



С Пискин

День 11

Sta далее 42 pedago@42.fr

Аннотация: Данный документ является предметом Day11 на C Piscine @ 42.

содержание

I	инструкции	2
II	предисловие	4
III	Exercice 00: ft_create_elem	6
IV	Exercice 01: ft_list_push_back	7
V	Exercice 02: ft_list_push_front	8
VI	Exercice 03: ft_list_size	9
VII	Exercice 04: ft_list_last	10
VIII	Exercice 05: ft_list_push_params	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_phi	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

Инструкции

Глава I

- Только эта страница будет служить в качестве ссылки: не доверять слухам.
- Осторожно! Этот документ может потенциально изменить до часа до submit- Сионе.
- Убедитесь, что у вас есть соответствующие разрешения на вашем фи ле и каталогах.
- Вы должны следовать процедурам представления для каждого упражнения.
- Ваши упражнения будут проверены и рассортированы вашими одноклассниками.
- Кроме того, ваши упражнения будут проверены и классифицированы по программе под названием Moulinette.
- Moulinette очень дотошный и строгий в своей оценке вашей работы. Она полностью автоматизирована и не существует никакого способа вести с ним переговоры. Так что если вы хотите, чтобы избежать неприятных сюрпризов, быть тщательным, насколько это возможно.
- Moulinette не очень открытым. Он не будет пытаться понять ваш код, если он не уважает нормы. Moulinette опирается на программу под названием Norminator чтобы проверить, если ваш фи ле уважать нормы. TL; DR: это было бы идиотизмом представить часть работы, которая не проходит Norminator «S проверка.
- Эти упражнения тщательно выложены порядка ди FFI трудности - от простого к сложному. Мы не будет принимать во внимание успешно завершена сложнее упражнение, если проще один не вполне функциональны.
- Использование запрещенной функции считается мошенничеством. Мошенники получить - 42, и этот сорт не является предметом переговоров.
- Если ft_putchar () является авторизованным функция, мы будем компилировать код с нашей ft_putchar.c.
- Вы должны будете только представить основную функцию (), если мы просим программу.

- Moulinette компилирует с этим фл флажков: -Wall -Wextra -Werror и использования НКА.
- Если ваша программа не компилируется, вы получите 0.
- Вы не можете оставить никакого дополнительных фи ль в каталоге, чем то Специфическое едь в этой теме.
- Есть вопрос? Попросите ваш партнёр по правой руке. В противном случае, попытать пэрром на левой стороне.
- Ваш справочник называется Google / человек / Интернет /
- Проверьте «С» Piscine часть форума в интрасети.
- Изучите примеры тщательно. Они очень хорошо могли бы назвать подробности, которые явно не упомянутые в этой теме ...
- По Одина, Тор! Используй свой мозг !!!
- Для следующих упражнений, вы должны использовать следующую структуру:

```
бурейиЕ структуры      s_list
{ структура s_list
    * следующий;
    * данные;
    t_list;
}
```

- Вы должны включить эту структуру в ФИ ле ft_list.h и представить его для каждого упражнения.
- Из физических упражнений 01 и далее, мы будем использовать наш ft_create_elem, поэтому принимать меры (это может быть полезно иметь свой прототип в фи ле ft_list.h ...).

Глава II

Предисловие


SPOILER ALERT HE
ЧИТАЙТЕ NEXT PAGE

Вы были предупреждены.

- В Звездные войны, Dark Vador Отец Люка.
- В Обычные подозреваемые, Вербальные является Кайзер Soze.
- В Бойцовский клуб, Тайлер Дерден и рассказчик тот же человек.
- В Шестой Сенс, Брюс Уиллис мертв с самого начала.
- В Другие, жители дома привидение и наоборот.
- В Бэмби, мать Бэмби умирает.
- В Деревня, монстры жители и фильм на самом деле происходит в наше время.
- В Гарри Поттер, Дамблдор умирает.
- В Планета обезьян, фильм происходит на земле.
- В Игра престолов, Робб Старк и Джо Ф.Ф. Рей Баратеон умереть в день их свадьбы.
- В Сумерки, Вампиры светить под солнцем.
- В Stargate SG-1, сезон 1, эпизод 18, О'Нилл и Картер в Антарктиде.
- В Восстание темного рыцаря, Миранда Тейт Талия Алгул.
- В Super Mario Bros, Принцесса в другом замке.

