

Piscina C C 12

Sumário: Este documento é o tema do módulo C 12 da Piscina C da 42.

Versão: 7.2

# Conteúdo

1	Preambulo	2
II	Instruções	4
ш	Exercício 00 : ft_create_elem	6
IV	Exercice 01 : ft_list_push_front	7
V	Exercício 02 : ft_list_size	8
VI	Exercício 03 : ft_list_last	9
VII	Exercício 04 : ft_list_push_back	10
VIII	Exercício 05 : ft_list_push_strs	11
IX	Exercício 06 : ft_list_clear	12
$\mathbf{X}$	Exercício 07 : ft_list_at	13
XI	Exercício 08 : ft_list_reverse	14
XII	Exercício 09 : ft_list_foreach	15
XIII	Exercício 10 : ft_list_foreach_if	16
XIV	Exercício 11 : ft_list_find	17
XV	Exercício 12 : ft_list_remove_if	18
XVI	Exercício 13 : ft_list_merge	19
XVII	Exercício 14 : ft_list_sort	20
XVIII	Exercício 15 : ft_list_reverse_fun	21
XIX	Exercício 16 : ft_sorted_list_insert	22
XX	Exercício 17 : ft_sorted_list_merge	23
XXI	Submissão e avaliação	24

# Capítulo I Preâmbulo

ALERTA DE SPOILER NÃO LEIA A PRÓXIMA PÁGINA

#### You've been warned.

- In Star Wars, Dark Vador is Luke's Father.
- In The Usual Suspects, Verbal is Keyser Soze.
- In Fight Club, Tyler Durden and the narrator are the same person.
- In Sixth Sens, Bruce Willis is dead since the beginning.
- In The others, the inhabitants of the house are ghosts and vice-versa.
- In Bambi, Bambi's mother dies.
- In The Village, monsters are the villagers and the movie actually takes place in our time.
- In Harry Potter, Dumbledore dies.
- In Planet of apes, the movie takes place on earth.
- In Game of thrones, Robb Stark and Joffrey Baratheon die on their wedding day.
- In Twilight, Vampires shine under the sun.
- In Stargate SG-1, Season 1, Episode 18, O'Neill and Carter are in Antartica.
- In The Dark Knight Rises, Miranda Tate is Talia Al'Gul.
- In Super Mario Bros, The princess is in another castle.

#### Capítulo II

#### Instruções

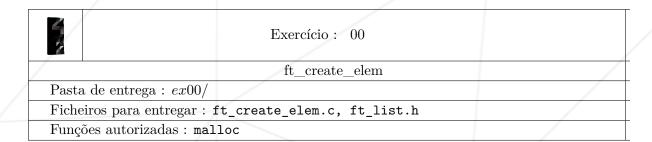
- Somente este documento servirá de referência; não confie nos boatos.
- Releia bem o enunciado antes de entregar seus exercícios. A qualquer momento pode haver alterações.
- Os exercícios são ordenados precisamente do mais simples ao mais complexo. Em caso algum consideraremos um exercício mais complexo se outro mais simples não tiver sido perfeitamente realizado.
- Tenha atenção aos direitos dos seus ficheiros 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, sejam rigorosos!
- Os exercícios de shell devem ser executados com /bin/sh.
- <u>Não deve</u> deixar no repositório de entrega <u>nenhum</u> outros ficheiros além daqueles explicitamente especificados pelos enunciados 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...

Piscina C C 12

- Para os seguintes exercicios, é necessário usar a estrutura seguinte:
- Deves incluir esta estrutura no ficheiro ft\_list.h e entregar esse ficheiro em cada exercicio.
- A partir do exercicio 01, iremos usar o nosso ft\_create\_elem, então tem isso em consideração (pode ser util ter o prototipo no ficheiro ft\_list.h...).

