

ALGORITHMIQUE  
ET  
PROGRAMMATION EN LANGAGE C

~

**Exercices – Mini TP**

**2018-2019**

**ING1**

**Elisabeth Rendler**  
Enseignant

# SOMMAIRE

<b>1</b>	<b>DECOMPOSITION DE PROBLEME ET ALGORITHME</b>	<b>4</b>
Exercice 101	Instructions .....	4
Exercice 102	Instructions .....	4
Exercice 103	Instructions et boucle.....	5
Exercice 104	Instructions et boucle.....	6
Exercice 105	Instructions, boucle et tests .....	6
Exercice 106	Instructions, boucle et test.....	7
Exercice 107	Instructions, boucle et test.....	7
Exercice 108	Instructions, boucle et test.....	8
Exercice 109	Instructions, boucle et test.....	9
Exercice 110	Cas d'étude.....	9
Exercice 111	Cas d'étude.....	10
Exercice 112	Cas d'étude.....	10
Exercice 113	Cas d'étude.....	10
Exercice 114	Cas d'étude.....	10
Exercice 115	Cas d'étude.....	11
Exercice 116	Résultat d'algorithme.....	11
Exercice 117	Résultat d'algorithme.....	11
Exercice 118	Algorithme .....	11
<b>2</b>	<b>ORDINATEUR ET ENVIRONNEMENT DE TRAVAIL</b>	<b>12</b>
Exercice 201	Ordinateur .....	12
Exercice 202	Arborescence des fichiers .....	12
Exercice 203	Environnement de travail : CodeBlocks.....	12
<b>3</b>	<b>STOCKAGE DES DONNEES : VARIABLES DE TYPE SCALAIRE</b>	<b>13</b>
Exercice 301	Variables .....	13
Exercice 302	Déclaration de variables .....	13
Exercice 303	Affectation de variables .....	13
Exercice 304	Modification de variables .....	13
Exercice 305	Variables scalaires, permutation .....	14
Exercice 306	Autres affectations.....	14
Exercice 307	évolution du contenu d'une variable par la trace d'exécution .....	14
<b>4</b>	<b>CALCULS, COMPARAISONS ET OPERATEURS LOGIQUES</b>	<b>15</b>
Exercice 401	Booléens .....	15
Exercice 402.....	.....	15
Exercice 403.....	.....	15
Exercice 404.....	.....	16
Exercice 405.....	.....	16
Exercice 406.....	.....	17
Exercice 407.....	.....	17
Exercice 408.....	.....	17
<b>5</b>	<b>ENTREES/SORTIES CLAVIER/ECRAN</b>	<b>18</b>
Exercice 501.....	.....	18
Exercice 502.....	.....	18
Exercice 503.....	.....	18
Exercice 504.....	.....	18
<b>6</b>	<b>TESTS 21</b>	
Exercice 601 - Niveau 1 .....	.....	21
Exercice 602 - Niveau 1 .....	.....	21
Exercice 603 - Niveau 2 .....	.....	21
Exercice 604 - Niveau 2 .....	.....	21
Exercice 605 - Niveau 2 .....	.....	21
<b>7</b>	<b>BOUCLES DE REPETITION</b>	<b>22</b>

Exercice 701 - Niveau 1 .....	22
Exercice 702 - Niveau 1 .....	22
Exercice 703 - Niveau 2 .....	22
Exercice 704 - Niveau 2 .....	23
Exercice 705 - Niveau 2 .....	23
Exercice 706 - Niveau 3 .....	23
Exercice 707 - Niveau 3 .....	23
Exercice 708 - Niveau 3 .....	24
Exercice 709 - Niveau 3 .....	24
Exercice 710 - Niveau 3 .....	24
<b>8 SOUS-PROGRAMMES ET PASSAGE DES PARAMETRES PAR VALEUR</b>	<b>25</b>
Exercice 801 - Niveau 1 .....	25
Exercice 802 - Niveau 2 .....	25
Exercice 803 - Niveau 2 .....	25
Exercice 804 - Niveau 2 .....	26
<b>9 POINTEURS ET PASSAGE DES PARAMETRES PAR ADRESSE</b>	<b>27</b>
Exercice 901 - Niveau 1 .....	27
Exercice 902 - Niveau 1 .....	27
Exercice 903 - Niveau 2 .....	28
Exercice 904 - Niveau 2 .....	28
Exercice 905 - Niveau 3 .....	28
<b>10 TABLEAUX</b>	<b>29</b>
Exercice 1001 - Niveau 1 .....	29
Exercice 1002 - Niveau 2 .....	29
Exercice 1003 - Niveau 2 .....	29
Exercice 1004 - Niveau 2 .....	30
Exercice 1005 - Niveau 2 .....	30
Exercice 1006 - Niveau 2 .....	30
Exercice 1007 - Niveau 2 .....	31
Exercice 1008 - Niveau 3 .....	31
<b>11 CHAINES DE CARACTERES</b>	<b>32</b>
Exercice 1101 - Niveau 1 .....	32
Exercice 1102 - Niveau 1 .....	32
Exercice 1103 - Niveau 1 .....	32
Exercice 1104 - Niveau 2 .....	32
Exercice 1105 - Niveau 3 .....	33
Exercice 1106 - Niveau 3 .....	33
Exercice 1107 - Niveau 3 .....	34
Exercice 1108 - Niveau 3 .....	34
<b>12 GENERATION DE NOMBRES ALEATOIRES</b>	<b>35</b>
Exercice 1201 - Niveau 1 .....	35
Exercice 1202 - Niveau 1 .....	35
Exercice 1203 - Niveau 2 .....	35
Exercice 1204 - Niveau 2 .....	36
TP : niveau 2 .....	37
<b>13 STRUCTURES</b>	<b>38</b>
Exercice 1301 - Niveau 1 .....	38
Exercice 1302 - Niveau 2 .....	38
Exercice 1203 - Niveau 3 .....	39
<b>14 FICHIERS : ASCII ET BINAIRES</b>	<b>40</b>
Exercice 1401 - Niveau 1 .....	40
Exercice 1402 - Niveau 1 .....	40
Exercice 1403 - Niveau 2 .....	40
Exercice 1404 - Niveau 2 .....	40
Exercice 1405 - Niveau 3 .....	41
<b>15 PROGRAMMATION EVENEMENTIELLE</b>	<b>42</b>

Exercice 1501 - Niveau 1 .....	42
Exercice 1502 - Niveau 1 .....	42
Exercice 1503 - Niveau 1 .....	42
Exercice 1504 - Niveau 2 .....	42
Exercice 1505 - Niveau 2 .....	42
Exercice 1506 - Niveau 2 .....	43
Exercice 1507 - Niveau 2 .....	43
Exercice 1508 - Niveau 3 .....	44
Exercice 1509 - Niveau 3 .....	44
Exercice 1510 - Niveau 3 .....	44
<b>16 ALLOCATION DYNAMIQUE DE MEMOIRE</b>	<b>45</b>
Exercice 1601 - Niveau 1 .....	45
Exercice 1602 - Niveau 1 .....	45
Exercice 1603 - Niveau 2 .....	45
Exercice 1604 - Niveau 2 .....	45
Exercice 1605 - Niveau 1 .....	46
Exercice 1606 - Niveau 2 .....	46
Exercice 1607 - Niveau 2 .....	46
Exercice 1608 - Niveau 2 .....	47
TP : niveau 2 .....	48
<b>17 RECURSIVITE</b>	<b>49</b>
Exercice 1701 - Niveau 2 .....	49
Exercice 1702 - Niveau 2 .....	49
Exercice 1703 - Niveau 2 .....	49
Exercice 1704 - Niveau 2 .....	49
Exercice 1705 - Niveau 2 .....	49
Exercice 1706 - Niveau 2 .....	50
Exercice 1707 - Niveau 3 .....	50
Exercice 1708 - Niveau 3 .....	50
<b>18 LISTES CHAINEES</b>	<b>51</b>
Exercice 1801 - Niveau 1 .....	51
Exercice 1802 - Niveau 1 .....	51
Exercice 1803 - Niveau 2 .....	51
Exercice 1804 - Niveau 2 .....	51
Exercice 1805 - Niveau 2 .....	51
Exercice 1806 - Niveau 2 .....	52
<b>ANNEXE A : SEQUENCES D'ECHAPPEMENT</b>	<b>53</b>
<b>ANNEXE B : PRIORITES DES OPERATEURS</b>	<b>53</b>
<b>ANNEXE C : CLASSIFICATION, CONVERSION DE CARACTERE : &lt;CTYPE.H&gt;</b>	<b>54</b>
<b>ANNEXE D : TRAITEMENT DE CHAINES DE CARACTERES : &lt;STRING.H&gt;</b>	<b>55</b>
<b>ANNEXE E : TABLE ASCII ET TABLE ASCII ETENDUE</b>	<b>56</b>

