



# Piscina C

Ziua 13

Staff 42 [piscina@academyplus.ro](mailto:piscina@academyplus.ro)

*Sumar: Acest document este subiectul zilei 13 a piscinei C din cadrul Academy+Plus.*

# Cuprins

I	Instructiuni	2
II	Preambul	4
III	Exercitiu 00 : btree_create_node	5
IV	Exercitiu 01 : btree_apply_prefix	6
V	Exercitiu 02 : btree_apply_infix	7
VI	Exercitiu 03 : btree_apply_suffix	8
VII	Exercitiu 04 : btree_insert_data	9
VIII	Exercitiu 05 : btree_search_item	10
IX	Exercitiu 06 : btree_level_count	11
X	Exercitiu 07 : btree_apply_by_level	12
XI	Instructiuni intermediare	13
XII	Exercitiu 08: rb_insert	14
XIII	Exercitiu 09: rb_remove	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 la 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!
- Pentru exercitiile de azi vom utiliza structura urmatoare:

```
typedef struct      s_btree
{
    struct s_btree  *left;
    struct s_btree  *right;
    void            *item;
}                   t_btree;
```

- Va trebui sa puneti aceasta structura intr-un fisier `ft_btree.h` si sa-l utilizati la fiecare exercitiu.
- Incepand cu exercitiul 01 vom utiliza `btree_create_node`; luati masurile necesare (ar putea fi interesant sa fie inclus un prototip in `ft_btree.h...`).

# Capitolul II

## Preambul


Vedeti mai jos lista realizarilor lui Venom :

- In League with Satan (single, 1980)
- Welcome to Hell (1981)
- Black Metal (1982)
- Bloodlust (single, 1983)
- Die Hard (single, 1983)
- Warhead (single, 1984)
- At War with Satan (1984)
- Hell at Hammersmith (EP, 1985)
- American Assault (EP, 1985)
- Canadian Assault (EP, 1985)
- French Assault (EP, 1985)
- Japanese Assault (EP, 1985)
- Scandinavian Assault (EP, 1985)
- Manitou (single, 1985)
- Nightmare (single, 1985)
- Possessed (1985)
- German Assault (EP, 1987)
- Calm Before the Storm (1987)
- Prime Evil (1989)
- Tear Your Soul Apart (EP, 1990)
- Temples of Ice (1991)
- The Waste Lands (1992)
- Venom '96 (EP, 1996)
- Cast in Stone (1997)
- Resurrection (2000)
- Anti Christ (single, 2006)
- Metal Black (2006)
- Hell (2008)
- Fallen Angels (2011)

Subiectul de azi e mai usor daca lucrati ascultand Venom.

# Capitolul III

## Exercitiu 00 : btree\_create\_node


	Exercitiu: 00
btree_create_node	
Director de lucru: <i>ex00/</i>	
Fisier(e) de iesire: <b>btree_create_node.c</b> , <b>ft_btree.h</b>	
Functii autorizate: <b>malloc</b>	
Observatii: <b>n/a</b>	

- Scrieti functia **btree\_create\_node** care alocă un element nou, initializează **item**-ul la valoarea parametrului și toate celelalte elemente la 0.
- Adresa nodului creat este returnată.
- Ea trebuie să aibă prototipul următor:

```
t_btree *btree_create_node(void *item);
```

# Capitolul IV

## Exercitiu 01 : btree\_apply\_prefix


	Exercitiu: 01
btree_apply_prefix	
Director de lucru: <i>ex01/</i>	
Fisier(e) de iesire: <i>btree_apply_prefix.c, ft_btree.h</i>	
Funcții autorizate: Niciuna	
Observatii: n/a	

- Scrieti functia `btree_apply_prefix` care aplica functia transmisa in parametru `item`-ului fiecarui nod, parcurgand arborele in modul `prefix`.
- Ea trebuie sa aiba prototipul urmator:

```
void btree_apply_prefix(t_btree *root, void (*applyf)(void *));
```

# Capitolul V

## Exercitiu 02 : btree\_apply\_infix

	Exercitiu: 02
btree_apply_infix	
Director de lucru: ex02/	
Fisier(e) de iesire: btree_apply_infix.c, ft_btree.h	
Functii autorizate: Niciuna	
Observatii: n/a	


- Scrieti functia `btree_apply_infix` care aplica functia transmisa ca parametru la `item`-ul fiecarui nod, parcurgand arborele in maniera `infix`.
- Ea trebuie sa aiba prototipul urmator:

```
void btree_apply_infix(t_btree *root, void (*applyf)(void *));
```



# Capitolul VI

## Exercitiu 03 : btree\_apply\_suffix


	Exercitiu: 03
btree_apply_suffix	
Director de lucru: ex03/	
Fisier(e) de iesire: btree_apply_suffix.c, ft_btree.h	
Funcții autorizate: Niciuna	
Observatii: n/a	

- Scrieti functia `btree_apply_suffix` care aplica functia transmisa ca parametru la `item`-ul fiecarui nod, parcurgand arborele in modul `suffix`.
- Ea trebuie sa aiba prototipul urmator:

```
void btree_apply_suffix(t_btree *root, void (*applyf)(void *));
```

# Capitolul VII

## Exercitiu 04 : btree\_insert\_data


	Exercitiu: 04
btree_insert_data	
Director de lucru: ex04/	
Fisier(e) de iesire: btree_insert_data.c, ft_btree.h	
Functii autorizate: btree_create_node	
Observatii: n/a	

