



# Piscina C

Ziua 10

Staff Academy+Plus [contact@academyplus.ro](mailto:contact@academyplus.ro)

*Sumar: Acest document este subiectul zilei 10 a piscinei C din cadrul Academy+Plus.*

# Cuprins

I	Instructiuni	2
II	Preambul	4
III	Exercitiu 00 : Makefile	5
IV	Exercitiu 01 : ft_foreach	6
V	Exercitiu 02 : ft_map	7
VI	Exercitiu 03 : ft_any	8
VII	Exercitiu 04 : ft_count_if	9
VIII	Exercitiu 05 : ft_is_sort	10
IX	Exercitiu 06 : do-op	11
X	Exercitiu 07 : ft_sort_wordtab	13
XI	Exercitiu 08 : ft_advanced_sort_wordtab	14
XII	Exercitiu 09 : ft_advanced_do-op	15

# Capitolul I

## Instructiuni

- Utilizati doar aceste pagini ca referinta; nu plecati urechea la zgomotul de pe coridor.
- Subiectul se poate schimba cu cel mult o ora inainte de incepere.
- Fiti atenti la drepturile pe care le aveti asupra fisierelor si directoarelor.
- Trebuie sa urmati procedurile de parcurgere pentru toate exercitiile voastre.
- Exerciitiile voastre vor fi corectate de colegii vostri de piscina.
- Pe langa colegii vostri, veti fi corectati de un program numit Moulinette.
- Aplicatia Moulinette este foarte stricta in notare. Ea este total automatizata. Este imposibil sa comentati in legatura cu nota primita. Fiti foarte rigurosi pentru a evita surprizele.
- Moulinette nu e foarte desteapta. Ea nu poate intelege codul care nu respecta Standardele de scriere a codului (Norme).
- Utilizarea unei functii interzise este un caz de inselaciune (trisare). Toate aceste cazuri sunt sanctionate cu nota -42.
- Daca `ft_putchar()` este o functie valida, veti compila fisierul `ft_putchar.c`.
- Nu trebuie sa creati o functie `main()` decat atunci cand vi se cere sa scrieti un program.
- Exerciitiile sunt strict ordonate de la cele simple spre cele complexe. In nici un caz nu vom lua in considerare un exercitiu complex rezolvat daca unul anterior, mai simplu, nu a fost rezolvat perfect.
- Aplicatia Moulinette se compileaza cu flag-urile: `-Wall -Wextra -Werror`.
- Daca programul vostru nu se compileaza, veti primi nota 0.

- Nu lasati in directorul de lucru niciun fisier, altul decat cele specificate de enuntul exercitiului.
- Aveti intrebari? Intrebat-l pe vecinul din dreapta. Daca nu, incercati la cel din stanga.
- Manualele voastre de referinta sunt Google / man / Internet / ....
- Puteti folosi forumul de pe Intranet pentru discutii legate de Piscina!
- Cititi cu atentie exemplele. Va pot oferi informatii suplimentare pentru elementele neclare din enunt...
- Reflectati la asta. Aveti mare grija!

# Capitolul II

## Preambul

Citat din filmul V de la Vendetta:

Vezi! Vezi în mine imaginea unui umil Veteran de VodeVil, distribuit în mod Vicios în roluri de Victimă și de Venetic de Vicisitudinile Vieții. Această Vedere, mai mult decât un Vulgar Văruit cu Vanitate, este un Vestigiu al Vox populi astăzi Vacantă, eVaporată. Totuși, această Vitează Vizită a unei Vexări trecute se reVede reînViată și a jurat să înVingă această Venală și Virulenă Vermină care preamărește Viciul și Varsă în Vicioasa Violentă și Violarea Vorace a Volițiunii. Un singur Verdict: Vendeta. O Vendetă ca o ofrandă Votivă, dar nu în Van, căci Valoarea și Voracitatea sa Vor Veni într-o zi să-l Valorifice pe cel Vigilent și Virtuos. In mod Veridic, această Volbură de Vorbărie Virează Veridic spre PălăVrageală, așa că lasă-mă pur și simplu să adaug că este o Veritabilă onoare să te întâlnesc.


Spune-mi V.



Avoid Aliterations. Always.

