POLYTECH PARIS-SUD

Et3 - Tronc commun

Programmation C

TP 1 : Introduction

1 Objectifs :

— Prendre en main la chaîne de compilation
— Maîtriser les interactions avec l'utilisateur : `scanf`/`printf`
— Savoir repérer et corriger des erreurs avec un débogueur

2 « Hello World ! »

1. Écrivez un programme simple qui affiche dans le terminal « Hello World ! ».
2. Améliorez le programme en intégrant des variables. Pourquoi pas un message incluant une date (3 variables entières) ou le prix au kilo d'un produit, une certaine quantité de ce produit et son prix ?
3. Noël n'est pas encore pour maintenant, mais écrivez un programme qui dessine un sapin dans le terminal.

```
     *
    ***
   *****
    ***
  *******
   *****
 *********
***********
    |||
    |||
```

4. Allez sur le net et chercher un dessin en ASCII qui vous plaisent. Puis écrivez un programme qui imprimera le dessin ASCII dans le terminal.
5. Écrivez une fonction d'affichage `void print_hello()` dans laquelle vous affichez une salutation. Appelez cette fonction depuis la fonction `main`.
   N.B. Si vous mettez la fonction d'affichage après la fonction `main`, il est nécessaire de mettre le prototype de la fonction au début de votre fichier.
6. Déplacez la fonction `void print_hello()` vers un nouveau fichier `affichage.c` qui se situe dans le même dossier que votre fichier actuel. Créez un fichier `affichage.h` dans lequel vous mettez le prototype de cette fonction. N'oubliez pas d'inclure ce fichier .h au début du fichier dans lequel se situe la fonction `main`.

3 Qui a dit bug ? Pas avec GDB !

Le programme suivant cherche à afficher les multiples de 3 jusqu'à la valeur `max`.

```c
#include <stdio.h>

void print_my_value(int value) {
  printf("Valeur suivante = %d\n", value);
}

int main() {
    int max=10, i=1;

    /* afficher les multiples de 3 jusqu'a la valeur de max */
    while (i != max) {
      print_my_value(i); /* Appel a la fonction d'affichage ci-dessus */
      i = i * 3;
    }
    return 0;
}
```

1. Copiez le code, compilez-le et exécutez-le [1].
2. Nous allons ensuite essayer de comprendre le code avec GDB :
   - Compilez et lancez le programme avec GDB.
   - Exécutez le programme avec GDB en entrant la commande `run`.
   - Placez un « breakpoint » juste après le début de la boucle `while`. Re-exécuter avec `run`. Quelle est la valeur de la variable `i` ?
   - Supprimez ce « breakpoint » et en créez un nouveau. Ce nouveau « breakpoint » doit arrêter le programme *uniquement* lorsque la variable i vaut 81.
   - Avancez dans le programme avec la commande `step` jusqu'à ce que la variable `i` vaille 27. Maintenant recommencez et faites la même chose avec la commande `next`.
   - Recommencez l'exécution en retapant la commande `run` (assurez-vous de garder le breakpoint au début de la boucle `while`). Quand le programme s'arrête, insérez un « watchpoint » sur la variable `i`. Vous pouvez continuer l'exécution avec la commande `continue`. Le programme doit s'arrêter à chaque fois que la valeur de la variable `i` change. Observez les anciennes et les nouvelles valeurs de `i` en suivant ses changements avec la commande `continue`.
   - Quelle erreurs ont été commises dans l'écriture du programme par rapport à son objectif ? Proposez une solution.

4 Entrée utilisateur et affichage

1. Écrivez un programme pour demander à l'utilisateur son âge. Affichez à l'écran son âge et ses années de naissance possibles. Ex : « Vous avez 16 ans. Vous êtes donc né(e) en 1999 ou 2000 ».
2. Écrivez un programme pour demander à l'utilisateur un nombre flottant. Affichez-le trois fois :
   - avec 5 chiffres après la virgule, en affichant aussi les zéros finaux.
   - avec un minimum de 7 caractères affichés (avant et après la virgule compris), 2 chiffres après la virgule et où les caractères ajoutés au début de l'affichage du flottant sont des espaces.
3. Affichez le motif suivant en 4 appels à `printf`. Vous utiliserez uniquement des `int`.
   ```
   1
   01
   001
   0001
   ```
4. Déclarez deux chaînes de caractères `player1` et `player2`. Demandez à l'utilisateur de rentrer le nom du joueur 1, lisez l'entrée de l'utilisateur avec `scanf` et affichez « Hello », suivi de son nom. Ensuite demander le nom du deuxième joueur et affichez « Hello », suivi de son nom. Quelles sont les limites de la fonction `scanf` ? Par exemple, essayez avec toutes sortes de caractères et essayez avec des noms de longueurs différentes [2].

1. Pour arrêter un programme en cours d'exécution, tapez Ctrl-C, y compris quand vous l'avez lancé sous GDB avec `run`.
2. La fonction `scanf` pose un certain nombre de problèmes. Vous verrez plus tard la fonction `gets`.

5. Écrivez un programme qui demande l'âge de l'utilisateur et le félicite d'avoir vécu `age` années.
6. Modifiez le programme précedent pour rendre possible les entrées non entières.

5 Les types

Soit `int x_0 = 3, x_1 = 5, x_2 = 9, x_3 = 1;`
Calculer la moyenne en mettant le résultat dans `int moy`.
Comment faire pour avoir un résultat exact ?

6 Passage en caisse

Supposez que vous voulez simuler un passage en caisse dans un magasin.

Ecrivez un programme qui :

1. demande à l'utilisateur le prix de 4 objet achetés (demandez les prix séparément, p. ex. « Veuillez entrer le prix de l'achat 1 : ... ») et lire l'entrée utilisateur pour chaque prix ;
2. affiche le prix total des 4 achats que l'utilisateur doit payer ;
3. demande à l'utilisateur le montant qu'il donne et lire cette somme (qui doit être supérieure ou égale à la somme à payer) ;
4. affiche la monnaie rendue à l'utilisateur ;
5. affiche la monnaire à rendre en spécifiant le nombre de chaque billet/pièce tel que le nombre total de billets et pièces soit minimal. Par exemple, un rendu de 27€ donnerait « 2 billets de 20 euros, 1 billet de 5 euros et une pièce de 2 euros ». On pourra utiliser l'opérateur modulo (%).