L3 – APP - LSI **Lahloukarim@free.fr**	**Septembre 2019** **Programmation Langage C/c++** **Abdelkrim LAHLOU**

TP-1 : Les expressions et les instructions de base

I – Expressions :

- Ecrire un programme qui détermine le jour de la semaine correspondant à une date à partir de l'algorithme donné ci-dessous, qui est valable pour les dates postérieures à 1582. La date est donnée sous la forme : *j* / *m* / *a*; avec *j* pour jour, *m* pour mois, et *a* pour année, ces valeurs devront être saisies par l'utilisateur. Attention, pour les mois de Janvier et Février, il faut augmenter *m* de 12 et diminuer *a* de 1. Ainsi le 12 Janvier 1854 sera considéré sous la forme : 12 / 13 / 1853, le 23 Février 2000 sous la forme 23 / 14 / 1999.

Dans l'algorithme fourni ci-dessous, la notation [ ] correspond à la partie entière (pour tronquer les chiffres après la virgule).

```
1) Calculer  s qui vaut la partie entière de a divisé par 100

2) Calculer  JD = 1720996,5 - s + [s/4] + [365,25*a] + [30,6001*(m+1)] + j

             JD = JD - [JD/7]*7

3) calculer  JS = [JD] MOD 7

4) afficher JS

5) éventuellement, afficher le jour sous forme de texte.

   si JS =  0, le jour est mardi,
   si JS =  1, le jour est mercredi,
   ....
   si JS =  6, le jour est lundi.
```

II - Tests, conditions, sélection et Boucles

- Ecrire un programme qui affiche une question et 4 possibilités de réponses, chacune précédée d'un numéro entre 1 et 4. L'utilisateur doit sélectionner le numéro de la réponse choisie, et le programme doit indiquer s'il s'agit de la bonne réponse ou si c'est une erreur.
- Ecrire un programme qui indique si une année est bissextile ou non : une année multiple de 4 est en général bissextile. Attention toutefois, les années séculaires (multiples de 100) ne sont pas bissextiles, sauf si elles sont multiples de 400 ! (1400 et 1900 ne sont pas bissextiles, mais 1200 et 2000 le sont).
- Saisie sécurisée :

On veut vérifier que le principe de saisie sécurisée fonctionne bien (car il faudra l'appliquer systématiquement en projet).

Vérification de la valeur saisie

Premier type de contrôle à faire : s'assurer que la valeur saisie respecte certaines conditions. On suppose, dans un premier temps, que l'utilisateur entrera un entier lorsqu'on demande un entier, ou un réel lorsque l'on demande un réel. Par contre, il peut se tromper, par exemple entrer une valeur négative alors qu'on lui demande une distance, ou un âge par exemple.

Ecrire un programme qui est capable de re-proposer une saisie jusqu'à ce que la valeur saisie par l'utilisateur soit bien comprise entre deux bornes *a* et *b*, telles que $a < b$. Il n'est pas interdit de rappeler les valeurs des bornes *a* et *b* dans un message précédant la saisie proprement dite.

scanf a-t-il bien reconnu l'entrée au clavier ?

L'instruction *scanf*, utilisée pour la saisie, est un peu plus évoluée que ce que nous avons présenté en cours, elle permet de faire une vérification de ce qui a été entré au clavier. Cela permet par exemple de contrôler que lorsqu'un entier est demandé, on puisse détecter si l'utilisateur n'a pas entré du texte à la place.

Pour cela, on peut utiliser une variable entière qui pourra stocker le résultat de *scanf()* de la manière suivante :

```
long resul_sais;
…
resul_sais=scanf("format",&variable);
```

Si `resul_sais` est égal à 1, alors c'est que la saisie est valide (c'est à dire que l'utilisateur n'a pas entré du texte alors qu'on lui demandait un entier par exemple).

Appliquez cette méthode pour faire une saisie totalement sécurisée pour l'exemple suivant : On veut saisir une date sous la forme heure, minute, secondes. La (les) saisie(s) doi(ven)t être proposée(s) jusqu'à ce que les valeurs soient valides.

Calculs de puissance, de factorielle et de fonctions classiques

- Ecrire un programme qui calcule x^y, ou *x* est un nombre quelconque, et *y* un entier quelconque (positif ou négatif).
- Ecrire un programme qui calcule n!, la factorielle de n.

