

Aula 10:

Estruturas e Tipos definidos pelo usuário

Disciplina: Fundamentos de Programação

Prof. Luiz Olmes

olmes@unifei.edu.br



Nas aulas anteriores...

▶ **O QUE JÁ ESTUDAMOS?**

- ▶ Algoritmos.
- ▶ Linguagem C.
- ▶ Variáveis, operadores e tipos.
- ▶ Estruturas de controle condicionais.
- ▶ Estruturas de controle iterativas.
- ▶ Vetores.
- ▶ Matrizes.
- ▶ Strings.

▶ **OBJETIVOS:**

- ▶ Estruturas
- ▶ Comando typedef
- ▶ Vetores de estruturas

Estruturas

- ▶ Os tipos de dados da linguagem C podem ser classificados em duas categorias:
 - ▶ Tipos básicos: `char`, `double`, `float`, `int` e `void`.
 - ▶ Tipos compostos homogêneos: *arrays*.
- ▶ Dependendo da situação que deseja-se modelar num programa, esses tipos podem não ser suficientes.
 - ▶ Algumas situações exigem que tipos distintos sejam agregados sob uma mesma variável. Exemplo: um tipo `Aluno`, composto por nome e matrícula.
- ▶ A linguagem C permite criar novos tipos a partir de seus tipos básicos.
- ▶ Uma das formas é através do comando `struct`.

Definição: struct

- ▶ Uma **struct** é um **conjunto de variáveis sob o mesmo nome**, onde cada variável pode possuir qualquer tipo.
- ▶ A ideia de uma **struct** é criar um **tipo de dado** que contenha vários **membros**, que, por sua vez, são **outras variáveis**.
- ▶ Ou seja, uma **struct** cria uma variável que contém outras variáveis dentro de si.
- ▶ **structs** são usadas para definir **registros**, que serão armazenados em arquivos, e na definição de nós de **estruturas de dados** (listas, filas, pilhas, árvores, etc.).

Definição: struct

► Declaração:

```
typedef struct nome_da_struct
{
    tipo variavel1;
    ...
    tipo variavelN;
} nome_da_struct;
```

Definição: struct

► Declaração:

Definição da estrutura

Rótulo (nome) da estrutura

```
typedef struct nome_da_struct  
{  
    tipo variavel1;  
    ...  
    tipo variavelN;  
} nome_da_struct;
```

Variáveis declaradas dentro da estrutura: membros

Identificador utilizado para nomear variáveis do tipo da estrutura.

Termina com ponto e vírgula.

Declarando uma struct

- ▶ A struct a seguir representa o cadastro de uma pessoa:

```
typedef struct pessoa
{
    char nome[50];
    int idade;
    char endereco[100];
} pessoa;
```

Declarando uma struct

- ▶ A struct a seguir representa o cadastro de uma pessoa:

```
typedef struct pessoa
{
    char nome[50];
    int idade;
    char endereco[100];
} pessoa;
```

Informações armazenadas
sobre pessoas.

Declarando uma struct

- ▶ A struct a seguir representa o cadastro de uma pessoa:

```
typedef struct pessoa
{
    char nome[50];
    int idade;
    char endereco[100];
} pessoa;
```

Informações armazenadas
sobre pessoas.

- ▶ A struct a seguir representa as cartas de um baralho (espanhol¹):

```
typedef struct carta
{
    char valor; // A, 2, 3, 4, 5, ..., J, Q, K
    char naipe[7]; // paus, copas, espada, ouro
} carta;
```

Declarando uma struct

- ▶ A struct a seguir representa o cadastro de uma pessoa:

```
typedef struct pessoa
{
    char nome[50];
    int idade;
    char endereco[100];
} pessoa;
```

Informações armazenadas
sobre pessoas.

- ▶ A struct a seguir representa as cartas de um baralho (espanhol):

```
typedef struct carta
{
    char valor; // A, 2, 3, 4, 5, ..., J, Q, K
    char naipe[7]; // paus, copas, espada, ouro
} carta;
```

Informações
armazenadas sobre
as cartas.

Declarando variáveis de uma struct

- ▶ A definição de uma **struct** é normalmente realizada em **escopo global**:
 - ▶ Entre os `#includes` e a função principal.
 - ▶ Por ser um tipo de dado definido pelo programador, todas as partes do código devem ter acesso a este tipo.
- ▶ A definição de uma estrutura **não** aloca espaço em memória.
- ▶ Cada definição cria um novo tipo de dado que pode ser usado para definir variáveis.
- ▶ Variáveis de structs são definidas do mesmo modo que as variáveis de outros tipos.

Declarando variáveis de uma struct

```
1. #include...
2.
3. typedef struct carta
4. {
5.     char valor; // A, 2, 3, 4, 5, ..., J, Q, K
6.     char naipe[7]; // paus, copas, espada, ouro
7. } carta;
8.
9. int main()
10. {
11.     carta c1, c2;
12.     ...
13.     return 0;
14. }
```

Escopo global

Duas variáveis do tipo carta: cada uma possui um valor para face e naipe.

Instrução typedef

- ▶ A linguagem C permite que o programador defina tipos de dados com base em outros tipos já existentes através do comando **typedef**.

