Aula 13:
- Estruturas (parte 2)

Prof. João Henrique Kleinschmidt
Material elaborado pelo prof. Jesús P. Mena-Chalco

3Q-2018

Estruturas (=registros)

Estruturas

```c
#include<stdio.h>

int main()
{
    struct ponto3D {
        double x;
        double y;
        double z;
    };

    struct ponto3D p1;   /*um registro p1 do tipo ponto3D*/

    printf("%ld\n", sizeof(p1));
    printf("%f %f %f\n", p1.x, p1.y, p1.z);

}
```

```
24
0.000000 0.000000 0.000000
```

maiorDistancia.c

```c
#include <stdio.h>
#include <math.h>

struct ponto3D {
    double x;
    double y;
    double z;
};

double dEuclidiana(struct ponto3D p, struct ponto3D q) {
    return sqrt(pow(p.x-q.x,2) + pow(p.y-q.y,2) + pow(p.z-q.z,2));
}


int main() {
    int i, j, n;
    double maiorD = 0;

    scanf("%d", &n);
    struct ponto3D v[n] ;

    // Leitura dos pontos
    for (i=0; i<n; i++)
        scanf("%lf %lf %lf", &(v[i].x), &(v[i].y), &(v[i].z) );

    // Busca pela maior distancia entre quaisquer 2 pontos 3D
    for (i=0; i<n-1; i++)
        for (j=i+1; j<n; j++)
            if ( maiorD < dEuclidiana(v[i], v[j]) )
                maiorD = dEuclidiana(v[i], v[j]);

    printf("%lf", maiorD);
}
```

Typedef

O tipo "*struct*" pode não ser muito prático para digitar diversas vezes, e não ajuda na organização do código.

```
typedef struct fracao {
        int num;
        int den;
} fracao;

fracao x;
x.num = 1;
x.den = 2;
```

A instrução typedef pode ser utilizada para definir quaisquer tipos:

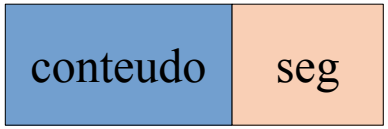
```
typedef unsigned int uint;
uint x;
```

Lista ligada (lista encadeada)

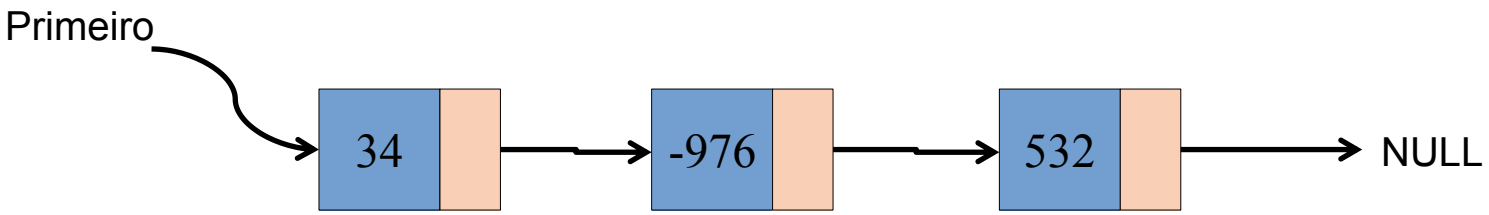
Uma lista ligada

Uma lista ligada é composta por nós. Cada nó armazena:

- dado relacionado a ele; e
- endereço (ponteiro) para próximo nó na lista.




```c
struct cel {
    int conteudo;
    struct cel *seg;
};

typedef struct cel celula;
```



listaLigada1.c

```c
int main() {
    int i, n, numero;

    scanf("%d", &n);
    celula* primeiro = NULL;

    // Leitura dos pontos e criacao dos elementos
    for (i=0; i<n; i++) {
        scanf("%d", &numero);
        primeiro = inserirElemento(primeiro, numero);
    }
}
```

```c
celula* inserirElemento(celula* primeiro, long int numero) {
    celula* novo   = (celula *) malloc(sizeof(celula));
    if (novo==NULL){
        printf("\nNao foi possivel criar celula");
        exit(1);
    }
    novo->seg      = NULL;
    novo->conteudo = numero;

    celula* p = primeiro;
    if (p==NULL)
        primeiro = novo;
    else {
        while (p->seg!=NULL)
            p = p->seg;
        p->seg = novo;
    }
    return primeiro;
}
```

```c
struct cel {
    int conteudo;
    struct cel *seg;
};

typedef struct cel celula;
```

Exercício 2

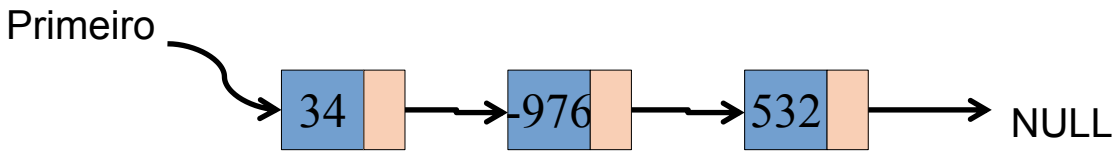
Modifique o programa anterior para imprimir o maior elemento armazenado na lista ligada.

