

8.5pts

- ```
1 #include <stdio.h>
2 int main () {
3 int i;
4 for (i = 1; i <= 5; i++) {
5 printf("Tour no%d\n", i);
6 }
7 return 0;
8 }
```

a) Relever une instruction d'initialisation : -----0,5pt

b) Donner le nombre de fois que cette boucle va s'exécuter : -----0,5pt

c) Dire ce que fait l'instruction `i++` : -----0,5pt

d) Expliquer ce que fait la ligne 7 : -----1pt

e) Réécrire les lignes 4 à 6 en remplaçant la boucle `for` par la boucle `do...while` 2pts

**5,5pts**

- Page 1 sur 2

b) Paramètre : -----  
-----

2) Donner une différence entre : 2pts

a) Paramètre effectif et paramètre formel :-----  
-----  
-----

b) Variable globale et Variable locale -----  
-----  
-----

3) Donner la syntaxe d'appel d'une fonction en Algorithmique 1,5pt

## **PARTIE 2 : EVALUATION DES COMPETENCES (6points)**

Ecrire un Algorithme qui :

- Lit 20 nombres entiers dans un Tableau
- Calcule la somme et la moyenne
- Puis les affiche.

# CORRIGÉ DE L'ÉPREUVE D'INFORMATIQUE

Collège Privé Laïc Mongo Beti

Année Scolaire 2025/2026 - Évaluation N°2

## PARTIE 1 : ÉVALUATION DES RESSOURCES (14 Points)

### Exercice 1 : Sur le Langage C (8.5 pts)

#### 1) Différence entre les bibliothèques `stdio.h` et `math.h` (1 pt)

- `stdio.h` (Standard Input Output) : Contient les déclarations des fonctions et macros relatives aux entrées/sorties standards (affichage à l'écran, saisie au clavier, gestion de fichiers).
- `math.h` : Contient les déclarations des fonctions mathématiques standards (calculs trigonométriques, exponentielles, racines carrées, puissances, etc.).

#### 2) Citer une fonction contenue dans chacune et donner son rôle (1 pt - 0.5 pt par bibliothèque)

- Dans `stdio.h` : `printf()` (sert à afficher du texte et des données formatées sur la console) ou `scanf()` (sert à lire des données saisies par l'utilisateur au clavier).
- Dans `math.h` : `sqrt(x)` (calcule la racine carrée de x) ou `pow(x, y)` (calcule x élevé à la puissance y).

#### 3) Syntaxe de déclaration d'un tableau en langage C (1 pt)

```
type_données nom_du_tableau[taille];
```

*Exemple : `int notes[20];`*

#### 4) Analyse du programme C (4.5 pts)

- Relever une instruction d'initialisation** (0.5 pt) :  
`i = 1;` (C'est la partie initialisation de la boucle `for`, à la ligne 4).
- Nombre de fois que la boucle va s'exécuter** (0.5 pt) :  
**5 fois** (pour les valeurs de `i` allant de 1 à 5).
- Ce que fait l'instruction `i++`** (0.5 pt) :  
Elle incrémente la valeur de la variable `i` de 1 à chaque fin d'itération (c'est le pas de la boucle).
- Ce que fait la ligne 7 `return 0;`** (1 pt) :  
Elle termine l'exécution de la fonction `main()` et retourne la valeur 0 au système d'exploitation, ce qui indique que le programme s'est exécuté sans erreur (succès).
- Réécrire les lignes 4 à 6 en remplaçant la boucle `for` par `do...while`** (2 pts) :

```
1 int i = 1; // Initialisation n cessaire avant la boucle
2 do {
3 printf("Tour n0%d\n", i);
4 i++; // Incr mentation
5 } while (i <= 5); // Condition de arr t
6
```

Listing 1 – Boucle `do...while`

**Exercice 2 : Sur Les Fonctions et les Procédures (5.5 pts)****1) Définir les expressions suivantes (2 pts - 1 pt chacune)**

- a) **Fonction** : C'est un sous-programme (un bloc d'instructions) qui effectue une tâche spécifique et qui **retourne un résultat** (une valeur) au programme qui l'a appelée.
- b) **Paramètre** : C'est une variable déclarée dans l'en-tête d'une fonction ou procédure, qui permet de recevoir des données (arguments) provenant de l'extérieur pour être traitées dans le corps de la fonction.

**2) Donner une différence entre (2 pts - 1 pt chacune)****a) Paramètre effectif et paramètre formel :**

- *Paramètre formel* : C'est la variable fictive définie dans l'en-tête de la fonction (elle sert de "placeholder").
- *Paramètre effectif (ou argument)* : C'est la valeur réelle ou la variable concrète passée à la fonction lors de son appel dans le programme principal.

**b) Variable globale et Variable locale :**

- *Variable globale* : Déclarée en dehors de toute fonction (au niveau du programme principal). Elle est accessible et modifiable par toutes les fonctions du programme.
- *Variable locale* : Déclarée à l'intérieur d'une fonction (ou d'un bloc). Elle n'existe et n'est visible que dans cette fonction ; elle est détruite à la fin de son exécution.

**3) Syntaxe d'appel d'une fonction en Algorithmique (1.5 pt) :**

NomDeLaFonction(argument1, argument2, ...)

*Si la fonction retourne une valeur qu'on souhaite récupérer, on écrit :*

`variable_resultat <- NomDeLaFonction(argument1, argument2, ...)`

**PARTIE 2 : ÉVALUATION DES COMPÉTENCES (6 points)**

Écrire un Algorithme qui lit 20 nombres entiers dans un Tableau, calcule la somme et la moyenne, puis les affiche.

```
Algorithme Calcul_Somme_Moyenne

// D clARATION des constantes et variables
Const
 N = 20
Variables
 i : entier
 somme : entier
 moyenne : reel
 Tab : tableau [1..N] de entiers

// Corps de l'algorithme
D but
 somme <- 0

 // 1. Lecture des 20 nombres entiers dans le tableau et calcul de la
 somme
 Pour i de 1 N faire
 Ecrire("Saisissez le nombre num ro ", i, " : ")
 Lire(Tab[i])
```

```
 somme <- somme + Tab[i]
 FinPour

 // 2. Calcul de la moyenne
 moyenne <- somme / N

 // 3. Affichage des r sultats
 Ecrire("La somme des nombres est : ", somme)
 Ecrire("La moyenne des nombres est : ", moyenne)
Fin
```

Listing 2 – Algorithme Calcul\_Somme\_Moyenne

*"La Perfection N'habite Pas Dans Les Hommes Mais Dans Leurs Intentions"*