Instrução typedef

- ▶ A linguagem C permite que o programador defina tipos de dados com base em outros tipos já existentes através do comando **typedef**.

Definição da struct

```
typedef struct carta  
{  
    char valor;  
    char naipe[7];  
} carta;
```

typedef: cria um tipo chamado carta a partir da definição da struct

Instrução typedef

- ▶ A linguagem C permite que o programador defina tipos de dados com base em outros tipos já existentes através do comando **typedef**.
- ▶ Declarar a struct **sem** o typedef faz com seja necessário repetir a palavra struct na criação de variáveis do tipo da struct.

```
1. #include...
2.
3. struct carta
4. {
5.     char valor;
6.     char naipe[7];
7. };
8.
9. int main()
10. {
11.     struct carta c1, c2;
12.     ...
13.     return 0;
14. }
```

Instrução typedef

- ▶ O rótulo da struct e o identificador do typedef não precisam ser os mesmos.

```
typedef struct card  
{  
    char valor;  
    char naipe[7];  
} carta;
```

Rótulo

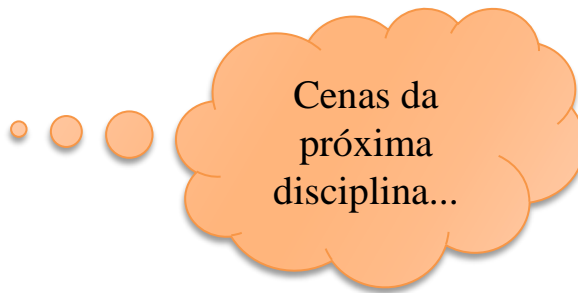
Identificador

Instrução typedef

- ▶ O rótulo da struct e o identificador do typedef não precisam ser os mesmos.

```
typedef struct card
{
    char valor;
    char naipe[7];
} carta;
```

```
int main()
{
    carta c1;
    struct card c2;
    ...
}
```



Cenas da
próxima
disciplina...

Acesso aos campos da struct

- ▶ O operador de acesso aos campos da struct é o operador **ponto** (`.`).
- ▶ Ele é usado para referenciar os membros (variáveis) de uma estrutura.
- ▶ Sintaxe:

`<nome_da_variável_do_tipo_struct>.<nome_do_campo>`

- ▶ Exemplo:

```
carta c;  
c.valor = 'A';  
strcpy(c.naipes, "copas");
```

```
scanf("%c", &c.valor);  
scanf("%s", c.naipes);  
printf("%s", c.naipes);
```

Estruturas

- ▶ **Exemplo 1:** criar uma estrutura para representar pontos no espaço. Cada ponto é definido pelas coordenadas X, Y e Z. A seguir, leia dois pontos e mostre a distância entre eles.
- ▶ **Exemplo 2:** criar uma estrutura para armazenar os dados de veículos (placa, marca, modelo e cor). A seguir, leia os dados de dois carros e informe se eles são da mesma montadora e se são da mesma cor.

Vetores de struct

- ▶ No exemplo da `struct carta`, considerando que um baralho possui 52 cartas, pode-se fazer a seguinte declaração:

```
int main()
{
    carta c1, c2, ..., c51, c52;
    ...
}
```

Vetores de struct

- ▶ No exemplo da **struct carta**, considerando que um baralho possui 52 cartas, pode-se fazer a seguinte declaração:

```
int main()
{
    carta c1, c2, ..., c51, c52;
    ...
}
```

- ▶ Porém, a representação dessas 52 cartas pode ser simplificada usando um **vetor de struct**:

```
int main()
{
    carta c[52];
    ...
}
```

Vetores de struct: exemplo (ler e imprimir)

```
1. #include <stdio.h>
2.
3. typedef struct carta
4. {
5.     char valor;
6.     char naipe[7];
7. } carta;
8.
9. int main()
10. {
11.     carta c[52];
12.     int i;
13.
14.     for (i = 0; i < 52; i++)
15.     {
16.         scanf("%c", &c[i].valor);
17.         scanf("%s", c[i].naipe);
18.     } // Fim for
19.
20.     for (i = 0; i < 52; i++)
21.     {
22.         printf("%c \t", c[i].valor);
23.         printf("%s \n", c[i].naipe);
24.     }
25.
26.     return 0;
27. } // Fim main
```

Estruturas

- ▶ **Exemplo 3:** criar uma estrutura para representar alunos, caracterizados por matrícula e nome. Leia os dados de 5 alunos em um vetor da estrutura. Apresenta um menu ao usuário com as seguintes opções: 1 – pesquisar aluno por matrícula. 2 – pesquisar aluno por nome. 9 – sair. Seu programa deve permanecer em loop até que o usuário saia com a opção 9.
- ▶ **Exemplo 4:** criar uma estrutura para representar cartas de um baralho (espanhol). A seguir, gere o baralho completo e mostre-o na tela. Por fim, embaralhe as cartas e apresente o resultado na tela.

Dúvidas?



Aula 10:

Estruturas e Tipos definidos pelo usuário

Disciplina: Fundamentos de Programação

Prof. Luiz Olmes

olmes@unifei.edu.br

