

## Estrutura de dados heterogênea - Registros



# Retomando...

- Strings
  - Entrada de dados
    - scanf();
    - gets();
  - Saída de informações
    - printf();
    - puts();
- 
- Funções específicas para strings:
  - strcpy, strcmp, strcat, strlen, etc..

# Registros

- Registro: novo tipo de variável;
- Coleção de valores (campos) agrupados em um nome;
- Suporta diferentes tipos de dados;
- Pode ser visto como um vetor onde cada campo corresponde a um índice.

Registros representam conjunto de informações de tipos diferentes e são representados por uma variável.

# Sintaxe C



Sistemas para Internet  
UFSM

```
struct nome_da_estrutura
{
    lista de campos;
};

int main()
{
    struct nome_da_estrutura VARIÁVEL;
}
```

- Onde:
  - **struct**: define que será declarado um **registro** com diversos campos.
  - **nome\_da\_estrutura**: nome dado à estrutura.



# Exemplo: linguagem C

Ler código, nome, quantidade e preço de um produto de uma ferragem.

**3 passos:**

```
struct estoque ← 1º: Criar o registro
{
    int Codigo;
    char Produto[50];
    int Qtd;
    float Preço;
};

int main()
{
    struct estoque item; ← 3º: Declarar variável para
                             acessar o registro.
}
```

2º: Informar campos que compõem o registro.



# Acesso e leitura do registro

```
struct estoque
{
    int Codigo;
    char Produto[50];
    int Qtd;
    float Preco;
};

int main ()
{
    struct estoque item;
    scanf ("%d", &item.Codigo);
    gets (item.Produto);
    scanf ("%d", &item.Qtd);
    scanf ("%f", &item.Preco);
    return 0;
}
```



# Acesso e leitura do registro

```
struct estoque {
    intCodigo;
    char Produto[50];
    int Qtd;
    float Preco;
};

int main () {
    struct estoque item;
    scanf ("%d", &item.Codigo);
    gets (item.Produto);
    scanf ("%d", &item.Qtd);
    scanf ("%f", &item.Preco);
    printf ("%d", item.Codigo);
    puts (item.Produto);
    printf ("%d", item.Qtd);
    printf ("%f", item.Preco);
    return 0;
}
```



# Exercício: Notas de aluno

- Desenvolva um algoritmo com um registro que contenha o nome do **aluno**, **nota 1**, **nota 2**, e **média final**. Faça a leitura dos valores dos campos e apresente ao usuário.

```
struct disciplina{
    char nome[20];
    float nota1, nota2;
    float media;
};

int main () {
    struct disciplina aluno;
    gets (aluno.nome);
    scanf ("%f %f", &aluno.nota1, &aluno.nota2);
    aluno.media = (aluno.nota1 + aluno.nota2)/2;
    puts (aluno.nome);
    printf ("%f", aluno.media);
    return 0;
}
```



# Conjunto de registros

- Funciona com as mesmas regras de um vetor.
  - Ex: Ler código, nome, quantidade e preço de 5 produtos de uma ferragem.

```
struct ferragem
{
    int Codigo;
    char Produto[50];
    int Qtd;
    float Preco;
};

int main ()
{
    struct ferragem item[5];
    ...
}
```



# Exemplo: ferragem

Ler **código**, **nome**, **quantidade** e **preço** de 5 produtos de uma ferragem e depois mostrar seus valores ao usuário.

```
int main (){
    int i;
    for(i=0; i<5; i++){
        scanf("%d", &item[i].Codigo);
        gets(item[i].Produto);
        scanf("%d", &item[i].Qtd);
        scanf("%f", &item[i].Preco);
    }

    for(i=0; i<5; i++){
        printf("%d", item[i].Codigo);
        puts(item[i].Produto);
        printf("%d", item[i].Qtd);
        printf("%f", item[i].Preco);
    }
    ...
}
```

# Exercício

- Desenvolver um algoritmo para cadastrar informações de 5 produtos de uma loja de conveniências de um posto de combustível.
- Cada produto deve conter um código, nome, preço normal e preço para estudante (metade do preço normal, que deve ser calculado pelo programa).
- Depois apresentar os produtos na tela.



# Solução

```
struct controle{
    int codigo;
    char nome[30];
    float prNormal;
    float prEstudante;
};

int main(){
    int k;
    struct controle posto[5];
    for(k=0; k<5; k++){
        scanf("%d",&posto[k].codigo);
        gets(posto[k].nome);
        scanf("%f", &posto[k].prNormal);
        posto[k].prEstudante = posto[k].prNormal / 2;
    }

    for(k=0; k<5; k++){
        printf("%d",posto[k].codigo);
        puts(posto[k].nome);
        printf("%.2f", posto[k].prNormal);
        printf("%.2f", posto[k].prEstudante);
    }
    return 0;
}
```

# Declaração de structs: formas alternativas

- Definir um novo nome para a struct;
- Declarar a variável na própria struct.



# Formas alternativas: criar novo nome (1)

Ler código, nome, quantidade e preço de um produto de uma ferragem.

```
struct estoque ← 1º: Criar o registro
{
    int Codigo;
    char Produto[50];
    int Qtd;
    float Preco;
};
```

2º: Informar campos que compõem o registro.

**Esta técnica pode ser aplicada a qualquer tipo de dados**

```
typedef struct estoque Controle;
```

```
int main() {
    Controle item; ← 3º: Declarar variável para acessar o registro.
}
```



## Formas alternativas: criar novo nome (2)

Ler código, nome, quantidade e preço de um produto de uma ferragem.

```
typedef struct
```

```
{
```

```
    int Codigo;
```

```
    char Produto[50];
```

```
    int Qtd;
```

```
    float Preco;
```

```
} estoque;
```

1º: Criar o registro com seu nome ao final da estrutura.