Глава III

Exercice 00: ft_create_elem


	Exercice: 00
	ft_create_elem
Пошаговая в каталоге: <i>бывший</i> 00 / Файлы для включения в: ft_create_elem.c, ft_list.h	
Допустимые функции: таНос	
Примечания: н /	

- Создание функции `ft_create_elem` которая создает новый элемент `t_list` тип.
- Он должен назначить данные к данному аргументу и следующий в `NULL`.
- Вот как это должно быть прототип:

```
t_list *ft_create_elem ( недействительным " данные );
```

Глава IV

Exercice 01: ft_list_push_back


	Exercice: 01
	ft_list_push_back
Пошаговая в каталоге: <i>бывший</i> 01 / Файлы для включения в: ft_list_push_back.c, ft_list.h	
Допустимые функции: ft_create_elem	
Примечания: н /	

- Создание функции ft_list_push_back который добавляет новый элемент t_list введите в конце списка.
- Он должен назначить данные к данному аргументу.
- При необходимости, он будет обновлять указатель на начало списка.
- Вот как это должно быть прототип:

```
недействительным ft_list_push_back (t_list ** begin_list, недействительным * данные);
```


Глава V

Exercice 02: ft_list_push_front


	Exercice: 02
ft_list_push_front	
Пошаговая в каталоге: <i>бывший 02</i> / Файлы для включения в: ft_list_push_front.c,	
ft_list.h	
Допустимые функции: ft_create_elem	
Примечания: н /	

- Создание функции ft_list_push_front который добавляет новый элемент типа t_list к началу списка.
- Он должен назначить данные к данному аргументу.
- При необходимости, он будет обновлять указатель на начало списка.
- Вот как это должно быть прототип:

```
недействительным ft_list_push_front (t_list ** begin_list, недействительным * данные);
```

Глава VI

Exercise 03: ft_list_size


	Exercise: 03
ft_list_size	
Пошаговая в каталоге: <i>бывший 03</i> / Файлы для включения в: ft_list_size.c, ft_list.h	
Допустимые функции: Ничего	
Примечания: н /	

- Создание функции ft_list_size которая возвращает количество элементов в списке.
- Вот как это должно быть прототип:

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

Глава VII

Exercise 04: ft_list_last


	Exercise: 04
ft_list_last	
Пошаговая в каталоге: <i>бывший</i> 04 / Файлы для включения в: ft_list_last.c, ft_list.h	
Допустимые функции: Ничего	
Примечания: н /	

- Создание функции ft_list_last которая возвращает последний элемент списка.
- Вот как это должно быть прототип:

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

Глава VIII

Exercice 05: ft_list_push_params


	Exercice: 05
ft_list_push_params	
Пошаговая в каталоге: <i>бывший</i> 05 / Файлы для включения в: ft_list_push_params.c,	
ft_list.h	
Допустимые функции: ft_create_elem	
Примечания: н /	

- Создание функции ft_list_push_params который создает новый список, который включает в себя аргументы командной строки.
- Первый аргумент должен быть в конце списка.
- -адрес первого Линка фи в списке возвращается.
- Вот как это должно быть прототип:

```
t_list * ft_list_push_params ( int* переменный ток, голец то средний);
```

Глава IX

Exercise 06: ft_list_clear


	Exercise: 06
	ft_list_clear
	Пошаговая в каталоге: <i>бывший</i> 06 / Файлы для включения в: ft_list_clear.c,
	ft_list.h
	Допустимые функции: свободно
	Примечания: н /

- Создание функции ft_list_clear который удаляет все ссылки из списка.
- Затем он будет присвоить указатель в списке на нулевое значение.
- Вот как это должно быть прототип:

```
недействительным ft_list_clear (t_list ** begin_list);
```

Глава X

Exercise 07: ft_list_at


	Exercise: 07
ft_list_at	
Пошаговая в каталоге: <i>бывший 07</i> / Файлы для включения в: ft_list_at.c, ft_list.h	
Допустимые функции: Ничего	
Примечания: н /	

- Создание функции ft_list_at которая возвращает Nth элемент списка.
- В случае ошибки, она должна возвращать нулевой указатель.
- Вот как это должно быть прототип:

```
t_list * ft_list_at (t_list * begin_list, неподписанных INT NBR);
```

Глава XI

Exercise 08: ft_list_reverse


	Exercise: 08
ft_list_reverse	
Пошаговая в каталоге: <i>бывший 08</i> / Файлы для включения в: ft_list_reverse.c,	
ft_list.h	
Допустимые функции: Ничего	
Примечания: н /	

- Создание функции ft_list_reverse который меняет порядок элементов а список. Вы можете использовать только указатели, связанные ст и далее.
- Вот как это должно быть прототип:

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

Глава XII

Exercise 09: ft_list_foreach

	Exercise: 09
ft_list_foreach	
Пошаговая в каталоге: <i>бывший 09</i> / Файлы для включения в: ft_list_foreach.c, ft_list.h	
Допустимые функции: Ничего	
Примечания: н /	

- Создание функции ft_list_foreach который применяет функцию, заданную в качестве аргумента к информации внутри каждой из ссылок списка.
- Вот как это должно быть прототип:


```
наблюдательным ft_list_foreach (t_list * begin_list, наблюдательным (* e) (наблюдательным *));
```

- Функция указываемого e будет использоваться следующим образом:

```
(* e) (list_ptr -> данные);
```


Глава XIII

Exercise 10: ft_list_foreach_if

	Exercise: 10
ft_list_foreach_if	
Пошаговая в каталоге: <i>бывший 10</i> / Файлы для включения в: ft_list_foreach_if.c, ft_list.h	
Допустимые функции: Ничего	
Примечания: н /	

- Создание функции ft_list_foreach_if который применяет функцию, заданную в качестве аргумента к информации, содержащейся в некоторых звеньях списка. Ссылка информации, а также сравнительная функция должна позволить нам выбрать правильные ссылки из списка: те, которые «равны» в справочной информации.

