

Felipe Cunha

Conceito/definição

- São coleções de variáveis relacionadas agrupadas sob um único nome.
- Podem conter variáveis de muitos tipos de dados diferentes.
- São usadas para declarar registros a serem armazenados em arquivo.
- Estruturas de dados mais complexas: listas, filas, pilhas e árvores.
- Ponteiros e estruturas facilitam a formação dessas estruturas.
- Tipos de dados derivados (e não primitivos)

```
struct card {  
char * face;  char *suit;  
};
```

- STRUCT é a palavra chave para estrutura;
- TAG DE ESTRUTURA: neste caso é a palavra CARD. A Tag de Estrutura é usada para declarar variáveis desse tipo de estrutura, é o identificador deste tipo de estrutura.
- As variáveis dentro da estrutura chamam-se MEMBROS.

- Obs.: dois tipos de estruturas diferentes podem ter membros com o mesmo nome;
- Estruturas podem ter vários tipos de dados diferentes.

Exemplo:

```
struct funcionario {  
    char nome[20];  
    char sobrenome[20];    int idade;  
    char sexo;    double salario;  
};
```

Estrutura Autorreferenciada

- É uma estrutura que contém um membro que é um ponteiro para o mesmo tipo de estrutura. São usadas para criar listas interligadas.

Exemplo:

```
struct funcionario2 {  
    char nome[20];  
    char sobrenome[20]; int idade;  
    char sexo; double salario;  
    struct funcionario2 *ePtr; //estrutura autorreferenciada  
}
```

Declaração de variáveis

- Declarações de estruturas não criam espaço na memória;
- Declarações de estruturas criam um novo tipo de dado;
- Esse novo tipo de dado é usado para declarar variáveis que então ocupam espaço na memória;

```
struct card aCard, deck[52], *carPtr; ou
```

```
struct card {  
    char *face; char *suit;  
} aCard, deck[52], *carPtr;
```

aCard: é uma variável do tipo struct card;

deck: é um vetor com 52 posições do tipo struct card;

carPtr: é um ponteiro para struct card;

- **Operações válidas:**
 - Atribuição de variáveis da estrutura a variáveis da estrutura de mesmo tipo;
 - Coleta de endereço de uma variável de estrutura (operador &);
 - Acesso aos membros de uma variável de estrutura;
 - Uso do operador **sizeof** para determinar o tamanho de uma variável de estrutura.
-
- **NÃO PODEMOS:**
 - Comparar estruturas usando `==` e `!=`

- Parecido com vetores e matrizes.
- Exemplo:

```
struct card aCard = { "Três", "Copas" } ;
```

- Inicializa o membro da estrutura FACE com o valor TRÊS e o membro SUIT com o valor COPAS.
- Atenção: se o número de inicializadores na lista for menor que os membros na estrutura, os membros restantes serão automaticamente inicializados em zero, ou NULL se o membro for um ponteiro.

Acesso aos membros

- Dois operadores são usados para acessar os membros de uma estrutura:
- Operador de membro de estrutura ou operador de ponto (.) é um ponto de fato!
- Operador de ponteiro de estrutura ou operador de seta (->) é uma seta de fato!

Exemplo 1: `printf("%s", aCard.suit);`

Exemplo 2: `printf("%s", cardPtr->suit);`

- Ambos imprimem na tela o conteúdo de SUIT;

Acesso aos membros

```
/* Usando operadores de membro da
estrutura e de ponteiro da estrutura */
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

/* Declaração da estrutura da carta */
struct card {
    char *face; //ponteiro
    char * suit; //ponteiro
}; //fim da estrutura
```

Acesso aos membros

```
int main(void) {  
    struct card aCard; //declaração de variável  
    struct card *cardPtr; //declaração de ponteiro  
    //coloca strings em aCard  
    aCard.face = "Ás";  
    aCard.suit = "Espadas";  
    //atribui o endereço de aCard a cardPtr  
    cardPtr = &aCard;  
    printf(" %s%s%s \n %s%s%s \n %s%s%s \n ", aCard.face, "  
de ", aCard.suit, cardPtr->face, " de ", cardPtr->suit,  
(*cardPtr).face, " de ", (*cardPtr).suit);  
    return 0;  
}
```

Uso de Estruturas

- As estruturas podem ser passadas a funções ao:
 - Passar membros da estrutura individuais
 - Passar uma estrutura inteira
 - Passar um ponteiro para uma estrutura
- Quando as estruturas ou membros individuais da estrutura são passados a uma função, eles são passados por valor
- Os membros das estruturas passados por valor não podem ser modificados pela função utilizada
- Para passar uma estrutura por referência:
 - Passe o endereço da variável da estrutura

Uso de Estruturas

- Um array pode ser passado por valor usando uma estrutura;
- Para isto faça:
 - Crie uma estrutura que tenha o array como membro;
 - Estruturas são passadas por valor, de modo que o array também é passado por valor;
- ERRO LÓGICO: supor que estruturas como arrays sejam passadas automaticamente por referência e tentar modificar os valores da estrutura passadas por valor na função utilizada;
- DICA: passar estruturas por referência é mais eficiente do que passar estruturas por valor;

- **TYPEDEF:** oferece um mecanismo de criação de sinônimos para tipos de dados previamente definidos;
- Usado para definir um tipo da estrutura, de modo que a tag da estrutura não é necessária;

- **Exemplo 1:**

```
typedef struct card Card;
```

- **Exemplo 2:**

```
typedef struct {  
    char *face;  char *suit;  
} Card;
```

Exemplo 1

```
/* Cria uma estrutura para armazenar dados de um aluno */  
#include <stdio.h>  
  
#include <stdlib.h>  
  
struct aluno {  
    int nmat; //número da matrícula  
    float nota[3]; //notas  
    float media; //média  
};
```

Exemplo 1

```
int main () {  
    struct Aluno Jose; //declara uma variável do tipo struct  
    Jose.nmat = 456;  
    Jose.nota[0] = 7.5;  
    Jose.nota[1] = 5.2;  
    Jose.nota[2] = 8.4;  
    Jose.media = (Jose.nota[0] + Jose.nota[1] +  
Jose.nota[2]) / 3.0;  
    printf("Matrícula: %d \n ", Jose.nmat);  
    printf("Média: %2f \n ", Jose.media);  
    return 0;  
}
```


The image features a classic 'The End' title card. It consists of several concentric red circles that create a tunnel-like effect, leading to a solid black circle in the center. The words 'The End' are written in a white, cursive script font across the black circle. The 'The' is on the left and 'End' is on the right, with the black circle acting as the letter 'E' in 'End'.

The End