC
Responsable du cours : S. PIECHOWIAK

1. SEANCE 1 : PRISE EN MAIN DE L'ENVIRONNEMENT CODE::BLOCKS

L'objectif de ce TP est de prendre en main l'IDE **CODE::BLOCKS**. Pour cela, on va reprendre les exercices vus en Cours et en TD :

- Calcul du kième nombre premier
- Calcul du premier nombre premier supérieur à k
- Calcul de la somme des k premiers nombres premiers
- Calcul de la valeur approchée de $\cos(x) = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k}}{(2j)!}$ à ε près
- Déterminer si 2 nombres entiers A et B sont amis (la somme des diviseurs de A est égale à la somme des diviseurs de B).

2. SEANCE 2 : LES TABLEAUX A UNE DIMENSION : LES ALGORITHMES DE TRI

On se propose d'étudier les performances de différents algorithmes de tri de nombres entiers. La structure de données choisie pour mémoriser les nombre est le simple tableau de dimension 1.

1. Définir le type `TTableauDeNombres` comme étant un tableau de taille maximale 1000.
2. Donner l'algorithme de tri par sélection, puis sa traduction en langage C
3. Donner l'algorithme de tri par insertion, puis sa traduction en langage C
4. Donner l'algorithme de tri à bulles, puis sa traduction en langage C
5. Donner un algorithme de génération aléatoire de NB nombres entiers placés dans un tableau.
6. On souhaite comparer différentes caractéristiques de ces algorithmes :
   - le nombre de déplacement des valeurs
   - le nombre de comparaisons des valeurs
7. Donner un programme qui remplit aléatoirement un tableau de taille 1000 qui tri ce tableau avec chacune des méthodes de tri et affiche le nombre de déplacements et le nombre de comparaisons.

8. Afin de fiabiliser les résultats obtenus, on souhaite répéter 50 fois ce travail et afficher non-pas le nombre de déplacements et le nombre de comparaisons mais la moyennes des valeurs obtenues au cours des 50 essais. Quelles sont vos conclusions ?

3. SEANCE 3: TABLEAUX A UNE DIMENSION : CALCUL DE POLYNOMES

On se propose de faire quelques opérations sur des polynômes réels de degré maximal 50. Pour rappel, un polynôme est une somme de monômes et chaque monôme possède un coefficient réel et un degré (nous supposons que les degrés sont des entiers positifs ou nuls). Une représentation simple d'un tel polynôme consiste à utiliser un tableau dont les indices sont les degrés des monômes et les valeurs leurs coefficients.

Par exemple, le polynôme $\boldsymbol{P(X) = 5X^7 + 3X^5 - 9X^3 + 8X^2 + 17}$ sera représenté par le tableau :

17	0	8	-9	0	3	0	5		
0									
	1	2	3	4	5	6	7		

A chaque polynôme est associé le tableau qui le représente et son degré (qui permet de connaitre l'indice du monôme de plus haut degré).

1. Définir le type **`TPolynome`** comme étant un tableau de taille maximale 50 (on définira cette taille maximale comme une constante).
2. Définir un sous-programme de saisie d'un polynôme.
3. Définir un sous-programme d'affichage d'un polynôme.
4. Définir un sous-programme qui permet de calculer la valeur d'un polynôme P(X) pour une valeur de X donnée.
5. Définir un sous-programme qui calcule la somme formelle de 2 polynômes donnés (de type **`TPolynome`**).
6. Définir un sous-programme qui calcule la dérivée formelle d'un polynôme donné (de type **`TPolynome`**).
7. Définir un sous-programme qui calcule le produit de 2 polynômes donnés (de type **`TPolynome`**).
8. Ecrire un programme principal qui affiche un menu permettant des tester les sous-programmes précédents. L'organisation des menus sera de la forme suivante :

```
< 1 > Saisir un polynôme
      < 1 > polynôme A
      < 2 > polynôme B
      < 0 > Quitter
< 2 > Calculer la somme S = A + B
< 3 > Calculer le produit P = A × B
< 4 > Calculer la dérivée D = P'
< 5 > Afficher un polynôme
      < 1 > Afficher A
      < 2 > Afficher B
      < 3 > Afficher S
      < 4 > Afficher P
```

```
      < 5 > Afficher D
      < 0 > Quitter
< 0 > Quitter le programme
```

Le menu principal s'affichera d'abord. Puis, en fonction du choix de l'utilisateur, les sous-menus s'afficheront. Par exemple, l'utilisateur voit le menu principal :

< 1 > Saisir un polynôme
< 2 > Calculer la somme S = A + B
< 3 > Calculer le produit P = A × B
< 4 > Calculer la dérivée D = P'
< 5 > Afficher un polynôme
< 0 > Quitter le programme

et sélectionne la commande 5. A ce moment, le sous-menu suivant apparait :

< 1 > Afficher A
< 2 > Afficher B
< 3 > Afficher S
< 5 > Afficher P
< 5 > Afficher D
< 0 > Quitter

4. SEANCE 4: TABLEAUX A DEUX DIMENSIONS : CALCUL MATRICIEL

On se propose de faire quelques opérations sur des matrices réelles de taille maximale 50 × 50.

