

ALGORITHMIQUE

ET

PROGRAMMATION EN LANGAGE C

~

Exercices – Mini TP

2022-2023

ING1 - ING3 nouveaux - Prepac

Elisabeth Rendler
Enseignant

SOMMAIRE

1. DECOMPOSITION DE PROBLEME ET ALGORITHME

[Exercice 101 Instructions](#)
[Exercice 102 DTI, ACD](#)
[Exercice 104 Boucle : Instructions](#)
[Exercice 105 Boucle : DTI, algorithme](#)
[Exercice 106 Boucle : DTI, ACD](#)
[Exercice 107 Boucle : DTI, algorithme](#)
[Exercice 108 Boucle et test : Instructions](#)
[Exercice 109 Boucle et test : DTI, algorithme](#)
[Exercice 110 Boucle et test : DTI et organigramme](#)
[Exercice 111 Cas d'étude : ACD et organigramme](#)
[Exercice 112 Cas d'étude : ACD et organigramme](#)
[Exercice 113 Cas d'étude :](#)
[Exercice 114 Cas d'étude : boucle](#)
[Exercice 115 Cas d'étude : boucle](#)
[Exercice 116 Résultat d'algorithme](#)
[Exercice 117 Résultat d'algorithme](#)

2. ORDINATEUR ET ENVIRONNEMENT DE TRAVAIL

[Exercice 201 Ordinateur](#)
[Exercice 202 Arborescence des fichiers](#)
[Exercice 203 Environnement de travail : CodeBlocks](#)

3. STOCKAGE DES DONNEES : VARIABLES DE TYPE SCALAIRE

[Exercice 301 Variables](#)
[Exercice 302 Déclaration de variables](#)
[Exercice 303 Affectation de variables](#)
[Exercice 304 Modification de variables](#)
[Exercice 305 Variables scalaires, permutation](#)
[Exercice 306 Variables scalaires, permutation](#)
[Exercice 307 Autres affectations](#)
[Exercice 308 Cast de variables](#)
[Exercice 309 Cast](#)

4. ENTREES/SORTIES

[Exercice 401 - Niveau 1](#)
[Exercice 402 - Niveau 1](#)
[Exercice 403 - Niveau 2](#)
[Exercice 404 - Niveau 2](#)
[Exercice 405 – Niveau 2](#)

5. CALCULS, COMPARAISONS ET OPERATEURS LOGIQUES

[Exercice 501 - Niveau 1](#)
[Exercice 502 - Niveau 1](#)
[Exercice 503 - Niveau 2](#)
[Exercice 504 - Niveau 2](#)
[Exercice 505 - Niveau 2](#)

6. TESTS

[Exercice 601 - Niveau 1](#)
[Exercice 602 - Niveau 1](#)
[Exercice 603 Niveau 1](#)
[Exercice 604 - Niveau 2](#)
[Exercice 605 - Niveau 2](#)

7. BOUCLES DE REPETITION

[Exercice 701 - Niveau 1](#)
[Exercice 702 - Niveau 1](#)
[Exercice 703 - Niveau 2](#)
[Exercice 704 - Niveau 2](#)
[Exercice 705 - Niveau 2](#)
[Exercice 706 - Niveau 3](#)
[Exercice 707 - Niveau 3](#)
[Exercice 708 - Niveau 3](#)
[Exercice 709 - Niveau 3](#)
[Exercice 710 - Niveau 3](#)

8. SOUS-PROGRAMMES ET PASSAGE DES PARAMETRES PAR VALEUR

[Exercice 801 - Niveau 1](#)
[Exercice 802 - Niveau 2](#)
[Exercice 803 - Niveau 2](#)
[Exercice 804 - Niveau 2](#)

9. POINTEURS ET PASSAGE DES PARAMÈTRES PAR ADRESSE

[Exercice 901 - Niveau 1](#)
[Exercice 902 - Niveau 1](#)
[Exercice 903 - Niveau 1](#)
[Exercice 904 - Niveau 1](#)
[Exercice 905 - Niveau 2](#)
[Exercice 906 - Niveau 2](#)
[Exercice 907 - Niveau 2](#)

10. TABLEAUX

[Exercice 1001 - Niveau 1](#)
[Exercice 1002 - Niveau 1](#)
[Exercice 1003 - Niveau 2](#)
[Exercice 1004 - Niveau 2](#)
[Exercice 1005 - Niveau 2](#)
[Exercice 1006 - Niveau 2](#)
[Exercice 1007 - Niveau 2](#)
[Exercice 1008 - Niveau 3](#)

11. CHAINES DE CARACTÈRES

[Exercice 1101 - Niveau 1](#)
[Exercice 1102 - Niveau 1](#)
[Exercice 1103 - Niveau 2](#)
[Exercice 1104 - Niveau 2](#)
[Exercice 1105 - Niveau 2](#)
[Exercice 1106 - Niveau 2](#)
[Exercice 1107 - Niveau 3](#)
[Exercice 1108 - Niveau 3](#)

12. GÉNÉRATION DE NOMBRES ALÉATOIRES

[Exercice 1201 - Niveau 1](#)

[Exercice 1202 - Niveau 1](#)

[Exercice 1203 - Niveau 2](#)

[Exercice 1204 - Niveau 2](#)

[TP : Image, histogramme et binarisation](#)

[Exercice 1205](#)

[Exercice 1206](#)

[Exercice 1207](#)

13. STRUCTURES

[Exercice 1301 - Niveau 1](#)

[Exercice 1302 - Niveau 1](#)

[Exercice 1303 - Niveau 2](#)

[Exercice 1304 - Niveau 2](#)

[Exercice 1305 - Niveau 3](#)

14. FICHIERS : ASCII ET BINAIRES

[Exercice 1401 - Niveau 1](#)

[Exercice 1402 - Niveau 1](#)

[Exercice 1403 - Niveau 2](#)

[Exercice 1404 - Niveau 2](#)

[Exercice 1405 - Niveau 3](#)

[Exercice 1406 - Niveau 3](#)

15. ALLOCATION DYNAMIQUE DE MEMOIRE

[Exercice 1501 - Niveau 1](#)

[Exercice 1502 - Niveau 2](#)

[Exercice 1503 - Niveau 1](#)

[Exercice 1504 - Niveau 2](#)

[Exercice 1505 - Niveau 2](#)

[Exercice 1506 - Niveau 2](#)

[TP : Gestion de rendez-vous de consultations médicales](#)

16. RECURSIVITE

[Exercice 1601 - Niveau 1](#)

[Exercice 1602 - Niveau 2](#)

[Exercice 1603 - Niveau 2](#)

[Exercice 1604 - Niveau 2](#)

[Exercice 1605 - Niveau 2](#)

[Exercice 1606 - Niveau 3](#)

[Exercice 1607 - Niveau 3](#)

[Exercice 1608 - Niveau 3](#)

[Exercice 1609 - Niveau 3](#)

17. LISTES CHAINEES

[Exercice 1701 - Niveau 1](#)

[Exercice 1702 - Niveau 1](#)

[Exercice 1703 - Niveau 2](#)

[Exercice 1704 - Niveau 2](#)

[Exercice 1705 - Niveau 3](#)

[Exercice 1706 - Niveau 3](#)

[ANNEXE A : SEQUENCES D'ECHAPPEMENT](#)

[ANNEXE B : PRIORITES DES OPERATEURS](#)

[ANNEXE C : CLASSIFICATION, CONVERSION DE CARACTERE : <CTYPE.H>](#)

[ANNEXE D : TRAITEMENT DE CHAINES DE CARACTERES : <STRING.H>](#)

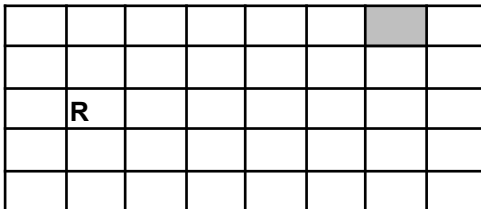
[ANNEXE E : TABLE ASCII ET TABLE ASCII ETENDUE](#)

1. DECOMPOSITION DE PROBLEME ET ALGORITHME

Exercice 101 Instructions

Ecrire le minimum d'instructions qui permet à un robot **R** d'arriver à la case grise. Vous avez ces instructions à votre disposition :

- *Avancer à droite*
- *Avancer à gauche*
- *Monter d'une case*
- *Descendre d'une case*



Exercice 102 DTI, ACD

Ecrire le DTI et une ACD qui saisit 5 réels et les affiche à l'écran. Cet exercice se fait sans boucle.

Exercice 103 DTI, algorithme

Vous devez calculer le carré d'un entier donné par l'utilisateur.

Ecrire le DTI et l'algorithme pour obtenir le résultat demandé.

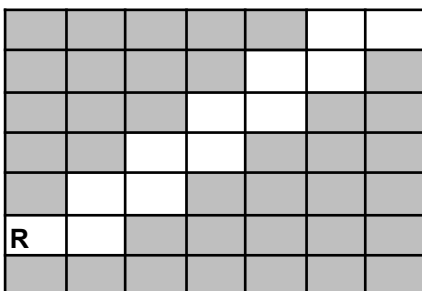
Exercice 104 Boucle : Instructions

Ecrire le minimum d'instructions qui permet à un robot **R** de monter cet escalier. Vous avez ces instructions à votre disposition :

- *Avancer à droite*
- *Avancer à gauche*
- *Monter d'une case*
- *Descendre d'une case*
- *Pour X fois Faire*

Bloc instructions

Fpour



Remarque :

Dans le cas de la boucle, « Pour X fois Faire.....Bloc instructions...Fpour »,

- X est un nombre à modifier en fonction de ce que le robot doit faire
- Bloc d'instructions est un bloc qui peut contenir de instructions simples (avancer, descendre, ...) mais aussi d'autres boucles si besoin

Exercice 105 Boucle : DTI, algorithme

Ecrire un DTI, un algorithme (avec une boucle) qui affiche 20 entiers saisis. Chacun est compris dans l'intervalle [-55 ; 100]

Exercice 106 Boucle : DTI, ACD

Ecrire un DTI et une ACD qui saisit un entier et affiche à l'écran la factorielle de cet entier.

Exercice 107 Boucle : DTI, algorithme

Ecrire un DTI et un algorithme qui saisit dix entiers et qui affiche la moyenne de ces nombre entiers.

Exercice 108 Boucle et test : Instructions

Ecrire le minimum d'instructions qui permet à un robot **R** d'aller d'un bord du terrain vers un autre tout en ramassant les euros € qui se trouve sur son chemin et en les mettant dans sa bourse dans qu'il se trouve sur une case .


Vous avez ces instructions à votre disposition :


- *Avancer à droite*
- *Descendre d'une case*
- *Pour X fois Faire*

Bloc instructions

Fpour

- *Si Condition Alors Bloc d'instructions FSI*
- *Si Condition Alors Bloc d'instructions Sinon Bloc d'instructions FSI*
- *Mur à droite*
- *Mur en bas*
- *Argent*
- *Porte-monnaie*
- *Ramasser l'argent*
- *Déposer l'argent*
- *Sortie du terrain*

R												
		€										
												

								€				
												

Remarque :

Dans le cas de la boucle, « Pour X fois Faire.....Bloc instructions...Fpour »,

- X est un nombre à modifier en fonction de ce que le robot doit faire
- Bloc d'instructions est un bloc qui peut contenir de instructions simples (avancer, descendre, ...) mais aussi d'autres tests et boucles si besoin

Dans le cas des tests ;

- Conditions est à remplacer selon l'obstacle se trouvant sur le chemin du robot **R**
- Bloc d'instructions est un bloc qui peut contenir de instructions simples (avancer, descendre, ...) mais aussi d'autres tests et boucles si besoin

Exercice 109 Boucle et test : DTI, algorithme

Ecrire un DTI et un algorithme qui :

- saisit un réel. Ce réel est considéré comme le montant des achats effectués.
- saisit un entier. Ce nombre est considéré comme l'argent donné par le client pour payer ses achats. Cette somme doit être supérieure ou égale à la précédente
- afficher le nombre de pièces (1 cent, 2 cent, 5 cent, 10 cent, 20 cent, 50, cent, 1 euro, 2 euro) ainsi que le nombre de billets (5 euro, 10 euro, 20 euro, 50 euro, 100 euro, 500 euro) nécessaires pour rendre la monnaie au client à la suite de ses achats. La monnaie sera toujours rendue avec le moins de pièces ou de billets possible.

