Fiche n° 7 de TP

Instructions itératives (2)

Objectifs : manipulation des instructions itératives `for` et `while` ; lecture sur l'entrée d'une suite de valeurs de longueur indéterminée.

Prérequis : syntaxe des fonctions `scanf` et `printf`.

Travail minimum : exercices 1, 4 (questions 1 et 2) et 3.

Exercice 1

1) Que fait la ligne `printf("%*d", 10, 10);` ?

2) Écrire un programme demandant un entier à l'utilisateur et affichant tous les entiers compris entre 1 et l'entier saisi de façon à ce que les chiffres des unités se trouvent sur la diagonale (comme dans l'exercice 1 de la fiche TP2). Donnez l'invariant de boucle en commentaire.

3) Que faut-il changer pour que ce soit le chiffre le plus à gauche de chaque nombre qui soit sur la diagonale ?

```
1
 2
  3
   4
    5
     6
      7
       8
        9
         10
          11
          ...
```

Indication : Vous pourrez écrire une fonction `nb_digits` qui renvoie le nombre de chiffres d'un entier passé en paramètre. Si cette fonction possède une boucle, n'oubliez pas de donner son invariant.

4) Pour que le résultat reste lisible à l'écran, on demande à ce que l'entier saisi soit compris entre 1 et 120. Si l'entier donné par l'utilisateur n'est pas dans cet intervalle, alors le programme doit redemander l'entier à l'utilisateur.
Si l'utilisateur ne rentre pas un entier, le programme doit s'arrêter en renvoyant `EXIT_FAILURE`.
Modifiez votre programme pour qu'il ait ce comportement.

Exercice 2

On considère le jeu programmé dans le fichier `casino.c`.

1) Téléchargez le et examinez le.

2) Faites le fonctionner. Est-il facile de gagner ? Expliquez et proposez une solution pour rendre le jeu plus compliqué.

3) Pour rendre le jeu un peu plus intéressant, on voudrait qu'il ne s'arrête pas dès que le joueur a trouvé le nombre mais qu'il puisse continuer tant qu'il lui plait. Lorsque le nombre choisi par le joueur est le même que celui choisi par l'ordinateur, le joueur reçois `GAIN` euros dans sa cagnote mais l'ordinateur ne lui indique pas qu'il l'a trouvé.
Il peut continuer à jouer ou s'arrêter à chaque étape.
Modifiez le programme afin qu'il prenne en compte les nouvelles règles du jeu.

4) Modifiez une nouvelle fois le programme afin qu'à la fin du jeu, il demande au joueur si il veut recommencer. Si le joueur répond par `'o'` alors la partie recommence avec un nouvel entier à rechercher. Les gains doivent être cumulés et affiché après chaque partie.

5) Il reste peut-être un problème lors de la saisie. Que se passe-t-il si vous écrivez plusieurs caractères à la place d'un nombre ? Si vous n'avez pas pensé à cette possibilité, il est possible que le programme affiche `Voulez-vous recommencer` et termine sans attendre votre réponse

```
Entrez la tentative 1 :
5
Entrez la tentative 2 :
6
Entrez la tentative 3 :
7
Entrez la tentative 4 :
rrr
Vous avez perdu 6 euros!.
Voulez-vous rejouer?Au revoir


------------------
(program exited with code: 0)
```

Expliquez ce phénomène. Corrigez votre programme afin de ne plus avoir ce comportement.

Exercice 3

1) Écrire un programme `gen_numbers` demandant un nombre entier `n` strictement positif et affichant, sur l'entrée standard, `n` nombres entiers compris entre 0 et 20 séparés par des ` | `.
Par exemple :

```
5
11 | 3 | 10 | 1 | 0 |
```

Aucun autre caractère ne doit être affiché sur la sortie standard (pas de phrase ou de question), sauf, éventuellement, pour la gestion des erreurs.

2) Tapez `./gen_numbers > notes.txt` dans un terminal. Que se passe-t-il ?

3) Écrire un programme `max` permettant de lire des entiers sur l'entrée standard sous la forme `n1 | n2 | n3 |.... | nk` et qui calcule et affiche le maximum de ces entiers.

4) Donnez une ligne de commande permettant de tester ce programme sur le fichier créé à la question 2).

Exercice 4

Écrivez un programme qui prend en entrée un entier positif, puis qui l'affiche à l'envers. Par exemple si l'entier saisi est 1234, le programme doit afficher 4321.

Exercice 5

1) Écrivez une fonction `void quo_rem(int a, int b, int *q, int *r)` qui prend en paramètres deux entiers a et b, puis qui affecte au contenu de la variable pointée par `q` le quotient de la division euclidienne de a par b, et à la variable pointée par `r` son reste.
Contraintes : Vous devrez utiliser une boucle en vous inspirant des formules suivantes :

$$\text{quotient}(a,b) = \begin{cases} 0 & \text{si } 0 \leqslant a < b, \\ 1 + \text{quotient}(a-b,b) & \text{sinon} \end{cases} \text{, et}$$

$$\text{reste}(a,b) = \begin{cases} a & \text{si } 0 \leqslant a < b, \\ \text{reste}(a-b,b) & \text{sinon.} \end{cases}$$

Par exemple, pour calculer le quotient et le reste de 23 et 5, vous ferez les étapes suivantes :

```
a = 23, b = 5, q = 0
a = 18, b = 5, q = 1
a = 13, b = 5, q = 2
a = 8, b = 5, q = 3
a = 3, b = 5, q = 4
```

donc comme $a < b$ nous pouvons conclure et nous obtenons `q = 4, r = 3` c'est à dire que `23 = 5 x 4 + 3`.

2) Écrivez ensuite un programme qui demande à l'utilisateur deux entiers positifs a et b en entrée puis qui affiche le quotient et le reste de a et b. On fera bien attention que les entiers saisis soient positifs ou nuls et on redemandera à l'utilisateur de corriger les saisies tant que celui-ci se trompera.

Voici une trace d'utilisation du programme :

```
Saisissez deux entiers naturels (positifs ou nuls), nous allons
calculer leur reste et leur quotient
-3 -2
Erreur ! Vous devez saisir deux entiers naturels (positifs ou nuls). Recommencez !
-3 1
Erreur ! Vous devez saisir deux entiers naturels (positifs ou nuls). Recommencez !
15 3
15 = 3 x 5 + 0
```