

# Árvore binária ordenada recursiva


Murilo Dantas

# Código

```
#include<iostream.h>
#include<conio.h>

// Definindo o registro que representará
// cada elemento da árvore binária
struct ARVORE
{
    int num;
    ARVORE *esq, *dir;
};

ARVORE* inserir(ARVORE* aux, int num)
{
    if (aux == NULL)
    {
        aux = new ARVORE();
        aux->num = num;
        aux->esq = NULL;
        aux->dir = NULL;
    }
    else if (num < aux->num)
        aux->esq=inserir(aux->esq, num);
    else
        aux->dir=inserir(aux->dir, num);
    return aux;
}
```



```
int consultar(ARVORE *aux, int num, int achou)
{
    if (aux != NULL && achou == 0)
    {
        if (aux->num == num)
        {
            achou = 1;
        }
        else if (num < aux->num)
            achou = consultar(aux->esq, num, achou);
        else
            achou = consultar(aux->dir, num, achou);
    }
    return achou;
}
```

# Código

```
void mostraremordem(ARVORE *aux)
{
    if (aux != NULL)
    {
        mostraremordem(aux->esq);
        cout << aux->num << " ";
        mostraremordem(aux->dir);
    }
}

void mostrarpreordem(ARVORE *aux)
{
    if (aux != NULL)
    {
        cout << aux->num << " ";
        mostrarpreordem(aux->esq);
        mostrarpreordem(aux->dir);
    }
}

void mostrarposordem(ARVORE *aux)
{
    if (aux != NULL)
    {
        mostrarposordem(aux->esq);
        mostrarposordem(aux->dir);
        cout << aux->num << " ";
    }
}
```

```
ARVORE* remover(ARVORE *aux, int num)
{
    ARVORE *p, *p2;
    if (aux->num == num)
    {
        if (aux->esq == aux->dir)
        { // o elemento a ser removido não tem filhos
            delete aux;
            return NULL;
        }
        else if (aux->esq == NULL)
        { // o elemento a ser removido
            // não tem filho para a esquerda
            p = aux->dir;
            delete aux;
            return p;
        }
        else if (aux->dir == NULL)
        { // o elemento a ser removido
            // não tem filho para a direita
            p = aux->esq;
            delete aux;
            return p;
        }
        else
        { // o elemento a ser removido
            // tem filho para ambos os lados
            p2 = aux->dir;
            p = aux->dir;
            while (p->esq != NULL)
            {
                p = p->esq;
            }
            p->esq = aux->esq;
            delete aux;
            return p2;
        }
    }
}
```

```
    else if (aux->num < num)
        aux->dir = remover(aux->dir, num);
    else
        aux->esq = remover(aux->esq, num);
    return aux;
}

ARVORE* desalocar(ARVORE* aux)
{
    if (aux != NULL)
    {
        aux->esq = desalocar(aux->esq);
        aux->dir = desalocar(aux->dir);
        delete aux;
    }
    return NULL;
}
```

# Código

```
void main()
{
    ARVORE *raiz = NULL;
    // o ponteiro aux é um ponteiro auxiliar
    ARVORE *aux;
    // o ponteiro aux1 é um ponteiro auxiliar
    int op, achou, numero;
    do
    {
        clrscr();
        cout << "\nMENU DE OPÇÕES\n";
        cout << "\n1 - Inserir na árvore";
        cout << "\n2 - Consultar um nó da árvore";
        cout << "\n3 - Consultar toda a árvore em
            ↳ ordem";
        cout << "\n4 - Consultar toda a árvore em
            ↳ pré-ordem";
        cout << "\n5 - Consultar toda a árvore em
            ↳ pós-ordem";
        cout << "\n6 - Excluir um nó da árvore";
        cout << "\n7 - Esvaziar a árvore";
        cout << "\n8 - Sair";
        cout << "\nDigite sua opção: ";
        cin >> op;
        if (op < 1 || op > 8)
            cout << "\nOpção inválida!!";
```

```
    }
    else if (op == 1)
    {
        cout << "\nDigite o número a ser
            ↳ inserido na árvore: ";
        cin >> numero;
        raiz = inserir(raiz, numero);
        cout << "\nNúmero inserido na
            ↳ árvore!!";
    }
    else if (op == 2)
    {
        if (raiz == NULL)
        {
            // a árvore está vazia
            cout << "\nÁrvore vazia!!";
        }
        else
        {
            // a árvore contém elementos
            cout << "\nDigite o elemento a ser
                ↳ consultado";
            cin >> numero;
            achou = 0;
            achou =
                ↳ consultar(raiz, numero, achou);
            if (achou == 0)
                cout << "\nNúmero não
                    ↳ encontrado na árvore!";
            else
                cout << "\nNúmero encontrado
                    ↳ na árvore!";
        }
    }
}
```

# Código

```
else if (op == 3)
{
    if (raiz == NULL)
    {
        // a árvore está vazia
        cout << "\nÁrvore vazia!!";
    }
    else
    {
        // a árvore contém elementos
        // e estes serão mostrados em ordem
        cout << "\nListando todos os
        ↳ elementos da árvore em ordem";
        mostraremordem(raiz);
    }
}
else if (op == 4)
{
    if (raiz == NULL)
    {
        // a árvore está vazia
        cout << "\nÁrvore vazia!!";
    }
    else
    {
        // a árvore contém elementos
        // e estes serão mostrados
        // em pré-ordem
```

```
        cout << "\nListando todos os
        ↳ elementos
        ↳ da árvore
        ↳ em pré-ordem";
        mostrarpreordem(raiz);
    }
}
else if (op == 5)
{
    if (raiz == NULL)
    {
        // a árvore está vazia
        cout << "\nÁrvore vazia!!";
    }
    else
    {
        // a árvore contém elementos
        // e estes serão mostrados em
        // pós-ordem
        cout << "\nListando todos os
        ↳ elementos
        ↳ da árvore
        ↳ em pós-ordem";
        aux = raiz;
        mostrarposordem(aux);
    }
}
```

```
else if (op == 6)
{
    if (raiz == NULL)
        cout << "\nÁrvore vazia!!";
    else
    {
        cout << "\nDigite o número que
        ↳ deseja excluir:";
        cin >> numero;
        achou = 0;
        ↳ consultar(raiz, numero, achou);
        if (achou == 0)
            cout << "\nNúmero não encontrado na
            ↳ árvore!";
        else
        {
            raiz = remover(raiz, numero);
            cout << "\nNúmero excluído da
            ↳ árvore!";
        }
    }
}
else if (op == 7)
{
    if (raiz == NULL)
        cout << "\nÁrvore vazia!!";
    else
    {
        raiz=desalocar(raiz);
        cout << "\nÁrvore
        ↳ esvaziada!!";
    }
}
getch();
}while (op != 8);
raiz = desalocar(raiz);
}
```

**Perguntas?**

# Bibliografia da aula

- ASCENCIO, A. F. G.; ARAÚJO, G. S. Estrutura de dados. Algoritmos, análise da complexidade e implementação em Java e C/C++. 1ª edição. Pearson. 2010.