

**Algorithme et structures de données 2**

Projet Gestion d’un agenda

L2 – Groupe C

2023

ASSOUAD Adrien

BIAL Thibault

CHAN PENG Julien

Sommaire

[Introduction 2](#_Toc153129569)

[I. Notions importantes : les listes à niveau 3](#_Toc153129570)

[II. Les structures et les fonctions utilisées 4](#_Toc153129571)

[III. Les fonctions de recherches 6](#_Toc153129572)

[a. La recherche classique 6](#_Toc153129573)

[b. La recherche par niveau 7](#_Toc153129574)

[IV. Stockage de contacts dans une liste à niveaux 9](#_Toc153129575)

[Conclusion 11](#_Toc153129576)

# Introduction

Dans le cadre de ce module (Algorithme et structure de données 2), nous devions réaliser un projet en trinôme intitulé « Gestion d’un agenda ». Le but de ce projet est donc de réaliser un programme en C qui permet de gérer un agenda en utilisant une structure de données intermédiaire avec les listes. Cette structure de données permettra, dans le même temps, de :

* Conserver un ordre entre ses éléments. Ainsi, on pourra comparer les éléments deux à deux, et déterminer lequel des deux doit être rangé « avant l’autre » dans cette structure de données.
* Effectuer les opérations standard (insertion, recherche, suppression) avec une complexité qui reste faible, notamment plus faible qu’avec une liste simplement chaînée.

Concrètement, les objectifs de ce projet sont d’utiliser et comprendre le fonctionnement des listes à niveau et des listes de manière générale, ainsi que les structures et les fonctions nécessaires pour les cellules et les listes à niveau.

# Notions importantes : les listes à niveau

Au cours du semestre nous avons appris à créer et manipuler des listes chaînées reliant des cellules. Une cellule contient une valeur et pointe vers une autre cellule, et une liste chaînée pointe vers la cellule *head.*

Pour ce projet nous utiliserons le même principe mais sur plusieurs niveaux représentés par une hauteur. Le principe est le même, la seule différence est que chaque cellule pointe la suivante sur le même niveau et ne pointe pas la cellule au niveau le plus bas comme on pourrait le croire.

Voici une illustration d’une liste à niveau :

Le principe de recherche est le suivant : on commence la recherche au niveau le plus haut. Si on ne trouve pas la valeur à ce niveau, on continue la recherche au niveau précédent en repartant de la cellule à laquelle on était arrivé.

Exemple :

On recherche la valeur 7. Au niveau 2, on ne le trouve pas, on descend au niveau 1 à partir de la cellule stockant 4 (car si 7 est dans la liste, il est après 4) Au niveau 1, on ne le trouve pas, on descend au niveau 0 à partir de la cellule stockant 6 (car si 7 est dans la liste, il est après 6) Au niveau 0, on le trouve.

# Les structures et les fonctions utilisées

Les structures utilisées pour ce projet sont celles de la cellule et de la liste. La structure *t\_d\_cell* contient une valeur entière, un niveau et un pointeur vers la suivante.

Pour la structure *t\_d\_list*, elle contient simplement *niv\_max* qui indique la hauteur de la liste et un pointeur vers la première cellule.

Nous pouvons représenter ces structures de cette manière :

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Voici les codes respectifs de ces structures :

Une image contenant texte, Police, capture d’écran, conception

Description générée automatiquementUne image contenant texte, Police, capture d’écran, conception

Description générée automatiquement

Un certain nombre de fonctions nous ont permis de réaliser ce projet comme celles permettant de créer une cellule avec les informations saisies par l'utilisateur. Et d'autres pour créer et afficher une liste, ainsi que d'insérer une valeur en tête de liste et la trier.

Une image contenant texte, Police, capture d’écran

Description générée automatiquementUne image contenant texte, capture d’écran, Police

Description générée automatiquement

**Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement**

# Les fonctions de recherches

L’utilisation de liste à niveaux permet d’accélérer les recherches pour obtenir une complexité qui s’approche de celle de la recherche par dichotomie. Les valeurs étant stockées selon un critère de tri permettent d’utiliser la recherche dichotomique pour implémenter des traitements. Dans ce projet, nous avons traité deux types de recherche : la recherche classique et la recherche par niveau.

