

29. Enumerações e uniões

DIM0321

2015.1

Outline

1 Enumerações

2 Uniões

1 Enumerações

2 Uniões

Contexto

Muitas vezes o conjunto de valores aceitáveis para uma variável é restrito

Exemplo

- lógico: verdadeiro/falso
- sexo: homem/mulher
- tipo de peça de xadrez: peão, cavalo, bispo, torre, rei, dama

Codificação

Codificação com tipos de base

- Por exemplo, inteiros (peão = 1, cavalo = 2, bispo = 3, ...)
- Abordagem difícil de compreensão.
- Não é possível saber que vai ter apenas n valores.
- O significado dos valores não é transparente.

Enumeração

- Tipo especial de variável que deve conter um número restrito de valores distintos
- A representação **concreta** usa uma sequência automática de números naturais.

Sintaxe / declaração

```
1 enum peca { // 0 tipo enum é tratado como um inteiro
2     PEAO,    // 0
3     CAVALO,  // 1
4     BISPO,   // 2
5     TORRE,   // 3
6     REI,     // 4
7     DAMA     // 5
8 };
9 enum peca p;
10
11 enum logico { // Declaração no momento da declaração do enum
12     V,
13     F
14 } booleano, primos[100];
15
16 typedef enum peca peca_t; // Podemos usar typedef
17
18 typedef enum sexo {
19     HOMEM = 4,
20     MULHER = 3
21 } sexo_t;
22
23 typedef enum cor { // É possível atribuir valores diferentes do padrao 0,1,2,3,...
24     PRETO , // = 0
25     BRANCO = 2,
26     AMARELO , // = 3
27     VERMELHO = 9
28 } cor_t;
```

Exemplo

```
1  #include <stdio.h>
2  int main(void)
3  {
4      enum dia
5      {
6          DOMINGO,
7          SEGUNDA,
8          TERCA,
9          QUARTA,
10         QUINTA,
11         SEXTA,
12         SABADO
13     } dia_semana = QUARTA;
14
15     enum dia primeiro_dia = DOMINGO;
16     enum dia ultimo_dia = SABADO;
17
18     printf( "Primeiro dia da semana: %i\n", primeiro_dia );
19     printf( "Ultimo dia da semana: %i\n", ultimo_dia );
20     printf( "Hoje = %i\n", dia_semana );
21     printf( "Amanha = %i\n", ++dia_semana );
22     dia_semana = SABADO;
23     printf( "Outro dia da semana = %i\n", dia_semana );
24     return 0;
25 }
```

Exemplo

```
1  #include <stdio.h>
2
3  typedef enum {
4      NEGATIVO,
5      POSITIVO
6  } sinal_t;
7
8  typedef struct {
9      sinal_t sinal;
10     unsigned int numerador;
11     unsigned int denominador;
12 } racional_t;
13
14 void imprime_racional(racional_t r)
15 {
16     printf("%c", r.sinal == POSITIVO ? '+' : '-');
17     printf("%u/%u", r.numerador, r.denominador);
18 }
19
20 racional_t mult_racional(racional_t r1, racional_t r2)
21 {
22     racional_t resultado;
23     resultado.sinal = (r1.sinal == r2.sinal ? POSITIVO : NEGATIVO);
24     resultado.numerador = r1.numerador * r2.numerador;
25     resultado.denominador = r1.denominador * r2.denominador;
26     return resultado;
27 }
28
29 int main(void)
30 {
31     racional_t x = { POSITIVO, 2, 5 };
32     racional_t y = { NEGATIVO, 1, 2 };
33     racional_t z = mult_racional(x, y);
34     imprime_racional(x);
35     printf(" * ");
36     imprime_racional(y);
37     printf(" = ");
38     imprime_racional(z);
39     return 0;
40 }
```

1 Enumerações

2 Uniões

Contexto

- Sistemas com pouco espaço de memória disponível: em certos casos precisamos economizar espaço de memória alocado desnecessariamente em registros.
- Em outros casos precisamos criar estruturas contendo uma mistura de diferentes tipos em um único campo. Ex: Conjunto de números inteiros e de números de ponto flutuante armazenados em um único vetor.

União

Definição

- Tipo especial de registro
- Ele armazena apenas um dos valores de seus campos na memória.
- O compilador aloca espaço suficiente apenas para **o maior** dos campos, em

termos de bytes alocados. Todos os campos compartilham o mesmo espaço na memória. A atribuição de um valor a um dos campos também altera os valores dos outros campos.