- Вот как это должно быть прототип:

```
недействительным ft_list_foreach_if (t_list * begin_list, недействительным (* e) ( недействительным *), недействительным * data_ref, недействительным (* CMP) ()
```

- Функции указываемого e и CMP будет использоваться следующим образом:


```
(* e) (list_ptr -> данные); (* CMP) (list_ptr -> Данные,  
data_ref);
```



Например, функция CMP может быть ft_strcmp ...

Глава XIV

Exercise 11: ft_list_ фи-й


	Exercise: 11 ft_list_
	фи-й
Пошаговая в каталоге: <i>бывший</i> 11 /	Файлы для включения в: ft_list_find.c,
ft_list.h	
Допустимые функции: Ничего	
Примечания: н /	

- Создание функции ft_list_find который возвращает адрес ссылки первой, чьи данные «равно» к справочным данным.
- Вот как это должно быть прототип:

```
t_list * ft_list_find (t_list * begin_list, недействительным * data_ref, ИНТ ( " CMP") ());
```

Глава XV

Exercise 12: ft_list_remove_if


	Exercise: 12
ft_list_remove_if	
Пошаговая в каталоге: <i>бывший 12</i> / Файлы для включения в: ft_list_remove_if.c, ft_list.h	
Допустимые функции: свободно	
Примечания: н /	

- Создание функции ft_list_remove_if который стирает о далее список всех элементов, чьи данные «равны» справочным данным.
- Вот как это должно быть прототип:

```
недействительным ft_list_remove_if (t_list ** begin_list, недействительным * data_ref, int (* cmp) ());
```

Глава XVI

Exercice 13: ft_list_merge


	Exercice: 13
ft_list_merge	
Пошаговая в каталоге: <i>бывший</i> 13 / Файлы для включения в: ft_list_merge.c,	
ft_list.h	
Допустимые функции: Ничего	
Примечания: н /	

- Создание функции ft_list_merge который размещает элементы списка begin2 в конце списка в другой begin1.
- Создание элемента не разрешается.
- Вот как это должно быть прототип:

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

Глава XVII

Exercice 14: ft_list_sort

	Exercice: 14
ft_list_sort	
Пошаговая в каталоге: <i>бывший 14</i> / Файлы для включения в: ft_list_sort.c, ft_list.h	
Допустимые функции: Ничего	
Примечания: н /	

- Создание функции ft_list_sort который сортирует содержимое списка в порядке возрастания путем сравнения двух звеньев, благодаря функции, которая может сравнить данные, содержащиеся в этих двух ссылок.
- Вот как это должно быть прототип:


```
недействительным ft_list_sort (t_list ** begin_list, int (* CMP) ());
```



La fonction ЧМК pourrait être par Exemple ft_strcmp.

Глава XVIII

Exercise 15: ft_list_reverse_fun


	Exercise: 15
ft_list_reverse_fun	
Пошаговая в каталоге: <i>бывший 15</i> / Файлы для включения в: ft_list_reverse_fun.c, ft_list.h	
Допустимые функции: Ничего	
Примечания: н /	

- Создание функции ft_list_reverse_fun который меняет порядок элементов списка. Вы можете использовать только указатели, связанные ст и далее.
- Вот как это должно быть прототип:

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

Глава XIX

Exercise 16: ft_sorted_list_insert


	Exercise: 16
ft_sorted_list_insert	
Пошаговая в каталоге: <i>бывший</i> 16 / Файлы для включения в: ft_sorted_list_insert.c, ft_list.h	
Допустимые функции: ft_create_elem	
Примечания: н /	

- Создание функции ft_sorted_list_insert которая создает новый элемент и вставляет его в список, отсортированный так что остается sortend в порядке возрастания.
- Вот как это должно быть прототип:

```
надействительным ft_sorted_list_insert (t_list ** begin_list, надействительным * данные, int ( * CMP) ());
```

Глава XX

Exercise 17: ft_sorted_list_merge

	Exercise: 17
ft_sorted_list_merge	
Пошаговая в каталоге: <i>бывший 17</i> / Файлы для включения в: ft_sorted_list_merge.c, ft_list.h	
Допустимые функции: Ничего	
Примечания: н /	

- Создание функции ft_sorted_list_merge который объединяет в себе элементы отсортированного списка begin2 в другом отсортированном списке begin1, так что begin1 остается отсортирован в порядке возрастания.
- Вот как это должно быть прототип:

```
недействительным ft_sorted_list_merge (t_list ** begin_list1, t_list * begin_list2, инт ( * CMP) ());
```