



# Piscina C

## C 12

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

*Versão: 9.0*

# Conteúdo

I	Preâmbulo	2
II	Instruções	4
III	Intruções IA	6
IV	Exercício 00 : ft_create_elem	9
V	Exercício 01 : ft_list_push_front	10
VI	Exercício 02 : ft_list_size	11
VII	Exercício 03 : ft_list_last	12
VIII	Exercício 04 : ft_list_push_back	13
IX	Exercício 05 : ft_list_push_strs	14
X	Exercício 06 : ft_list_clear	15
XI	Exercício 07 : ft_list_at	16
XII	Exercício 08 : ft_list_reverse	17
XIII	Exercício 09 : ft_list_foreach	18
XIV	Exercício 10 : ft_list_foreach_if	19
XV	Exercício 11 : ft_list_find	20
XVI	Exercício 12 : ft_list_remove_if	21
XVII	Exercício 13 : ft_list_merge	22
XVIII	Exercício 14 : ft_list_sort	23
XIX	Exercício 15 : ft_list_reverse_fun	24
XX	Exercício 16 : ft_sorted_list_merge	25
XXI	Exercício 17 : ft_sorted_list_merge	26
XXII	Submissão e avaliação	27

# Capítulo I

## Preâmbulo

ALERTA DE SPOILER  
NÃO LEIA A PRÓXIMA PÁGINA

## Foi avisado.

- In Star Wars, Dark Vader 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 The Sixth Sense, Bruce Willis has been 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 Antarctica.
- 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 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 textitflags `-Wall -Wextra -Werror`, e utiliza `cc`.
- Se o seu programa não compila, terá 0.
- 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, é necessário usar a estrutura seguinte:

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

- 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

## Intruições IA

### Contexto

A Piscina C é intensa. É o teu primeiro grande desafio na 42 — um mergulho profundo na resolução de problemas, autonomia e comunidade.

Nesta fase, o teu principal objetivo é obter uma base sólida — através do esforço, da repetição e, acima de tudo, da partilha de aprendizagens com os teus colegas.

Na era da IA, os atalhos são fáceis de encontrar. No entanto, é importante considerar se o uso da IA está realmente a ajudar-te a crescer — ou apenas a impedir-te de desenvolver competências reais.

A Piscine também é uma experiência humana — e, por agora, nada substitui isso. Nem mesmo a IA.

Para uma visão mais completa da nossa posição sobre a IA — como ferramenta de aprendizagem, como parte do currículo TIC e como uma expectativa crescente no mercado de trabalho — consulta o FAQ dedicado disponível no intranet.

### ● Mensagem principal

- ✎ Constrói bases sólidas sem atalhos.
- ✎ Desenvolve verdadeiramente competências técnicas e interpessoais.
- ✎ Vive a aprendizagem entre pares, começa a aprender a aprender e a resolver novos problemas.
- ✎ A jornada de aprendizagem é mais importante do que o resultado.
- ✎ Aprende os riscos associados à IA e desenvolve práticas de controlo eficazes e contramedidas para evitar os erros mais comuns.

## ● Regras para os alunos:

- Deves aplicar o raciocínio nas tarefas atribuídas, especialmente antes de recorrereres à IA.
- Não deves pedir respostas diretas à IA.
- Deves aprender sobre a abordagem global da 42 em relação à IA.

## ● Resultados esperados:

Nesta fase, vais ter os seguintes resultados:

- Obter bases sólidas em tecnologia e programação.
- Compreender por que razão e de que forma a IA pode ser perigosa durante esta fase.

## ● Comentários e exemplos:

- Sim, sabemos que a IA existe — e sim, pode resolver os teus projetos. Mas estás aqui para aprender, não para provar que a IA já aprendeu. Não percas tempo (nem o nosso) apenas para demonstrar que a IA consegue resolver o problema.
- Aprender na 42 não é sobre saber a resposta — é sobre desenvolver a capacidade de encontrar uma. A IA dá-te a resposta diretamente, mas isso impede-te de construir o teu próprio raciocínio. E o raciocínio exige tempo, esforço e envolve falhas. O caminho para o sucesso não deve ser fácil.
- Lembra-te que nos exames a IA não está disponível — sem internet, sem telemóveis, etc. Vais perceber rapidamente se dependeste demasiado da IA no teu processo de aprendizagem.
- A aprendizagem entre pares expõe-te a ideias e abordagens diferentes, melhorando as tuas competências interpessoais e a tua capacidade de pensar de forma divergente. Isso é muito mais valioso do que conversar com um bot. Por isso, não sejas tímido — fala, faz perguntas e aprende em conjunto!
- Sim, a IA fará parte do currículo — tanto como ferramenta de aprendizagem como tema de estudo. Terás até a oportunidade de construir o teu próprio software de IA. Para saberes mais sobre a nossa abordagem em crescendo, consulta a documentação disponível no intranet.



**✓ Boa prática:**


Estou com dificuldades num novo conceito. Pergunto a alguém ao meu lado como o abordou. Falamos durante 10 minutos — e de repente faz sentido. Percebo.

**✗ Má prática:**

Uso a IA em segredo, copio algum código que parece estar certo. Durante a avaliação por pares, não consigo explicar nada. Falho. Durante o exame — sem IA — fico novamente bloqueado. Falho.

# Capítulo IV

## 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 V

## Exercício 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 VI

## Exercício 02 : ft\_list\_size


	Exercício : 02
	ft_list_size
	Pasta de entrega : <i>ex02/</i>
	Ficheiros para entregar : <i>ft_list_size.c, 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 VII

## 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 VIII

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


	Exercício : 04
	ft_list_push_back
	Pasta de entrega : <i>ex04/</i>
	Ficheiros para entregar : <i>ft_list_push_back.c, ft_list.h</i>
	Funções autorizadas : <i>ft_create_elem</i>

- 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 IX

## 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 X

## 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 XI

## 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 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 XII

## 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 XIII

## 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 XIV

## 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 XV

## 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 XVI

## 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, ft_list.h</i>
	Funções autorizadas : <i>free</i>

- 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 XVII

## 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 XVIII

## 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 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 XIX

## 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 XX

## Exercício 16 : ft\_sorted\_list\_merge

	Exercício : 16
	ft_sorted_list_merge
	Pasta de entrega : <i>ex16/</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

## 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 XXII

## 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.