Cahier de TP n°1

Prise en main de l'environnement de développement, compilation, édition des liens

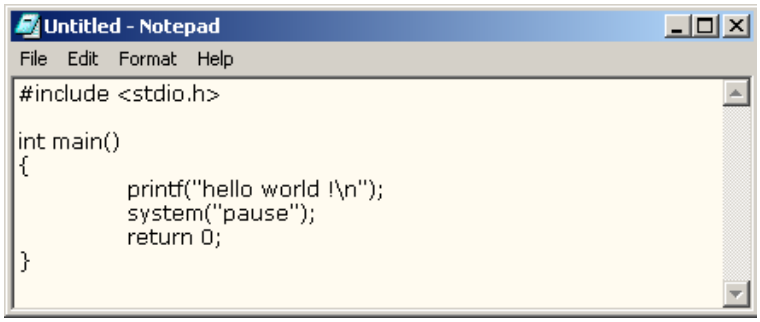
Vous êtes bien installées devant votre ordinateur ? votre compte est bien ouvert ? commençons donc par lister les outils à utiliser pour faire nos premiers programmes.

La première chose à faire est de créer une hiérarchie de répertoires pour les TPs d'informatique, afin de ne pas stocker tous les fichiers au même endroit. Avec l'explorateur de fichiers, créez un répertoire nommé 'INFO' ou 'LangageC', puis allez dans ce répertoire et créez un répertoire nommé 'TP1'. Allez dans ce répertoire TP1, il doit normalement être vide. Pensez à créer un nouveau répertoire pour chaque séance de TP, cela vous permettra de bien classer vos fichiers.

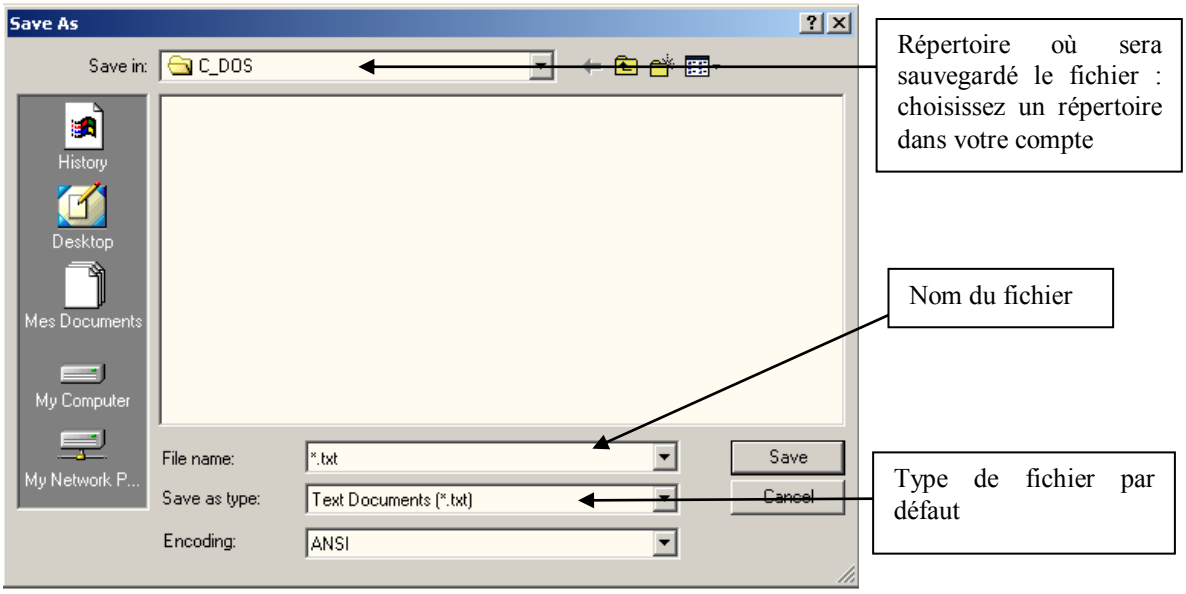
Pour écrire les programme source en langage C, nous utiliserons un éditeur de texte quelconque, par exemple NotePad, le bloc-notes (accessible par : Démarrer – Programmes - Accessoires – Bloc Notes).

