Tipos Enumerados e Estruturas Heterogêneas

Prof^a.Dr^a.Thatyana de Faria Piola Seraphim (ECO)
Prof.Dr.Enzo Seraphim (ECO)

Universidade Federal de Itajubá

thatyana@unifei.edu.br
seraphim@unifei.edu.br

Tipos Enumerados

- A linguagem C possui os tipos de dados pré-definidos que são os chamados **tipos básicos**.
- Uma outra forma de definir novos tipos de dados é usar a **enumeração**.
- **Enumeração**: é um conjunto de constantes inteiras que especificam todos os valores legais que uma variável desse tipo pode ter.
- Em C, a enumeração é definida pela palavra chave **enum**.
- Uma variável **enum** é sempre inteira e para cada um dos valores do conjunto, é atribuído um nome significativo.

Sintaxe do enum

```
enum e_nome{<elem_1>,<elem_2>,...,<elem_n>};
```

Tipos Enumerados

- Cada elemento do **enum** representa um valor inteiro.
- Os elementos podem ser usados em qualquer lugar que um inteiro possa ser usado.
- O valor do primeiro elementos do **enum** é 0.

Exemplo usando enum

```
1  #include<stdio.h>
2  enum e_semana{domingo,segunda,
3   terca,quarta,quinta,sexta,sabado};
4  int main(int argc, char *argv[]){
5    enum e_semana sem;
6    printf("Semana = %d, %d, %d",
7    domingo, terca, sabado);
8    sem = sexta;
9    printf("Semana = %d", sem);
10   return 0;
11 }//end main
```

Equivalente

```
#define DOMINGO 0
#define SEGUNDA 1
#define TERCA 2
#define QUARTA 3
#define QUINTA 4
#define SEXTA 5
#define SABADO 6
```

Tipos Enumerados

Inicializando o primeiro elemento do enum

```
1  #include<stdio.h>
2  enum e_semana{domingo=1,segunda,terca,quarta,quinta,
3                sexta,sabado};
4  int main(int argc, char *argv[]){
5    printf("Semana= %d, %d, %d \n", domingo, terca, sabado);
6    return 0;
7  }//end main
```

Inicializando todos os elementos do enum

```
1  #include<stdio.h>
2  enum e_semana{domingo=1,segunda=2,terca=4,quarta=8,quinta=16,
3                sexta=32,sabado=64};
4  int main(int argc, char *argv[]){
5    printf("Semana= %d, %d, %d \n", domingo, terca, sabado);
6    return 0;
7  }//end main
```

Definição de Tipos

typedef

- A diretiva **typedef** (*type definition*) permite ao programador redefinir um tipo de dado dando-lhe um novo nome.
- Essa forma de programação ajuda em dois sentidos:
  - Fica mais fácil entender para que serve tal tipo de dado.
  - É a única forma de referenciar uma estrutura de dados dentro de outra.
- **typedef** faz o compilador assumir que o novo nome é um determinado tipo de dado, e então o programador utiliza o novo nome da mesma forma que usaria o antigo.
- A declaração da diretiva **typedef** é feita fora da função *main*.
- Com o uso da diretiva **typedef** não cria-se um novo tipo de dado, apenas redefine um novo nome para o tipo já existente.

Definição de Tipos

typedef

Sintaxe

```
typedef tipo t_nome;
```

- **tipo**: qualquer tipo de dado permitido.
- **nome**: o novo nome para esse tipo.

- A diretiva **typedef** é bastante usada quando o programador precisa definir um tipo booleano.

Definição do tipo bool

```
typedef enum {false,true} bool;
```

enum bool a;

bool a;

Definição de Tipos

typedef

Uso da diretiva *typedef*

```
1  #include<stdio.h>
2  typedef float t_decimal;
3  int main(int argc, char *argv[]){
4    t_decimal p1;
5     printf("Digite a nota = ");
6     scanf("%f \n", &p1);
7     printf("A nota foi = %f\n", p1);
8     return 0;
9  }//end main
```

Definição de Tipos

typedef

enum sem typedef

```
1  #include<stdio.h>
2  enum e_semana{domingo,segunda,terca,quarta,quinta,
3                sexta,sabado};
4  int main(int argc, char *argv[]){
5    enum e_semana sem;
6    printf("Semana = %d, %d, %d\n", domingo, terca, sabado);
7    sem = sexta;
8    printf("Semana = %d\n", sem);
9    return 0;
10 }//end main
```

