

ALGORITHMIQUE

ET

PROGRAMMATION EN LANGAGE C

~

**Exercices – Mini TP**

**2020-2021**

**ING1 - ING3 nouveaux - Prepac**

**Elisabeth Rendler**

Enseignant

**SOMMAIRE**

[1. DECOMPOSITION DE PROBLEME ET ALGORITHME 5](#_Toc42610564)

[Exercice 101 Instructions 5](#_Toc42610565)

[Exercice 102 DTI, ACD 5](#_Toc42610566)

[Exercice 103 DTI, algorithme 5](#_Toc42610567)

[Exercice 104 Boucle : Instructions 5](#_Toc42610568)

[Exercice 105 Boucle : DTI, algorithme 6](#_Toc42610569)

[Exercice 106 Boucle : DTI, ACD 6](#_Toc42610570)

[Exercice 107 Boucle : DTI, algorithme 6](#_Toc42610571)

[Exercice 108 Boucle et test : Instructions 6](#_Toc42610572)

[Exercice 109 Boucle et test : DTI, algorithme 7](#_Toc42610573)

[Exercice 110 Boucle et test : DTI et organigramme 7](#_Toc42610574)

[Exercice 111 Cas d’étude : ACD et organigramme 7](#_Toc42610575)

[Exercice 111 Cas d’étude : ACD et organigramme 7](#_Toc42610576)

[Exercice 112 Cas d’étude 8](#_Toc42610577)

[Exercice 113 Cas d’étude : boucle 8](#_Toc42610578)

[Exercice 114 Cas d’étude : boucle 8](#_Toc42610579)

[Exercice 115 Résultat d’algorithme 8](#_Toc42610580)

[Exercice 116 Résultat d’algorithme 9](#_Toc42610581)

[2. ORDINATEUR ET ENVIRONNEMENT DE TRAVAIL 10](#_Toc42610582)

[Exercice 201 Ordinateur 10](#_Toc42610583)

[Exercice 202 Arborescence des fichiers 10](#_Toc42610584)

[Exercice 203 Environnement de travail : CodeBlocks 10](#_Toc42610585)

[3. STOCKAGE DES DONNEES : VARIABLES DE TYPE SCALAIRE 11](#_Toc42610586)

[Exercice 301 Variables 11](#_Toc42610587)

[Exercice 302 Déclaration de variables 11](#_Toc42610588)

[Exercice 303 Affectation de variables 11](#_Toc42610589)

[Exercice 304 Modification de variables 11](#_Toc42610590)

[Exercice 305 Variables scalaires, permutation 12](#_Toc42610591)

[Exercice 306 Autres affectations 12](#_Toc42610592)

[Exercice 307 Cast de variables 12](#_Toc42610593)

[4. ENTREES/SORTIES 14](#_Toc42610594)

[Exercice 401 - Niveau 1 14](#_Toc42610595)

[Exercice 402 - Niveau 1 14](#_Toc42610596)

[Exercice 403- Niveau 2 14](#_Toc42610597)

[Exercice 404 - Niveau 2 14](#_Toc42610598)

[Exercice 405 – Niveau 2 14](#_Toc42610599)

[5. CALCULS, COMPARAISONS ET OPERATEURS LOGIQUES 17](#_Toc42610600)

[Exercice 501 - Niveau 1 17](#_Toc42610601)

[Exercice 502 - Niveau 1 17](#_Toc42610602)

[Exercice 503 - Niveau 2 17](#_Toc42610603)

[Exercice 504 - Niveau 2 18](#_Toc42610604)

[Exercice 505 - Niveau 2 18](#_Toc42610605)

[Exercice 506 - Niveau 2 19](#_Toc42610606)

[6. TESTS 20](#_Toc42610607)

[Exercice 601 - Niveau 1 20](#_Toc42610608)

[Exercice 602- Niveau 1 20](#_Toc42610609)