Nous allons d'ailleurs tout de suite l'exécuter et saisir le programme suivant (un exemple tiré du cours) :

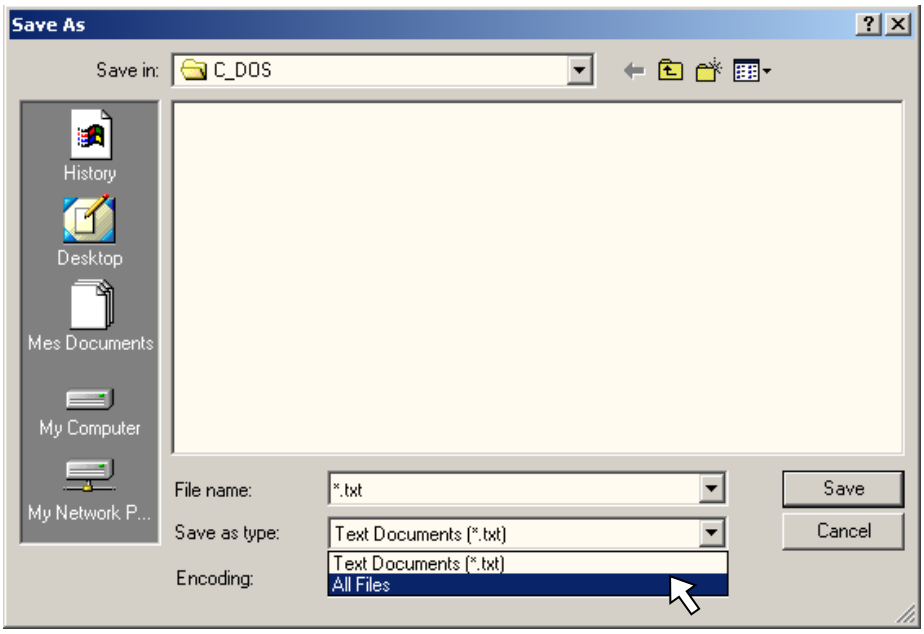
```
#include <stdio.h>

int main()
{
	printf("hello world !\n");
	system("pause");
	return 0;
}
```



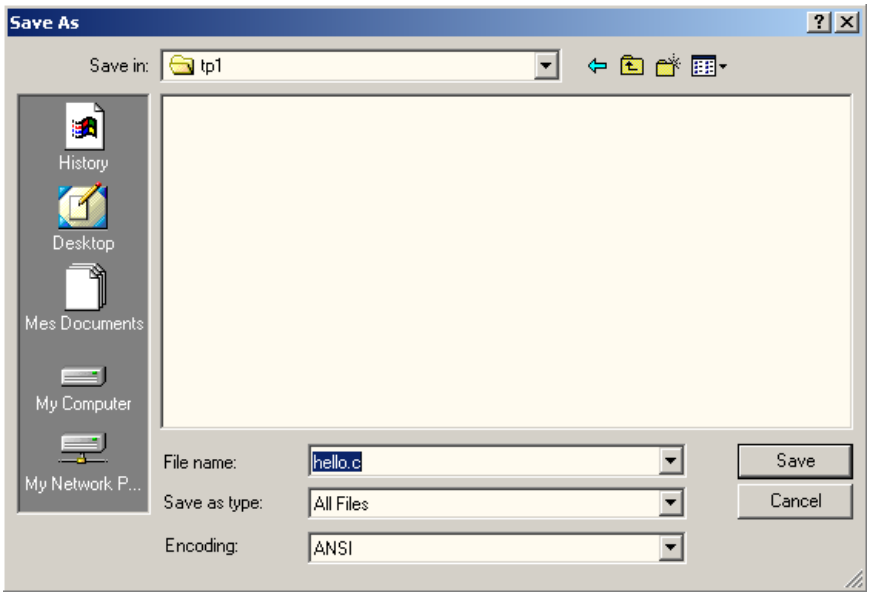
pour sauvegarder ce programme, sélectionner dans le menu 'file' (ou 'fichier') l'option 'save as …' (ou 'enregistrer sous…'); la fenêtre suivante apparaît alors:



Il faut d'abord sélectionner le répertoire TP1 qui vient d'être créé. Ensuite, on constate que le bloc-notes propose de sauvegarder le fichier comme un fichier texte (c'est ce qu'il fait par défaut), et donc de lui ajouter une extension .txt, ce qui ne nous convient pas, car nous voulons le sauvegarder avec l'extension .c. Il faut donc changer le champ 'Save as type:' de la manière suivante :



Choisir le type 'All Files', puis taper le nom du fichier 'hello.c' dans le champ 'File name :', comme indiqué sur le dernier schéma ci-dessous.



