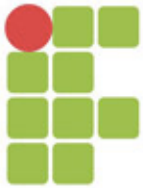
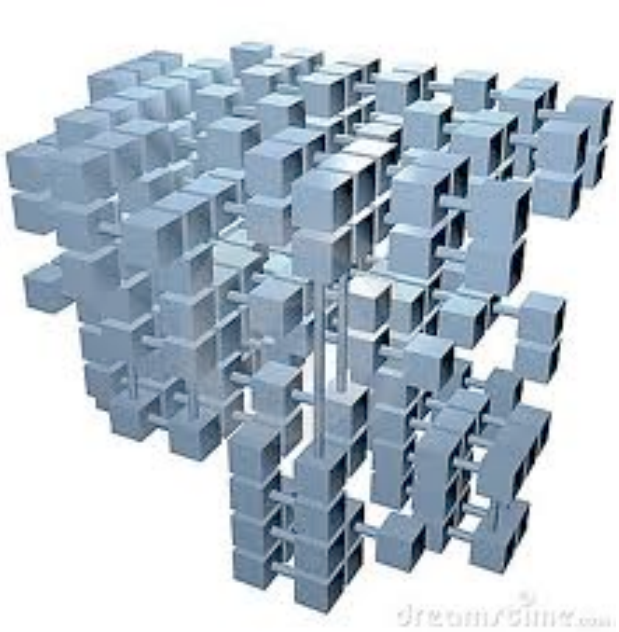


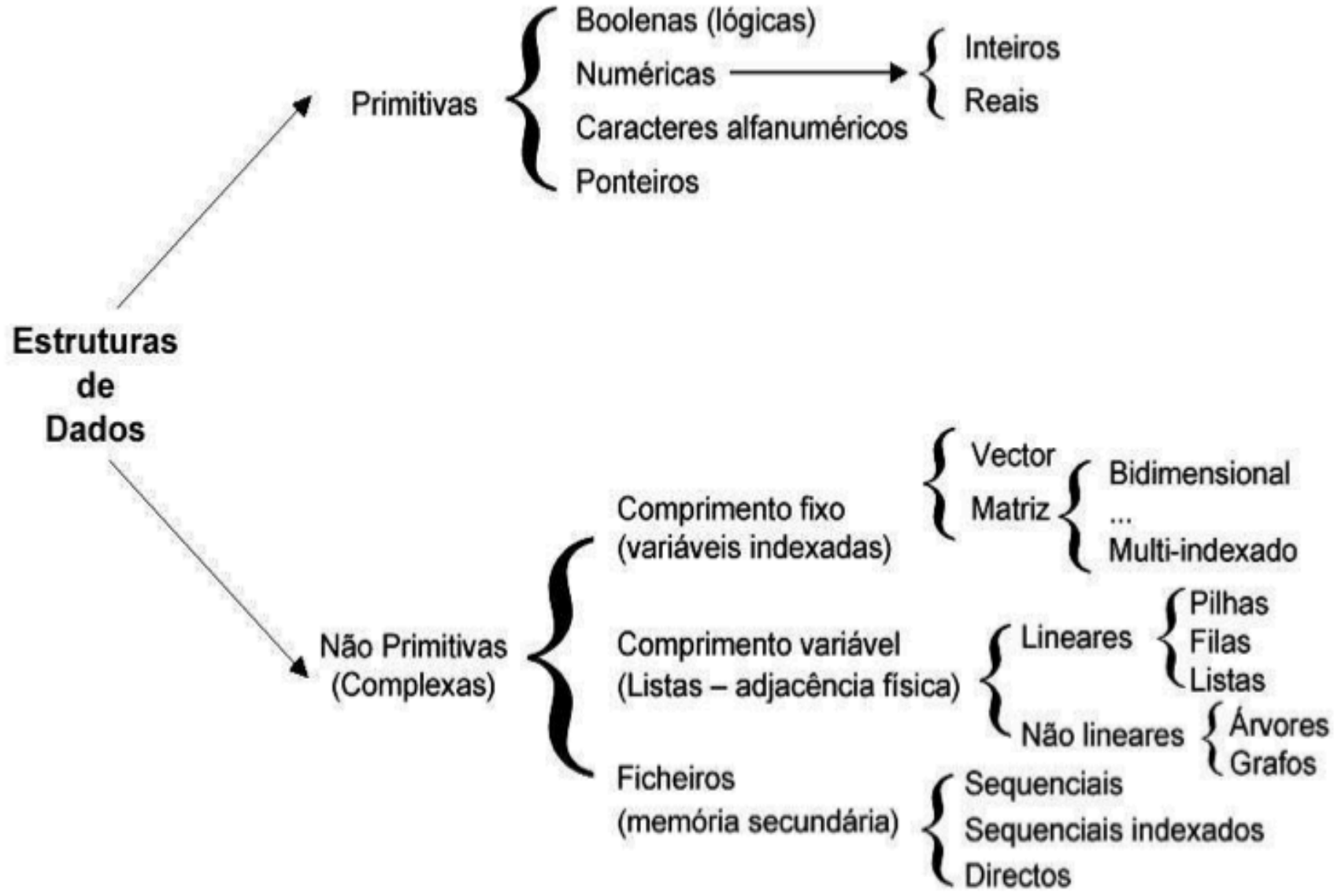
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
GOIÁS

