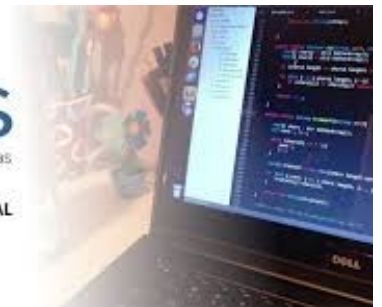


# *Tecnologia em Análise e Desenvolvimento de Sistemas - TADS*

## ***Estrutura de Dados***

***Prof. Luciano Vargas Gonçalves***

*E-mail: [luciano.goncalves@riogrande.ifrs.edu.br](mailto:luciano.goncalves@riogrande.ifrs.edu.br)*



# Estrutura da Dados

- **Aula 3 – Struct - Programação em C**

# Structs

- Estrutura estática de Dados em C
  - Uma estrutura é ***um grupo ou conjunto de itens***, no qual cada ***item*** é identificado por um identificador próprio (tipo e nome), sendo cada um deles conhecido como um **membro** da estrutura.
  - Estrutura é uma versão resumida de uma classe de OO, cada membro pode ser interpretado como um atributo de classe.
  - Os vários MEMBROS formam um Struct;
    - Uma Struct pode ser considerada um novo tipo de dados composto.

# Structs

- Estrutura de Dados em C
  - Usa o comando Struct na sua definição;
  - Necessita um nome;
  - Membros possuem tipo e nome;
  - Fazem parte do bloco da Struct { };

```
struct nome_da_estrutura{  
    tipo nome_parametro;  
    tipo nome_parametro;  
};
```

Membros da Struct

Definição

# Structs

- Estrutura de Dados em C
  - Exemplo de estrutura para armazenar os dados de uma pessoa;

```
struct nome_da_estrutura{  
    tipo nome_parametro;  
    tipo nome_parametro;  
};
```

Definição

```
struct pessoa{  
    int cod;  
    char nome [15];  
    char sobrenome [20];  
    int idade;  
    char telefone [10];  
};
```

Membros

Exemplo

# Structs – Declaração e Atribuição

- Struct pode ser vista como um novo tipo de dados definido pelo programador.
  - Um novo tipo de dados que se declara e/ou atribuir valores aos membros.
  - Forma direta, uso de chaves { }

```
//DECLARACAO E ATRIBUICAO
```

```
struct pessoa maria = {2, "Maria", "Aparecida", 23, "45433333"};
```

Novo tipo de Dados



Nome da estrutura

Valores para os membros

```
struct pessoa{  
    int cod;  
    char nome [15];  
    char sobrenome [20];  
    int idade;  
    char telefone [10];  
};
```

# Structs – Declaração e Atribuição

- Declaração separada da atribuição de valores aos membros.
  - Usa o operador “.” para acessar os membros;

```
struct pessoa joao; //DECLARAÇÃO DE STRUCT
```

```
//ATRIBUIÇÃO DE VALORES
```

```
joao.cod = 1;  
joao.idade = 30;  
strcpy(joao.nome, "Joao Carlos");  
strcpy(joao.sobrenome, "Farias");  
strcpy(joao.telefone, "1212454533");
```

```
struct pessoa{  
    int cod;  
    char nome [15];  
    char sobrenome [20];  
    int idade;  
    char telefone [10];  
};
```

Struct pessoa;  
Tipo especial

Nome.membro para atribuir valores;

# Structs

- Operador ponto “.”, usado em Struct criadas estaticamente
  - Operador ( . ) para acessar membros da Struct;
    - “**nome.membro**”

```
struct pessoa joao; //DECLARAÇÃO DE STRUCT
```

```
//ATRIBUIÇÃO DE VALORES
```

```
joao.cod = 1;
```

```
joao.idade = 30;
```

```
strcpy(joao.nome, "Joao Carlos");
```

```
strcpy(joao.sobrenome, "Farias");
```

```
strcpy(joao.telefone, "1212454533");
```

Strings

Operador “PONTO”

```
struct pessoa{  
    int cod;  
    char nome [15];  
    char sobrenome [20];  
    int idade;  
    char telefone [10];  
};
```

Struct Pessoa



# Structs

- Estrutura de Dados em C
  - Consultar valores de membros, operador PONTO “.”

```
//LEITURA DE UM STRUCT
```

```
printf("Pessoa: %s %s \n",joao.nome,joao.sobrenome);  
printf("\tCodigo: %d e idade %d \n",joao.cod,joao.idade);  
printf("\tTelefone: %s \n",joao.telefone);  
  
printf("Pessoa: %s %s \n",maria.nome,maria.sobrenome);  
printf("\tCodigo: %d e idade %d \n",maria.cod,maria.idade);  
printf("\tTelefone: %s \n",maria.telefone);
```

```
struct pessoa{  
    int cod;  
    char nome [15];  
    char sobrenome [20];  
    int idade;  
    char telefone [10];  
};
```

