

Mini Projet en programmation C

■ FIA - FÉDÉRATION INTERNATIONALE DE L'AUTOMOBILE

— **Section 1 - Présentation du Projet**

Ce mini-projet doit être réalisé **en binôme**. Il a pour objectif de développer un système de gestion informatisé pour un championnat de Formule 1, en appliquant les concepts de programmation modulaire en langage C.

L'application devra permettre de gérer les pilotes, les écuries et les Grands Prix, ainsi que d'établir des classements détaillés. Le projet mettra l'accent sur une architecture modulaire claire et bien structurée, favorisant la maintenance, la réutilisabilité et l'évolutivité du code.

Chaque fonctionnalité devra être implémentée dans un module dédié, en respectant les bonnes pratiques de programmation en C.

La présentation du projet fera l'objet d'une soutenance de **15 minutes** lors de la dernière séance du module XTI101, prévue durant la semaine du 3 novembre. L'ordre de passage des binômes sera communiqué ultérieurement.

🏆 **Section 2 - Organisation des Championnats**

Le championnat de Formule 1 suit un système de points Standardisé :

1ère place : 25 points

2ème place : 18 points

3ème place : 15 points

4ème place : 12 points

5ème place : 10 points

6ème place : 8 points

7ème place : 6 points

8ème place : 4 points

9ème place : 2 points

10ème place : 1 point

Le championnat comprend deux classements distincts :

- Championnat des Pilotes : Classement individuel basé sur les points accumulés
- Championnat des Constructeurs : Classement des écuries basé sur la somme des points de leurs deux pilotes

Section 3 - Modélisation des Données

Cette section détaille les **6 structures principales** qui constituent l'architecture de votre application. Chaque structure est accompagnée de sa déclaration en C, d'un tableau détaillé des attributs et d'exemples d'utilisation.

3.1 Structures de données

3.1.1 Structure Date 📆

Attribut	Type	Contraintes	Description	Exemple
jour	int	$1 \leq \text{jour} \leq 31$	Jour du mois avec validation selon le mois	15
mois	int	$1 \leq \text{mois} \leq 12$	Mois de l'année (janvier = 1)	6 (juin)
annee	int	$1900 \leq \text{annee} \leq 2100$	Année de la course avec gestion bissextile	2024

3.1.2 Structure Heure ⏰

Attribut	Type	Contraintes	Description	Exemple
heure	int	$0 \leq \text{heure} \leq 23$	Heure au format 24h	14 (14h00)
minute	int	$0 \leq \text{minute} \leq 59$	Minutes	30

3.1.3 Structure Pilote 🚗

Attribut	Type	Description	Exemple	Contraintes
Nom	char[50]	Nom de famille du pilote	"Verstappen"	Non vide, max 49 caractères
prenom	char[50]	Prénom du pilote	"Max"	Non vide, max 49 caractères
nationalite	char[50]	Pays de représentation	"Pays-Bas"	Format pays standard
ecurie	char[50]	Nom de l'écurie d'appartenance	"Red Bull Racing"	Doit exister dans les écuries
points	int	Points accumulés durant la saison	258	≥ 0 , calculé automatiquement
numero	int	Numéro de voiture unique	1	1-99, unique par saison
Age	int	Âge du pilote en années	26	$18 \leq \text{age} \leq 50$
Actif	int	Statut du pilote (suppression logique)	1	0 ou 1

3.1.4 Structure Écurie 🏁

Attribut	Type	Description	Exemple	Contraintes
Nom	char[50]	Dénomination officielle	"Scuderia Ferrari"	Unique, non vide
Pays	char[50]	Pays du siège social	"Italie"	Format pays standard
Points	int	Somme des points des 2 pilotes	512	≥ 0 , calculé automatiquement
anneeCreation	int	Année de création de l'écurie	1950	$1900 \leq \text{année} \leq \text{année actuelle}$
Directeur	char[50]	Nom du directeur d'écurie	"Frédéric Vasseur"	Non vide
Actif	int	Statut de l'écurie	1	0 ou 1

