

# Tipos Enumerados: (enum)

## MC102 Algoritmos e Programação de Computadores

Aula de Laboratório 11

Instituto de Computação  
Primeiro Semestre de 2012

21 de maio de 2012



- Tipos Enumerados é um tipo de dado abstrato, cujos valores são atribuídos a um conjunto finito de identificadores.

### Declaração

```
enum nomeDoTipo {ident1, ident2, ..., identN};
```

- Geram uma saída com valores inteiros e possui a versatilidade de atribuir os identificadores do tipo enumerado para tais variáveis;
- Um tipo enumerado deixa seu código mais legível.



MC102 - Algoritmos e Programação de Computadores

3 / 17

Tipos Enumerados

Registros

Redefinido um tipo

Ponteiros para Registros

Tipos Enumerados

Registros

Redefinido um tipo

Ponteiros para Registros

## Conteúdo

- Tipos Enumerados
- Registros
- Redefinido um tipo
- Ponteiros para Registros



MC102 - Algoritmos e Programação de Computadores

2 / 17

## Ponteiros

### Exemplo1

```
1 #include <stdio.h>
2 /*aqui criamos um novo tipo enumerado
3 que pode ser usado por qualquer funcao*/
4
5 enum meses {jan, fev, mar, abr, mai, jun,
6 jul, ago, set, out, nov, dec};
7 int main(){
8     enum meses a,b;//criamos 2 variaveis do tipo "enum
9     meses"
10    a = jan;
11    b = jun;
12    if(a != b){
13        printf("%d e um mes diferente de %d\n", a, b);
14        //sera impresso "0 e um mes diferente de 5"
15    }
16 }
```



MC102 - Algoritmos e Programação de Computadores

4 / 17

## Ponteiros

### Exemplo2

```

1 #include <stdio.h>
2 /*aqui criamos um novo tipo enumerado
3 que pode ser usado por qualquer funcao*/
4
5 enum meses {jan=1,fev ,mar ,abr ,mai ,jun ,
6   jul ,ago ,set ,out ,nov ,dec };
7 int main(){
8   enum meses a,b;//criamos 2 variaveis do tipo "enum
9   meses"
10  a = jan;
11  b = jun;
12  if(a != b){
13    printf("%d e um mes diferente de %d\n", a, b);
14    //sera impresso "0 e um mes diferente de 5"
15  }

```



## Registros

### Exemplo

```

struct Aluno {
  char nome[45];
  int idade;
  char sexo;
};

```

- Para declarar uma variável do tipo aluno deve-se utilizar:
  - **struct Aluno a, b;**
- Pode-se acessar cada variável individualmente. A sintaxe é:
  - **variável-registro.nome-campo.**

Ex.:

```

struct Aluno aluno;
aluno.nome;

```



## Registros

- É um mecanismo para agrupar variáveis, que podem ser de tipos diferentes, mas que fazem sentido estarem juntas.

### Declaração

```

struct nome-do-registro {
  tipo1 ident1;
  tipo2 ident2;
  ...
  tipoN identN;
};

```



## Registros

### Exemplo - Usando Registros

```

1 #include <stdio.h>
2 struct Aluno{
3   char nome[45];
4   int idade;
5   char sexo;
6 };
7 int main(){
8   struct Aluno a, b;
9   printf("Digite o nome:");
10  scanf("%[^\n]", a.nome);
11  printf("Digite a idade:");
12  scanf("%d", &a.idade);
13  printf("Digite o sexo:");
14  getchar(); //usado para limpar o buffer de entrada
15  scanf("%c", &a.sexo);
16  b = a;
17  printf("Nome:%s Id:%d S:%c\n",b.nome,b.idade ,b.sexo);
18 }

```



## Registros

### Exemplo - Vetor de Registros

```

1 #include <stdio.h>
2 struct Aluno{
3     int ra;
4     float nota;
5 };
6 int main(){
7     int i;
8     struct Aluno turma[5];
9     for (i = 0; i < 5; i++) {
10        printf ("RA do aluno[%d]: ", i);
11        scanf ("%d", &turma[i].ra);
12        printf ("Nota do aluno: ");
13        scanf ("%f", &turma[i].nota);
14    }
15    for (i = 0; i < 5; i++)
16        printf("RA:%d, Nota:%.2f\n",turma[i].ra , turma[i].
17            nota);

```



MC102 - Algoritmos e Programação de Computadores

9 / 17

## Registros

### Problema

Defina a estrutura Aluno que possui os campos RAa e a nota média. O seu programa deve armazenar o RA e a nota de 10 alunos e exibir a média da turma.