# Capítulo III

Exercício 00 : ft\_create\_elem



- Escreve a função ft\_create\_elem que cria um novo elemento do tipo t\_list.
- Deve atribuir data ao parâmetro fornecido e next a NULL.
- Deve ser prototipada da seguinte forma:

# Capítulo IV

# Exercice 01: ft\_list\_push\_front

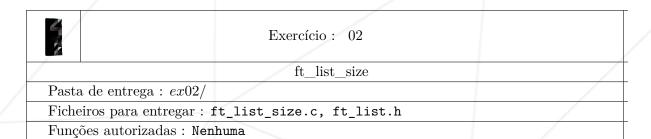
	Exercício: 01	
	ft_list_push_front	
Pasta de entrega : $ex01/$		
Ficheiros para entregar : :	ft_list_push_front.c, ft_list.h	
Funções autorizadas : ft_	create_elem	

- Escreve a função ft\_list\_push\_front que acrescenta ao início da lista um novo elemento de tipo t\_list.
- Deve atribuir data ao parâmetro fornecido.
- Se necessário, vai atualizar o ponteiro para o início da lista.
- Deve ser prototipada da seguinte forma:

roid ft\_list\_push\_front(t\_list \*\*begin\_list, void \*data);

# Capítulo V

Exercício 02 : ft\_list\_size

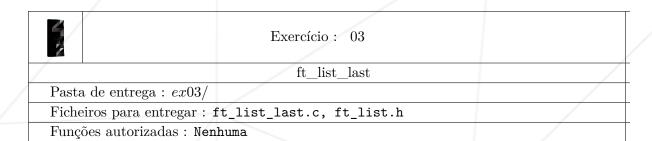


- Escreve a função ft\_list\_size que retorna o número de elementos da lista.
- Deve ser prototipada da seguinte forma:

int ft\_list\_size(t\_list \*begin\_list);

# Capítulo VI

Exercício 03 : ft\_list\_last



- Escreve a função ft\_list\_last que retorna o último elemento da lista.
- Deve ser prototipada da seguinte forma:

t\_list \*ft\_list\_last(t\_list \*begin\_list);

# Capítulo VII

### Exercício 04 : ft\_list\_push\_back

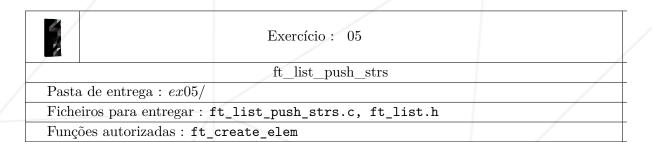
	Exercício: 04	
/	ft_list_push_back	
Pasta de entrega : $ex04/$		/
Ficheiros para entregar : :	ft_list_push_back.c, ft_list.h	/
Funções autorizadas : ft_	create_elem	/

- Escreve a função ft\_list\_push\_back que acrescenta no final da lista um novo elemento de tipo t\_list.
- Deve atribuir data ao parâmetro fornecido.
- Se necessário, vai atualizar o ponteiro para o início da lista.
- Deve ser prototipada da seguinte forma:

void ft\_list\_push\_back(t\_list \*\*begin\_list, void \*data);

### Capítulo VIII

Exercício 05 : ft\_list\_push\_strs



- Escreve a função ft\_list\_push\_strs que cria uma nova lista, incluindo nela todas as strings apontadas pelos elementos do array strs.
- size é o tamanho de strs.
- O primeiro elemento do deve estar no final da lista.
- O endereço do primeiro elemento da lista é retornado.
- Deve ser prototipada da seguinte forma:

t\_list \*ft\_list\_push\_strs(int size, char \*\*strs);

# Capítulo IX

Exercício 06 : ft\_list\_clear

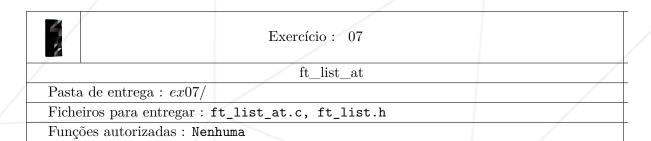
	Exercício: 06	
	ft_list_clear	/
Pasta de entrega : $ex06/$		
Ficheiros para entregar : ft_list_clear.c, ft_list.h		/
Funções autorizadas : fr	ee	

- Escreve a função ft\_list\_clear que remove e liberta todos os elementos da lista.
- O free\_fct é usado para libertar cada data
- Deve ser prototipada da seguinte forma:

void ft\_list\_clear(t\_list \*begin\_list, void (\*free\_fct)(void \*));

