Editeurs de texte

Pour les travaux pratiques sur le langage C, vous utiliserez l'éditeur de texte de votre choix pour saisir vos programmes. Les noms de fichiers, qui contiennent vos programmes, doivent avoir l'extension ".c". Dans la suite des TP/TD, on appellera les fichiers qui contiennent des programmes C des fichiers sources.

Nous vous demandons de créer un repertoire par séance de TP. Vous pouvez utiliser la commande suivante :

$$mkdir \quad tp1\_votre\_nom$$

La commande :

$$cd$$

vous permet de changer de répertoires.

Les principales étapes d'une compilation d'un programme C

Pour pouvoir tester et évaluer un programme C, il faut utiliser un compilateur. Il s'agit d'un outil informatique qui permet la création de fichiers (ou de codes) exécutables par la machine.

Les principales étapes d'un compilateur sont :

1. Prétraitement des fichiers sources (ou préprocesseur) : comme son nom l'indique, cette étape consiste à faire un traitement préalable de votre fichier source avant la phase de compilation proprement dite. Il s'agit principalement d'un travail de substitution d'un texte (pour être précis dictée par une directive) par un autre texte. Par exemple, si votre programme source contient la directive suivante :

   ```
   #define pi 3.141592
   ```

   alors le préprocesseur remplace toutes les occurrences de pi par 3.141592.

   Le préprocesseur supprime également les commentaires.

2. Compilation/assemblage : cette étape consiste d'abord à vérifier que votre programme respecte la syntaxe des instructions du langage C. Si votre programme ne contient aucune erreur, un code binaire (appelé aussi fichier objet) est généré.

3. Editions de lien. Pour réaliser une application, on est souvent amené à décomposer un problème en un ensemble de sous-problèmes plus faciles à programmer. Le programme C associé à un sous-problème peut-être écrit dans un fichier source indépendant. Chaque fichier source peut-être alors compilé à part. L'édition de liens consiste à relier les différents fichiers objets, générés pendant la phase de compilation, afin de produire un seul fichier exécutable.

La commande gcc

Dans un premier temps, nous utiliserons qu'un seul fichier source pour écrire vos programmes. Vous utiliserez la commande suivante pour lancer le préprocesseur, compiler vos fichiers source et générer du code exécutable :

$$gcc \quad nom.c - Wall - o \quad nom$$

Dans cette commande "nom.c" est le nom de votre fichier source. Le fichier "nom" est l'identifiant du fichier exécutable qui sera produit après la phase de la compilation (si l'option -o n'est pas mentionnée, le compilateur C génère par défaut un fichier exécutable nommé a.out). Pour tester votre programme, il suffit de lancer la commande "./nom". L'option "-Wall" demande au compilateur de générer tous les avertissements d'erreurs (warnings).

Remarques

- Si vous voulez juste produire le résultat du pré-processeur, utiliser l'option -E :

$$gcc - E \quad nom.c$$

- De même, si vous voulez générer uniquement le fichier objet et non l'exécutable, utiliser l'option -c.
- Il est recommandé de rajouter les options "-ansi -pedantic" pour vérifier que votre programme C respecte les normes ANSI.

Rappels sur les commandes d'entrées/sorties printf et scanf

La fonction printf est de la forme :

printf(chaîne de caractère s, arg1, arg2, arg3, ...);

Elle imprime le texte contenu dans s de gauche à droite. A la rencontre du caractère "%" une séquence, dite d'échappement (ou format d'impression), est exécutée. La séquence qui débute par % utilise au fur et à mesure (un par un) les arguments arg1, arg2, arg3, ...
Les formats d'impression les plus utilisés sont :

- %c : impression d'un caractère.
- %d : int [1].
- %u : unsigned integer.
- %o : octal non signé.
- %x (%X) : hexadécimal non signé [2]
- %ld et %lu sont utilisés pour le type long int et long unsigned int.
- %f : float
  - Ecriture classique [-] xx.xxxxxx
  - Par défaut, on n'affiche que six chiffres après la virgule.
- %e (ou %E): float
  - Ecriture en notation scientifique
  - Par défaut, on n'affiche que six chiffres après la virgule.
- %g : float
  - Ecriture compacte de %e ou %f
- %s : chaîne de caractères.

Impression de caractères spéciaux (les plus utilisés):

Caractère	Signification	Code ASCII
\n	Retour à la ligne	10
\t	Tabulation horizontale	9
\"	Imprime les guillemets	34
%%	Impression du caractère %	37

Pour la lecture des données, on utilisera la fonction scanf. Sa syntaxe est de la forme :
scanf(une chaîne de caractères, &var1, &var2,...);

Scanf balaye des séquences précédées par le caractère "%". Chaque séquence qui débute par % indique le type de données à lire. Les variables var1, var2 indiquent les emplacements où seront stockées les données lues. Quelques formats de lecture :

[1] s'applique aussi pour char, short et en version signée
[2] x pour minuscule et X pour majuscule

- %c : lecture d'un seul caractère.
- %u : lecture d'un entier non signé.
- %d : lecture d'un entier.
- %o : nombre entier en octal
- %f : lecture d'un nombre réel de type float.
- %s : chaîne de caractères.