# Capitolul III

## Exercitiu 00 : Makefile

	Exercitiu: 00
Makefile	
Director de lucru: <code>ex00/</code>	
Fisier(e) de iesire: <code>Makefile</code>	
Functii autorizate: <code>Niciuna</code>	
Observatii: <code>n/a</code>	


- Scrieti `Makefile` care va compila `libft.a`.
- `Makefile` va cauta fisierele sursa din directorul `srcs`.
- `Makefile` va cauta fisierele header in directorul `includes`.
- Biblioteca va fi plasata in radacina exercitiului.
- `Makefile` va trebui de asemenea sa implementeze regulile `clean`, `fclean` si `re` in plus fata de regula `all`.
- Regula `fclean` este echivalentul unui `make clean` si stege si binarul creat in urma lui `make`. Regula `re` este echivalentul unei comezi `make fclean` apoi un `make`
- Nu vom considera decat `Makefile`-ul voastru si pe care il vom testa cu fisierele noastre. In cadrul acestui exercitiu, nu lucrati decat cu cele 5 functii obligatorii pentru biblioteca voastra (`ft_putchar`, `ft_putstr`, `ft_strcmp`, `ft_strlen` si `ft_swap`).



Atentie la wildcard-uri!

# Capitolul IV

## Exercitiu 01 : ft\_foreach

	Exercitiu: 01
	ft_foreach
Director de lucru: ex01/	
Fisier(e) de iesire: ft_foreach.c	
Functii autorizate: Niciuna	
Observatii: n/a	

- Scrieti o functie `ft_foreach` care, pentru un tablou de intregi dat, va aplica o functie asupra tuturor elementelor acestui tablou. Aceasta functie se va aplica in ordinea elementelor tabloului.
- Functia va avea prototipul urmator:


```
void ft_foreach(int *tab, int length, void(*f)(int));
```

- De exemplu, functia `ft_foreach` va putea fi apelata in felul urmator pentru a afisa toti intregii din tablou:

```
ft_foreach(tab, 1337, &ft_putnbr);
```

# Capitolul V

## Exercitiu 02 : ft\_map

	Exercitiu: 02
ft_map	
Director de lucru: <i>ex02/</i>	
Fisier(e) de iesire: <b>ft_map.c</b>	
Functii autorizate: <b>malloc</b>	
Observatii: <b>n/a</b>	


- Scrieti o functie **ft\_map** care, pentru un tablou de intregi dat, va aplica o functie asupra tuturor elementelor tabloului (in ordine) si va returna un tablou cu toate valorile de retur. Aceasta functie va fi aplicata in ordinea elementelor tabloului.
- Functia va avea prototipul urmatoare:

```
int      *ft_map(int *tab, int length, int(*f)(int));
```



# Capitolul VI

## Exercitiu 03 : ft\_any

	Exercitiu: 03
ft_any	
Director de lucru: ex03/	
Fisier(e) de iesire: ft_any.c	
Functii autorizate: Niciuna	
Observatii: n/a	


- Scrieti o functie `ft_any` care va returna 1 daca functia `f` transmisa ca parametru, aplicata pe cel putin un element din tablou returneaza 1. Functia `ft_any` va returna 0 in caz contrar (prin aplicarea functiei `f` asupra tuturor elementelor din tablou obtinem de fiecare data 0).
- Functia va avea prototipul urmator:

```
int ft_any(char **tab, int(*f)(char*));
```

- Tabloul se va termina cu 0.

# Capitolul VII

## Exercitiu 04 : ft\_count\_if

	Exercitiu: 04
	ft_count_if
Director de lucru: ex04/	
Fisier(e) de iesire: ft_count_if.c	
Functii autorizate: Niciuna	
Observatii: n/a	


- Scrieti o functie `ft_count_if` care va returna numarul de elemente ale tabloului care, dand functia `f` ca parametru, returneaza 1.
- Functia va avea prototipul urmator:

```
int ft_count_if(char **tab, int(*f)(char*));
```

- Tabloul se va termina cu elementul 0.

# Capitolul VIII

## Exercitiu 05 : ft\_is\_sort


	Exercitiu: 05
	ft_is_sort
Director de lucru: ex05/	
Fisier(e) de iesire: ft_is_sort.c	
Functii autorizate: Niciuna	
Observatii: n/a	

- Scrieti o functie `ft_is_sort` care returneaza 1 daca tabloul e sortat si 0 in caz contrar.
- Functia transmisa ca parametru va returna un intreg negativ daca primul argument este inferior celui de-al doilea, 0 daca sunt egali si un intreg pozitiv altfel.
- Functia va avea prototipul urmator:

```
int ft_is_sort(int *tab, int length, int(*f)(int, int));
```

# Capitolul IX

## Exercitiu 06 : do-op

	Exercitiu: 06
do-op	
Director de lucru: <code>ex06/</code>	
Fisier(e) de iesire: <code>Makefile</code> , si fisierele programului vostru	
Functii autorizate: <code>write</code>	
Observatii: <code>n/a</code>	

- Scrieti un program care se numeste `do-op`.
- Programul trebuie sa fie lansat cu trei argumente: `do-op valoarea1 operator valoarea2`
- Exemplu:

```
$>./do-op 42 "+" 21
63
$>
```


- Caracterul `operator` va mapa functia corespunzatoare dintr-un tablou de pointeri la functii.
- Directorul vostru va contine un `Makefile` cu o regula `all` si o regula `clean`.
- In cazul unei expresii false ca `./do-op foo slash bar`, programul afiseaza 0.
- Daca numarul de argumente nu este corect, `do-op` nu afiseaza nimic.

