MODULE 0 : PREMIERS PAS — PARLER À LA MACHINE

CAHIER DE L'ÉLÈVE : Théorie Illustrée, Laboratoires Guidés, Exercices, Devoirs & Aide-Mémoire C++

> **Bienvenue dans l'aventure olympique !** Tu n'as jamais écrit une seule ligne de code de ta vie ? C'est parfait ! Toutes les médaillées de l'Olympiade Européenne d'Informatique pour les Filles (EGOI) ont commencé exactement là où tu es aujourd'hui. Dans ce **Module 0**, nous partons de zéro absolu : tu vas découvrir comment réfléchit un ordinateur, écrire tes premiers programmes en **C++** (le langage roi des Olympiades), dialoguer avec le Juge Automatique, compiler ton code dans le **Terminal Linux** et sauvegarder tes solutions avec **Git**.

Leçon 1 : Qu'est-ce qu'un Algorithme, le Juge Olympique et le C++ ?

1.1 L'Ordinateur et la Recette de Cuisine (L'Algorithme)

Un ordinateur est une machine capable d'effectuer **100 millions d'opérations par seconde**, mais il n'a aucune initiative. Pour qu'il résolve un problème, tu dois lui donner une suite d'instructions ultra-précises, dans le bon ordre : c'est ce qu'on appelle un **algorithme**, exactement comme la recette de ta grand-mère pour préparer un *diri kole ak pwa* (riz collé aux pois). Si tu oublies d'écrire « allumer le réchaud », le riz ne cuira jamais !

- **Le Code Source (C++) :** Les humains parlent créole ou français, mais le processeur de l'ordinateur (le **CPU**) ne comprend que des `0` et des `1`. Nous écrivons donc nos recettes dans un langage intermédiaire ultra-rapide appelé **C++** (dans un fichier terminé par `.cpp`).
- **Le Compilateur (`g++`) :** C'est le traducteur officiel. Il lit ton fichier `solution.cpp`, vérifie qu'il n'y a aucune faute d'orthographe (syntaxe), et le traduit en langage machine exécutable à la vitesse de l'éclair.

1.2 Comment Fonctionne une Épreuve EGOI ? Le Juge Automatique

À l'EGOI, aucun professeur ne corrige ta feuille à la main. Tu envoies ton fichier C++ à un robot serveur appelé le **Juge Automatique** (comme sur *France-IOI* ou *CSES*). Le Juge injecte des nombres secrets à l'entrée de ton programme (`cin`), chronomètre ton code (souvent **1.0 seconde maximum**) et vérifie ta réponse affichée (`cout`).

Sigle Officiel	Nom du Verdict	Signification & Action à Faire
AC	Accepted (Accepté)	Bravo ! Ton programme a trouvé la bonne réponse sur tous les tests secrets dans le temps imparti.
WA	Wrong Answer	Ton programme marche mais donne un mauvais résultat sur un cas piège (ex: nombres négatifs ou très grands).
TLE	Time Limit Exceeded	Ton programme est trop lent (il a dépassé 1 seconde, soit environ 10^8 opérations) ou tourne en boucle infinie.
CE	Compilation Error	Faute de grammaire C++ : tu as oublié un point-virgule `;` ou une accolade `}` .

1.3 Ton Laboratoire C++ & Tes Premières Boîtes (Variables)

Tout programme C++ d'Olympiade commence par le même **squelette magique**. Chaque instruction se termine obligatoirement par un point-virgule `;` (qui joue le rôle du point à la fin d'une phrase en français).

```
#include <iostream>  // Importe la boîte à outils pour lire (cin) et afficher (cout)
#include <string>    // Importe la boîte à outils pour manipuler du texte
using namespace std; // Évite d'écrire "std::" devant chaque mot

int main() {
    // C'est ici, entre les deux accolades, que ton programme commence !
    cout << "Bonjour Team Haiti EGOI 2027 !" << "\n";
    return 0; // Signale à l'ordinateur que tout s'est bien terminé
}
```

En C++, une **variable** est un bocal de cuisine sur lequel tu colles une étiquette ET sur lequel tu dois indiquer **ce qu'on a le droit de ranger dedans** (son type). Voici les 5 types fondamentaux du Syllabus EGOI :

Type C++	Rôle en Français	Exemple de Code C++	Règle d'Or & Piège Olympique
`int`	Nombre entier standard	`int age = 15;`	Va jusqu'à **2 milliards** (2×10^9). Au-delà, la boîte déborde (*Overflow*) !
`long long`	Nombre entier géant	`long long pop = 5000000000LL;`	Va jusqu'à **9×10^{18}**. **Utilise toujours `long long` pour les grandes sommes !**
`double`	Nombre décimal (à virgule)	`double temp = 31.5;`	La virgule s'écrit toujours avec un **point** `.` !
`string`	Chaîne de texte	`string ville = "Jacmel";`	Toujours entouré de **guillemets doubles** `" "` (un seul caractère utilise `'a'`).
`bool`	Booléen (Vrai ou Faux)	`bool est_prete = true;`	Uniquement `true` (vrai) ou `false` (faux), tout en **minuscules**.