[Exercice 603Niveau 1 20](#_Toc42610610)

[Exercice 604 - Niveau 2 20](#_Toc42610611)

[Exercice 605 - Niveau 2 20](#_Toc42610612)

[7. BOUCLES DE REPETITION 21](#_Toc42610613)

[Exercice 701 - Niveau 1 21](#_Toc42610614)

[Exercice 702 - Niveau 1 21](#_Toc42610615)

[Exercice 703 - Niveau 2 21](#_Toc42610616)

[Exercice 704 - Niveau 2 22](#_Toc42610617)

[Exercice 705 - Niveau 2 22](#_Toc42610618)

[Exercice 706 - Niveau 3 22](#_Toc42610619)

[Exercice 707 - Niveau 3 22](#_Toc42610620)

[Exercice 708 - Niveau 3 22](#_Toc42610621)

[Exercice 709 - Niveau 3 23](#_Toc42610622)

[Exercice 710 - Niveau 3 23](#_Toc42610623)

[8. SOUS-PROGRAMMES ET PASSAGE DES PARAMETRES PAR VALEUR 24](#_Toc42610624)

[Exercice 801 - Niveau 1 24](#_Toc42610625)

[Exercice 802 - Niveau 2 24](#_Toc42610626)

[Exercice 803 - Niveau 2 24](#_Toc42610627)

[Exercice 804 - Niveau 2 25](#_Toc42610628)

[9. POINTEURS ET PASSAGE DES PARAMETRES PAR ADRESSE 26](#_Toc42610629)

[Exercice 901 - Niveau 1 26](#_Toc42610630)

[Exercice 902 - Niveau 1 26](#_Toc42610631)

[Exercice 903 - Niveau 1 27](#_Toc42610632)

[Exercice 904 - Niveau 1 27](#_Toc42610633)

[Exercice 905 - Niveau 2 27](#_Toc42610634)

[Exercice 907 - Niveau 2 28](#_Toc42610635)

[10. TABLEAUX 29](#_Toc42610636)

[Exercice 1001 - Niveau 1 29](#_Toc42610637)

[Exercice 1002 - Niveau 2 29](#_Toc42610638)

[Exercice 1003 - Niveau 2 29](#_Toc42610639)

[Exercice 1004 - Niveau 2 30](#_Toc42610640)

[Exercice 1005 - Niveau 2 30](#_Toc42610641)

[Exercice 1006 - Niveau 2 30](#_Toc42610642)

[Exercice 1007 - Niveau 2 30](#_Toc42610643)

[Exercice 1008 - Niveau 3 31](#_Toc42610644)

[11. CHAINES DE CARACTERES 32](#_Toc42610645)

[Exercice 1101 - Niveau 1 32](#_Toc42610646)

[Exercice 1102 - Niveau 1 32](#_Toc42610647)

[Exercice 1103 - Niveau 1 32](#_Toc42610648)

[Exercice 1104 - Niveau 2 32](#_Toc42610649)

[Exercice 1105 - Niveau 2 33](#_Toc42610650)

[Exercice 1106 - Niveau 2 33](#_Toc42610651)

[Exercice 1107 - Niveau 3 34](#_Toc42610652)

[Exercice 1108 - Niveau 3 34](#_Toc42610653)

[12. GENERATION DE NOMBRES ALEATOIRES 35](#_Toc42610654)

[Exercice 1201 - Niveau 1 35](#_Toc42610655)

[Exercice 1202 - Niveau 1 35](#_Toc42610656)

[Exercice 1203 - Niveau 2 35](#_Toc42610657)

[Exercice 1204 - Niveau 2 36](#_Toc42610658)

[TP : niveau 2 37](#_Toc42610659)

[13. STRUCTURES 38](#_Toc42610660)

[Exercice 1301 - Niveau 1 38](#_Toc42610661)

[Exercice 1302 - Niveau 1 38](#_Toc42610662)