2º: Informar campos que compõem o registro.

```
int main() {
```

```
    estoque item;
```

```
}
```

3º: Declarar variável para acessar o registro.

# Declarar variável na struct (variável global)



Sistemas para Internet  
UFSM

Ler código, nome, quantidade e preço de um produto de uma ferragem.

**3 passos:**

```
struct estoque ← 1º: Criar o registro
{
    int Codigo;
    char Produto[50];
    int Qtd;
    float Preço;
} Item;
↑
3º: Declarar variável para acessar o registro.
```

2º: Informar campos que compõem o registro.



# Criar vetor de structs

Ler código, nome, quantidade e preço de 5 produtos de uma ferragem.

**3 passos:**

```
struct estoque ← 1º: Criar o registro
{
    int Codigo;
    char Produto[50];
    int Qtd;
    float Preço;
} Item[5];
```

2º: Informar campos que compõem o registro.

3º: Declarar variável para acessar o registro.



# Passagem de parâmetros

```
1 struct disciplina{
2     char nome[20];
3     float nota1, nota2;
4     float media;
5 };
6
7 typedef struct disciplina disc;
8
9 void registros (disc al) ←
10 {
11     al.media = (al.nota1 + al.nota2)/2;
12     puts(al.nome);
13     printf ("%f", al.media);
14 }
15
16 int main () {
17     disc aluno;
18     gets (aluno.nome);
19     scanf ("%f %f", &aluno.nota1, &aluno.nota2);
20     registros(aluno);
21     return 0;
22 }
```

# Passagem de parâmetros: vetor de structs



Sistemas para Internet  
UFSM

```
4 struct disciplina{
5     char nome[20];
6     float nota1, nota2;
7     float media;
8 };
9
10 typedef struct disciplina disc;
11
12 void registros (disc al[2]) ←
13 {
14     int i;
15     for (i=0; i<2; i++)
16     {
17         al[i].media = (al[i].nota1 + al[i].nota2)/2;
18         puts(al[i].nome);
19         printf ("%f \n", al[i].media);
20     }
21 }
22
23 int main () {
24     disc aluno[2];
25     int i;
26     for (i=0; i<2; i++)
27     {
28         fflush(stdin);
29         gets (aluno[i].nome);
30         scanf ("%f %f", &aluno[i].nota1, &aluno[i].nota2);
31     }
32     registros(aluno);
33 }
```



# Retorno de struct

```
4 struct disciplina{
5     char nome[20];
6     float nota1, nota2;
7     float media;
8 };
9
10 typedef struct disciplina disc;
11
12 disc registros () {
13     disc al;
14     gets (al.nome);
15     scanf ("%f %f", &al.nota1, &al.nota2);
16     return al;
17 }
18
19 int main () {
20     disc aluno;
21     int i;
22     aluno = registros();
23     aluno.media = (aluno.nota1 + aluno.nota2)/2;
24     puts(aluno.nome);
25     printf("%.2f", aluno.media);
26     return 0;
27 }
```



# Registros aninhados

- Refere-se a técnica de criar um registro onde, dentre seus campos, existe um que é do tipo struct;
- Exemplo:
  - Criar um registro para guardar os seguintes dados de um livro: título, ano de publicação e número de páginas;
  - Depois, criar outro registro para guardar os dados de um leitor (nome e idade) e também as informações sobre o livro que ele está lendo.

```
struct livro{  
    char titulo[50];  
    int paginas;  
    int ano;  
};  
struct biblioteca{  
    char nome[50];  
    int idade;  
    struct livro exemplar;  
};
```



# Registros aninhados

- Como acessar os campos dos registros:

- Através da criação de variáveis:

```
struct livro exemplar;  
gets(exemplar.titulo);  
scanf("%d", &exemplar.paginas);  
scanf("%d", &exemplar.ano);
```

- Neste caso, a variável “exemplar” possui apenas os campos definidos para o registro livro, sem relação nenhuma com os campos do leitor definidos em biblioteca.

```
struct livro{  
    char titulo[50];  
    int paginas;  
    int ano;  
};  
struct biblioteca{  
    char nome[50];  
    int idade;  
    struct livro exemplar;  
};
```



# Registros aninhados

- Como acessar os campos dos registros:

- Através da criação de variáveis:

```
struct biblioteca leitor;  
gets(leitor.nome);  
scanf("%d", &leitor.idade);  
gets(leitor.exemplar.titulo);  
scanf("%d", &leitor.exemplar.paginas);  
scanf("%d", &leitor.exemplar.ano);
```

```
struct livro{  
    char titulo[50];  
    int paginas;  
    int ano;  
};  
struct biblioteca{  
    char nome[50];  
    int idade;  
    struct livro exemplar;  
};
```

- Neste caso, é possível acessar os campos do registro livro através de uma variável do tipo struct biblioteca uma vez que um dos seus campos refere-se a livro.



# Registros aninhados

- Exemplo completo:

```
struct livro{
    char titulo[50];
    int paginas;
    int ano;
};
struct biblioteca{
    char nome[50];
    int idade;
    struct livro exemplar;
};
int main(){
    struct biblioteca leitor;
    gets(leitor.nome);
    scanf("%d", &leitor.idade);
    fflush(stdin);
    gets(leitor.exemplar.titulo);
    scanf("%d", &leitor.exemplar.paginas);
    scanf("%d", &leitor.exemplar.ano);

    printf("\nDados do livro: \n");
    puts(leitor.nome);
    printf("%d\n", leitor.idade);
    puts(leitor.exemplar.titulo);
    printf("%d\n", leitor.exemplar.paginas);
    printf("%d\n", leitor.exemplar.ano);
    return 0;
}
```