Exercice 110 Boucle et test : DTI et organigramme

Vous devez calculer la somme des entiers compris entre un entier A et un autre entier B saisis, bornes comprises.

Les contraintes sont :

- interdiction d'utiliser la formule mathématique \sum
- lecture des entiers A et B

Ecrire le DTI , l'organigramme pour obtenir le résultat demandé.

Exercice 111 Cas d'étude : ACD et organigramme

Vous avez un fichier affiché à votre écran. Vous souhaitez l'imprimer.

Ecrire une ACD et un organigramme pour obtenir l'impression de votre document

Exercice 112 Cas d'étude : ACD et organigramme

Vous devez construire cette table.



Les pièces qui sont à votre disposition sont :

- 1 haut de la table
- 4 pieds de table
- 2 morceaux de bois pour éviter l'écartement de deux pieds
- 1 traverse de bois centrale

Ecrire une ACD et un organigramme pour obtenir cette table à partir des pièces fournies.

Exercice 113 Cas d'étude :

Ecrire l'algorithme décrivant la programmation d'un automate distributeur de billets de banque.

Hypothèses : En l'absence, à vous de poser les vôtres.

Exercice 114 Cas d'étude : boucle

Vous devez afficher les n nombres naturels en commençant par 1. Les contraintes sont :

- lecture de l'entier n

Ecrire le DTI, l'algorithme pour obtenir le résultat demandé dont n est saisi par l'utilisateur.

Ex : **Saisie de n** : 10
Résultat affiché : 1 2 3 4 5 6 7 8 9 10

Exercice 115 Cas d'étude : boucle

Vous devez calculer le résultat de rang n de la suite récurrente suivante

$$U_0 = -16$$

$$U_n = U_{n-1} / 2 + 4$$

Les contraintes sont :

- obligation d'utiliser des boucles pour calculer le résultat
- lecture de l'entier n et blindage (=test si la valeur est exploitable)

Ecrire le DTI, l'algorithme pour obtenir le résultat demandé dont n est donné par l'utilisateur.

Exercice 116 Résultat d'algorithme

Vous avez cet algorithme. Que fait cet algorithme ?

```
Initialisation :  
    u prend la valeur de 7.7  
    s prend la valeur de 0  
Pour i de 1 à 8 faire  
    s prend la valeur de s+u  
    u prend la valeur de  
1.775*u  
FinPour  
  
Afficher s
```

Exercice 117 Résultat d'algorithme

Vous avez cet algorithme. Que fait cet algorithme ?

```
Début  
    entier : A, res, i  
    Saisie de la valeur A  
    Res <- 1  
  
    Pour i de 1 à 8 faire  
        Res <- Res * A  
    FinPour  
  
    Afficher Res  
Fin
```

Exercice 201 Ordinateur

Faire des recherches sur les notions suivantes :

- composants externes et internes d'un ordinateur
- processeur
- formatage du disque dur
- mémoire vive, disque dur
- principe de compilation
- code compilé, code interprété

Exercice 202 Arborescence des fichiers

- Quelle différence existe-t-il entre un répertoire et un fichier ?
- Quel type de fichier connaissez-vous ? Comment connaît-on le type d'un fichier ?
- Quelles informations nous donnent les propriétés d'un fichier ?
- Comment sont organisés les fichiers et les répertoires sur l'ordinateur ? Faites un schéma rapide.
- Quel est le principe de changement d'emplacement d'un fichier d'un répertoire à un autre ?
- Quel est celui principe de la suppression d'un fichier de l'ordinateur?

Exercice 203 Environnement de travail : CodeBlocks

- **Pré-requis** : installation de CodeBlocks avec compilateur
- Ouvrir CodeBlocks
- Explorer le menu de l'environnement
- Créer un projet (langage c)
- Regarder sur l'explorateur de fichiers ce qui a été créé pour ce nouveau projet
- Compiler le programme et regarder sur l'explorateur de fichiers ce qui a été créé.
- Exécuter le programme à partir de CodeBlocks. Que voyez sur la fenêtre console affichée à l'écran ?
- Avant de fermer la fenêtre console, regarder la propriété de la fenêtre pour modifier l'affichage
de caractères, du curseur, pour modifier la taille de la fenêtre : se positionner sur la barre supérieure de la fenêtre, faire un click droit et choisir l'option 'propriétés'

3. STOCKAGE DES DONNEES : VARIABLES DE TYPE SCALAIRE

Exercice 301 Variables

- Combien y a-t-il de types scalaires en C ? Nommez-les et donner leur plage de valeurs ?
- Par défaut, sont-ils signés ou non signés ?
- Quelle est la place de la déclaration des variables locales dans un code bien écrit ?
- Comment déclarer une variable ?
- A quoi sert l'indentation d'un programme ?

Exercice 302 Déclaration de variables

Dans un programme, les déclarations suivantes ont été écrites. Indiquer, expliquer et corriger les erreurs rencontrées.

- `int 0t, ti, p0;`
- `freste float;`
- `double div-total;`
- `float tata, t2345, char c, cc :`
- `short Err_, _E_,`

Exercice 303 Affectation de variables

- Qu'est-ce qu'une affectation ?
- Ouvrir CodeBlocks.
- Déclarer une variable de type entier, nommée E1.
- Donner la valeur 78 à E1.
- Donner la valeur 100 à E1.
- Donner la valeur 100.3 à E1.
- Quelle valeur aura E1 en mémoire ? Faites un schéma mémoire.
- Déclarer une variable de type réel, nommée R1.
- Donner la valeur 45.2 à R1
- Donner la valeur 456 à R1.
- Quelle valeur aura R1 en mémoire ? Faites un schéma mémoire.
- Déclarer une variable de type caractère, nommée Carac.
- Donner la valeur du caractère 'j' à Carac.
- Quelle valeur aura Carac en mémoire ? Faites un schéma mémoire.
- Est-ce possible de donner la valeur 65 à Carac ? Si oui, faites-le, si non expliquez pourquoi.

Exercice 304 Modification de variables

- Déclarer trois variables :
 - **ent1** de type entier
 - **nbReel** de type réel ou flottant
 - **initiale** de type caractère
- Affecter 10 à la variable **ent1**.
- Affecter 5.5 à la variable **nbReel** et multiplier par 3 **nbReel**.
- Ajouter à ent1 la valeur de **nbReel**. Quelle est la valeur en mémoire de chacune des deux variables ?

- Affecter la valeur 'F' à la variable **initiale**. Est-ce possible d'ajouter 2 à **initiale**? Justifier votre réponse en donnant dans le cas positif la valeur de **initiale**.
- Affecter la valeur de **initiale** à **ent1** et à **nbReel**. Quelles sont les valeurs de ces variables ?
- Affecter la valeur 1000 à **ent1**. Affecter la valeur de **ent1** à **initiale**. Est-ce possible ? Justifier votre réponse.
- Représenter la trace d'exécution

Exercice 305 Variables scalaires, permutation

Soit deux variables **lettre1** et **lettre2** de type caractère.

- Affecter une valeur à chaque variable
- Réaliser l'échange des contenus des 2 variables sans perdre les valeurs affectées

Exercice 306 Variables scalaires, permutation

Soit deux variables **mesure1** et **mesure2** de type réel.

- Affecter une valeur à chaque variable
- Multiplier par 3 **mesure1**
- Réaliser l'échange des contenus des 2 variables sans perdre les valeurs affectées
- Faire un schéma mémoire

Exercice 307 Autres affectations

Soit les instructions suivantes :

```
// 0. Déclaration des variables
    int compteur, indice;

// 1. Début du code
    compteur = 10 ;
    indice = compteur;
    ++ compteur ;
    compteur ++ ;
    indice*=2 ;
    compteur = ++compteur *4 ;
    indice = indice*7+45/5 ;
    compteur = compteur*(3-54/6) ;
```

- Quelle différence entre les instructions `compteur++` et `++compteur` ?
- Réécrire l'instruction : `indice*=2` ;
- Faire un schéma mémoire de ces instructions et donner la valeur de ces variables

Exercice 308 Cast de variables

Soit le code suivant :

```
float x;

float y;
```

```
y = (float)22/(float)7;
```

```
x = 22/7;
```

Quelles sont les valeurs de x et y? Expliquez ce qui s'est passé.

Exercice 309 Cast

Ecrire un programme permettant de tester tous les cas de figure de la division en C afin d'illustrer le problème des CAST.

- division d'un int par un int, rangé dans un int et affichée en %d
- division d'un int par un int, rangé dans un float et affichée en %f
- division d'un float par un int, rangé dans un float et affichée en %f
- division normale stockée dans un réel (entier impair divisé par entier pair) et affichée en %f
- division castée stockée dans un réel (entier impair divisé par entier pair) et affichée en %f
- ...

Jouez sur les types des opérandes, les types de la variable de récupération du résultat, le format d'affichage du résultat à l'écran...

4. ENTREES/SORTIES

Exercice 401 - Niveau 1

- Dans un programme, afficher à l'écran « Bonjour » sans passer par une variable.
- Compléter votre programme en saisissant au clavier l'année en cours. Cette valeur est à stocker dans une variable déclarée. Enfin afficher cette valeur à l'écran.
- Si la valeur 2021 est saisie, afficher à l'écran « L'annee en cours est 2021 ». C'est-à-dire afficher un texte avec la valeur entière saisie. Mettre un passage à la ligne.
- Afficher l'opposé de l'entier saisi précédemment avec un passage à la ligne.
- Compléter votre programme, en déclarant 3 entiers (nommés note1, note2, note3) et 1 réel (nommé *moyenne*). Saisir les trois entiers. Calculer la moyenne correspondante et affectant le résultat à la variable dédiée. Afficher, toutes les données en jeu : celles saisies et celle calculée.

Exercice 402 - Niveau 1

Dans un programme, déclarer une variable de type caractère et lui affecter une valeur entière choisie au hasard (positive ou négative).

Afficher la valeur de cette variable en utilisant les deux formats %d et %c, Que remarquez-vous ?

Réessayer en demandant cette fois une valeur comprise entre 97 et 122. Que remarquez-vous ? Quelles valeurs permettent d'afficher les caractères : *, @, #, \$?

Exercice 403 - Niveau 2

Ecrire un algorithme et un programme qui permet de convertir un caractère de minuscule en majuscule. Chaque caractère est saisi par l'utilisateur puis converti. Chaque caractère sera saisi. Le résultat de la conversion sera affiché.

Vous devez baser votre raisonnement sur le codage de l'information dans la table d'interprétation sémantique ASCII (mais sans entrer de valeur « en dur » genre 32) et sur le décalage Min/Maj.

Remarques :

La fonction C « toupper » ne doit pas être utilisée.

Exercice 404 - Niveau 2

Ecrire un algorithme et un programme qui à partir d'un entier saisi affiche le nombre saisi et son carré.

Exercice 405 – Niveau 2

Ecrire un programme qui affiche à l'écran la taille des types : entier, caractère et réel.

Remarque : Utiliser l'opérateur **sizeof()**.

IMPORTANT

Règles d'écriture des programmes en C à l'ECE

Vous êtes ici pour apprendre à « bien » programmer. Or, chaque entreprise à ses propres règles de « bonne programmation ».

Vous allez donc commencer par apprendre à respecter les règles de ceux pour qui vous travaillez. Il sera toujours bien assez tôt, le jour où vous pourrez faire ce que bon vous semble.

A l'ECE, nous appliquons les règles suivantes, parfois différentes de l'entreprise.

- Les **sous-programmes ne devraient pas excéder 25 lignes**. Les décomposer sinon.
- Il faut **éviter plus de 5 paramètres** par sous-programmes.
- Les **noms** de variables ou de sous-programmes doivent être **explicites** mais pas trop longs.
- Les fonctions et les variables sont écrites en minuscule ou commencent par une minuscule (voir notation hongroise).
- Les **goto** sont **interdits** (sauf les gotoxy ou gotoligcol qui servent à positionner le curseur à l'écran)
- Aucune variable globale
- Le code doit être **commenté** et correctement indenté.

5. CALCULS, COMPARAISONS ET OPERATEURS LOGIQUES

Exercice 501 - Niveau 1

- Qu'est-ce qu'un booléen ? Quand obtient-on une réponse booléenne ?
- En C, existe-t-il un type booléen. Si non, comment simule-t-on un booléen ?
- En C, quelle est la valeur booléenne de la lettre 'k', du caractère '0' (zéro) et du chiffre 0 ?

