# Relatório Tarefa 3

## Identificação:

Gabriel Losekann Paiva Murga

2320818

## Objetivo:

Inserir por nível as chaves: 10, 5, 15, 3, 7, 13, 20, 1, 4, 6 em uma árvore binária e exibir a árvore criada.

## Estrutura do programa:

O programa possui 4 funções e uma struct:

1. newNode: Onde eu crio um nó na minha árvore.
2. insertByLevel: Essa função insere os nós na arvore por nível, através de uma fila.
3. printPerOrd: Função recursiva que printa a arvore em pré-ordem
4. main: cria o vetor com as chaves, criar a cabeça da lista e chama as demais funções
5. struct tree: Possui como chave um valor inteiro, e dois ponteiros que apontam para o tipo struct tree, chamados esquerda e direita

## Solução:

**NewNode:** Recebe como parâmetro um inteiro referente a chave e retorna um ponteiro tipo Tree. De início aloca memória para o novo nó que está sendo criado, depois define a chave do novo nó como a chave que foi dada como parâmetro e os ponteiros esquerda e direita como nulos. No final retorna esse novo nó criado.

***insertByLevel:*** Recebe como parâmetro um ponteiro que aponta para um ponteiro de Tree que denominei como root, além disso recebe um inteiro chamado key. Começo minha função vendo se o conteúdo de root é null, ou seja, se ainda não foi adicionado nada na arvore, pois o root aponta para o número 10, o nível 0 dessa árvore. Se for null, eu insiro o nó com newNode(key) e dou return.

Caso não seja nulo, eu crio uma lista de ponteiros para árvore, e crio duas variáveis de controle com inteiros, e as defino como zero, uma chamada exit e outra arrival. Defino queue[arrival++]=\*root;, dessa forma a minha queue[1] vai ser a minha raiz/ cabeça.

Após isso, crio um looping com a condição de que, arrival>exit, ou seja, enquanto ainda houver gente na lista eu continuo o meu looping. Logo no início do looping, crio um novo ponteiro para Tree e defino como o nó atual, faço a seguinte atribuição Tree\* currentNode=queue[exit++], dessa forma a primeira vez que eu entra no looping, meu nó atual vai ser minha raiz pois vou estar em queue[1] . Após isso verifico se o ponteiro para esquerda do meu nó atual é nulo, se for, faço que currentNode->=newNode(key) e retorno a minha função, caso não seja, eu adiciono esse valor que está a minha esquerda para o final da minha fila da seguinte maneira queue[arrival++]=currentNode->left;. Após isso faço o primeiro procedimento só que para a direita. Esse looping funciona pois no início eu vou aumentando a minha fila de saída e apropriando o meu nó atual a esse valor, dessa forma eu consigo percorrer os nós da arvores e fazer as verificações que eu preciso.

**printPerOrd:** Essa função recebe como parâmetro a raiz da arvore e não ter return por ser uma função recursiva. Primeiramente ela verifica se a raiz é nula, caso não seja, printa o valor de sua chave, após isso se chamar recursivamente para seu nó à esquerda, que vai sempre ficar chamando o nó para esquerda de seus nós à esquerda por recursividade, e depois vai se chamar recursivamente para seu nó à direita.

**main:** Cria o vetor de inteiro “v” que possui os valores das chaves da minha árvore, criar a raiz da árvore, pega o tamanho do meu vetor com sizeof, e divide pelo o tamanho de um inteiro, para saber as quantidade de inteiros nele, faço um looping com for com esse dado, para que seja possível percorrer meu vetor e chamo minha função insertbylevel , colocando como parâmetro o endereço da minha raiz e v[i]. Após isso, chamo a minha função para exibir a lista.

## Observações:

**Saída do Programa:**

Pre-order tree:

10 5 3 1 4 7 6 15 13 20

**Dificuldade:**

A maior dificuldade do programa foi fazer a função insertbylevel, não consegui pensar sozinho em uma forma de como fazer isso, tive que pesquisar na internet, mas no final consegui entender o processo de criação de fila e achei uma forma muito prática.