[Exercice 1303 - Niveau 2 38](#_Toc42610663)

[Exercice 1303 - Niveau 2 39](#_Toc42610664)

[Exercice 1304 - Niveau 3 39](#_Toc42610665)

[14. FICHIERS : ASCII ET BINAIRES 40](#_Toc42610666)

[Exercice 1401 - Niveau 1 40](#_Toc42610667)

[Exercice 1402 - Niveau 1 40](#_Toc42610668)

[Exercice 1403 - Niveau 2 40](#_Toc42610669)

[Exercice 1404 - Niveau 2 40](#_Toc42610670)

[Exercice 1405 - Niveau 3 40](#_Toc42610671)

[Exercice 1406 - Niveau 3 41](#_Toc42610672)

[15. PROGRAMMATION EVENEMENTIELLE 42](#_Toc42610673)

[Exercice 1501 - Niveau 1 42](#_Toc42610674)

[Exercice 1502 - Niveau 1 42](#_Toc42610675)

[Exercice 1503 - Niveau 1 42](#_Toc42610676)

[Exercice 1504 - Niveau 2 42](#_Toc42610677)

[Exercice 1505 - Niveau 2 42](#_Toc42610678)

[Exercice 1506 - Niveau 2 43](#_Toc42610679)

[Exercice 1507 - Niveau 2 43](#_Toc42610680)

[Exercice 1508 - Niveau 3 44](#_Toc42610681)

[Exercice 1509 - Niveau 3 44](#_Toc42610682)

[Exercice 1510 - Niveau 3 44](#_Toc42610683)

[16. ALLOCATION DYNAMIQUE DE MEMOIRE 45](#_Toc42610684)

[Exercice 1601 - Niveau 1 45](#_Toc42610685)

[Exercice 1602 - Niveau 1 45](#_Toc42610686)

[Exercice 1603 - Niveau 2 45](#_Toc42610687)

[Exercice 1604 - Niveau 1 45](#_Toc42610688)

[Exercice 1605 - Niveau 2 46](#_Toc42610689)

[Exercice 1606 - Niveau 2 46](#_Toc42610690)

[Exercice 1607 - Niveau 2 46](#_Toc42610691)

[TP : niveau 2 47](#_Toc42610692)

[17. RECURSIVITE 48](#_Toc42610693)

[Exercice 1701 - Niveau 1 48](#_Toc42610694)

[Exercice 1702 - Niveau 2 48](#_Toc42610695)

[Exercice 1703 - Niveau 2 48](#_Toc42610696)

[Exercice 1704 - Niveau 2 48](#_Toc42610697)

[Exercice 1705 - Niveau 2 48](#_Toc42610698)

[Exercice 1706 - Niveau 2 48](#_Toc42610699)

[Exercice 1707 - Niveau 3 49](#_Toc42610700)

[Exercice 1708 - Niveau 3 49](#_Toc42610701)

[18. LISTES CHAINEES 50](#_Toc42610702)

[Exercice 1801 - Niveau 1 50](#_Toc42610703)

[Exercice 1802 - Niveau 1 50](#_Toc42610704)

[ANNEXES 53](#_Toc42610705)

[ANNEXE A : SEQUENCES D'ECHAPPEMENT 53](#_Toc42610706)

[ANNEXE B : PRIORITES DES OPERATEURS 53](#_Toc42610707)

[ANNEXE C : CLASSIFICATION, CONVERSION DE CARACTERE : <CTYPE.H> 54](#_Toc42610708)

[ANNEXE D : TRAITEMENT DE CHAINES DE CARACTERES : <STRING.H> 55](#_Toc42610709)

[ANNEXE E : TABLE ASCII ET TABLE ASCII ETENDUE 56](#_Toc42610710)

[BIBLIOGRAPHIE 58](#_Toc42610711)

## 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 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*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **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 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
* affichage le nombre de pièces (1cent, 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

### 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.

Les contraintes sont :

* interdiction d’utiliser la formule mathématique ∑
* 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 111 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 112 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 113 Cas d’étude : boucle

Vous devez afficher les n nombres naturels. 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 114 Cas d’étude : boucle

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

U0 = -16

Un = Un-1 / 2 + 4

Les contraintes sont :

* interdiction d’utiliser la formule mathématique
* lecture de l’entier n

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

### Exercice 115 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 116 Résultat d’algorithme

Vous avez cet algorithme. Que fait cet algorithme ?

