Remarque : N'oubliez pas d'utiliser le proxy `cache-etu.univ-artois.fr` avec le port 3128 pour l'accès au réseau informatique.

Éditeurs de texte

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

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

```
mkdir tp1_votre_nom
```

La commande suivante vous permet de changer de répertoire :

```
cd nom_du_répertoire
```

Les principales étapes de la compilation d'un programme C

Pour tester et exécuter un programme C, il faut utiliser un compilateur, qui est un outil informatique permettant de créer des fichiers exécutables par la machine.

Les principales étapes d'un compilateur sont :

1. **Prétraitement des fichiers sources (ou préprocesseur)** : cette étape consiste à effectuer un traitement préalable du fichier source avant la compilation. Elle consiste principalement à remplacer certains textes selon les directives du programme. 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 vérifie que votre programme respecte la syntaxe du langage C. Si aucune erreur n'est détectée, un code binaire (appelé fichier objet) est généré.

3. **Édition de liens** : pour réaliser une application, un problème peut être décomposé en plusieurs sous-problèmes, chacun programmé dans un fichier source indépendant. Chaque fichier source peut être compilé séparément. L'édition de liens relie les différents fichiers objets générés afin de produire un seul fichier exécutable.

La commande `gcc`

Dans un premier temps, nous utiliserons un seul fichier source pour écrire vos programmes. La commande suivante permet de lancer le préprocesseur, compiler vos fichiers source et générer un exécutable :

```
gcc nom.c -Wall -o nom
```

Dans cette commande :

- `nom.c` est le nom du fichier source.
- `nom` est le fichier exécutable produit (par défaut, si `-o` n'est pas utilisé, le compilateur génère `a.out`).
- `-Wall` demande au compilateur d'afficher tous les avertissements.

Pour exécuter votre programme, utilisez la commande :

```
./nom
```

Remarques

- Pour obtenir uniquement le résultat du préprocesseur, utilisez l'option `-E` :

```
gcc -E nom.c
```

- Pour générer uniquement le fichier objet sans exécutable, utilisez l'option `-c`.
- Il est recommandé d'ajouter les options `-ansi -pedantic` pour vérifier que votre programme C respecte les normes ANSI.

Rappels sur les commandes d'entrée/sortie `printf` et `scanf`

La fonction `printf` s'utilise ainsi :

```
printf("chaîne de caractères", arg1, arg2, ...);
```

Elle affiche le texte contenu dans la chaîne de gauche à droite. Lorsqu'un caractère `%` est rencontré, une séquence de format est interprétée et utilise les arguments correspondants.

Les formats les plus utilisés sont :

- `%c` : caractère
- `%d` : entier (int, char signé, short)
- `%u` : entier non signé
- `%o` : entier en octal
- `%x` (`%X`) : entier en hexadécimal
- `%ld`, `%lu` : long int, long unsigned int
- `%f` : float (6 chiffres après la virgule par défaut)
- `%e` (`%E`) : float en notation scientifique
- `%g` : float, écriture compacte
- `%s` : chaîne de caractères

Caractères spéciaux :

Caractère	Signification	Code ASCII
`\n`	Retour à la ligne	10
`\t`	Tabulation horizontale	9
`\"`	Guillemets	34
%%	Pourcentage	37

Pour la lecture des données, utilisez `scanf` :

```
scanf("chaîne de format", &var1, &var2, ...);
```

Chaque séquence `%` indique le type de donnée à lire. Quelques formats courants :

- `%c` : caractère
- `%d` : entier
- `%u` : entier non signé
- `%o` : entier octal
- `%f` : nombre réel 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:

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.

Exercice 6:

- 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 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:

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);
}
```