Struct pessoa



Operador “PONTO” para acessar um membro

# Exemplo1.c

- Exemplo completo
  - Estrutura Pessoa

```
1  #include <stdio.h>
2  struct pessoa{
3      int cod;
4      char nome [15];
5      char sobrenome [20];
6      int idade;
7      char telefone [10];
8  };
9  int main()
10 {
11     //DECLARAÇÃO DE STRUCT
12     struct pessoa joao;
13
14     //ATRIBUIÇÃO DE VALORES
15     joao.cod = 1;
16     joao.idade = 30;
17     strcpy(joao.nome,"Joao Carlos");
18     strcpy(joao.sobrenome, "Farias");
19     strcpy(joao.telefone, "1212454533");
20
21     //DECLARACAO DE OUTRA PESSOA E ATRIBUICAO
22     struct pessoa maria = {2,"Maria","Aparecida",23,"45433333"};
23
24     //LEITURA DE UM STRUCT
25     printf("Pessoa: %s %s \n",joao.nome,joao.sobrenome);
26     printf("\tCodigo: %d e idade %d \n",joao.cod,joao.idade);
27     printf("\tTelefone: %s \n\n",joao.telefone);
28
29     //SAÍDA NO TERMINA - PRINT
30     printf("Pessoa: %s %s \n",maria.nome,maria.sobrenome);
31     printf("\tCodigo: %d e idade %d \n",maria.cod,maria.idade);
32     printf("\tTelefone: %s \n\n",maria.telefone);
33     return 0;
34 }
```

# Exemplo1.c

- Exemplo completo
  - Estrutura Pessoa
  - Troca informações de Maria (idade e telefone)

```
//Para alterar o valor de um Membro
    maria.idade = 45;
    strcpy(maria.telefone , "2222333");

//SAÍDA NO TERMINA - PRINT
    printf("Pessoa: %s %s \n",maria.nome,maria.sobrenome);
    printf("\tCodigo: %d e idade %d \n",maria.cod,maria.idade);
    printf("\tTelefone: %s \n\n",maria.telefone);
    return 0;
```

Pessoa: Maria Aparecida  
Codigo: 2 e idade 23  
Telefone: 45433333



Pessoa: Maria Aparecida  
Codigo: 2 e idade 45  
Telefone: 2222333



# Comando Typedef

- Fornece um mecanismo para criação de sinônimos para novos tipos de dados;
  - Elimina o uso do comando Struct na definição do tipo;

Definição

```
typedef struct nome_struct  nome_tipo (apelido);
```

Exemplos;

```
//NOVA DEFINIÇÃO PARA PESSOA  
typedef struct pessoa Cliente;  
typedef struct pessoa Fornecedor;
```

São equivalentes (struct pessoa = Cliente )

# Comando Typedef

- Exemplo, *Struct pessoa* passa se chamar de *Cliente*;


```
1  #include <stdio.h>
2
3  struct pessoa{
4      int cod;
5      char nome [15];
6      char sobrenome [20];
7      int idade;
8      char telefone [10];
9      char tipo [10];
10 };
11 //NOVA DEFINIÇÃO PARA PESSOA
12 typedef struct pessoa Cliente;
13
14 typedef struct pessoa Fornecedor;
```

NOVOS TIPOS,  
CLIENTE E  
FORNECEDOR



# Comando Typedef

- Exemplo, Struct
- passa a ser chamada de Cliente e Fornecedor;



```
struct pessoa{
    int cod;
    char nome [15];
    char sobrenome [20];
    int idade;
    char telefone [10];
    char tipo [10];
};

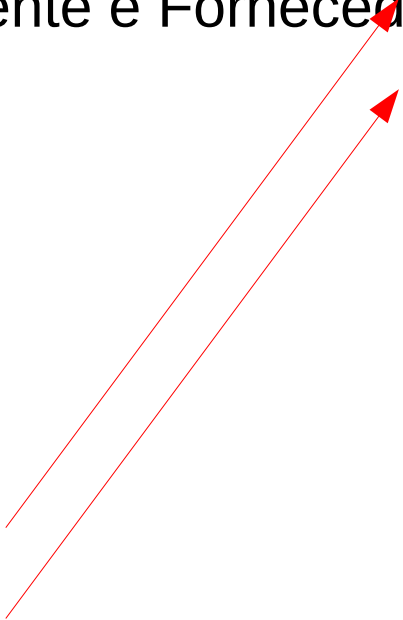
typedef struct pessoa Cliente;

typedef struct pessoa Fornecedor;
```

```
Cliente joao; //DECLARAÇÃO DE STRUCT
Fornecedor pedro; //DECLARAÇÃO DE STRUCT

//ATRIBUIÇÃO DE VALORES
joao.cod = 1;
joao.idade = 30;
strcpy(joao.nome, "Joao Carlos");
strcpy(joao.sobrenome, "Farias");
strcpy(joao.telefone, "1212454533");
strcpy(joao.tipo, "Cliente");


//ATRIBUIÇÃO DE VALORES
pedro.cod = 2;
pedro.idade = 50;
strcpy(pedro.nome, "Pedro");
strcpy(pedro.sobrenome, "Farias");
strcpy(pedro.telefone, "212454533");
strcpy(pedro.tipo, "Fornecedor");
```



# Comando Typedef

- Exemplo, **Struct pessoa** passa se chamar de **Cliente**;
  - Outra alternativa. Fazer a definição na própria estrutura.

```
3  typedef struct pessoa{
4      int cod;
5      char nome [15];
6      char sobrenome [20];
7      int idade;
8      char telefone [10];
9      char tipo [10];
10 }Cliente;
```