3.1.5 Structure ResultatCourse 🏆

Attribut	Type	Description	Exemple	Validation
nomPilote	char[50]	Nom du pilote	"Hamilton"	Doit exister dans les pilotes
prenomPilote	char[50]	Prénom du pilote	"Lewis"	Cohérent avec le nom
nationalite	char[50]	Nationalité du pilote	"Royaume-Uni"	Cohérent avec le pilote
Position	int	Position finale (classement)	1	$1 \leq \text{position} \leq 20$, unique
tempsRealise	char[20]	Temps total de course	"1:24:35.123"	Format H:MM:SS.sss
pointsObtenus	int	Points FIA selon position	25	Calculé automatiquement

3.1.6 Structure GrandPrix 🏁

Attribut	Type	Description	Exemple	Spécificité
nomCircuit	char[50]	Nom officiel du circuit	"Circuit de Monaco"	Unique par saison
Pays	char[50]	Pays organisateur	"Monaco"	-
nombreTours	int	Nombre de tours à effectuer	78	$10 \leq \text{tours} \leq 100$
Date	Date	Structure imbriquée Date	{26, 5, 2024}	Validation complète
Horaire	Heure	Structure imbriquée Heure	{15, 00}	Format 24h
Réultats	ResultatCourse[20]	Tableau de structures ResultatCourse	Tableau de 20 résultats max	Gestion dynamique
nombreRésultats	int	Compteur des résultats saisis	20	$0 \leq \text{nombre} \leq 20$
Actif	int	Statut du Grand Prix	1	0 ou 1

3.2 Schéma des Relations entre Structures

Diagramme des Relations Structurelles

GrandPrix

```

└── Date date (structure imbriquée)
    |   └── int jour
    |   └── int mois
    |   └── int annee
    └── Heure horaire (structure imbriquée)
        |   └── int heure
        |   └── int minute
    └── ResultatCourse resultats[20] (tableau de structures)
        └── Chaque ResultatCourse contient :
            └── nomPilote (référence vers Pilote)
            └── prenomPilote (référence vers Pilote)
            └── pointsObtenus → mise à jour Pilote.points

```

Pilote

```

└── ecurie (référence vers Ecurie.nom)
    └── points → contribue à Ecurie.points

```



Relations Clés à Implémenter :

1. **Composition** : GrandPrix contient Date et Heure
2. **Agrégation** : GrandPrix contient un tableau de ResultatCourse
3. **Association** : Pilote est associé à une Écurie
4. **Référence** : ResultatCourse référence un Pilote existant
5. **Calcul automatique** : Points Écurie = Somme points des 2 pilotes

⌚ Section 4 - Architecture Modulaire en C

L'architecture modulaire facilite la maintenance et la réutilisabilité du code. Chaque module est composé de deux fichiers :

- Fichier header (.h) : Interface publique du module
- Fichier source (.c) : Implémentation des fonctionnalités

📘 Structure des Modules

Module	Fichier .h	Fichier .c	Responsabilité
Pilotes	pilote.h	pilote.c	Gestion complète des pilotes
Écuries	ecurie.h	ecurie.c	Gestion complète des écuries
Grands Prix	grandprix.h	grandprix.c	Gestion complète des courses
Classements	classement.h	classement.c	Algorithmes de tri et affichage

🔧 Rôle des Fichiers

- **Fichiers .h (header)** : Déclarations des structures de données, prototypes de fonctions, constantes (#define), inclusion des bibliothèques nécessaires
- **Fichiers .c (source)** : Implémentation des fonctions, logique métier, algorithmes de traitement
- **main.c** : Point d'entrée du programme, initialisation des données, boucle principale de l'application

📅 Section 5 - Approche Progressive

Le développement se fera de manière progressive avec :

- Développement des fonctions de gestion complètes (CRUD)
- Implémentation des fonctions de modification sécurisées
- Gestion des relations entre pilotes et écuries
- Validation des données et gestion des erreurs

⌚ Section 6 - Fonctionnalités Attendues

👤 Gestion des Pilotes

- **Ajouter un pilote** : Saisie complète des informations personnelles et sportives
- **Mettre à jour les points** : Modification manuelle ou automatique après course
- **Supprimer un pilote** : Retrait sécurisé avec vérification des dépendances
- **Afficher la liste** : Consultation de tous les pilotes avec leurs statistiques

