

Version: 1.0



Selection

C-Arbres-Binaires

Résumé

Ce projet porte sur l'implémentation et la manipulation d'arbres binaires en C.

#C

#Structure

#ArbresBinaires

42

Avertissement relatif à la propriété intellectuelle

L'ensemble du contenu présenté dans ce module de formation — y compris, sans s'y limiter, les textes, images, graphiques et autres éléments — est protégé par les droits de propriété intellectuelle détenus par l'Association 42.

Conditions d'utilisation

- **Usage personnel** : L'utilisation du contenu de ce module est strictement réservée à un usage personnel. Toute reproduction, diffusion, modification, utilisation publique ou commerciale est interdite sans l'autorisation écrite préalable de l'Association 42.
- **Respect de l'intégrité** : Il est interdit de modifier, altérer ou adapter le contenu de manière à en dénaturer le sens ou à porter atteinte à son intégrité.

Protection des droits

Toute violation des présentes conditions constitue une atteinte aux droits de propriété intellectuelle et peut faire l'objet de poursuites. L'Association 42 se réserve le droit d'engager toute action nécessaire à la protection de ses droits, notamment par voie judiciaire et en réclamant des dommages et intérêts le cas échéant.

Pour toute question relative à l'utilisation du contenu ou pour obtenir une autorisation, veuillez contacter : legal@42.fr

Table des matières

1	Instructions	1
2	Préambule	5
3	Informations pour ce projet	6
4	Exercice 0 : btree_create_node	7
5	Exercice 1 : btree_apply_prefix	8
6	Exercice 2 : btree_apply_infix	9
7	Exercice 3 : btree_apply_suffix	10
8	Exercice 4 : btree_insert_data	11
9	Exercice 5 : btree_search_item	12
10	Exercice 6 : btree_level_count	13
11	Exercice 7 : btree_apply_by_level	14
12	Rendu et évaluation par les pairs	15

Chapitre 1

Instructions

- Seule cette page servira de référence : ne faites pas confiance aux rumeurs.
- Attention ! Ce document pourrait potentiellement changer jusqu'à la soumission.
- Assurez-vous d'avoir les permissions appropriées sur vos fichiers et répertoires.
- Vous devez suivre les procédures de soumission pour tous vos exercices.
- Vos exercices seront vérifiés et notés par vos camarades de classe.
- De plus, vos exercices seront vérifiés et notés par un programme appelé Moulinette.
- Moulinette est très méticuleuse et stricte dans son évaluation de votre travail. Elle est entièrement automatisée, et il n'y a aucun moyen de négocier avec elle. Donc, pour éviter les mauvaises surprises, soyez aussi minutieux que possible.
- Moulinette n'est pas très ouverte d'esprit. Elle n'essayera pas de comprendre votre code s'il ne respecte pas la Norme. Moulinette s'appuie sur un programme appelé `norminette` pour vérifier si vos fichiers respectent la norme. En bref : il serait stupide de soumettre un travail qui ne passe pas la vérification de `norminette`.
- Ces exercices sont soigneusement disposés par ordre de difficulté - du plus facile au plus difficile. Nous ne considérerons pas un exercice plus difficile réussi si un exercice plus facile n'est pas parfaitement fonctionnel.
- Utiliser une fonction interdite est considéré comme de la tricherie. Les tricheurs obtiennent -42, et cette note est non négociable.
- Vous n'aurez à soumettre une fonction `main()` que si nous demandons un programme.
- Moulinette compile avec ces flags : `-Wall -Wextra -Werror`, et utilise `cc`.
- Si votre programme ne compile pas, vous obtiendrez 0.
- Vous ne pouvez pas laisser aucun fichier supplémentaire dans votre répertoire autre que ceux spécifiés dans le sujet.
- Vous avez une question ? Demandez à votre pair à droite. Sinon, essayez votre pair à gauche.
- Votre guide de référence s'appelle `Google / man / Internet / ....`
- Consultez le Slack Piscine.
- Examinez attentivement les exemples. Ils pourraient très bien appeler à des détails qui ne sont pas explicitement mentionnés dans le sujet...

- Par Odin, par Thor! Utilisez votre cerveau!!!



N'oubliez pas d'ajouter l'*en-tête standard 42* dans chacun de vos fichiers .c/.h. Norminette vérifie son existence de toute façon!



Norminette doit être lancée avec le flag `-R CheckForbiddenSourceHeader`. Moulinette l'utilisera aussi.

● Contexte

La Piscine C est intense. C'est votre premier grand défi à 42 — une plongée profonde dans la résolution de problèmes, l'autonomie et la communauté.

Durant cette phase, votre objectif principal est de construire vos fondations — à travers la lutte, la répétition, et surtout l'échange d'**apprentissage par les pairs**.