# Ponteiro de Struct

- Podemos criar um ponteiro para acessar uma Struct (\*p\_juca);
- Utiliza-se o operador “->” seta para referenciar o membro;

```
11 int main()
12 {
13     system("clear");
14
15     //DECLARAÇÃO De um Struct e um ponteiro para Struct
16     Pessoa juca; //propria estrutura
17     Pessoa *p_juca; //ponteiro para struct ←
18
19     //ATRIBUIÇÃO DE VALORES
20     juca.cod = 1;
21     juca.idade = 30;
22     strcpy(juca.nome, "Juca");
23     strcpy(juca.sobrenome, "Farias");
24     strcpy(juca.telefone, "1212454533");
25
26     //ponteiro recebe o endereço da estrutura
27     p_juca = &juca; ←
28
29     //p_juca->idade += 100;
30
31     //SAÍDA DOS VALORES ARMAZENADOS NA STRUCT
32     printf("\n\t Nome: %s %s \n", juca.nome, juca.sobrenome);
33     printf("\tCodigo: %d e idade %d \n", juca.cod, juca.idade);
34     printf("\tTelefone: %s \n", juca.telefone);
35
36     //SAÍDA DOS VALORES ARMAZENADOS NA STRUCT
37     printf("\n\t Nome: %s %s \n", p_juca->nome, p_juca->sobrenome);
38     printf("\tCodigo: %d e idade %d \n", p_juca->cod, p_juca->idade);
39     printf("\tTelefone: %s \n", p_juca->telefone);
40
41     return 0;
```



# Funções com Structs

- Passagem por Valor

- A cópia de um Cliente é armazenada em uma variável ***cli***
  - Mostra os dados de um cliente recebido por parâmetro;

```
void mostrarDadosCliente(Cliente cli){
```



- Passagem por Referência

- O endereço do Cliente é enviado para o ponteiro ***\*cli***
  - *Preenche os campos de um Cliente recebido por referência*

```
void lerDadosCliente(Cliente *cli, int cod){
```



# Funções com Structs

- Passagem por Valor, função mostraDadosCliente;
- Passagem por Referência, função lerDadosCliente;

```
14 void mostrarDadosCliente(Cliente cli){
15     printf("\nCliente:\n\t Nome: %s %s \n", cli.nome, cli.sobrenome);
16     printf("\tCodigo: %d e idade %d \n", cli.cod, cli.idade);
17     printf("\tTelefone: %s \n", cli.telefone);
18 }
19
20 //passagem por referência (ponteiro)
21 void lerDadosCliente(Cliente *cli, int cod){
22     cli->cod = cod;
23     printf("\nInforme seu nome:");
24     scanf("%s", cli->nome);
25     printf("\nInforme seu Sobrenome:");
26     scanf("%s", cli->sobrenome);
27     printf("\nInforme sua idade:");
28     scanf("%d", &cli->idade);
29     printf("\nInforme seu telefone:");
30     scanf("%s", cli->telefone);
31 }
```

//declaração e alocação estática  
Cliente joao, maria;

lerDadosCliente(&joao, codigo++);  
mostrarDadosCliente(joao);

lerDadosCliente(&maria, codigo++);  
mostrarDadosCliente(maria);

# Exemplo3.c

- Chamada de funções
  - Passagem por valor
  - Passagem por referência

```
1 #include <stdio.h>
2 #include <locale.h> /*permite acentuação*/
3 #include <stdlib.h>
4
5 typedef struct pessoa{
6     int cod;
7     char nome [15];
8     char sobrenome [20];
9     int idade;
10    char telefone [10];
11 }Cliente;
12
13 void mostrarDadosCliente(Cliente cli){ //PASSAGEM POR VALOR
14     printf("\nCliente:\n\t Nome: %s %s \n",cli.nome,cli.sobrenome);
15     printf("\tCodigo: %d e idade %d \n",cli.cod,cli.idade);
16     printf("\tTelefone: %s \n",cli.telefone);}
17
18 void lerDadosCliente(Cliente *cli, int cod){ //PASSAGEM POR REFERENCIA (ponteiro)
19     cli->cod = cod;
20     printf("\nInforme seu nome:");
21     scanf("%s",cli->nome);
22     printf("\nInforme seu Sobrenome:");
23     scanf("%s",cli->sobrenome);
24     printf("\nInforme sua idade:");
25     scanf("%d",&cli->idade);
26     printf("\nInforme seu telefone:");
27     scanf("%s",cli->telefone); }
28
29 int main(){
30     system("clear");
31     setlocale(LC_ALL, "");
32     int codigo = 1;
33
34     //declaração e alocação estática
35     Cliente joao, maria;
36     lerDadosCliente(&joao,codigo++);
37     mostrarDadosCliente(joao);
38     lerDadosCliente(&maria,codigo++);
39     mostrarDadosCliente(maria);
40     exit(0);
41 }
```



# Exercício 1

- Defina uma estrutura para armazenar dados de um apartamento:
  - Ex: Nome do condomínio, Numero, andar, quantidade cômodos, valor do aluguel, valor condomínio, box e etc.

```
5  typedef struct apartamento{  
6      char condomio [30];  
7      int num, andar, qtd_comodos,box;  
8      double v_aluguel, v_condominio;  
9  }Apartamento;
```

- Crie dois apartamentos, atribua valores aos seus membros;
- Crie uma função para imprimir os valores de um apartamento (passagem por valor);
  - Mostre os dados dos apartamentos criados;
- Crie uma função para alterar o valor do aluguel de um apartamento (passagem por referência);
  - Mostre os dados do apartamento com valor alterado;