Gestion des Écuries

- **Ajouter une écurie** : Création avec informations complètes (nom, pays, directeur)
- **Mettre à jour les points** : Calcul automatique basé sur les pilotes affiliés
- **Supprimer une écurie** : Suppression avec gestion des pilotes associés
- **Afficher la liste** : Vue d'ensemble de toutes les écuries participantes

Gestion des Grands Prix

- **Ajouter un Grand Prix** : Configuration complète (circuit, date, tours)
- **Mettre à jour les résultats** : Saisie du classement final avec temps
- **Supprimer un Grand Prix** : Effacement avec recalculation des points
- **Afficher la liste** : Calendrier complet des courses de la saison

Affichages et Classements

- **Classement d'une course** : Résultats détaillés avec temps et nationalités
- **Classement général pilotes** : Tri par points décroissants avec détails
- **Classement constructeurs** : Ranking des écuries avec points totaux

Section 7 - Spécifications Techniques

IMPORTANT : Pour faciliter les tests et optimiser le temps lors de la soutenance, votre programme **DOIT** inclure des données pré-initialisées en dur dans votre code

Données Minimales Requises

Votre fichier contenir *initialisation.c* doit contenir

✓ Au minimum :

- **5 écuries** avec toutes leurs informations complètes
- **10 pilotes** répartis dans les écuries (2 pilotes par écurie)
- **3 Grands Prix** avec leurs résultats complets (20 positions par course)

```
// Exemple d'initialisation dans main.c ou initialisation.c void initialiserDonneesTest() {
    // Initialisation des écuries
    Ecurie ecuries[5] = { {"Red Bull Racing", "Autriche", 0, 2005, "Christian Horner", 1}, {"Scuderia Ferrari", "Italie", 0, 1950, "Frédéric Vasseur", 1}, {"Mercedes-AMG", "Allemagne", 0, 2010, "Toto Wolff", 1}, {"McLaren Racing", "Royaume-Uni", 0, 1966, "Andrea Stella", 1}, {"Aston Martin", "Royaume-Uni", 0, 2021, "Mike Krack", 1} };

    // Initialisation des pilotes Pilote
    pilotes[10] = { {"Verstappen", "Max", "Pays-Bas", "Red Bull Racing", 0, 1, 27, 1}, {"Perez", "Sergio", "Mexique", "Red Bull Racing", 0, 11, 34, 1}, {"Leclerc", "Charles", "Monaco", "Scuderia Ferrari", 0, 16, 27, 1}, {"Sainz", "Carlos", "Espagne", "Scuderia Ferrari", 0, 55, 30, 1}, {"Hamilton", "Lewis", "Royaume-Uni", "Mercedes-AMG", 0, 44, 39, 1}, {"Russell", "George", "Royaume-Uni", "Mercedes-AMG", 0, 63, 26, 1}, {"Norris", "Lando", "Royaume-Uni", "McLaren Racing", 0, 4, 25, 1}, {"Piastri", "Oscar", "Australie", "McLaren Racing", 0, 81, 23, 1}, {"Alonso", "Fernando", "Espagne", "Aston Martin", 0, 14, 43, 1}, {"Stroll", "Lance", "Canada", "Aston Martin", 0, 18, 26, 1} };

    // Initialisation d'un Grand Prix avec
    résultats GrandPrix gp1; strcpy(gp1.nomCircuit, "Circuit de Monaco");
    strcpy(gp1.pays, "Monaco"); gp1.nombreTours = 78; gp1.date = (Date){26, 5, 2024};
    gp1.horaire = (Heure){15, 0}; gp1.nombreResultats = 10; gp1.actif = 1;

    // Résultats du Grand Prix (exemple pour les 10 premières
    positions) ResultatCourse resultats[10] = { {"Verstappen", "Max", "Pays-Bas", 1, "1:44:12.456", 25}, {"Leclerc", "Charles", "Monaco", 2, "1:44:18.234", 18}, {"Hamilton", "Lewis", "Royaume-Uni", 3, "1:44:25.678", 15}, {"Norris", "Lando", "Royaume-Uni", 4, "1:44:32.123", 12}, {"Sainz", "Carlos", "Espagne", 5, "1:44:39.456", 10}, {"Russell", "George", "Royaume-Uni", 6, "1:44:45.789", 8}, {"Perez", "Sergio", "Mexique", 7, "1:44:52.234", 6}, {"Alonso", "Fernando", "Espagne", 8, "1:44:58.567", 4}, {"Piastri", "Oscar", "Australie", 9, "1:45:04.890", 2}, {"Stroll", "Lance", "Canada", 10, "1:45:11.123", 1} };

    // Copie des résultats dans la structure GrandPrix
    for(int i = 0; i < 10; i++) {
        gp1.resultats[i] = resultats[i];
    }
    // TODO: Répéter pour les autres Grands Prix... }
```

