ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

Annee universitaire : 2024/2025

GESTIONNAIRE DES RESTAURANTS

Realiser par :
Aymen BALLAGE
Yahya BENMANSOUR HASSANI

Encadre par:
Prof. Kamal OMARI

Sommaire

Liste des figures

1 Introduction

Ce rapport présente un projet de développement d'un gestionnaire de restaurants implémenté en langage C. Le système permet de gérer efficacement les plats d'un restaurant en utilisant des structures de données dynamiques. L'application offre des fonctionnalités complètes pour l'ajout, la modification, la suppression et la recherche de plats, ainsi que des options de tri personnalisées.

L'importance de ce projet réside dans sa capacité à démontrer la maîtrise des concepts fondamentaux de la programmation en C, notamment les structures de données, les pointeurs, la gestion de fichiers et l'interface utilisateur en mode console.

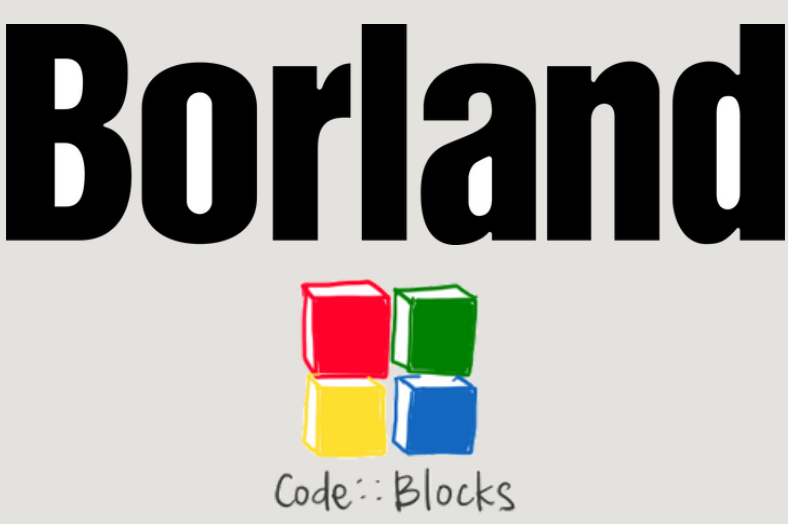
2 Objectifs et contexte

Contexte :

Dans le domaine de la restauration, la gestion efficace des menus est essentielle. Ce projet s'inscrit dans le cadre de l'apprentissage des structures de données en C, en particulier les listes chaînées, pour créer un système de gestion de plats de restaurant.

Objectifs principaux :

- Implémenter une structure de données dynamique pour stocker les informations des plats
- Développer une interface utilisateur intuitive en mode console
- Gérer les opérations CRUD (Create, Read, Update, Delete) sur les plats
- Implémenter des mécanismes de recherche et de tri
- Assurer la persistance des données grâce à la gestion de fichiers



3 Méthodologie

Langage et outils :

- Langage C (standard C99)
- Bibliothèques utilisées : stdio.h, stdlib.h, string.h, conio.h, windows.h, time.h
- Environnement de développement : Code::Blocks (Dans notre cas , car on n'a utilise le compilateur Borland qu'il n'est pas compatible avec VSCodium).

Structures de données :

```
typedef struct PLAT {
   char nom[30];
   float prix;
   char categorie[10];
   int desc_position;
} PLAT;

typedef struct LIST {
   PLAT dish;
   struct LIST* next;
} LIST;
```

Fonctionnalités implémentées :

- Ajout de plats (début, fin, après un plat spécifique)
- Affichage des plats
- Modification des détails des plats
- Suppression de plats (début, fin, plat spécifique)
- Recherche par nom ou catégorie
- Tri alphabétique des plats
- Gestion des descriptions dans un fichier séparé

Gestion des fichiers :

- Les descriptions des plats sont stockées dans un fichier texte ("descriptionRES.txt")
- Utilisation de positions pour accéder rapidement aux descriptions

4 Résultats

Fonctionnalités opérationnelles :

Menu principal avec deux options :

- Gestion des restaurants
- Quitter

Sous-menu de gestion avec 8 options :

- Ajout de plats (3 modes)
- Modification des plats
- Suppression de plats (3 modes)
- Recherche de plats (par nom, catégorie)
- Affichage des plats
- Gestion des catégories (à implémenter)
- Options de tri
- Retour au menu principal

Exemple d'affichage :

Nom	Prix	Catégorie	Description
Pizza	12.50	Italienne	Pizza margherita avec fromage
Salade César	8.75	Entrée	Salade et sauce césar

5 Analyse

Points forts :

- Structure de données efficace utilisant des listes chaînées
- Interface utilisateur colorée et interactive
- Séparation des données (description dans un fichier séparé)
- Gestion des erreurs avec messages d'alerte
- Tri fonctionnel des plats par nom

Limitations :

- Gestion des catégories non complètement implémentée
- Pas de sauvegarde/chargement de la liste complète des plats
- Interface limitée au mode console
- Recherche avancée non implémentée

Complexités techniques résolues :

- Gestion des positions dans le fichier de description
- Manipulation précise des pointeurs dans les listes chaînées
- Synchronisation entre la liste en mémoire et le fichier de description

6 Conclusion

Ce projet a permis de mettre en pratique les concepts fondamentaux du langage C, en particulier les structures de données dynamiques et la gestion de fichiers. Le gestionnaire de restaurants développé offre des fonctionnalités de base complètes pour la gestion des plats.

