NOTIONS DE BASE

Exercice 1 (/fr/exercices-c/notions-de-base/exercice-1): Ecrire un programme C qui permet d'afficher le message suivant: *Bonjour*.

Exercice 2 (/fr/exercices-c/notions-de-base/exercice-2): Ecrire un programme C qui permet de lire en entrée un entier constitué de trois chiffres et d'afficher celui-ci inversé. Exemple: si l'entrée est *123* on affiche *321*.

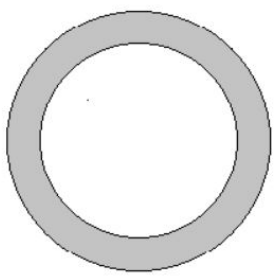
Exercice 3 (/fr/exercices-c/notions-de-base/exercice-3): Ecrire un programme C qui permet de lire deux nombres réels, et d'afficher ensuite leur produit, avec une précision de trois chiffres après la virgule.

Exercice 4 (/fr/exercices-c/notions-de-base/exercice-4): Ecrire un programme C qui permet de permuter le contenu de deux variables entières en passant par une troisième variable auxiliaire. Ceci et en affichant les deux variables avant et après permutation.

Exercice 5 (/fr/exercices-c/notions-de-base/exercice-5): Ecrire un programme C qui lit en entrée trois entiers et affiche leur moyenne avec une précision de deux chiffres après la virgule.

Exercice 6 (/fr/exercices-c/notions-de-base/exercice-6): Ecrire un programme C qui lit en entrée un caractère alphabétique entre *a* et *y*, qui peut être soit une majuscule ou une minuscule. Et affiche la lettre qui vient juste après lui dans l'ordre alphabétique.

Exercice 7 (/fr/exercices-c/notions-de-base/exercice-7): Ecrire un programme C qui lit deux réels R_1 et R_2 qui représentent les rayons de deux cercles concentriques, et renvoie ensuite l'aire de la surface comprise entre les deux cercles (surface grise).
Remarque: R_1 peut être supérieur à R_2, comme il peut lui être inférieur.



Exercice 8 (/fr/exercices-c/notions-de-base/exercice-8): Ecrire un programme C qui lit deux entiers est affiche le plus grand d'entre eux.

Exercice 9 (/fr/exercices-c/notions-de-base/exercice-9): Ecrire un programme C qui trouve pour un réel les deux carrés parfaits les plus proches qui l'encadrent.

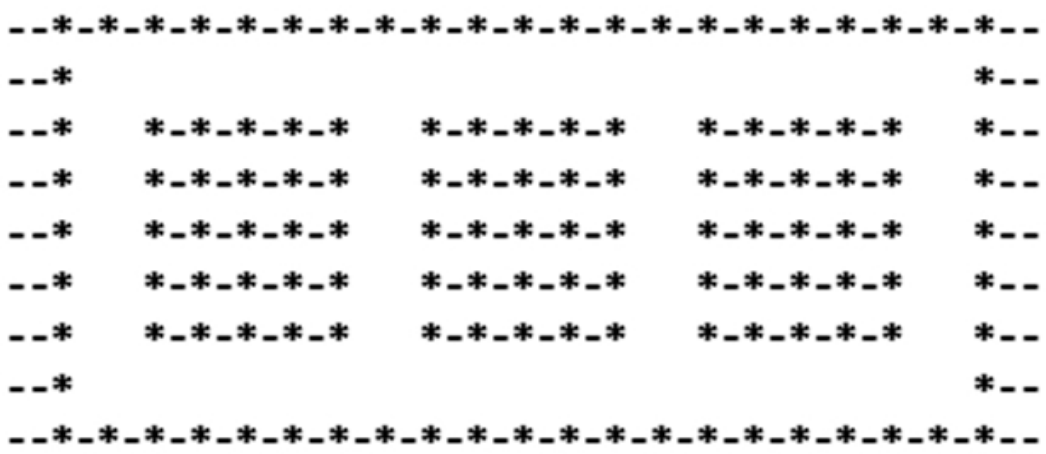
On rappelle qu'un carré parfait est un entier dont la racine carrée est aussi un entier. Exemple: *9 = 3x3* et *4 = 2x2* sont des carrés parfaits ; or, *5* ne l'est pas.