Il ne reste plus qu'à cliquer sur le bouton Save pour sauvegarder le fichier. N'oubliez pas que le bloc-notes est assez têtu, il faut donc bien vérifier avant chaque sauvegarde qu'il ne remet pas l'extension .txt par défaut.

Vous pouvez vérifier avec l'explorateur de fichiers que le fichier hello.c se trouve bien dans le répertoire TP1.

A vos marques…prêts ? compilez !

Pour la compilation, nous allons reprendre ce que nous avons vu en cours :

Il faut dans un premier temps ouvrir une fenêtre DOS : Démarrer – Programmes – Accessoires – Invite de Commandes. Il faut ensuite se rendre dans le répertoire où se situe le fichier hello.c.

Pour faire la compilation, il faut utiliser le programme nommé gcc.exe, dont le nom complet est E:\apps\Dev-Cpp\bin. Pour éviter d'avoir à taper entièrement ce nom, nous allons mettre à jour la variable d'environnement nommé `path` qui contient une liste de répertoires à explorer pour trouver un programme. Pour cela, tapez la commande suivante :

```
path=%path%;E:\apps\Dev-Cpp\bin
```

pour savoir si cette mise à jour a bien été effectuée, tapez la commande

```
gcc
```

Vous devriez obtenir le message suivant :

```
gcc: no input files
```

Si vous obtenez celui-ci :

```
'gcc' is not recognized as an internal or external command,
operable program or batch file.
```

C'est que la mise à jour ne s'est pas bien faite, vérifiez bien que vous n'avez pas fait de faute de frappe !

Nous pouvons maintenant passer à la compilation proprement dite :

rappel : pour compiler un fichier .c, on utilise la commande suivante :

```
gcc -c nom_du_fichier.c
```

donc, pour notre exemple, on utilisera la commande :

```
gcc -c hello.c
```

le compilateur indique s'il trouve des erreurs et éventuellement peut indiquer des avertissements (warning), qui n'empêchent pas la compilation mais peuvent gêner l'exécution du programme.

Vérifiez que vous avez maintenant dans le répertoire TP1 le fichier hello.o. S'il ne s'y trouve pas, le compilateur a du vous signaler au moins 1 erreur qui a rendu la compilation impossible.

Edition des liens

Dernière étape avant l'obtention du fichier exécutable : l'édition des liens. A partir du fichier .o obtenu par compilation, on génère le fichier exécutable à l'aide de la commande suivante :

```
gcc nom_du_fichier.o -o nom_du_fichier.exe
```

dans notre exemple, cette commande est donc :

```
gcc hello.o -o hello.exe
```

on peut choisir de nommer le fichier exécutable autrement, à la seule condition que son extension soit .exe.

exemple :

```
gcc hello.o -o bonjour.exe
```

Après cette commande, vous devez normalement avoir un fichier hello.exe (ou d'un autre nom si vous avez choisi un autre nom) dans votre répertoire.

Exécution du programme :

Vous avez deux possibilités pour exécuter votre programme :

Par la fenêtre DOS, en donnant comme commande le nom du fichier exécutable sans son extension : hello ou bonjour pour l'exemple précédent;

```
hello
hello world!
press any key to continue . . .
```

Par l'explorateur de fichiers, en double-cliquant sur l'icône correspondant au fichier exécutable : le système ouvre automatiquement une fenêtre DOS dans laquelle le programme s'exécute, elle est refermée à la fin du programme.

Et maintenant…à vous de jouer !

TP 1 : variables, affectations, affichage et saisie.

Reprenons quelques exercices du premier cahier de TD : le but est d'arriver à un résultat concret tout en travaillant la traduction langage algorithmique ⇔ langage C.

- Ecrivez un programme qui calcule et affiche la moyenne de trois nombres entiers saisis au clavier

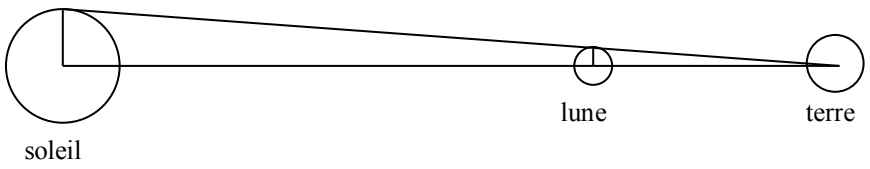
- Ecrivez un programme qui calcule et affiche la variance de trois nombre entiers saisis au clavier.

