Université de Yaoundé I

Faculté des Sciences

Département d'Informatique

RAPPORT DE TRAVAUX PRATIQUES

INF231 : Structure de Données II

TP2 : Programme de Manipulation de Listes Chainées en C

Responsable : Pr. Melatagia
Matière : INF231 - Structure de Données II
Année académique : 2025-2026

Groupe de TP

Nom	Prénom	Matricule
KENGNI DIEKONG	BERNARD LOIC	23V2146
PETANG	DANIEL	23V2121
MBEZELE ZOA	DANIELLE NAOMI	24G2837
ABDEL ADY TCHALLA NGANDO		23V2538
AZAMBOU	MARTHE NEFERTYTI	23V2357
TEKENG KAMWELE	JUNIOR CAMBELL	23U2686

Table des matières

1 Introduction

Ce rapport présente les travaux pratiques réalisés dans le cadre du cours de Structure de Données II (INF231) sous la direction du Professeur Melatagia. L'objectif de ce TP était de développer un programme en C permettant de manipuler des listes chainées à travers diverses opérations mathématiques.

Le programme implémente cinq fonctionnalités principales :

1. Lire un élément et supprimer toutes les occurences dans la liste
2. Insertion d'un élément dans une liste simplement chaînée triée
3. Insertion d'un élément dans une liste doublement chaînée triée
4. Insertion en tête et en queue dans une liste simplement chaînée circulaire
5. Insertion en tête et en queue dans une liste doublement chaînée circulaire

Ce document détaille pour chaque fonctionnalité le problème abordé, le principe de résolution, le dictionnaire de données, l'algorithme utilisé et l'analyse de complexité. Il présente également la structure du programme et les choix d'implémentation.

2 Structure du programme

Le programme est organisé en trois fichiers principaux :

— `main.c` : Contient la fonction principale et le menu interactif
— `tp2.h` : Contient les déclarations des fonctions et constantes
— `tp2.c` : Contient l'implémentation des fonctions

2.1 Compilation et exécution

Le programme peut être compilé à l'aide de la commande suivante :

```
make
```

L'exécution se fait ensuite avec :

```
./main
```

2.2 Menu principal

Le programme propose un menu interactif permettant à l'utilisateur de sélectionner l'opération souhaitée. Une boucle principale assure la continuité de l'exécution jusqu'à ce que l'utilisateur choisisse de quitter le programme.

2.3 Gestion des entrées

Une attention particulière a été portée à la robustesse de la gestion des entrées utilisateur. La fonction `lireEntier` assure la validation des entrées numériques en rejetant les caractères non autorisés et en vérifiant les plages de valeurs.

3 Dossiers algorithmiques

3.1 Lire un élément et supprimer toutes les occurences dans la liste

3.1.1 Problème

Enoncer le probleme ici. EX (Déterminer si un tableau est trié par ordre croissant ou décroissant.)

3.1.2 Principe

Enoncer le principe ici. EX (Vérifier les relations entre éléments consécutifs.)

3.1.3 Dictionnaire de données

— Completez EX (tab : tableau d'entiers)
— Completez
— Completez
— Completez jusqu'a n-ieme item

3.1.4 Algorithme

```
Ecrire l'algorithme ici
```

3.1.5 Complexité

— Temps : Completez
— Espace : Completez
— Totale : Completez

3.2 Insertion d'un élément dans une liste simplement chaînée triée

3.2.1 Problème

Enoncer le probleme ici.

3.2.2 Principe

Enoncer le principe ici.

3.2.3 Dictionnaire de données

— Completez EX (tab : tableau d'entiers)
— Completez
— Completez
— Completez jusqu'a n-ieme item

3.2.4 Algorithme

```
Ecrire l'algorithme ici
```

3.2.5 Complexité

— Temps : Completez
— Espace : Completez
— Totale : Completez

3.3 Insertion d'un élément dans une liste doublement chaînée triée

3.3.1 Problème

Enoncer le probleme ici.

3.3.2 Principe

Enoncer le principe ici.

3.3.3 Dictionnaire de données

— Completez EX (tab : tableau d'entiers)
— Completez
— Completez
— Completez jusqu'a n-ieme item

3.3.4 Algorithme

```
Ecrire l'algorithme ici
```

3.3.5 Complexité

— Temps : Completez
— Espace : Completez
— Totale : Completez

3.4 Insertion en tête et en queue dans une liste simplement chaînée circulaire

3.4.1 Problème

Enoncer le probleme ici.

3.4.2 Principe

Enoncer le principe ici.