Exercice 10 (/fr/exercices-c/notions-de-base/exercice-10): Ecrire un programme C qui lit une fraction au format *a/b* où *a* et *b* sont deux entiers, et donne son équivalent décimal avec une précision de quatre chiffres après la virgule.

Ex: si l'utilisateur entre *3/2*, le programme doit afficher: *3/2 = 1.5000*

Exercice 11 (/fr/exercices-c/notions-de-base/exercice-11): Ecrire un programme C qui lit un entier V représentant un volume en litres. Puis calcule le nombre de canettes de 33cl que peut remplir en entier une citerne contenant un volume V de soda.

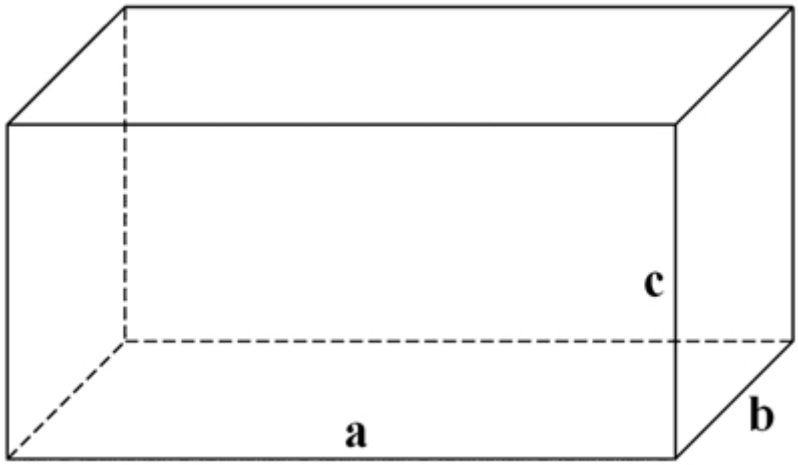
Exercice 12 (/fr/exercices-c/notions-de-base/exercice-12): Ecrire un programme C qui permet de tracer la forme suivante :



Exercice 13 (/fr/exercices-c/notions-de-base/exercice-13): Ecrire un programme C qui lit un nombre réel et affiche sa partie fractionnaire.

Exemple : la partie fractionnaire de 3.09 est 0.09.

Exercice 14 (/fr/exercices-c/notions-de-base/exercice-14): Ecrire un programme C qui lit les dimensions (nombres entiers) a, b et c d'un parallélépipède rectangle puis calcule et affiche sa superficie.



Exercice 15 (/fr/exercices-c/notions-de-base/exercice-15): Ecrire un programme C qui lit un entier n. Puis affiche les trois entiers impairs qui le suivent.

Exemple : pour 5 on affiche 7 9 11, et pour 2 on affiche 3 5 7.

CONDITIONS

Exercice 1 (/fr/exercices-c/conditions/exercice-1): Ecrire un programme C qui permet de dire si un entier est pair ou impair.

Exercice 2 (/fr/exercices-c/conditions/exercice-2): Ecrire un programme C qui permet de comparer deux entiers *a* et *b*, et d'afficher selon le cas l'un des messages suivants: *a=b*, *a>b* ou *a<b*.

Exercice 3 (/fr/exercices-c/conditions/exercice-3): Ecrire un programme C qui lit trois entiers pour les afficher ensuite dans un ordre croissant.

Exercice 4 (/fr/exercices-c/conditions/exercice-4): Ecrire un programme C qui lit un nombre réel et détermine s'il est entier ou non.