Exercice 502 - Niveau 1

Afin d'analyser des résultats d'examen, 4 variables permettent de décrire l'environnement : les variables numériques Nlv, Nf, Nm, Np qui indiquent respectivement, pour un candidat donné :

- o des notes littéraires : langue vivante (Nlv), français (Nf)
- o des notes scientifiques : mathématiques (Nm), et physique (Np).

On suppose que les notes sont calculées sur 20 et qu'elles ont toutes le même coefficient.

Formez les expressions logiques (et seulement elles) correspondant aux situations suivantes :

- la moyenne des quatre notes est supérieure à 10. Vous nommerez cette moyenne : *moyenneG*
- les notes de mathématiques et de français sont supérieures à la moyenne des quatre notes, *moyenneG*
- il y a au moins une note supérieure à 10
- toutes les notes sont supérieures à 10
- la moyenne (10) est obtenue pour l'un des deux types (littéraire et scientifique)
- la moyenne des quatre notes, *moyenneG*, est supérieure ou égale à 10 et la moyenne (10) est obtenue pour l'un des deux types

Ecrire un programme qui met en place toutes ces instructions. Afficher les résultats de chacun des points traités par le message qui convient. A la saisie, vous vérifiez que la valeur de toutes les notes est comprise entre 0 et 20.

Exercice 503 - Niveau 2

Etablir les tables de vérité pour les opérateurs logiques ET, OU, NON.

a	b	a ET b	a OU b	NON a
Vrai	Vrai			
Vrai	Faux			
Faux	Vrai			
Faux	Faux			

A partir de ces résultats, indiquez les valeurs des expressions logiques suivantes selon les valeurs de a et b :

a	b	a ET (NON b)	b ET (NON a)	(a ET (NON b)) OU (b ET (NON a))
Vrai	Vrai			

Vrai	Faux			
Faux	Vrai			
Faux	Faux			

Exercice 504 - Niveau 2

A partir de ces tables, déterminer la valeur booléenne de ces expressions.

Données de départ :

int i, a;
float f;

i = 7;
a = -2 ;
f = 5.5;

Expression à évaluer sur papier :

f > 5
(i + f) <= 1
a >= -10*(i + f)
(i >= 6) && ((a %2) !=0)
(i >= 6) || ((a %2) !=0)
(f < 11) && (i > 100)

Exercice 505 - Niveau 2

Vous savez que la machine ne se préoccupe pas de l'indentation de vos programmes. Cela n'apporte qu'un confort pour les lecteurs de code.

Pour vous en convaincre, voici des algorithmes écrits avec une **indentation un peu vague**. Essayez de vous y retrouver et d'en comprendre l'importance de respecter une indentation correcte dans vos programmes.

Rappelez-vous que sans bloc, la machine ne conditionne ou ne répète qu'une seule instruction.

Appliquez ces règles sur l'exemple ci-dessous en tentant de suivre l'évolution du contenu des variables X, Y et Z au fur et à mesure de l'exécution du code en réalisant une trace d'exécution (ce qui revient à faire tourner le code à la main).

X, Y, et Z étant des variables numériques, on considère les deux séquences algorithmiques S1 et S2.

Pour chacune des deux séquences, donner les traces d'exécution de X, Y, et Z si l'on suppose qu'à l'état initial ces trois variables ont les valeurs :

- | | | |
|----------|-------|-------|
| a) X :=4 | Y :=1 | Z :=4 |
| b) X :=4 | Y :=5 | Z :=4 |
| c) X :=1 | Y :=3 | Z :=1 |

Priorité des opérateurs : **ANNEXE**

Testez les codes qui suivent sur papier puis sur Codblocks pour vous assurer d'avoir trouvé les bonnes réponses.

<u>Séquence S1</u>	<u>Séquence S2</u>
<pre>{ if ((X<5 Y>2) &&(Z>3)) { X =1 ; if (Z-Y) >0 Z =10 ; Y =Y+Z ; } else X =2 ; Z =Y+Z ; }</pre>	<pre>{ if ((X<5) ((Y>2) && (Z>3))) { X =1 ; } if ((Z-Y) >0) { Z =10 ; Y =Y+Z ; } else { X =2 ; Z =Y+Z ; } }</pre>

Rappel : en l'absence de bloc début-fin ou {-} les boucles et les tests ne portent que sur l'instruction qui suit immédiatement.

6. TESTS

Exercice 601 - Niveau 1

Ecrire l'algorithme, l'organigramme puis le programme permettant de déterminer si une valeur saisie est multiple de 5, et, supérieure ou égale à -3 ?

Exercice 602 - Niveau 1

Donner l'algorithme, l'organigramme et le programme qui déterminent le nombre de valeurs différentes saisies parmi trois variables.

ex :

Saisie	Message Affiché
8 8 8	Une seule valeur
4 5 4	Deux valeurs différentes
7 9 3	Trois valeurs différentes

Exercice 603 Niveau 1

Ecrire l'algorithme et le programme qui déterminent si une variable saisie au clavier par l'utilisateur est paire ou non.

Le message affiché est : soit « la valeur est paire », soit « la valeur est impaire ». Cela dépend de la parité de la valeur saisie.

Remarques :

Utiliser les opérateurs vus.

Exercice 604 - Niveau 2

Ecrivez l'algorithme, l'organigramme et le programme permettant de déterminer si le caractère entré par l'utilisateur est un chiffre, une lettre majuscule, une lettre minuscule ou autre chose, en vous efforçant à nouveau d'exploiter la table ASCII.

Table ASCII : **ANNEXE**

Exercice 605 - Niveau 2

Ecrire un programme qui teste si un nombre est positif ou non. Ce test se fera avec les deux formules :

- **if (condition) { instructions de then } else { instruction de else }**
- **(condition) ? instruction de then : instruction de else ;**

7. BOUCLES DE REPETITION

Exercice 701 - Niveau 1

- Ecrire l'algorithme et le programme qui affiche à l'écran 10 fois la même chaîne « Bienvenue en TP : indice x ». Vous ajouterez à la fin de cette chaîne le numéro de l'indice de boucle **for** tel que présenté ci-dessous
 Bienvenue en TP : indice 1
 Bienvenue en TP : indice 2
 Bienvenue en TP : indice 3
 Bienvenue en TP : indice 4
 Bienvenue en TP : indice 5
 Bienvenue en TP : indice 6
 Bienvenue en TP : indice 7
 Bienvenue en TP : indice 8
 Bienvenue en TP : indice 9
 Bienvenue en TP : indice 10
- Modifier votre programme saisissant le nombre d'affichages à faire dans cette boucle. Le nombre saisi doit toujours être positif (c'est un blindage à écrire avec une boucle **do...while**)
- Réécrire le programme avec les mêmes fonctionnalités avec une boucle **while**

Exercice 702 - Niveau 1

Ecrire l'algorithme et le programme qui affiche les nombres de 2 à 98 allant de 4 en 4. Nous aurons à l'écran les premiers affichages suivants :

2
6
10
14
18
Etc.

Réécrire le programme avec une boucle **while**.

Exercice 703 - Niveau 2

Ecrire l'algorithme et le programme C qui affiche les multiples de 3 compris entre l'entier positif saisi et son opposé.

Exemple :

Saisie	15	10
Affichage	-15	-9
	-12	-6
	-9	-3
	-6	0
	-3	3
	0	6
	3	9
	6	
	9	
	12	
	15	

Exercice 704 - Niveau 2

1- Ecrire l'algorithme, l'organigramme et le programme calculant la moyenne de nombres entiers positifs saisis par l'utilisateur. Quand un nombre négatif est saisi, le programme affiche le nombre de valeurs positives saisies et la valeur de leur moyenne.

Faites tourner votre solution à la main sur des exemples en indiquant la trace d'exécution listant l'évolution du contenu de chacune des ressources. Vous penserez à tous les cas possibles.

2- Ajouter à l'algorithme/organigramme et au programme, la fonctionnalité suivante :

A chaque affichage des résultats, le programme demande explicitement à l'utilisateur s'il souhaite quitter le programme ou non. Si ce n'est pas le cas le programme recommence automatiquement depuis le début.

Exercice 705 - Niveau 2

Ecrire l'algorithme, l'organigramme et le programme qui convertit un entier naturel en chiffres romains, en utilisant l'ancienne notation **uniquement additive** : exemple : 4 (IIII), 9 (VIIII), 900 (DCCCC)

La valeur convertie sera affichée à l'écran sans être stockée en mémoire.

Rappelons les éléments de base :

I : 1, V : 5, X : 10, L : 50, C : 100, D : 500, M : 1000.

Exercice 706 - Niveau 3

Simulation d'épargne.

Vous épargnez une somme de 30€.

Tous les mois, votre épargne vous rapporte 0.3% mensuel d'intérêts. Les intérêts se cumulent.

Ecrivez l'algorithme, l'organigramme et le programme permettant d'afficher le solde de votre épargne au bout d'un nombre de mois saisi par l'utilisateur.

Remarque :

Ce calcul se fera à l'aide d'une boucle et non d'une formule mathématique.

Exercice 707 - Niveau 3

Ecrire l'algorithme, l'organigramme et le programme qui affiche les **nombres premiers** compris entre 1 et l'entier positif saisi. Cet entier sera forcément supérieur à 1 tant que ce n'est pas le cas, il faut boucler sur la saisie.

Exercice 708 - Niveau 3

Ecrire l'algorithme, l'organigramme et le programme qui saisit un entier n et affiche le résultat U(n) défini par :

$$U(0) = 3$$

$$U(n+1) = 3.U(n)+4$$

Le traitement se fera par une boucle

Exercice 709 - Niveau 3

Ecrire l'algorithme, l'organigramme et le programme qui saisit un entier n et affiche le résultat $U(n)$ défini par :

$$U(0) = 1$$

$$U(1) = 1$$

$$U(n+1) = U(n) + U(n-1)$$

Exercice 710 - Niveau 3

Ecrire un programme qui propose à l'utilisateur de dessiner à l'écran certaines figures composées d'étoiles. La figure **ne sera pas stockée** dans une matrice en mémoire.

Les figures seront proposées par un menu (triangle, carré, sablier...). La hauteur de la figure sera saisie par l'utilisateur. Cette saisie est à blinder.

Exemple : hauteur = 4

```
  *
 * * *
 * * * *
 * * * * *
 * * * * * *
```

```
* * * *
* * * *
* * * *
* * * *
```

```
* * * * * * *
* * * * *
* * *
*
*
* * *
* * * * *
* * * * * *
```


8. SOUS-PROGRAMMES ET PASSAGE DES PARAMETRES PAR VALEUR

Exercice 801 - Niveau 1

Ecrire l'algorithme et le code des fonctionnalités suivantes :

- Ecrire un sous-programme qui calcule le carré d'un nombre passé en paramètre et qui rend le résultat de son calcul
- Ecrire un programme principal qui :
 - saisit deux nombres entiers
 - calcule les carrés des deux entiers saisis (appel du sous-programme précédent pour chacun des entiers)
 - affiche les résultats obtenus

Exercice 802 - Niveau 2

- Ecrire le sous-programme nommé *menu* qui propose les opérations suivantes :
 1. Ajouter 1
 2. Multiplier par 2
 3. Soustraire 4
 4. Somme de n premiers entiers (n étant passé en paramètre)
 5. Quitter

Le sous-programme demande alors de taper un entier entre 1 et 5. Il rendra au programme appelant la valeur saisie.

Gérer le blindage de la valeur saisie.

- Ecrire un sous-programme pour chaque opération que le menu propose.
- Ecrire le programme principal qui :
 - saisit un entier
 - affiche le menu
 - appelle le sous-programme précédent selon la valeur récupérée du menu et en passant l'entier saisi en paramètre.