Rappel n! = n x (n-1) x (n-2) x … x 2 x 1; 0! = 1

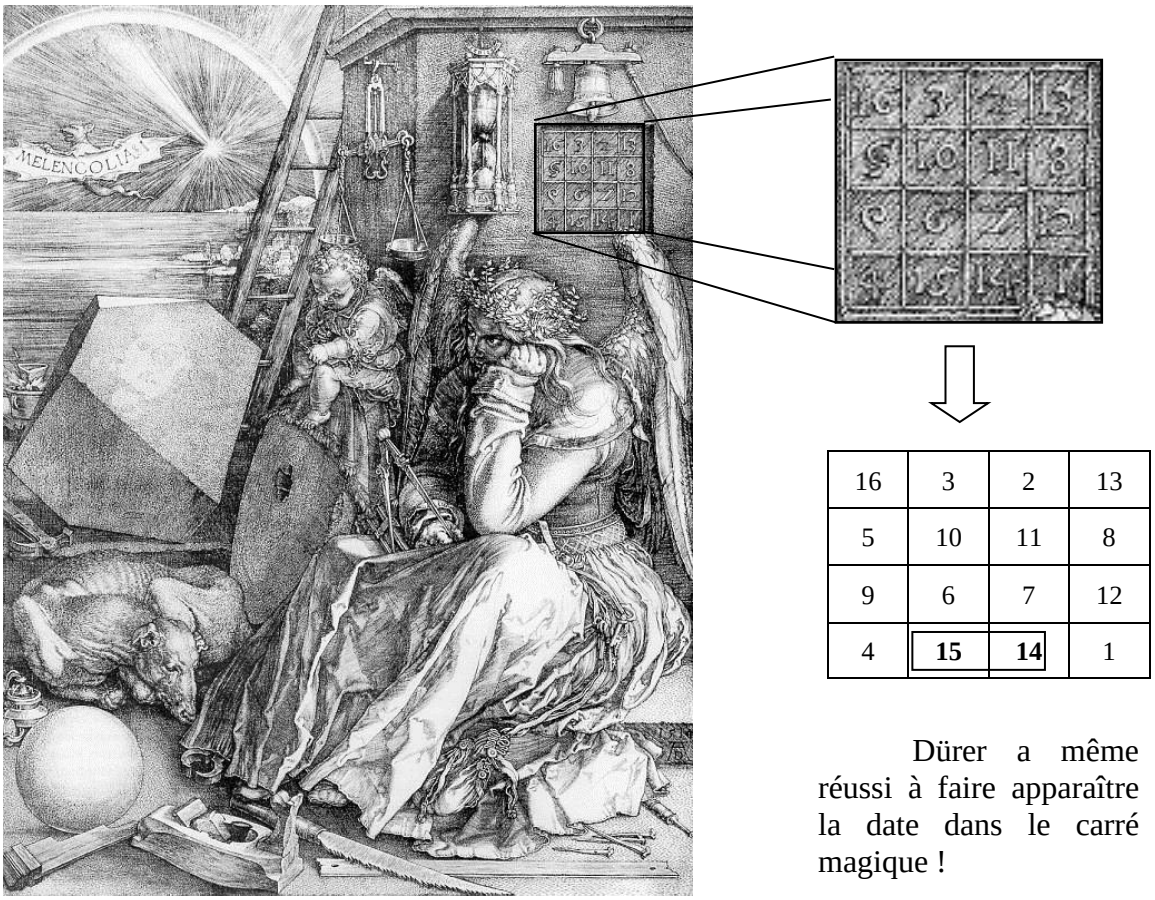
- Ecrire un programme qui calcule le nombre de combinaisons de p éléments parmi n, sans remise (le tirage du loto, par exemple, correspond à une combinaison de 6 éléments parmi 49), donné par la formule : n!/p! (n-p)!

III - Tableaux à plusieurs dimensions : Carré magique

Un carré magique d'ordre *N* est un tableau à deux dimensions, carré. Cela signifie que la taille utile des deux dimensions est la même. Dans ce tableau seront stockées tous les entiers compris entre 1 et N^2. Cependant,

ces valeurs sont rangées de telle sorte que la somme des nombres rangés dans chaque ligne, chaque colonne, et chaque diagonale est la même.