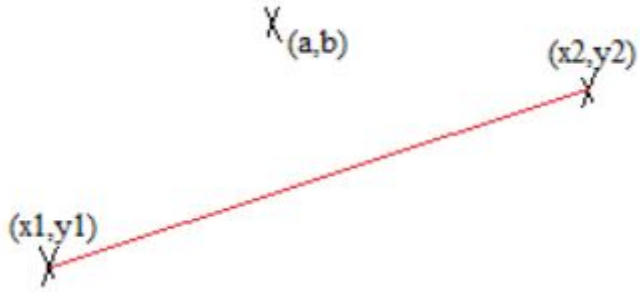
Exercice 5 (/fr/exercices-c/conditions/exercice-5): Ecrire un programme C qui lit un caractère et détermine s'il fait partie des alphabets ou non. Et s'il l'est, dire en plus s'il est une minuscule ou une majuscule.

Exercice 6 (/fr/exercices-c/conditions/exercice-6): Ecrire un programme C qui lit une date au format *15/09/2012* et l'affiche sous le format suivant: *15-Septembre-2012*.

Exercice 7 (/fr/exercices-c/conditions/exercice-7): Ecrire un programme C qui lit un entier et dit s'il est un carré parfait ou non.

On rappelle qu'un entier est carré parfait, si sa racine carrée est entière. Ex: les nombre *1* (*1x1*), *4* (*2x2*) et *9* (*3x3*) sont tous des carrés parfaits.

Exercice 8 (/fr/exercices-c/conditions/exercice-8): Ecrire un programme C qui lit les coordonnées des deux extrémités d'un segment, et lit ensuite les coordonnées d'un point dans le plan et dit si ce dernier se trouve ou non sur le segment.



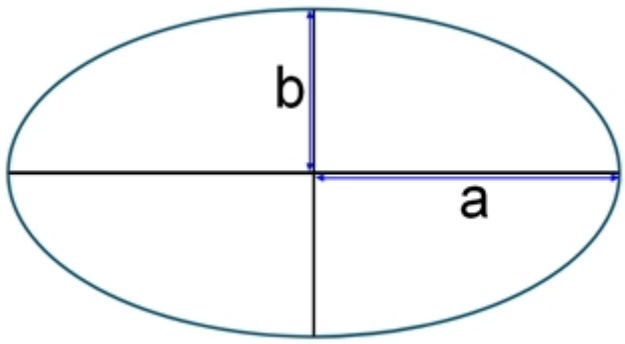
Exercice 9 (/fr/exercices-c/conditions/exercice-9): Ecrire un programme C qui lit deux instants dans le format *HH:MM:SS*, et affiche un des messages suivants:

- Le premier instant vient avant le deuxième;
- Le deuxième instant vient avant le premier;
- Il s'agit du même instant.

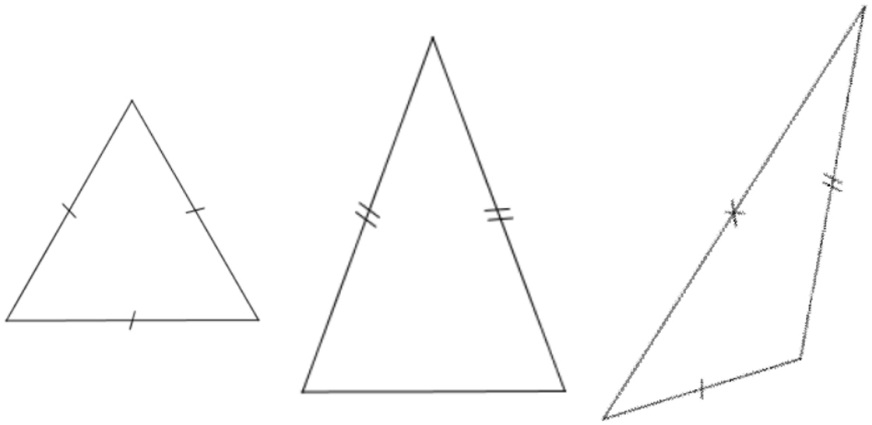
Exercice 10 (/fr/exercices-c/conditions/exercice-10): Ecrire un programme C qui affiche d'une manière aléatoire un des jours de la semaine.

Exercice 11 (/fr/exercices-c/conditions/exercice-11): Ecrire un programme C qui lit un nombre impair n et confirme que 32 divise $(n^2+3)(n^2+7)$.