Le programme principal récupère la valeur de résultat de l'opération et l'affiche.
 - s'exécute tant que l'utilisateur ne demande pas de sortir c'est-à-dire qu'il se termine lorsqu'on tape 5 (option de sortie du menu)
- Dessiner le graphe d'appel

Exercice 803 - Niveau 2

Ecrire l'algorithme et le code qui simulent une horloge :

- Ecrire un sous-programme qui, à partir d'un nombre passé en paramètre, calcule la minute suivante et rend le résultat. Ce résultat sera rendu au programme appelant. La valeur des heures varie entre 0 et 59.
- Ecrire un sous-programme qui calculera l'heure suivante à partir d'une heure passée en paramètre et rendra au programme appelant le résultat. La valeur des heures varie entre 0 et 23.
- Ecrire ce programme principal : il se charge d'initialiser l'heure à 0h00 et affichera à chaque tour de boucle l'heure de votre horloge qui se met à jour. Pour avoir le temps de voir les affichages, mettez un temps d'attente avec l'instruction ***sleep(1)*** de la bibliothèque ***unistd.h***.
- Dessiner le graphe d'appel de cet exercice

Version élaborée :

- Le programme principal peut saisir les heures et les minutes de départ de l'horloge.
- Le programme principal peut saisir les heures et les minutes d'arrêt de l'horloge.
- Mettre en place une horloge dont l'affichage des heures va de 0 à 11 mais avec la précision AM (*ante meridiem*) ou PM (*post meridiem*) selon le moment de la journée.

Exercice 804 - Niveau 2

- Ecrire un sous-programme recevant deux entiers x1 et x2 par valeur. Le sous-programme affiche la valeur des paramètres reçus, ajoute **10** à l'entier x1 puis affiche de nouveau la valeur des paramètres reçus.
- Ecrire ensuite le programme principal qui :
 - o demande à l'utilisateur la saisie de 2 entiers val1 et val2
 - o affiche leur valeur respective
 - o appelle le sous-programme précédent
 - o affiche val1 et val2 après l'appel pour vérifier leur contenu.
- Que constatez-vous au sujet de val1 et val2 ?
- Faites un schéma mémoire.
- Qu'aurait-il fallu faire pour que la valeur des modifications effectuée dans le sous-programme se répercute au niveau du programme principal ?

IMPORTANT

Pour tous les exercices de ce chapitre et ce jusqu'à la fin de l'année, il est demandé de structurer votre projet Code Blocks ainsi :

- Un fichier pour le programme principal
- Un fichier pour les sous-programmes
- Un fichier header

9. POINTEURS ET PASSAGE DES PARAMÈTRES PAR ADRESSE

Exercice 901 - Niveau 1

Soit le code suivant :

```
int* pt ;
int val1, val2, val3 ;
val1 = 10 ;
val2 = 20 ;
val3 = 1 ;
pt = &val1 ;
*pt = 100 ;
```

Que représente pt, &pt et *pt ? Faire un schéma mémoire de cette séquence de code.

val1	pt+1	pt+2	val2		val3		pt
108	?	?	20		1		&val1

Que représente les instructions suivantes : attention à la priorité de opérations

- *pt + 3 103
- pt + 2 adresse de val1 + 2
- *pt += 5 ; *pt=*pt+5 = 105
- (*pt)++ ; ++ après prioritaire sur * 106 *pt++ : d'abord pt++ puis le *
- ++*pt ; 107 le ++ avant donc pas besoin de parenthèses
- *(pt + 2)++ ; ?
- *pt += val3 ; 108 *pt = *pt +val3;
- *(pt + val3) ; ?

Exercice 902 - Niveau 1

- Déclarer un entier val1, un caractère non-signé val2, et un caractère nommé lettre.
- Déclarer un pointeur sur chacune de ces trois variables nommés pt1, pt2 et pt3.
- Déclarer un pointeur sur chacun des 3 pointeurs précédents que vous nommerez ppt1, ppt2 et ppt3.
- Réaliser les affectations permettant de « raccrocher » tous les pointeurs sur leur variable respective.
- Initialiser les 3 premières variables par une affectation directe et affichez leur contenu à l'écran.
- Modifier les 3 premières variables en utilisant cette fois une affectation indirecte à 1 indirection.
- Afficher de nouveau la valeur des 3 variables val1, val2 et lettre en utilisant cette fois la première indirection
- Modifier les 3 variables par une affectation utilisant la 2ème indirection.
- Afficher de nouveau la valeur des 3 variables val1, val2 et lettre en utilisant cette fois la deuxième indirection.
- Faire un schéma mémoire de toutes les étapes de cet exercice.

Exercice 903 - Niveau 1

- Ecrire l'analyse et le sous-programme qui :
 - Calcule la somme de deux entiers reçus en paramètre
 - La somme est passée en paramètre par adresse afin que le main() puisse récupérer sa valeur
- Ecrire le programme qui :
 - Saisit deux nombres entiers
 - Appelle le sous-programme précédent
 - Affiche le résultat
- Faire le schéma mémoire

Que changez-vous, dans le programme, si le sous-programme devient une fonction, c'est-à-dire si le sous-programme rend le résultat ? Modifier le main() en conséquence.

Faire le schéma mémoire.

Exercice 904 - Niveau 1

- Ecrire l'algorithme et le sous-programme qui :
 - saisit des entiers négatifs. La saisie se terminera dès qu'un nombre strictement positif est saisi.
 - calcule le maximum, le minimum et la moyenne de cette série d'entiers négatifs (sans stockage des valeurs saisies dans un tableau).
 - Le dernier nombre (l'entier positif) indiquant la fin de saisie ne sera pas pris en compte dans les calculs.
 - La moyenne sera rendue comme un résultat normal (instruction **return**). Le maximum et le minimum seront passés par adresse en paramètre.
- Ecrire l'algorithme et le programme principal qui :
 - appelle le sous-programme précédent
 - affiche les trois valeurs rendues par le sous-programme.
- Faire un schéma mémoire
- Dessiner le graphe d'appel

Exercice 905 - Niveau 2

- Ecrire le sous-programme qui :
 - saisit l'identifiant d'un article, son prix unitaire HT (hors taxe) et le taux de TVA (taxe sur la valeur ajoutée)
 - calcule le prix TTC (toute taxe comprise) de l'article
 - rend toutes les données saisies (identifiant, prix unitaire, TVA) et celles calculées, (passage par adresse)
 - rend la donnée du produit (prix TTC) par l'instruction **return**
- Ecrire le programme principal qui :
 - appelle le sous-programme précédent
 - affiche toutes les valeurs rendues par les sous-programmes
- Faire un schéma mémoire
- Dessiner le graphe d'appel

Exercice 906 - Niveau 2

- Ecrire l'algorithme et le sous-programme qui calcule l'aire et le périmètre :
 - d'un rectangle
 - d'un triangle rectangle
 - d'un cercle

Le sous-programme recevra en paramètre le type de la figure 'c' pour cercle, 'r' pour rectangle ou 't' pour triangle.

Les paramètres du sous-programme seront le type de la figure et les deux résultats calculés (aire et périmètre). Les calculs sont à rendre au programme appelant. Le reste des données utiles au calcul demandé sera saisi dans le sous-programme. Les données saisies seront celles utiles en fonction du type de la figure demandée :

 - pour les rectangles : longueur et largeur
 - pour les triangles rectangle : la longueur des côtés de l'angle droit
 - pour les cercles : le rayon
- Ecrire le programme principal qui :
 - s'exécute tant que l'utilisateur ne demande pas la sortie
 - saisit l'initiale minuscule de la forme choisie (rectangle, triangle ou cercle)
 - affiche les résultats obtenus du sous-programme appelé
 - termine son exécution à la demande de l'utilisateur sinon il boucle pour un nouveau calcul
- Faire un schéma mémoire
- Dessiner le graphe d'appel

Remarque :

Toutes les saisies sont à blinder.

Vous utiliserez $\pi = 3.14$ comme une constante

Exercice 907 - Niveau 2

- Ecrire l'algorithme et le sous-programme qui :
 - calcule le prix d'un billet (prix de base du billet 50 euro) à partir d'un âge donné :
 - pour les enfants (moins de 5 ans), le billet est gratuit
 - pour les enfants/adolescents (entre 5 et 17 ans), la réduction du billet est de 75%
 - pour les jeunes adultes (entre 18 et 29 ans), la réduction du billet est de 50%
 - pour les personnes (entre 30 et 60 ans), le tarif est normal (aucune réduction)
 - pour les personnes âgées, aucun billet n'est rendu. (Voir avec votre chargé de TP la convention pour exprimer ce cas)

N'oubliez pas de blinder les saisies.
 - rend le prix du billet demandé (paramètre par adresse).
- Ecrire l'algorithme et un sous-programme qui calcule le prix pour un groupe de personnes de la même tranche d'âge.
 - Vous saisissez le nombre de billets à acheter et une valeur de la tranche d'âges des personnes concernées. (1 accompagnateur pour 5 mineurs -2 accompagnateurs maximum- sinon 1 accompagnateur pour 10 majeurs).

Pour le calcul de l'ensemble du groupe, vous devez appeler le sous-programme précédant. Le résultat du calcul sera rendu au programme principal (le total du prix du billet calculé est renvoyé par l'instruction **return**, les autres seront rendus par adresse).
- Ecrire le programme principal qui :
 - calcule le prix de billet d'une classe d'élèves ou d'étudiants avec leurs accompagnateurs
 - affiche le prix total des billets d'un groupe avec le nombre et l'âge des participants ainsi que ceux des accompagnateur(s)
 - recommence tant que l'utilisateur n'aura pas demandé de quitter l'application
- Dessiner le graphe d'appel

10. TABLEAUX

Exercice 1001 - Niveau 1

Ecrire l'algorithme et le programme qui :

- remplit un tableau de 13 entiers saisis par l'utilisateur. Le tableau est à une dimension.
- affiche tous les éléments de ce tableau à l'écran. Pensez à bien séparer chaque élément par un caractère d'échappement : nouvelle ligne ou tabulation (**ANNEXE**)

Exercice 1002 - Niveau 1

- Dessiner la représentation schématique d'un tableau défini comme suit : float
tab[7][4]
7 lignes
4 colonnes
- Faire le programme qui saisit et affiche ce tableau.

Exercice 1003 - Niveau 2

- Ecrire un sous-programme permettant le remplissage d'un tableau de 10 notes (notes réelles et blindées entre 0.0 et 20.0). Le tableau sera reçu en paramètre.
- Ecrire un sous-programme permettant l'affichage d'un tableau de 10 notes passé en paramètre : une note par ligne.
- Ecrire un sous-programme permettant :
 - la saisie et l'affichage de 10 notes stockées dans un tableau en appelant les sous-programmes précédents
 - le calcul de la moyenne de ces 10 notes. La valeur de la moyenne est rendue au programme appelant
- Ecrire un sous-programme permettant la recherche du maximum et du minimum dans un tableau de 10 notes. Les résultats sont à retourner à l'appelant.
- Ecrire le programme principal qui met en place tous ces sous-programmes
- Dessinez le graphe d'appel correspondant à tout l'exercice.

Exercice 1004 - Niveau 2

- Ecrire un sous-programme qui, à partir d'un tableau de réels passé en paramètre, saisit chaque élément de ce tableau
- Ecrire un sous-programme qui affiche tous les éléments d'un tableau passé en paramètre
- Ecrire le sous-programme qui trie par ordre croissant les éléments du tableau (**méthode de tri à bulle**)
- Ecrire un autre sous-programme qui trie avec la **méthode du tri par insertion**
- Ecrire le programme principal qui met en place tous ces sous-programmes. Le sous-programme d'affichage sera appelé une fois après la saisie et une fois après le tri. Pour le tri, le sous-programme appelé sera celui que l'utilisateur choisira.

La taille du tableau est 20.

Exercice 1005 - Niveau 2

- Ecrire un sous-programme qui, à partir d'un tableau d'entiers passé en paramètre, saisit chaque élément de ce tableau
- Ecrire un sous-programme qui affiche tous les éléments d'un tableau passé en paramètre
- Ecrire le sous-programme qui trie par ordre croissant les éléments du tableau (**méthode de tri par insertion**)
- Ecrire sous-programme qui recherche un entier dans la table. L'entier recherché et la table sont passés en paramètre. Un message indiquera si l'élément est trouvé ou non
- Ecrire le programme principal qui met en place tous ces sous-programmes.

La taille du tableau est 10.