- `struct s { char c; int i; } = 5 bytes`
- `union u { char c; int i; } = 4 bytes`

Sintaxe / declaração

De um novo tipo / muito parecido a um registro. Só o primeiro campo pode ser inicializado na declaração.

```
1 union num {  
2     int i;  
3     float f;  
4 };  
5  
6 union num v[5];  
7  
8 typedef union num num_t;  
9  
10 num_t n;
```

Exemplo

```
#include <stdio.h>
#define MAX 100
```

```
typedef enum {
    TIPO_INT,
    TIPO_DOUBLE
} tipo;
```

```
typedef struct {
    tipo tipo;
    union {
        int i;
        double d;
    } uniao;
} mixnum;
```

```
1 | int main(void)
2 | {
3 |     int i, soma = 0, inteiro = 0, real = 0;
4 |     double produto = 1.;
5 |     mixnum numero[MAX];
6 |     for (i = 0; i < MAX; i++) {
7 |         scanf("%i", &numero[i].tipo);
8 |         if (numero[i].tipo == TIPO_INT)
9 |             scanf("%i", &numero[i].uniao.i);
10 |        else
11 |            scanf("%lf", &numero[i].uniao.d);
12 |    }
13 |    for (i = 0; i < MAX; i++) {
14 |        if (numero[i].tipo == TIPO_INT) {
15 |            soma += numero[i].uniao.i;
16 |            inteiro = 1;
17 |        } else {
18 |            produto *= numero[i].uniao.d;
19 |            real = 1;
20 |        }
21 |    }
22 |    if (inteiro) printf("Soma dos inteiros: %i\n", soma);
23 |    if (real) printf("Produto dos reais: %g\n", produto);
24 |    return 0;
25 | }
```

Exemplo

```
1 typedef enum {
2     NEGATIVO,
3     POSITIVO
4 } sinal_t;
5
6 typedef struct {
7     sinal_t sinal;
8     unsigned int numerador;
9     unsigned int denominador;
10 } racional_t;
11
12 typedef union {
13     unsigned int natural;
14     int inteiro;
15     racional_t racional;
16 } num_t;
17
18 #include <stdio.h>
19 int main(void)
20 {
21     num_t num1, num2, num3, num4;
22     printf("sizeof (unsigned int) = %lu\n"
23           "sizeof (int) = %lu\n"
24           "sizeof (double) = %lu\n"
25           "sizeof (sinal_t) = %lu\n"
26           "sizeof (racional_t) = %lu\n"
27           "sizeof (num_t) = %lu\n",
28           sizeof(unsigned int), sizeof(int),
29           sizeof(double), sizeof(sinal_t),
30           sizeof(racional_t), sizeof(num_t));
31
32     num1.natural = 1;
33     num2.inteiro = 1;
34     num4.racional.sinal = POSITIVO;
35     num4.racional.numerador = 1;
36     num4.racional.denominador = 1;
37     printf("sizeof (num1) = %lu\n"
38           "sizeof (num2) = %lu\n"
39           "sizeof (num3) = %lu\n"
40           "sizeof (num4) = %lu\n",
41           sizeof(num1), sizeof(num2),
42           sizeof(num3), sizeof(num4));
43
44     return 0;
45 }
```

União / bit-field

```
1  #include <stdio.h>
2
3  typedef union {
4      float f;
5      struct {
6          unsigned mantissa : 23;
7          unsigned exponent : 8;
8          unsigned sign : 1;
9      } parts;
10 } myfloat;
11
12
13 int main(void)
14 {
15     myfloat f;
16     scanf("%f", &f.f);
17     printf("correct mantissa : %u\n", f.parts.mantissa);
18     printf("correct exponent : %u\n", f.parts.exponent);
19     return 0;
20 }
```

Exercício

O que faz esse trecho de código ?

```
1  #include <stdio.h>
2
3  typedef union {
4      struct { /* registro anônimo que não declara campo */
5          unsigned char blue;
6          unsigned char green;
7          unsigned char red;
8          unsigned char alpha;
9      };
10     unsigned int value;
11 } color_t;
12
13 int main( void )
14 {
15     color_t color;
16     scanf( "%u", &color.value );
17     printf( "%hhu\n", color.alpha );
18     printf( "%hhu\n", color.red );
19     printf( "%hhu\n", color.green );
20     printf( "%hhu\n", color.blue );
21     return 0;
22 }
```

Referências

Precisões

[KR88] 6 (6.8, 6.9)

[Bac13] 8.2, 8.3



André Backes, *Linguagem C completa e descomplicada*, Elsevier, 2013.



Brian W. Kernighan and Dennis M. Ritchie, *The (ANSI) C Programming Language*, 2nd ed., Prentice Hall Professional Technical Reference, 1988.

Perguntas ?



<http://dimap.ufrn.br/~richard/dim0321>