



C Piscine

C 012

*Summary:* このドキュメントは、C Piscine @ 42の C 12モジュール用の課題である。

# Contents

I	Foreword	2
II	Instructions	4
III	Exercice 00 : ft_create_elem	6
IV	Exercice 01 : ft_list_push_front	7
V	Exercice 02 : ft_list_size	8
VI	Exercice 03 : ft_list_last	9
VII	Exercice 04 : ft_list_push_back	10
VIII	Exercice 05 : ft_list_push_strs	11
IX	Exercice 06 : ft_list_clear	12
X	Exercice 07 : ft_list_at	13
XI	Exercice 08 : ft_list_reverse	14
XII	Exercice 09 : ft_list_foreach	15
XIII	Exercice 10 : ft_list_foreach_if	16
XIV	Exercice 11 : ft_list_find	17
XV	Exercice 12 : ft_list_remove_if	18
XVI	Exercice 13 : ft_list_merge	19
XVII	Exercice 14 : ft_list_sort	20
XVIII	Exercice 15 : ft_list_reverse_fun	21
XIX	Exercice 16 : ft_sorted_list_insert	22
XX	Exercice 17 : ft_sorted_list_merge	23
XXI	Submission and peer-evaluation	24

# **Chapter I**

## **Foreword**

SPOILER ALERT  
DON'T READ THE NEXT PAGE

## 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.

# Chapter II

## Instructions

- 課題に関する噂に惑わされないよう気をつけ、信用しないこと。
- この書類は、提出前に変更になる可能性があるため、気をつけること。
- ファイルとディレクトリへの権限があることを、あらかじめ確認すること。
- すべての課題は、提出手順に従い行うこと。
- 課題の確認と評価は、あなたの周りにいるPiscine受験者により行われる。
- 課題の確認と評価は、Piscine受験者に加えて、Moulinetteと呼ばれるプログラムによっても行われる。
- Moulinetteは、大変細かい評価を行う。これはすべて自動で行われるため、交渉の余地はない。
- Moulinetteは、コーディング規範（Norm）を遵守しないコードを解読することができない。そのため、Moulinetteはnorminetteと呼ばれるプログラムを使用し、あなたのファイルがコーディング規範を遵守しているか確認を行う。せっかくの取り組みが、norminetteの確認により無駄にならないよう、気をつけること。
- 問題は、簡単なものから徐々に難しくなるように並べられている。簡単な問題が解けていない場合は、難しい問題が解けていたとしても 加点されることはない。
- 使用が禁止されている関数を使用した場合は、不正とみなされる。不正者は-42の評価をつけられ、この評価に対する交渉の余地はない。
- 課題がプログラムの提出を要求する場合のみ、main()関数を提出すること。
- Moulinetteは以下のフラッグを用いて、ccでコンパイルする。 -Wall -Wextra -Werror
- プログラムがコンパイルされなかった場合、評価は0になる。
- 課題で指定されていないものは、どんなファイルもディレクトリ内に置かないこと。


- 質問がある場合は、隣の人に聞くこと。それでも分からない場合は、反対側の席の人に聞くこと。
- 助けてくれるのは、Google / 人間 / インターネット / ...と呼ばれているものたちである。
- 出力例には、問題文に明記されていない細部まで表示されている場合があるため、入念に確認すること。
- この課題では、以下の構造体を使用すること。

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

- 上記の構造体がft\_list.hファイルの中に含まれていることを確認し、各問題を提出すること。
- Exercise 01以降は、ft\_create\_elemを使用する。ft\_list.hファイルにプロトタイプを入れておくと、便利な場合がある。

# Chapter III

## Exercice 00 : ft\_create\_elem


	Exercice 00
	ft_create_elem
	提出するディレクトリ : <i>ex00/</i>
	提出するファイル : <i>ft_create_elem.c, ft_list.h</i>
	使用可能な関数 : <i>malloc</i>

- `t_list`型の新たな要素を作成する関数`ft_create_elem`を作成せよ。
- `data`に与えられた引数を受け渡し、`next`に`NULL`を代入すること。
- プロトタイプ例)

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

# Chapter IV

## Exercice 01 : ft\_list\_push\_front

	Exercise 01
	ft_list_push_front
	提出するディレクトリ : <i>ex01/</i>
	提出するファイル : <i>ft_list_push_front.c, ft_list.h</i>
	使用可能な関数 : <i>ft_create_elem</i>


- `t_list`型の新たな要素を、リストの先頭に追加する関数`ft_list_push_front`を作成せよ。
- `data`に与えられた引数を代入すること。
- 必要に応じて、リストの先頭にあるポインタを更新すること。
- プロトタイプ例)