Exercice 1006 - Niveau 2

- Ecrire le sous-programme qui saisit toutes les valeurs d'un tableau de 20 réels. Le tableau est passé en paramètre.
- Ecrire le sous-programme qui trie un tableau de réel passé en paramètre (**méthode de tri par insertion**)
- Faire la programme qui :
 - appelle le sous-programme de saisie
 - appelle le sous-programme de tri
 - saisit un réel et appelle le sous-programme de recherche

Exercice 1007 - Niveau 2

- Dessiner la représentation schématique d'un tableau défini comme suit : `char tab[3][4][2]`
- Ecrire un sous-programme qui saisit un caractère (alphabétique, numérique uniquement) et qui rend ce caractère au programme appelant
Ne pas mettre de valeur en dur pour le blindage de la saisie !
- Ecrire un sous-programme qui affiche un tableau passé en paramètre
- Ecrire un sous-programme qui compte le nombre d'occurrences d'un caractère saisi (= le nombre de fois où apparaît cet élément dans le tableau). Le résultat sera rendu au programme appelant et le caractère saisi passera en paramètre par adresse pour être rendu à l'appelant lui aussi.
- Ecrire le programme qui :
 - saisit les éléments du tableau
 - affiche tout le tableau
 - appelle du sous-programme de recherche d'occurrences
 - affiche le caractère saisi ainsi que son nombre d'occurrences dans le tableau

Exercice 1008 - Niveau 3

- Ecrire le sous-programme qui saisit un tableau à deux dimensions de caractères :
4 lignes et 5 colonnes
cf **matrice saisie** dans l'exemple
- Ecrire le sous-programme qui transforme les voyelles minuscules en majuscules :
cf **matrice transformée** dans l'exemple
Dans ce sous-programme, une variable décalage sera mise en place. Ne pas mettre de valeur en dur !
- Ecrire le sous-programme qui affiche le tableau de caractères
- Ecrire le sous-programme qui inverse les valeurs du tableau : la valeur qui se trouve à la première place se trouve à la dernière et inversement, la valeur qui se trouve à l'avant-dernière place se positionne à la deuxième place etc. :
cf **matrice inversée** dans l'exemple
- Ecrire le programme qui met en place ces fonctions. L'affichage interviendra après chaque étape comme présenté dans l'exemple ci-dessous.

Exemple :

Matrice saisie :

F	#	G	D	%
\$	I	k	*	u
I	A	&	=	p
T	Z	b	e	e

Matrice transformée :

F	#	G	D	%
\$	I	k	*	U
I	A	&	=	P
T	Z	B	E	E

Matrice inversée :

E	E	B	Z	T
P	=	&	A	I
U	*	K	I	\$
%	D	G	#	F

11. CHAINES DE CARACTÈRES

Exercice 1101 - Niveau 1

Ecrire l'algorithme et le programme qui :

- saisit deux chaînes de caractères
- compte leur nombre de caractères respectifs sans utiliser `strlen()`
- affiche les chaînes et le nombre de caractères de chaque chaîne qui a été calculé

Remarque :

- La première chaîne de caractères ne doit pas accepter les espaces. Elle n'est constituée que d'un seul mot.
- La deuxième chaîne sera une phrase constituée de plusieurs mots séparés par un espace chacun.

Exercice 1102 - Niveau 1

Ecrire l'algorithme et le programme qui :

- saisit deux chaînes
- compare alphabétiquement ces deux chaînes
 - Version 1 : ne pas utiliser `strcmp()`
 - Version 2 : utiliser `strcmp()`
- affiche le résultat de la comparaison : par exemple : « les chaînes sont identiques » ou « la 1^{er} chaîne est avant la 2^{ème} chaîne dans l'ordre alphabétique » ou le contraire.

Exercice 1103 - Niveau 2

- Ecrire le sous-programme qui saisit une chaîne et la rend à l'appelant
- Ecrire l'algorithme et le sous-programme qui compte le nombre de caractères numériques, le nombre de minuscules dans une chaîne de caractère passée en paramètre. Les résultats seront rendus au programme appelant. Vous n'utiliserez au nombre en 'dur' !!!
- Ecrire l'algorithme et le sous-programme qui compte le nombre de mots (=ensembles de caractères séparés par un espace) stockés dans la chaîne passée en paramètre. Rendre ce résultat à l'appelant
- Ecrire le programme qui :
 - appelle les sous-programmes
 - affiche la chaîne ainsi que les résultats obtenus par les différents sous-programmes

Exercice 1104 - Niveau 2

Ecrire l'algorithme et le code d'un programme et des sous-programmes suivants.

Ecrire un sous-programme appelé `menu()` dont les fonctionnalités sont les suivantes :

- saisir une chaîne de caractères, l'afficher à l'écran et la rendre à l'appelant
- convertir les caractères de la chaîne saisie en majuscule (vous ne modifierez que les caractères minuscules mais pas les autres et utiliser la table ASCII sans valeur en dur !). La chaîne sera affichée à l'écran

- compter le nombre de voyelles (minuscules et majuscules) dans une chaîne
- convertir les caractères majuscule de la chaîne saisie en minuscule (utiliser la table ASCII sans valeur en dur !)
- comparer 2 chaînes à saisir (utilisez strcmp) et afficher les messages qui correspondent
- concaténer 2 chaînes dans la première (attention aux débordements) et afficher le résultat obtenu
- crypter une chaîne de caractères : le cryptage s'applique par un décalage circulaire sur le même alphabet.

La valeur du décalage est saisie par l'utilisateur

Ex : valeur saisie 1 :

Le cryptage est le suivant

Caractère	Caractère crypté
A	B
M	N
Z	A

Prendre la lettre suivante
(décalage de 1)

- décrypter une chaîne de caractères : le cryptage s'applique par un décalage circulaire inverse au précédent sur le même alphabet.

La valeur du décalage est saisie par l'utilisateur.

Ex : valeur saisie 3 :

Le décryptage est le suivant

Caractère	Caractère décrypté
A	X
M	J
Z	W

Prendre la lettre qui se situe 3
places avant (décalage inverse de 3)

- quitter l'application

Ecrire un sous-programme pour chaque fonctionnalité du menu décrit plus haut.

Vous blinderez votre programme pour garantir que vous travaillez sur des chaînes non vides et redemanderez la saisie dans le cas contraire.

Ecrire le main(). Le programme principal appellera le menu et recommencera l'exécution tant que l'utilisateur n'aura pas demandé explicitement l'arrêt.

Exercice 1105 - Niveau 2

- Ecrire le sous-programme qui remplit un tableau de chaînes de caractères ayant 10 places. Ce tableau est passé en paramètre
- Ecrire un sous-programme qui recherche un mot dans le tableau de chaînes de caractères et qui indique s'il est présent ou non.
Le tableau et le mot recherché sont passés en paramètre.
- Ecrire le programme principal qui met en place ces sous-programmes ainsi que la saisie du mot à retrouver.

- Pour tout l'exercice écrire l'algorithme correspondant

Exercice 1106 - Niveau 2

Ecrire l'algorithme et le code C de programme et des sous-programmes suivants :

Les différentes fonctionnalités seront proposées par un menu. Ce menu est un sous-programme qui retournera l'option choisie.

Les fonctionnalités sont :

- saisir et afficher une phrase contenant espaces et ponctuations.
La phrase saisie aura au plus 50 caractères utiles.
- calculer le nombre de voyelles d'une phrase reçue en paramètre. Ce nombre sera affiché puis rendu au programme appelant.
- concaténer deux chaînes de caractères. La première chaîne est passée en paramètre et la deuxième est construite comme miroir de la première.

Les deux phrases sont séparées par un séparateur « : »

exemple :

phrase passée en paramètre :	« il fait beau »
résultat :	« il fait beau : uaeb tiaf li »

La phrase résultat est rendue au programme appelant.

- Calculer le nombre de consonnes de la phrase passée en paramètre. Le résultat sera rendu au programme appelant

Chaque fonctionnalité proposée sera gérée dans un sous-programme.

Vous blinderez votre programme pour garantir que vous travaillez sur des chaînes non vides et redemanderez la saisie dans le cas contraire.

Le programme principal mettra en jeu l'ensemble des sous-programmes écrits. Il recommencera l'exécution tant que l'utilisateur n'aura pas demandé explicitement l'arrêt.

Exercice 1107 - Niveau 3

- Ecrire l'algorithme et le sous-programme qui construit une chaîne de caractère avec les deux premiers caractères suivis des deux derniers caractères d'une chaîne passée en paramètre. La chaîne construite doit être renvoyée au programme appelant.

Si cette chaîne passée en paramètre a moins de quatre caractères, un message indique que la saisie n'est pas valable et on rend une chaîne vide au programme appelant.

Ex :

Cas 1

chaîne passée en paramètre : « ordinateur »

chaîne rendue : « orur »

Cas 2 :

chaîne passée en paramètre : « ode »

chaîne rendue : « »

message affiché : « chaîne non valable »

- Ecrire l'algorithme et le sous-programme qui saisit une chaîne et vérifie si cette chaîne existe dans une première chaîne reçue en paramètre. Rendre un indicateur qui fera office de booléen.

Ex :

Cas1 :

Chaine 1 : babar
Chaine 2 : bar
Résultat rendu : 1

Cas 2 :
Chaine 1 : arbre
Chaine 2 table
Résultat rendu : 0

- Ecrire le programme principal mettant en jeu les sous-programmes après la saisie d'une chaîne. Les résultats reçus seront affichés.
Ce programme principal se terminera qu'à la demande explicite de l'utilisateur.

Exercice 1108 - Niveau 3

- Ecrire le sous-programme qui saisit un verbe du 1^{er} groupe de conjugaison (test de la terminaison « -er »). Ce verbe est rendu au programme appelant.
- Ecrire l'algorithme et le sous-programme qui affiche la conjugaison au présent du verbe passé en paramètre.
Ex : verbe passé en paramètre : chanter
Affichage :
 - Je chante
 - Tu chantes
 - Il/Elle chante
 - Nous chantons
 - Vous chantez
 - Ils chantent
- Ecrire le programme principal qui met en place les deux sous-programmes

Quelques exemples de fonctions de la librairie <string.h> : **ANNEXE**

Table ASCII : **ANNEXE**

12. GÉNÉRATION DE NOMBRES ALÉATOIRES

Exercice 1201 - Niveau 1

- Ecrire un sous-programme qui génère aléatoirement un nombre. Ce nombre est rendu au programme appelant
 - Version 1 : génération aléatoire sans restriction d'intervalle
 - Version 2 : génération aléatoire dans [1 ;200]
- Ecrire l'algorithme et le programme principal qui :
 - appelle le sous-programme générant un nombre aléatoire
 - demande à l'utilisateur de retrouver le nombre généré.
 - affiche en fin de programme le nombre d'essais réalisés pour découvrir la valeur du nombre aléatoire

Exercice 1202 - Niveau 1

- Ecrire le sous-programme qui remplit un tableau d'entiers aléatoires (20 places). Dans le tableau toutes les valeurs n'apparaîtront qu'une seule fois, aucun doublon n'est permis.
- Ecrire le sous-programme qui compte
 - le nombre de multiples de 10
 - le nombre d'entiers inférieurs à 100
 - le pourcentage de multiples de 3 se trouvant dans le tableau (passé en paramètre).Ces nombres seront rendus à l'appelant.
- Ecrire un programme principal qui :
 - appelle le sous-programme de remplissage de tableau
 - appelle le sous-programme de traitement du tableau
 - affiche les résultats obtenus
- Vous écrierez l'algorithme de toutes les fonctionnalités

Exercice 1203 - Niveau 2

Ecrire l'algorithme et le programme affichant un menu proposant différents cas de génération de nombres aléatoires.

1. affichage d'un nombre aléatoire entier dans la plage de valeur globale du générateur aléatoire
2. affichage d'un nombre aléatoire entier compris entre 0 et une valeur « seuil haut » saisie par l'utilisateur
3. affichage d'un nombre aléatoire entier compris entre la valeur « seuil bas » et « seuil haut » saisies par l'utilisateur
4. affichage de **n** nombres aléatoires (n saisi par l'utilisateur) entre les seuils bas et haut saisis par l'utilisateur
5. affichage de **n** nombres aléatoires flottants à deux décimales entre 0 et 1 (bornes comprises). n saisi par l'utilisateur.
6. affichage de **n** nombres aléatoires flottants à trois décimales entre -50 et 90 (bornes comprises). n saisi par l'utilisateur.

Chaque fonctionnalité sera codée dans un sous-programme.

