



C Piscine

C 12

Summary: Bu döküman, 42'nin C Piscine'sinin C 12 modülünün dersidir.

Contents

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

Chapter I

Önsöz

İZLEME KEYFİNİZ KAÇABİLİR
BİR SONRAKİ SAYFAYI OKUMAYIN

Uyarıldınız.

- In Yıldız Savaşları filmlerinde Darth Vader, Luke'un babası.
- In Olağan Şüpheliler filminde Verbal, aslında Keyser Söze.
- In Dövüş Kulübü filminde Tyler Durden ve hikayeyi anlatan aslında aynı kişi.
- In Altıncı His filminde Bruce Willis filmin başından itibaren aslında ölü idi.
- In Diğerleri filminde evin yerleşik halkı hayaletler.
- In Bambi filminde Bambi'nin annesi ölüyor.
- In Köy filminde köylüler canavar ve film bizim zamanımızda geçiyor.
- In Harry Potter'da Dumbledore ölüyor.
- In Maymunlar Cehennemi filmi gezegenimizde geçiyor.
- In Taht Oyunları dizisinde Robb Stark ve Joffrey Baratheon düğün günlerinde ölüyorlar.
- In Alacakaranlık film serisinde vampirler güneş ışığı altında parlıyor.
- In Stargate SG-1 dizisinin ilk sezonunun 18. bölümünde, O'Neill ve Carter Antarktika'dalar.
- In Kara Şövalye Yükseliyor filminde, Miranda Tate aslında Talia Al'Ghul.
- In Super Mario Kardeşler'de prenses başka bir kalede.

Chapter II

Yönergeler

- Lütfen sadece bu sayfayı referans alınız: söylentilere kulak asmayınız.
- Dikkat! Dokümanın gönderim öncesinde değişme ihtimali vardır.
- Lütfen dosyalarınız ve dizileriniz için gerekli yetkilere sahip olduğunuzdan emin olunuz.
- Bütün çalışmalarınız için gönderim talimatlarını takip ediniz.
- Çalışmalarınız sınıf arkadaşlarınız tarafından kontrol edilip notlandırılacaktır.
- Aynı zamanda, çalışmalarınız Moulinette adlı program tarafından da kontrol edilip notlandırılacaktır.
- Moulinette değerlendirmelerinde çok titiz ve katıdır. Otomatik bir program olmasından dolayı görüş alışverişi mümkün değildir. Sürpriz bir sonuçla karşılaşmak için çalışmalarınızı dikkatlice yapınız.
- Moulinette çok açık görüşlü değildir. Kodunuz Norm'a uymadığı takdirde onu anlamaya çalışmayacaktır. Moulinette dosyalarınızın norm'a uyup uymadığını kontrol etmek için **norminette** adında bir program kullanmaktadır. TL;DR: **norminette**'in kontrolünden geçemeyecek bir dosya teslim etmek akılsızca olacaktır.
- Çalışmalar en kolaydan en zora olacak şekilde zorluklarına göre sıralanmıştır. Daha zor bir çalışma başarıyla tamamlanmış bile olsa daha kolay bir çalışmanın tamamıyla fonksiyonel olmaması durumunda dikkate **alınmayacaktır**.
- Yasaklanmış bir fonksiyon kullanmak hile olarak görülmektedir. Bunu yapan kişiler -42 puan alacaktır, ve bu not pazarlığa tabi değildir.
- Sizden program istersek sadece bir main() fonksiyonu göndermeniz gerekir.
- Moulinette çalışmaları şu şekilde sınıflandırır: -Wall -Wextra -Werror ve gcc
- Eğer programınız sınıflandırılmazsa, 0 alırsınız.
- Dizinizde konunun başlığındakiler dışında hiçbir dosya bırakmayınız.
- Bir sorunuz mu var? Sağınızdaki arkadaşınıza sorun. Olmadı solunuzdakine...