À l'ère de l'IA, les raccourcis sont faciles à trouver. Cependant, il est important de considérer si votre utilisation de l'IA vous aide vraiment à grandir — ou si elle se contente de vous empêcher de développer de vraies compétences.

La Piscine est aussi une expérience humaine — et pour l'instant, rien ne peut remplacer cela. Pas même l'IA.

Pour un aperçu plus complet de notre position sur l'IA — en tant qu'outil d'apprentissage, dans le cadre du programme ICT, et en tant qu'attente croissante sur le marché du travail — veuillez vous référer à la FAQ dédiée disponible sur l'intranet.

● Message principal

- 👉 Construire des fondations solides sans raccourcis.
- 👉 Développer réellement des compétences techniques et personnelles.
- 👉 Expérimenter un véritable apprentissage par les pairs, commencer à apprendre à apprendre et à résoudre de nouveaux problèmes.
- 👉 Le parcours d'apprentissage est plus important que le résultat.
- 👉 Apprendre les risques associés à l'IA, et développer des pratiques de contrôle efficaces et des contre-mesures pour éviter les pièges courants.

● Règles pour les apprenants :

- Vous devez appliquer le raisonnement à vos tâches assignées, surtout avant de vous tourner vers l'IA.
- Vous ne devez pas demander de réponses directes à l'IA.
- Vous devez apprendre l'approche globale de 42 sur l'IA.

● Résultats de la phase :

Dans cette phase de fondation, vous obtiendrez les résultats suivants :

- Acquérir des bases techniques et de codage appropriées.
- Savoir pourquoi et comment l'IA peut être dangereuse pendant cette phase.

● Commentaires et exemple :

- Oui, nous savons que l'IA existe — et oui, elle peut résoudre vos activités. Mais vous êtes ici pour apprendre, pas pour prouver que l'IA a appris. Ne gaspillez pas votre temps (ou le nôtre) juste pour démontrer que l'IA peut résoudre le problème donné.
- Apprendre à 42 ne consiste pas à connaître la réponse — il s'agit de développer la capacité à en trouver une. L'IA vous donne la réponse directement, mais cela vous empêche de construire votre propre raisonnement. Et le raisonnement prend du temps, des efforts, et implique des échecs. Le chemin vers le succès n'est pas censé être facile.
- Gardez à l'esprit que lors des examens, l'IA n'est pas disponible — pas d'internet, pas de smartphones, etc. Vous réaliserez rapidement si vous avez trop compté sur l'IA dans votre processus d'apprentissage.
- L'apprentissage par les pairs vous expose à différentes idées et approches, améliorant vos compétences interpersonnelles et votre capacité à penser de manière divergente. C'est bien plus précieux que de simplement discuter avec un bot. Alors ne soyez pas timide — parlez, posez des questions, et apprenez ensemble !
- Oui, l'IA fera partie du programme — à la fois comme outil d'apprentissage et comme sujet en soi. Vous aurez même l'occasion de construire votre propre logiciel d'IA. Afin d'en apprendre davantage sur notre approche crescendo, vous passerez par la documentation disponible sur l'intranet.

✓ Bonne pratique :

Je suis bloqué sur un nouveau concept. Je demande à quelqu'un à proximité comment il l'a abordé. Nous parlons pendant 10 minutes — et soudain, ça fait clic. Je comprends.

✗ Mauvaise pratique :

J'utilise secrètement l'IA, je copie du code qui semble correct. Pendant l'évaluation par les pairs, je ne peux rien expliquer. J'échoue. Pendant l'examen — sans IA — je suis à nouveau bloqué. J'échoue.

Chapitre 2

Préambule

Félicitations !

Vous avez déverrouillé une page pleine de faits fascinants et de curiosités. Bonne exploration !

- **Tour Eiffel** : Elle peut grandir de 15 cm en été à cause de la dilatation du métal.
- **Abeilles** : Elles ont des poils sur les yeux pour collecter le pollen.
- **Poulpes** : Ils ont trois cœurs. Deux pour les branchies, un pour le reste du corps.
- **Vénus** : Un jour y est plus long qu'une année.
- **Bananes** : Elles sont légèrement radioactives (potassium-40).
- **Corps humain** : La guerre la plus courte a duré 38 minutes (Royaume-Uni vs Zanzibar, 1896).
- **Cléopâtre** : Elle n'était pas Égyptienne, mais d'origine grecque macédonienne.
- **Chiens** : Ils comprennent jusqu'à 250 mots et peuvent résoudre des calculs simples.
- **Arbres** : Ils communiquent via un réseau fongique souterrain (« wood wide web »).
- **Fourmis** : Elles peuvent porter plus de 10 fois leur poids.

Restez curieux, et continuez d'explorer le monde !