Début

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

Fin

## 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
* 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 connait-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’

## 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 reél ou flottant
  + **initiale** de type caractère
* Ajouter 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…

## 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.
* Afficher l’inverse 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 (nommée *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, charactère et réel.

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

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 = a

ddk (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 ;  **}** |

## CALCULS, COMPARAISONS ET OPERATEURS LOGIQUES

### Exercice 501 - Niveau 1

* 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 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 :

* des notes littéraires : langue vivante (Nlv), 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 :

* 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. Vous vérifiez que la valeur de toutes les notes saisies 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) | b ( 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 ces codes sous Codeblocks pour vérifier vos résultats.

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

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

## 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 :

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 603Niveau 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 (sans entrer de valeur « en dur » (genre 32)).

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* **;**

## BOUCLES DE REPETITION

### Exercice 701 - Niveau 1

* Ecrire l’algorithme et le programme qui affiche à l’écran 10 fois la même chaine « Bienvenue en TP : indice x ». Vous ajouterez à la fin de cette chaine 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’analyse 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  -12  -9  -6  -3  0  3  6  9  12  15 | -9  -6  -3  0  3  6  9 |

### 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 demandés, l’utilisateur demande explicitement de quitter le programme. 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 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’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.

### 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.

### 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

### 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.

Exemple : hauteur = 4

\*\*\*\*\*\*\*

\*\*\*\*\*

\* \*\*\*\* \*\*\*

\*\*\* \*\*\*\* \*

\*\*\*\*\* \*\*\*\* \*

\*\*\*\*\*\*\* \*\*\*\* \*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

## 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)
  + affiche les résultats obtenus

### 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)
  + affiche les résultats obtenus

### Exercice 802 - Niveau 2

* Ecrire l’algorithme et le sous-programme d’un 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 l’algorithme 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

### 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.
* 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 la fonction ***sleep()*** de la bibliothèque *windows.h* ou bien une boucle qui n’incrémente 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** |

## POINTEURS ET PASSAGE DES PARAMETRES PAR ADRESSE

### Exercice 901 - Niveau 1

Soit le code suivant :

int\* pt ;

int val1, val2, val3 ;

val1 = 10 ;

val2 = 20 ;

val3 = 1 ;

pt = &val ;

\*pt = 100 ;

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

Que représente les instructions suivantes :

* \*p +3
* p+2
* \*p += 5 ;
* (\*p)++ ;
* ++\*p ;
* \*(p+2)++ ;
* \*p+=val3 ;
* \*(p+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 passé en paramètre par adresse
* 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 ? 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 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’algorithme 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

### Exercice 905 - Niveau 2

* Ecrire l’algorithme 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’algorithme et le programme principal qui :
  + appelle le sous-programme précédent
  + affiche tous les résultats rendus
* 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 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 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

Remarque :

Toutes les saisies sont à blinder.

### 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 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’algorithme et un sous-programme qui calcule le prix pour un groupe de personnes de la même tranche d’âge.

Vous saisirez 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

## TABLEAUX

### Exercice 1001 - Niveau 1

Ecrire l’algorithme 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’algorithme 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.
* Ecrire le sous-programme permettant l’affichage d’un tableau de 10 notes passé en paramètre.
* Ecrire un sous - programme permettant :
  + la saisie et l’affichage de 10 notes stockées dans un tableau en appelant les sous-programme précédent
  + le calcul de la moyenne de ces 10 notes

Quelle serait la conséquence d’un affichage de cette moyenne dans le programme principal ?

