



Piscina C

C 13

Sumário: Este documento é o tema do módulo C 13 da Piscina C da 42.

Conteúdo

I	Instruções	2
II	Preâmbulo	4
III	Exercice 00 : btree_create_node	5
IV	Exercício 01 : btree_apply_prefix	6
V	Exercício 02 : btree_apply_infix	7
VI	Exercício 03 : btree_apply_suffix	8
VII	Exercício 04 : btree_insert_data	9
VIII	Exercício 05 : btree_search_item	10
IX	Exercício 06 : btree_level_count	11
X	Exercício 07 : btree_apply_by_level	12

Capítulo I

Instruções

- Somente esta página servirá de referência, não confie nos boatos.
- Releia bem o tema antes de entregar seus exercícios. A qualquer momento o tema pode mudar.
- Atenção aos direitos de seus arquivos e suas pastas.
- Você deve seguir o procedimento de entrega para todos os seus exercícios.
- Os seus exercícios serão corrigidos por seus colegas de piscina.
- Além dos seus colegas, haverá a correção de um programa chamado Moulinette.
- A Moulinette é muito rigorosa na sua avaliação. Ela é completamente automatizada. É impossível discutir sua nota com ela. Tenha um rigor exemplar para evitar surpresas.
- A Moulinette não tem a mente muito aberta. Ela não tenta entender o código que não respeita a Norma. A Moulinette utiliza o programa **norminette** para verificar a norma dos seus arquivos. Então é uma tolice entregar um código que não passa pela **norminette**.
- Os exercícios são rigorosamente ordenados do mais simples ao mais complexo. Em nenhum caso daremos atenção, nem levaremos em conta um exercício complexo se outro mais simples não tiver sido perfeitamente realizado.
- A utilização de uma função proibida é um caso de fraude. Qualquer fraude é punida com nota de **-42**.
- Você não deve entregar uma função `main()` se nós pedirmos um programa.
- A Moulinette compila com as sinalizações `-Wall -Wextra -Werror`, e utiliza `gcc`.
- Se o seu programa não compila, você terá 0.

- Você não deve deixar em sua pasta nenhum outro arquivo além daqueles explicitamente especificados nos enunciados dos exercícios.
- Você tem alguma dúvida? Pergunte ao seu vizinho da direita. Ou tente também perguntar ao seu vizinho da esquerda.
- Seu manual de referência se chama `Google / man / Internet /`
- Considere discutir no fórum Piscina do seu Intra, assim como no slack da sua Piscina!
- Leia atentamente os exemplos. Eles podem muito bem pedir coisas que não estão especificadas no tema...
- Reflita. Por favor, por Odin! Por tudo que é mais sagrado.
- Para os exercícios de hoje, vamos usar a seguinte estrutura:

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

- Você deve colocar essa estrutura em um arquivo `ft_btree.h` e entregá-lo a cada exercício.
- A partir do exercício 01 vamos utilizar nosso `btree_create_node`, tome as medidas necessárias (pode ser interessante ter seu protótipo em `ft_btree.h...`).

Capítulo II

Preâmbulo


Veja a seguir uma lista dos discos da banda **Venom**:

- In League with Satan (single, 1980)
- Welcome to Hell (1981)
- Black Metal (1982)
- Bloodlust (single, 1983)
- Die Hard (single, 1983)
- Warhead (single, 1984)
- At War with Satan (1984)
- Hell at Hammersmith (EP, 1985)
- American Assault (EP, 1985)
- Canadian Assault (EP, 1985)
- French Assault (EP, 1985)
- Japanese Assault (EP, 1985)
- Scandinavian Assault (EP, 1985)
- Manitou (single, 1985)
- Nightmare (single, 1985)
- Possessed (1985)
- German Assault (EP, 1987)
- Calm Before the Storm (1987)
- Prime Evil (1989)
- Tear Your Soul Apart (EP, 1990)
- Temples of Ice (1991)
- The Waste Lands (1992)
- Venom '96 (EP, 1996)
- Cast in Stone (1997)
- Resurrection (2000)
- Anti Christ (single, 2006)
- Metal Black (2006)
- Hell (2008)
- Fallen Angels (2011)

O tema de hoje ficará mais fácil se você trabalhar escutando **Venom**.

