Travaux dirigés 3 : expressions booléennes et structures de contrôle *if else*, *for* et *while*.

1 Évaluation d'expressions booléennes

Qu'affiche le programme suivant (répondre sans utiliser l'ordinateur) ?

```
1  #include <stdlib.h> /* EXIT_SUCCESS */
2  #include <stdio.h> /* printf */
3  #include <stdbool.h> /* booleens */
4
5  int main()
6  {
7      bool beau_temps = true;
8      bool pas_de_vent = false;
9
10     printf("%d\n", beau_temps && pas_de_vent);
11     printf("%d\n", beau_temps || pas_de_vent);
12     printf("%d\n", !(beau_temps) || pas_de_vent);
13     printf("%d\n",!(!(beau_temps) || pas_de_vent) == (beau_temps && !(pas_de_vent)));
14
15     return EXIT_SUCCESS;
16 }
```

Correction.

```
0
1
0
1 /* toujours vrai : théorème de De Morgan : NON (a OU b) = NON a ET NON b */
```

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.

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

2.1 Élévation à la puissance

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

Correction. *Trivial (for).*

2.2 Somme d'une série d'entiers positifs saisis par l'utilisateur

Écrire un programme `add.c` qui lit des entiers saisis par l'utilisateur et dès que l'utilisateur saisit un zéro (ou un nombre négatif), affiche le total des entiers positifs saisis jusque là.

2.3 Test de primalité

Donner **l'algorithme** d'un programme qui demande à l'utilisateur d'entrer un nombre entier positif n et teste si n est premier. Utiliseriez vous un *for* ou un *while* ?

Correction. _ *L'idée est qu'il faut le faire avec un while plutôt qu'un for. L'algorithme doit s'énoncer à peu près comme ceci :*

- *saisie de n*
- *Pour chaque entier d entre* 1 *et n exclus :*
  -
  - *Si d divise n, terminer la boucle et afficher n n'est pas premier*
- *Si la boucle précédente s'exécute juste qu'au bout sans avoir rencontré de diviseur de n, afficher que l'entier n est premier.*

La traduction d'une boucle de parcours pouvant avoir une fin prématurée se fera plutôt avec un `while`*. On ne présente pas le* `break` *dans ce cours.*

```
...
int main()
{
  bool premier = true;
  int n;
  int d = 2;

  /* saisie */
  printf("n?");
  scanf("%d",&n);

  /* test de primalite */
  while (premier && (d < n))
  {
     if (n % d == 0)
     {
        premier = FALSE;
     }
     d = d + 1;
  }

  /* affichage */
  if (premier)
  {
    printf("%d est premier\n", n);
  }
  else
  {
    printf("%d n'est pas premier, car divisible par %d\n", n, d - 1);
  }


  return EXIT_SUCCESS;
}
```

---

3 Le nombre secret

Programmer le jeu du nombre à découvrir. Le joueur doit deviner un nombre secret choisi 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 un programme C `deviner.c` et testez-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). Le nom `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 :

```
1  #include <stdlib.h> /* EXIT_SUCCESS, rand, srand */
2  #include <stdio.h> /* printf */
3  #include <time.h> /* time */
4
5  #define NB_MAX 15 /* nombre secret entre 0 et NB_MAX inclus */
6
7  int main()
8  {
9      int nombre_secret; /* nombre secret à deviner */
10
11     /* initialisation du générateur de nombres pseudo-aléatoires */
12     srand(time(NULL)); /* à ne faire qu'une fois */
13
14     /* tirage du nombre secret */
15     nombre_secret = rand() % (NB_MAX + 1); /* entre 0 et NB_MAX inclus */
16
17     /* exploitation du secret */
18     printf("Tu ne devineras jamais que mon secret est %d\n", nombre_secret);
19
20     return EXIT_SUCCESS;
21 }
```

Correction.

```
#include <stdlib.h> /* EXIT_SUCCESS, rand, srand */
#include <stdio.h> /* printf, scanf */
#include <time.h> /* time */
#include <stdbool.h>

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

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

```
/* declaration de fonctions utilisateurs */

int main()
{
    int nombre_secret; /* nombre secret à deviner */
    int choix; /* choix de l'utilisateur pour le nombre secret */
    bool trouve = false; /* true si trouvé nombre secret */

    /* 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 */
    while(!trouve) /* pas trouvé nombre secret */
    {
        /* demande nombre à l'utilisateur */
        printf("Votre choix (nombre entre 0 et %d) ?\n",NB_MAX);
        scanf("%d",&choix);

        if(choix == nombre_secret) /* trouvé */
        {
            trouve = true;
        }
        else /* pas trouvé */
        {
            /* donne indice */
            if(choix > nombre_secret)
            {
                printf("Plus petit.\n");
            }
            else
            {
                printf("Plus grand.\n");
            }
        }
    }
    /* trouvé nombre secret */

    printf("Vous avez trouvé le nombre secret.\n");

    return EXIT_SUCCESS;
}

/* Definition de fonctions utilisateurs */
```

---

4 Exercices complémentaires

4.1 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
****
*  *
*  *
****
```

Correction.

```
 double for et un if comme ça :
if ( (i == 0) || (j == 0) || (i == n - 1) || (j == n - 1) )
{
  printf("*");
}
else
{
  printf(" ");
}
```