## 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*

	R						

### Exercice 102 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*

	R						

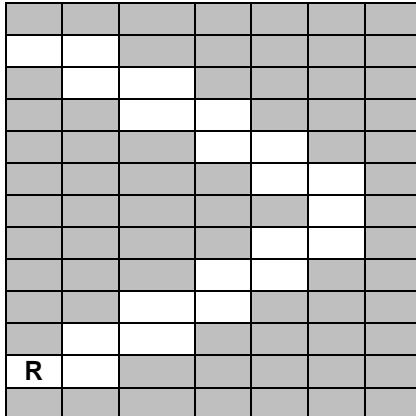


### Exercice 104 Instructions et boucle

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

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

**Bloc d'instructions**




Remarque :

Dans le cas de la boucle, « Faire X fois »,

- 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

Cette boucle est équivalente à : « Pour i de 1 à X Faire....Bloc instruction... Fpour »

### Exercice 105 Instructions, boucle et tests

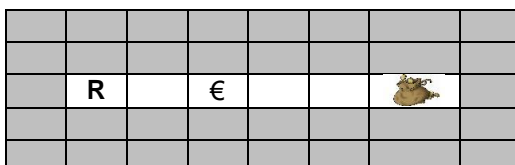
Ecrire le minimum d'instructions qui permet à un robot **R** d'arriver à la case grise. Sur son chemin, **R** ramasse les euros € qu'il trouve et les met dans sa bourse dans qu'il se trouve sur une case .

Vous avez ces instructions à votre disposition :

- *Avancer à droite*
- *Descendre d'une case*
- *Ramasser l'argent*
- *Déposer l'argent*
- *Faire X fois*

**Bloc d'instructions**

- **Si Condition Alors Bloc d'instructions FSI**
- **Si Condition Alors Bloc d'instructions Sinon Bloc d'instructions FSI**



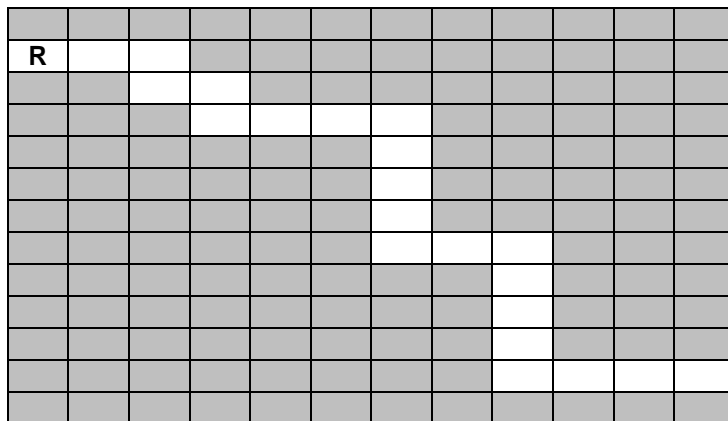
### Exercice 106 Instructions, boucle et test

Ecrire le minimum d'instructions qui permet à un robot **R** d'aller d'un bord du terrain vers un autre.  
Vous avez ces instructions à votre disposition :

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

#### **Bloc d'instructions**

- *Si **Condition** Alors **Bloc d'instructions** FSI*
- *Si **Condition** Alors **Bloc d'instructions** Sinon **Bloc d'instructions** FSI*
- *Mur à droite*
- *Mur en bas*



Remarque :

Dans le cas de la boucle, « Faire X fois »,


- 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

Cette boucle est équivalente à : « Pour i de 1 à X Faire....Bloc instruction... Fpour »

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 107 Instructions, boucle et test

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*
- *Faire X fois*

#### **Bloc d'instructions**

- *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

R												
		€										
												
									€			
												

Remarque :

Dans le cas de la boucle, « Faire X fois »,

- 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

Cette boucle est équivalente à : « Pour i de 1 à X Faire....Bloc instruction... Fpour »

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 108 Instructions, boucle et test

Ecrire le minimum d'instructions qui permet à un robot **R** de changer toutes les cases grises en cases blanches.

Vous avez ces instructions à votre disposition :

- Avancer à droite
- Avancer à gauche
- Monter d'une case
- Descendre d'une case
- Tant que **Condition** faire  
    **Bloc d'instructions**
- Si **Condition** Alors **Bloc d'instructions** FSI
- Si **Condition** Alors **Bloc d'instructions** Sinon **Bloc d'instructions** FSI
- Case grise
- Changer en case blanche
- Bord gauche
- Bord droit
- Bord en bas
- Non (=négation)

R								

### Exercice 109 Instructions, boucle et test

Ecrire le minimum d'instructions qui permet à un robot **R** de se cacher au fond de la grotte.  
Vous avez ces instructions à votre disposition :

- *Avancer tout droit*
- *Tourner à gauche*
- *Faire **X** fois*  
    **Bloc d'instructions**
- *Tant que **Condition** faire*  
    **Bloc d'instructions**
- *Si **Condition** Alors **Bloc d'instructions** FSI*
- *Si **Condition** Alors **Bloc d'instructions** Sinon **Bloc d'instructions** FSI*
- *Mur en face*
- *Non (= négation)*

R								

Remarque :

Dans le cas de la boucle, « Faire X fois »,

- 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

Cette boucle est équivalente à : « Pour i de 1 à X fois Faire....Bloc instruction... Fpour »

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 110 Cas d'étude

Vous avez un fichier de travail affiché à votre écran. Vous souhaitez l'imprimer.  
Déterminer et ordonner les actions nécessaires pour obtenir l'impression de votre document

### Exercice 111 Cas d'étude

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

Déterminer et ordonner les actions nécessaires pour obtenir cette table à partir des pièces fournies.

### Exercice 112 Cas d'étude

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

Déterminer et ordonner les actions nécessaires pour obtenir le résultat demandé.

### Exercice 113 Cas d'étude

Vous devez calculer la somme des entiers compris entre un entier A et un autre entier B lus.

Les contraintes sont :

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

Déterminer et ordonner les actions nécessaires pour obtenir le résultat demandé.

### Exercice 114 Cas d'étude

Vous devez afficher les n premiers '*nombres premiers*' naturels.

Les contraintes sont :

- lecture de l'entier n

Déterminer et ordonner les actions nécessaires pour obtenir le résultat demandé dont n est donné par l'utilisateur.

Ex : **Saisie de n** : 10  
**Résultat obtenu** : 1 3 5 7

### Exercice 115 Cas d'étude

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 :

- interdiction d'utiliser la formule mathématique
- lecture de l'entier  $n$

Déterminer et ordonner les actions nécessaires 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 ?

