

```

#include<iostream>
#include<string.h>
#include<stdlib.h>

using namespace std;

typedef struct {
    char nome[30];
    int idade;
    int ra;
    float nota;
} TipoItem;

typedef struct No *Apontador;

typedef struct No {
    TipoItem item;
    Apontador prox;
    Apontador ant;
} NoAluno;

typedef struct {
    Apontador Primeiro, Ultimo;
} TipoLista;

void criaLista(TipoLista *Lista);
void insereNo(TipoItem aluno, TipoLista *Lista);
void removeNo(int ra, TipoLista *Lista);
void buscaNoSemRetorno(int ra, TipoLista *Lista);
NoAluno* buscaNoComRetorno(int ra, TipoLista *Lista);
void buscaRecurativa(int ra, Apontador p);
NoAluno* buscaRecurativaComRetorno(int ra, Apontador p);
void imprimeLista(TipoLista *Lista);
void imprimeListaRecurativa(Apontador p);

main(){
    TipoItem aluno;
    TipoLista Lista;

    // cria lista dinâmica
    criaLista(&Lista);

    // inserção
    cout << endl << "INSERINDO RA: 1234" << endl;
    strcpy(aluno.nome, "Teste");
    aluno.idade=23;
    aluno.ra=1234;
    aluno.nota = 8.5;
    insereNo(aluno,&Lista);

    // inserção
    cout << "INSERINDO RA: 2345" << endl;
    strcpy(aluno.nome, "Teste2");
    aluno.idade=25;
    aluno.ra=2345;
    aluno.nota = 6.5;
    insereNo(aluno,&Lista);

    // inserção
    cout << "INSERINDO RA: 2345" << endl;
    strcpy(aluno.nome, "Teste2");
    aluno.idade=25;
    aluno.ra=2345;
    aluno.nota = 6.5;
    insereNo(aluno,&Lista);

    // inserção
    cout << "INSERINDO RA: 5678" << endl;
    strcpy(aluno.nome, "Teste3");
    aluno.idade=27;
    aluno.ra=5678;
    aluno.nota = 4.5;
    insereNo(aluno,&Lista);

```

```

// imprime lista
imprimeLista(&Lista);
cout << endl;

// testa busca sem retorno
cout << endl << "BUSCA SEM RETORNO - PROCURANDO RA: 2345" << endl;
buscaNoSemRetorno(2345,&Lista);

// testa busca com retorno
cout << endl << "BUSCA COM RETORNO - PROCURANDO RA: 2345" << endl;
NoAluno *p = buscaNoComRetorno(2345,&Lista);
cout << "ENCONTRADO! Nome: " << p->item.nome << " Idade: " << p->item.idade << " RA: " << p->item.ra << " Nota: " << p->item.nota << endl;

// testa busca recursiva sem retorno
cout << endl << "BUSCA RECURSIVA SEM RETORNO - PROCURANDO RA: 2345" << endl;
buscaRecursiva(2345,Lista.Primeiro->prox);

// testa busca recursiva com retorno
cout << endl << "BUSCA RECURSIVA COM RETORNO - PROCURANDO RA: 2345" << endl;
p = buscaRecursivaComRetorno(2345,Lista.Primeiro->prox);
cout << "ENCONTRADO! Nome: " << p->item.nome << " Idade: " << p->item.idade << " RA: " << p->item.ra << " Nota: " << p->item.nota << endl;

// remoção
cout << endl << "REMOVENDO RA: 2345" << endl;
removeNo(2345,&Lista);

// imprime lista recursiva
cout << endl << "IMPRESSÃO RECURSIVA DA LISTA" << endl;
imprimeListaRecursiva(Lista.Primeiro->prox);
cout << endl << endl;
}

// inicia lista duplamente encadeada usando nó-cabeça, ou seja,
// primeiro nó da lista não contém elemento válido
void criaLista(TipoLista *Lista){
    Lista->Primeiro = (Apontador) malloc(sizeof(NoAluno)); // aloca espaço para nó cabeça
    Lista->Ultimo = Lista->Primeiro; // equivalente a (*Lista).Ultimo = (*Lista).Primeiro;
    Lista->Primeiro->prox = NULL; // é obrigatório atribuir NULL
}

// insere novo nó na lista duplamente encadeada
void insereNo(TipoItem aluno, TipoLista *Lista){
    // primeiro checa se novo elemento já está na lista
    Apontador p = buscaRecursivaComRetorno(aluno.ra,Lista->Primeiro->prox);
    if (p==NULL) { // se não está na lista, insere
        Lista->Ultimo->prox = (Apontador) malloc(sizeof(NoAluno)); // aloca espaço para novo nó
        Apontador q = Lista->Ultimo;
        Lista->Ultimo = Lista->Ultimo->prox;
        Lista->Ultimo->item = aluno;
        Lista->Ultimo->ant = q;
        Lista->Ultimo->prox = NULL;
    } else { // se já está na lista, dá mensagem de erro
        cout << "Elemento já está na lista!" << endl;
    }
}

// imprime todos os nós da lista iterativamente
void imprimeLista(TipoLista *Lista){
    Apontador p;
    p = Lista->Primeiro->prox;
    cout << endl << "IMPRESSÃO DA LISTA" << endl;
    while (p!=NULL){
        cout << "ANTERIOR: ";
        if (p->ant!=NULL) cout << p->ant->item.nome;
        cout << " ATUAL: " << p->item.nome << " Idade: " << p->item.idade << " RA: " << p->item.ra << " Nota: " << p->item.nota << " ";
        cout << "PRÓXIMO: ";
        if (p->prox!=NULL) cout << p->prox->item.nome << endl;
        p = p->prox;
    }
}