- Scrieti functia `btree_insert_data` care insereaza elementul `item` intr-un arbore. Arborele transmis ca parametru va fi ordonat in felul urmator: pentru fiecare `nod` toate elementele inferioare se vor plasa in partea stanga si toate elementele superioare sau egale la dreapta. Se transmite ca parametru o functie de comparare avand acelasi comportament cu `strcmp`.
- Parametrul `root` point-eaza pe nodul radacina al arborelui. Dupa primul apel, point-eaza spre NULL.
- Ea va trebui sa aiba prototipul urmator:

```
void btree_insert_data(t_btree **root, void *item, int (*cmpf)(void *, void *));
```

# Capitolul VIII

## Exercitiu 05 : btree\_search\_item


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

- Scrieti functia `btree_search_item` care returneaza primul element corespunzator valorii de referinta transmisa ca parametru. Arborele va trebui sa fie parcurs in modul `infix`. Daca elementul nu este gasit, functia va trebui sa returneze `NULL`.
- Ea trebuie sa aiba prototipul urmator:

```
void *btree_search_item(t_btree *root, void *data_ref, int (*cmpf)(void *, void *));
```

# Capitolul IX

## Exercitiu 06 : btree\_level\_count


	Exercitiu: 06
btree_level_count	
Director de lucru: ex06/	
Fisier(e) de iesire: btree_level_count.c, ft_btree.h	
Functii autorizate: Niciuna	
Observatii: n/a	

- Scrieti o functie `btree_level_count` care returneaza dimensiunea celei mai lungi ramuri din arborele transmis ca parametru.
- Ea trebuie sa aiba prototipul urmator:

```
int btree_level_count(t_btree *root);
```

# Capitolul X

## Exercitiu 07 : btree\_apply\_by\_level

	Exercitiu: 07
btree_apply_by_level	
Director de lucru: ex07/	
Fisier(e) de iesire: btree_apply_by_level.c, ft_btree.h	
Functii autorizate: malloc, free	
Observatii: n/a	

- Scrieti functia `btree_apply_by_level` care aplica functia transmisa ca parametru fiecarui nod al arborelui. Arborele trebuie parcurs nivel cu nivel. Functia apelata va lua trei parametri:
  - Primul parametru, de tip `void *`, corespunde elementului nod;
  - Al doilea parametru, de tip `int`, corespunde nivelului pe care se afla: 0 pentru radacina, 1 pentru copii, 2 pentru nepoti, etc.;
  - Al treilea parametru, de tip `int`, este 1 daca e vorba de primul nivel al nod-ului, si 0 in caz contrar.
- Ea trebuie sa aiba prototipul urmator:

```
void btree_apply_by_level(t_btree *root, void (*applyf)(void *item, int current_level, int is_first_elem))
```

# Capitolul XI

## Instructiuni intermediare

- Vom lucra in continuare cu arbori rosii si negri.


```
enum    e_rb_color
{
    RB_BLACK,
    RB_RED
};

typedef struct s_rb_node
{
    struct s_rb_node *parent;
    struct s_rb_node *left;
    struct s_rb_node *right;
    void *data;
    enum e_rb_color  color;
} t_rb_node;
```

- Nota: aceasta structura reia la inceput aceleasi campuri ca structura precedenta. E posibil de asemenea sa se reutilizeze functiile deja scrise pentru arborii rosii si negri. Pentru cei care sunt ceva mai avansati in programare, e vorba aici de o forma rudimentara de polimorfism in C.
- Va trebui sa puneti aceasta structura intr-un fisier `ft_btree_rb.h` si sa-l folositi la fiecare exercitiu.

# Capitolul XII

## Exercitiu 08: rb\_insert


	Exercitiu: 08
	rb_insert
	Director de lucru: <i>ex08/</i>
	Fisier(e) de iesire: <i>rb_insert.c, ft_btree_rb.h</i>
	Functii autorizate: <i>malloc</i>
	Observatii: <i>n/a</i>

- Scrieti functia `rb_insert` care adauga o valoare noua intr-un arbore astfel incat acesta va respecta constrangerile arborelui rosu si negru. Parametrul `root` pointeaza spre nodul radacina al arborelui. Dupa primul apel el va point-a spre NULL. Se va transmite de asemenea ca parametru o functie de comparare avand acelasi comportament cu `strcmp`.
- Ea trebuie sa aiba prototipul urmator:

```
void rb_insert(struct s_rb_node **root, void *data, int (*cmpf)(void *, void *));
```

# Capitolul XIII

## Exercitiu 09: rb\_remove

	Exercitiu: 09
	rb_remove
Director de lucru: <i>ex09/</i>	
Fisier(e) de iesire: <b>rb_remove.c</b> , <b>ft_btree_rb.h</b>	
Functii autorizate: <b>free</b>	
Observatii: <b>n/a</b>	

- Scrieti functia **rb\_remove** care suprima o valoare dintr-un arbore astfel incat el sa pastreze constrangerile unui arbore rosu si negru. Parametrul **root** point-eaza spre nodul radacina al arborelui. Se va transmite de asemenea ca parametru o functie de comparatie avand acelasi comportament ca **strcmp**, precum si un pointer spre functia **freef** care va fi apelata avand ca parametru elementul arborelui care trebuie sters.
- Ea trebuia sa aiba prototipul urmator:

```
void rb_remove(struct s_rb_node **root, void *data, int (*cmpf)(void *, void *), void (*freef)(void *));
```