Capítulo III

Exercice 00 : btree_create_node


	Exercício : 00
btree_create_node	
Pasta de entrega : <i>ex00/</i>	
Arquivos para entregar : <code>btree_create_node.c</code> , <code>ft_btree.h</code>	
Funções autorizadas : <code>malloc</code>	

- Escreva a função `btree_create_node` que aloca um novo elemento, inicie seu `item` com o valor do parâmetro e todos os outros elementos com 0.
- O endereço do nó criado é retornado.
- Ela deverá ser prototipada da seguinte forma:

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

Capítulo IV

Exercício 01 : btree_apply_prefix


	Exercício : 01
btree_apply_prefix	
Pasta de entrega : <i>ex01/</i>	
Arquivos para entregar : <i>btree_apply_prefix.c, ft_btree.h</i>	
Funções autorizadas : <i>Nenhuma</i>	

- Escreva a função `btree_apply_prefix` que aplica a função passada como parâmetro ao `item` de cada nó, percorrendo a árvore de forma **prefix**.
- Ela deverá ser prototipada da seguinte forma:

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

Capítulo V

Exercício 02 : btree_apply_infix


	Exercício : 02
btree_apply_infix	
Pasta de entrega : <i>ex02/</i>	
Arquivos para entregar : <code>btree_apply_infix.c</code> , <code>ft_btree.h</code>	
Funções autorizadas : Nenhuma	

- Escreva a função `btree_apply_infix` que aplica a função passada como parâmetro ao `item` de cada nó, percorrendo a árvore de forma `infix`.
- Ela deverá ser prototipada da seguinte forma:

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


Capítulo VI

Exercício 03 : btree_apply_suffix


	Exercício : 03
btree_apply_suffix	
Pasta de entrega : <i>ex03/</i>	
Arquivos para entregar : <i>btree_apply_suffix.c, ft_btree.h</i>	
Funções autorizadas : <i>Nenhuma</i>	

- Escreva a função `btree_apply_suffix` que aplica a função passada como parâmetro ao `item` de cada nó, percorrendo a árvore de forma `suffix`.
- Ela deverá ser prototipada da seguinte forma:

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

Capítulo VII

Exercício 04 : btree_insert_data


	Exercício : 04
btree_insert_data	
Pasta de entrega : <i>ex04/</i>	
Arquivos para entregar : <i>btree_insert_data.c, ft_btree.h</i>	
Funções autorizadas : <i>btree_create_node</i>	

- Escreva a função `btree_insert_data` que insere o elemento `item` em uma árvore. A árvore passada como parâmetro será organizada: por cada nó todos os elementos inferiores encontram-se à esquerda e todos os elementos superiores ou iguais, à direita. Vamos enviar como parâmetro uma função de comparação que tem o mesmo comportamento que `strcmp`.
- O parâmetro `root` aponta para o nó raiz da árvore. Na primeira chamada, ele aponta para `NULL`.
- Ela deverá ser prototipada da seguinte forma:

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

Capítulo VIII

Exercício 05 : btree_search_item


	Exercício : 05
btree_search_item	
Pasta de entrega : <i>ex05/</i>	
Arquivos para entregar : <i>btree_search_item.c, ft_btree.h</i>	
Funções autorizadas : Nenhuma	

- Escreva a função `btree_search_item` que retorna o primeiro elemento correspondente ao dado de referência passado como parâmetro. A árvore deverá ser percorrida de forma **infix**. Se o elemento não for encontrado, a função deverá retornar `NULL`.
- Ela deverá ser prototipada da seguinte forma:

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

Capítulo IX

Exercício 06 : btree_level_count


	Exercício : 06
btree_level_count	
Pasta de entrega : <i>ex06/</i>	
Arquivos para entregar : <code>btree_level_count.c</code> , <code>ft_btree.h</code>	
Funções autorizadas : Nenhuma	

- Escreva a função `btree_level_count` que retorna a altura do maior ramo passado como parâmetro.
- Ela deverá ser prototipada da seguinte forma:

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

Capítulo X

Exercício 07 : btree_apply_by_level

	Exercício : 07
btree_apply_by_level	
Pasta de entrega : <i>ex07/</i>	
Arquivos para entregar : <i>btree_apply_by_level.c, ft_btree.h</i>	
Funções autorizadas : <i>malloc, free</i>	

- Escreva a função `btree_apply_by_level` que aplica a função passada como parâmetro a cada nó da árvore. A árvore deve percorrer andar por andar. A função chamada terá três parâmetros:
 - O primeiro parâmetro, de tipo `void *`, corresponde ao item do nó;
 - O segundo parâmetro, de tipo `int`, corresponde ao nível no qual estamos: 0 para a raiz, 1 para seus filhos, 2 para seus netos, etc. ;
 - O terceiro parâmetro, de tipo `int`, vale 1 se for do primeiro nó do nível, caso contrário: 0.
- Ela deverá ser prototipada da seguinte forma:

```
void btree_apply_by_level(t_btree *root, void (*applyf)(void *item, int current_level, int is_first_elem))
```