Exercice 12 (/fr/exercices-c/conditions/exercice-12): Ecrire un programme C qui lit deux entiers représentant le grand rayon a et le petit rayon b d'une ellipse. Puis détermine si oui ou non l'aire de cette ellipse est supérieure ou égale à 100.



Exercice 13 (/fr/exercices-c/conditions/exercice-13): Ecrire un programme C qui lit trois entiers qui représentent les longueurs des côtés d'un triangle, et permet de dire s'il s'agit d'un triangle équilatéral, isocèle ou scalène.



Equilatéral **Isocèle** **Scalène**

Exercice 14 (/fr/exercices-c/conditions/exercice-14): Ecrire un programme C qui lit un caractère et détermine ensuite s'il est un chiffre, une lettre de l'alphabet ou un autre type de caractères.

Exercice 15 (/fr/exercices-c/conditions/exercice-15): Ecrire un programme C qui permet de lire deux nombres réels a et b, puis un entier n et afficher le résultat de la division de a par b avec une précision de n ($0 \leq n \leq 3$) chiffres après la virgule.

BOUCLES

Exercice 1 (/fr/exercices-c/boucles/exercice-1): Ecrire un programme C qui définit un nombre magique (un nombre secret), et lit des entiers à l'entrée jusqu'à ce que l'utilisateur trouve ce nombre. En lui indiquant à chaque fois s'il est en dessus ou au-dessous du nombre magique.

Exercice 2 (/fr/exercices-c/boucles/exercice-2): Ecrire un programme C qui lit un entier puis détermine s'il est premier ou non.
On rappelle qu'un entier est dit premier s'il a exactement deux diviseurs différents; *1* et lui-même.
Ex: *2, 3, 7, 17, 101* sont tous premiers, et *4, 10, 27* ne le sont pas.

Exercice 3 (/fr/exercices-c/boucles/exercice-3): Ecrire un programme C qui lit une série d'entiers positifs inférieurs à *100* terminée par *0*. Et qui doit négliger toute entrée strictement supérieure à *100*. Puis calcule et affiche la somme et le max des éléments de cette série.

Exercice 4 (/fr/exercices-c/boucles/exercice-4): Ecrire un programme C qui lit un entier et l'affiche inversé. On choisira de ne pas afficher chiffre par chiffre mais de construire l'entier inversé puis l'afficher.

Ex: si l'entrée est *12345* on doit afficher l'entier *54321*.

Exercice 5 (/fr/exercices-c/boucles/exercice-5): Ecrire un programme C qui lit un entier puis affiche tous les nombres premiers qui lui sont inférieurs.

Exercice 6 (/fr/exercices-c/boucles/exercice-6): Ecrire un programme C qui calcule le $n^{ième}$ terme de la suite de Fibonacci, définie comme suit: $U_n=U_{n-1}+U_{n-2}$ où $U_1=U_0=1$.

Exercice 7 (/fr/exercices-c/boucles/exercice-7): Ecrire un programme C qui utilise le principe de dichotomie pour trouver la solution de l'équation $x^3+12x^2+1=0$ dans l'intervalle $[-15,-10]$ avec une précision de $0,00001$.

TABLEAUX

Exercice 1 (/fr/exercices-c/tableaux/exercice-1): Ecrire un programme C qui lit un entier *n*, puis *n* autres entiers positifs dans un tableau, l'affiche puis calcul la somme, le max, et le min de ses éléments.

Exercice 2 (/fr/exercices-c/tableaux/exercice-2): Ecrire un programme C qui lit un entier *n*, puis *n* autres éléments dans un tableau. Et, affiche ce dernier avant et après la suppression de toutes les occurrences d'un nombre entré par l'utilisateur.

Exercice 3 (/fr/exercices-c/tableaux/exercice-3): Ecrire un programme C qui lit un entier *n*. Puis *n* autres entiers inférieurs à *100*, dans un tableau. Et affiche le nombre d'occurrences de chaque élément du tableau de la façon suivante:

Si le tableau est: 1 2 5 2 1 2, on affiche:

1 est répété 2 fois.
2 est répété 3 fois.
5 est répété 1 fois.