Pour **écouter** le Juge Automatique, on utilise `cin >> variable;` (les flèches vont vers la variable). Pour **parler** au Juge, on utilise `cout << valeur << "\n";` (les flèches vont vers l'écran).

```
string prenom;
int score_jour1, score_jour2;
cin >> prenom >> score_jour1 >> score_jour2; // Lit le prénom et les 2 scores envoyés
par le Juge
int total = score_jour1 + score_jour2;
cout << "Bravo " << prenom << ", ton total est : " << total << " points !\n";
```

Exercices d'Entraînement — Leçon 1 (À faire sans aide)

Exercice 1.1 (Convertisseur Gourdes → Dollars) : Complète le programme C++ ci-dessous pour qu'il lise un montant en Gourdes à virgule (`double montant_htg`) avec `cin` , le divise par `130.0` dans une variable `montant_usd` , et affiche le résultat avec `cout` .

```
// Écris ton code C++ ici : #include <iostream> using namespace std; int main() { return 0; }
```

Exercice 1.2 (Détective de Bugs) : Chacune des 4 lignes C++ ci-dessous contient une erreur qui bloque le compilateur ou fausse le calcul. Entoure l'erreur et réécris la ligne corrigée en dessous :

```
1) string pays = Haïti;
2) double taille = 1,68;
3) int reponse = 5 / 2;   // On voudrait obtenir 2.5 dans une boîte double !
4) cout >> "Victoire !" >> "\n"
```

```
// Écris tes 4 lignes corrigées ici : 1) 2) 3) 4)
```

Leçon 2 : Prendre des Décisions (`if / else`) et Répéter (Boucles)

2.1 Donner de l'Intelligence au Code : `if` , `else if` , `else`

Pour que ton programme prenne une décision, on utilise `if` (Si), `else if` (Sinon si) et `else` (Sinon). En C++, la question se met entre **parenthèses** `( )` et les actions à exécuter s'enferment entre **accolades** `{ }` . Pour combiner deux conditions, on écrit `&&` (ET logique) ou `||` (OU logique).

Piège mortel à éviter : Un seul signe égal `=` range une valeur dans la boîte (`x = 10;`). Deux signes égal `==` comparent deux valeurs pour poser une question (`if (x == 10)`).

```
int score;
cin >> score;
if (score >= 85) {
    cout << "Qualifiee Equipe Nationale A\n";
} else if (score >= 60) {
    cout << "Qualifiee Camp EGOI\n";
} else {
    cout << "Continue l'entrainement !\n";
}
```

2.2 Répéter à la Vitesse de l'Éclair : Les Boucles `for` et `while`

- **La boucle `for` (Quand on connaît le nombre de tours) :** `for (int i = 0; i < N; i++)` comporte 3 parties séparées par des points-virgules : 1) On démarre le compteur `i` à `0` ; 2) On continue tant que `i < N` ; 3) À la fin de chaque tour, `i++` ajoute `1` à `i` .
- **La boucle `while` (Tant qu'une condition est vraie) :** Elle tourne tant que la condition entre parenthèses reste vraie. Très utile quand on ne sait pas d'avance combien d'étapes seront nécessaires !

```cpp
// Afficher les 5 premiers tours d'un entraînement à Port-au-Prince :
for (int tour = 1; tour <= 5; tour++) {
    cout << "Tour de piste numero " << tour << "\n";
}
```

Exercices d'Entraînement — Leçon 2

Exercice 2.1 (Tarif du Tap-Tap Scolaire) : Écris un bloc `if / else if / else` qui lit l'âge d'une passagère (`int age`) : si `age < 12` , affiche `"Gratuit"` ; sinon si `age <= 18` , affiche `"Tarif Eleve : 25 HTG"` ; sinon affiche `"Tarif Adulte : 50 HTG"` .

```cpp
// Écris ton code C++ ici :
```

Exercice 2.2 (La Somme Olympique des Nombres Pairs) : Utilise une boucle `for` pour additionner tous les nombres pairs de `2` à `20` inclus dans une variable `int total = 0;` , puis affiche le résultat final.

```cpp
// Écris ton code C++ ici :
```