Saisie de la valeur  $A$

$Res$  prend la valeur 1

Pour  $i$  de 1 à 8 faire

$Res$  prend la valeur de  $Res * A$

FinPour

Afficher  $Res$

### Exercice 118 Algorithme

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

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

## **2 ORDINATEUR ET ENVIRONNEMENT DE TRAVAIL**

### **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
- compilation vs interprétation

### **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 ? Quelle est leur utilité ?
- 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 ainsi que celui de suppression d'un fichier ?

### **Exercice 203 Environnement de travail : CodeBlocks**

- Ouvrir CodeBlocks
- Explorer le menu de l'environnement
- Créer un projet
- 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 des caractères, du curseur, pour modifier la taille de la fenêtre...

### 3 STOCKAGE DES DONNEES : VARIABLES DE TYPE SCALAIRE

#### Exercice 301 Variables

- Combien y a-t-il de types scalaires en C ? Quels sont-ils ?
- Qu'est ce qui les différencie ?
- 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 ?
- 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 ?
- 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 ?
- Déclarer une variable de type caractère, nommée Carac.
- Donner la valeur du caractère 'j' à Carac.

#### Exercice 304 Modification de variables

A la suite de l'exercice 303, faire les modifications suivantes sur les variables :

- Ajouter 10 à la variable E1.
- Affecter 5.5 à la variable R1 et multiplier par 3 R1.
- Ajouter à E1 la valeur de R1. Quelle est la valeur en mémoire de chacune des deux variables
- Affecter la valeur 'F' à la variable Carac. Est-ce possible d'ajouter 2 à Carac ? Justifier votre réponse en donnant dans le cas positif la valeur de Carac.
- Affecter la valeur de Carac à E1 et à R1. Quelles sont les valeurs de ces variables ?
- Affecter la valeur 1000 à E1. Affecter la valeur de E1 à Carac. Est-ce possible ? Justifier votre réponse.

### Exercice 305 Variables scalaires, permutation

Soit deux variables Var1 et Var2 de type caractère.

- Affecter deux valeurs aux variables
- Réaliser l'échange des contenus des 2 variables sans perdre les données

### Exercice 306 Autres affectations

Soit les instructions suivantes :

```
// 0. Déclaration des variables
int i, j;
// 1. Début du code
i=10 ;
j=i;
++i ;
i++ ;
j*=2 ;
```

- Qu'est-ce que l'instruction i++ et ++i ?
- Réécrire l'instruction : j\*=2 ;

### Exercice 307 évolution du contenu d'une variable par la trace d'exécution

Représenter la trace d'exécution de l'exercice 304 et l'exercice 305.

## 4 CALCULS, COMPARAISONS ET OPERATEURS LOGIQUES

### Exercice 401 Booléens

- Qu'est-ce qu'un booléen ? Quand obtient-on une réponse booléenne ?
- En C, 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 402

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 :

- des notes littéraires : langue vivante (Nlv), de français (Nf)
- 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 :

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

### Exercice 403

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



#### Exercice 404

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 :

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

#### Exercice 405

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 ces codes sous Codeblocks pour vérifier vos résultats.

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

#### Exercice 406

Donner l'analyse qui détermine le nombre de valeurs différentes saisies parmi trois variables.

ex :

8, 8 et 8 saisis par l'utilisateur donne 1 valeur  
8,1,8 saisis donne 2 valeurs différentes  
8,2,5 saisis donne 3 valeurs différentes.

#### Exercice 407

Ecrire l'analyse calculant la moyenne de nombres entiers positifs saisis par l'utilisateur. La saisie d'un nombre négatif entraîne l'affichage du nombre de valeurs positives saisies et le calcul de la 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.

#### Exercice 408

Ecrire une analyse qui détermine 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.

## 5 ENTREES/SORTIES CLAVIER/ECRAN

### Exercice 501

- Dans un programme, afficher à l'écran « Bonjour » sans passer par une variable.
- Compléter votre programme en saisissant au clavier votre prénom. Cette saisie est à stocker dans une variable. Enfin afficher « Bonjour » suivi de la valeur stockée dans la variable, à l'écran.
- Compléter votre programme, en demandant votre âge (un entier). L'âge est stocké dans une variable. Vous l'afficherez ensuite. Afficher l'ensemble à l'écran.
- Compléter votre programme, en demandant votre moyenne des notes de baccalauréat (un réel) et en l'affichant après. Afficher l'ensemble à l'écran.

### Exercice 502

Dans un programme, déclarer un char et lui affecter une valeur choisie au hasard.

Afficher la valeur entrée en utilisant les deux formats %d et %c, qu'est-ce que ça donne ?

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

### Exercice 503

Ecrire l'analyse et le programme C qui permet de convertir caractère par caractère, des minuscules en majuscules. Chaque caractère est saisi par l'utilisateur puis converti.

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 504

Ecrire une analyse et un programme qui à partir d'un entier saisi affiche le nombre et la valeur absolue correspondant.

Séquence d'échappement pour les E/S : [ANNEXE](#)

# 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)
- Le code doit être **commenté** et correctement indenté.

## Ecriture Hongroise – Règles dans l'entreprise

L'écriture hongroise est très utilisée aujourd'hui dans de nombreuses API (Interface de programmation) comme Windows, X11 et bien d'autres.

Son but est de permettre une normalisation et une meilleure lisibilité des programmes.

Les applications actuelles sont souvent le travail de plusieurs développeurs et cela nécessite que chacun comprenne ce que les autres ont fait.

Rien ne vaut quelques exemples.

## EXEMPLES :

### Ecriture classique : mauvaise

```
#include <stdio.h>
int addk (int a, int b); /* prototype de la fonction addk */
/* la fonction addk retourne une valeur entière */
/* et accepte 2 entiers en arguments */

main()
{
    int a = 5;
    int b = 9;
    int résultat = addk (a , b); // beurk, un accent
    printf("%d",résultat); /* impression de la valeur 14 (0 + 5 + 9)*/
    résultat = addk (a , b);
    printf("%d",résultat); /* impression de la valeur 28 (14 + 5 + 9) */
}

int addk (int u, int v)
{ static int k = 0; /* la variable k est automatique */
  k = k + u + v;
  return(k); }
```

### Version écriture hongroise : correcte

```
#include <stdio.h>
int AjouteK (int valA, int valB); /* prototype de la fonction AjouteK */
/* la fonction AjouteK retourne une valeur entière */
/* et accepte 2 entiers en arguments */

int AjouteK (int valU, int valW)
{
    static int valK = 0; /* la variable valK est automatique */
    valK = valK + valU + valW;
    return(valK);
}

// Et le main, à la fin ce qui le rend facile à trouver
int main()
{
    // 0. Déclaration des variables
    int valA = 5;
    int valB = 9;
    int resultat = 0 ;

    // 1. traitements
    resultat = AjouteK (valA , valB);
    printf("%d",resultat); /* impression de la valeur 14 (0 + 5 + 9)*/
    resultat = AjouteK(valA , valB);
    printf("%d",resultat); /* impression de la valeur 28 (14 + 5 + 9) */
    return 0 ;
}
```

## 6 TESTS

### Exercice 601 - Niveau 1

Ecrire l'analyse puis le code C permettant de déterminer pour une valeur saisie si elle est multiple de 5 et de 2 ?

### Exercice 602 - Niveau 1

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

### Exercice 603 - Niveau 2

En vous efforçant à nouveau d'exploiter la table ASCII mais sans entrer de valeur « en dur » (genre 32), écrivez l'analyse et le programme C permettant de déterminer si le caractère entré par l'utilisateur est un chiffre, une lettre majuscule, une lettre minuscule ou autre chose... Après analyse, vous afficherez à l'écran la nature du caractère saisi.

Table ASCII : [ANNEXE](#)

### Exercice 604 - Niveau 2

Réécrire en C l'exercice 406.