Pas nécessairement dans un ordre précis, mais chaque élément ne doit être cité qu'une seule fois.

Exercice 4 (/fr/exercices-c/tableaux/exercice-4): Ecrire un programme C qui construit la table de multiplication des entiers entre 1 et 9 puis l'affiche.

Exercice 5 (/fr/exercices-c/tableaux/exercice-5): Ecrire un programme C qui lit un entier *n* inférieur à *10*, et une matrice (tableau à deux dimensions) carrée *nxn*. Puis l'affiche et vérifie si tous les entiers entre *1* et n^2 y sont présents ou non.

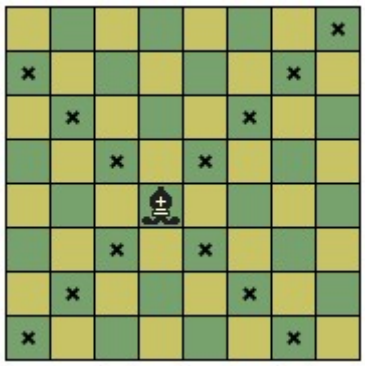
Ex: la matrice *3x3* suivante vérifie cette contrainte.

2 5 3
1 9 6
7 4 8

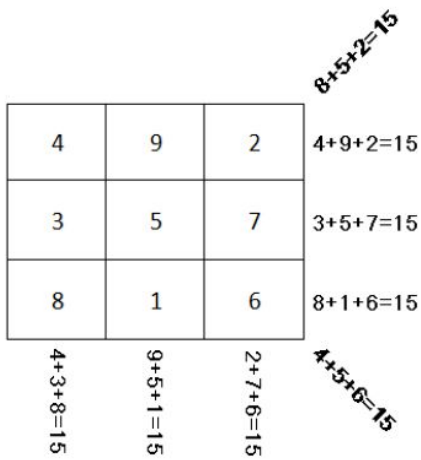
Exercice 6 (/fr/exercices-c/tableaux/exercice-6): Ecrire un programme C qui lit un tableau de taille *nxm*, et cherche tous les points-col qui s'y trouvent.

Un point-col est un élément qui est max sa ligne et min sur sa colonne.

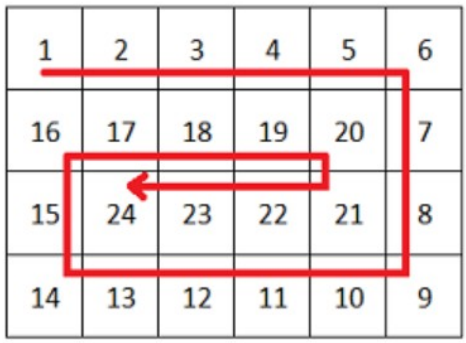
Exercice 7 (/fr/exercices-c/tableaux/exercice-7): Ecrire un programme C qui lit une position *(i,j)* d'un tableau *8x8*, rempli par des zéros. Et marque toutes les cases des deux diagonales qui passent par cette position en y mettant des uns.



Exercice 8 (/fr/exercices-c/tableaux/exercice-8): Ecrire un programme C qui lit un tableau *nxn*, et vérifie s'il est magique ou non. Un tableau est dit magique, si la somme des éléments de chaque ligne, de chaque colonne et de chaque une des deux diagonales est la même. En plus il doit contenir tous les entiers entre 1 et n^2.
L'exemple suivant montre un carré magique de dimension *3x3* :



Exercice 9 (/fr/exercices-c/tableaux/exercice-9): Ecrire un programme C qui lit les dimensions *n* et *m* d'un tableau à deux dimensions, le remplit d'une manière spirale en utilisant les entiers de l'intervalle *1* à *n*m* comme indiqué sur le tableau ci-dessous:



CHAÎNES DE CARACTÈRES

Exercice 1 (/fr/exercices-c/chaines-de-caracteres/exercice-1): Ecrire un programme C qui lit une chaîne de caractères et vérifie si elle est palindrome ou non. On rappelle qu'une chaîne de caractères est dite palindrome, si elle se lit de la même manière dans les deux sens. Exemple: *non*, *touot* et *1234321* sont toutes des chaînes de caractères palindromes.

