



Piscina C

C 12

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

Versão: 7.2

Conteúdo

I	Preâmbulo	2
II	Instruções	4
III	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
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.
- Leia 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`.
- Não deve deixar no repositório de entrega nenhum 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...

```
typedef struct          s_list
{
    struct s_list      *next;
    void               *data;
}                      t_list;
```

- Para os seguintes exercícios, é necessário usar a estrutura seguinte:
- Deves incluir esta estrutura no ficheiro `ft_list.h` e entregar esse ficheiro em cada exercício.
- A partir do exercício 01, iremos usar o nosso `ft_create_elem`, então tem isso em consideração (pode ser útil ter o prototipo no ficheiro `ft_list.h...`).

Capítulo III

Exercício 00 : ft_create_elem


	Exercício : 00
	ft_create_elem
	Pasta de entrega : <i>ex00/</i>
	Ficheiros para entregar : <i>ft_create_elem.c</i> , <i>ft_list.h</i>
	Funções autorizadas : <i>malloc</i>

- 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:

```
t_list *ft_create_elem(void *data);
```

Capítulo IV

Exercice 01 : ft_list_push_front


	Exercício : 01
	ft_list_push_front
	Pasta de entrega : <i>ex01/</i>
	Ficheiros para entregar : <code>ft_list_push_front.c</code> , <code>ft_list.h</code>
	Funções autorizadas : <code>ft_create_elem</code>

- 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:

```
void      ft_list_push_front(t_list **begin_list, void *data);
```


Capítulo V

Exercício 02 : ft_list_size


	Exercício : 02
	ft_list_size
	Pasta de entrega : <i>ex02/</i>
	Ficheiros para entregar : <i>ft_list_size.c</i> , <i>ft_list.h</i>
	Funções autorizadas : Nenhuma

- 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


	Exercício : 03
	ft_list_last
	Pasta de entrega : <i>ex03/</i>
	Ficheiros para entregar : <i>ft_list_last.c</i> , <i>ft_list.h</i>
	Funções autorizadas : Nenhuma

- 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 : <i>ex04/</i>	
Ficheiros para entregar : <code>ft_list_push_back.c</code> , <code>ft_list.h</code>	
Funções autorizadas : <code>ft_create_elem</code>	

- 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


	Exercício : 05
	ft_list_push_strs
	Pasta de entrega : <i>ex05/</i>
	Ficheiros para entregar : <code>ft_list_push_strs.c</code> , <code>ft_list.h</code>
	Funções autorizadas : <code>ft_create_elem</code>

- 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 : <i>ex06/</i>
	Ficheiros para entregar : <i>ft_list_clear.c</i> , <i>ft_list.h</i>
	Funções autorizadas : <i>free</i>

- 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


	Exercício : 07
	ft_list_at
	Pasta de entrega : <i>ex07/</i>
	Ficheiros para entregar : <i>ft_list_at.c, ft_list.h</i>
	Funções autorizadas : Nenhuma

- Escreve a função `ft_list_at` que retorna o n-ésimo elemento da lista, sabendo que o primeiro elemento é o elemento 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


	Exercício : 08
	ft_list_reverse
	Pasta de entrega : <i>ex08/</i>
	Ficheiros para entregar : <code>ft_list_reverse.c</code>
	Funções autorizadas : Nenhuma

- 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

	Exercício : 09
	ft_list_foreach
	Pasta de entrega : <i>ex09/</i>
	Ficheiros para entregar : <i>ft_list_foreach.c, ft_list.h</i>
	Funções autorizadas : Nenhuma

- 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

	Exercício : 10
	ft_list_foreach_if
	Pasta de entrega : <i>ex10/</i>
	Ficheiros para entregar : <i>ft_list_foreach_if.c, ft_list.h</i>
	Funções autorizadas : Nenhuma

- 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

	Exercício : 11
	ft_list_find
	Pasta de entrega : <i>ex11/</i>
	Ficheiros para entregar : <i>ft_list_find.c, ft_list.h</i>
	Funções autorizadas : Nenhuma

- 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 : <i>ex12/</i>
	Ficheiros para entregar : <i>ft_list_remove_if.c</i> , <i>ft_list.h</i>
	Funções autorizadas : <i>free</i>

- Escreve a função `ft_list_remove_if` que apaga da lista todos os elementos cujo `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 : <i>ex13/</i>	
Ficheiros para entregar : <code>ft_list_merge.c</code> , <code>ft_list.h</code>	
Funções autorizadas : Nenhuma	

- 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

	Exercício : 14
	ft_list_sort
	Pasta de entrega : <i>ex14/</i>
	Ficheiros para entregar : <i>ft_list_sort.c</i> , <i>ft_list.h</i>
	Funções autorizadas : Nenhuma

- Escreve a função `ft_list_sort` que organiza em ordem crescente o conteúdo da lista, ao comparar dois elementos ao comparar a data 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 : <i>ex15/</i>
	Ficheiros para entregar : <i>ft_list_reverse_fun.c, ft_list.h</i>
	Funções autorizadas : Nenhuma

- 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

	Exercício : 16
	ft_sorted_list_insert
	Pasta de entrega : <i>ex16/</i>
	Ficheiros para entregar : <i>ft_sorted_list_insert.c, ft_list.h</i>
	Funções autorizadas : <i>ft_create_elem</i>

- 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

	Exercício : 17
	ft_sorted_list_merge
	Pasta de entrega : <i>ex17/</i>
	Ficheiros para entregar : <i>ft_sorted_list_merge.c, ft_list.h</i>
	Funções 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.