Exercice 1204 - Niveau 2

- Ecrire le sous-programme remplit un tableau de 15 lignes et 20 colonnes avec des caractères minuscules (**code ASCII obtenu aléatoirement**). Ce tableau est rendu au programme appelant.
- Ecrire le sous-programme qui compte le nombre d'occurrences de chaque lettre apparaissant dans le tableau. Ces résultats seront stockés dans une matrice de compteurs de 26 places. Les tableaux sont passés en paramètre.
- Ecrire le programme principal qui :
 - appelle les sous-programmes précédents
 - affiche les résultats obtenus en précisant bien la lettre correspondante avant son compteur
- Vous écrirez l'algorithme de toutes les fonctionnalités

Table ASCII : **ANNEXE**

TP : Image, histogramme et binarisation

TP : niveau 2

L'objectif de la suite de ce TP est de manipuler le tableau de valeurs correspondant à chaque point (appelé PIXEL) d'une image. Vous ne manipulerez pas une vraie image ; n'ayant pas d'outil de visualisation des images en mode console, vous ne pourrez manipuler que des nombres sans visualiser le résultat de vos traitements.

Exercice 1205

Ecrire un programme remplissant aléatoirement un tableau à deux dimensions (matrice) de 10 lignes et 20 colonnes avec des valeurs comprises entre 0 et 255 (bornes comprises).

Afficher cette matrice à l'écran en respectant l'aspect rectangulaire de la matrice.

Exercice 1206

L'histogramme d'une image étudie la répartition statistique de chaque valeur de niveau de gris dans une image.

Son principe consiste, dans un tableau histogramme de 256 cases de type entier, à compter combien l'image contient de pixels de niveau de gris 0 et à stocker cette valeur dans la case d'indice 0 du tableau histogramme, puis combien l'image contient de pixels de valeur 1 à ranger dans la case d'indice 1... ainsi de suite jusqu'à compter le nombre de pixels de valeur 255.

Attention !! Il y a une méthode de calcul rapide et une méthode très lente. Réfléchissez et ayez l'algorithme optimal !

Cet exercice est à faire en complétant le programme précédent.

Affichez cet histogramme de manière lisible à l'écran.

Exercice 1207

La binarisation d'une image consiste en **une image secondaire créée, à partir de l'originale** afin de ne pas modifier cette dernière, ainsi :

- mettre à 0 tous les pixels de l'image originale inférieurs à une valeur seuil saisie par l'utilisateur (seuil entre 100 et 200)
- mettre à 255 tous les pixels de l'image originale supérieurs ou égaux à cette valeur seuil.

Compléter à nouveau le programme précédent pour y inclure la binarisation et affichez la valeur des pixels de l'image binarisée en respectant l'aspect de l'image.

13. STRUCTURES

Exercice 1301 - Niveau 1

- Ecrire la définition du type **t_film** contenant :
 - le ~~titre~~ numéro du film (entier)
 - son année de réalisation (entier)
 - la première lettre du nom du réalisateur (caractère)
 - la durée en minutes (entier)
 - un booléen indiquant si vous l'avez vu ou non (entier court ou encore mieux caractère d'un point de vue taille mémoire)
 - une note sur 10 (entier)
- Ecrire un sous-programme chargé de remplir une instance de cette structure ~~dont l'adresse est reçue en paramètre~~ et retourne la structure. Pensez au blindage des saisies : par exemple, l'année entre 1930 et 2022, la durée entre 20 et 300 minutes etc.
- Ecrire un sous-programme qui affiche les données d'une structure passée en paramètre
- Ecrire un programme principal qui met en place ces deux sous-programmes

Exercice 1302 - Niveau 1

- Définir une structure contenant les champs suivants :
 - ~~Une chaîne de 20 caractères~~ Un caractère
 - Un tableau de 15 entiers
- Ecrire un sous-programme qui remplit les champs de la structure ~~passée en paramètre par adresse~~ et retourne la structure
- Ecrire un sous-programme qui affiche les champs de la structure passée en paramètre
- Ecrire le programme qui met en jeu les deux sous-programmes précédents

Exercice 1303 - Niveau 2

Soit une structure **etudiant** contenant les champs :

- Le numéro étudiant (entier)
 - ~~Le nom~~
 - ~~Le prénom~~
 - ~~L'adresse (uniquement le nom de la ville)~~ Le code postal (entier)
 - L'année de naissance (entier)
 - Un tableau de 5 notes (tableau de 5 réels)
-
- Définir la structure **t_etudiant**.
 - Ecrire un sous-programme chargé de remplir une instance de cette structure ~~dont l'adresse est passée par adresse~~ et de retourner cette structure
 - Ecrire un sous-programme qui remplit un tableau de 20 structures t_etudiant passé en paramètre. Vous appellerez le sous-programme précédent pour chacune des structures. Aucun doublon de numéro étudiant n'est permis
 - Ecrire un sous-programme qui affiche les données d'une structure passée en paramètre
 - Ecrire un sous-programme qui affiche un tableau de 20 structures t_etudiant passé en paramètre. Vous appellerez le sous-programme précédent pour chacune des structures
 - Ecrire un programme principal qui :
 - Déclare un tableau de 20 structures t_etudiant, pour le nombre d'élément utilisez une constante dans la déclaration et les boucles de parcours ! Cette constante pourra évoluer au cours de la mise en place de votre exercice ☺
 - Appelle le sous-programme qui remplit tous les éléments du tableau.
 - Appelle le sous-programme qui affiche tous les éléments du tableau
 - Faire un schéma mémoire du tableau de structures

Exercice 1304 - Niveau 2

Cet exercice est la suite du précédent. Les sous-programmes et le tableau de structures sont conservés autant que possible.

- Ajouter les sous-programmes suivants :
 - Initialisation du tableau de structures : tous les numéros étudiant sont à -1
 - Suppression d'une structure identifiée par son numéro étudiant. Donc affectation du numéro étudiant à -1
 - Modification d'une structure identifiée par son numéro étudiant. Le champ adresse sera modifié.
 - Menu proposant les fonctionnalités de :
 - Ajout (une seule structure à chaque appel). Faites attention que le tableau ait au moins une place vide !
 - Suppression (une seule structure identifiée par son numéro d'étudiant)
 - Affichage de toutes les structures existantes
 - Modification d'une structure identifiée par son numéro d'étudiant
- Modifier le programme principal :
 - Ecrire une boucle de traitement dont on ne sortira qu'à la demande de l'utilisateur.
 - Appel du menu
 - Traitement de la fonctionnalité demandée
 - Adapter la valeur de la constante qui détermine le nombre d'éléments du tableau, pour cet exercice...
- Dessiner un graphe d'appel

Exercice 1305 - Niveau 3

Soit une structure *date*, dont l'alias de type est nommé **t_date**, contenant les 3 champs entiers suivants:

- Le jour
- Le mois
- L'année

Soit une structure *article*, dont l'alias de type est nommé **t_article**, contenant les champs suivants :

- Le numéro de désignation (entier)
- Le numéro de catégorie (entier)
- La date de péremption (de type **t_date** défini plus bas)
- La quantité en stock (entier)
- Le prix de vente (réel)

- Définir les structures **t_article** et **t_date**.
- Ecrire un sous-programme chargé de remplir une instance de structure **t_article** ~~dont l'adresse est passée par adresse~~ et de retourner cette structure
- Ecrire un sous-programme qui remplit un tableau de 20 structures **t_article** passé en paramètre en appelant le sous-programme précédent
- Ecrire un sous-programme qui affiche une structure **t_article** passée en paramètre
- Ecrire un sous-programme qui affiche un tableau de 20 structures **t_article** passé en paramètre en appelant le sous-programme précédent
- Ecrire le sous-programme qui reçoit une catégorie en paramètre et qui affiche les articles de la catégorie donnée
- Ecrire un sous-programme qui compte le nombre d'articles dont la date de péremption a dépassé une date passée en paramètre
- Ecrire un programme principal qui :
 - Déclare un tableau de 20 structures **t_article**. Pour le nombre d'éléments, utilisez une constante dans la déclaration et les boucles de parcours ! Cette constante pourra évoluer au cours de la mise en place de votre exercice ☺
 - Appelle le sous-programme qui remplit tous les éléments du tableau
 - Appelle le sous-programme qui affiche tous les éléments du tableau
 - Saisit une catégorie d'articles et appelle le sous-programme qui affiche les articles de cette catégorie
 - Saisit une date complète et appelle le sous-programme qui compte le nombre d'articles ayant dépassé cette date

14. FICHIERS : ASCII ET BINAIRES

Exercice 1401 - Niveau 1

Ecrire un programme C qui permet de stocker dans un fichier ASCII 15 entiers saisis et 20 entiers aléatoires. Dans le fichier, les nombres seront chacun sur une ligne.

Vous ouvrirez ensuite ce fichier à l'aide du bloc-notes pour en vérifier le contenu.

Exercice 1402 - Niveau 1

Ecrire un programme C qui permet de lire tous les entiers d'un fichier ASCII. Ce fichier est créé via le bloc-notes (ou c'est celui créé dans l'exercice précédent). Chaque entier se trouve sur une ligne.

Vous afficherez les nombres lus pour vérifier l'efficacité de votre programme.

Exercice 1403 - Niveau 2

Ecrire l'algorithme et le programme qui :

- lit des entiers dans un fichier texte (ASCII) qui sera créé auparavant. Le nombre d'entiers à lire n'est pas connu.
- ajoute 10 à chaque entier
- écrit les résultats obtenus ainsi :
 - les multiples de 5 dans le fichier 'mult5.txt'
 - les multiples de 7 dans le fichier 'mult7.txt'.

Dans les deux fichiers créés, chaque nombre sera sur une ligne.

Vous ouvrirez ensuite le fichier source et les fichiers destination à l'aide du bloc-notes pour en vérifier le contenu.

Exercice 1404 - Niveau 2

Ecrire le programme qui :

- saisit 10 chaînes de caractères
- stocke ces chaînes dans un fichier nommé *stockage1.txt*. Chaque chaîne saisie se trouve sur une ligne du fichier

Exercice 1405 - Niveau 3

- Soit la structure **t_film** contenant l'identifiant, le titre, le réalisateur, l'année, le genre... Définir cette structure **t_film**.
- Ecrire un sous-programme qui sauvegarde les éléments d'un tableau de structures dans un fichier texte. Le nom du fichier est reçu en paramètre.
Chaque attribut se trouve sur une ligne.
- Ecrire un sous-programme de chargement de données. Il reçoit en paramètres le tableau de films et le nom d'un fichier. Ce sous-programme lira le fichier. Les données lues seront stockées dans le tableau de structures passé en paramètre.
Chaque ligne correspond à une donnée de structure **t_film**.
Vérifier que le tableau est suffisamment grand pour accueillir toutes les données
- Ecrire un sous-programme qui affiche tous les champs des éléments du tableau de films passé en paramètre
- Ecrire le programme principal qui :
 - définit un tableau de 10 structures.

Utilisez une constante pour le nombre de structures et les parcours de tableau à venir !

- initialise le tableau : l'identifiant du film a une valeur -1
- saisit le nom du fichier de chargement (de type fichier texte)
- charge les films du fichier dans le tableau
- affiche les films du tableau
- ajoute deux nouveaux films dans le tableau
- saisit le nom du fichier de sauvegarde (de type fichier texte)
- sauvegarde tous les films dans le fichier de sauvegarde

Remarque :

Les sous-programmes écrits précédemment seront tous utilisés

Dans le fichier initial, il n'y aura pas plus de 8 films.

Exercice 1406 - Niveau 3

- Définir la structure date :
 - jour : entier
 - mois : entier
 - année : entier
- Définir la structure livre suivante :
 - identifiant : entier
 - titre : chaîne de caractères
 - auteurs : tableau de chaînes de caractères (5 auteurs possibles dans le tableau)
 - thème : tableau de chaînes de caractères (5 thèmes possible dans le tableau)
 - prix HT : réel
 - parution : structure date
- Ecrire un sous-programme qui saisit une structure livre dont l'adresse est en paramètre
- Ecrire un sous-programme qui remplit les éléments d'un tableau de livres dont le tableau est passé en paramètre
- Ecrire un sous-programme d'affichage d'un livre dont la structure est en paramètre
- Ecrire un sous-programme qui affiche l'ensemble de livres dont le tableau est passé en paramètre
- Ecrire un sous-programme qui sauvegarde dans un fichier binaire l'ensemble des livres mais aussi dans un fichier texte. Le tableau est passé en paramètre
- Ecrire un sous-programme qui charge dans un tableau les livres lus d'un fichier binaire
- Ecrire un sous-programme menu qui propose toutes les fonctionnalités précédentes et rend le choix au programme appelant
- Ecrire un programme principal qui met en place tous ces sous-programmes en les appelant après la saisie du choix de l'utilisateur. Ce programme s'exécutera tant que l'utilisateur ne demande pas de quitter.