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



# Chapter V

## Exercice 02 : ft\_list\_size


	Exercise 02
	ft_list_size
	提出するディレクトリ : <i>ex02/</i>
	提出するファイル : <i>ft_list_size.c</i> , <i>ft_list.h</i>
	使用可能な関数 : None

- リストにある要素の数を返す関数`ft_list_size`を作成せよ。
- プロトタイプ例)

```
int ft_list_size(t_list *begin_list);
```

# Chapter VI

## Exercice 03 : ft\_list\_last


	Exercise 03
	ft_list_last
	提出するディレクトリ : <i>ex03/</i>
	提出するファイル : <i>ft_list_last.c, ft_list.h</i>
	使用可能な関数 : None

- リストの最後にある要素を返す関数`ft_list_last`を作成せよ。
- プロトタイプ例)

```
t_list *ft_list_last(t_list *begin_list);
```

# Chapter VII

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


	Exercise 04
	ft_list_push_back
	提出するディレクトリ : <i>ex04/</i>
	提出するファイル : <i>ft_list_push_back.c, ft_list.h</i>
	使用可能な関数 : <i>ft_create_elem</i>

- t\_list型の新たな要素を、リストの最後に追加する関数ft\_list\_push\_backを作成せよ。
- dataに与えられた引数を代入すること。
- 必要に応じて、リストの先頭にあるポインタを更新すること。
- プロトタイプ例)

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

# Chapter VIII

## Exercice 05 : ft\_list\_push\_strs


	Exercise 05
	ft_list_push_strs
	提出するディレクトリ : <i>ex05/</i>
	提出するファイル : <i>ft_list_push_strs.c, ft_list.h</i>
	使用可能な関数 : <i>ft_create_elem</i>

- 配列strsの、すべての文字列が格納されているリストを作成する関数ft\_list\_push\_strsを作成せよ。
- sizeは、配列strsの要素の数である。
- 配列strsの最初の要素が、リストの最後の要素になること。
- リストの最初のリンクのアドレスが返されること。
- プロトタイプ例)

```
t_list *ft_list_push_strs(int size, char **strs);
```

# Chapter IX

## Exercice 06 : ft\_list\_clear


	Exercise 06
	ft_list_clear
	提出するディレクトリ : <i>ex06/</i>
	提出するファイル : <i>ft_list_clear.c, ft_list.h</i>
	使用可能な関数 : <i>free</i>

- リストにあるすべてのリンクを削除し、動的に確保されたメモリを解放する関数 *ft\_list\_clear* を作成せよ。
- *free\_fct* は、*data* のために動的に確保されたメモリを解放する。
- プロトタイプ例)

```
void ft_list_clear(t_list *begin_list, void (*free_fct)(void *));
```

# Chapter X

## Exercice 07 : ft\_list\_at


	Exercise 07
	ft_list_at
	提出するディレクトリ : <i>ex07/</i>
	提出するファイル : <i>ft_list_at.c, ft_list.h</i>
	使用可能な関数 : None

- リストのn番目にある要素を返す関数ft\_list\_atを作成せよ。リストの最初の要素は、0番目に配置されている。
- エラーがある場合は、NULLポインタを返すこと。
- プロトタイプ例)

```
t_list *ft_list_at(t_list *begin_list, unsigned int nbr);
```

# Chapter XI

## Exercice 08 : ft\_list\_reverse


	Exercise 08
ft_list_reverse	
提出するディレクトリ : <i>ex08/</i>	
提出するファイル : <i>ft_list_reverse.c</i>	
使用可能な関数 : None	

- リストにある要素の順序を反対にする関数`ft_list_reverse`を作成せよ。その際、各要素の値は変更しないこと。
- この関数は、我々の`ft_list.h`を使用してテストを行うため、注意すること。
- プロトタイプ例)

```
void ft_list_reverse(t_list **begin_list);
```

# Chapter XII

## Exercice 09 : ft\_list\_foreach

	Exercise 09
ft_list_foreach	
提出するディレクトリ : <i>ex09/</i>	
提出するファイル : <i>ft_list_foreach.c, ft_list.h</i>	
使用可能な関数 : None	

- 各リンクの要素に、引数として与えられた関数を適用する関数ft\_list\_foreachを作成せよ。
- 関数ポインタfは、リストと同じ順序で適用すること。
- プロトタイプ例)

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


- 関数ポインタfは、以下のように使用される。

```
(*f)(list_ptr->data);
```



# Chapter XIII

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

	Exercise 10
	ft_list_foreach_if
	提出するディレクトリ : <i>ex10/</i>
	提出するファイル : <i>ft_list_foreach_if.c, ft_list.h</i>
	使用可能な関数 : None