3.4.3 Dictionnaire de données

— Completez EX (tab : tableau d'entiers)
— Completez
— Completez
— Completez jusqu'a n-ieme item

3.4.4 Algorithme

```
Ecrire l'algorithme ici
```

3.4.5 Complexité

— Temps : Completez
— Espace : Completez
— Totale : Completez

3.5 Insertion en tête et en queue dans une liste doublement chaînée circulaire

3.5.1 Problème

Enoncer le probleme ici.

3.5.2 Principe

Enoncer le principe ici.

3.5.3 Dictionnaire de données

— Completez EX (tab : tableau d'entiers)
— Completez
— Completez
— Completez jusqu'a n-ieme item

3.5.4 Algorithme

```
Ecrire l'algorithme ici
```

3.5.5 Complexité

— Temps : Completez
— Espace : Completez
— Totale : Completez

4 Conclusion

Ce TP a permis de mettre en œuvre diverses opérations fondamentales sur les listes chainées en langage C. Chaque algorithme a été analysé en termes de complexité temporelle et spatiale, permettant de comprendre leurs performances relatives.

Les principales difficultés rencontrées ont concerné la.. . La solution implémentée avec les fonctions `lireEntier`, etc (autres fonction qui ont resolu les principales difficultés) assure une expérience utilisateur fiable.

Ce travail illustre l'importance des structures de données et des algorithmes dans la résolution de problèmes mathématiques courants, et démontre la capacité du langage C à implémenter efficacement ces solutions.

Annexes

Annexe A : Code source

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "tp2.h"

int main() {
    printf("\t\tBIENVENUE DANS LE MENU PRINCIPAL\n");
    char input[100];
    char n;

    do {
        printf("0- Quitter\n1- Lire un élément et supprimer toutes les occurences dans la liste\n2- Insertion d'un élément dans une liste simplement chaînée triée\n3- Insertion d'un élément dans une liste doublement chaînée triée\n4- Insertion en tête et en queue dans une liste simplement chaînée circulaire\n5- Insertion en tête et en queue dans une liste doublement chaînée circulaire\n");
        printf("Entrez le chiffre correspondant à votre choix : ");

        if (fgets(input, sizeof(input), stdin) == NULL) {
            break;
        }

        if (strlen(input) > 2 || (strlen(input) == 1 && input[0] == '\n')) {
            printf("Erreur ! Veuillez entrer un seul caractère.\n");
            continue;
        }

        n = input[0];

        switch(n) {
            case '1': ; break;
            case '2': ; break;
            case '3': ; break;
            case '4': ; break;
            case '5': ; break;
            case '0': printf("Au Revoir !\n"); break;
            default: printf("Erreur ! Veuillez choisir une option valable.\n");
        }
        printf("\n");
    } while(n != '0');

    return 0;
}
```

tp2.h

```c
#ifndef _TP2_
#define _TP2_

int lireEntier(const char* message, int min, int max);

#endif
```

tp2.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

```c
#include <ctype.h>
#include <string.h>
#include <errno.h>
#include "tp2.h"

#define MAX 100

int lireEntier(const char* message, int min, int max) {
    char buffer[100];
    long valeurLong;
    char *endptr;

    do {
        printf("%s", message);
        if (fgets(buffer, sizeof(buffer), stdin) == NULL) {
            printf("Erreur de lecture.\n");
            continue;
        }

        // Supprimer le saut de ligne
        buffer[strcspn(buffer, "\n")] = '\0';

        // Convertir en long
        errno = 0;
        valeurLong = strtol(buffer, &endptr, 10);

        // Vérifications
        if (endptr == buffer) {
            printf("Erreur: Veuillez entrer un nombre entier valide.\n");
        } else if (*endptr != '\0') {
            printf("Erreur: Caractères supplémentaires non autorisés.\n");
        } else if (errno == ERANGE || valeurLong < min || valeurLong > max) {
            printf("Erreur: La valeur doit être comprise entre %d et %d (valeur saisie: %ld).\n", min, max, valeurLong);
        } else {
            return (int)valeurLong;
        }
    } while (1);
}
```

Annexe B : Makefile

```
CC = gcc
CFLAGS = -Wall -Wextra
OBJECTS = main.o tp2.o

all: main

main: $(OBJECTS)
 $(CC) $(CFLAGS) -o main $(OBJECTS)

main.o: main.c tp2.h
 $(CC) $(CFLAGS) -c main.c

tp2.o: tp2.c tp2.h
 $(CC) $(CFLAGS) -c tp2.c

clean:
 rm -f main $(OBJECTS)
```