- Başvuru kılavuzunuzun adı `Google / man / the Internet / ...` 'dır.
- Intranetteki forumun "C Piscine" kısmını ya da Slack'deki Piscine bölümünü kontrol edin.
- Konu içerisinde net bir şekilde belirtilmemiş detayları anlayabilmek için örnekleri dikkatlice inceleyiniz.
- Odin ve Thor adına ! Kafayı çalıştırın !!!
- Bu dersin çalışmaları için aşağıdaki yapıyı kullanmanız gerekmektedir :

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

- Bu yapıyı bir `ft_list.h` dosyasına koyup, her çalışma için yollamanız gerekmektedir.
- Çalışma 01'den itibaren `ft_create_elem`'imizi kullanacağız, buna göre önleminizi alın. (Prototipini `ft_list.h` dosyasında bulundurmak işe yarayabilir...).

Chapter III

Çalışma 00 : ft_create_elem


	Exercise 00
	ft_create_elem
	Turn-in directory : <i>ex00/</i>
	Files to turn in : ft_create_elem.c , ft_list.h
	Allowed functions : malloc

- **t_list** tipinde yeni bir öge yaratacak bir **ft_create_elem** fonksiyonu oluşturun.
- It should assign to the given argument and **next** to NULL. Verilmiş olan değişkene **data** ve NULL'a da **next** atamalıdır.
- Prototipi şu şekilde olmalıdır :

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

Chapter IV

Çalışma 01 : ft_list_push_front


	Exercise 01
ft_list_push_front	
Turn-in directory : <i>ex01/</i>	
Files to turn in : <code>ft_list_push_front.c</code> , <code>ft_list.h</code>	
Allowed functions : <code>ft_create_elem</code>	

- Liste başına `t_list` tipinde yeni bir öge ekleyen `ft_list_push_front` oluşturun.
- Değişkene `data` atamalıdır.
- Eğer gerekli ise, liste başındaki işaretleyiciyi güncelleyecektir.
- Prototipi şu şekilde olmalıdır :

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


Chapter V

Çalışma 02 : ft_list_size


	Exercise 02
ft_list_size	
Turn-in directory : <i>ex02/</i>	
Files to turn in : ft_list_size.c , ft_list.h	
Allowed functions : None	

- Listedeki öğelerin sayısını geri bildirecek olan bir **ft_list_size** fonksiyonu oluşturun.
- Prototipi şu şekilde olmalıdır :

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

Chapter VI

Exercice 03 : ft_list_last


	Exercise 03
	ft_list_last
	Turn-in directory : <i>ex03/</i>
	Files to turn in : ft_list_last.c , ft_list.h
	Allowed functions : None

- Listedeki son öğeyi geri bildirecek olan bir **ft_list_last** fonksiyonu oluşturun.
- Prototipi şu şekilde olmalıdır :

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

Chapter VII

Çalışma 04 : ft_list_push_back


	Exercise 04
ft_list_push_back	
Turn-in directory : <i>ex04/</i>	
Files to turn in : <code>ft_list_push_back.c</code> , <code>ft_list.h</code>	
Allowed functions : <code>ft_create_elem</code>	

- Liste sonuna `t_list` tipinde bir öge ekleyecek olan bir `ft_list_push_back` fonksiyonu oluşturun.
- Verilmiş olan değişkene `data` atamalıdır.
- Eğer gerekli ise, liste başındaki işaretleyiciyi güncelleyecektir.
- Prototipi şu şekilde olmalıdır :

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

Chapter VIII

Çalışma 05 : ft_list_push_strs


	Exercise 05
ft_list_push_strs	
Turn-in directory : <i>ex05/</i>	
Files to turn in : <code>ft_list_push_strs.c</code> , <code>ft_list.h</code>	
Allowed functions : <code>ft_create_elem</code>	