MC102 - Algoritmos e Programação de Computadores

10 / 17

## Registros

### Solução

```

1 #include <stdio.h>
2 struct aluno{
3     int ra;
4     float nota;
5 };
6 typedef struct aluno Aluno;
7 int main(){
8     int i;
9     float media=0;
10    Aluno turma[10];
11    for(i=0; i<10; i++){
12        printf("Digite o RA:");
13        scanf("%d", &turma[i].ra);
14        printf("Digite a nota do aluno:");
15        scanf("%f", &turma[i].nota);
16    }
17    for(i=0; i<10; i++){
18        media += turma[i].nota;
19    }
20    printf("Media da turma:%.2f\n",media/10);
21 }

```



MC102 - Algoritmos e Programação de Computadores

11 / 17

## Redefinido um tipo

- A linguagem C, por questão de organização, lhe permite criar um tipo que faz a mesma coisa que outro tipo já existente.

### Declaração

```
typedef <tipo-ja-existente> <tipo-novo>;
```



MC102 - Algoritmos e Programação de Computadores

12 / 17

## Redefinido um tipo

- A linguagem C, por questão de organização, lhe permite criar um tipo que faz a mesma coisa que outro tipo já existente.

### Declaração

```
typedef <tipo-ja-existente> <tipo-novo>;
```

### Exemplo

```
typedef float nota; //cria o tipo nota, valor real
nota p1; //declara p1 do tipo nota, que é um real
```



## Redefinido um tipo

### Exemplo - utilizando com Registros

```
1 #include <stdio.h>
2 struct Aluno{
3     int ra;
4     float nota;
5 };
6
7 //redefinimos tipo struct Aluno como Aluno
8 typedef struct Aluno Aluno;
9
10 int main(){
11     int i;
12     Aluno turma[5]; //Sem typedef seria: struct Aluno turma[5]
13     for (i = 0; i < 5; i++) {
14         printf ("RA do aluno[%d]: ", i);
15         scanf ("%d", &turma[i].ra);
16         printf ("Nota do aluno: ");
17         scanf ("%f", &turma[i].nota);
18     }
19     for (i = 0; i < 5; i++)
20         printf("RA:%d, Nota:%.2f\n",turma[i].ra , turma[i].nota);
21 }
```



## Redefinido um tipo

### Exemplo

```
1 #include <stdio.h>
2
3 typedef double nota;
4
5 int main(){
6     nota p1;
7     printf("Digite a nota:");
8     scanf("%lf",&p1);
9     printf("A nota digitada foi: %.2lf\n",p1);
10 }
```



## Ponteiros para Registros

- É possível criar ponteiro para um struct;

### Exemplo

```
typedef struct Aluno Aluno;
int main{
    Aluno a, *p;
    p = &a;
    ...
}
```



## Ponteiros para Registros

- É possível criar ponteiro para um struct;

### Exemplo

```
typedef struct Aluno Aluno;
int main{
    Aluno a, *p;
    p = &a;
    ...
}
```

- Usando Ponteiros existe basicamente **duas formas** de acessar os campos da estrutura:
  - 1 Utilizando o operador **\*** juntamente com o operador **.** (ponto):  
**Ex.:** `(*p).ra = 858585;`
  - 2 Utilizando o operador **->**:  
**Ex.:** `p->ra = 858585;`



## Questões?

Obrigado!

Para informação:

Página dos Laboratórios (Tarefas): <http://susy.ic.unicamp.br:9999/mc102ab>

Página do Curso: <http://www.lrc.ic.unicamp.br/~geraldoms/mc102>

E-mail:

[geraldoms\[at\]lrc.ic.unicamp.br](mailto:geraldoms[at]lrc.ic.unicamp.br)

[brhenrique.fischer\[at\]gmail.com](mailto:brhenrique.fischer[at]gmail.com)



## Ponteiros para Registros

### Exemplo - Ponteiro para um Registros

```
1 #include <stdio.h>
2 struct Coordenada{
3     double x;
4     double y;
5 };
6 typedef struct Coordenada Coord;
7 int main(){
8     Coord c1, c2, *c3, *c4;
9     c3 = &c1;
10    c4 = &c2;
11    c1.x = -1;
12    c1.y = -1.5;
13    c2.x = 2.5;
14    c2.y = -5;
15    (*c3).x = 1.5;
16    (*c3).y = 1.5;
17    c4->x = -1;
18    c4->y = -1;
19    printf("Coordenadas de c1: (%.21f,%.21f)\n",c1.x, c1.y);
20    printf("Coordenadas de c2: (%.21f,%.21f)\n",c2.x, c2.y);
21 }
```

