Petit cours de programmation C

1. Généralités

1.1. Le C ?

Le C est un langage de programmation de haut niveau, contrairement aux langages dits de bas niveau : asm, ... qui sont très proches du langage machine, mais qui est tout de même assez proche du langage machine, comparé à certains langages comme le Visual Basic, le pascal.... Il est ainsi très puissant et très polyvalent.

1.2. Ce dont vous avez besoin

Un éditeur de texte, un compilateur et un éditeur de liens.
Exemples de compilateurs :

- Sous Windows : gcc (gratuit et libre), MS Visual C ou C++…
- Sous Linux : gcc (libre : fournis avec beaucoup de distrib').

Un éditeur de liens sont souvent fournis avec.
N'hésitez pas nous plus à vous procurer un ou plusieurs livres sur le C (et/ou des tutoriaux trouvés sur le net, ...comme le mien ;o)). La référence en la matière : *Le Langage C* par Kerningham & Ritchie, les concepteurs du langage.
Et aussi, n'hésitez pas à fréquenter les newsgroup : fr.comp.lang.c notamment.

2. Composition d'un programme C

2.1. Les fichiers "headers"

Ce sont des fichiers dont l'extension est traditionnellement ".h" (bien qu'on puisse utiliser n'importe quelle extension). Un grand nombre de "headers systèmes" est fourni avec les compilateurs. Par exemple, 'stdio.h' qui permet, entre autre, d'utiliser printf , scanf ... (voir plus loin : Entrée et sortie).
On les inclut dans le programme via l'instruction :

```
#include <fichier.h>
```

En fait, juste avant la compilation, le compilateur remplace cette instruction par le texte contenu dans ce fichier.
Les headers contiennent traditionnellemen les **déclarations** des fonctions, constantes, etc. Ils sont très souvent associés à un fichier .c contenant les **définitions** correspondantes, ou alors d'une bibliothèque (.lib) qui est une sorte de fichier C précompilé (on n'a pas accès au code, mais on peut l'inclure dans notre programme).

2.2. Les fonctions

Un programme est constitué d'une ou plusieurs fonctions. Une seule est obligatoire, c'est le point d'entrée du programme. Elle se nomme *main*.
Une fonction est de la forme :

```
[Type-retour] nom-fonction (liste de paramètres…)
```

Ex:

```
#include <stdio.h>

double f1 (int unEntier)
{
      return 2.5*unEntier ;
}

int main(void)
{
      printf("%f", f1(3));
      return 0;
}
```

2.3. Déclaration de variable

Les variables sont des sortes de cases en mémoire dont la taille varie suivant le type. Elles sont déclarées au début du programme ou d'un bloc (morceau de code entre accolades) et peuvent stocker des valeurs.
Ex :

```
int i = 0 ;  /* définition d'un variable i initialisée à 0 */
```

2.4. Déroulement de la compilation et de l'édition des liens

Lorsque vous lancez "build all" ou "make all" dans votre compilateur, il se déroule plusieurs étapes :

1. Le compilateur remplace les portions de texte qui doivent être remplacés par d'autres (ex : les fichiers "headers", les macros,....).
2. La compilation proprement dite s'effectue : le compilateur transcrit le ou les sources (*.c) en langage machine. -> fichiers objets (*.obj ou *.o).
3. Le linker (ou lieur ou éditeur de lien) crée un fichier exécutable (*.exe) à partir de ces fichiers objets.

Attention : toute instruction C se termine par un point-virgule (";") !

3. Le contenu d'un prog

3.1. Entrée et sortie (I/O)

3.1.1. printf

`printf` est l'instruction qui permet l'affichage formaté sur la sortie standard (très souvent, l'écran...). On peut afficher du texte, des valeurs, faire des retours à la ligne (voir plus bas les séquences d'échappement), etc... La syntaxe est la suivante (ne pas oublier d'inclure "stdio.h" qui contient la déclaration de printf !) :

```
printf("[texte à afficher,...]");
```

Exemple :

```
#include <stdio.h>
int main(void)
{
      printf("Vive video.game.free.fr !");
      return 0;
}
```

Lors de l'exécution du programme ainsi obtenu, l'écran affiche fièrement :

```
Vive video.game.free.fr !
```

On peut aussi insérer dans le texte des "séquences d'échappement" : elles sont introduites par un "backslash" ("\" , Alt Gr + 8) précédant un autre caractère ou plusieurs chiffres.

TABLEAU DES SEQUENCES D'ECHAPPEMENT.

Séquence	Signification
\n	Retour à la ligne (= Entrée dans un traitement de texte)
\t	tabulation horizontale (= touche TAB)
\v	tabulation verticale (peu utilisé)
\b	efface le dernier caractère (= backspace)
\r	carriage return : le chariot retourne au début de la ligne

\f	saut de page
\a	déclenche un bip sonore du haut-parleur interne du PC.
\’	affiche une apostrophe
\"	affiche un guillemet
\\	affiche un « backslash » (\)
\ddd	affiche codes ASCII en notation octale
\xdd	affiche codes ASCII en notation hexadécimale

Exemple :

```
#include <stdio.h>

int main(void)
{
      printf("Vive\n\tvideo.game.free.fr !");
}
```

Le programme affiche désormais :

```
Vive
video.game.free.fr !
```

Magique, non ?

On peut aussi faire afficher à printf les données contenues dans des variables.... La syntaxe est alors la suivante :

```
printf("%[lettre en fonction de la variable]",[variables dans l'ordre
d'apparition...]);
```

Ex:

```
int variable = 2;
int jour = 23;
int annee = 2000;
char mois = 'J';
printf("%i points\n", variable);
printf("Aujourd'hui le %i %c %i\n", jour, mois, annee);
```

...qui afficherait alors :

```
2 points
Aujourd'hui le 23 J 2000
```

Voilà pour cette petite initiation.
N’hésitez pas à fouiller sur le web pour trouver mieux (et surtout plus complet).
Bon courage !