



Disciplina: Estrutura de Dados II

Professor: Adilso Nunes de Souza

Orientações:

- O aluno que não participou da atividade síncrona poderá assistir a gravação da aula, consultar o material disponibilizado e elaborar um resumo, com suas palavras, do conteúdo abordado e enviar por e-mail: adilsosouza@ifsul.edu.br para validar as presenças desta parte da aula.

-Realizar os exercícios propostos abaixo, para entregar compacte todos os arquivos .cpp em um único diretório com o nome do aluno e realize a entrega na atividade.

Lista de exercícios 3

1 – Seguindo o modelo implementado de árvore crie as funcionalidades solicitadas:

1.1 Encontrar a altura da sub-árvore esquerda e direita

1.2 Encontrar a altura da árvore

//criar a função

int altura (arvore *t)

```
{
    if (t == NULL)
        return 1;
    else
    {
        int he = 1 + altura (t->sae);
        int hd = 1 + altura (t->sad);
        if (he < hd)
            return hd;
        else
            return he;
    }
}
```

//chamada no main

```
case 4:
    system("cls");
    if(testa_vazia(t)) /* Verifica se a árvore está vazia */
        cout << "\n\nArvore vazia!!\n";
    else
    {
        cout << "ALTURA SAE: " << endl;
        if(t->sae == NULL)
            cout << "-1" << endl;
        else
            cout << altura(t->sae);
        cout << "\nALTURA SAD: " << endl;
        if(t->sad == NULL)
            cout << "-1" << endl;
        else
            cout << altura_sad(t->sad);
        cout << "\nALTURA DA ARVORE: " << endl;
        if(altura(t->sae) > altura(t->sad))
            cout << altura(t->sae);
```

```

        else
            cout << altura(t->sad);
    }
    getchar();

    break;

```

1.3 Limpar todos os nós da árvore, liberando as respectivas memórias.

```

//criar a função
arvore* limpa_arvore(arvore *t)
{
    if(t != NULL)
    {
        limpa_arvore(t->sae);
        limpa_arvore(t->sad);
        delete(t);
    }
    return NULL;
}

//chamada no main
case 5:
    system("cls");
    if(testa_vazia(t)) /* Verifica se a árvore está vazia */
        cout << "\n\nArvore vazia!!\n";
    else
    {
        t = limpa_arvore(t);
        cout << "\nTodos os nos foram eliminados. Arvore vazia";
    }
    getchar();
    break;

```

1.4 Dado um determinado valor, se o mesmo estiver contido na árvore encontrar o nível que ele se encontra.

```

//criar a função
void acha_nivel(arvore *t, int v, int nivel)
{
    if(!testa_vazia(t))
    {
        nivel++;
        if (t->info == v)
        {
            cout << "O Valor: " << t->info << " esta no nivel: ";
            cout << nivel << endl;
        }
        acha_nivel(t->sae, v, nivel);
        acha_nivel(t->sad, v, nivel);
        nivel--;
    }
}

//chamada no main
case 6:
    system("cls");

```

```

if(testa_vazia(t)) /* Verifica se a árvore está vazia */
    cout << "\n\nArvore vazia!!\n";
else
{
    cout << "Informe o valor a ser consultado o nivel: ";
    cin >> num;
    fflush(stdin);
    //consulta se o valor pertence a arvore
    if(consulta(t, num))
    {
        acha_nivel(t, num, -1);
    }
    else
        cout << "VALOR NAO PERTENCE A ARVORE.";

}
getchar();
break;

```

OBS: Inclua no menu as opções para acionar as funções criadas.

```

cout << "| 4 - Mostrar Altura   |" << endl;
cout << "| 5 - Limpar a arvore   |" << endl;
cout << "| 6 - Acha nivel       |" << endl;

```