### Exercice 605 - Niveau 2

Ecrire un programme C qui teste si un nombre est positif ou non. Ce test sera fait 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'analyse et le programme C qui affiche à l'écran 10 fois la même chaîne « Bienvenue en TP ». Vous ajouterez à la fin de cette chaîne le numéro de l'indice de boucle tel que présenté ci-dessous  
    Bienvenue en TP pour l'indice 1  
    Bienvenue en TP pour l'indice 2  
    Bienvenue en TP pour l'indice 3  
    Bienvenue en TP pour l'indice 4  
    Bienvenue en TP pour l'indice 5  
    Bienvenue en TP pour l'indice 6  
    Bienvenue en TP pour l'indice 7  
    Bienvenue en TP pour l'indice 8  
    Bienvenue en TP pour l'indice 9  
    Bienvenue en TP pour l'indice 10
- Modifier votre programme saisissant le nombre d'affichages donc d'itérations à faire.
- Modifiez votre programme afin que le nombre d'itérations à faire soit toujours positif mais inférieur à 10 (ceci s'appelle un blindage de saisie)

### Exercice 702 - Niveau 1

Ecrire l'analyse et le programme C 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.

Pour cet exercice, vous ne devez modifier que les éléments de la **boucle for**.

### Exercice 703 - Niveau 2

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

Pour cet exercice, **vous ne devez pas modifier** les éléments de la **boucle for**.

Saisie : 5  
Affichage : -15  
          -12  
          -9  
          -6

- 3  
0  
3  
6  
9  
12  
15

### Exercice 704 - Niveau 2

Ecrire l'analyse et le programme qui :

- saisit des entiers positifs. La saisie se terminera dès qu'un nombre strictement négatif est saisi.
- affiche le maximum d'une série de nombres saisis au clavier (sans tableau).
- affiche le nombre total de valeurs saisies par l'utilisateur sans prendre en compte le nombre négatif

### Exercice 705 - Niveau 2

Ecrire l'analyse et le programme C 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 mais ne sera pas 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 2 de chaque mois sur un compte vous rapportant 0.3% mensuel, avec des intérêts calculés tous les 30 jours, le premier de chaque mois. Les intérêts se cumulent.

Ecrivez l'analyse descendante, l'algorithme ou l'organigramme puis le code C permettant d'afficher le solde de votre épargne au bout d'un nombre de mois saisi par l'utilisateur.

### Exercice 707 - Niveau 3

Ecrire l'analyse et le programme C qui affiche les nombres premiers compris entre 0 et l'entier positif saisi.

N'oubliez pas de blinder la saisie



### Exercice 708 - Niveau 3

Ecrire l'analyse et le programme C qui saisit un entier n et affiche le résultat U(n) défini par :

$$U(0) = 3$$

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

### Exercice 709 - Niveau 3

Ecrire l'analyse et le programme C 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 C 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.

Exemple : hauteur=4

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

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

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

## 8 SOUS-PROGRAMMES ET PASSAGE DES PARAMETRES PAR VALEUR

### Exercice 801 - Niveau 1

Ecrire l'analyse et le code C 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 un nombre entier et affiche son carré, calculé par le sous-programme précédent

### Exercice 802 - Niveau 2

- Ecrire l'analyse et le sous-programme d'un menu qui propose les opérations suivantes :
  1. Ajouter 1
  2. Multiplier par 2
  3. Soustraire 4
  4. Carré de l'entier
  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.

Pour l'option 4, vous utiliserez le sous-programme écrit dans l'exercice précédent (réutilisation de sous-programme)

- Ecrire un sous-programme pour chaque opération que le menu propose.
- Ecrire l'analyse et 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 de l'exercice

### Exercice 803 - Niveau 2

Ecrire l'analyse 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.

- Ecrire un sous-programme qui calculera l'heure suivante à partir d'une heure passée en paramètre et rendra au programme appelant le résultat.
- Le programme principal 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 à chaque tour de boucle. Pour avoir le temps de voir les affichages, mettez un temps d'attente avec `pause()`, `sleep()`...ou bien une boucle qui n'incrmente les minutes qu'au bout d'un certain nombre de tours. Ecrire ce programme principal.
- 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.
- Mettre en place une horloge dont l'affichage des heures va de 0 à 12 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 2 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 :
  - demande à l'utilisateur la saisie de 2 entiers val1 et val2
  - affiche leur valeur respective
  - appelle le sous-programme précédent
  - affiche val1 et val2 après l'appel pour vérifier leur contenu.
- Que constatez-vous au sujet de val1 et val2 ?
- Qu'aurait-il fallu faire pour que la modification effectuée dans le sous-programme se répercute au niveau du programme principal ?
- Faites la modification proposée et vérifiez son efficacité.

## 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 PARAMETRES PAR ADRESSE

### Exercice 901 - 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 902 - Niveau 1

- Ecrire l'analyse 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 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 sont passés par adresse en paramètre.
- Ecrire l'analyse et le programme principal qui :
  - appelle le sous-programme précédent
  - affiche les résultats rendus par le sous-programme.
- Faire un schéma mémoire
- Dessiner le graphe d'appel de cet exercice

### Exercice 903 - Niveau 2

- Ecrire l'analyse et le sous-programme qui :
  - saisit l'identifiant d'un article, son prix unitaire et le taux de TVA
  - calcule le prix TTC de l'article
  - rend toutes les données saisies (identifiant, prix unitaire, TVA) et calculées par adresse
  - rend la donnée du produit (prix TTC) par l'instruction **return**.
- Ecrire l'analyse et le programme principal qui :
  - appelle le sous-programme précédent
  - affiche tous les résultats rendus par le sous-programme.
- Faire un schéma mémoire
- Dessiner le graphe d'appel de cet exercice

### Exercice 904 - Niveau 2

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

Le sous-programme recevra 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 résultats calculés. Les calculs sont à rendre au programme appelant. Le reste des données utiles au calcul demandé sera saisi dans le sous-programme.
- Ecrire l'analyse et le programme principal qui :
  - s'exécute tant que l'utilisateur ne demande pas la sortie
  - saisit l'initiale 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
- Faire un schéma mémoire
- Dessiner le graphe d'appel de cet exercice

### Exercice 905 - Niveau 3

- Ecrire l'analyse 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 de 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 entre 18 et 39 ans, la réduction du billet est de 50%
    - pour les personnes jeunes entre 40 et 60 ans (!), le tarif est normal (aucune réduction)
  - N'oubliez pas de blinder les saisies.
  - rend le prix du billet demandé (paramètre par adresse).
- Ecrire l'analyse 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 l'âge des personnes concernées.

Pour le calcul de l'ensemble du groupe, vous devez appeler le sous-programme précédent.

Le résultat du calcul sera rendu au programme principal (résultat obtenu renvoyé par l'instruction **return**).
- Ecrire le programme principal qui
  - calcule le prix de billet d'une classe d'élèves avec leurs accompagnateurs.
  - affiche le prix total des billets d'un groupe
  - recommence tant que l'utilisateur n'aura pas demandé de quitter l'application
- Dessiner le graphe d'appel de cet exercice

## 10 TABLEAUX

### Exercice 1001 - Niveau 1

Ecrire l'analyse et le programme qui :

- remplit un tableau de 13 entiers saisis par l'utilisateur.
- 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 2

- Ecrire l'analyse et le sous-programme permettant le remplissage d'un tableau de 10 notes (notes réelles et blindées entre 0 et 20). Le tableau sera reçu en paramètre par le tube.
- Ecrire l'analyse et le sous-programme permettant l'affichage d'un tableau de 10 notes passé en paramètre.
- En appelant les sous-programmes précédant, écrire un programme permettant la saisie et l'affichage de 10 notes stockées dans un tableau.  
Puis ce sous-programme calculera dans un nouveau sous-programme la moyenne de ces 10 notes. L'affichage de cette moyenne sera fait dans un autre sous-programme à écrire.
- Quelle serait la conséquence d'un affichage de cette moyenne dans le programme principal ? Ecrivez le nouveau sous-programme correspondant à cette nouvelle contrainte de cahier des charges (sans supprimer l'ancien).
- Ecrire l'analyse et le sous-programme permettant la recherche du minimum dans un tableau de 10 notes. Réfléchir au(x) paramètre(s) nécessaire(s) à ce traitement.
- Ecrire l'analyse et le sous-programme permettant la recherche du maximum dans un tableau de 10 notes. Réfléchir au(x) paramètre(s) nécessaire(s) à ce traitement.
- Ecrire le programme principal qui met en place tous ces sous-programmes
- Dessinez le graphe d'appel correspondant à tout l'exercice.