1. Définir le type matrice comme un tableau de taille 50 × 50.
2. Ecrire un sous programme qui permet à l'utilisateur d'entrer la taille d'une matrice (nbLignes × nbColonnes) puis de saisir le contenu de cette matrice.
3. Ecrire un sous programme qui permet d'afficher le contenu d'une matrice donnée M (nbLignes × nbColonnes) . L'affichage se fera ligne par ligne si nbColonnes < 20 et valeur par valeur si nbColonnes >= 20. Dans ce dernier cas, chaque ligne affichée sera de la forme : M[i,j] = valeur.
4. Ecrire un sous-programme qui met à zéro la diagonale d'une matrice carrée (nbLignes = nbColonnes).
5. Ecrire un sous-programme qui calcule la transposée d'une matrice réelle (nbLignes × nbColonnes).
6. Ecrire un sous-programme qui multiplie les valeurs d'une matrice réelle (nbLignes × nbColonnes) par un nombre réel donné (ce nombre est saisi auparavant).
7. Ecrire un sous-programme qui additionne 2 matrices réelles de même taille (nbLignes × nbColonnes).
8. Ecrire un sous-programme qui multiplie 2 matrices réelles A (nbLignesA × nbColonnesA) et B (nbLignesB × nbColonnesB) où nbColonnesA = nbLignesB.

Ecrire un programme principal qui affiche un menu permettant des tester les sous-programmes précédents. L'organisation des menus sera de la forme suivante :

```
< 1 > Saisir une matrice
      < 1 > matrice A(taille et contenu)
```

```
      < 2 > matrice B(taille et contenu)
      < 3 > mettre la matrice A à zéro
      < 4 > mettre la matrice B à zéro
      < 5 > mettre la matrice C à zéro
      < 0 > Quitter les saisies
< 2 > Mettre à zéro la diagonale d'une matrice
      < 1 > Mettre à zéro la diagonale de A
      < 2 > Mettre à zéro la diagonale de B
      < 3 > Mettre à zéro la diagonale de C
      < 0 > Quitter la mise à zéro des diagonales
< 3 > Calculer la transposée d'une matrice
      < 1 > Calculer la transposée de A
      < 2 > Calculer la transposée de B
      < 3 > Calculer la transposée de C
      < 0 > Quitter le calcul des transposées
< 4 > Calculer C = A + B
< 5 > Calculer C = A × B
< 6 > Afficher une matrice
      < 1 > Afficher A
      < 2 > Afficher B
      < 3 > Afficher C
      < 0 > Quitter l'affichage
< 0 > Quitter le programme
```

Le menu principal s'affichera d'abord. Puis en fonction du choix de l'utilisateur, les sous-menus s'afficheront. Par exemple, l'utilisateur voit le menu principal :

```
< 1 > Saisir une matrice
< 2 > Mettre à zéro la diagonale d'une matrice
< 3 > Calculer la transposée d'une matrice
< 4 > Calculer C = A + B
< 5 > Calculer C = A × B
< 6 > Afficher une matrice
< 0 > Quitter le programme
```

et sélectionne la commande **6**. A ce moment, le sous-menu suivant apparait :

```
    < 1 > Afficher A
    < 2 > Afficher B
    < 3 > Afficher C
    < 0 > Quitter l'affichage
```

5. SEANCES 5 ET 6 : LES TABLEAUX DE DIMENSION 2 : LE JEU DE LA VIE.

On souhaite simuler une colonie d'individus qui évoluent au cours du temps. Cette colonie est représentée par un tableau de dimension 2 dont les cases sont occupées ou libres.

A chaque itération, on calcule quelle sera la génération suivante à partir de la génération courante. Pour cela, on doit respecter des règles simples d'évolution :

A. lorsqu'une case est vide et qu'elle est entourée exactement de 3 cases occupées, elle donne naissance à un individu (la case devient occupée dans la génération suivante).
B. lorsqu'une case est occupée et qu'elle est entourée exactement de 2 ou 3 cases occupées, elle reste occupée par un individu (la case reste occupée dans la génération suivante).
C. dans tous les autres cas, la cases reste vide ou elle le devient (l'individu meurt) dans la génération suivante.