Estrutura de Dados

Prof. Sirlon Diniz de Carvalho



Estrutura de Dados Fundamentos



Estrutura de Dados
Struct

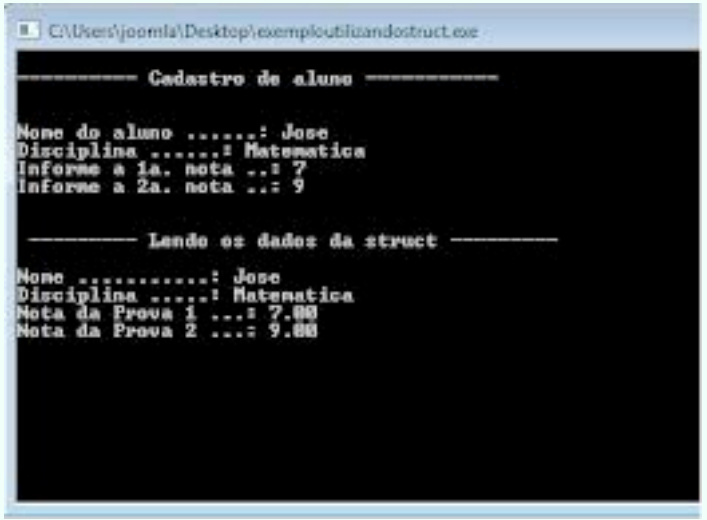
Registros (Estruturas Heterogêneas – Struct)

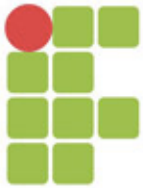
- Estrutura (struct) ou registro em C:
  - coleção de um ou mais valores, agrupados sob um único nome
  - constituem um recurso importante para organizar os dados
  - Possibilita tratar um grupo de valores como uma única variável

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
GOIÁS

Estrutura de Dados Struct

- Vetores e matrizes (Comparação)
  - Estruturas de dados homogêneas
  - Armazenam valores de um mesmo tipo
  - Podem ser de tipos primitivos: int, double, float, char
- Exemplo:
  - int vet[10]
  - char vetc[5]
  - float mat[4][[4]





INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
GOIÁS

Estrutura de Dados Struct

- Problemas reais
  - Coleções de dados de tipos diferentes
  - Exemplo: ficha de um cadastro de cliente
    - Nome: string
    - Endereço: string
    - Telefone: string
  - Elementos do registro
    - São campos ou membros da struct
    - Utilizado para armazenar informações de um mesmo objeto

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
GOIÁS

Estrutura de Dados
Struct

- Sintaxe na Linguagem C
  - Palavra reservada struct
  - Preferencialmente deve ser declarada após incluir as bibliotecas e antes da main

```
struct <identificador_struct> {
        tipo <nome_variável_campo1>;
        tipo <nome_variável_campo2>;
        :
} <variáveis_estrutura>;
struct <identificador_struct> <var1>, <var2>;
```

Estrutura de Dados Struct

- Se o Compilador C for compatível com o padrão C ANSI
  - Informação contida em uma struct pode ser atribuída a outra struct do mesmo tipo
  - Não é necessário atribuir os valores de todos os elementos/campos separadamente
  - Por exemplo: <var1> = <var2>;
    - Todos os campos de <var1> receberão os valores correspondentes dos campos de <var2>