* 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 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 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.

La taille du tableau est 20.

### 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 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 non
* Ecrire le programme principal qui met en place tous ces sous-programmes.

La taille du tableau est 10.

### Exercice 1005 - Niveau 2

* Ecrire le sous-programme qui saisisse 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 1006 - Niveau 2

* 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 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 un sous-programme qui :
  + saisit un caractère (alphabétique, numérique ou autres)
  + compte le nombre d’occurrences (= le nombre de fois où apparait un élément) de ce caractère saisi. Le résultat sera rendu au programme appelant et le caractère saisi passera en paramètre par adresse
* 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

* 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 |

## CHAINES DE CARACTERES

### Exercice 1101 - Niveau 1

Ecrire l’algorithme et le programme qui :

* saisit deux chaines de caractères
* compte leur nombre de caractères respectifs
* affiche les chaines et leur taille calculée

Remarque :

La première chaine de caractères ne doit pas accepter les espaces. Elle n’est constituée que d’un seul mot.

La deuxième chaine 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 chaines
* compare alphabétiquement ces deux chaines
* affiche le résultat de la comparaison : par exemple : « les chaines sont identiques » ou « la 1er chaine est avant la 2ieme chaine dans l’ordre alphabétique » ou le contraire.

### Exercice 1103 - Niveau 1

* Ecrire le sous-programme qui saisit une chaine 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 chaine de caractère passée en paramètre. Les résultats seront rendus au programme appelant.
* Ecrire l’algorithme et le sous-programme qui compte le nombre de mots (ensembles de caractères séparé par un espace) stockés dans la chaine passée en paramètre. Rendre ce résultat à l’appelant
* Ecrire le programme qui :
  + appelle les sous-programmes
  + affiche la chaine ainsi que les résultats obtenus

### Exercice 1104 - Niveau 2

Ecrire l’algorithme 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 chaine 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. 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 2

* Ecrire l’algorithme et le sous-programme qui remplit un tableau de chaine de caractères ayant 10 places. Ce tableau est passé en paramètre
* Ecrire un sous-programme qui recherche un mot dans le tableau de chaine 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.

### Exercice 1106 - Niveau 2

Ecrire l’algorithme 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 chaines 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 :

1er phrase passée en paramètre : « il fait beau »

2ieme phrase ‘miroir‘ : « uaeb tiaf li » phrase 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’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’algorithme et le sous-programme qui construit une chaine de caractère avec les deux premiers caractères suivis des deux derniers caractères d’une chaine passée en paramètre. La chaine contractée construite doit être renvoyée au programme appelant.

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

Cas 1

|  |  |
| --- | --- |
| chaine passée en paramètre | : «ordinateur » |
| chaine contractée    Cas 2 : | : « orur » |
| chaine passée en paramètre | : «ode » |
| chaine contractée | : « » |
| message affiché | : « chaine non valable » |

* Ecrire l’algorithme et le sous-programme qui saisit une chaine et vérifie si cette chaine existe dans une première chaine 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 chaine. Les résultats reçus seront affichés.

### Exercice 1108 - Niveau 3

* Ecrire le sous-programme qui saisit un verbe du 1er groupe de conjugaison (terminaison « –er »). Ce verbe est rendu au programme appelant.
* Ecrire l’algorithme et le sous-programme qui affiche la conjugaison au présent du verbe passer en paramètre.
* 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**

## 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’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

### Exercice 1202 - Niveau 1

* Ecrire le sous-programme qui remplit un tableau d’entiers aléatoires (20 places). Dans le tableau toutes les valeurs n’apparaitront 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 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 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

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 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.