## La recherche classique

Le principe de la recherche classique est de parcourir le niveau 0 qui contient toutes les valeurs. Il faut donc simplement parcourir une liste chaînée grâce à un pointeur *temp* de type *s\_d\_cell* qui va pointer sur la tête de la liste. Si dans le champ valeur de *temp* est la même que la valeur recherchée alors on peut s’arrêter ici vu qu’on a trouvé la valeur recherchée. Sinon, on passe à la cellule suivante grâce au pointeur *temp* qui devient *temp.next*. Et on répète ce processus jusqu’à la valeur recherchée.

Un exemple ci-contre :

Une image contenant texte, Police, ligne, reçu

Description générée automatiquement

Recherchons la valeur 31.



temp.val != val donc au passe à la cellule suivante.



temp.val != val donc au passe à la cellule suivante.



temp.val = val donc on s’arrête ici car on a trouvé la valeur recherchée.

Voici le code cette fonction :

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

## La recherche par niveau

Comme son nom l’indique, on va rechercher une valeur avec les différents niveaux. Comme dit précédemment, le principe de recherche est le suivant : on commence la recherche au niveau le plus haut. Si on ne trouve pas la valeur à ce niveau, on continue la recherche au niveau précédent en repartant de la cellule à laquelle on était arrivé.

Un exemple ci-contre :

Une image contenant texte, reçu, Police, ligne

Description générée automatiquement

Recherchons la valeur 56.

Une image contenant texte, Police, reçu, ligne

Description générée automatiquement



Une image contenant texte, ligne, Police, reçu

Description générée automatiquement

Une image contenant texte, Police, ligne, capture d’écran

Description générée automatiquement



Une image contenant texte, ligne, reçu, Police

Description générée automatiquement

Une image contenant texte, ligne, Police, capture d’écran

Description générée automatiquement



Une image contenant texte, Police, ligne, capture d’écran

Description générée automatiquement

Voici le code de cette fonction :

Une image contenant texte, capture d’écran

Description générée automatiquement

* Conclusion : comparaison des méthodes de recherche

En reportant les temps de recherche de ces deux méthodes pour différents niveaux de la liste sur un excel, on peut tracer le graphique suivant :

Une image contenant texte, ligne, Tracé, capture d’écran

Description générée automatiquement

On constate que le temps d’exécution de la recherche classique évolue de manière exponentielle, tandis que celui de la recherche par niveau reste constant. En effet, grâce au module timer (timer.c / timer.h), nous avons comparé les temps d’exécution de ces deux fonctions de recherche.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

On observe que le temps d’exécution de la recherche par niveau reste constant quasiment à 0 sec tandis que celui la recherche classique va augmenter lorsque le niveau de liste devient de plus en plus important.

On peut conclure que la méthode de recherche par niveau est clairement plus efficace que la recherche classique que ce soit au niveau de la mémoire mais aussi au niveau du temps d’exécution.

# Stockage de contacts dans une liste à niveaux

Une image contenant texte, capture d’écran, Police

Description générée automatiquementOn souhaite utiliser une telle liste pour stocker des contacts et des rendez-vous, en utilisant des recherches / insertions / modifications rapides, ainsi qu’une aide à la saisie (complétion automatique). Pour ceci, il est nécessaire de créer des structures pour stocker les informations telles que le nom et prénom ainsi que les dates des rendez-vous.

Pour les saisies de tableaux dynamiques de caractères, nous avons fait une fonction *char \*scanString(void)* qui effectue la saisie et retourne un tableau dynamique stockant uniquement les caractères nécessaires.

Nous avons réalisé toutes les fonctions liées à la création de rendez-vous, de contacts et d’un agenda.

Une image contenant texte, capture d’écran, Police

Description générée automatiquementUne image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Cependant, nous avons eu du mal à réaliser certaines fonctions qui restent toujours infonctionnelles. Nous n’arrivons pas à retrier les éléments de la liste lorsqu’on y ajoute de nouvelles informations.

# Conclusion

Ce projet d’algorithmique nous a offert la possibilité de renforcer nos notions en codage C et de travailler en groupe pour partager différents points de vue sur les méthodes de travail à appliquer. Ce projet « Gestion d’un agenda » a pour but de comprendre le principe des listes à niveau et de les appliquer dans un cas concret.

En termes d’organisation, nous nous sommes bien organisés mais la grande difficulté que nous avons rencontrée est la partie 3 du projet que n’avons pas réussi à mettre en place malgré nos efforts. Sinon, ce projet fut une expérience du point de vue connaissances et du point de vue de la gestion du temps et des attendus.