# Exercício 1

- Saída do Exercício 1

Apartamento:

Residencial: Avenida Brasil N°= 101 Andar= 1 Box= 10  
Aluguel: 300.500000 e Condomínio 93.000000

Apartamento:

Residencial: Avenida Brasil N°= 201 Andar= 2 Box= 20  
Aluguel: 300.330000 e Condomínio 100.000000

Apartamento:

Residencial: Avenida Brasil N°= 101 Andar= 1 Box= 10  
Aluguel: 330.550000 e Condomínio 103.000000

—



Com aumento de 10%

# Structs com ponteiros

- **Estrutura dados usando ponteiro \***

- Alocação dinâmica – MALLOC
- MALLOC = Alocar memória para armazenamento de dados;
  - Retorna o endereço da memória alocada;
  - Sizeof → define o tamanho necessário a ser alocado.

```
//Alocação dinâmica de memória|  
paulo = (Cliente*)malloc(sizeof(Cliente));
```

Cast (tipar)

Alocação de Memória

Endereço	Valor	Nome
xxxx01		
xxxx02		
xxxx03	Novo cliente alocado	
Xxxx0..		
xxxx09		
xxxx06		

# Structs com ponteiros

- **Estrutura dados usando ponteiro \*.**

- Declarar o ponteiro para Cliente “\*paulo”;

```
//declaração de um ponteiro de Cliente
Cliente *paulo;
//inicialização do ponteiro;
paulo = NULL;

//Alocação dinâmica de memória
paulo = (Cliente*)malloc(sizeof(Cliente));
```

Endereço	Valor	Nome
xxxx01	NULL	*paulo
xxxx02		
xxxx03	Novo cliente alocado	
Xxxx0..		
xxxx09		
xxxx06		

# Structs com ponteiros

- **Estrutura dados usando ponteiro \*.**

- Declarar o ponteiro para Cliente “\*paulo”;
- Recebe o endereço da nova estrutura;

```
//declaração de um ponteiro de Cliente
Cliente *paulo;
//inicialização do ponteiro;
paulo = NULL;

//Alocação dinâmica de memória
paulo = (Cliente*)malloc(sizeof(Cliente));
```

Endereço	Valor	Nome
xxxx01	xxx03	*paulo
xxxx02		
xxxx03	Novo cliente alocado	
Xxxx0..		
xxxx09		
xxxx06		



# Exemplo4.c

- Alocação com MALLOC
  - Reserva memória e retorna um ponteiro;
- Liberação com FREE;
  - Libera memória para realocação;

```
1  ~ #include <stdio.h>
2  #include <locale.h> /*permite acentuação*/
3  #include <stdlib.h>
4
5  > typedef struct pessoa{ ...
11 }Cliente;
12
13 //passagem por Valor
14 > void mostrarDadosCliente(Cliente cli){ ...
19
20 //passagem por referência (ponteiro)
21 > void lerDadosCliente(Cliente *cli, int cod){ ...
32
33 ~ int main()
34 {
35     system("clear");
36     setlocale(LC_ALL, "");
37     int codigo = 1;
38
39     //declaração de um ponteiro de Cliente
40     Cliente *paulo;
41     //inicialização do ponteiro;
42     paulo = NULL;
43
44     //Alocação dinâmica de memória
45     paulo = (Cliente*)malloc(sizeof(Cliente));
46
47     paulo->cod = 1;
48     paulo->idade = 30;
49     strcpy(paulo->nome, "Paulo Cesar");
50     strcpy(paulo->sobrenome, "Silva");
51     strcpy(paulo->telefone, "343545443");
52
53     mostrarDadosCliente(*paulo);
54
55     //Liberar memória para reciclagem
56     free(paulo);
57     mostrarDadosCliente(*paulo);
58     exit(0);
59 }
```

## Exercício 2

- Crie dois ponteiros de apartamentos, utilize alocação dinâmica (malloc). Após, atribua valores aos campos de um dos apartamentos(301).
  - Mostre o apartamento criado (301);
- Defina uma função que recebe um apartamento e retorna o box do apartamento;
  - Mostre apenas o box do apartamento 301;
- Defina um função para criar um novo apartamento, a função recebe por parâmetro os dados do apartamento (Condomínio, andar, número, etc) e retorna o ponteiro do novo apartamento (atribua ao 401);
  - Mostre os dados do apartamento 401 criado;

# Exercício 2

Apartamento:

Residencial: Avenida Brasil N°= 301 Andar= 3 Box= 23

Aluguel: 300.330000 e Condomínio 100.000000

Box do apartamento 301 é o box = 23

Apartamento:

Residencial: Avenida Brasil N°= 401 Andar= 4 Box= 17

Aluguel: 200.000000 e Condomínio 100.000000

ch 5 14 □

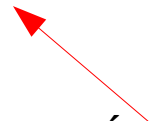
# Structs Aninhadas

- Uma estrutura é composta por outra estrutura;

- Exemplo:

- Cliente e Endereço
  - Cliente tem Endereço

```
1  #include <stdio.h>
2  #include <locale.h> /*para acentuação*/
3  #include <stdlib.h>
4
5  typedef struct endereco{
6      char nomeRua [15];
7      int numero;
8      int cep;
9  }Endereco;
10
11 typedef struct pessoa{
12     int cod;
13     char nome [15];
14     Endereco end;    //Ligação entre as estruturas
15 }Cliente;
16
```