  Formule de la variance : $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{m})^2$, où n est le nombre de valeurs, ici n=3. $\overline{m}$ est la moyenne des valeurs.

- Ecrire un programme qui calcule la différence, en secondes, entre deux dates (en heures, minutes, secondes) de la même journée.

- Ecrivez un programme de suivi d'épargne : un capital C est placé sur un compte dont le taux d'intérêt semestriel T est de 2,43 %. Le programme doit calculer et afficher le capital au bout de : 6 mois, 12 mois, 18 mois et 30 mois.

- On cherche à évaluer le diamètre du soleil en constatant que, lors d'une éclipse de lune, celle ci masque complètement le soleil. On utilise pour ce faire le théorème de Thalès appliqué à la figure suivante (pas à l'échelle) :



Données numériques : Distance terre-lune : 384 000 km.
Rayon de la lune : 1740 km
Distance terre –soleil : 150 000 000 km

Ecrire un programme qui calcule une approximation du diamètre du soleil.

Le diamètre mesuré du soleil est en fait de : 1 400 000 km. Que pensez-vous de la méthode utilisée ?

- On lâche un caillou dans un puits asséché et très sombre. On entend un bruit d'impact au bout de *T* secondes. Ecrire un programme qui calcule et affiche : la profondeur du puits et la vitesse du caillou à l'impact.

Un nouveau programme à partir d'un algorithme que l'on vous donne :

- 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 donné sous la forme : 12 / 13 / 1853, le 23 Février 2000 sera donné sous la forme 23 / 14 / 1999. Ce calcul ne sera pas fait par le programme, il vous faudra faire attention aux valeurs à fournir au programme.

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 =  7, le jour est lundi.
```

TP2 : Tests, conditions, sélection

- 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).

- On cherche à écrire un programme qui estime le débit d'une connexion ADSL en fonction de plusieurs paramètres : le type de modem, l'heure de connexion (pour estimer le trafic), et le NRA. On donne les données suivantes :

Le débit estimé est donné par : débit réel x (1 - coefficient d'occupation); le débit réel étant le maximum entre ce que peut fournir le NRA et ce que peut recevoir le modem, le coefficient d'occupation dépend du trafic, donc de l'heure de connexion. Pensez bien à la manière dont vous allez matérialiser le choix du NRA et du modem !

Heure de connexion	Coefficient d'occupation
08h01 – 20h59	23 %
21h00 – 01h30	40 %
01h31 – 08h00	12 %

NRA	Débit offert par le NRA
GOB75	5 Mb/s
CHO94	4 Mb/s

Modem	Débit max reçu par le modem
Terminator Yf32	4,32 Mb/s
PacWac 002	1,7 Mb/s
xDSL Monster 3.38	5,2 Mb/s

- Avec une structure de sélection, complétez le programme déterminant le jour de la semaine correspondant à une date donnée, en affichant en toutes lettres le jour de la semaine correspondant.

- Ecrire un programme qui résout une équation du second degré dans ℝ, puis dans ℂ.

- Ecrire un programme qui résout un système de deux équations à deux inconnues, ce programme devra traiter tous les cas particuliers.

Rappel : on note par $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ la quantité *ad-bc*, nommé déterminant de *a*, *b*, *c* et *d*.

Le système à résoudre est : $\begin{cases} a_1.x + b_1.y = c_1 \\ a_2.x + b_2.y = c_2 \end{cases}$

Etape 1 : on calcule $D = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}$. Si cette quantité *D* est nulle, alors, il faut calculer $\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}$. Si cette quantité est nulle, le système admet une infinité de solutions, sinon il n'admet aucune solution.

Si *D* n'est pas nul, alors on obtient *x* et *y* par :

$$x = \frac{\begin{vmatrix} c_1 & c_2 \\ b_1 & b_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}} ; y = \frac{\begin{vmatrix} a_1 & a_2 \\ c_1 & c_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}}$$

TP3 : Boucles

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 reproposer 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)!