## 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 1

* Définir une structure contenant les champs suivants :
  + Une chaine de 20 caractères
  + Un tableau de 15 entiers
* Ecrire un sous-programme qui remplit les champs de la structure passée en paramètre
* 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 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 1303 - Niveau 2

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

Modifier votre programme principal en :

* initialisant votre tableau de pointeurs avant tout traitement.
* écrivant une boucle de traitement dont on ne sortira qu’à la demande de l’utilisateur. Les fonctionnalités proposées le seront à travers un menu affiché à chaque tour de boucle
* écrivant un sous-programme menu proposant l’ajout, la suppression d’une structure et l’affichage du tableau. Ces fonctionnalités sont implémentées dans un sous-programme

L’ajout d’une nouvelle structure se fera quand l’utilisateur le demandera. Le tableau n’est pas rempli en une seule fois

### Exercice 1304 - Niveau 3

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

* + La désignation
  + La catégorie
  + La date de péremption (de type **t\_date** défini plus bas)
  + La quantité en stock
  + Le prix de vente

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

* + Le jour
  + Le mois
  + L’année
* 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
* 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 10 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 articles de cette catégorie
    - Saisit une date et appelle le sous-programme qui compte le nombre d’articles ayant dépassé cette date

## 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’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 ‘mutl5.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

Ecrire le programme qui :

* saisit 10 chaines de caractères
* stocke ces chaines dans un fichier nommé *stockage1.tx.* Chaque chaine saisie se trouve sur une ligne du fichier
* après la fermeture du 1er fichier, procède à la recopie de son contenu dans un autre fichier nommé *stockage2.txt*

### 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 la fonction de saisie de tous les champs d’une instance t\_film.
* Ecrire un sous-programme qui sauvegarde leséléments d’un tableau de structures 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. Toutes les données lues seront stockées dans un tableau de structures passé en paramètre
* Ecrire un sous-programme qui affiche tous les champs des éléments du tableau passé en paramètre

* Ecrire le programme principal qui :
  + définit un tableau de 10 structures.
  + saisit le nom du fichier de sauvegarde ( de type fichier texte)
  + charge les films du fichier dans le tableau
  + affiche les films du tableau
  + ajoute deux nouveaux films dans le tableau en utilisant le sous-programme de saisie à la place de ceux supprimés
  + sauvegarde tous les films dans le fichier de sauvegarde

Remarque :

Les sous-programmes écrits précédemment seront tous utilisés

### Exercice 1406 - Niveau 3

* Définir le structure date :
  + jour : entier
  + mois : entier
  + année : entier
* Définir la structure livre suivante :
  + identifiant : entier
  + titre : chaine de caractères
  + auteurs : tableau de chaines de caractères (5 auteurs possibles dans le tableau)
  + thème : tableau de chaines 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 qui 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.

## 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 disparait.

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 passé 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
* accélèreront 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 disparaisse 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](http://code-reference.com/c/datatypes/int) main() {

[textcolor(](http://code-reference.com/c/conio.h/textcolor)BLUE+BLINK);

[cprintf(](http://code-reference.com/c/conio.h/cprintf)"textcolor c example with a blue blinking text"); [getch(](http://code-reference.com/c/conio.h/getch)); [return](http://code-reference.com/c/control_structures/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.**

## 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 chaine 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 chaines 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 1

* Ecrire un programme qui saisit la taille d’un tableau de réels (tableau à une dimension) et alloue le tableau dynamiquement.
* 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

### Exercice 1605 - 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 1er sous-programme. Remplir le tableau aléatoirement avec des valeurs comprises entre -25 et 150. Sauvegarder les valeurs de ce tableau dans un fichier *résultat.txt*.

### Exercice 1606 - 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 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 1607 - Niveau 2

* Définir une structure de maillon dinosaure :
  + Nom (chaine de 15 caractères utiles)
  + Famille (chaine 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 : Gestion de rendez-vous de consultations médicales**

### 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 compléterez 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

* 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 pour :

* + 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 de la pertinence 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).