Exercice 2 (/fr/exercices-c/chaines-de-caracteres/exercice-2): Ecrire un programme C qui lit deux chaînes de caractères et les affiche dans l'ordre alphabétique, en utilisant les deux méthodes suivantes:

• En utilisant la fonction *strcmp*.
• Sans utiliser la fonction *strcmp*.

Par exemple, si on donne en entrée les deux chaînes suivantes: *acb* et *abcd*, le programme doit afficher la chaîne abcd puis *acb*.

Exercice 3 (/fr/exercices-c/chaines-de-caracteres/exercice-3): Ecrire un programme C qui lit deux chaînes de caractères et permute leurs contenus en utilisant les deux méthodes suivantes:

• Avec la fonction *strcpy*;

• Sans la fonction *strcpy*.

Exercice 4 (/fr/exercices-c/chaines-de-caracteres/exercice-4): Ecrire un programme C qui lit une chaîne de caractères, et transforme chaque caractère majuscule en minuscule et vice versa.

Exercice 5 (/fr/exercices-c/chaines-de-caracteres/exercice-5): Ecrire un programme C qui lit deux chaînes de caractères et vérifie si la deuxième est une sous chaîne de la première ou non.
Exemple: *tout* est une sous chaîne de *surtout*.

Exercice 6 (/fr/exercices-c/chaines-de-caracteres/exercice-6): Ecrire un programme C, qui lit une chaîne de caractères représentant une phrase, et affiche dans l'ordre alphabétique toutes les lettres qui ne figurent pas dans cette chaîne de caractères.

Exercice 7 (/fr/exercices-c/chaines-de-caracteres/exercice-7): Ecrire un programme C qui lit une chaîne de caractères et supprime toutes les occurrences d'un caractère entré par l'utilisateur.

Exercice 8 (/fr/exercices-c/chaines-de-caracteres/exercice-8): Ecrire un programme C qui lit une chaîne de caractères puis un ensemble de caractères et affiche le nombre de fois que chacun d'eux a apparu dans cette chaîne.

Exercice 9 (/fr/exercices-c/chaines-de-caracteres/exercice-9): Ecrire un programme C qui lit une chaîne de caractères et calcule le nombre d'occurrences de chacun de ses caractères.

Exercice 10 (/fr/exercices-c/chaines-de-caracteres/exercice-10): Ecrire un programme C qui lit une liste de prénoms puis demande à l'utilisateur d'entrer une lettre et affiche tous les prénoms commençant par cette lettre.

STRUCTURES

Exercice 1 (/fr/exercices-c/structures/exercice-1): Ecrire un programme C qui définit une structure *point* qui contiendra les deux coordonnées d'un point du plan. Puis lit deux points et affiche la distance entre ces deux derniers.

Exercice 2 (/fr/exercices-c/structures/exercice-2): Ecrire un programme C qui définit une structure *etudiant* où un étudiant est représenté par son nom, son prénom et une note. Lit ensuite une liste d'étudiants entrée par l'utilisateur et affiche les noms de tous les étudiants ayant une note supérieure ou égale à *10* sur *20*.

Exercice 3 (/fr/exercices-c/structures/exercice-3): Ecrire un programme C, qui lit les noms complets des étudiants et leurs moyennes dans un tableau de structures. Puis actualise ces moyennes en ajoutant un bonus de:

- *1* point pour les étudiants ayant une note strictement inférieure à *10*.
- *0.5* point pour les étudiants ayant une note entre *10* et *15* incluses.

N.B.: la structure doit avoir deux éléments: une chaîne de caractères et un réel.

Exercice 4 (/fr/exercices-c/structures/exercice-4): Ecrire un programme C, qui lit le nom, le prénom et l'âge de plusieurs personne dans un tableau de structures, puis insère une nouvelle personne dans une position entrée par l'utilisateur.

Exercice 5 (/fr/exercices-c/structures/exercice-5): Ecrire un programme C, qui lit un ensemble de villes avec leur nombre d'habitants dans un tableau de structures, les trie et les affiche dans l'ordre croissant des nombres d'habitants.

