Tópicos Avançados em Estrutura de Dados

Atividade 11

Bruna Galastri Guedes	18.00189-0
Daniel Ughini Xavier	18.00022-3
Rodolfo Cochi Bezerra	18.00202-0
Vítor Martin Simoni	18.00050-9
Leonardo Cury Haddad	18.00442-3
Leonardo de Barros Rodrigues	18.02401-7

01/08/2020

```
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;

    struct node * parent;
    struct node * firstchild;
    struct node * next;

};


struct node * root = NULL;

int size = 0;

struct node * cria_node(int valor){

    struct node * new_node;

    new_node = (struct node *) malloc (sizeof (struct node));
    new_node -> firstchild = NULL;
    new_node -> next = NULL;
    new_node -> parent = NULL;
    new_node -> data = valor;

    return new_node;

};

void insert_root (int valor){

    struct node * novo_node = cria_node(valor);
    root = novo_node;
    size = 1;
}

void posorder (struct node * ponteiro, int * soma) {

    struct node * trab = ponteiro -> firstchild;

    while (trab != NULL) {
        posorder(trab, soma);
        trab = trab -> next;
    }

    printf("\t %d", ponteiro -> data);

    *soma += ponteiro->data;

}


int main()
{
```

```
int somaJava, somaRuby;
somaJava = somaRuby = 0;

// criando a raiz

insert_root(1);

// nodes Java

struct node * java = cria_node(2);
struct node * grades = cria_node(8);
struct node * slides = cria_node(2);
struct node * slide01 = cria_node(3);
struct node * slide02 = cria_node(2);
struct node * slide03 = cria_node(4);
struct node * exercicios = cria_node(1);
struct node * exerc01 = cria_node(3);
struct node * exerc02 = cria_node(2);
struct node * exerc03 = cria_node(4);

// nodes Ruby

struct node * ruby = cria_node(1);
struct node * grades2 = cria_node(5);
struct node * projetos = cria_node(1);
struct node * papers = cria_node(2);
struct node * thread = cria_node(9);
struct node * buy = cria_node(8);
struct node * demos = cria_node(1);
struct node * market = cria_node(7);

// montando a arvore

root -> firstchild = java;

//path Java

java -> parent = root;
java -> next = ruby;
java -> firstchild = grades;

grades -> parent = java;
grades -> next = slides;

slides -> parent = java;
slides -> next = exercicios;
slides -> firstchild = slide01;

slide01 -> parent = slides;
slide01 -> next = slide02;

slide02 -> parent = slides;
slide02 -> next = slide03;

slide03 -> parent = slides;
```

```
exercicios -> parent = java;
exercicios -> firstchild = exerc01;

exerc01 -> parent = exercicios;
exerc01 -> next = exerc02;

exerc02 -> parent = exercicios;
exerc02 -> next = exerc03;

exerc03 -> parent = exercicios;

//path Ruby

ruby -> parent = root;
ruby -> firstchild = grades2;

grades2 -> parent = ruby;
grades2 -> next = projetos;

projetos -> parent = ruby;
projetos -> firstchild = papers;

papers -> parent = projetos;
papers -> next = demos;
papers -> firstchild = thread;

thread -> parent = papers;
thread -> next = buy;

buy -> parent = papers;

demos -> parent = projetos;
demos -> firstchild = market;

market -> parent = demos;


// executando os cálculos

printf("Usando POSTORDER\n");
printf("Total KB de arquivos /ruby \n");

posorder(ruby, &somaRuby);

printf("\nTamanho: %d Kb\n", somaRuby);

printf("Total KB de arquivos /java \n");

posorder(java, &somaJava);

printf("\nTamanho: %d Kb\n", somaJava);

printf("Total KB de toda estrutura:\n");
printf("Tamanho: %d Kb\n", somaJava + somaRuby);
```

```
    return 0;
}
```

Questão 1

a)

Total KB de arquivos /ruby Tamanho: 34 Kb

b)

Total KB de arquivos /java Tamanho: 31 Kb

c)

Total KB de toda a estrutura: Tamanho: 65 Kb