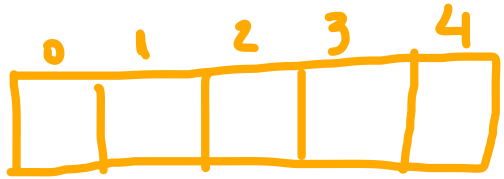
Definição de Tipos

typedef

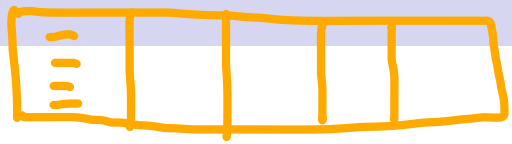
enum com typedef

```
1  #include<stdio.h>
2  typedef enum{domingo,segunda,terca,quarta,quinta,
3                  sexta,sabado} t_semana;
4  int main(int argc, char *argv[]){
5    t_semana sem;
6    printf("Semana = %d, %d, %d\n", domingo, terca, sabado);
7    sem = sexta;
8    printf("Semana = %d\n", sem);
9    return 0;
10 }//end main
```

int vet[5];



Estruturas Heterogêneas



- Os vetores são estruturas de dados que contêm um número determinado de elementos e todos os elementos têm o mesmo tipo de dado.
- O fato de todos os elementos terem o mesmo tipo de dado é uma grande limitação quando é necessário um grupo de elementos com diferentes tipos de dado.
- A solução para esse problema é utilizar um tipo de dado chamado **estrutura**.
- Os elementos individuais de uma estrutura são chamados de **membros**.
  - Cada membro (elemento) de uma estrutura pode conter dados de um tipo diferente dos outros membros.

Definição de estrutura

Uma estrutura é uma coleção de um ou mais tipos de elementos denominados **membros**, cada um podendo ser de um tipo diferente de dado.

- Um estrutura pode conter qualquer número de membros.
- Cada membro tem um nome único.
- Por exemplo: para armazenar os dados de uma coleção de séries de TV.

Nome do Membro	Tipo de dado
Título	vetor de tamanho 20
Produção	vetor de tamanho 30
Temporada	inteiro
Preço	ponto flutuante (float)
Data da compra	vetor de tamanho 8

Estruturas Heterogêneas

- A estrutura precisa ser declarada antes de ser usada.
- As estruturas são declaradas logo após a diretiva #include.

Sintaxe

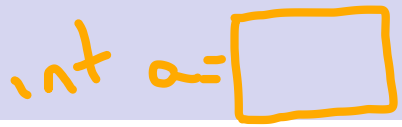
```
struct st_nomeEstrutura{
  tipoDadoMembro_1 nomeMembro_1;
  tipoDadoMembro_2 nomeMembro_2;
  ...
  tipoDadoMembro_n nomeMembro_n;
}; //end struct
```

Exemplo

```
struct st_serietv{
  char titulo[30];
  char producao[20];
  int temporada;
  float preco;
  char dataCompra[8];
};//end struct
```

struct st_serietv a = { titulo, produção, preço, temporada, data }

Estruturas Heterogêneas



- Uma estrutura pode ser acessada utilizando uma variável ou variáveis que devem ser definidas depois da declaração da estrutura.
- As variáveis de estruturas podem ser declaradas colocando o nome da estrutura seguida do nome da variável ou das variáveis antes de usá-las no programa.

Declaração das variáveis

```
1  #include<stdio.h>
2  struct st_serietv{
3    char titulo[30], producao[20], dataCompra[8];
4    int temporada;
5    float preco;
6  };//end struct
7  int main(int argc, char *argv[]){
8    struct st_serietv s1, s2;
9    ...
10 }//end main
```

Estruturas Heterogêneas

- A definição de novos tipos com a diretiva **typedef** também pode ser usada em estruturas.
- O uso da diretiva **typedef** permite a declaração e a inicialização de variáveis do tipo da estrutura.

typedef com estruturas

```
1  typedef struct{
2    char titulo[30];
3    char producao[20];
4    char dataCompra[8];
5    int temporada;
6    float preco;
7  }t_serietv;
```

typedef com estruturas

ANTES

```
struct st_serietv s1, s2;
```

USANDO *typedef*

```
t_serietv s1, s2;
```

Inicialização da estrutura

```
1  typedef struct{
2    char titulo[30];
3    char producao[20];
4    char dataCompra[8];
5    int temporada;
6    float preco;
7  }t_serietv;
```

- Uma vez que o tipo da estrutura foi definido, pode-se declarar e inicializar uma variável do tipo estrutura.