Endereço é membro,  
declarado dentro de pessoa, e  
é reservado memória para ele

# Cliente e Endereço

- Associação entre Cliente e Endereço;
  - Cliente joao;
  - Endereço minhaCasa;



```
Cliente joao, maria; //dois clientes novos
Endereco minhaCasa;

joao.cod = 1;
strcpy(joao.nome, "João Cesar");

maria.cod = 2;
strcpy(maria.nome, "Maria Cesar");

strcpy(minhaCasa.nomeRua, "Rua 24 Maio");
minhaCasa.numero = 332;
minhaCasa.cep = 96500333;

//associação de cliente e endereco
joao.end = minhaCasa;
maria.end = minhaCasa;

mostrarDadosCliente(joao);
mostrarDadosCliente(maria);
exit(0);
```

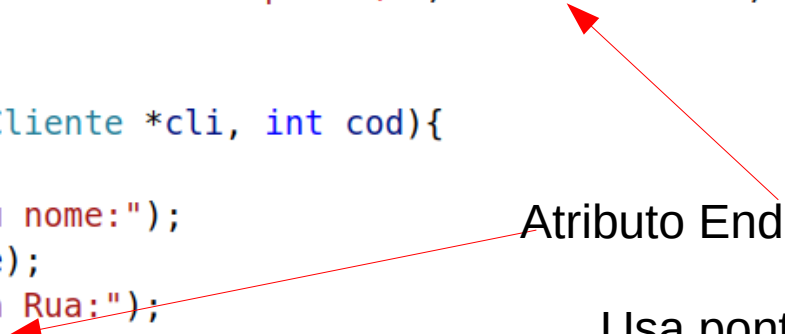
# Structs Aninhadas

- Exemplo de Cliente e Endereço

```
void mostrarDadosCliente(Cliente cli){  
    printf("Cliente: %s Código: %d \n",cli.nome,cli.cod);  
    printf("Endereço: %s Número:%d Cep:%d \n",cli.end.nomeRua,cli.end.numero,cli.end.cep);  
}
```

```
void lerDadosCliente(Cliente *cli, int cod){  
    cli->cod = cod;  
    printf("\nInforme seu nome:");  
    scanf("%s",cli->nome);  
    printf("\nInforme sua Rua:");  
    scanf("%s",cli->end.nomeRua);  
    printf("\nInforme Numero Casa:");  
    scanf("%d",&cli->end.numero);  
    printf("\nInforme seu CEP:");  
    scanf("%d",&cli->end.cep);  
}
```

Atributo End (endereço) é Parte de Cliente,  
Usa ponto para acesso aos membros de Endereço



# Exemplo5.c

- Associação Cliente e Endereço

```
1  #include <stdio.h>
2  #include <locale.h> /*para acentuação*/
3  #include <stdlib.h>
4
5  typedef struct endereco{
6      char nomeRua [15];
7      int numero;
8      int cep;
9  }Endereco;
10 typedef struct pessoa{
11     int cod;
12     char nome [15];
13     Endereco endereco; //Ligação entre as estruturas
14 }Cliente;
15 void mostrarDadosCliente(Cliente cli){
16     printf("Cliente: %s Código: %d \n",cli.nome,cli.cod);
17     printf("Endereço: %s Número:%d Cep:%d \n",cli.endereco.nomeRua,cli.endereco.numero,cli.endereco.cep);
18 }
19 void lerDadosCliente(Cliente *cli, int cod){
20     cli->cod = cod;
21     printf("\nInforme seu nome:");
22     scanf("%s",cli->nome);
23     printf("\nInforme sua Rua:");
24     scanf("%s",cli->endereco.nomeRua);
25     printf("\nInforme Numero Casa:");
26     scanf("%d",&cli->endereco.numero);
27     printf("\nInforme seu CEP:");
28     scanf("%d",&cli->endereco.cep);
29 }
30 int main(){
31     system("clear");
32     setlocale(LC_ALL, "");
33     Cliente joao, maria; //dois clientes novos
34     Endereco minhaCasa;
35     joao.cod = 1;
36     strcpy(joao.nome,"João Cesar");
37     maria.cod = 2;
38     strcpy(maria.nome,"Maria Cesar");
39     strcpy(minhaCasa.nomeRua,"Rua 24 Maio");
40     minhaCasa.numero = 332;
41     minhaCasa.cep = 96500333;
42     //associação entre cliente e endereco
43     joao.endereco = minhaCasa;
44     maria.endereco = minhaCasa;
45     mostrarDadosCliente(joao);
46     mostrarDadosCliente(maria);
47     exit(0);
48 }
```

