Aula 11 – Oficina de Programação
Estruturas

Profa. Elaine Faria

UFU - 2017

Estrutura de Dados

- Muitas vezes precisamos compor os dados para formar estruturas de dados complexas
- Variáveis compostas homogêneas (Arrays)
  - Conjunto de variáveis de mesmo tipo
- Variáveis compostas heterogêneas
  - Conjunto de variáveis de tipos diferentes
- Chamadas de:
  - **Estruturas (Struct)**
  - **Registros (Record)**

Aplicação de Estruturas (1)

- Estruturas podem ser usadas para armazenar informações relacionadas
- Exemplo 1: Produto

Livro (char[11])	L	i	n	g	u	a	g	e	m		C
Preco (float)	59,9000										
Autor (char[11])	D	.		R	i	t	c	h	i	e	

Aplicação de Estruturas (2)

- Exemplo 2: Ficha de cliente (cadastro)

Nome (char[10])	H	e	l	e	n	a				
Idade (int)	30									
Telefone (int)	5555-5555									
Cidade (char[10])	S	a	o		P	a	u	l	o	

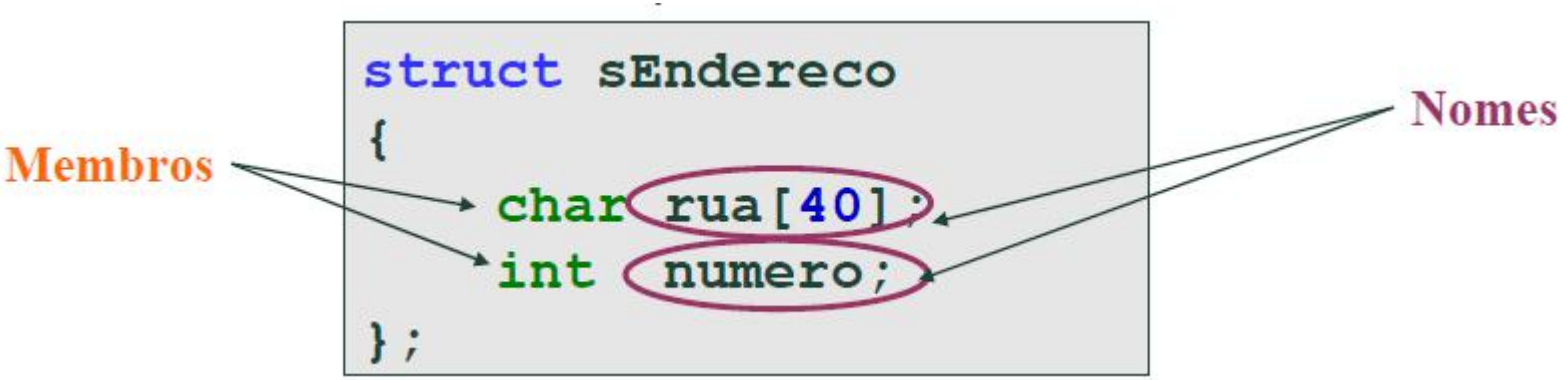
Definição de uma estrutura (registro) em C

```
struct identificacao_da_estrutura
{
    tipo1 nome1;
    tipo2 nome2;
    ...
    tipoN nomeN;
} ;
```

- Uma estrutura é um tipo de dado cujo formato é definido pelo programador

Estruturas

- Variáveis compostas heterogêneas (**estruturas**) são um conjunto de variáveis de tipos **diferentes** que são logicamente relacionadas.
- Essas variáveis compartilham o **mesmo identificador** e ocupam posições consecutivas de memória.
- Para as variáveis de uma estrutura:
  - Elas são denominadas **membros**;
  - São identificadas por **nomes**.



Exemplo - Declaração

- Vamos criar uma estrutura de endereço, que possa ser usada como se fosse um tipo de dado posteriormente
- Este código deve vir no início do programa, após os "*includes*"

```
struct sEndereco
{
    char rua[40];
    int numero;
    char cidade[30];
    char estado[2];
    long int CEP;
};
```

Declaração de uma variável do tipo identificacao_da_estrutura

```
struct identificacao_da_estrutura nome_da_variavel;
```

Exemplo - Programa

- Vamos criar uma programa que use a estrutura **sEndereco** e atribua valores a todas as variáveis da estrutura

```
int main( )
{
    //cria variavel ender1 como struct sEndereco
    struct sEndereco ender1;

    strcpy(ender1.rua,"Rua 7 de Setembro");
    ender1.numero = 405;
    strcpy(ender1.cidade,"Goiania");
    strcpy(ender1.estado,"GO");
    ender1.CEP = 06599604;
}
```

Declarando, atribuindo, imprimindo

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

struct sRetangulo
{
    float altura, largura;
};

int main(void)
{
    struct sRetangulo ret1;

    ret1.altura = 10;
    printf("Digite o valor da largura: ");
    scanf("%f", &ret1.largura);

    printf("Altura: %.1f", ret1.altura);
    printf("Largura: %.1f", ret1.largura);

    return 0;
}
```

Estruturas Rotuladas

- Estruturas rotuladas criam um “rótulo” que pode ser referenciado posteriormente no código.
- Criação de ***rótulos***.