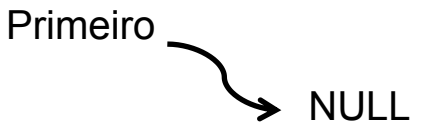
- Nome do arquivo: listaLigada2.c

```
celula* maiorElemento(celula* primeiro); // prototipo


int main() {
    int i, n, numero;

    scanf("%d", &n);
    celula* primeiro = NULL;

    // Leitura dos pontos e criacao dos elementos
    for (i=0; i<n; i++) {
        scanf("%d", &numero);
        primeiro = inserirElemento(primeiro, numero);
    }

    celula* maior = maiorElemento(primeiro);
    printf("\n%d", maior->conteudo);
}
```

listaLigada2.c

```
celula* maiorElemento(celula* primeiro) {
    celula* resposta = primeiro;
    celula* p;

    for (p=primeiro; p!=NULL; p=p->seg) {
        if (resposta->conteudo < p->conteudo)
            resposta = p;
    }
    return resposta;
}
```





Exercício 3

O seguinte programa é uma modificação do programa anterior utilizado para exemplificar como eliminar o elemento que está na primeira posição da lista.

- Nome do arquivo: listaLigada3.c

listaLigada3.c

```c
int main() {
    int i, n, numero;

    scanf("%d", &n);
    celula* primeiro = NULL;

    // Leitura dos pontos e criacao dos elementos
    for (i=0; i<n; i++) {
        scanf("%d", &numero);
        primeiro = inserirElemento(primeiro, numero);
    }

    printf("\n%d", primeiro->conteudo);
    primeiro = eliminarPrimeiroElemento(primeiro);
    printf("\n%d", primeiro->conteudo);

}
```

```c
celula* eliminarPrimeiroElemento(celula* primeiro) {
    celula* p = primeiro;

    primeiro = p->seg;
    free(p);
    return primeiro;
}
```

Exercício 4

O seguinte programa é uma modificação do programa anterior utilizado para exemplificar como eliminar o elemento que está em uma determinada posição da lista.

- Nome do arquivo: listaLigada4.c

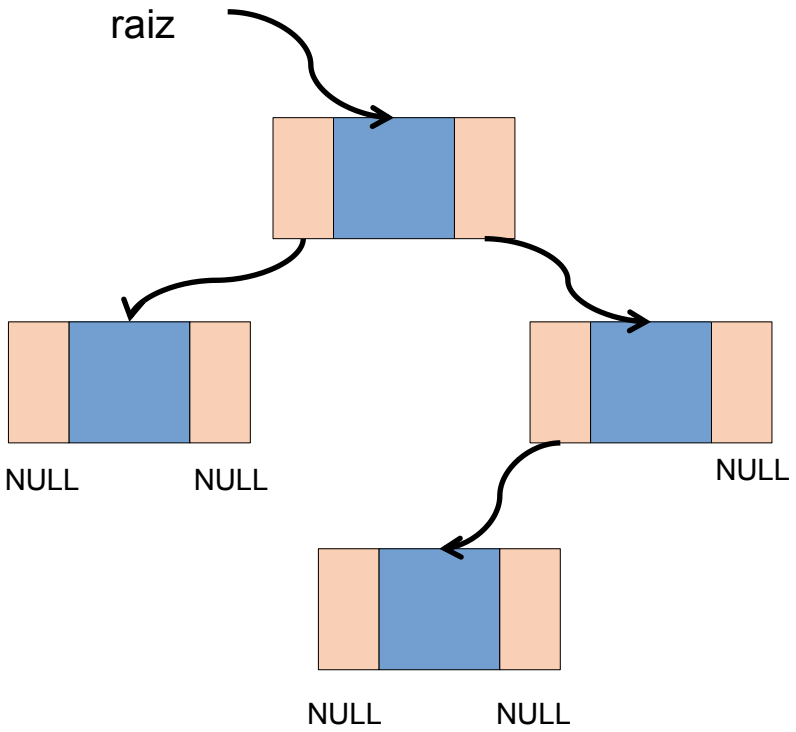
```
celula* eliminarElemento(celula* primeiro, int posicao) {
    celula* p = primeiro;

    if (posicao==0) {
        primeiro = p->seg;
        free(p);
        return primeiro;
    }

    while (posicao>1) {
        posicao--;
        p = p->seg;
    }
    // aqui p->seg aponta ao elemento a ser eliminado
    celula* lixo;
    lixo   = p->seg;
    p->seg = p->seg->seg;
    free(lixo);

    return primeiro;
}
```

Árvore (versão simples)

Uma árvore binária

Uma árvore binária


```c
struct cel {
    int conteudo;
    struct cel *esq;
    struct cel *dir;
};

typedef struct cel celula;
```