Les principaux enseignements tirés incluent :

- L'importance d'une bonne conception des structures de données
- La gestion minutieuse de la mémoire dans les programmes C
- L'utilité des fichiers pour la persistance des données
- Les défis de la création d'une interface utilisateur en mode console

7 Perspectives

Améliorations possibles :

1. Implémenter la sauvegarde/chargement complet de la liste des plats
2. Compléter la gestion des catégories
3. Ajouter une recherche avancée multicritère
4. Développer une interface graphique
5. Ajouter la gestion de plusieurs restaurants
6. Implémenter un système de notation des plats

Applications potentielles :

- Système de gestion de menu pour petits restaurants
- Base pour un système de commande en ligne
- Module d'un système plus complet de gestion de restaurant

8 Bibliographie

GeeksForGeeks

9 Section de codes

Structure de données et création de plat :

```
typedef struct PLAT {
    char nom[30];
    float prix;
    char categorie[10];
    int desc_position; // Pour suivre la position dans le fichier de description
} PLAT;

LIST* createMEAL(int* desc_counter) {
    LIST* meal = (LIST*)malloc(sizeof(LIST));
    // ... (vérification d'allocation)

    // Saisie des informations
    cprintf("Nom : ");
    fgets(meal->dish.nom,sizeof(meal->dish.nom),stdin);
    meal->dish.nom[strcspn(meal->dish.nom, "\n")] = 0;

    // ... (saisie des autres champs)

    // Gestion de la description
    meal->dish.desc_position = *desc_counter;
    (*desc_counter)++;
    fprintf(desc,"%s\n",descMEAL);

    return meal;
}
```

Tri des plats :

```
LIST* triCroissant(LIST* RES) {
   LIST *sorted = NULL;
   LIST *current = RES;

   while(current != NULL) {
      LIST *next = current->next;

      if(sorted == NULL || strcmp(current->dish.nom, sorted->dish.nom) < 0) {
         current->next = sorted;
         sorted = current;
      } else {
         LIST *temp = sorted;
         while(temp->next != NULL && strcmp(current->dish.nom, temp->next-
>dish.nom) > 0) {
            temp = temp->next;
         }
         current->next = temp->next;
         temp->next = current;
      }
      current = next;
   }
   return sorted;
}
```

Liste des figures

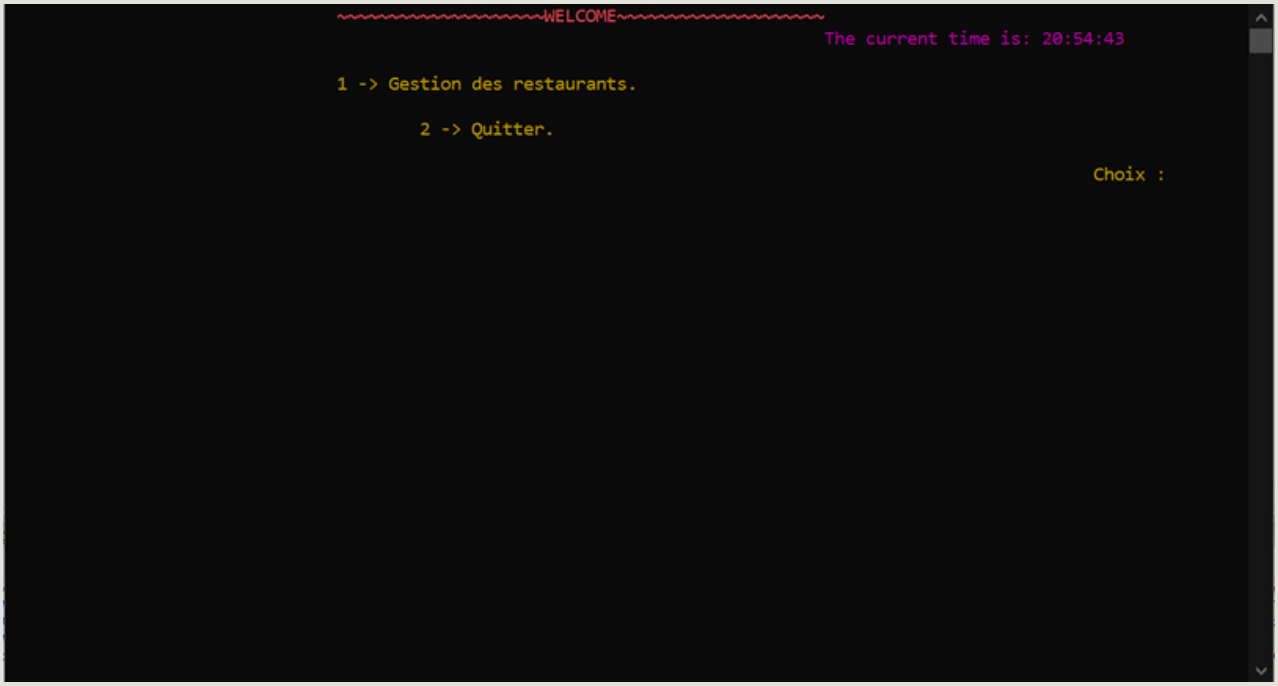
Figure 1 : Diagramme de la structure de données (liste chaînée)

```c
typedef struct PLAT{
    char nom[30];
    float prix;
    char categorie[10];
    int desc_position;
}PLAT;

typedef struct LIST{
    PLAT dish;
    struct LIST* next;
}LIST;

FILE* desc;
char descMEAL[100];
```

Figure 2 : Capture d'écran du menu principal



Liste des Figures

Figure 3 : Capture d'écran de l'affichage des plats