Exercice 6 (/fr/exercices-c/structures/exercice-6): Ecrire un programme C qui lit un ensemble de personnes avec leurs âges, dans un tableau de structures, et supprime ensuite toutes celles qui sont âgées de vingt ans et plus.

POINTEURS

Exercice 1 (/fr/exercices-c/pointeurs/exercice-1): Ecrire un programme C qui utilise la notion de pointeur pour lire deux entiers et calculer leur somme.

Exercice 2 (/fr/exercices-c/pointeurs/exercice-2): Ecrire un programme C qui utilise la notion de pointeur pour la permuter le contenu de deux variables de type *char*.

Exercice 3 (/fr/exercices-c/pointeurs/exercice-3): Ecrire un programme C qui remplit un tableau d'entiers et calcule la somme de ses éléments en utilisant un pointeur pour son parcours.

Exercice 4 (/fr/exercices-c/pointeurs/exercice-4): Ecrire un programme C qui lit une chaîne de caractères et affiche cette chaîne à partir de la première occurrence d'un caractère entré par l'utilisateur. En utilisant pour ceci la fonction *strchr* et un pointeur pour le parcours de la chaîne.

Exercice 5 (/fr/exercices-c/pointeurs/exercice-5): Ecrire un programme C qui définit une structure permettant de stocker le nom, le prénom et l'âge d'une personne. Lit ensuite ces informations pour deux personnes et affiche le nom complet de la moins âgée d'entre elles en utilisant une seule fonction *printf* pour l'affichage du résultat.

Exercice 6 (/fr/exercices-c/pointeurs/exercice-6): Ecrire un programme C qui vérifie si une chaîne de caractères est palindrome en utilisant deux pointeurs pour son parcours.

Exercice 7 (/fr/exercices-c/pointeurs/exercice-7): Ecrire un programme C qui réserve l'espace mémoire à un tableau d'entiers de taille entrée par l'utilisateur, le lit puis l'affiche.

Exercice 8 (/fr/exercices-c/pointeurs/exercice-8): Ecrire un programme C qui réserve l'espace mémoire à un tableau d'entiers à deux dimensions dont la taille est entrée par l'utilisateur, puis le lit et l'affiche.

Exercice 9 (/fr/exercices-c/pointeurs/exercice-9): Ecrire un programme C qui réserve l'espace mémoire à un tableau de caractères sous forme d'un triangle droit, le remplit par des étoiles (*) puis l'affiche.

FONCTIONS

Exercice 1 (/fr/exercices-c/fonctions/exercice-1): Ecrire un programme C qui définit et utilise une fonction de prototype int Somme(int,int) qui prend en paramètres deux entiers et renvoie leur somme.

Exercice 2 (/fr/exercices-c/fonctions/exercice-2): Ecrire un programme C qui définit et appelle une fonction *bonjour* qui affiche le message *Bonjour*.

Exercice 3 (/fr/exercices-c/fonctions/exercice-3): Ecrire un programme C qui détermine le max de quatre entiers à l'aide d'une fonction *Max_4*, et qui doit utiliser une autre fonction *Max_2* qui trouve le max de deux entiers.

Exercice 4 (/fr/exercices-c/fonctions/exercice-4): Ecrire un programme C qui affiche les carrés des éléments d'un tableau d'entiers en utilisant les deux méthodes suivantes: la première se base sur une fonction *Affiche_Carre* qui prends en paramètre le tableau est affiche les carrés de tout ses éléments, et la deuxième utilise une fonction *Carre* qui affiche le carré d'un entier entré en paramètre.

Exercice 5 (/fr/exercices-c/fonctions/exercice-5): Ecrire un programme C qui définit et utilise une fonction *Permuter* qui permute les valeurs de deux variables réelles.

Exercice 6 (/fr/exercices-c/fonctions/exercice-6): Ecrire un programme C, qui définit et utilise une fonction *Inserer*, qui insère un entier dans un tableau. L'entier à insérer et la position d'insertion sont lus par cette fonction même.