## RECURSIVITE

### Exercice 1701 - Niveau 1

* Ecrire l’algorithme et le sous-programme récursif qui saisit un tableau de 20 réels.
* Ecrire l’algorithme 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 calcule la somme des valeurs du tableau dynamique d’entiers reçu en paramètre.

Le résultat sera affiché dans le programme principal.

Le tableau aura été alloué et rempli dans le programme principal.

### 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 :

U0 = 7

Un = 3\*Un-1 +5/2

Mettre en place le 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 le code des sous-programmes permettant de trier un tableau de flottants de dimension n (n entier constant).

* Le sous-programme « saisie » **récursif** permettra de saisir n flottants
* Le sous-programme « affiche », **récursif** **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 » **récursif** 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 recombiner 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.

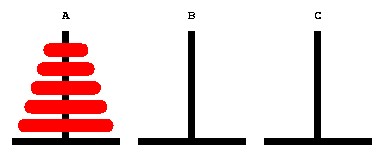
Faire des recherches sur cette méthode afin d’en déduire l’analyse récursive du traitement de tri afin d’écrire un sous-programme récursif correspondant.

### Exercice 1708 - Niveau 3

Les **tours de Hanoï** sont un [jeu de réflexion](https://fr.wikipedia.org/wiki/Jeu_de_r%C3%A9flexion) imaginé par le [mathématicien](https://fr.wikipedia.org/wiki/Math%C3%A9maticien) français [Édouard Lucas,](https://fr.wikipedia.org/wiki/%C3%89douard_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.



## LISTES CHAINEES

### Exercice 1801 - Niveau 1

* Ecrire une structure de maillon qui a un réel et un champ vers le maillon suivant.
* Ecrire un sous-programme qui créé 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.
* 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 voiture qui contient :
  + un numéro d’immatriculation (chaine)
  + une marque (chaine)
  + un modèle(chaine)
  + 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.

* Ecrire un sous-programme qui créé 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.
* 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. Le programme prendra fin à la demande de l’utilisateur.

**Exercice 1803 - Niveau 2**

* Ecrire une structure de maillon d’**une liste doublement chainé** (lien vers le précédent et lien vers le suivant). Le maillon comporte un champ de type caractère.
* Ecrire un sous-programme qui créé 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 chaine
* Ecrire un sous-programme qui supprime tous les maillons dont la valeur du caractère est passée en paramètre
* Ecrire un programme principal qui met en place ces fonctionnalités par un menu. La sortie du programme sera demandée par l’utilisateur

**Exercice 1804 - Niveau 2**

* Ecrire une structure de maillon d’**une liste chainée circulaire**. Le maillon comporte un champ de type entier.
* Ecrire un sous-programme qui créé 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 chaine
* Ecrire un sous-programme qui compte le nombre de maillon de la chaine
* Ecrire un programme principal qui met en place ces fonctionnalités par un menu. La sortie du programme sera demandée par l’utilisateur

**Exercice 1805 - Niveau 3**

* Ecrire la définition de type (typedef) pour une structure contenant les informations relatives à un **étudiants** ECE :
  + Numéro (entier)
  + Nom (chaine)
  + Prénom (chaine)
  + Adresse (chaine)
  + Promotion (entier)
  + Groupe (entier)
* Ecrire la définition de type (typedef) pour une structure contenant les informations relatives à un **enseignant** ECE :
  + Numéro (entier)
  + Nom (chaine)
  + Prénom (chaine)
  + Adresse (chaine)
  + Département (chaine)
  + Tableau d’entier (à deux dimensions) indiquant la promotion d’étudiants et le nombre d’heures d’intervention correspondant

Y’a-t-il des informations communes aux 2 types que vous venez de définir ?

Est-ce qu’il est possible d’organiser autrement vos définitions de type pour éviter les redondances ?