Exercice 1:

Compiler et exécuter le programme suivant :

```
/****
Mon premier programme C
 ***/

#include <stdio.h>
int main (void)
{
  printf ("Voici mon premier programme C. \n");
  printf ("Il est réalisé pendant les séances de TP. \n");
  return(0);
}
```

Exercice 2:

Le but de cet exercice est d'écrire la séquence d'instructions qui permet d'afficher le message suivant (la réalisation du jeu proprement dit se fera dans un TP ultérieur) :

```
**********************************************************
**          Bienvenu au jeu de Mastermind               **
**          Les caractères autorisés sont :             **
**          b : blanc, j : jaune, r : rouge,            **
**          v : vert, n : noir et g : gris.             **
**     Chaque combinaison doit avoir 4 caractères       **
**     Exemple : rbjg                                   **
**     Vous avez droit à 10 essais.                     **
**          Bon courage                                 **
**********************************************************
```

Exercice 3:

Compiler et exécuter le programme suivant :

```
#include <stdio.h>
int main (void)
{
    int a, b;
    a=16;
    b=016;
    printf ("Pourquoi la variable a (=%d) a une valeur différente de
            la variable b (=%d) ? \n", a, b);
    return(0);
}
```

Commentez le résultat de l'exécution de ce programme.

Exercice 4:

- Ecrire un programme C qui lit un entier (de type unsigned short int), qui représente un code ASCII en décimal, et affiche le caractère associé,
  - Si l'utilisateur rentre le nombre 90, votre programme affichera le caractère 'Z'.
- Ecrire un programme C qui lit un caractère (de type char) et affiche le code ASCII associé en décimal, en octal et en hexadécimal (minuscule et majuscule).
  - Si l'utilisateur rentre le caractère 'Y', votre programme affichera :
    Le code ASCII associé au caractère Y est : 89 (en décimal), 131 (en octal), 59 (en hexadécimal minuscule) et 59 (en hexadécimal majuscule).

Exercice 5:

Considérons le programme suivant :

```
1 #include <stdio.h>
2 int main (void)
3 {
4 unsigned char c='B';
5 int j=13;
6 ...
7 return(0);
8 }
```

- Donne la valeur des variables c et j en hexadécimal (rappel le code ASCII de 'A' est 65 en décimal).
- Compléter le programme ci-dessous pour confirmer votre résultat.
- Compléter le programme ci-dessous pour afficher le contenu des variables c et j en hexadécimal ainsi que leur adresse.

Exercice 6:

Sachant que le code ASCII de '\a' (bip) est 7 et que le code ASCII de '0' est 48, dites qu'affiche le programme C suivant :

```
#include <stdio.h>
int main (void)
{
    printf("Impression 1 : %c. \n", 7);
    printf("Impression 2 : %c. \n", '7');
    printf("Impression 3 : %d. \n", 7);
    printf("Impression 4 : %d. \n", '7');
    return(0);
}
```

Exercice 7:

Grâce aux différentes options de la fonction printf, un étudiant a réalisé les affichages suivants du nombre d'or (≈1.61803398875) :

```
Ecriture no.  1   :  1.618034
Ecriture no.  2   :   1.618034
Ecriture no.  3   :  01.618034
Ecriture no.  4   :    1.618034
Ecriture no.  5   :  001.618034
Ecriture no.  6   :     1.618034
Ecriture no.  7   :  0001.618034
Ecriture no.  8   :        1.618
Ecriture no.  9   :  0000001.618
```

Ecrire un programme C qui reproduit l'affichage ci-dessus (sachez que l'écriture 1 a été obtenue en utilisant uniquement le format "%f").

Exercice 8:

- Exécuter le programme suivant :

```
#include <stdio.h>
int main (void)
{
unsigned char  i='A';
int j=1;
printf ("\n \t i=%d \t \"j=%d\" \n", i, j);
printf ("\n \t i=%c \t \" \n", i);
return(0);
}
```

- Un étudiant, au moment de la saisie du programme ci-dessous, s'est rendu compte que la touche \ ne fonctionne pas. Proposer une ré-écriture du programme ci-dessous sans utiliser le caractère \.
- **Indication :** Utiliser les codes ASCII des caractères retour à la ligne, etc.

Exercice 9:

Ecrire un programme C qui, sans utiliser les codes ASCII associés aux caractères,

- lit un caractère minuscule et
- le transforme en un caractère majuscule.

Nous rappelons que :

- les codes ASCII des lettres majuscules (respectivement minuscules) se suivent.
- Par exemple,
  - le code ASCII de la lettre 'B' est égal à celui de la lettre 'A' plus 1.
  - Le code ASCII de la lettre 'C' est égal à celui de la lettre 'B' plus 1 et
  - ainsi de suite.