Estrutura de Dados
Struct

- Para acessar os campos da struct
  - Utiliza-se o nome da variável struct, seguido de ponto, seguido do nome do campo

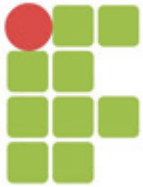
- Por exemplo:
  - <var1>.<nome_variável_campo2>;

Estrutura de Dados
Struct

- Sintaxe na Linguagem C
  - Por exemplo um struct endereço que guarda os elementos nome, rua, cidade, estado e cep

```
struct endereco{
        char nome[30];
        char rua[40];
        long int cep;
};
```

- Foi feita apenas a declaração da struct, ainda não foi criada nenhuma variável da struct endereço
- O comando para declarar uma variável com esta struct é:
  - struct endereco info_end;



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
GOIÁS

Estrutura de Dados Struct

- Vimos como se faz para acessar os membros de uma struct: nome_variável.nome_membro
- Considerando o exemplo anterior
  - para inicializar o cep da variável info_end que é uma variável da struct endereço, faz-se: info_end.cep = 74000000;
  - Para obter o nome da pessoa e colocar na string nome da struct se poderia utilizar:
    - gets(info_end.nome);
  - Para imprimir o endereço faz-se:
    - printf("%s", info_end.rua);

Estrutura de Dados
Struct

– Exercício 1

- Criar uma estrutura chamada QAluno, que armazene a média, as faltas e o nome de um aluno. Na função main: criar uma variável que é uma estrutura QAluno; ler o nome, a média e as faltas de um aluno e armazenar na variável criada; exibir na tela o nome, a média e as faltas do aluno.

Estrutura de Dados
Struct

– Exercício 2

- Criar uma estrutura chamada QAluno, que armazene o nome, as notas de N1, N2 e N3 e as faltas de um aluno. Na função main: criar uma variável que é uma estrutura QAluno; ler o nome, as notas e as faltas de um aluno e armazenar na variável criada; exibir na tela o nome, a média e as faltas do aluno.

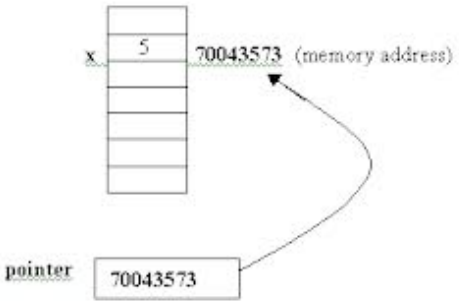
– Exercício 3

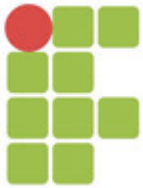
- A partir do exercício anterior, faça um programa que leia e armazene as notas de toda a turma

Estrutura de Dados
Ponteiros

Ponteiros

- Conceito:
  - Um **ponteiro** indica o **endereço** onde um valor é armazenado na memória do computador, ao contrário de uma **variável** que representa um **valor** numérico real.
  - Em C os ponteiros frequentemente fornecem técnicas muito efetivas para representar e manipular dados, especialmente estruturas de dados.





INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
GOIÁS

Estrutura de Dados
Ponteiros

Ponteiros em C

– Declarando variáveis ponteiro
  - Pode-se declarar uma variável ponteiro para representar o endereço de qualquer tipo de dado em C.
  - Sintaxe:

    tipo *variável;

– Exemplo:

```
int x;
int *pi; /* compilador sabe que pi é ponteiro */
        /* pi é um ponteiro para inteiro */
```

Estrutura de Dados
Ponteiros

Ponteiros em C

- O símbolo * é chamado operador de indireção. Se aplicado a uma variável ponteiro, o símbolo indica o valor real armazenado no endereço contido no ponteiro.
- O símbolo * ***versus*** o símbolo &
  - O símbolo & é conhecido como operador endereço. Anexado no início de um nome de variável, esse operador representa o endereço na memória onde o valor da variável está armazenado.

Estrutura de Dados
Ponteiros

Ponteiros em C

– O operador “&” quando aplicado sobre uma variável retorna o seu endereço

– Exemplo:

```
int x = 10, *pi;
pi = &x;
printf(“&x: %p pi: %p”, &x, pi);
```

Qual o resultado?