```
struct rotulo_da_estrutura
{
    tipo1 nome1;
    tipo2 nome2;
    ...
    tipoN nomeN;
};
```

Estruturas Rotuladas

```c
#include <stdio.h>

struct sHora {
   int hora;
   int minuto;
   int segundo;
};

int main(void){
   struct sHora H;
   H.hora = 10;
   H.minuto = 15;
   H.segundo = 30;
   printf("%d:%d:%d", H.hora, H.minuto, H.segundo);
   return 0;
}
```

Estruturas Rotuladas e Nomeadas

- Uma estrutura rotulada e nomeada pode ser definida da seguinte forma:

```
typedef struct rotulo_da_estrutura
{
    tipo1 nome1;
    tipo2 nome2;
    ...
    tipoN nomeN;
} id_tipo_da_estrutura;
```

Estruturas Rotuladas e Nomeadas

```c
#include <stdio.h>
typedef struct Hora {
   int hora;
   int minuto;
   int segundo;
} THora;

int main(void) {
   THora H;
   H.hora = 10;
   H.minuto = 15;
   H.segundo = 30;
   printf("%d:%d:%d", H.hora, H.minuto, H.segundo);
   return 0;
}
```

Exercício 1

a) Crie uma estrutura Livro com os seguintes campos:
   1. Titulo
   2. Autor
   3. Número de Páginas
   4. Preço
   5. Ano de publicação

b) Defina uma variável do tipo da Estrutura Livro

c) Atribua valores para cada um dos campos da estrutura Livro

d) Imprima os valores dos campos

Exercício 2

a) Escreva um programa que possua uma variável capaz de armazenar o **nome**, a **idade**, o **sexo** e o **peso** de uma pessoa. Teste a variável atribuindo e lendo os valores dela.

b) Defina um tipo de estrutura rotulada para representar números complexos da forma **a + b.i**, sendo **a** a parte real e **b** a imaginária. Crie também uma função para calcular a soma de dois números complexos, codificando também um programa para testar o seu funcionamento

Estruturas Aninhadas

- Estruturas em que um ou mais de seus membros também sejam estruturas.

```
typedef struct rotulo_estrutura1 {
    tipo1 nome1;
    tipoN nomeN;
} id_estrutura1;


typedef struct rotulo_estrutura2 {
    id_estrutura1 nome;
    tipoN nomeN;
} id_estrutura2;
```

Estruturas Aninhadas

- Exemplo:

```c
#include <stdio.h>
#include <string.h>

typedef struct sHora
{
  int hora;
  int minuto;
  int segundo;
} Hora;

typedef struct sRelogio
{
  Hora H;
  char modelo[10];
} Relogio;
```

Estruturas Aninhadas

```c
#include <stdio.h>
#include <string.h>

typedef struct sHora
{
  int hora;
  int minuto;
  int segundo;
} Hora;

typedef struct sRelogio
{
  Hora H;
  char modelo[10];
} Relogio;
```

```c
int main(void)
{
  Relogio r1;

  r1.H.hora = 10;
  r1.H.minuto = 15;
  r1.H.segundo = 30;

  strcpy(r1.modelo,"Cassio");

  printf("Modelo: %s\n", r1.modelo);

  printf("%d:%d:%d",r1.H.hora,
                    r1.H.minuto,
                   r1.H.segundo);

  return 0;
}
```

Exercício 4

a) Defina um tipo de estrutura para armazenar os dados de um vôo:
   1. nomes das cidades de origem e destino,
   2. datas e horários de partida e chegada.

b) Utilize a estrutura **hora** do exemplo anterior.

c) Crie um programa para testar as funcionalidades criadas, declarando variáveis e funções necessárias.

Arrays e Estruturas

- É possível combinar *arrays* e estruturas para criação de diferentes estruturas de dados.
- Podemos ter uma estrutura contendo um membro do tipo *array*, ou;
- Criar um *array* cujo os elementos sejam estruturas

Declarando Arrays de Estruturas

→ Dada a estrutura listada abaixo:

```
struct lista
{
    char titulo[30];
    char autor[30];
    int regnum;
    double preco;
};
```

Membros do tipo array

a) Declare um vetor com 50 elementos do tipo **lista**

Declarando arrays de Estruturas

```
struct lista livro[50];
```

→ **livro** é um vetor de 50 elementos.

→ Cada elemento do vetor é uma estrutura do tipo struct **lista**

→ O que significa **livro[0]**, **livro[1], livro[2]**, etc?

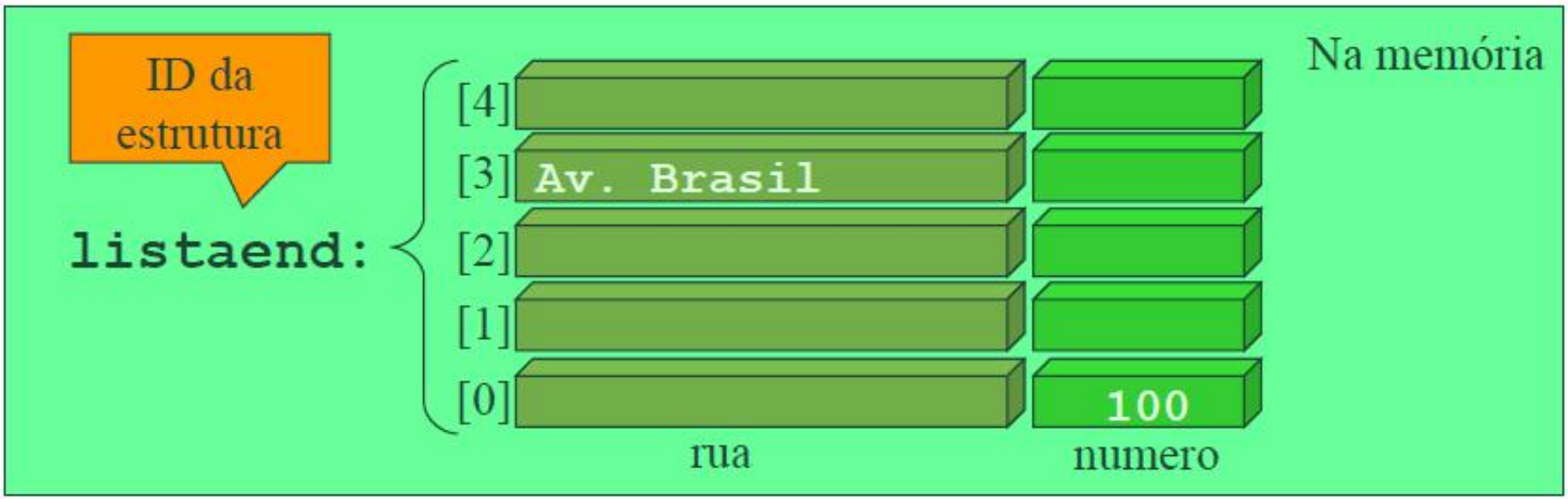
** Por meio dessa instrução o compilador providencia espaço de memória para 50 estruturas do tipo struct **lista.**

Arrays e Estruturas – Trecho de exemplo

```
...
struct sEndereco
{
    char rua[40];
    int  numero;
};

int main(void) {

   struct sEndereco listaend[5];

   listaend[0].numero = 100;

   strcpy(listaend[3].rua,"Av. Brasil");
   ...
}
```

Arrays e Estruturas – Trecho de exemplo

```
...
struct sEndereco
{
     char rua[40];
     int  numero;
};
int main(void) {
   struct sEndereco listaend[5];
   listaend[0].numero = 100;
   strcpy(listaend[3].rua,"Av. Brasil");
   ...
}
```



Arrays e Estruturas - Exemplo

```c
#include <stdio.h>
struct sHora {
   int hor;
   int min;
   int seg;
};

int main(void) {

   struct sHora H[5];
   H[0].hor = 10;
   H[0].min = 15;
   H[0].seg = 30;
   printf("%d:%d:%d", H[0].hor, H[0].min, H[0].seg);
   return 0;
}
```

Exercício 5

a) Crie um programa que permita armazenar o nome, a altura e a data de nascimento de 10 pessoas. Cada pessoa deve ser representada por uma *struct*.

b) A data de nascimento também deve ser uma *struct*.

c) O nome, altura e data de nascimento de cada pessoa devem ser informados pelo teclado.

d) Imprimir o nome da pessoa com a maior altura.