Estrutura de Dados

Enum

Profa.: Márcia Sampaio Lima

EST - UEA

ENUM

- Tipo de dado definido pelo usuário.
- Define uma variável que vai receber apenas um conjunto restrito de valores.
- **enum** é um conjunto de valores inteiros representados por identificadores.

ENUM

- Enumerations (Enum) são um outro método de definir constantes:
  - #define
  - const int a = 1;
  - enumerations.

ENUM

- Criando um novo tipo de dado:
  - As enumerations definem um novo tipo de variável e limita desde logo os valores.

```
enum colors {black, blue, green, cyan, red, purple, yellow, white};
```

Torre de Hanoi

- Interpretação:
  - Cada constante enumerada (também chamado de enumerator) tem um valor inteiro (**caso não seja especificado ele começa em zero**)

black	blue	green	cyan	purple	yellow	white
0	1	2	3	4	5	6

ENUM

- **Sintaxe:**

```
enum < nome > { < const1 > , < const2 > , · · · , < constN >}
```

ENUM

- Exemplo clássico: usar uma enum para armazenar os meses do ano.

enum meses_do_ano { Janeiro, Fevereiro, Março, Abril, Maio, Junho, Julho, Agosto, Setembro, Outubro, Novembro, Dezembro**}**

ENUM

- Por padrão o primeiro valor de uma enum é inicializado com zero.
- Mas é possível determinar o valor inicial:
  - Declarar o primeiro valor para o identificador;
  - Os valores seguintes serão automaticamente calculados com um incremento de uma unidade.

ENUM

enum colors {black, blue, green, cyan, red, purple, yellow, white};

black	blue	green	cyan	purple	yellow	white
0	1	2	3	4	5	6

ENUM

enum meses_do_ano { Janeiro, Fevereiro, Março, Abril, Maio, Junho, Julho, Agosto, Setembro, Outubro, Novembro, Dezembro**}**

Janeiro	Fevereiro	Março	Abril	Maio	Junho	Julho	Agosto	Setembro	Outubro	Novembro	Dezembro
0	1	2	3	4	5	6	7	8	9	10	11

ENUM

- Trocando a ordem dos valores:
  - Atribuir o valor 1 para janeiro:

```
enum meses_do_ano { Janeiro=1, Fevereiro, Março, Abril, Maio,
Junho, Julho, Agosto, Setembro, Outubro, Novembro, Dezembro}
```

Janeiro	Fevereiro	Março	Abril	Maio	Junho	Julho	Agosto	Setembro	Outubro	Novembro	Dezembro
1	2	3	4	5	6	7	8	9	10	11	12

ENUM

enum forma {quadrado=5, retangulo,triangulo=27, circulo, elipse}

Quadrado	Rectângulo	Triangulo	Circulo	Elipse
5	6	27	28	29

ENUM

- Vantagem:
  - Se uma variável é declarada tipo enumeração, tem um tipo único e os seus valores estão limitados e poderão ser verificados durante a compilação.

ENUM

```
1  #define BRASIL 0
2  #define ITALIA 1
3  #define PORTUGAL 2
4  #define ALEMANHA 3
```

E eis a maneira de fazer o mesmo utilizando `enums`:

```
1  enum pais
2  {
3          BRASIL,
4          ITALIA,
5          PORTUGAL,
6          ALEMANHA
7  };
```

ENUM

```
1  enum Paises
2  {
3          BRASIL = 2,
4          ITALIA,  //ITALIA é igual a 3 (2 + 1)
5          PORTUGAL = 1,
6          ALEMANHA  //ALEMANHA é igual a 2 (1 + 1)
7  };
```

ENUM

- **Conversão**
  - A conversão de uma constante de um enum para um inteiro é feita automaticamente.

```
1   enum Paises
2   {
3           BRASIL,
4           ARGENTINA,
5           VENEZUELA
6   };
7
8   ...
9
10  int inteiro = VENEZUELA + 2; //4
```

```c
#include <stdio.h>

enum cores { AZUL = 1, VERDE, BRANCO, };

typedef enum cores tipo_cores ;


int main() {

    tipo_cores cor = VERDE ;

   If(cor == 1) { printf("Cor azul \n"); }
   if(cor == 2) { printf("Cor verde \n"); }
   if(cor == 3 ) { printf("Cor branco \n"); }

   return 0 ;
}
```

```
#include <iostream>
#include <stdlib.h>
using namespace std;

enum { ARGENTINA = 3, ITALIA, BRASIL };

int main() {

    int n = BRASIL ;

    switch(n) {
        case BRASIL : cout <<  "Brasil invencível como de costume\n" ; break;
        case ARGENTINA : cout << "Argentina um dia quem sabe\n" ; break;
        case ITALIA : cout << "Foi sorte” << endl ; break;
        default : cout << "Se estou vivo teve erro do sistema xx” << endl;
    }
        cout << "The end !!";
        return (0);
}
```