- `strs` ögesinde yer alan öge tarafından gösterilen bütün diziyi içeren bir `ft_list_push_strs` fonksiyonu oluşturun.
- `size`, `strs`'nin boyutudur.
- The first element should be at the end of the list. İlk öge, listenin sonunda yer almalıdır.
- İlk bağlantının adresi listede geri döner.
- Prototipi şu şekilde olmalıdır :

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

Chapter IX

Çalışma 06 : ft_list_clear


	Exercise 06
ft_list_clear	
Turn-in directory : <i>ex06/</i>	
Files to turn in : ft_list_clear.c , ft_list.h	
Allowed functions : free	

- Listedeki bütün bağlantıları çıkartacak ve serbest bırakacak olan bir **ft_list_clear** fonksiyonu oluşturun.
- Her bir **data**'yı serbest bırakmak için **free_fct** kullanılır.
- Prototipi şu şekilde olmalıdır :

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

Chapter X

Çalışma 07 : ft_list_at


	Exercise 07
ft_list_at	
Turn-in directory : <i>ex07/</i>	
Files to turn in : <code>ft_list_at.c</code> , <code>ft_list.h</code>	
Allowed functions : None	

- `nbr` 0'a eşit olduğunda listenin ilk ögesi olduğunun bilindiği durumda, listenin N'inci ögesinin geri dönüşünü yapacak olan bir `ft_list_at` fonksiyonu oluşturun.
- Bir hata durumunda, geri dönüş boş bir işaretçi olmalıdır.
- Prototipi şu şekilde olmalıdır :

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

Chapter XI

Çalışma 08 : ft_list_reverse


	Exercise 08
ft_list_reverse	
Turn-in directory : <i>ex08/</i>	
Files to turn in : ft_list_reverse.c	
Allowed functions : None	

- Listedeki öğelerin sırasını tersine çevirecek bir **ft_list_reverse** fonksiyonu oluşturun. Bütün öğelerin değeri aynı kalmalıdır.
- Fonksiyon bizim kendi **ft_list.h**'mizi kullanacaktır. Dikkat edin.
- Prototipi şu şekilde olmalıdır :

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

Chapter XII

Çalışma 09 : ft_list_foreach

	Exercise 09
ft_list_foreach	
Turn-in directory : <i>ex09/</i>	
Files to turn in : <code>ft_list_foreach.c</code> , <code>ft_list.h</code>	
Allowed functions : None	

- Listenin her ögesine değişken olarak verilen fonksiyonu uygulayan bir `ft_list_foreach` fonksiyonu oluşturun.
- `f` listedeki aynı sırası ile uygulanmalıdır.
- Prototipi şu şekilde olmalıdır :


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

- `f` tarafından gösterilen fonksiyon şu şekilde kullanılacaktır :

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


Chapter XIII

Çalışma 10 : ft_list_foreach_if

	Exercise 10
ft_list_foreach_if	
Turn-in directory : <i>ex10/</i>	
Files to turn in : <i>ft_list_foreach_if.c</i> , <i>ft_list.h</i>	
Allowed functions : None	

- Listenin bazı öğelerine değişken olarak verilen fonksiyonu uygulayan bir `ft_list_foreach_if` fonksiyon oluşturun.
- Fonksiyonu sadece `cmp`'nin `data_ref` ile beraber olduğu öğelere uygulayın, `cmp`'nin geri dönüşü 0 olacaktır.
- `f` listedeki aynı sırası ile uygulanmalıdır.
- Prototipi şu şekilde olmalıdır :

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

- `f` ve `cmp` tarafından gösterilen fonksiyonlar şu şekilde kullanılacaktır:


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



Örneğin, `cmp` fonksiyonu, `ft_strcmp` olabilir...