Estrutura de Dados
Ponteiros

Ponteiros em C

– O operador "*" quando aplicado sobre um ponteiro retorna o dado apontado

– Exemplo:


```
void main () {
    int *tmp_ptr;
    int x, y;
    x = 10;
    tmp_ptr = &x;
    y = *tmp_ptr; /* (*tmp_ptr) = 10 */
}
```

Estrutura de Dados
Ponteiros

Ponteiros em C

– Ponteiros são variáveis tipadas:

- (int *) # (float *) # (char *)

– Exemplo:

```
main() {
    int *ip, x;
    float *fp, z;
     ip = &x;  /* OK */
     fp = &z;  /*OK */
    ip = &z;  /*erro */
     fp = &x;  /*erro */
}
```

Estrutura de Dados
Ponteiros

Ponteiros em C

– Exemplo:

```
void main() {
    int x = 10;
    int *pi;
    pi=&x;      /*  *pi==10    */
    (*pi)++;    /*   *pi == 11 */
     printf("%d", x);
}
```

Qual o resultado?

Estrutura de Dados
Ponteiros

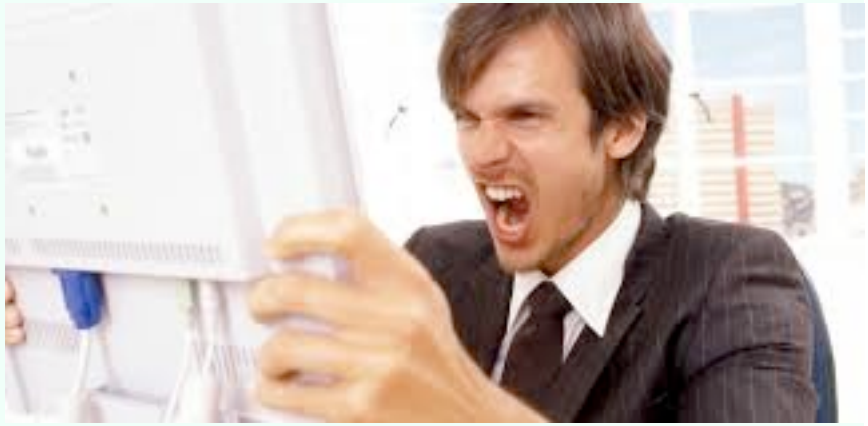
Ponteiros em C

– Exemplo:

```
void main() {
    int x = 10;
    int *pi, *pj;
    pi=&x;      /*  *pi==10    */
    pj=pi;      /*   *pj==10    */
    (*pi)++;    /*   (*pi, *pj, x) == 11    */
    (*pj)++;    /*   (*pi, *pj, x) == 12    */
    printf("%d", x);       /*   ==> 12    */
}
```

Estrutura de Dados
Ponteiros

- Exercício 1
  - defina e inicialize uma variável inteira
  - defina um ponteiro para inteiro
  - modifique o valor da variável através do ponteiro
  - Imprima e verifique os novos valores da variável



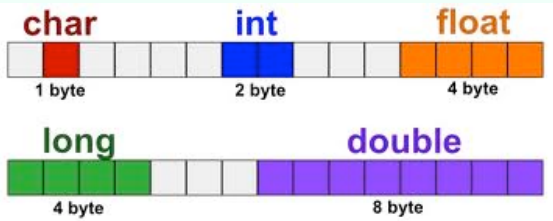
Estrutura de Dados
Memória

Alocação Estática e Dinâmica de Memória

- Alocação estática (variáveis)
  - acontece antes que o programa comece a ser executado: Exemplo: *char c; int i; int v[10];*

- Alocação dinâmica
  - a quantidade de memória somente é conhecida em tempo de execução
  - é gerenciada pelas funções ***malloc*** e ***free***, que estão na biblioteca stdlib, portanto, é necessário incluí-la

```
#include <stdlib.h>
```



