```
    node->data = key;
    btree->elements = inicial_elements + 1;
}

void
tree_binary_insert_arr(BinTree *btree, key_t* arr, key_t size) {
    for (key_t k = 0; k < size; k++ ) {
        tree_binary_insert(btree, arr[k]);
    }
}

void
tree_binary_print_inorder(BinTree *btree) {

    /* Helper function to print the entire binary tree */
    void inorder(BinTree *btree, idx_t idx) {
        if (idx == 0) return;
        BinTreeNode* node = btree->root + idx;
        (void) inorder(btree, node->idx_left);
        (void) printf("%d ", node->data);
        (void) inorder(btree, node->idx_right);
    }

    BinTreeNode* root = btree->root;
    (void) printf("In-order traversal of the binary tree:\n");
    (void) printf("%d ", root->data);
    (void) inorder(btree, root->idx_left);
    (void) inorder(btree, root->idx_right);
    (void) puts("\n");
}

void
tree_binary_print(BinTree *btree) {

    uint32_t levels = 0;
    uint32_t n_elem = btree->elements;

    /* Contar niveis */
    while (n_elem > 1) {
        n_elem = n_elem >> 1;
        levels++;
    }

    BinTreeNode *root = btree->root;
    printf("%2d\n", root->data);

    /* Imprimir cada nivel */
    uint32_t idx = 1;
    uint32_t n_nodes = 1;
    for (uint32_t l = 0; l < levels; l++) {
        /* Cada nivel tem o dobro dos elementos no maximo  */
        n_nodes = n_nodes << 1;
        for (uint32_t i = 0; i < n_nodes; i++) {
            if (idx == btree->elements-1) break;
            printf("%2d  ", (root+idx)->data );
            idx++;
        }
        puts("");
    }

}
```