### Exercice 1003 - 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 l'analyse et le sous-programme qui trie par ordre croissant les éléments du tableau (**méthode de tri à bulle**)
- Ecrire l'analyse et 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 chaque partie de cette exercice l'analyse est demandée.  
La taille du tableau est 17.

### Exercice 1004 - 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 l'analyse et le sous-programme qui trie par ordre croissant les éléments du tableau (**méthode de tri par insertion**)
- Ecrire l'analyse et le sous-programme qui recherche un entier dans la table. L'entier recherché et la table sont passés en paramètre (recherche de l'élément par un parcours de tableau). Il faut rendre un résultat indiquant que l'entier est trouvé ou non. Ce résultat est à rendre au programme appelant.
- Ecrire l'analyse et 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. L'entier à rechercher est saisi avant l'appel au sous-programme de recherche (question 3).

Pour chaque partie de cette exercice l'analyse est demandée.  
La taille du tableau est 9.

### Exercice 1005 - Niveau 2

- Ecrire l'analyse et le sous-programme qui saisisse toutes les valeurs d'un tableau de 20 réels. Le tableau est passé en paramètre.
- Ecrire l'analyse et le sous-programme qui trie un tableau de réel passé en paramètre (**méthode de tri par insertion**)
- Ecrire l'analyse et le sous-programme qui recherche un élément dans un tableau. Les paramètres sont le tableau et le réel recherché. La méthode de recherche est la **dichotomie**. Le résultat de la recherche sera affiché dans le sous-programme
- Faire l'analyse et 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 1006 - Niveau 2

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

### Exercice 1007 - Niveau 2

- Dessiner la représentation schématique d'un tableau défini comme suit : `char tab[3][4][2]`  
3 lignes  
4 colonnes  
2 profondeurs
- Ecrire l'analyse et le programme qui :
  - saisit un caractère (alphabétique, numérique ou autres)
  - compte le nombre d'occurrences (= le nombre de fois où apparaît un élément) de ce caractère saisi. Le résultat sera rendu au programme appelant et le caractère saisi passera en paramètre par adresse.
- Faire l'analyse et la programme qui :
  - saisit les éléments du tableau
  - affiche tout le tableau
  - appel du sous-programme de recherche d'occurrences
  - affiche le caractère saisi ainsi que son nombre d'occurrence dans le tableau

### Exercice 1008 - Niveau 3

- Ecrire l'analyse et le sous-programme qui saisit un tableau à deux dimensions de caractères : 4 lignes et 5 colonnes : cf **matrice saisie** dans l'exemple
- Ecrire l'analyse et le sous-programme qui transforme les voyelles minuscules en majuscules : cf **matrice transformée** dans l'exemple
- Ecrire l'analyse et le sous-programme qui affiche le tableau de caractères
- Ecrire l'analyse et 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 l'analyse et 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 CARACTERES

### Exercice 1101 - Niveau 1

Ecrire l'analyse et le programme qui :

- saisit deux chaînes de caractères
- compte leur nombre de caractères respectif
- affiche les chaînes et leur taille calculée.

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'analyse et le programme qui :

- saisit deux chaînes
- compare alphabétiquement ces deux chaînes
- affiche le résultat de la comparaison : par exemple : les chaînes sont identiques ou la 1<sup>er</sup> chaîne est avant la 2<sup>ème</sup> chaîne dans l'ordre alphabétique ou le contraire.

### Exercice 1103 - Niveau 1

- Ecrire l'analyse et le sous-programme qui saisit une chaîne et la rend à l'appelant
- Ecrire l'analyse 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.
- Ecrire l'analyse et le sous-programme qui compte le nombre de mots (ensembles de caractères séparé par un espace) stockés dans la chaîne passée en paramètre. Rendre ce résultat à l'appelant
- Ecrire l'analyse et le programme qui :
  - appelle les sous-programmes
  - affiche la chaîne ainsi que les résultats obtenus

### Exercice 1104 - Niveau 2

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

Les différentes options seront proposées par un menu, lui-même sous-programme. Ce sous-programme retournera l'option choisie. Les fonctionnalités sont :

- 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). La chaîne sera affichée à l'écran
- compter le nombre de voyelles (minuscules et majuscules) dans une chaîne
- convertir la chaîne saisie en minuscules
- comparer 2 chaînes à saisir (utilisez strcmp)
- concaténer 2 chaînes dans la première (attention aux débordements)

- crypter une chaîne en appliquant un décalage circulaire dont la valeur est entrée par l'utilisateur
- décrypter une chaîne de caractère cryptée en appliquant un décalage circulaire inverse à partir d'une valeur entrée par l'utilisateur

Chaque fonctionnalité proposée sera gérée dans un sous-programme (c'est-à-dire que le sous-programme correspondant à la fonctionnalités demandée sera appelé dans le programme principal)  
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'ensembles des sous-programmes écrits. Il recommencera l'exécution tant que l'utilisateur n'aura pas demandé explicitement l'arrêt.

### Exercice 1105 - Niveau 3

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

### Exercice 1106 - Niveau 3

Ecrire l'analyse et le code C de programme et du sous-programme suivants :

Les différentes options seront proposées par un menu, lui-même sous-programme. Ce sous-programme 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 phrase est passée en paramètre et la deuxième est le miroir de la première. Les deux phrases sont séparées par un séparateur « : »  
exemple :  
1<sup>er</sup> phrase passée en paramètre : « il fait beau »  
2<sup>ème</sup> phrase 'miroir' : « uaeb tiaf li »  
phrase résultat : « il fait beau : uaeb tiaf li »

La phrase résultat est rendue au programme appelant.

- calcul du nombre de consonnes de la phrase passée en paramètre. Dans ce sous-programme, il est demandé d'utiliser le sous-programme précédent qui dénombre les voyelles.  
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'ensembles 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'analyse 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 contractée 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 contractée : « orur »

Cas 2 :

chaîne passée en paramètre : «ode »

chaîne contractée : impossible à construire

message affiché : « chaîne non valable »

- Ecrire l'analyse et le sous-programme qui saisit une chaîne et vérifie si cette chaîne existe ou non dans une première chaîne reçue en paramètre. Rendre un indicateur qui fera office de booléen.
- Ecrire le programme principal mettant en jeu les sous-programmes après la saisie d'une chaîne. Les résultats reçus seront affichés.