# Exemplo5.c

- Associação Cliente e Endereço

```
1 #include <stdio.h>
2 #include <locale.h> /*para acentuação*/
3 #include <stdlib.h>
4
5 typedef struct endereco{
6     char nomeRua [15];
7     int numero;
8     int cep;
9 }Endereco;
10 typedef struct pessoa{
11     int cod;
12     char nome [15];
13     Endereco endereco; //Ligação entre as estruturas
14 }Cliente;
15 void mostrarDadosCliente(Cliente cli){
16     printf("Cliente: %s Código: %d \n",cli.nome,cli.cod);
17     printf("Endereço: %s Número:%d Cep:%d \n",cli.endereco.nomeRua,cli.endereco.numero,cli.endereco.cep);
18 }
19 void lerDadosCliente(Cliente *cli, int cod){
20     cli->cod = cod;
21     printf("\nInforme seu nome:");
22     scanf("%s",cli->nome);
23     printf("\nInforme sua Rua:");
24     scanf("%s",cli->endereco.nomeRua);
25     printf("\nInforme Numero Casa:");
26     scanf("%d",&cli->endereco.numero);
27     printf("\nInforme seu CEP:");
28     scanf("%d",&cli->endereco.cep);
29 }
30 int main(){
31     system("clear");
32     setlocale(LC_ALL, "");
33     Cliente joao, maria; //dois clientes novos
34     Endereco minhaCasa;
35     joao.cod = 1;
36     strcpy(joao.nome,"João Cesar");
37     maria.cod = 2;
38     strcpy(maria.nome,"Maria Cesar");
39     strcpy(minhaCasa.nomeRua,"Rua 24 Maio");
40     minhaCasa.numero = 332;
41     minhaCasa.cep = 96500333;
42     //associação entre cliente e endereco
43     joao.endereco = minhaCasa;
44     maria.endereco = minhaCasa;
45     mostrarDadosCliente(joao);
46     mostrarDadosCliente(maria);
47     exit(0);
48 }
```

Cliente: João Cesar Código: 1  
Endereço: Rua 24 Maio Número:332 Cep:96500333

Cliente: Maria Cesar Código: 2  
Endereço: Rua 24 Maio Número:332 Cep:96500333



# Exercício 3

- Crie a estrutura proprietário, com nome e cpf como membros;
- Adicione a estrutura Apartamento o membro, Proprietário;

```
5  typedef struct proprietario{
6      char nome [30];
7      char cpf [14];
8  }Proprietario;
9
10 typedef struct apartamento{
11     char condomio [30];
12     int num, andar, qtd_comodos,box;
13     double v_aluguel, v_condominio;
14     Proprietario dono;
15 }Apartamento;
```

# Exercício 3

- Defina dois proprietários e atribua aos apartamentos 101 e 201;
  - Mostre os dados dos apartamentos junto com os proprietários;
- Implemente a função `mostraProprietário`;
  - Mostre o proprietário de um apartamento;
- Modifique a struct `Apartamento` para que proprietário seja um ponteiro para proprietário;
  - Faça a inclusão de um mesmo proprietário a dois apartamentos

# Structs Aninhadas

- Ponteiro como Membro
  - Estrutura CASAL
    - Composta de uma data;
    - Dois ponteiros para Pessoa;
      - Marido e Esposa;

Ponteios para Struct Pessoa



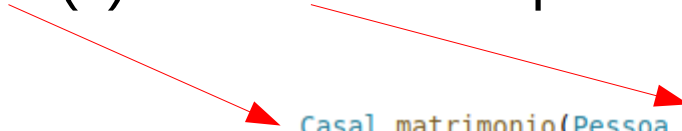
```
typedef struct {  
    char nomeRua [15];  
    int numero;  
    char cep [10];  
}Endereco;
```

```
typedef struct {  
    int cod;  
    int idade;  
    char nome [15];  
    Endereco end;  
}Pessoa;
```

```
typedef struct casal{  
    char data[10];  
    Pessoa *marido; //dois ponteiros para Pessoas  
    Pessoa *esposa;  
}Casal;
```

# Structs Aninhadas

- Função matrimônio
  - Retorna um Casal (c) e recebe duas pessoas



```
Casal matrimonio(Pessoa *p1,Pessoa *p2){
    Casal c;
    strcpy(c.data,"02/06/20");
    c.marido = p1;
    c.esposa = p2;
    return c;
}

void imprimeCertidaoCasamento(Casal cs){
    printf("\n\nCertidão de Casamento!\n");
    printf("\n Nada data %s Casaram-se neste cartório ",cs.data);
    printf("\n \t%s e %s",cs.marido->nome,cs.esposa->nome);
    printf("\n\n Dou fé a este Matrimonio!\n");
}
```

# Structs Aninhadas

- Passagem por referência
  - Pessoa &joao e &maria
- Retorno por Valor
  - Casal joaoEmaria
- Passagem por valor
  - Casal joaoEmaria

```
int main(){
    system("clear");
    setlocale(LC_ALL, "");
    Pessoa joao,maria;
    joao.cod = 1;
    joao.idade = 30;
    strcpy(joao.nome,"Joao Carlos");
    strcpy(joao.endereco.nomeRua, "Duque de caxias");
    joao.endereco.numero = 54;
    strcpy(joao.endereco.cep, "96234-234");
    maria.cod = 2;
    maria.idade = 26;
    strcpy(maria.nome,"Maria Aparecida");
    strcpy(maria.endereco.nomeRua, "Duque de caxias");
    maria.endereco.numero = 45;
    strcpy(maria.endereco.cep, "96234-234");

    //matrimonio é o relacionamento entre duas pessoas
    Casal joaoEmaria = matrimonio(&joao,&maria);
    imprimeCertidaoCasamento(joaoEmaria);
    exit(0);
}
```

