



# Piscina C

## C 12

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

# 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

# Capítulo I

## Preâmbulo

ALERTA DE SPOILER  
NÃO LEIA A PRÓXIMA PÁGINA

## Foi você que pediu.

- Em **Star Wars**, Darth Vader é o pai de Luke Skywalker.
- Em **Os Suspeitos**, Verbal é Keyser Soze.
- Em **Clube da Luta**, Tyler Durden e o narrador são a mesma pessoa.
- Em **O Sexto Sentido**, Bruce Willis está morto desde o início.
- Em **Os Outros**, os moradores da casa são os fantasmas e vice-versa.
- Em **Bambi**, a mãe de Bambi morre.
- Em **A Vila**, os monstros são os próprios habitantes e a ação se situa, na verdade, na nossa época.
- Em **Harry Potter**, Dumbledore morre.
- Em **O Planeta dos Macacos**, a ação se situa na Terra.
- Em **Game of Thrones**, Robb Stark e Joffrey Baratheon morrem na noite de seus respectivos casamentos.
- Em **Twilight**, os vampiros brilham no sol.
- Em **Stargate SG-1**, 1ª temporada, episódio 18, O'Neill e Carter estão na Antártida.
- Em **Batman - O Cavaleiro das Trevas Ressurge**, Miranda Tate é Talia Al'Gul.
- Em **Super Mario Bros**, a princesa está em outro castelo.

# Capítulo II

## Instruções

- Somente esta página servirá de referência, não confie nos boatos.
- Leia 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 estão 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 seu programa não compilar, 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 nas listas, vamos usar a seguinte estrutura:

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

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

# Capítulo III

## Exercício 00 : ft\_create\_elem


	Exercício : 00
	ft_create_elem
Pasta de entrega : ex00/	
Arquivos para entregar : ft_create_elem.c, ft_list.h	
Funções autorizadas : malloc	

- escreva a função `ft_create_elem` que cria um novo elemento de tipo `t_list`.
- Ela deverá atribuir `data` ao parâmetro fornecido e `next` a `NULL`.
- Ela deverá 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 : ex01/
	Arquivos para entregar : ft_list_push_front.c, ft_list.h
	Funções autorizadas : ft_create_elem


- escreva a função `ft_list_push_front` que acrescenta ao início da lista um novo elemento de tipo `t_list`.
- Ela deverá atribuir `data` ao parâmetro fornecido.
- Se necessário, ela vai atualizar o ponteiro para o início da lista.
- Ela deverá 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>
	Arquivos para entregar : <i>ft_list_size.c, ft_list.h</i>
	Funções autorizadas : Nenhuma

- escreva a função `ft_list_size` que retorna o número de elementos da lista.
- Ela deverá 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 : ex03/	
Arquivos para entregar : ft_list_last.c, ft_list.h	
Funções autorizadas : Nenhuma	

- escreva a função `ft_list_last` que retorna o último elemento da lista.
- Ela deverá 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/	
Arquivos para entregar : ft_list_push_back.c, ft_list.h	
Funções autorizadas : ft_create_elem	

- escreva a função `ft_list_push_back` que acrescenta no final da lista um novo elemento de tipo `t_list`.
- Ela deverá atribuir `data` ao parâmetro fornecido.
- Se necessário, ela vai atualizar o ponteiro para o início da lista.
- Ela deverá 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>	
Arquivos para entregar : <code>ft_list_push_strs.c</code> , <code>ft_list.h</code>	
Funções autorizadas : <code>ft_create_elem</code>	

- escreva a função `ft_list_push_strs` que cria uma nova lista, incluindo nela as cadeias de caracteres apontadas pelos elementos da matriz `strs`.
- `size` é o tamanho de `strs` O primeiro elemento da matriz estará no final da lista.
- O endereço do primeiro elemento da lista é retornado.
- Ela deverá 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/
	Arquivos para entregar : ft_list_clear.c, ft_list.h
	Funções autorizadas : free

- escreva a função `ft_list_clear` que remove e libera o conjunto dos elementos da lista.
- Cada `data` também deverá ser liberado com o auxílio de `free_fct`
- Ela deverá 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 : ex07/	
Arquivos para entregar : ft_list_at.c, ft_list.h	
Funções autorizadas : Nenhuma	

- escreva a função `ft_list_at` que retorna o n-ésimo elemento da lista, sabendo que o primeiro elemento é o elemento 0.
- Ela retornará um ponteiro nulo em caso de erro.
- Ela deverá 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>
	Arquivos para entregar : <code>ft_list_reverse.c</code>
	Funções autorizadas : Nenhuma

- escreva a função `ft_list_reverse` que inverte a ordem dos elementos da lista. Somente os conjuntos de ponteiros são permitidos.
- Atenção: neste exercício vamos usar nosso próprio `ft_list.h`
- Ela deverá 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 : ex09/
	Arquivos para entregar : ft_list_foreach.c, ft_list.h
	Funções autorizadas : Nenhuma

- escreva a função `ft_list_foreach` que aplica uma função dada como parâmetro ao valor contido em cada elemento da lista.
- `f` deve ser aplicada na ordem dos elementos da lista
- Ela deverá 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 : ex10/
	Arquivos para entregar : ft_list_foreach_if.c, ft_list.h
	Funções autorizadas : Nenhuma

- escreva a função `ft_list_foreach_if` que aplica uma função dada como parâmetro ao valor contido em determinados elementos da lista.
- `f` só será aplicada nos elementos que passados como argumento a `cmp` com `data_ref`, `cmp` retornem 0
- `f` deve ser aplicada na ordem dos elementos da lista
- Ela deverá 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 : ex11/
	Arquivos para entregar : ft_list_find.c, ft_list.h
	Funções autorizadas : Nenhuma

- escreva a função `ft_list_find` que retorna o endereço do primeiro elemento cujo dado comparado a `data_ref` com o auxílio de `cmp` faz com que `cmp` retorne 0.
- Ela deverá 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/
	Arquivos para entregar : ft_list_remove_if.c, ft_list.h
	Funções autorizadas : free

- escreva a função `ft_list_remove_if` que apaga da lista todos os elementos cujo dado 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 liberado com o auxílio de `free_fct`
- Ela deverá 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>
	Arquivos para entregar : <code>ft_list_merge.c</code> , <code>ft_list.h</code>
	Funções autorizadas : Nenhuma

- escreva 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.
- Ela deverá 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 : ex14/
	Arquivos para entregar : ft_list_sort.c, ft_list.h
	Funções autorizadas : Nenhuma

- escreva a função `ft_list_sort` que organiza em ordem crescente o conteúdo da lista, ao comparar dois elementos usando uma função de comparação de dados dos dois elementos.
- Ela deverá 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/	
Arquivos para entregar : ft_list_reverse_fun.c, ft_list.h	
Funções autorizadas : Nenhuma	

- escreva a função `ft_list_reverse_fun` que inverte a ordem dos elementos da lista.
- Ela deverá 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 : ex16/
	Arquivos para entregar : ft_sorted_list_insert.c, ft_list.h
	Funções autorizadas : ft_create_elem

- escreva 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.
- Ela deverá 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>
	Arquivos para entregar : <code>ft_sorted_list_merge.c</code> , <code>ft_list.h</code>
	Funções autorizadas : Nenhuma

- escreva 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.
- Ela deverá 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);
```