### Exercice 1108 - Niveau 3

- Ecrire l'analyse et le sous-programme qui saisit un verbe du 1<sup>er</sup> groupe de conjugaison (terminaison « -er »). Ce verbe est rendu au programme appelant.
- Ecrire l'analyse et le sous-programme qui affiche la conjugaison au présent du verbe passer en paramètre.
- Ecrire l'analyse et 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 GENERATION DE NOMBRES ALEATOIRES

### Exercice 1201 - Niveau 1

- Ecrire un sous-programme qui génère aléatoirement un nombre. Ce nombre est rendu au programme appelant.
- Ecrire l'analyse 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

### Exercice 1202 - Niveau 1

- Ecrire un sous-programme qui génère aléatoirement un nombre. Ce nombre est rendu au programme appelant.
- Ecrire l'analyse et le sous-programme qui remplit un tableau d'entiers aléatoires (20 places). Pour cela, le sous-programme précédent sera appelé. Dans le tableau toutes les valeurs n'apparaîtront qu'une seule fois, aucun doublon n'est permis.
- Ecrire l'analyse et le sous-programme qui compte le nombre de multiples de 10, le nombre d'entiers inférieurs à 100 ainsi que 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

### Exercice 1203 - Niveau 2

Ecrire l'analyse 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 l'analyse et 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 l'analyse et 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.

Table ASCII : [ANNEXE](#)

### 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 0 du tableau histogramme, puis combien l'image contient de pixels de valeur 1 à ranger dans la case 1... ainsi de suite jusqu'à compter le nombre de pixels à 255.

**Attention !!** Il y a une méthode de calcul rapide et une méthode très lente. Réfléchissez !

Compléter le programme précédent pour qu'il calcule l'histogramme de l'image après son remplissage aléatoire.

Affichez cet histogramme de manière lisible à l'écran.

#### Exercice 1207

La binarisation d'une image consiste, **dans une image secondaire afin de ne pas modifier l'image originale**, à mettre à 0 tous les pixels de l'image originale inférieurs à une valeur seuil entrée par l'utilisateur ; et à 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 du film
  - son année de réalisation
  - le réalisateur
  - la durée en minutes
  - un booléen disant si vous l'avez vu ou pas
  - une note sur 10
- Ecrire un sous-programme chargé de remplir une instance de cette structure dont l'adresse est reçue en paramètre
- Ecrire un sous-programme qui affiche les données d'une structure
- Ecrire un programme principal qui met en place ces deux sous-programmes

### Exercice 1302 - Niveau 2

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

- Le nom
  - Le prénom
  - L'année de naissance
  - Un tableau de 5 notes
- 
- Définir la structure **t\_etudiant**.
  - Ecrire un sous-programme chargé de remplir une instance de cette structure dont l'adresse est passée par adresse
  - Ecrire un sous-programme qui remplit un tableau de 20 structures **t\_etudiant** passé en paramètre en appelant le sous-programme précédent
  - Ecrire un sous-programme qui affiche une structure
  - Ecrire un sous-programme qui affiche un tableau de 20 structures **t\_etudiant** passé en paramètre en appelant le sous-programme précédent
  - Ecrire un programme principal qui :
    - Déclare un tableau de 20 structures **t\_etudiant**
    - Appelle le sous-programme qui remplit tous les éléments du tableau
    - Appelle le sous-programme qui affiche tous les éléments du tableau

### Exercice 1203 - Niveau 3

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

- La désignation
  - La catégorie
  - La date de péremption (composée de jour, mois et année)
  - La quantité en stock
  - Le prix de vente
- 
- Définir les structures **t\_article** et **t\_date**.
  - Ecrire un sous-programme chargé de remplir une instance de cette structure dont l'adresse est passée par adresse
  - 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 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 l'analyse et le sous-programme qui reçoit une catégorie en paramètre et qui affiche les produits 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 (rappel la date est composée de trois champs définis dans une structure **t\_date**).
  - Ecrire un programme principal qui :
    - Déclare un tableau de 15 structures **t\_article**
    - 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 produits de cette catégorie
    - Saisit une date et appelle le sous-programme qui compte le nombre de produit 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 le contenu de 15 entiers saisis, sur une même ligne, séparé par une tabulation ('\t').  
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 et 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'analyse 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, 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

- Soit la structure **t\_film** contenant l'identifiant, le titre, le réalisateur, l'année, le genre... Définir la structure t\_film.
- Ecrire la fonction de saisie de tous les champs d'une instance t\_film.
- Ecrire un sous-programme qui sauvegarde les instances des structures d'un tableau dans un fichier texte. Le nom du fichier est reçu en paramètre.
- Ecrire un sous-programme recevant en paramètres le nom d'un fichier et dont le rôle est de lire la totalité du contenu du fichier pour construire le tableau des films archivés.
- Ecrire un sous-programme qui vide tous les champs d'un film stocké dans le tableau dont l'indice passe en paramètre

- Ecrire le programme principal qui :
  - définit un tableau de 15 structures.
  - saisit le nom du fichier texte
  - charge les films du fichier dans le tableau
  - affiche les films du tableau
  - supprime 2 films. Les indices de ces films sont saisis
  - ajoute deux nouveaux films dans le tableau en utilisant le sous-programme de saisie à la place de ceux supprimés
  - modifie tous les films datant de 2012 en les mettant une valeur d'année saisie
  - sauvegarde tous les films du tableau dans le fichier de départ. Le nom du fichier est reçu en paramètre. Il est le même que le fichier de départ.

### Exercice 1405 - Niveau 3

- Définir le 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 places
  - thème : tableau de chaînes de caractères : 5 places
  - 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 PROGRAMMATION EVENEMENTIELLE

### Exercice 1501 - Niveau 1

Ecrire l'analyse et le programme C affichant un **X** qui se déplace seul à l'écran.

En mémoire, le **X** sera stocké dans une matrice de caractères à 2 dimensions 10 lignes x 20 colonnes. La position initiale du **X** est aléatoire au niveau du bord gauche de l'écran. Son déplacement est autonome. Il commence du bord droit et se dirige vers la gauche, puis disparaît.

L'affichage de chaque étape du déplacement du **X** correspond à l'affichage complet de la matrice après effacement de l'écran. L'affichage sera écrit dans un sous-programme dont la matrice est passée en paramètre.

(Utilisez la fonction `system(« cls »)` ; ou `system(« clear »)` ; `clrscr()` ; (en fonction des systèmes) pour nettoyer l'écran et éviter le déplacement diagonal)

### Exercice 1502 - Niveau 1

Améliorer l'analyse et le programme précédent pour que le **X** réapparaisse à gauche lorsqu'il est sorti à droite.

### Exercice 1503 - Niveau 1

Modifier encore l'analyse et le programme pour que l'utilisateur puisse cette fois changer la direction de déplacement du **X** en utilisant les touches 2, 4, 6 et 8 du pavé numérique.

Assurez-vous que le verrouillage numérique du pavé soit actif, sinon choisir d'autres touches pour le changement de direction. De toutes façons, il suffit d'appuyer une seule fois sur la touche pour changer de direction car **X** est autonome !

Le **X** réapparaîtra en haut ou en bas, à gauche ou à droite, à l'opposé de sa sortie de l'écran.

### Exercice 1504 - Niveau 2

Ajouter une bordure à votre cadre de jeu pour observer l'effet stroboscopique (= effet clignotant) à l'écran.

### Exercice 1505 - Niveau 2

Pour éviter cet effet clignotant de l'écran, tenter de remplacer l'opération d'effacement complet de l'écran suivi du réaffichage complet de la matrice terrain avec ses bordures par une opération de mise à jour des parties mobiles uniquement à l'écran. Les bordures n'étant plus réaffichées à chaque fois, l'effet clignotant disparaît.

Pour cela et **SOUS WINSOWS UNIQUEMENT**, améliorer votre programme en y intégrant la fonction `gotoligcol( lig , col )` qui permet de placer le curseur d'affichage à l'endroit voulu.

Inclure d'abord la librairie **windows.h**.