### Capítulo X

Exercício 07 : ft\_list\_at

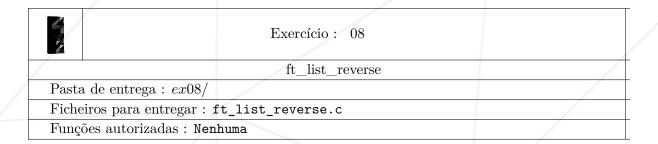


- Escreve a função ft\_list\_at que retorna o n-ésimo elemento da lista, sabendo que o primeiro elemento é quando o nbr for 0
- Em caso de erro, retorna um ponteiro nulo.
- Deve ser prototipada da seguinte forma:

t\_list \*ft\_list\_at(t\_list \*begin\_list, unsigned int nbr);

### Capítulo XI

Exercício 08 : ft\_list\_reverse

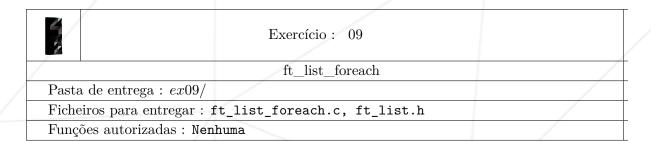


- Escreve a função ft\_list\_reverse que inverte a ordem dos elementos da lista. O valor de cada elemento deve manter se o mesmo.
- Atenção: neste exercício vamos usar nosso próprio ft\_list.h
- Deve ser prototipada da seguinte forma:

void ft\_list\_reverse(t\_list \*\*begin\_list);

# Capítulo XII

Exercício 09 : ft\_list\_foreach



- Escreve a função ft\_list\_foreach que aplica uma função fornecida como parâmetro a cada elemento da lista.
- f deve ser aplicada na ordem dos elementos da lista
- Deve ser prototipada da seguinte forma:

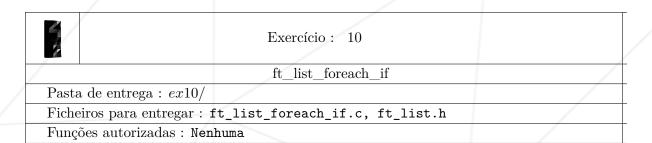
```
void ft_list_foreach(t_list *begin_list, void (*f)(void *));
```

• A função apontada por f será utilizada da seguinte forma:

(\*f)(list\_ptr->data);

#### Capítulo XIII

#### Exercício 10: ft\_list\_foreach\_if



- Escreve a função ft\_list\_foreach\_if que aplica uma função dada como parâmetro em determinados elementos da lista.
- f só será aplicada nos elementos quando o cmp com data ref, cmp retornem 0
- f deve ser aplicada na ordem dos elementos da lista
- Deve ser prototipada da seguinte forma:

```
void ft_list_foreach_if(t_list *begin_list, void (*f)(void *), void
*data_ref, int (*cmp)())
```

• As funções apontadas por f e por cmp serão usadas da seguinte forma:

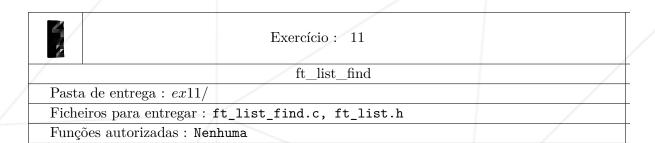
```
(*f)(list_ptr->data);
(*cmp)(list_ptr->data, data_ref);
```



A função cmp pode ser, por exemplo, ft\_strcmp...

### Capítulo XIV

# Exercício 11: ft\_list\_find



- Escreve a função ft\_list\_find que retorna o endereço do primeiro elemento cujo data comparado a data\_ref com cmp faz com que cmp retorne 0.
- Deve ser prototipada da seguinte forma:

```
t_list *ft_list_find(t_list *begin_list, void *data_ref, int (*cmp)());
```

• A função apontada por cmp será usada da seguinte forma:

(\*cmp)(list\_ptr->data, data\_ref);

### Capítulo XV

# Exercício 12: ft\_list\_remove\_if

	Exercício: 12	
/	ft_list_remove_if	/
Pasta de entrega : $ex12$ /		
Ficheiros para entregar :	ft_list_remove_if.c, ft_list.h	
Funções autorizadas : fr	ee	/