Le tableau de livres aura 10 places.

15. ALLOCATION DYNAMIQUE DE MEMOIRE

Exercice 1501 - Niveau 1

- Ecrire un sous-programme qui saisit deux phrases dans un contenu fixe chacune. Ce sous-programme alloue un espace mémoire ajusté pour y stocker les 2 phrases saisies. La phrase ajustée sera rendue au programme appelant
- Ecrire un programme principal qui :
 - appelle le sous-programme
 - affiche la phrase dynamique rendue.

Exercice 1502 - Niveau 2

Ecrire un programme qui :

- déclare un tableau de 10 chaînes dynamiques
- remplit ce tableau
- trie alphabétiquement le tableau (faites uniquement les changements de pointeurs)
- copie tous les mots dans une chaîne dynamique nommée **unePhrase**. Dans **unePhrase**, chaque mot sera séparé des autres par un espace. Chaque fois qu'un élément est copié dans **unePhrase**, il sera supprimé du tableau.
- Afficher **unePhrase** à l'écran

Exercice 1503 - Niveau 1

- Ecrire un sous-programme qui :
 - saisit la taille d'un tableau de réels (donc tableau à une dimension)
 - alloue le tableau dynamiquement
 - saisit tous les éléments du tableau
 - rend le tableau au programme (donc son adresse) à l'appelant
- Ecrire un sous-programme d'affichage de ce tableau
- Ecrire un sous-programme de tri de ce tableau
- Ecrire un programme principal qui met en place ces sous-programmes

PS :

Pensez à bien choisir vos paramètres et vos données qui seront retournées par **return** !

Exercice 1504 - Niveau 2

- Dans un sous-programme, lire un entier dans un fichier *donnee.txt*. Cet entier est la taille du tableau d'entiers à allouer dynamiquement. Rendre au programme principal cet entier.
- Dans un sous-programme, écrire dans un fichier *mult3.txt* les multiples de 3 et dans un autre fichier *autres.txt* les autres nombres issus d'un tableau dynamique passé en paramètre.
- Dans le programme principal :
 - Créer un tableau dynamique à partir de l'entier saisi dans le 1^{er} sous-programme
 - Remplir le tableau avec des valeurs aléatoires comprises entre -25 et 150
 - Sauvegarder les valeurs de ce tableau dans un fichier *resultat.txt*.
 - Sauvegarder les multiples de 3 en appelant le 2^{ème} sous-programme
 - Libérer toutes les données dynamiques

Exercice 1505 - Niveau 2

- Ecrire un sous-programme permettant l'allocation dynamique d'une matrice d'entiers après avoir demandé la valeur des 2 dimensions à l'utilisateur (la saisie des dimensions est à blinder)
Toutes les données seront rendues à l'appelant. Justifiez si vous utilisez l'instruction `return` ou le passage par adresse.
- Ecrire un sous-programme permettant le remplissage **aléatoire** de la matrice précédente avec des valeurs entre 0 et 255 (bornes comprises).
- Ecrire un sous-programme permettant l'affichage de la matrice d'entiers.
- Ecrire le programme principal tel qu'il joue son rôle de chef d'orchestre.
- Dessiner le graphe d'appel de votre programme en faisant apparaître le fonctionnement des tubes et des mécanismes de retour.
- Structurer votre projet avec la bibliothèque personnelle au format `.h`

PS :

Dans tout cet exercice, réfléchissez à prendre les données adaptées en paramètres.

Exercice 1506 - Niveau 2

- Définir une structure de maillon dinosaure :
 - Nom (chaîne de 15 caractères utiles)
 - Famille (chaîne de 10 caractères utiles)
 - Date de disparition (entier)
 - Type : Carnivore/herbivore (caractère)
- Ecrire un sous-programme appelé *creer()* qui :
 - alloue de la place mémoire à une structure
 - remplit tous les champs
 - rend l'adresse obtenue.
- Ecrire un sous-programme appelé *afficher()* qui affiche une structure dont l'adresse est passée en paramètre.
- Ecrire un sous-programme appelé *toutAfficher()* qui affiche toutes les données du tableau passé en paramètre. Pour cela vous utiliserez le sous-programme précédent
- Ecrire un sous-programme appelé *rechercher()* qui recherche et affiche tous les éléments dans le tableau de pointeurs de structures passé en paramètre selon 2 critères (passés aussi en paramètre) :
 - Date de disparition
 - Type de dinosaurePour cela vous utiliserez le sous-programme *afficher()*.
- Ecrire un programme principal qui :
 - déclare un tableau de pointeurs de structures (20 places). Utilisez une constante qui sera ajuster au fur et à mesure de l'écriture de votre exercice !
 - appelle le sous-programme *creer()* pour chaque élément du tableau
 - affiche tous les éléments du tableau avec le sous-programme *afficher()*
 - recherche les éléments du tableau selon une date et un type. Ces données seront saisies avant l'appel du sous-programme
 - libérer toutes les données dynamiques

TP : niveau 2

- Définir le type correspondant à la structure d'un rendez-vous :
 - un champ date au format jour/mois/année : type t_date à définir
 - un champ heure début : type entier
 - un champ heure fin : type entier
 - un champ objet (=motif du rendez-vous) : type chaine dynamique
 - un booléen rappel : type entier

Vous complétez cette description par tous les champs que vous jugeriez nécessaire d'ajouter.

- Définir la structure de données la plus adaptée à vos choix de stockage : tableau de rendez-vous ou tableau de pointeurs sur rendez-vous en justifiant votre choix.
- Ecrire les sous-programmes suivant pour structurer notre TP :
 - créer un RDV (attention à la place mémoire disponible)
 - ajouter un RDV dans le tableau
 - afficher tous les RDV
 - rechercher un RDV correspondant à un critère donné
 - afficher un RDV correspondant à un critère donné
 - supprimer un rendez-vous correspondant à un critère donné
 - supprimer tous les RDV

Réfléchir pour chaque sous-programme aux paramètres à avoir ainsi qu'aux données à rendre par return éventuellement.
- Afin de ne pas être obligé de tout ressaisir à chaque fois et de ne pas perdre les rendez-vous créés, coder la sauvegarde des données dans un fichier.
Créer les sous-programmes pour :
 - Sauvegarder les RDV dans un fichier (vous choisirez texte ou binaire)
 - Charger les RDV à partir d'un fichier de sauvegarde

- Au lancement du programme :
 1. Le sous-programme de chargement sera lancé systématiquement afin de monter en mémoire tous les RDV contenus dans le fichier. (Vous pourrez discuter avec votre chargé de TD-TP de la pertinence de cette méthode).
 2. Saisir un jour considéré comme la date d'aujourd'hui.
 3. Vous afficherez à l'écran tous les rendez-vous du jour (c'est-à-dire ayant la même valeur que la date saisie en précisant s'ils ont ou non un rappel à faire).

16. RECURSIVITE

Exercice 1601 - Niveau 1

- Ecrire un sous-programme récursif qui saisit un tableau de 20 réels.
- Ecrire un sous-programme récursif qui affiche ce tableau.
- Ecrire le programme qui déclare le tableau et met en place les deux sous-programmes.

PS :

Pour chaque point vous écrirez un algorithme

Exercice 1602 - Niveau 2

Ecrire un sous-programme récursif qui calcule la somme des valeurs du tableau dynamique d'entiers reçu en paramètre.

Ecrire le programme principal qui :

- saisit la taille du tableau d'entiers, l'alloue
- remplit le tableau
- appelle le sous-programme et affiche son résultat

PS :

N'oubliez pas d'écrire un algorithme

Exercice 1603 - Niveau 2

Ecrire un sous-programme récursif permettant de calculer la longueur d'une chaîne de caractères (comme le ferait la fonction STRLEN mais sans l'utiliser bien sûr !).

Ecrire un programme qui :

- saisit la chaîne
- appelle le sous-programme précédent en passant la chaîne en paramètre
- affiche le résultat du sous-programme

PS :

N'oubliez pas d'écrire un algorithme

Exercice 1604 - Niveau 2

Ecrire un sous-programme récursif qui calcule le résultat de la fonction de récurrence suivante :

$$U_0 = 7$$

$$U_n = 3 \cdot U_{n-1} + 5/2$$

Mettre en place l'appel au sous-programme dans un programme principal.

PS :

N'oubliez pas d'écrire un algorithme

Exercice 1605 - Niveau 2

Ecrire un sous-programme récursif qui calcule la puissance d'un nombre. Ce sous-programme rend le résultat à l'appelant.

Ecrire le programme principal qui :

- saisit le nombre et la puissance
- appelle le sous-programme et passe en paramètre le nombre et la puissance
- affiche le résultat obtenu.

PS :

N'oubliez pas d'écrire un algorithme

Exercice 1606 - Niveau 3

Ecrire le code des sous-programmes permettant de trier un tableau de n flottants (n est un entier saisi). Dans tout l'exercice le tableau sera passé en paramètre. Les sous-programmes à écrire sont les suivants :

- Le sous-programme « saisie » récursif permettra de saisir n flottants et de les stocker dans le tableau passé en paramètre
- Le sous-programme « affiche », récursif également, permettra d'afficher les réels du tableau passé en paramètre. Vous afficherez les réels dans l'ordre (du 1er au dernier). Qu'est ce qui changerait si on affichait les éléments dans l'ordre inverse (du dernier au 1er).
- Le sous-programme « Tri » récursif permettra de trier le tableau de flottants dans un ordre croissant. Le tri utilisé est le **tri par fusion**.
- Le programme principal qui met en place tous ces sous-programmes.

Exercice 1607 - Niveau 3

- Ecrire le sous-programme récursif qui saisit un tableau de 30 caractères (alphabétique, numérique, ponctuation ou autres). Le tableau sera passé en paramètre
- Ecrire le sous-programme récursif qui affiche un tableau passé en paramètre
- Ecrire le sous-programme récursif qui trie un tableau dans l'ordre croissant selon la méthode du **tri rapide (quick sort)**. Le tableau est passé en paramètre.
- Ecrire un programme principal qui met en place ces sous-programmes.

Exercice 1608 - Niveau 3

- Ecrire un sous-programme récursif qui saisit un tableau de 15 entiers, passé en paramètre.
- Ecrire un sous-programme récursif qui trie un tableau passé en paramètre par ordre croissant
- Ecrire un sous-programme récursif qui recherche un entier saisi par l'utilisateur dans un tableau passé en paramètre. Le résultat sera rendu à l'appelant.
- Ecrire un programme principal qui met en place ces sous-programmes.

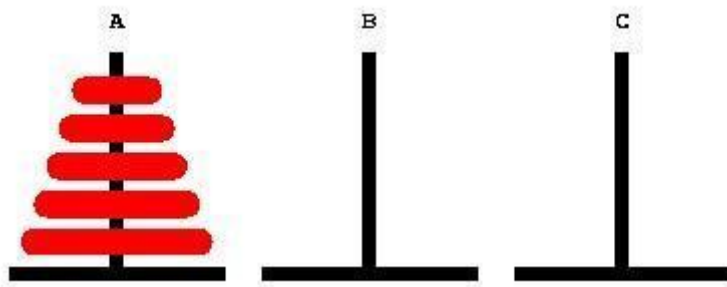
Exercice 1609 - Niveau 3

Les **tours de Hanoï** sont un jeu de réflexion imaginé par le mathématicien français Édouard Lucas, et consistant à déplacer des disques de diamètres différents d'une tour de « départ » à une tour d'« arrivée » en passant par une tour « intermédiaire », et ceci en un minimum de coups, tout en respectant les règles suivantes :

- on ne peut déplacer plus d'un disque à la fois ;
- on ne peut placer un disque que sur un autre disque plus grand que lui ou sur un emplacement vide.

Considérons trois tours A, B et C avec au départ A, tour de 5 disques cf dessin.

Ecrire un sous-programme récursif qui déplace la tour A emplacement vers l'emplacement B.



17. LISTES CHAINEES

Exercice 1701 - Niveau 1

- Ecrire une structure de maillon composé d'un réel et d'un pointeur vers le maillon suivant.
- Ecrire un sous-programme qui crée un maillon, saisit le réel et rend l'adresse du maillon créé.
- Ecrire un sous-programme qui affiche un maillon (uniquement la valeur réel)
- Ecrire un sous-programme récursif qui affiche tous les réels de la liste dans l'ordre (du premier au dernier)
- Ecrire un programme qui insère 10 maillons dans une liste déclarée (**chainage simple**). La création de cette chaîne se fera dans une boucle. Ce programme met ensuite en place les sous-programmes décrits ci-dessus.