Avantages de cette Approche

- ✓ **Gain de temps** considérable lors de la soutenance
- ✓ **Tests rapides** de toutes les fonctionnalités
- ✓ **Démonstration fluide** sans saisies manuelles
- ✓ **Données cohérentes** et réalistes pour l'évaluation
- ✓ **Focus sur la présentation** plutôt que sur la saisie

Recommandation

- Créez un fichier séparé **initialisation.h** et **initialisation.c** pour l'organisation
- Commentez clairement vos données de test
- Assurez-vous que les données sont cohérentes (pilotes existants, écuries valides, etc.)
- Prévoyez également la possibilité d'ajouter manuellement des données pendant la démo
- Testez votre initialisation avant la soutenance

Contraintes Techniques

- Utilisation exclusive du langage C (norme C99 ou supérieure)
- Compilation sans erreurs ni avertissements
- Gestion appropriée de la mémoire (allocation/libération)
- Validation des saisies utilisateur
- Code commenté et bien structuré
- Respect des conventions de nommage
- **IMPORTANT :** Travail en mémoire uniquement (pas de sauvegarde fichier)

💡 Section 8 - Exercice de Créativité

Pour enrichir votre projet, vous pouvez implémenter deux fonctionnalités supplémentaires qui vous semblent pertinentes pour votre projet.

📋 Section 9 - Modalités d'Évaluation

⌚ Critères d'Évaluation

Fonctionnalité (30%)

Toutes les fonctions demandées sont opérationnelles

Qualité du code (20%)

Structure, lisibilité, commentaires, respect des conventions

Architecture (20%)

Organisation modulaire, séparation des responsabilités

Présentation (30%)

Clarté de l'exposé, maîtrise du projet, réponses aux questions

⌚ Section 10 - Conseils et Bonnes Pratiques

Développement

- Commencer par les structures de données avant les fonctions
- Tester chaque fonction individuellement avant l'intégration
- Utiliser des noms de variables et fonctions explicites
- Commenter le code de manière claire et concise
- Gérer les cas d'erreur et les saisies incorrectes

Préparation du Projet

- Préparer des données de démonstration réalistes et variées
- Maîtriser parfaitement toutes les fonctionnalités de votre application
- Être capable d'expliquer vos choix techniques et algorithmiques
- Anticiper les questions du jury sur l'architecture et l'implémentation

Présentation

- Utiliser les données pré-initialisées en dur pour ne pas perdre de temps lors de la soutenance
- Expliquer la logique de votre architecture modulaire
- Présenter les défis techniques rencontrés et leurs solutions
- Montrer la possibilité d'ajouter de nouvelles données en plus des données initialisées

📁 Section 11 – Dépôt du travail

 **Premier dépôt obligatoire :** À remettre sur Moodle à la fin de votre **avant-dernière séance (semaine du 27 octobre 2025)**. Déposez votre travail réalisé à cette date **sous forme d'archive** (dossier zippé contenant l'ensemble des fichiers de votre projet) dans l'espace de dépôt correspondant à votre groupe. Le nom de l'archive doit correspondre aux noms des deux personnes formant le binôme, par exemple : *DupontMartin_LefebvreClair.zip*. Un seul dépôt par binôme.

 **Deuxième dépôt obligatoire :** À remettre sur Moodle **au plus tard la veille de votre dernière séance de XTI101 avant 23h59**, sous forme d'archive (dossier zippé contenant l'ensemble des fichiers de votre projet). Utilisez l'espace de dépôt correspondant à votre groupe. Le nom de l'archive doit respecter le même format que le premier dépôt : *DupontMartin_LefebvreClair.zip*. Un seul dépôt par binôme.