Chapter XIV

Çalışma 11 : ft_list_find

	Exercise 11
	ft_list_find
	Turn-in directory : <i>ex11/</i>
	Files to turn in : <code>ft_list_find.c</code> , <code>ft_list.h</code>
	Allowed functions : None

- `cmp` ile `data_ref`'e kıyasla ilk öğenin verisinin adresinin 0'a geri dönüşünü sağlayacak olan bir `ft_list_find` fonksiyonu oluşturun.
- Prototipi şu şekilde olmalıdır :


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

- `cmp` ile gösterilen fonksiyon şu şekilde kullanılacaktır :

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

Chapter XV

Çalışma 12 : ft_list_remove_if

	Exercise 12
ft_list_remove_if	
Turn-in directory : <i>ex12/</i>	
Files to turn in : ft_list_remove_if.c , ft_list.h	
Allowed functions : free	

- **cmp** kullanan ve **data_ref** ile karşılaştırıldığında geri dönüşü 0 olan bütün verileri listeden çıkartan bir **ft_list_remove_if** fonksiyonu oluşturun.
- Silinecek olan bir öğeden veri **free_fct** kullanarak serbest bırakılmalıdır.
- Prototipi şu şekilde olmalıdır :


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

- **cmp** ve **free_fct** tarafından gösterilen fonksiyon şu şekilde kullanılacaktır :

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

Chapter XVI

Çalışma 13 : ft_list_merge


	Exercise 13
ft_list_merge	
Turn-in directory : <i>ex13/</i>	
Files to turn in : <code>ft_list_merge.c</code> , <code>ft_list.h</code>	
Allowed functions : None	

- `begin2` listesinin öğelerini, `begin1` listesinin sonuna koyacak bir `ft_list_merge` fonksiyonu oluşturun.
- Öğe oluşturmaya izin yoktur.
- Prototipi şu şekilde olmalıdır :

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

Chapter XVII

Çalışma 14 : ft_list_sort

	Exercise 14
	ft_list_sort
	Turn-in directory : <i>ex14/</i>
	Files to turn in : ft_list_sort.c , ft_list.h
	Allowed functions : None

- Öğelerin verisini bir fonksiyon kullanıp karşılaştırarak, listenin öğelerini küçükten büyüğe doğru dizen bir **ft_list_sort** fonksiyonu oluşturun.
- Prototipi şu şekilde olmalıdır :

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

- **cmp** tarafından gösterilen fonksiyon şu şekilde kullanılacaktır :


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



Örneğin, **cmp**, **ft_strcmp** olabilir.

Chapter XVIII

Çalışma 15 : ft_list_reverse_fun


	Exercise 15
ft_list_reverse_fun	
Turn-in directory : <i>ex15/</i>	
Files to turn in : <code>ft_list_reverse_fun.c</code> , <code>ft_list.h</code>	
Allowed functions : None	

- Listedeki öğelerin sırasını tersine çeviren bir `ft_list_reverse_fun` fonksiyonu oluşturun.
- Prototipi şu şekilde olabilir :

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

Chapter XIX

Çalışma 16 : ft_sorted_list_insert

	Exercise 16
ft_sorted_list_insert	
Turn-in directory : ex16/	
Files to turn in : ft_sorted_list_insert.c, ft_list.h	
Allowed functions : ft_create_elem	

- Yeni bir öge oluşturup, bu öğeyi listeye küçükten büyüğe sıralamasını bozmayacak şekilde yerleştiren bir `ft_sorted_list_insert` fonksiyonu oluşturun.
- Prototipi şu şekilde olmalıdır :


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

- `cmp` tarafından gösterilen fonksiyon şu şekilde kullanılacaktır :

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

Chapter XX

Çalışma 17 : ft_sorted_list_merge

	Exercise 17
ft_sorted_list_merge	
Turn-in directory : <i>ex17/</i>	
Files to turn in : <code>ft_sorted_list_merge.c</code> , <code>ft_list.h</code>	
Allowed functions : None	

- `begin1` listesinin küçükten büyüğe sıralamasını koruyarak `begin2` listesinin öğelerini ona entegre edecek olan bir `ft_sorted_list_merge` fonksiyonu oluşturun.
- Prototipi şu şekilde olmalıdır :

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

- `cmp` tarafından gösterilen fonksiyon şu şekilde kullanılacaktır :

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