```

```

// imprime todos os nós da lista recursivamente
void imprimeListaRecursiva(Apontador p){
    if (p!=NULL){
        cout << "ANTERIOR: ";
        if (p->ant!=NULL) cout << p->ant->item.nome;
        cout << "          ATUAL: " << p->item.nome << " Idade: " << p->item.idade << " RA: " << p->item.ra << " Nota: " << p->item.nota << "          ";
        cout << "PRÓXIMO: ";
        if (p->prox!=NULL) cout << p->prox->item.nome << endl;
        imprimeListaRecursiva(p->prox);
    }
}

// remove um nó da lista
void removeNo(int ra, TipoLista *Lista){
    // primeiro busca endereço do nó a ser removido
    Apontador p = buscaRecursivaComRetorno(ra,Lista->Primeiro->prox);
    if (p!=NULL) { // se encontrou o nó, remove
        Apontador q=p->ant;
        q->prox = p->prox;
        p->prox->ant=q;
        free(p);
    } else { // se não encontrou o nó, dá mensagem de erro
        cout << "Registro não encontrado para remoção!" << endl;
    }
}

// busca iterativa sem retorno
void buscaNoSemRetorno(int ra, TipoLista *Lista){
    Apontador p;
    p = Lista->Primeiro->prox;
    while ((p!=NULL) && (p->item.ra!=ra)){
        p = p->prox;
    }
    if (p!=NULL){
        cout << "ENCONTRADO! Nome: " << p->item.nome << " Idade: " << p->item.idade << " RA: " << p->item.ra << " Nota: " << p->item.nota << endl;
    } else {
        cout << "Registro não encontrado!" << endl;
    }
}

// busca recursiva sem retorno
void buscaRecursiva(int ra, Apontador p){
    if (p!=NULL){
        if (p->item.ra!=ra){
            buscaRecursiva(ra,p->prox);
        } else {
            cout << "ENCONTRADO! Nome: " << p->item.nome << " Idade: " << p->item.idade << " RA: " << p->item.ra << " Nota: " << p->item.nota << endl;
        }
    } else {
        cout << "Registro não encontrado!" << endl;
    }
}

// busca recursiva com retorno
NoAluno* buscaRecursivaComRetorno(int ra, Apontador p){
    if (p!=NULL){
        if (p->item.ra!=ra){
            buscaRecursivaComRetorno(ra,p->prox);
        } else {
            return p;
        }
    } else {
        return NULL;
    }
}

// busca iterativa com retorno
NoAluno* buscaNoComRetorno(int ra, TipoLista *Lista){
    Apontador p;
    p = Lista->Primeiro->prox;
    while ((p!=NULL) && (p->item.ra!=ra)){

```

```
    p = p->prox;
}
if (p!=NULL){
    return p;
} else {
    return NULL;
}
}
```