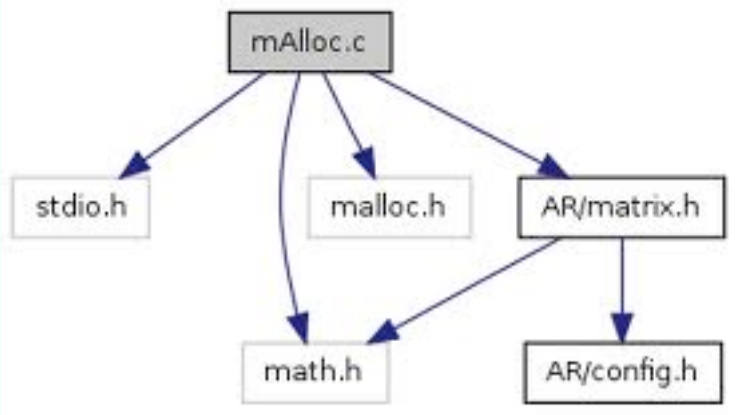
Estrutura de Dados
Memória

A Função **malloc**

– Abreviatura de memory allocation

– Aloca um bloco de bytes consecutivos na memória do computador e devolve o endereço desse bloco

– O número de bytes é especificado no argumento da função.



Estrutura de Dados
Memória

Malloc: exemplo de alocação de 1 byte:

```
char *ponteiro;
ponteiro = malloc(1);
scanf( "%c", ponteiro);
```

- O endereço devolvido por malloc é do tipo "genérico" **void ***.
- O programador armazena esse endereço num ponteiro de tipo apropriado.
- No exemplo acima, o endereço é armazenado num ponteiro para char.

Estrutura de Dados
Memória

Malloc: exemplo de alocação de vários bytes

- Utiliza-se o operador sizeof, que diz quantos bytes o tipo especificado tem
- Exemplo:

```
typedef struct {
    int dia, mes, ano;
} data;
data *d;
d = malloc(sizeof (data));
d->dia = 31; d->mes = 12; d->ano = 2013;
```

- **Atenção**: cuidado com overhead!!!

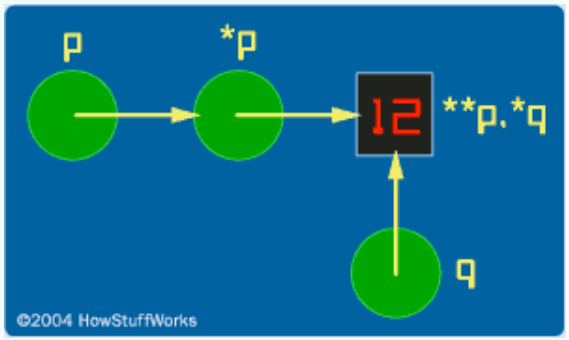
Estrutura de Dados
Memória

A Função **free**

– Variáveis estáticas declaradas em uma função, desaparecem quando a função termina

– Variáveis alocadas dinamicamente prevalecem mesmo depois que a execução da função termina

– A função ***free*** libera a porção de memória alocada por ***malloc***



– Exemplo:

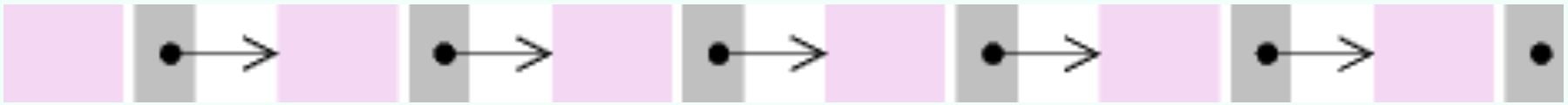
free(ptr);

ptr = NULL; //não deixar ponteiros perdidos

Estrutura de Dados
Listas

Listas encadeadas – Conceito

- representação de uma sequência de “objetos” na memória do computador.
- listas em C são implementadas através de estruturas (associadas aos nós) armazenadas na memória.
- Uma lista encadeada é uma sequência de estruturas (células) e cada uma contém um dado de algum tipo e o endereço da célula seguinte.