* Ecrire la définition de type (typedef) pour une structure maillon.
* Ecrire une fonction de « comptage » 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 fonctions qui permettent les manipulations des listes chaînées :
  + Ajout d’un maillon (l’ajout conservera les maillons triés
  + Suppression
  + Recherche
  + Modification
  + Etc.
* Ecrire le programme principal mettant en jeu toutes ces fonctionnalités

**Exercice 1806 - Niveau 3**

* Ecrire une définition d’une structure maillon (**chainage simple**) contenant les données d’un article :
  + Numéro (entier)
  + Libellé (chaine)
  + Catégorie(chaine)
  + Fournisseur(chaine)
  + Prix d’achat (réel)
  + Prix de vente(réel)
  + Quantité en stock (entier)
* Ecrire un sous-programme qui créé 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 liste chainée. La liste chainée qu’il intègrera sera celle de sa catégorie. Le nouveau maillon et le tableau sont passé en paramètre
* Ecrire un sous-programme qui affiche chaque liste chainée 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 chainées d’articles. Les listes chainées seront de chainage simple.

# 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

|  |  |  |
| --- | --- | --- |
|  | ***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 \*CH1, 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 \*CH1) 8.6.2.** ajoute *CH2* à la fin de *CH1*; retourne *CH1*

**char \*strncat(char \*CH1, const char \*CH1, 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 \*CH1)**

8.5. / 8.6.2.

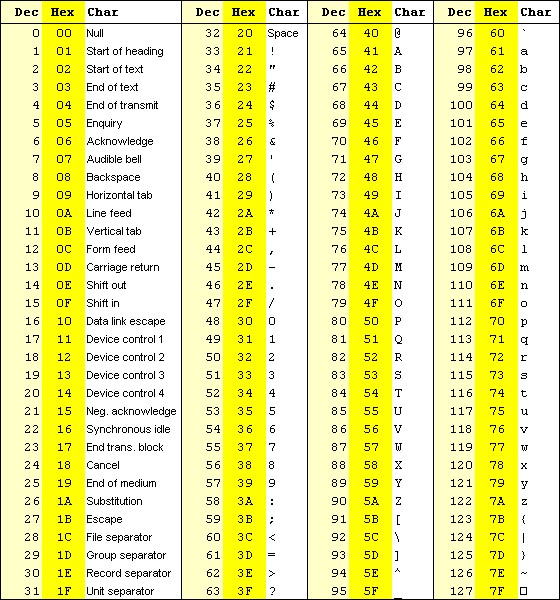
compare *CH1* et *CH2* lexicographiquement et fournit un résultat:

négatif si *CH1* précède *CH2* zéro si *CH1* est égal à *CH2*

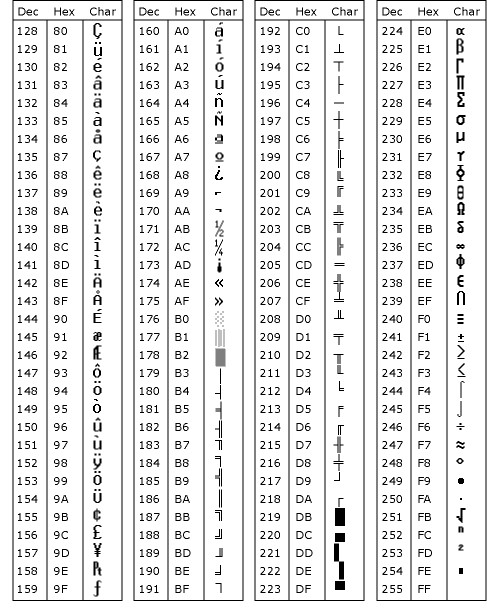
positif si *CH1* suit *CH2*

### ANNEXE E : TABLE ASCII ET TABLE ASCII ETENDUE

La table ASCII traduit en caractères les 128 premiers codes.



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



# BIBLIOGRAPHIE

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

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

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