- Vedeti un exemplu de teste ale Moulinette-ei:

```
$> make clean
$> make
$> ./do-op
$> ./do-op 1 + 1
2
$> ./do-op 42amis - -20toto12
62
$> ./do-op 1 p 1
0
$> ./do-op 1 + toto3
1
$>
$> ./do-op toto3 + 4
4
$> ./do-op foo plus bar
0
$> ./do-op 25 / 0
Stop : division by zero
$> ./do-op 25 % 0
Stop : modulo by zero
$>
```

# Capitolul X

## Exercitiu 07 : ft\_sort\_wordtab


	Exercitiu: 07
ft_sort_wordtab	
Director de lucru: ex07/	
Fisier(e) de iesire: ft_sort_wordtab.c	
Functii autorizate: Niciuna	
Observatii: n/a	

- Scrieti functia `ft_sort_wordtab` care sorteaza in ordine `ascii` cuvintele obtinute cu ajutorul `ft_split_whitespace`.
- Ordonarea se va face schimband valoarea pointerilor din tablou.
- Ea va avea prototipul urmator:

```
void ft_sort_wordtab(char **tab);
```

# Capitolul XI

## Exercitiu 08 : ft\_advanced\_sort\_wordtab

	Exercitiu: 08
ft_advanced_sort_wordtab	
Director de lucru: ex08/	
Fisier(e) de iesire: ft_advanced_sort_wordtab.c	
Functii autorizate: Niciuna	
Observatii: n/a	

- Scrieti functia `ft_advanced_sort_wordtab` care ordoneaza, in functie de valoarea returnata de functia transmisa ca parametru, cuvintele obtinute cu ajutorul `ft_split_whitespace`.
- Ordonarea se va face schimband pointer-ii tabelului.
- Ea va avea prototipul urmator:


```
void ft_advanced_sort_wordtab(char **tab, int(*cmp)(char *, char *));
```



Un apel `ft_advanced_sort_wordtab()` cu al doilea parametru `ft_strcmp` va da avelasi rezultat ca `ft_sort_wordtab()`.

# Capitolul XII

## Exercitiu 09 : ft\_advanced\_do-op

	Exercitiu: 09
ft_advanced_do-op	
Director de lucru: ex09/	
Fisier(e) de iesire: Makefile, si fisierele programului vostru	
Functii autorizate: write	
Observatii: n/a	

- Scrieti un program care sa functioneze exact ca do-op in cele mai mici detalii: trebuie sa includeti fisierul `ft_opp.h` ce va defini care pointer la functie corespunde fiecarui caracter.

```
$>cat ft_opp.h
#ifndef __FT_OPP_H__
# define __FT_OPP_H__

t_opp gl_opptab[] =
{
    {"-", &ft_sub},
    {"+", &ft_add},
    {"*", &ft_mul},
    {"/", &ft_div},
    {"%", &ft_mod},
    {"", &ft_usage}
};

#endif /* __FT_OPP_H__ */
$>
```

- Trebuie sa creati cel putin 6 functii: `ft_add`, `ft_sub`, `ft_mul`, `ft_div`, `ft_mod`, `ft_usage`.



- `ft_usage` afiseaza caracterele posibile (definite in `ft_opp.h`) ca in exemplul de mai jos:

```
$> make clean
$> make
$> ./ft_advanced_do-op
$> ./ft_advanced_do-op 1 + 1
2
$> ./ft_advanced_do-op 1 p 1
error : only [ - + * / % ] are accepted.
$> ./ft_advanced_do-op 1 + toto3
1
$> ./ft_advanced_do-op 25 / 0
Stop : division by zero
$> ./ft_advanced_do-op 25 % 0
Stop : modulo by zero
$>
```

- Trebuie sa definiti tipul `t_opp` corespunzator structurii `s_opp` permitand compilarea proiectului vostru.
- Nu scrieti NIMIC in fisierul `ft_opp.h`, nici chiar definitia lui `t_opp`. In schimb, nu uitati sa adaugati header-ul, pentru ca fisierul sa corespunda normelor. Includeti de asemenea propriile voastre fisiere daca e necesar.
- Nu afisati o eroare decat pentru operatorii ce nu au corespondent in `ft_opp.h`.
- Ganditi-va la toate punctele precedente, pentru ca vom schimba sigur fisierul `ft_opp.h`...



Un operator poate fi compus din mai multe caractere.