Cet objet mathématique a longtemps fasciné les mathématiciens et les artistes à l'époque de la renaissance, à tel point que le célèbre artiste Albrecht Dürer (1471 – 1528) en a fait apparaître un sur l'une de ses gravures nommé Melancholia.



Melancholia (Albrecht Dürer – 1514)

16	3	2	13
5	10	11	8
9	6	7	12
4	**15**	**14**	1

Dürer a même réussi à faire apparaître la date dans le carré magique !

Il existe un algorithme simple de création de carré magique pour les ordres *N* impairs.

Phase initiale :

Dans un premier temps, il est nécessaire d'initialiser toutes les cases du carré magique avec la valeur 0, cette dernière permettant de repérer qu'une case est libre ou déjà occupée par un nombre qui y serait rangé.

Il faut placer le chiffre 1 dans la case située juste au dessus de la case milieu du carré (cette case existe car le carré est d'ordre *N* impair). Si taille est la variable contenant la taille utile des deux dimensions du tableau, quels sont les indices de cette case où l'on commence le remplissage du carré magique ?

Phase de remplissage:

On continue en plaçant les nombres 2, 3, …, jusqu'à N^2, selon la diagonale de direction : vers le haut et vers la droite. Soient deux variable lig et col contenant les indices de la ligne et de la colonne de la case en cours de traitement. Quelles opérations appliquer à lig et col pour traduire : "vers le haut" et "vers la droite" ? Aidez-vous d'un schéma au besoin.

Bien entendu, par cette progression, on dépasse assez rapidement les bords du carré magique. Quelles sont les conditions à écrire pour tester que les indices lig et col dépassent du tableau ?

- Si l'indice lig dépasse par le haut, alors on repart de la dernière ligne du tableau, en conservant la même valeur pour col.
- Si l'indice col dépasse par la droite, on repart de la première colonne du tableau, en conservant la même valeur pour lig.
- Si l'indice col dépasse par la gauche, on repart de la dernière colonne du tableau, en conservant la même valeur pour lig.

Vérification de la case atteinte :

Il peut arriver que l'on atteigne une case contenant un nombre rangé précédemment. Comment tester si une case est déjà occupée par un nombre ?

Dans le cas où on atteint une case occupée, on se déplace dans la diagonale vers le haut et vers la gauche (et non plus vers la droite) tant que la case atteinte n'est pas libre. Il faut également contrôler, lors de ce déplacement, que l'on ne dépasse pas les limites du tableau en appliquant les mêmes règles que précédemment. Dès que l'on a réussi à placer le nombre, la progression reprend dans la diagonale vers le haut et vers la droite.

Ecrivez un programme qui remplit un carré magique d'ordre *N* impair et qui affiche ce carré une fois rempli. Le programme devra vérifier que *N* est impair (si *N* est pair, on ne remplit pas le carré), et que $N < 20$ (au delà de $N=20$, le carré devient difficile à visualiser sur l'écran). Votre programme doit utiliser de l'allocation dynamique. Vous vérifierez sur un ou deux exemples que vous obtenez bien un carré magique.

IV - Tableaux à plusieurs dimensions : Vote

Le maire de votre municipalité vous contacte pour lui écrire un programme en Langage C qui permet de gérer les résultats d'une élection présidentielle dans votre commune.
Le programme nous invite à saisir les noms des candidats en lice (on doit vérifier qu'on a au moins 2 candidats sinon on génère un message d'erreur : une élection présidentielle doit comporter au moins 2 candidats).
Puis, le programme nous invite à saisir les résultats du dépouillement pour chaque candidat dans une liste de bureaux de vote.
Le nombre de bureaux de votes de votre municipalité doit être géré de manière dynamique vu qu'il peut changer d'une élection à une autre.

Ecrire le programme C qui permet de saisir les résultats du dépouillement dans chaque bureau de vote pour chacun des candidats.
Le programme doit afficher pour chaque bureau de vote le nombre d'électeurs prévus, le nombre de votants, le nombre de bulletins nuls et le classement des candidats.
Le programme affiche par la suite le résultat définitif de votre municipalité :

- nombre d'électeurs prévus
- nombre de votants
- nombre de bulletins nuls
- le classement des candidats par ordre de mérite selon le nombre de voix obtenues en affichant leurs noms et leur nombre de voix obtenues.