1. Définir le type TGeneration comme un tableau de dimension maximale 50 × 50
2. Donner un sous-programme qui permet à un utilisateur de saisir sa première génération.
3. Donner un sous-programme qui affiche une génération.
4. Donner un sous-programme qui calcule la génération qui suit la génération courante (selon les 3 règles données ci-dessus).
5. Donner un programme qui saisit une première génération et qui affiche les NG générations suivantes (NG est saisi par l'utilisateur).
6. On souhaite arrêter le programme dès que les générations n'évoluent plus (c'est à dire que les générations N et N+1 sont strictement égales). Donner un programme qui s'arrête lorsque les générations n'évoluent plus.

6. SEANCES 7 ET 8 : LES STRUCTURES ET LES FICHIERS EN C

On se propose de gérer une discothèque. Chaque disque est caractérisé par :

- un numéro (qui sert d'identifiant au disque)
- un support (CD, K7, USB)
- un titre d'album (une chaine);
- les différents morceaux (un tableau d'au plus 20 morceaux)

La discothèque complète est un ensemble de disques.

Chaque morceau est caractérisé par :

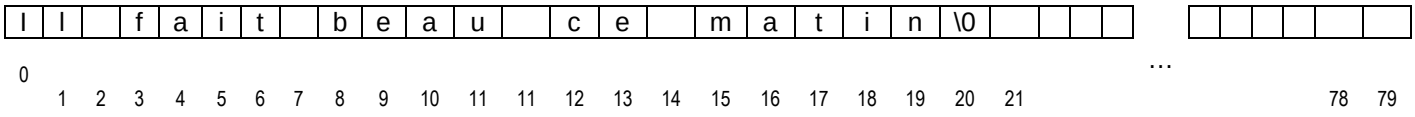
- son titre
- son auteur
- son genre
- sa durée (temps en minutes)

1. Définir la structure de donnée TMorceau qui rassemble les informations d'un morceau.
2. Définir un sous-programme de saisie d'un nouveau morceau.
3. Définir un sous-programme qui affiche toutes les informations d'un sur un morceau connu par son rang dans le disque (par exemple affichage du 3ème morceau d'un disque).
4. Définir la structure de donnée TDisque qui rassemble les informations d'un disque.
5. Définir un sous-programme de saisie d'un nouveau disque.
6. Définir un sous-programme qui affiche toutes les informations (y-compris les morceaux) d'un disque connu par son numéro.
7. Définir le type TDiscotheque qui est une structure à seulement 2 champs : nb (le nombre de disques présents) et disques (le tableau rassemblant les nb disques).
8. Définir un sous-programme qui calcule le temps total d'un disque connu par son numéro, c'est à dire la somme des durées des morceaux qui le compose.
9. Définir un sous-programme qui calcule le temps total d'une discothèque.
10. Définir un sous-programme qui affiche la liste de tous les morceaux d'un auteur connu par son nom.

11. Définir un sous-programme qui enregistre une discothèque dans un fichier.
12. Définir un sous-programme qui lit une discothèque dans un fichier.

7. SEANCE 9: LES CHAINES DE CARACTERES

Les chaînes de caractères sont des tableaux de caractères dont le dernier caractère « utile » est '\0'. Par exemple la chaîne « Il fait beau ce matin » peut être mémorisée dans un tableau T de 80 caractères :



1. Ecrire un programme qui compte le nombre de caractères d'une phrase CH.
2. Ecrire un programme qui lit une chaîne de caractères CH et qui convertit toutes les majuscules dans des minuscules et vice-versa. Le résultat sera mémorisé dans la même variable CH et affiché après la conversion.
3. Ecrire un programme qui supprime la première occurrence d'une chaîne de caractères CH1 dans une chaîne de caractères CH2.

```
Exemples:  PHON  ALPHONSE     ALSE
           EI    PIERRE       PIERRE
           T     TOTALEMENT   OTALEMENT
```

4. Ecrire un programme qui calcule le nombre de mots dans une phrase CH.
5. Ecrire un programme qui remplace la première occurrence d'une chaîne de caractères CH1 par la chaîne CH2 dans une chaîne de caractères CH. Utiliser une chaîne de sauvegarde FIN pendant le remplacement.

```
Exemple :          IE  ARTE PIERRE      PARTERRE
                   EI  IE   PIERRE      PIERRE
                   TOT FIN  TOTALEMENT  FINALEMENT
```

6. Ecrire un programme qui remplace toutes les occurrences d'une chaîne de caractères CH1 par la chaîne CH2 dans une chaîne de caractères CH. Utiliser une chaîne de sauvegarde FIN pendant le remplacement.