- 引数として渡された関数を、リストの複数の要素に適用する関数ft\_list\_foreach\_ifを作成せよ。
- data\_refと要素を比較し、cmpが0を返す場合のみ、関数ポインタfを適用すること。
- 関数ポインタfは、リストと同じ順序で適用すること。
- プロトタイプ例)

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

- 関数ポインタfと関数ポインタcmpは、以下のように使用される。


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



例えば、関数ポインタcmpはft\_strcmp関数になる。

# Chapter XIV

## Exercice 11 : ft\_list\_find

	Exercise 11
	ft_list_find
	提出するディレクトリ : <i>ex11/</i>
	提出するファイル : <i>ft_list_find.c, ft_list.h</i>
	使用可能な関数 : None

- 関数ポインタcmpを利用してdata\_refとリストの各要素を比較し、最初に差異が0になる要素のアドレスを返す関数ft\_list\_findを作成せよ。
- プロトタイプ例)


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

- 関数ポインタcmpは、以下のように使用される。

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

# Chapter XV

## Exercice 12 : ft\_list\_remove\_if

	Exercise 12
	ft_list_remove_if
	提出するディレクトリ : <i>ex12/</i>
	提出するファイル : <i>ft_list_remove_if.c, ft_list.h</i>
	使用可能な関数 : <i>free</i>

- 関数ポインタcmpを利用してdata\_refとリストの各要素を比較し、差異が0になる要素をリストから削除する関数ft\_list\_remove\_ifを作成せよ。
- free\_fctを使用し、要素のdataのメモリを解放すること。
- プロトタイプ例)


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

- 関数ポインタcmpと関数ポインタfree\_fctは、以下のように使用される。

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

# Chapter XVI

## Exercice 13 : ft\_list\_merge


	Exercise 13
	ft_list_merge
	提出するディレクトリ : <i>ex13/</i>
	提出するファイル : <i>ft_list_merge.c, ft_list.h</i>
	使用可能な関数 : None

- begin2リストにある要素を、begin1リストの最後に配置する関数ft\_list\_mergeを作成せよ。
- 要素を作成することはできない。
- プロトタイプ例)

```
void ft_list_merge(t_list **begin_list1, t_list *begin_list2);
```

# Chapter XVII

## Exercice 14 : ft\_list\_sort

	Exercise 14
	ft_list_sort
	提出するディレクトリ : ex14/
	提出するファイル : ft_list_sort.c, ft_list.h
	使用可能な関数 : None

- 2つの要素が所有しているデータを比較する関数を使用し、リストにある要素を、昇順に並べ替える関数ft\_list\_sortを作成せよ。
- プロトタイプ例)

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

- 関数ポインタcmpは、以下のように使用される。


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



例えば、関数ポインタcmpはft\_strcmp関数になる。

# Chapter XVIII

## Exercice 15 : ft\_list\_reverse\_fun


	Exercise 15
ft_list_reverse_fun	
提出するディレクトリ : <i>ex15/</i>	
提出するファイル : <i>ft_list_reverse_fun.c</i> , <i>ft_list.h</i>	
使用可能な関数 : None	

- リストにある要素の順序を逆にする関数`ft_list_reverse_fun`を作成せよ。
- プロトタイプ例)

```
void ft_list_reverse_fun(t_list *begin_list);
```

# Chapter XIX

## Exercice 16 : ft\_\_sorted\_\_list\_\_insert

	Exercise 16
ft_sorted_list_insert	
提出するディレクトリ : <i>ex16/</i>	
提出するファイル : <i>ft_sorted_list_insert.c, ft_list.h</i>	
使用可能な関数 : <i>ft_create_elem</i>	

- 新たな要素を作成し、昇順に並べられた順序を維持してリストに挿入する関数 `ft_sorted_list_insert` を作成せよ。
- プロトタイプ例)


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

- 関数ポインタ `cmp` は、以下のように使用される。

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

# Chapter XX

## Exercice 17 : ft\_sorted\_list\_merge

	Exercise 17
ft_sorted_list_merge	
提出するディレクトリ : <i>ex17/</i>	
提出するファイル : <i>ft_sorted_list_merge.c, ft_list.h</i>	
使用可能な関数 : None	

- 並べ替えられたリストbegin2の要素を、並べ替えられたリストbegin1と統合する関数ft\_sorted\_list\_mergeを作成せよ。その際、begin1は、昇順に並べられた順序を維持していること。
- プロトタイプ例)

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

- 関数ポインタcmpは、以下のように使用される。

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



# Chapter XXI

## Submission and peer-evaluation

課題は、いつも通り Git リポジトリに提出すること。リポジトリ内の提出物のみが、レビュー中の評価対象となる。ファイルの名前が正しいことを確認すること。



この課題の要件で求められているファイルのみを提出すること。