



# Piscina C

## C 13

*Sumário: ESTE documento é o enunciado do módulo C 13 da Piscina C da 42.*

*Versão: 4.2*

# 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
XI	Submissão e avaliação	13

# Capítulo I

## Instruções

- Somente este documento servirá de referência; não confie nos boatos.
- Leia bem o enunciado antes de entregar os seus exercícios. A qualquer momento pode haver alterações.
- Tenha atenção aos direitos dos seus ficheiros e pastas.
- Deverá seguir o procedimento de entrega para todos os exercícios.
- Os seus exercícios serão corrigidos pelos seus colegas de piscine.
- Além dos seus colegas, a Moulinette também corrigirá os seus exercícios.
- A Moulinette é extremamente rígida na sua avaliação. É completamente automatizada, e é impossível discutir a sua nota com ela. Portanto, seja rigoroso!
- A Moulinette não tem uma mente muito aberta: não tenta entender código que não respeita a Norma. A Moulinette utiliza o programa `norminette` para verificar a norma dos ficheiros. Seria uma tontice entregar código que não passa pela `norminette`...
- Os exercícios são ordenados precisamente do mais simples ao mais complexo. Em caso algum consideraremos um exercício mais complexo se outro mais simples não tiver sido perfeitamente realizado.
- A utilização de qualquer função proibida é um caso de fraude. Qualquer fraude é punida com nota de -42.
- Deve entregar uma função `main()` se for pedido um programa.
- A Moulinette compila com as flags `-Wall -Wextra -Werror`, e utiliza `cc`.
- Se o seu programa não compila, terá 0.
- Você não deve deixar em sua pasta nenhum outro arquivo além daqueles explicitamente especificados pelos 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...
- Não deve deixar no repositório de entrega nenhum outro ficheiro além daqueles explicitamente especificados pelo enunciado dos exercícios.
- Tem alguma dúvida? Pergunte ao seu vizinho da direita. Tente, também, com o seu vizinho da esquerda.
- A bibliografia para consulta chama-se `Google / man / Internet / ....`
- Considere discutir os exercícios no Slack da sua piscine!
- Leia atentamente os exemplos: podem demonstrar coisas que não estão especificadas no enunciado...
- Para os seguintes exercícios, será usada a seguinte estrutura :

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

- Deverá incluir esta estrutura no ficheiro `ft_btree.h` e submeter o ficheiro em cada exercício.
- A partir do exercício 01, a função `btree_create_node` será usada regularmente, por isso ajuste o seu código (poderá ser útil ter o seu protótipo no ficheiro `ft_btree.h...`).

# Capítulo II

## Preâmbulo


Here's the list of releases for 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)

Today's subject will seem easier if you listen to **Venom**.

# Capítulo III

## Exercice 00 : btree\_create\_node


	Exercício : 00
	btree_create_node
	Pasta de entrega : <i>ex00/</i>
	Ficheiros 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. Inicialize o novo `item` com o valor do parâmetro passado e todos os outros elementos com 0.
- O endereço do nó criado é retornado.
- 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>	
Ficheiros para entregar : <i>btree_apply_prefix.c, ft_btree.h</i>	
Funções autorizadas : Nenhuma	

- Escreva a função `btree_apply_prefix` que aplica a função passada como parâmetro ao item de cada nó, usando `prefix traversal` para percorrer a árvore.
- 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>	
Ficheiros 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, usando `infix traversal` para percorrer a árvore.
- 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>	
Ficheiros para entregar : <code>btree_apply_suffix.c</code> , <code>ft_btree.h</code>	
Funções autorizadas : Nenhuma	

- Escreva a função `btree_apply_suffix` que aplica a função passada como parâmetro ao `item` de cada nó, usando `sufix traversal` para percorrer a árvore.
- 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>
	Ficheiros para entregar : <b>btree_insert_data.c</b> , <b>ft_btree.h</b>
	Funções autorizadas : <b>btree_create_node</b>

- 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. Será enviada 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, deverá apontar para **NULL**.
- 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>
	Ficheiros para entregar : <code>btree_search_item.c</code> , <code>ft_btree.h</code>
	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 usando `infix traversal`. Se o elemento não for encontrado, a função deverá retornar `NULL`.
- 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>
	Ficheiros para entregar : <b>btree_level_count.c</b> , <b>ft_btree.h</b>
	Funções autorizadas : <b>Nenhuma</b>

- Escreva a função `btree_level_count` que retorna a profundidade do maior ramo passado como parâmetro.
- 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>
	Ficheiros para entregar : <b>btree_apply_by_level.c</b> , <b>ft_btree.h</b>
	Funções autorizadas : <b>malloc</b> , <b>free</b>

- Escreva a função **btree\_apply\_by\_level** que aplica a função passada como parâmetro a cada nó da árvore. A árvore deve percorrer nível de profundidade por nível de profundidade. A função chamada terá três parâmetros:
  - O primeiro parâmetro, do tipo **void \***, corresponde ao item do nó;
  - O segundo parâmetro, do tipo **int**, corresponde ao nível de profundidade no qual estamos: 0 para a raiz, 1 para seus filhos, 2 para seus netos, etc. ;
  - O terceiro parâmetro, do tipo **int**, vale 1 se for o primeiro nó do nível, caso contrário vale 0.
- 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))
```

# Capítulo XI

## Submissão e avaliação

Entrega o teu trabalho no teu repositório `Git`, como habitual. Apenas o trabalho dentro do teu repositório será avaliado durante a defesa. Não hesites em confirmar os nomes dos teus ficheiros para ter a certeza que estão corretos.



Apenas precisas de entregar os ficheiros pedidos no enunciado deste projeto.