Fiche d'Exercices en C

1. Variables

Exercice :
Déclare trois variables de types différents (`int` , `float` , `double`), affecte-leur des valeurs et affiche-les avec `printf` .

```c
int main() {
    // Déclarer une variable entière, flottante, et double
    // Afficher les valeurs
    return 0;
}
```

2. Les types et les conversions implicites

Exercice 1 :
Écrire un programme qui additionne une variable `int` et une variable `float` . Afficher le résultat.

```c
int main() {
    // Déclarer une variable int et une variable float
    // Additionner les deux et afficher le résultat
    return 0;
}
```

Exercice 2 :
Multiplier une variable `int` par un `double` , puis observer la conversion implicite.

```c
int main() {
    // Déclarer une variable int et une variable double
    // Multiplier les deux et afficher le résultat
    return 0;
}
```

3. Deviner le nombre d'instructions

Une instructions est une action effectué par le processeur.
Les instructions sont repérable par la présence d'opérateur comme :

- `+ - * / %`
- `=`
- `== != < > <= >=`
- `()` pour l'appel de fonction comme `printf()`
- `[]` pour les tableaux
- `int float double char` qui sont des type de déclaration de variable qui alloue un bloc mémoire.

Si vous comptez le nombres d'opérateurs comme ceux là vous connaitrez donc le nombres d'instrcutions.

Exercice :
Pour chaque extrait de code ci-dessous, devinez combien d'instructions sont exécutées.

Extrait 1 :

```c
int a = 5, b = 2;
a = a + b;
b = b * a;
```

Extrait 2 :

```c
int x = 1;
x += 2;
if (x > 1) {
    x = 0;
}
```

4. Exercices simples sur `printf` et `scanf`

Exercice 1 :
Écrire un programme qui demande à l'utilisateur d'entrer un nombre entier et l'affiche ensuite.

```c
int main() {
    // Utiliser scanf pour lire un entier de l'utilisateur
    // Utiliser printf pour afficher cet entier
    return 0;
}
```

Exercice 2 :
Demander à l'utilisateur deux nombres à virgule flottante et afficher leur somme.

```c
int main() {
    // Utiliser scanf pour lire deux floats
    // Calculer et afficher la somme
    return 0;
}
```

5. Introduction aux conditions (`if`)

Définition rapide :
Le bloc `if` permet d'exécuter du code seulement si une condition est vraie.

Exercice 1 :
Écrire un programme qui demande un nombre entier à l'utilisateur et affiche s'il est positif ou négatif.

```c
int main() {
    // Utiliser scanf pour lire un entier
    // Utiliser if pour vérifier si le nombre est positif ou négatif
    return 0;
}
```

Exercice 2 :
Demander l'âge de l'utilisateur et afficher un message si l'âge est supérieur ou égal à 18.

```c
int main() {
    // Lire l'âge avec scanf
    // Utiliser if pour vérifier si l'âge est >= 18
    return 0;
}
```

6. La boucle `while`

Explication rapide :
La boucle `while` répète un bloc d'instructions tant qu'une condition est vraie.

Exercice 1 :
Écrire un programme qui affiche les nombres de 1 à 10 à l'aide d'une boucle `while` .

```
int main() {
    // Utiliser une boucle while pour afficher les nombres de 1 à 10
    return 0;
}
```

Exercice 2 :
Écrire un programme qui demande à l'utilisateur d'entrer un nombre et affiche la table de multiplication de ce nombre jusqu'à 10.

```
int main() {
    // Lire un entier avec scanf
    // Utiliser une boucle while pour afficher la table de multiplication
    return 0;
}
```

Exercice 3 :
Créer une boucle infinie qui affiche un message.

```
int main() {
    // Créer une boucle infinie qui affiche "Boucle infinie"
    return 0;
}
```

7. Explication des tableaux

Définition rapide :
Un tableau en C est une collection d'éléments du même type stockés de façon contiguë en mémoire.

L'accès aux éléments se fait avec un indice (`tableau[i]`). On peut utiliser `sizeof()` pour obtenir la taille totale du tableau.

Exercice 1 :
Déclarer un tableau de 5 entiers, affecter des valeurs, et les afficher avec une boucle `while` .

```
int main() {
    // Déclarer un tableau d'entiers
    // Affecter des valeurs
    // Afficher les valeurs avec une boucle for
    return 0;
}
```

Exercice 2 :
Utiliser `sizeof()` pour calculer la taille d'un tableau et le nombre d'éléments qu'il contient.

```
int main() {
    // Calculer la taille du tableau avec sizeof
    // Calculer le nombre d'éléments
    return 0;
}
```

8. La boucle `for`

Explication rapide :
La boucle `for` est une forme arrangée de la boucle `while` et s'utilise principalement lorsqu'on connaît à l'avance le nombre d'itérations.

Exercice 1 :
Écrire un programme qui affiche les nombres de 1 à 10 à l'aide d'une boucle `for` .

```
int main() {
    // Utiliser une boucle for pour afficher les nombres de 1 à 10
    return 0;
}
```

Exercice 2 :
Afficher les éléments d'un tableau d'entiers à l'aide d'une boucle `for` .

```c
int main() {
    // Utiliser une boucle for pour parcourir et afficher un tableau
    return 0;
}
```

9. Adresses mémoire et opérateur `&`

Explication rapide :
L'opérateur `&` permet d'obtenir l'adresse mémoire d'une variable.

Exercice 1 :
Déclarer deux variables entières et afficher leurs adresses mémoires à l'aide de `printf`.

```c
int main() {
    // Déclarer deux variables
    // Afficher leurs adresses mémoires avec printf
    return 0;
}
```

Exercice 2 :
Déclarer un tableau d'entiers et afficher les adresses de ses éléments avec `*(tableau + i)` plutôt que `tableau[i]`.

```c
int main() {
    // Déclarer un tableau
    // Utiliser *(tableau + i) pour afficher chaque élément
    return 0;
}
```