Programmation en C – Les fonctions – réponses au QCM spé ISN – TS

Question 1 : Un prototype de fonction :

- **permet d'indiquer le nombre et le type de paramètres pris en entrée de la fonction.**
- **permet d'indiquer le type de variable de sortie de la fonction**
- se place dans la fonction principale main
- contient le bloc d'instructions exécutées par la fonction lors de son appel

Question 2 : On considère le prototye suivant :
int fonction (int * parametre1, int parametre2) ;

- **La fonction prend deux paramètres en entrée.**
- La fonction en renvoie rien.
- **parametre1 est passé par adresse.**
- parametre2 est durablement modifié lors de l'appel de la fonction.

*parametre1 permet d'accéder à l'adresse de parametre1
parametre2 est passé par valeur : on utilise une copie de sa valeur, il n'est pas durablement modifié.

Question 3 : Pour qu'une fonction ne renvoie rien, on utilise le type de variable :

- int
- float
- char
- **void**

void signifie *vide* en anglais

Question 4 : On considère la fonction suivante :

```
int fonction (int parametre) {
    return parametre * parametre ;
}
```

Si on appelle la fonction avec en paramètre une variable nommée *truc* et valant 8 :

- **alors truc vaudra toujours 8 après l'appel de la fonction**
- alors truc vaudra 64 après l'appel de la fonction
- alors la fonction ne reverra rien
- alors la fonction renverra la valeur 8
- **alors la fonction renverra la valeur 64**

parametre est passé par valeur, donc *truc* n'est pas durablement modifié par l'appel de la fonction. Il valait 8 avant l'appel et garde cette valeur ensuite.
La fonction renvoie un entier, de valeur 8*8 = 64.

Question 5 : On considère le code suivant :

```
#include <stdio.h>
int ma_fonction (int * tab, int dim) ;
int main() {
    int entier, resultat = 9 ;
    int tableau[10] = {1, 3, 4, 3, 2, 0, 3, 7, 3, 5} ;
    resultat = ma_fonction (tableau, 10) ;
    printf("%d", resultat) ;
    return 0 ;
}
```

```
// suite du code :
int ma_fonction (int * tab, int dim) {
    int i, compteur = 0 ;
    for (i = 0; i < 10; i = i + 1) {
        if (tab[i] == 3) {
            compteur = compteur + 1 ;
        }
    }
    return compteur;
}
```

- La fonction modifie la dimension du tableau
- En fin d'appel de la fonction, *compteur* vaut 5
- **Avant l'appel de la fonction, *resultat* vaut 9**
- Après l'appel de la fonction, *resultat* n'a pas changé, il vaut toujours 9
- **Après l'appel de la fonction, *resultat* vaut 4**

Question 6 : On considère la fonction suivante :

```
void fonction (int * tableau, int dimension1, int dimension2) {
    int i, j;
    for (i = 0 ; i < dimension1 ; i = i + 1) {
        for (j = 0 ; j < dimension2 ; j = j + 1) {
            printf("%d", tableau[i][j]);
        }
        printf("\n");
    }
}
```

- **C'est une fonction qui ne renvoie rien.**
- Cette fonction permet de remplir un tableau à deux dimensions.
- **Cette fonction permet d'afficher le contenu d'un tableau à deux dimensions.**
- Le tableau sera durablement modifié