# Exemplo6.c

- Exemplo de Casal
  - Com uso da função matrimônio

```
5 typedef struct {
6     char nomeRua [15];
7     int numero;
8     char cep [10];
9 }Endereco;
10 typedef struct {
11     int cod;
12     int idade;
13     char nome [15];
14     Endereco endereco;
15 }Pessoa;
16 typedef struct casal{
17     char data[10];
18     Pessoa *marido; //dois ponteiros para Pessoas
19     Pessoa *esposa;
20 }Casal;
21 Casal matrimonio(Pessoa *p1,Pessoa *p2){
22     Casal c;
23     strcpy(c.data,"02/06/20");
24     c.marido = p1;
25     c.esposa = p2;
26     return c; }
27 void imprimeCertidaoCasamento(Casal cs){
28     printf("\n\nCertidão de Casamento!\n");
29     printf("\n Nada data %s Casaram-se neste cartório ",cs.data);
30     printf("\n \t%s e %s",cs.marido->nome,cs.esposa->nome);
31     printf("\n\n Dou fé a este Matrimonio!\n"); }
32 int main(){
33     system("clear");
34     setlocale(LC_ALL, "");
35     Pessoa joao,maria;
36     joao.cod = 1;    joao.idade = 30;
37     strcpy(joao.nome,"Joao Carlos");
38     strcpy(joao.endereco.nomeRua, "Duque de caxias");
39     joao.endereco.numero = 54;
40     strcpy(joao.endereco.cep, "96234-234");
41     maria.cod = 2;    maria.idade = 26;
42     strcpy(maria.nome,"Maria Aparecida");
43     strcpy(maria.endereco.nomeRua, "Duque de caxias");
44     maria.endereco.numero = 45;
45     strcpy(maria.endereco.cep, "96234-234");
46
47     //matrimonio é o relacionamento entre duas pessoas
48     Casal joaoEmaria = matrimonio(&joao,&maria);
49     imprimeCertidaoCasamento(joaoEmaria);
50     exit(0);
51 }
```

# Exemplo7.c

- Estrutura DATA
  - Dia,mês,ano

Certidão de Casamento!

Nada data 15/3/2022 Casaram-se neste cartório

Joao Carlos e Maria Aparecida

Dou fé a este Matrimonio!

Saída no terminal

```
6 typedef struct data{
7     int dia,mes,ano;
8 }Data;
9
10 > typedef struct { ...
14 }Endereco;
15 > typedef struct { ...
20 }Pessoa;
21
22 typedef struct casal{
23     Data dataCasamento;
24     Pessoa *marido; //dois ponteiros para Pessoas
25     Pessoa *esposa;
26 }Casal;
27
28 Casal matrimonio(Pessoa *p1,Pessoa *p2, Data data){
29     Casal c;
30     c.dataCasamento = data;
31     c.marido = p1;
32     c.esposa = p2;
33     return c;
34 }
35 void imprimeCertidaoCasamento(Casal cs){
36     printf("\n\n\t\t Certidão de Casamento!\n");
37     printf("\n\t Nada data %d/%d/%d Casaram-se neste cartório\n",cs.dataCasamento.dia,cs.dataCasamento.mes,cs.dataCasamento.ano);
38     printf("\n\t\t %s e %s",cs.marido->nome,cs.esposa->nome);
39     printf("\n\n\t\t Dou fé a este Matrimonio!\n\n\n");
40 }
41 int main(){
42     system("clear");
43     setlocale(LC_ALL, "");
44     Pessoa joao,maria;
45     Data hoje = {15,03,2022} ;
46     joao.cod = 1;
47     joao.idade = 30;
48     strcpy(joao.nome,"Joao Carlos");
49     strcpy(joao.endereco.nomeRua, "Duque de caxias");
50     joao.endereco.numero = 54;
51     strcpy(joao.endereco.cep, "96234-234");
52     maria.cod = 2;
53     maria.idade = 26;
54     strcpy(maria.nome,"Maria Aparecida");
55     strcpy(maria.endereco.nomeRua, "Duque de caxias");
56     maria.endereco.numero = 45;
57     strcpy(maria.endereco.cep, "96234-234");
58     //matrimonio é o relacionamento entre duas pessoas
59     Casal joaoEmaria = matrimonio(&joao,&maria,hoje);
60     imprimeCertidaoCasamento(joaoEmaria);
61     exit(0);
62 }
```

# Exercício 4

- Defina uma Struct Data, com dia,mês e Ano;
- Insira na Struct Apartamento a data de compra;
- Crie um função para vender um apartamento, que recebe um apartamento, um ponteiro para pessoa (comprador) que se tornará o novo dono, e um data;
- Crie uma função para mostrar todos os detalhes do apartamento;

```
5  typedef struct proprietario{
6      char nome [30];
7      char cpf [14];
8  }Proprietario;
9
10 typedef struct data{
11     int dia,mes,ano;
12 }Data;
13
14 typedef struct apartamento{
15     char condomio [30];
16     int num, andar, qtd_comodos,box;
17     double v_aluguel, v_condominio;
18     Proprietario *dono;
19     Data compra;
20 }Apartamento;
```