Exercice 7 (/fr/exercices-c/fonctions/exercice-7): Ecrire un programme C qui définit et utilise une fonction de prototype *char * Reserver(int)* qui réserve l'espace mémoire, d'une manière dynamique, à une chaîne de caractères.

Exercice 8 (/fr/exercices-c/fonctions/exercice-8): Ecrire un programme C qui définit quatre fonctions pour effectuer les tâches suivantes:

- Réservation de l'espace mémoire pour un tableau d'entiers dynamique à deux dimensions;

- Lecture du tableau;
- Affichage du tableau;
- Libération de l'espace mémoire réservé.

Exercice 9 (/fr/exercices-c/fonctions/exercice-9): Ecrire un programme C qui définit et utilise une fonction *Trier* qui trie, dans un ordre croissant, les éléments d'un tableau d'entiers.

Exercice 10 (/fr/exercices-c/fonctions/exercice-10): Ecrire un programme C qui définit une fonction récursive qui calcule la somme des entiers entre zéro et un nombre passé en paramètre.

LISTES CHAÎNÉES

Exercice 1 (/fr/exercices-c/listes-chainees/exercice-1): Ecrire un programme C qui permet de créer et de lire une liste chaînée d'entiers et affiche ensuite ses éléments.

La structure utilisée pour les cellule de la liste doit être constituée d':

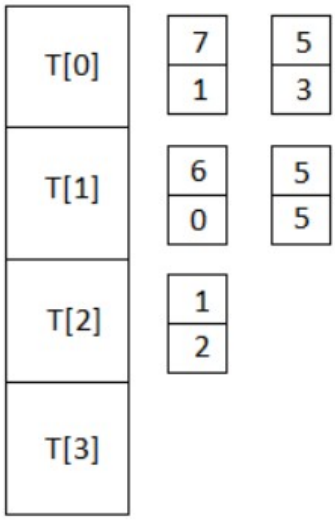
- un entier;
- un pointeur sur la structure de la liste.

Illustration:

Exercice 2 (/fr/exercices-c/listes-chainees/exercice-2): On définit une matrice creuse comme étant une matrice dont plus que la moitié des éléments sont nuls.

0	7	0	5	0	0
6	0	0	0	0	5
0	0	1	0	0	0
0	0	0	0	0	0

Pour minimiser l'espace occupé par ce type de matrice on choisi de les représenter sous forme d'un tableau de listes chaînées, de sorte que la $i^{ième}$ liste chaînée contient les éléments non nuls de la ligne *i* de la matrice et chacun d'eux accompagné du numéro de la colonne où il se trouve.



Tâches à faire:

- Définir la structure qui sera utilisée pour les cellules des listes chaînées;
- Transformer une matrice *M* de taille *nxm* en tableau de listes chaînées *T* comme défini précédemment;
- Afficher un élément *M[i][j]* à partir de *T*.

Exercice 3 (/3): Ecrire un programme C qui crée et lit une liste chaînée d'entiers, et lit ensuite un entier et une position et insère l'entier dans la position précisée.

Exercice 4 (/fr/exercices-c/listes-chainees/exercice-4): Ecrire un programme C qui crée et lit une liste chaînée d'entiers, puis supprime de cette liste toutes les occurrences d'un entier entré par l'utilisateur

Exercice 5 (/fr/exercices-c/listes-chainees/exercice-5): Ecrire un programme C qui inverse une liste chaînée en manipulant seulement ses pointeurs de liaison.

Exercice 6 (/fr/exercices-c/listes-chainees/exercice-6): Ecrire un programme C qui trie une liste chaînée d'entiers en utilisant le tri à bulles.

Exercice 7 (/fr/exercices-c/listes-chainees/exercice-7): Ecrire un programme C qui supprime tous les éléments redondants d'une liste chaînée d'entiers.

Exercice 8 (/fr/exercices-c/listes-chainees/exercice-8): Ecrire un programme C qui, à partir des éléments d'une liste chaînée, crée deux autres listes telles que la première contient les nombres pairs et la deuxième ceux impairs.