Indication :

Cette fonction emmène le curseur à la position `lig, col` à l'écran. Le coin haut gauche étant en `lig = 0` et `col = 0` on dit qu'il est en (0,0). Vous pouvez alors y afficher le caractère de votre choix à la position de votre choix.

```
void gotoligcol( int lig, int col )
{
    COORD mycoord;

    mycoord.X      = col;
    mycoord.Y      = lig;

    SetConsoleCursorPosition( GetStdHandle( STD_OUTPUT_HANDLE ), mycoord );
}
```

### Exercice 1506 - Niveau 2

Modifier votre programme, en écrivant des structures d'objets qui, lorsque X (=Pacman) rencontre ces objets, :

- augmenteront ou diminueront le score
- accéléreront ou diminueront la vitesse de Pacman
- etc..

Mettre ces objets dans un tableau de structures d'objets. Ces objets sont placés de façon aléatoire et disparaissent quand ils ont été « utilisés » par le Pacman. Il ne peut y avoir qu'un seul objet par case. Ces modifications seront le plus possible mises en places dans des sous-programmes.

Faites de mêmes avec des personnages plus ou moins hostiles au Pacman. Ces personnages sont autonomes et peuvent avoir des vitesses plus ou moins rapides. Vous mettrez le Pacman dans ce tableau de personnage.

Ces modifications seront le plus possible mises en place dans des sous-programmes.

### Exercice 1507 - Niveau 2

Améliorer votre programme au niveau de l'affichage afin de donner une couleur au Pacman, à chacun des éléments de votre programme.

Inclure d'abord la librairie **conio.h**.

Vous vous inspirerez de google pour mettre les couleurs :

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

```
#include<stdio.h>
#include<conio.h>

int main()
{
    textcolor(BLUE+BLINK);
    cprintf("textcolor c example with a blue blinking text");
    getch();
    return 0;
}
```

Améliorer votre programme au niveau de l'affichage afin de donner une couleur au Pacman, à chacun des éléments de votre programme.

Cette option couleur peut être supprimée selon la saisie d'un caractère de votre choix avant le lancement de la partie

### Exercice 1508 - Niveau 3

Créer un nouveau terrain où peut évoluer tous vos éléments et mettre en place un moyen pour que les éléments mobiles puissent ou non se télé-transporter d'un terrain à un autre. Pour cela établir des règles de télé-transportation d'un terrain à un autre.

### Exercice 1509 - Niveau 3

Prévoir une sauvegarde et un chargement de la partie interrompue. Ces sauvegardes seront faites dans un fichier.

Bien analyser ce que vous allez sauvegarder et sous quelle forme.

Prévoir aussi une sauvegarde qui permettra à la fin de chaque partie d'afficher les 5 meilleurs scores obtenus.

Vous pouvez mettre en place un chronomètre de partie avec la fonction `clock()` de la bibliothèque `time.h`.

### Exercice 1510 - Niveau 3

Pour les plus forts en programmation parmi vous, vous pouvez même faire en sorte que le PACMAN devienne un SNAKE en accolant un nouveau X au serpent à chaque fois que vous réussissez à « manger » un objet.

Remarques :

- **Aléatoire** : « `srand(time(NULL))` » à appeler une fois au début du programme puis « `rand()%x` » à chaque fois que vous avez besoin d'un entier dans l'intervalle `[0; x-1]`.
- Faire attention aux débordements mémoire.
- **Vous ferez valider chaque étape par votre chargé de TD/TP.**

## 16 ALLOCATION DYNAMIQUE DE MEMOIRE

### Exercice 1601 - Niveau 1

- Ecrire un sous-programme qui saisit deux phrases dans un contenu fixe puis les stocke l'une après l'autre dans une chaîne ajustée en mémoire. La phrase ajustée sera rendue au programme appelant
- Ecrire un programme principal qui appelle le sous-programme et qui affiche la phrase résultat.

### Exercice 1602 - Niveau 1

- Ecrire un sous-programme qui lit 10 mots au clavier (longueur maximale : 50 caractères). Chaque mot est stocké dans un endroit mémoire ajusté en mémoire. Chaque adresse de ces espaces mémoire est mise dans un tableau de pointeurs MOT (tableau de 10 places). Le tableau est passé en paramètre.
- Ecrire un sous-programme qui affiche tous les mots de ce tableau
- Ecrire l'analyse et le sous-programme qui efface les mots un à un, en suivant l'ordre alphabétique et en libérant leur espace en mémoire. Afficher à chaque fois les mots restants en attendant la confirmation de l'utilisateur (par 'Enter').
- Ecrire un programme principal qui met en jeu toutes ces fonctionnalités

### Exercice 1603 - Niveau 2

Ecrire un programme qui lit 10 mots au clavier (longueur : 50 caractères utiles) et attribue leurs adresses à un tableau de pointeurs MOT.

Trier les phrases alphabétiquement en n'échangeant que les pointeurs.

Copier les chaînes de MOT selon l'ordre alphabétique en une seule 'phrase' dont l'adresse est affectée à un pointeur PHRASE. Réserver l'espace nécessaire à la PHRASE avant de copier les mots. Libérer la mémoire occupée par chaque mot après l'avoir copié. Utiliser les fonctions de `<string>`.

### Exercice 1604 - Niveau 2

Ecrire un programme qui lit 10 phrases d'une longueur maximale de 200 caractères (utiles) au clavier. Ces phrases sont mémorisées dans un tableau de pointeurs sur **caractère** (donc en réservant dynamiquement l'emplacement en mémoire ajusté pour chaque chaîne). Ensuite, les phrases sont triées alphabétiquement les unes par rapport aux autres. Le tableau résultat est affiché.

### Exercice 1605 - Niveau 1

Ecrire un programme qui saisit la taille d'un tableau de réels et alloue le tableau dynamiquement.

Ecrire un sous-programme d'affichage de ce tableau

Ecrire un programme principal qui met en place ces sous-programmes

### Exercice 1606 - 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.
- Dans le programme principal, créer ce tableau dynamique à partir de l'entier saisi dans le 1<sup>er</sup> sous-programme. Remplir le tableau aléatoirement avec des valeurs comprises entre -25 et 150. Sauvegarder les valeurs de ce tableau dans un fichier *resultat.txt*.

### Exercice 1607 - Niveau 2

- Ecrire une fonction permettant l'allocation dynamique d'une matrice d'entiers après avoir demandé la valeur des 2 dimensions à l'utilisateur.
- Ecrire une procédure permettant le remplissage **aléatoire** de la matrice précédente avec des valeurs entre 0 et 255 (bornes comprises).
- Ecrire un sous-programme permettant le tri en ordre croissant des valeurs de la matrice d'entiers. Dans ce programme créer une nouvelle matrice résultat et stocker les valeurs triées du 1<sup>er</sup> tableau.

Exemple :

8	10	12
15	23	45
51	52	56

- 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.
- Vous dessinerez le graphe d'appel de votre programme en faisant apparaître le fonctionnement des tubes et des mécanismes de retour.
- Ecrire et inclure à votre projet la bibliothèque personnelle au format .h

## Exercice 1608 - Niveau 2

- Définir une structure de maillon dinosaure :
  - Nom (chaîne de 15 caractères utiles)
  - Famille (chaîne de 10 caractères utiles)
  - Date de disparition (entier)
  - Carnivore/herbivore (caractère)
- Ecrire un sous-programme appelé *creer()* qui alloue de la place mémoire à une structure, remplit tous les champs et rend l'adresse obtenue.
- Ecrire un sous-programme appelé *afficher()* qui affiche une structure dont l'adresse est passée en paramètre.
- Ecrire un programme principal qui :
  - déclare un tableau de pointeurs de structures
  - appelle le sous-programme *creer()* pour chaque élément du tableau
  - affiche tous les éléments du tableau avec le sous-programme *afficher()*



## **TP : niveau 2**

- Définir le type correspondant à la structure d'un rendez-vous :
  - un champ date au format jour/mois/année
  - un champ heure début
  - un champ heure fin
  - un champ libellé
  - un booléen rappel

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.
- 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
- Afin de ne pas être obligé de tout ressaisir à chaque fois et de ne pas perdre les rendez-vous créés, vous coderez maintenant la sauvegarde sur disque. Vous créerez les sous-programmes :
  - sauvegarder les RDV dans un fichier (vous choisirez texte ou binaire)
  - charger les RDV à partir d'un fichier de sauvegarde

Le sous-programme de chargement sera lancé systématiquement à l'ouverture de votre programme afin de monter en mémoire tous les RDV contenus dans le fichier. (Vous pourrez discuter avec votre chargé de TD-TP du bienfondé de cette méthode). Au passage, vous afficherez à l'écran tous les rendez-vous du jour. Si parmi ces rendez-vous certains ont le champ booléen rappel à vrai, vous afficherez une fenêtre pop-up de rappel affichant le libellé du RDV 15 minutes avant celui-ci.