Inicialização da estrutura

```
1  t_serietv s1 = {"Big Bang Theory", "Warner", "25/05/10", 3, 55.50};
```

Estruturas Heterogêneas

- Uma vez que a variável do tipo estrutura foi declarada, os membros da estrutura podem ser acessados através do operador ponto (.).
- O operador ponto conecta o nome de uma variável estrutura a um membro da estrutura.

Sintaxe para acesso

nomeVariavel.nomeMembro = *dados*;

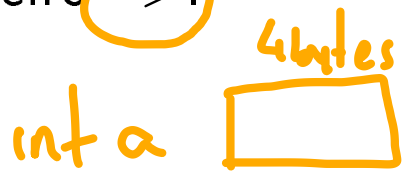
Exemplo de acesso

```
1 strcpy(s1.titulo,"Big Bang Theory");
2 strcpy(s1.producao,"Warner");
3 s1.temporada = 3;
4 printf("A serie foi produzida pela %s,", s1.producao);
5 printf(" e esta na temporada %d\n", s1.temporada);
```

scanf("%s", s1.titulo);
scanf("%d", &s1.temporada);

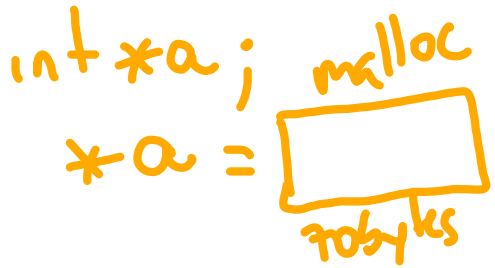
Estruturas Heterogêneas

- Quando a variável do tipo estrutura é um ponteiro, os dados são acessados usando o operador ponteiro ->.



Acesso dos dados por ponteiro

```
1  #include<stdio.h>
2  typedef struct{
3    char titulo[30], producao[20], dataCompra[8];
4    int temporada;
5    float preco;
6  }t_serietv;
7  int main(int argc, char *argv[]){
8    t_serietv *s1;
9    s1 = (t_serietv*) malloc(sizeof(t_serietv));
10   strcpy(s1->titulo,"Big Bang Theory");
11   s1->temporada = 3;
12   printf("%s esta na temporada %d \n",s1->titulo,s1->temporada);
13   free(s1);
14   return 0;
15 }//end main
```



Estruturas Heterogêneas

- Um estrutura pode conter outras estruturas chamadas **estruturas aninhadas**.
- As estruturas aninhadas economizam tempo na escrita de programas que utilizam estruturas semelhantes.

Exemplo

```
1  typedef enum{deposito,saque} t_operacao;
2  typedef struct{
3     int dia,mes,ano;
4  }t_data;
5  typedef struct{
6     long int numConta;
7     float quant;
8     t_operacao tipo;
9     t_data data;
10 }t_movimenta;
```

Acesso aos dados

```
t_movimenta operacao;
operacao.tipo = saque;
operacao.quant = 500.00;
operacao.data.mes = 10;
operacao.data.dia = 25;
```

Outro exemplo

```
1   typedef struct{
2     char endereco[50];
3     char cidade[20];
4     char estado[20];
5     long int cep;
6   }t_end;
7
8   typedef struct{
9     char nomeFunc[30];
10    t_end endFun;
11    float salario;
12  }t_func;
13
14  typedef struct{
15    char nomeCli[30];
16    t_end endCli;
17    float saldo;
18  }t_cli;
```

Acesso aos dados

```
1  t_fun f;
2  t_cli c;
3  f.salario = 1345.50;
4  strcpy(f.endFun.endereco,"Rua X,sn");
5  strcpy(c.nomeCli,"Sheldon Cooper");
6  strcpy(c.endCli.cidade,"Itajuba");
```

Agenda

```
1  #include<stdio.h>
2  typedef struct{
3    int dia;
4    int mes;
5    int ano;
6  }t_data;
7  typedef struct{
8    char nome[20];
9    char end[30];
10   int fone;
11   t_data nasc;
12 }t_item;
```

Agenda cont...

```
13  int main(int argc, char *argv[]){
14    int i;
15    t_item a[55];
16    for(i=0; i<55; i++){
17      printf("numero=%d \n", i);
18      printf("Digite nome: ");
19      scanf("%s", a[i].nome);
20      printf("Digite end: ");
21      scanf("%s", a[i].end);
22      printf("Digite fone: ");
23      scanf("%d", &a[i].fone);
24      printf("Digite nasc(dd-mm-aaaa): ");
25      scanf("%d-%d-%d", &a[i].nasc.dia, &a[i].nasc.mes,
26            &a[i].nasc.ano);
27    }//end for
28    return 0;
29  }//end main
```