Leçon 3 : Organiser ses Données (`vector` , `string`) et Fonctions

3.1 Le Train de Données Dynamique : `vector<int>`

Si le Juge Automatique t'envoie 100 000 températures, tu ne vas pas créer 100 000 variables ! Tu utilises un tableau dynamique `vector` (en ajoutant `#include <vector>` en haut). Imagine un train dont chaque wagon porte un numéro appelé **indice (index)**. **Règle absolue : le premier wagon porte toujours le numéro `0` !**

```cpp
#include <vector>
// Créer un train de 4 villes d'Haïti :
vector<string> villes = {"Jacmel", "Port-au-Prince", "Cap-Haitien", "Les Cayes"};
cout << villes[0] << "\n";        // Affiche : Jacmel (le wagon 0)
villes.push_back("Gonaives");     // Accroche un nouveau wagon à la fin du train
cout << villes.size() << "\n";    // .size() donne le nombre total de wagons : 5
cout << villes.back() << "\n";    // .back() affiche le tout dernier wagon : Gonaives
```

3.2 Les Chaînes de Caractères (`string`)

En C++, un texte `string` se comporte exactement comme un train de lettres (`char`) ! Si `string mot = "AYITI";` , alors `mot[0]` vaut `'A'` , et `mot.size()` vaut `5` . Tu peux parcourir toutes les lettres d'un mot avec une boucle `for` .

3.3 Créer ses Propres Machines : Les Fonctions C++

Une fonction est un mixeur à jus : on indique d'abord le **type du résultat** qui va sortir (ex: `int` ou `double`), le nom de la fonction, et les ingrédients entre parenthèses (les paramètres). Le mot-clé `return` renvoie le résultat final. **Attention : déclare toujours tes fonctions AU-DESSUS de `int main()` !**

```
// Fonction placée AVANT int main() :
long long calculer_carre(long long n) {
    return n * n;
}
// Puis à l'intérieur de int main() :
cout << "Le carre de 100000 est " << calculer_carre(100000) << "\n"; // 10000000000
```

Exercices d'Entraînement — Leçon 3

Exercice 3.1 (Météo d'Haïti avec `vector`) : Soit `vector<double> temps = {28.5, 31.0, 29.5, 33.2, 30.1};`. 1) Affiche la première température (wagon 0) et la dernière (avec `.back()`). 2) Ajoute `32.0` à la fin avec `.push_back()`. 3) Écris une boucle `for` qui additionne toutes les températures du vecteur.

```
// Écris ton code C++ ici :
```

Exercice 3.2 (Compteur de Lettres dans un Mot) : Écris une fonction `int compter_a(string mot)` qui parcourt le texte `mot` avec une boucle `for` et renvoie (`return`) combien de fois la lettre `'A'` (majuscule) apparaît.

```
// Écris ta fonction C++ ici :
```

Leçon 4 : Survivre dans le Terminal Linux et Compiler avec `g++`

Lors de la compétition officielle à l'EGOI, ton ordinateur tourne sous **Linux**. Pour transformer ton code `solution.cpp` en un programme ultra-rapide et le tester sur des fichiers d'entrée sans retaper les nombres au clavier, on utilise le **Terminal Linux**.

Exercice d'Entraînement — Leçon 4 (Mission Terminal Linux EGOI)

En t'aidant de l'Aide-Mémoire situé à la dernière page de ton cahier, écris la commande Linux exacte à taper pour accomplir chacune de ces 6 actions de compétition :

1. Créer un nouveau dossier nommé `egoi_haiti` :

---

2. Entrer à l'intérieur du dossier `egoi_haiti` :

---

3. Afficher la liste de tous les fichiers présents dans le dossier :

---

4. Compiler le fichier `solution.cpp` pour créer l'exécutable `solution` (avec détection des erreurs) :

---

5. Lancer le programme `./solution` en lui faisant lire automatiquement le fichier `entree.txt` :

---

6. Comparer ta sortie `ma_sortie.txt` avec la réponse officielle `attendu.txt` :

Leçon 5 : Git — La Machine à Remonter le Temps de la Compétitrice

Pendant une épreuve EGOI de 5 heures, chaque problème est divisé en **Sous-tâches (Subtasks)**. Imaginons que ton code valide la Sous-tâche 1 (25 points) à 10h30. Tu modifies ton code pour chercher les 100 points, mais à 11h15 plus rien ne marche ! Grâce à **Git**, tu peux revenir instantanément à ta version propre de 10h30 en 3 étapes :

1. `git status` et `git diff` : Tu vérifies quels fichiers et quelles lignes C++ tu as modifiés.
2. `git add solution.cpp` : Tu places ton code qui marche dans le carton de sauvegarde.
3. `git commit -m "Sous-tache 1 validee (25 pts)"` puis `git push` : Tu prends une photo officielle de ton code et tu l'envoies sur le serveur de l'équipe !