Estrutura de Dados
Listas

Listas encadeadas em C

– Declarando as estruturas da lista
  - Estrutura contem estrutura do tipo ponteiro e de dados

– Exemplo:

```
struct celulalista {
    int conteudo;
    struct celulalista *prox;
};
```

- Pode-se tratar as células como um novo tipo de dados e atribuir um nome a esse novo tipo

```
typedef struct celulalista celula;
```

Estrutura de Dados
Listas

Listas encadeadas em C

– Declarando células e ponteiros para células

celula c; //conteúdo da célula

celula *p; //ponteiro da célula

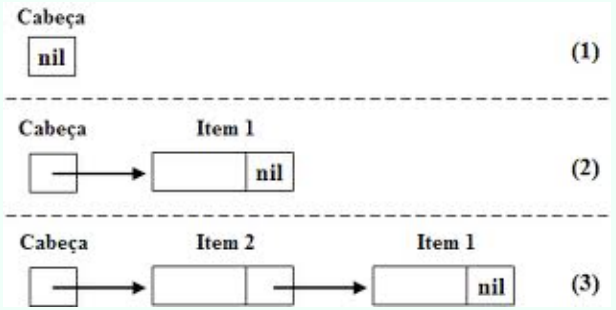
– Portanto, tem-se que:

- Se ***c*** é uma ***célula*** então c.conteudo é o conteúdo da célula e c.prox é o endereço da próxima célula
- Se ***p*** é o ***endereço*** de uma célula, então p->conteudo é o conteúdo da célula e p->prox é o endereço da próxima célula.
- Se ***p*** é o ***endereço*** da última célula da lista então temos que p->prox vale NULL .

Estrutura de Dados
Listas

Listas com cabeça

- Conteúdo da primeira célula é irrelevante
- A primeira célula é a cabeça
- A cabeça está sempre no mesmo lugar da memória mesmo que a lista esteja vazia
- Facilita a vida do programador
- Exemplo:




```
celula c, *ini;
c.prox = NULL;
ini = &c;
```

ou

```
celula *ini;
ini = malloc( sizeof (celula));
ini->prox = NULL;
```

Estrutura de Dados
Listas

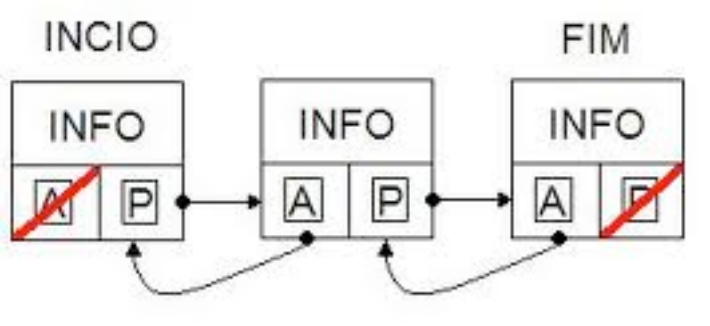
Listas sem cabeça

– conteúdo da primeira célula é tão relevante quanto o das demais

– a lista está vazia se o endereço de sua primeira célula é NULL

– Exemplo:

```
celula *ini;
ini = NULL;
```



Estrutura de Dados
Recursividade

Recursividade: conceito

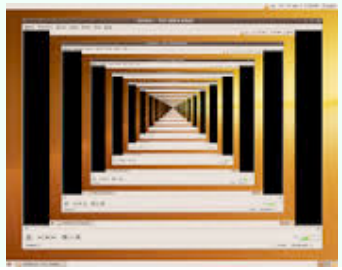
– Uma função que é dita recursiva é aquela que invoca ela mesma

– Exige-se cuidado para não provocar loop infinito

– Extremamente útil para se trabalhar com estruturas de dados árvores, grafos, listas etc.

– Problemas típicos

- programa leia um número inteiro e retorne a soma de todos os elementos de 1 até o número lido (1 + 2 + 3 + ... + n)
- Fatorial de um número: 5! (5 * 4 * 3 *2 *1)



Estrutura de Dados
Recursividade

– Exemplos de recursividade em C:

```
#include <stdio.h>
int soma(int n) {
    if(n == 1)
        return 1;
    else
        return ( n + soma(n-1) );
}
int main( ){
    int n;
    printf("Digite um inteiro positivo: ");
    scanf("%d", &n);
     printf("Soma: %d\n", soma(n));
}
```

Estrutura de Dados
Recursividade

– Exemplo de recursividade em C:

```
#include <stdio.h>
int fatorial(int n){
    if(n == 1)
        return 1;
    else
        return ( n * fatorial(n-1) );
}
int main(){
    int n;
    printf("Digite um inteiro positivo: ");
    scanf("%d", &n);
    printf("%d! = %d\n", n, fatorial(n));
}
```