# Array de struct

- Criando um array estático de estruturas clientes
  - vetorClientes[4]
  - Mesmo processo de array de tipos primitivos;
  - Cada Elemento do Array é uma Struct
  - Uso do operador “.” ponto;
  - Manipular o vetor é semelhante a um struct

```
int main()
{
    Cliente vetorClientes[4]; //DECLARAÇÃO DE VETOR DE STRUCT

    for(int i=0;i<4;i++){
        vetorClientes[i].cod = i+1;      //ACESSO A UM POSIÇÃO DO
        printf("\nInforme seu nome:");
        scanf("%s",vetorClientes[i].nome);
        printf("\nInforme seu Sobrenome:");
        scanf("%s",vetorClientes[i].sobrenome);
        printf("\nInforme sua idade:");
        scanf("%d",&vetorClientes[i].idade);
        printf("\nInforme seu telefone:");
        scanf("%s",vetorClientes[i].telefone);
    }
```

# Array de struct

- Criando um array de **ponteiros** para estruturas Clientes
  - Cada posição do vetor tem um ponteiro para uma struct Cliente;
  - As estruturas não foram alocadas até o momento;
    - Precisa usar o Malloc

```
Cliente *vetorClientes[4]; //DECLARAÇÃO DE UM VETOR DE PONTEIROS PARA CLIENTES
```



# Array de struct

- As estruturas não foram alocadas até o momento;
  - Precisa usar o Malloc

```
Cliente *vetorClientes[4]; //DECLARAÇÃO DE UM VETOR DE PONTEIROS PARA CLIENTES

for(int i=0;i<4;i++){
    //reserva memória para cada cliente
    vetorClientes[i] = (Cliente*)malloc(sizeof(Cliente));
}
```



# Exemplo8.c

- Vetor de ponteiros \*

```
1  #include <stdio.h>
2
3  typedef struct pessoa{
4      int cod;
5      char nome [15];
6      char sobrenome [20];
7      int idade;
8      char telefone [10];
9  }Cliente;
10
11 int main()
12 {
13     Cliente *vetorClientes[4]; //DECLARAÇÃO DE UM VETOR DE PONTEIROS PARA CLIENTES
14
15     for(int i=0;i<4;i++){
16         //reserva memória para cada cliente
17         vetorClientes[i] = (Cliente*)malloc(sizeof(Cliente));
18         vetorClientes[i]->cod = i+1; //Entrada de Dados com ponteiros
19         printf("\nInforme seu nome:");
20         scanf("%s",vetorClientes[i]->nome);
21         printf("\nInforme seu Sobrenome:");
22         scanf("%s",vetorClientes[i]->sobrenome);
23         printf("\nInforme sua idade:");
24         scanf("%d",&vetorClientes[i]->idade);
25         printf("\nInforme seu telefone:");
26         scanf("%s",vetorClientes[i]->telefone);
27     }
28
29     Cliente *p;
30
31     //LEITURA DE UM STRUCT
32     for(int i=0;i<4;i++){
33         p = vetorClientes[i]; //copia o endereço da struct para o ponteiro
34         printf("Cliente: %s %s \n",p->nome,p->sobrenome);
35         printf("\tCodigo: %d e idade %d \n",p->cod,p->idade);
36         printf("\tTelefone: %s \n",p->telefone);
37     }
38
39     return 0;
40 }
```

# Exemplo8B.c

- \*\*Vcli é um ponteiro de ponteiros
  - Equivale a um vetor de ponteiros
  - Uso dos parênteses para acesso aos elementos
    - (ponteiro) →
    - (\*(vcli+i)) →
      - Acesso aos elementos
      - Acesso por referência

```
1  #include <stdio.h>
2
3  typedef struct pessoa{
4      int cod;
5      char nome [15];
6      char sobrenome [20];
7      int idade;
8      char telefone [10];
9  }Cliente;
10
11 int main()
12 {
13     Cliente **vcli, *p; //DECLARAÇÃO DE um vetor de ponteiros (PONTEIRO DE PONTEIROS)
14     vcli = (Cliente*)malloc(sizeof(Cliente)*4);
15
16     for(int i=0;i<2;i++){
17         //reserva memória para cada cliente
18         vcli[i] = (Cliente*)malloc(sizeof(Cliente));
19         (*(vcli+i))->cod = i+1; //Entrada de Dados com ponteiros
20         p = *(vcli+i);
21
22         printf("\nInforme seu nome:");
23         scanf("%s",p->nome);
24         printf("\nInforme seu Sobrenome:");
25         scanf("%s",p->sobrenome);
26         printf("\nInforme sua idade:");
27         scanf("%d",&p->idade);
28         printf("\nInforme seu telefone:");
29         scanf("%s",p->telefone);
30     }
31
32     //LEITURA DE UM STRUCT
33     for(int i=0;i<2;i++){
34         p = *(vcli+i);
35         printf("Cliente: %s %s \n",p->nome,p->sobrenome);
36         printf("\tCodigo: %d e idade %d \n",p->cod,p->idade);
37         printf("\tTelefone: %s \n",p->telefone);
38     }
39
40     return 0;
41 }
```

# Exercício 5

- Crie um bloco de apartamentos com 4 apartamentos (vetor de apartamentos), defina os atributos de cada apartamento;
- Imprima o bloco de apartamentos com todos os dados dos apartamentos;
- Associe com a estrutura apartamento, um proprietário (pessoa) com a data de compra. Defina um mesmo proprietário para mais de um apartamento;
- Crie um condomínio, com 3 blocos de apartamentos, cada bloco com 4 apartamentos;
- Complete e Imprima as informações de todos os blocos do condomínio, associando proprietário e apartamento.



Dúvidas ??