Chapitre 3

Informations pour ce projet

Pour tous les exercices suivants, vous devez utiliser la structure suivante :

ft_btree.h

```
typedef struct      s_btree
{
    struct s_btree  *left;
    struct s_btree  *right;
    void            *item;
}                  t_btree;
```

Vous devez inclure cette structure dans un fichier `ft_btree.h` et le soumettre pour chaque exercice.

À partir de l'exercice 01, vous utiliserez votre fonction `btree_create_node`, donc prévoyez son prototype dans `ft_btree.h`...

Chapitre 4

Exercice 0 : btree_create_node

	Exercice: 0	
btree_create_node		
Répertoire: ex0/		
Fichiers à Rendre: btree_create_node.c, ft_btree.h		
Autorisé: malloc		

- Créez la fonction `btree_create_node` qui alloue un nouvel élément.
- Elle doit initialiser son `item` avec la valeur passée en argument, et les autres champs à 0.
- Elle retourne l'adresse du nœud créé.

Prototype :

```
t_btree *btree_create_node(void *item);
```

Chapitre 5

Exercice 1 : btree_apply_prefix

	Exercice: 1	
btree_apply_prefix		
Répertoire: ex1/		
Fichiers à Rendre: btree_apply_prefix.c, ft_btree.h		
Autorisé: None		

- Créez une fonction `btree_apply_prefix` qui applique une fonction à chaque item du nœud, en parcourant l'arbre en **préfixe**.

Prototype :

```
void btree_apply_prefix(t_btree *root, void (*applyf)(void *));
```

Chapitre 6

Exercice 2 : btree_apply_infix

	Exercice: 2	
btree_apply_infix		
Répertoire: ex2/		
Fichiers à Rendre: btree_apply_infix.c, ft_btree.h		
Autorisé: None		

- Créez une fonction `btree_apply_infix` qui applique une fonction à chaque item du nœud, en parcourant l'arbre en **infixe**.

Prototype :

```
void btree_apply_infix(t_btree *root, void (*applyf)(void *));
```

Chapitre 7

Exercice 3 : btree_apply_suffix

	Exercice: 3	
btree_apply_suffix		
Répertoire: ex3/		
Fichiers à Rendre: btree_apply_suffix.c, ft_btree.h		
Autorisé: None		

- Créez une fonction `btree_apply_suffix` qui applique une fonction à chaque item du nœud, en parcourant l'arbre en **suffixe**.

Prototype :

```
void btree_apply_suffix(t_btree *root, void (*applyf)(void *));
```

Chapitre 8

Exercice 4 : btree_insert_data

	Exercice: 4	
btree_insert_data		
Répertoire: ex4/		
Fichiers à Rendre: btree_insert_data.c, ft_btree.h		
Autorisé: btree_create_node		

- Créez une fonction `btree_insert_data` qui insère un nouvel élément dans un arbre binaire trié.
- Tous les éléments inférieurs sont à gauche, les autres à droite.
- Une fonction de comparaison de type `strcmp` est passée en paramètre.

Prototype :

```
void btree_insert_data(t_btree **root, void *item, int (*cmpf)(void *, void *));
```

Chapitre 9

Exercice 5 : btree_search_item

	Exercice: 5	
btree_search_item		
Répertoire: ex5/		
Fichiers à Rendre: btree_search_item.c, ft_btree.h		
Autorisé: None		

- Créez une fonction `btree_search_item` qui retourne l'élément correspondant à la donnée recherchée.
- L'arbre est parcouru en infixe. Retourne `NULL` si l'élément n'est pas trouvé.

Prototype :

```
void *btree_search_item(t_btree *root, void *data_ref, int (*cmpf)(void *, void *));
```

Chapitre 10

Exercice 6 : btree_level_count

	Exercice: 6	
btree_level_count		
Répertoire: ex6/		
Fichiers à Rendre: btree_level_count.c, ft_btree.h		
Autorisé: None		

- Créez une fonction `btree_level_count` qui retourne la hauteur maximale de l'arbre passé en paramètre.

Prototype :

```
int btree_level_count(t_btree *root);
```

Chapitre 11

Exercice 7 : btree_apply_by_level

	Exercice: 7	
btree_apply_by_level		
Répertoire: ex7/		
Fichiers à Rendre: btree_apply_by_level.c, ft_btree.h		
Autorisé: malloc, free		

- Créez une fonction `btree_apply_by_level` qui applique une fonction à chaque élément, niveau par niveau.
- La fonction appelée prend en paramètre :
 - un `void *item`
 - un `int current_level`
 - un `int is_first_elem`

Prototype :

```
void btree_apply_by_level(t_btree *root, void (*applyf)(void *, int, int));
```

Chapitre 12

Rendu et évaluation par les pairs

Soumettez votre projet dans votre dépôt Git comme d'habitude. Seul le contenu de ce dépôt sera évalué. N'oubliez pas de vérifier le nom des fichiers et leur bonne orthographe.



Vous ne devez soumettre que les fichiers demandés dans le sujet.