Devoir à la Maison & Mini-Projet 0 (À Rendre au Coach — Noté sur 20)

Devoir Maison — Exercice 1 : Nettoyage de Capteurs Solaires à Jacmel (6 points)

Un panneau solaire lit `N` mesures entières sur l'entrée standard (`cin >> N;` puis `N` entiers). Les valeurs négatives ou nulles (`<= 0`) sont des pannes de capteur. Écris un programme C++ complet qui lit `N` , lit les `N` mesures dans une boucle, range uniquement les mesures strictement positives (`> 0`) dans un `vector<int> valides` , puis affiche sur une première ligne le nombre de mesures valides et sur une deuxième ligne le nombre de pannes ignorées.

```
// Écris ta solution C++ complète ici :
```

Devoir Maison — Exercice 2 : Le Célèbre Problème « Weird Algorithm » — CSES #1 (7 points)

Voici ton tout premier vrai problème d'Olympiade internationale ! On lit un entier de départ `long long n;` (attention, utilise bien `long long` car `n` peut dépasser la limite d'un `int` !). Tant que `n` est différent de `1` (`while (n != 1)`) : si `n` est pair (`n % 2 == 0`), divise-le par 2 ; sinon, multiplie-le par 3 et ajoute 1 (`n = 3 * n + 1`). Affiche toutes les valeurs de `n` séparées par un espace jusqu'à atteindre `1` .

```
// Écris ton code C++ pour Weird Algorithm ici :
```

Devoir Maison — Exercice 3 : Quiz Rapide C++, Linux & Git (7 points)

1. Pourquoi faut-il utiliser `long long` au lieu de `int` quand on additionne un million de nombres valant chacun 10^6 ?

---

2. Quelle est la différence entre l'opérateur `/` (division) et l'opérateur `%` (modulo) sur deux entiers en C++ ?

---

3. Quelle commande exacte permet de compiler `code.cpp` en un programme nommé `prog` ?

---

4. Tu viens de casser ton fichier `code.cpp` et tu veux annuler tes modifications pour revenir au dernier commit Git :

---

5. Que signifie le verdict **TLE** sur une plateforme de compétition comme CSES ?

Mini-Projet 0 : « Le Défi du Nombre Mystère Olympique » (Atelier de Clôture)

Dans un fichier nommé `projet0_mystere.cpp`, complète les 4 blocs `// TODO` ci-dessous pour programmer le jeu du Nombre Mystère, compile-le avec `g++` dans le terminal, joue une partie et sauvegarde ton fichier avec Git :

```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
using namespace std;

int main() {
    srand(time(0));
    int secret = (rand() % 100) + 1; // L'ordinateur choisit un nombre secret entre 1
et 100
    int essais_max = 7;
    vector<int> historique;

    // TODO 1 : Crée une boucle for pour 'tentative' allant de 1 jusqu'à essais_max
inclus
    // TODO 2 : Lis la proposition de la joueuse avec cin >> choix; et ajoute-la à
'historique' avec push_back()
    // TODO 3 : Utilise if / else if / else pour afficher "BRAVO !", "C'est PLUS GRAND
!" ou "C'est PLUS PETIT !"
    // TODO 4 : En cas de victoire, affiche tous les essais de 'historique' avec une
boucle et fais return 0;
    return 0;
}
```

Fiche Aide-Mémoire Officielle — Module 0 EGOI (À Garder Près du Clavier)

Catégorie	Syntaxe C++ / Commande Linux	Action & Astuce de Compétition EGOI
Squelette & E/S Rapides	`#include <iostream>` / `<vector>` `cin >> x;` / `cout << x << "\n";`	Toujours utiliser `"\n"` plutôt que `endl` (car `endl` ralentit l'affichage et peut causer un TLE !).
Types & Modulo	`int` ($\leq 2\times10^9$) / `long long` ($\leq 9\times10^{18}$) `a % b` (Reste de la division)	`x % 2 == 0` teste si `x` est pair. Dès qu'une multiplication ou somme dépasse 2 milliards, utilise `long long`.
Conditions & Boucles	`if (x >= 10 && y == 5) { ... }` `for (int i = 0; i < n; i++) { ... }`	Attention : `==` compare deux valeurs, `=` modifie la variable. Une boucle de `0` à `i < n` fait exactement `n` tours.
Vecteurs & Fonctions	`vector<int> v;` `v.push_back(x);` `v[0]`, `v.back()`, `v.size()`	Le premier élément est toujours à l'indice `0`, le dernier est à l'indice `v.size() - 1` (ou `v.back()`).
Linux & Compilation C++	`ls` / `cd dossier` / `cd ..` / `mkdir` `g++ -O2 -Wall sol.cpp -o sol` `./sol < in.txt > out.txt`	`-O2` active la vitesse olympique maximale ; `-Wall` affiche tous les avertissements de bugs cachés. `diff a.txt b.txt` compare deux fichiers.
Git Sauvegarde	`git status` / `git diff` / `git add .` `git commit -m "Msg"` / `git restore`	Fais un `git commit` dès que tu valides une Sous-tâche (ex: 20 points) avant d'essayer d'optimiser ton algorithme !