



Projet Burger QuizBot - Guide Développeur (Partie Jeu)

Ce document détaille les étapes pour coder le cœur du jeu.

Objectif : Avoir une version jouable avec 3 questions (sans encore gérer les fichiers externes) pour valider la logique.



Étape 1 : Les Fondations (Structures)

Avant de coder le jeu, il faut définir nos "briques". Crée un fichier nommé structures.h et mets-y ce code. Cela servira pour tout le projet.

```
#ifndef STRUCTURES_H
#define STRUCTURES_H
```

```
// Structure représentant un Joueur
```

```
typedef struct {
    char pseudo[50];
    int score_actuel;
    int meilleur_score;
} User;
```

```
// Structure représentant une Question
```

```
typedef struct {
    int id;
    char enonce[256];    // Le texte de la question
    char option_A[100];  // Choix A
    char option_B[100];  // Choix B
    char option_C[100];  // Choix C
    char bonne_reponse;  // 'A', 'B' ou 'C'
} Question;
```

```
#endif
```



Étape 2 : Créer des données de test

Comme on ne gère pas encore la lecture de fichier (c'est compliqué), on va créer un tableau de 3 questions "en dur" dans le code pour pouvoir tester la boucle de jeu tout de suite.

Dans ton fichier main.c (ou game.c), tu devras initialiser un tableau comme ceci :

// Exemple en C pour créer 3 questions manuellement

Question tab_questions[3];

// Question 1

tab_questions[0].id = 1;

strcpy(tab_questions[0].enonce, "Quelle est la couleur du cheval blanc d'Henri IV ?");

strcpy(tab_questions[0].option_A, "Blanc");

strcpy(tab_questions[0].option_B, "Gris");

strcpy(tab_questions[0].option_C, "Noir");

tab_questions[0].bonne_reponse = 'A';

// Question 2 ... (à toi de faire les suivantes)

Étape 3 : La Logique du Jeu (Pseudo-code)

Voici l'algorithme de la fonction principale jouer_partie.

Ta mission : Traduire ce pseudo-code en langage C.

Fonction : void jouer_partie(User *joueur, Question tab_questions[], int nb_questions)

DEBUT FONCTION

1. Initialiser une variable 'score' à 0.
2. Initialiser une variable 'reponse_user' (caractère).
3. AFFICHER "Bienvenue [Nom du joueur] ! Le quiz commence."
4. BOUCLE : Pour i allant de 0 à (nb_questions - 1) :

```
// A. Récupération de la question actuelle
Question q = tab_questions[i]
```

```
// B. Affichage
AFFICHER "Question n°" + (i + 1)
AFFICHER q.enonce
AFFICHER "A) " + q.option_A
AFFICHER "B) " + q.option_B
AFFICHER "C) " + q.option_C
```

```
// C. Saisie sécurisée (Boucle de vérification)
FAIRE
    AFFICHER "Votre réponse (A, B ou C) ? "
    LIRE reponse_user
    // Astuce : Convertir reponse_user en majuscule si c'est 'a' -> 'A'
TANT QUE (reponse_user n'est pas 'A' ET pas 'B' ET pas 'C')
```

```
// D. Vérification de la réponse
SI reponse_user EGAL q.bonne_reponse ALORS
    AFFICHER "Bonne réponse !"
    score = score + 1
SINON
    AFFICHER "Mauvaise réponse... La bonne était " + q.bonne_reponse
FIN SI
```

```
AFFICHER "-----"
```

FIN BOUCLE

5. Afficher le score final

AFFICHER "Partie terminée ! Score : " + score + " / " + nb_questions

6. Mise à jour du profil joueur

joueur->score_actuel = score

SI score > joueur->meilleur_score ALORS

 joueur->meilleur_score = score

 AFFICHER "Bravo, c'est un nouveau record !"

FIN SI

FIN FONCTION



Anti-sèche Technique (Pièges à éviter en C)

Voici les erreurs classiques qui font planter le programme. Lis bien ça !

1. Lire un caractère (scanf)

Quand tu demandes une lettre (A, B ou C) après avoir appuyé sur "Entrée" précédemment, le programme risque de lire le retour à la ligne (\n) au lieu de la lettre.

- **Mauvais** : `scanf("%c", &reponse);`
- **Bon** : `scanf(" %c", &reponse);` (Note l'espace avant le %c, c'est magique, ça nettoie le buffer).

2. Copier du texte

En C, on ne peut pas faire `question.enonce = "Texte";`.

- **Bon** : Utilise `strcpy(question.enonce, "Texte");`. N'oublie pas d'inclure `<string.h>`.

3. Pointeurs et Structures

- Si tu as une variable normale (Question q), tu accèdes aux champs avec un point : `q.enonce`.
- Si tu as un **pointeur** (User *joueur), tu accèdes aux champs avec une flèche : `joueur->pseudo`.

4. Comparer des chaînes de caractères

- **Mauvais** : `if (reponse == "A")` (Ça ne marche pas en C pour les chaînes).
- **Bon** : Pour un seul caractère (char), `if (reponse == 'A')` fonctionne. Pour du texte complet, utilise `strcmp`.

Bon courage ! Commence par faire marcher ça avec les 3 questions manuelles. Une fois que c'est bon, on s'occupera des fichiers.