- Escreve a função ft\_list\_remove\_if que apaga da lista todos os elementos cujo a data comparado a data\_ref, com o auxílio de cmp faz com que cmp retorne 0.
- O data de um elemento que será apagado deverá também ser libertado com o auxílio de free\_fct
- Deve ser prototipada da seguinte forma:

```
void ft_list_remove_if(t_list **begin_list, void *data_ref, int (*cmp)(), void (*free_fct)(void *)
```

• As funções apontadas por free\_fct e por cmp serão usadas da seguinte forma:

```
(*cmp)(list_ptr->data, data_ref);
(*free_fct)(list_ptr->data);
```

# Capítulo XVI

# Exercício 13: ft\_list\_merge

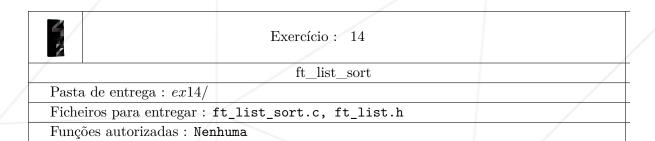
	Exercício: 13	
	ft_list_merge	
Pasta de entrega : $ex13/$		
Ficheiros para entregar : ft_list_merge.c, ft_list.h		
Funções autorizadas : Nen	huma	/

- Escreve a função ft\_list\_merge que coloca os elementos de uma lista begin2 no fim de outra lista begin1.
- A criação de elementos não é permitida.
- Deve ser prototipada da seguinte forma:

void ft\_list\_merge(t\_list \*\*begin\_list1, t\_list \*begin\_list2);

### Capítulo XVII

# Exercício 14: ft\_list\_sort



- Escreve a função ft\_list\_sort que organiza em ordem crescente o conteúdo da lista, ao comparar a data de dois elementos com uma função
- Deve ser prototipada da seguinte forma:

```
void ft_list_sort(t_list **begin_list, int (*cmp)());
```

• A função apontada por cmp será usada da seguinte forma:

```
(*cmp)(list_ptr->data, other_list_ptr->data);
```



A função cmp pode ser, por exemplo, ft\_strcmp.

# Capítulo XVIII

Exercício 15 : ft\_\_list\_\_reverse\_\_fun

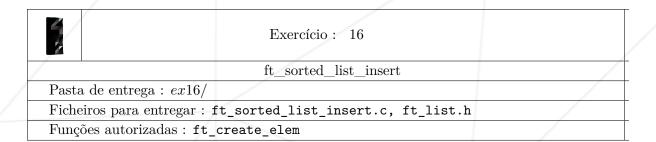
	Exercício: 15	
	ft_list_reverse_fun	
Pasta de entrega : $ex15/$		/
Ficheiros para entregar :	ft_list_reverse_fun.c, ft_list.h	/
Funções autorizadas : Ner	nhuma	/

- $\bullet\,$ Escreve a função ft\_list\_reverse\_fun que inverte a ordem dos elementos da lista.
- Deve ser prototipada da seguinte forma:

void ft\_list\_reverse\_fun(t\_list \*begin\_list);

### Capítulo XIX

Exercício 16: ft\_sorted\_list\_insert



- Escreve a função ft\_sorted\_list\_insert que cria um novo elemento e o insere em uma lista organizada de modo que a lista fique em ordem crescente.
- Deve ser prototipada da seguinte forma:

```
void ft_sorted_list_insert(t_list **begin_list, void *data, int (*cmp)());
```

• A função apontada por cmp será usada da seguinte forma:

(\*cmp)(list\_ptr->data, other\_list\_ptr->data);

### Capítulo XX

# Exercício 17: ft\_sorted\_list\_merge

1	Exercício: 17	
	ft_sorted_list_merge	/
Pasta	de entrega: $ex17/$	/
Fiche	iros para entregar : ft_sorted_list_merge.c, ft_list.h	
Funç	ões autorizadas : Nenhuma	/

- Escreve a função ft\_sorted\_list\_merge que integra os elementos de uma lista organizada begin2 em uma outra lista organizada begin1, de modo que a lista begin1 fique em ordem crescente.
- Deve ser prototipada da seguinte forma:

```
void ft_sorted_list_merge(t_list **begin_list1, t_list *begin_list2, int (*cmp)());
```

• A função apontada por cmp será usada da seguinte forma:

(\*cmp)(list\_ptr->data, other\_list\_ptr->data);

# Capítulo XXI

### 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.