En fin d'exécution, libérer les données dynamiques.

PS :

Pensez aux algorithmes !

N'oubliez pas de faire des dessins de mémoire. C'est plus simple ☺.

Exercice 1702 - Niveau 1

- Ecrire une structure de maillon voiture qui contient :
 - un numéro d'immatriculation (chaîne)
 - une marque (chaîne)
 - un modèle(chaîne)
 - une vitesse maximum(entier)
 - un type de carburant (caractère).

Ce maillon aura un champ vers le maillon suivant.

Vous pouvez ajouter une structure date de type `t_date` à définir pour la date de 1^{ère} mise en service

- Ecrire un sous-programme qui crée un maillon, saisit les données et rend l'adresse du maillon créé.
- Ecrire un sous-programme récursif qui affiche toutes les voitures de la liste.
- Ecrire un sous-programme récursif qui affiche les immatriculations des maillons à partir d'une marque saisie, passée en paramètre.
- Ecrire un sous-programme qui reçoit une immatriculation en paramètre et supprime cette voiture de la liste si elle existe

- Ecrire un programme principal qui :
 - insère autant de maillons que l'utilisateur le souhaite, dans une liste déclarée.
 - affiche tous les données des maillons d'une marque saisie
 - saisit une immatriculation et appelle le sous-programme. Le résultat sera affiché à l'écran :
 - 'suppression effectuée'
 - 'suppression impossible car voiture inconnue'
 - Le programme prendra fin à la demande de l'utilisateur.
 - En fin d'exécution, libérer les données dynamiques.

PS :

Pensez aux algorithmes !

N'oubliez pas de faire des dessins de mémoire. C'est plus simple ☺.

Exercice 1703 - Niveau 2

- Ecrire une structure de maillon d'**une liste doublement chaîné** (lien vers le précédent et lien vers le suivant). Le maillon comporte un champ de type caractère.
 - Ecrire un sous-programme qui crée un maillon, saisit le caractère et rend l'adresse du maillon créé.
 - Ecrire un sous-programme récursif qui affiche toutes les valeurs de la liste passée en paramètre
 - Ecrire un sous-programme qui ajoute un nouveau maillon dans la chaîne
 - Ecrire un sous-programme qui supprime tous les maillons dont la valeur du caractère est saisie
 - Ecrire un programme principal qui met en place ces fonctionnalités par un menu. La sortie du programme sera demandée par l'utilisateur
- En fin d'exécution, libérer les données dynamiques.

PS :

Pensez aux algorithmes !

N'oubliez pas de faire des dessins mémoire. C'est plus simple ☺

Exercice 1704 - Niveau 2

- Ecrire une structure de maillon d'**une liste chaînée circulaire**. Le maillon comporte un champ de type entier.
 - Ecrire un sous-programme qui crée un maillon, saisit l'entier et rend l'adresse du maillon créé.
 - Ecrire un sous-programme récursif qui affiche toutes les valeurs de la liste passée en paramètre
 - Ecrire un sous-programme qui ajoute un nouveau maillon dans la chaîne
 - Ecrire un sous-programme qui compte le nombre de maillon de la chaîne
 - Ecrire un programme principal qui met en place ces fonctionnalités par un menu. La sortie du programme sera demandée par l'utilisateur
- En fin d'exécution, libérer les données dynamiques.

PS :

Pensez aux algorithmes !

N'oubliez pas de faire des dessins de mémoire. C'est plus simple ☺.

Exercice 1705 - Niveau 3

- Ecrire la définition de type (typedef) pour une structure contenant les informations relatives à un **étudiants** ECE :
 - Numéro (entier)
 - Nom (chaîne)
 - Prénom (chaîne)
 - Adresse (chaîne)
 - Promotion (entier)
 - Groupe (entier)
- Ecrire la définition de type (typedef) pour une structure maillon.
- Ecrire une fonction de « comptage » récursif qui reçoit l'ancre d'une liste chaînée **simple** (attention à la liste vide) et qui retourne le nombre de cellules dans la liste.
- Ecrire les sous-programme des fonctions suivantes :
 - Ajout d'un maillon (l'ajout conservera les maillons triés)
 - Suppression selon un critère (à vous de définir lequel)
 - Recherche selon un critère (à vous de définir lequel)
 - Modification d'une donnée et sur les éléments choisis selon un critère (à vous de définir la donnée à modifier et le critère de sélection)
 - Etc.
- Ecrire le programme principal mettant en jeu toutes ces fonctionnalités. Tant que l'utilisateur ne demande pas la sortie, le programme s'exécutera.

En fin d'exécution, libérer les données dynamiques.

PS :

Pensez aux algorithmes !

N'oubliez pas de faire des dessins de mémoire. C'est plus simple ☺.

Exercice 1706 - Niveau 3

- Ecrire une définition d'une structure maillon (**chainage simple**) contenant les données d'un article :
 - Numéro (entier)
 - Nom (chaîne)
 - Catégorie(chaîne)
 - Fournisseur(chaîne)
 - Prix d'achat (réel)
 - Prix de vente(réel)
 - Quantité en stock (entier)
- Ecrire un sous-programme qui crée et remplit tous ses champs. L'adresse du nouveau maillon sera rendue à l'appelant
- Ecrire un sous-programme qui range le nouveau maillon dans un tableau de listes chaînées. La liste chaînée que le maillon intégrera sera celle de sa catégorie. Le nouveau maillon et le tableau sont passé en paramètre
- Ecrire un sous-programme qui affiche chaque liste chaînée du tableau passé en paramètre
- Ecrire un programme principal qui met en place ces fonctionnalités. Il se terminera à la demande de l'utilisateur. Le tableau aura 5 listes chaînées d'articles (donc uniquement 5 catégories). Les listes chaînées seront de chainage simple.

PS :

Pensez aux algorithmes !

N'oubliez pas de faire des dessins de mémoire. C'est plus simple ☺.

ANNEXES

ANNEXE A : SEQUENCES D'ÉCHAPPEMENT

\n	NL(LF)	nouvelle ligne
\t	HT	tabulation horizontale
\v	VT	tabulation verticale (descendre d'une ligne)
\a	BEL	sonnerie
\b	BS	curseur arrière
\r	CR	retour au début de ligne
\f	FF	saut de page
\\	\	trait oblique (back-slash)
\?	?	point d'interrogation
\'	'	apostrophe
\"	"	guillemets
\0	NUL	fin de chaîne

ANNEXE B : PRIORITES DES OPERATEURS

	Classes de priorités :	Ordre de l'évaluation :
Priorité 1 (la plus forte)	()	->
Priorité 2	! ++ --	<-
Priorité 3	* / %	->
Priorité 4	+ -	->
Priorité 5	< <= > >=	->
Priorité 6	== !=	->
Priorité 7	&&	->
Priorité 8		->
Priorité 9 (la plus faible)	= += -= *= /= %=	<-

ANNEXE C : CLASSIFICATION, CONVERSION DE CARACTERE : <CTYPE.H>

Fonctions de classification et de conversion de caractères

Les fonctions suivantes ont des arguments du type *int*, dont la valeur est **EOF** ou peut être représentée comme **unsigned char**.

int isupper(int C)

retourne une valeur différente de zéro, si C est une majuscule

int islower(int C)

retourne une valeur différente de zéro, si C est une minuscule

int isdigit(int C)

retourne une valeur différente de zéro, si C est un chiffre décimal

int isalpha(int C)

retourne une valeur différente de zéro, si **islower(C)** ou **isupper(C)**

int isalnum(int C)

retourne une valeur différente de zéro, si **isalpha(C)** ou **isdigit(C)**

int isxdigit(int C)

retourne une valeur différente de zéro, si C est un chiffre hexadécimal

int isspace(int C)

retourne une valeur différente de zéro, si C est un signe d'espacement

Les fonctions de **conversion** suivantes fournissent une valeur du type *int* qui peut être représentée comme caractère; la valeur originale de C reste inchangée: **int tolower(int C)**

retourne C converti en minuscule si C est une majuscule, sinon C

int toupper(int C)

retourne C converti en majuscule si C est une minuscule, sinon C

ANNEXE D : TRAITEMENT DE CHAINES DE CARACTERES : <STRING.H>

int strlen(const char *CH1) 8.6.2.
fournit la longueur de *CH1* sans compter le '\0' final

char *strcpy(char *CH1, const char *CH2) 8.6.2.
copie *CH2* vers *CH1* ('\0' inclus); retourne *CH1*

char *strncpy(char *CH1, const char *CH2, int N)
8.6.2. copie au plus *N* caractères de *CH2* vers *CH1*; retourne *CH1*. Remplit la fin de *CH1* par des '\0' si *CH2* a moins que *N* caractères

char *strcat(char *CH1, const char *CH2) 8.6.2. ajoute *CH2* à la fin de *CH1*; retourne *CH1*

char *strncat(char *CH1, const char *CH2, int N)
8.6.2. ajoute au plus *N* caractères de *CH2* à la fin de *CH1* et termine *CH1* par '\0'; retourne *CH1*

int strcmp(const char *CH1, const char *CH2)
8.5. / 8.6.2.
compare *CH1* et *CH2* lexicographiquement et fournit un résultat:
 négatif si *CH1* précède
 CH2 zéro si *CH1* est égal à
 CH2
 positif si *CH1* suit *CH2*

ANNEXE E : TABLE ASCII ET TABLE ASCII ETENDUE

(106)base 10=(6A)base 16 = (0110 1010)base 2

La table ASCII traduit en caractères les 128 premiers codes.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(72	48	H	104	68	h
9	09	Horizontal tab	41	29)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

La table ASCII **étendue** traduit en caractères les 128 codes suivants.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	Ł	224	E0	α
129	81	ü	161	A1	í	193	C1	⊥	225	E1	β
130	82	é	162	A2	ó	194	C2	⌈	226	E2	Γ
131	83	â	163	A3	ú	195	C3	⌋	227	E3	Π
132	84	ä	164	A4	ñ	196	C4	—	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	+	229	E5	σ
134	86	ã	166	A6	à	198	C6	⌋	230	E6	μ
135	87	ç	167	A7	ø	199	C7	⌋	231	E7	Υ
136	88	è	168	A8	¿	200	C8	⌋	232	E8	ϕ
137	89	ë	169	A9	¬	201	C9	⌋	233	E9	θ
138	8A	è	170	AA	¬	202	CA	⌋	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	⌋	235	EB	δ
140	8C	î	172	AC	¼	204	CC	⌋	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	=	237	ED	Φ
142	8E	ï	174	AE	«	206	CE	⌋	238	EE	ε
143	8F	Ë	175	AF	»	207	CF	⌋	239	EF	Π
144	90	É	176	B0	☼	208	D0	⌋	240	F0	≡
145	91	æ	177	B1	☼	209	D1	⌋	241	F1	±
146	92	Æ	178	B2	☼	210	D2	⌋	242	F2	≥
147	93	ô	179	B3	⌋	211	D3	⌋	243	F3	≤
148	94	ö	180	B4	⌋	212	D4	⌋	244	F4	⌋
149	95	ò	181	B5	⌋	213	D5	⌋	245	F5	⌋
150	96	û	182	B6	⌋	214	D6	⌋	246	F6	÷
151	97	ù	183	B7	⌋	215	D7	⌋	247	F7	≈
152	98	ÿ	184	B8	⌋	216	D8	⌋	248	F8	◊
153	99	ö	185	B9	⌋	217	D9	⌋	249	F9	•
154	9A	Ü	186	BA	⌋	218	DA	⌋	250	FA	•
155	9B	Ǝ	187	BB	⌋	219	DB	■	251	FB	√
156	9C	Ǝ	188	BC	⌋	220	DC	■	252	FC	ⁿ
157	9D	Ƴ	189	BD	⌋	221	DD	■	253	FD	²
158	9E	Ƴ	190	BE	⌋	222	DE	■	254	FE	■
159	9F	f	191	BF	⌋	223	DF	■	255	FF	

BIBLIOGRAPHIE

Polycopié : Algorithmique et programmation C : Fr. Ravaut, enseignant chercheur ECE

<http://castor-informatique.fr/>

<https://code-reference.com/c/conio.h/textcolor>