Algorithmique et langage C

Période 2

Boucles, hasard et temps, fonctions

Objectifs :

- Savoir écrire et analyser des programmes utilisant des boucle while, do/while et for
- Maîtriser la notion de fonction
- Savoir utiliser le générateur pseudo-aléatoire : fonctions rand() et srand()
- Découvrir les notions de "temps" en informatique : fonctions time() et clock()

A chaque fois qu'on demande une fonction il faut écrire une fonction main() qui l'appelle (une ou plusieurs fois) pour montrer son fonctionnement.

1/ Une boucle ☼

Écrire le programme qui affiche N fois « Bonjour » à l'écran. N sera saisit au clavier.

2/ les racines ☼

Écrire un programme qui affiche les racines carrées des nombres entier entre 0 et 10.

3/ Quelques fonctions simples ☼

3.1/ Écrire la fonction **`cube`** qui retourne n élevé au cube : **`int cube(int n) ;`**

3.2/ Écrire la fonction **`max`** qui retourne le plus grand des deux entiers passés en arguments :
`int max (int a, int b) ;`

3.3/ Écrire la fonction **`aleatoire`** qui retourne un entier pseudo aléatoire compris entre min et max inclus : **`int aleatoire (int min , int max) ;`**

4/ Fonctions utilisant des boucles ☼

4.1/Écrire la fonction **sum** qui retourne la somme des n premiers entiers avec n passé en argument :
int sum (int n) ; (on pourrait aussi le faire sans boucle en utilisant n*(n+1)/2)

4.2/ Écrire la fonction **produit** qui en utilisant une boucle retourne le produit de deux entiers :
int produit(int a , int b) ;

4.3/ Écrire la fonction **power** qui retourne x élevé à la puissance n (avec n entier relatif) :
float power (float x, int n) ;
4.4/ Écrire la fonction **factorielle** qui retourne n! : **int factorielle (int n) ;**

5/ Des petits dessins ☼

5.1/ Afficher à l'aide d'une boucle un rectangle d'étoiles dont le nombre de lignes est variable et le nombre colonnes est fixe (10) : **void rectangle10c (int lignes) ;**
5.2/ Comme ci dessus mais le nombre de colonnes est variable :
void rectangle (int lignes , int colonnes) ;
5.3/ Afficher ce motif avec un nombre de ligne variable :
void demiSapin(int lignes) ; ☼☼

```
*
***
*****
******
```

5.4/ Afficher ce motif avec un nombre de ligne variable : **void sapin(int lignes) ;** ☼☼

```
   *
  ***
 *****
*******
```

6/ Ce nombre est-il premier ? ☼☼☼

6.1/ Écrire la fonction premier qui retourne 1 si l'entier n passé en argument est un nombre premier et 0 sinon : **int premier(int n) ;**

6.2/ En utilisant la fonction précédente afficher la liste des nombres premiers inférieurs à 1000.

7/ Résistances en série et en parallèle ☼ ☼

7.1/ Écrire un programme qui calcule la résistance équivalente à un nombre quelconque de résistances câblées en série. La saisie s'arrête et le résultat s'affiche lorsque l'on rentre une valeur négative.
Votre programme doit utiliser seulement deux variables.

Exemple : Tapez la valeur de R : 10
Tapez la valeur de R : 20
Tapez la valeur de R : 5
Tapez valeur de R : -1
Résultat : 35

7.2/ Même problème pour des résistances câblées en parallèle.

8/ Un nombre à deviner ☼

Le programme tire un nombre entier au hasard entre 1 et 100 et essaye de le faire deviner à l'utilisateur en 10 essais maximum. Après chaque essai l'ordinateur indique "trop grand" ou "trop petit" au joueur ainsi que le nombre d'essais restants. Si au bout de 10 essais, le joueur n'a pas trouvé il a perdu et l'ordinateur lui indique la solution.

Conseil : Pour la mise au point, affichez le nombre tiré un début du programme.

9/ Le jeux des multiplications ☼ ☼

Le programme affiche une multiplication (au hasard deux chiffres par un chiffre) et attend que le joueur entre le résultat du calcul. Le programme affiche le temps mis pour répondre. Si le résultat est juste le programme propose une autre multiplication. Si le résultat est faux le programme s'arrête .

10/ fibonacci ŠŠŠŠ

Écrire la fonction **`fibonacci`** qui retourne le énième terme de la suite de fibonacci.

11 / Calcul de la racine carrée par la méthode de Newton ŠŠŠŠ

La suite définie par $x_0 = 1 , x_{n+1} = \frac{1}{2} \cdot (x_n + \frac{b}{x_n}) \quad avec\ b \geq 0$ converge vers la racine carré de b.

Écrire une fonction qui calcule la racine carré d'un nombre x à epsilon près en utilisant cette méthode (appelée méthode de Newton).
Travailler avec des variables de type double.