Université Paris-Nord — L1
Institut Galilée — Année 2011-2012
Licence 1 - 1er semestre

Travaux dirigés 6 : type booléen en C ; structures de contrôle *for* et *while*

1 Évaluation d'expressions booléennes

Soit le programme suivant :

```
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

#define FALSE 0
#define TRUE 1

/* Declaration de fonctions utilisateurs */

int main()
{
    int beau_temps = TRUE;
    int pas_de_vent = FALSE;

    printf("%d\n",beau_temps && pas_de_vent);
    printf("%d\n",beau_temps || pas_de_vent);
    printf("%d\n",! beau_temps || pas_de_vent);
    printf("%d\n",! (! beau_temps || pas_de_vent) == (beau_temps && ! pas_de_vent));

    return EXIT_SUCCESS;
}

/* Definition de fonctions utilisateurs */
```

1. Qu'affiche le programme ?
2. Modifiez le programme pour qu'il demande la valeur des booléens à l'utilisateur (0 pour FALSE, sinon TRUE).

2 Boucles *for* ou *while* ?

Ces boucles ont exactement la même sémantique et on peut facilement réécrire l'une en l'autre. Par convention, on préfère utiliser la boucle *for* lorsque l'on connaît le nombre d'itérations à l'avance ; on utilise *while* dans le cas contraire, lorsque le nombre d'itérations n'est pas connu à l'avance. Par exemple, pour faire la somme des éléments d'un tableau on utilisera *for* et pour trouver un élément dans un tableau, ne sachant pas où il se trouve, on choisira *while*. L'intérêt de respecter cette convention est que la lecture d'un programme est facilité en indiquant à quoi sert la boucle.

Résoudre les problèmes suivants en utilisant soit *for*, soit *while*.

2.1 Test d'égalité entre tableaux

Écrire un programme qui teste si deux tableaux de même `TAILLE` (constante symbolique) ont les mêmes données.

2.2 Nombre d'occurrences dans un tableau

Écrire un programme qui fait initialiser un tableau de taille `TAILLE` par l'utilisateur, demande à l'utilisateur un entier, et affiche le nombre d'occurrences de l'entier dans le tableau. Des exemples de sorties sont les suivants :

```
Saisissez 4 entiers : -2 -2 -2 3
Compte le nombre d'occurrences de quel entier ?
-2
Il y a 3 occurrences de -2 dans le tableau.

Saisissez 4 entiers : -2 -2 -2 3
Compte le nombre d'occurrences de quel entier ?
3
Il y a 1 occurrences de 3 dans le tableau.

Saisissez 4 entiers : -2 -2 -2 3
Compte le nombre d'occurrences de quel entier ?
0
Il y a 0 occurrences de 0 dans le tableau.
```

2.3 Élévation à la puissance

Écrire un programme qui demande à l'utilisateur d'entrer deux nombres entiers x et $n \geq 0$ puis calcule x^n et affiche le résultat.

2.4 Test de primalité

Écrire un programme qui demande à l'utilisateur d'entrer un nombre entier positif n, teste si n est premier puis affiche le résultat.

3 Carré d'étoiles

Écrire un programme qui demande à l'utilisateur d'entrer un nombre entier positif n et affiche un carré creux d'étoiles de côté n. Exemple (l'utilisateur entre 4) :

```
n? 4
****
*  *
*  *
****
```

Université Paris-Nord — L1
Institut Galilée — Année 2011-2012
Licence 1 - 1er semestre

Travaux pratiques 6 : évaluation d'expressions booléennes ; la structure de contrôle "while"

L'objectif de ce TP est de vous familiariser avec les expressions booléennes et leur utilisation avec la structure de contrôle "while".

1 Le nombre secret

Nous voulons programmer le jeu du nombre à découvrir. Le joueur doit deviner un nombre secret choisit par l'ordinateur entre 0 et `NB_MAX` (une constante du programme). S'il propose un nombre trop grand, l'ordinateur lui répond "Plus petit", s'il propose trop petit, l'ordinateur lui répond "Plus grand". Dans ces deux cas, il est invité à proposer un autre nombre. Le jeu s'arrête quand il devine juste. Un exemple d'exécution de ce jeu pourrait être :

```
Votre choix ?
8
Plus petit.
Votre choix ?
4
Plus petit.
Votre choix ?
2
Vous avez trouvé le nombre secret.
```

1. Proposez un algorithme en français pour le jeu.
2. Traduisez-le en langage C et exécutez-le.
3. Pourquoi préférez-vous une boucle `while` ici ?

Pour rendre le jeu intéressant, l'ordinateur doit choisir le nombre secret *au hasard*. La librairie C standard propose des fonctions renvoyant des nombres pseudo-aléatoires [1] déclarées dans `<stdlib.h>`. L'ordinateur va utiliser la fonction : `int rand();` qui renvoie un nombre pseudo-aléatoire entier entre 0 et la constante `RAND_MAX` (égale à 2147483647) inclus. Pour renvoyer un nombre pseudo-aléatoire entre 0 et `NB_MAX`, `NB_MAX` inclus (`NB_MAX` << `RAND_MAX`), il suffit de calculer le reste de la division entière de `rand()` par `(NB_MAX + 1)`, c'est-à-dire le nombre renvoyé par rand() modulo `(NB_MAX + 1)` (opérateur % en C). rand vient de random qui veut dire aléatoire en anglais.

Un exemple de programme illustrant l'utilisation de rand pour engendrer un nombre pseudo-aléatoire est le suivant :

```
#include <stdlib.h> /* EXIT_SUCCESS, rand, srand */
#include <time.h> /* time */
```

1. http ://fr.wikipedia.org/wiki/Générateur_de_nombres_pseudo-aléatoires

```
#define NB_MAX 15 /* nombre secret entre 0 et NB_MAX inclus */

int main()
{
    int nombre_secret; /* nombre secret à deviner */

    /* initialisation du générateur de nombres pseudo-aléatoires */
    srand(time(NULL)); /* à ne faire qu'une fois */

    /* tirage du nombre secret */
    nombre_secret = rand() % (NB_MAX + 1); /* entre 0 et NB_MAX inclus */

    /* manche joueur ... */

    return EXIT_SUCCESS;
}
```

2 Évaluation d'expressions booléennes

Une table de vérité donne la valeur d'une ou de plusieurs expressions booléennes, construits à partir de variables et d'opérateurs booléens, en fonction des valeurs des variables booléennes. Un exemple de table de vérité pour l'expression *a OU b* est donné dans le tableau ci-dessous (avec F : faux et V : vrai) :

a	b	a OU b
F	F	F
F	V	V
V	F	V
V	V	V

Écrire un programme qui demande à l'utilisateur les valeurs de deux variables booléennes `a` et `b` et qui affiche à l'écran la ligne correspondante de la table de vérité de l'ensemble des expressions : a ET b, a OU b, NON a, NON b, NON a ET b. Cette ligne à afficher (qui aura 7 colonnes) est déterminée par la valeur des deux variables a et b. Pour une indentation correcte des colonnes, vous pouvez utiliser dans printf `‘‘\t’’` qui affiche une tabulation.

Voici deux exemples de sortie :

```
entrez deux valeurs booleennes : 1 0
a       b       a ET b  a OU b  NON a   NON b   NON a ET b
1       0       0       1       0       1       0

entrez deux valeurs booleennes : -12 0
a       b       a ET b  a OU b  NON a   NON b   NON a ET b
-12     0       0       1       0       1       0
```