- Pour optimiser la gestion de la mémoire, vous adapterez votre définition de type(s) afin de pouvoir stocker vos rendez-vous en liste chaînée.
- Codez de nouveau votre bibliothèque de sous-programmes de manipulation des rendez-vous afin de les adapter aux listes chaînées (ne supprimez pas les autres sous-programmes).

## 17 RECURSIVITE

### Exercice 1701 - Niveau 2

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

### Exercice 1702 - Niveau 2

Ecrire un programme contenant un sous-programme récursif qui affiche le contenu d'un tableau dynamique d'entiers reçu en paramètre.

Le tableau aura été alloué et rempli dans le programme principal ou dans un autre sous-programme.

### Exercice 1703 - Niveau 2

Ecrire un programme contenant un sous-programme récursif permettant de calculer la longueur d'une chaîne de caractères (comme le ferait la fonction STRLEN).

### Exercice 1704 - Niveau 2

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

$$\begin{aligned}U_0 &= 7 \\ U_n &= 3 \cdot U_{n-1} + 5/2\end{aligned}$$

Mettre en place ce sous-programme dans un programme principal.

### Exercice 1705 - Niveau 2

Ecrire un sous-programme récursif qui calcule la puissance d'un nombre.

Le programme principal saisit le nombre et la puissance puis les passe en paramètre au sous-programme au moment de l'appel. Puis il affiche le résultat obtenu.

### Exercice 1706 - Niveau 2

Ecrire l'analyse et le code permettant de trier un tableau de flottants.

- Le sous-programme « saisie » sera **récuratif** et permettra de saisir n flottants
- Le sous-programme « affiche », **récuratif également**, permettra de les afficher. Les réels seront affichés du premier au dernier. Identifier ce qui changerait si on affichait les éléments du dernier au premier.
- Le sous-programme « Tri », qui devra lui encore être **récuratif**, permettra de trier le tableau de flottants dans un ordre croissant
- Le programme principal qui met en place tous ces sous-programme.

### Exercice 1707 - Niveau 3

La **méthode de diviser pour régner** est une méthode qui permet, parfois de trouver des solutions efficaces à des problèmes algorithmiques. L'idée est de découper le problème initial, de taille n, en plusieurs sous-problèmes de taille sensiblement inférieure, puis de recombinaison les solutions partielles.

Pour le tri d'un tableau de grande taille, cette méthode de tri est plus rapide qu'une méthode de tri déjà vue.

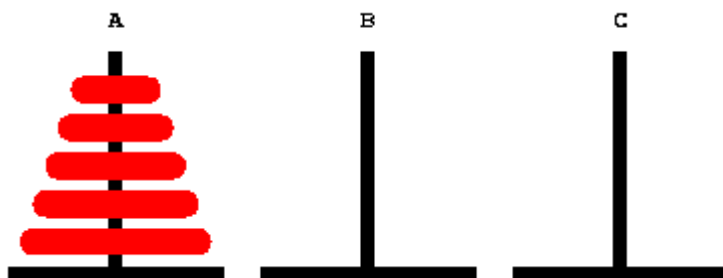
Faire des recherches sur cette méthode afin d'en déduire l'analyse récursive du traitement de tri afin d'écrire un sous-programme récursif correspondant.

### Exercice 1708 - 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 un tour ayant 5 disques. Ecrire un sous-programme récursif qui déplace la tour A emplacement vers l'emplacement B.



## 18 LISTES CHAINEES

### Exercice 1801 - Niveau 1

- Ecrire une structure de maillon qui a un réel et un champ vers le maillon.
- 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
- Ecrire un sous-programme récursif qui affiche tous les réels de la liste.
- Ecrire un programme qui insère 10 maillons dans une liste déclarée (chainage simple). Ce programme met ensuite en place les sous-programmes décrits ci-dessus.

### Exercice 1802 - Niveau 1

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

Vous pouvez ajouter une structure date de type `t_date` à définir.

Ce maillon aura un champ vers le maillon suivant.

- 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 les immatriculations des maillons à partir d'une marque saisie, passée en paramètre.
- Ecrire un sous-programme récursif qui affiche toutes les voitures de la liste.
- Ecrire un sous-programme qui reçoit une immatriculation en paramètre et supprime cette voiture de la liste.
- Ecrire un programme principal qui insère autant de maillons que l'utilisateur le souhaite, dans une liste déclarée. Ce programme met ensuite en place les sous-programmes décrits ci-dessus.

### Exercice 1803 - Niveau 2

Refaire l'exercice 2401, avec une liste doublement chaînée.

### Exercice 1804 - Niveau 2

Refaire l'exercice 2402, avec une liste doublement chaînée.

### Exercice 1805 - Niveau 2

Refaire les exercices 2401 et 2402, avec une liste circulaire.

## Exercice 1806 - Niveau 2

- Ecrire la définition de type (typedef) pour une structure contenant les informations relatives à un **étudiants** ECE.
- Ecrire la définition de type (typedef) pour une structure contenant les informations relatives à un **enseignant** ECE.  
Y'a-t-il des informations communes aux 2 types que vous venez de définir ?  
Auriez-vous pu organiser autrement vos définitions de type pour éviter les redondances ?
- Ecrire la définition de type (typedef) pour une structure permettant le chaînage des étudiants ECE comme définis précédemment.
- Ecrire une fonction de « comptage » qui reçoit l'ancre d'une liste chaînée **quelconque** (attention à la liste vide) et qui retourne le nombre de cellules dans la liste.
- Ecrire toutes les fonctions « classiques » qui permettent les manipulations des listes chaînées :
  - Ajout en tête
  - Ajout en queue
  - Suppression
  - Recherche
  - Modification
  - Etc.

(Intégrer la fonction de comptage de l'exercice précédent à cette librairie de fonctions)

**ECE**

PARIS ÉCOLE D'INGÉNIEURS

# ANNEXES

## ANNEXE A : SEQUENCES D'ECHAPPEMENT

\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

	<b>Classes de priorités :</b>	<b>Ordre de l'évaluation :</b>
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

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	ë	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	â	166	A6	à	198	C6	⌚	230	E6	μ
135	87	ç	167	A7	ó	199	C7	⌚	231	E7	Υ
136	88	ê	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	î	171	AB	½	203	CB	⌚	235	EB	δ
140	8C	ì	172	AC	¼	204	CC	⌚	236	EC	∞
141	8D	ï	173	AD	¡	205	CD	=	237	ED	Φ
142	8E	Ä	174	AE	«	206	CE	⌚	238	EE	ε
143	8F	Å	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	Œ	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	ƒ	187	BB	⌚	219	DB	⌚	251	FB	√
156	9C	£	188	BC	⌚	220	DC	⌚	252	FC	²
157	9D	¥	189	BD	⌚	221	DD	⌚	253	FD	²
158	9E	℔	190	BE	⌚	222	DE	⌚	254	FE	²
159	9F	ƒ	191	BF	⌚	223	DF	⌚	255	FF	²

# BIBLIOGRAPHIE

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

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

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