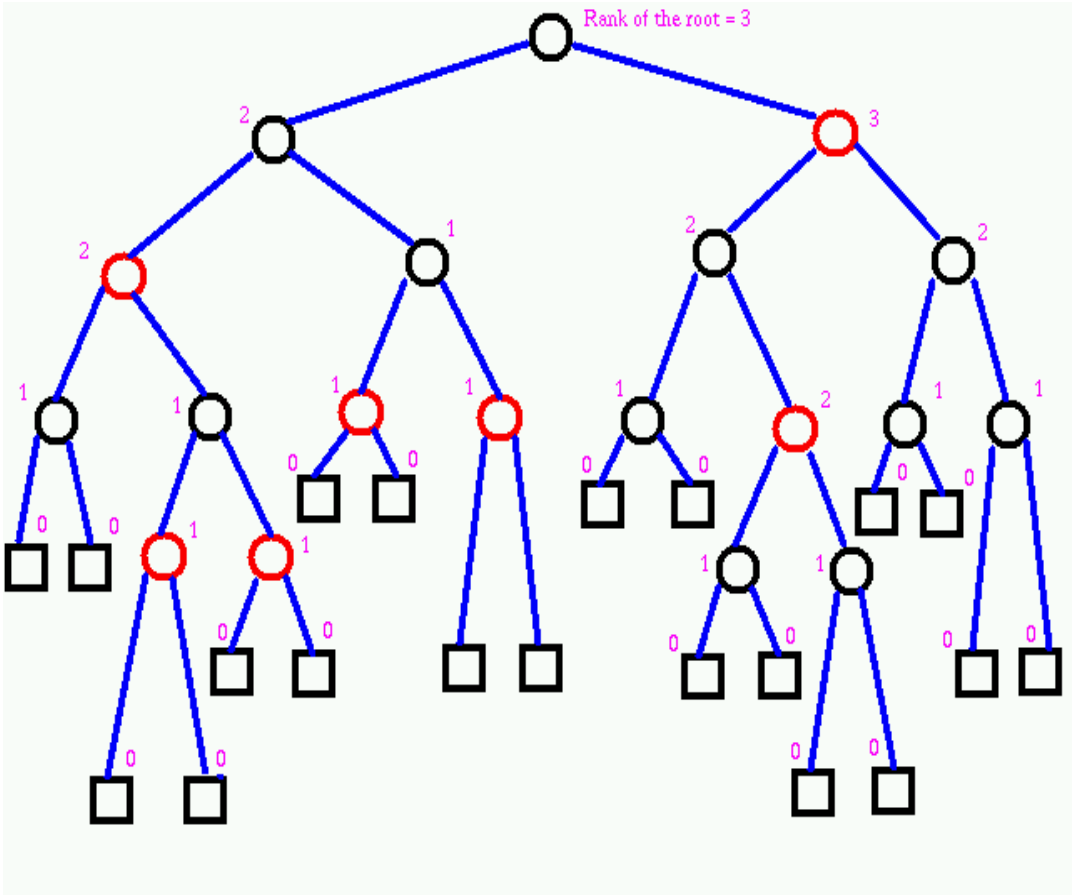
Uma árvore binária

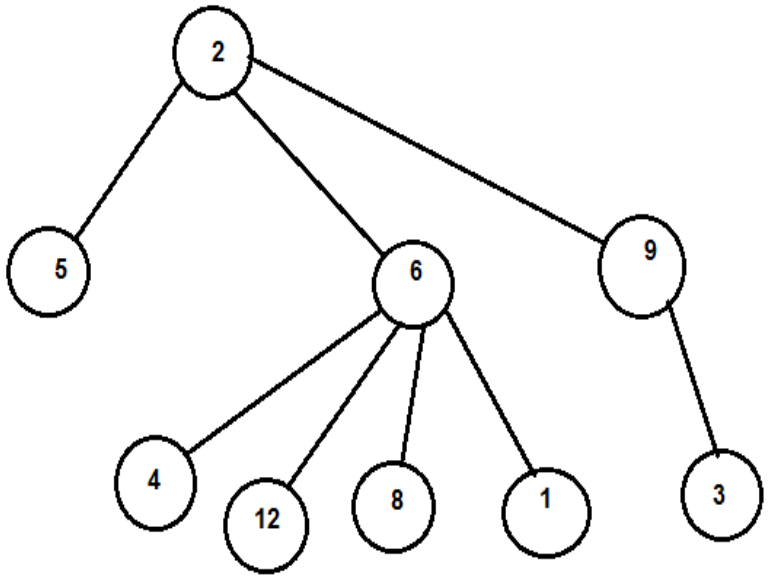

```c
struct cel {
    int conteudo;
    struct cel *esq;
    struct cel *dir;
};

typedef struct cel celula;
```

```c
celula* criarElemento (int numero) {
    celula* novo    = (celula *) malloc(sizeof(celula));
    novo->esq       = NULL;
    novo->dir       = NULL;
    novo->conteudo = numero;

    return novo;
}
```

Nome do arquivo: arvoreBinaria1.c

Árvores





Sobre a Prova 1