9- DEFINIÇÂO DE TIPOS

9.1 – Typedef

Serve para definir novos nomes ao tipo de dado.

Forma geral:

typedef <tipo > < novonome>

Exemplo 1:

typedef float balance; /* balance é outro nome para float. */

/* As seguintes declarações são equivalentes: */

float soma; e

balance soma;

- balance pode ser usado em outro typedef:

typedef balance tiposoma;

A declaração:

tiposoma soma;

é equivalente às anteriores.

9.2 Enumeração

Tipo definido pelo usuário

- são constantes simbólicas cujos valores podem ser definidos automaticamente.

- a vantagem é tornar o programa mais legível. Utilizam-se os tipos enumerados ao invés de números inteiros aos quais eles correspondem.

- o compilador trata o tipo enum como inteiro, você pode utilizá-lo em expressores, comparações, leitura e etc.

- define uma lista de valores que uma variável deste tipo pode assumir. O primeiro valor começa com 0 (a menos que definido de outra forma) e para formar o seguinte incrementa-se 1.

Forma geral:

enum <nome_tipo> {$valor_1$, $valor_{2,}$ … $valor_n$};

Exemplo1:

```
int cor;     /* azul  = 0
                vermelho = 1 */
```

fica bem mais legível e fácil usar o tipo enum:

```
enum tpCor {AZUL,VERMELHO};
enum tpCor cor;
```

ou

```
enum tpCor {AZUL,VERMELHO} cor;
if (cor ==AZUL)     /* if (cor ==1)  */
```

São valores constantes, colocar os valores em maiúsculo para diferenciá-los de variáveis, é uma boa prática de programação.

Exemplo 2: satisfação do cliente, representando uma escala ordinal insatisfeito pior que pouco satisfeito 0 < 1 ….

insatisfeito	0
pouco satisfeito	1
satisfeito	2
muito satisfeito	3

```
enum tpOpiniao
  {INSATISFEITO, POUCO, SATISFEITO, MUITO} opiniao;
   if (opiniao > SATISFEITO} …..
```

Exemplo 3: iniciando o primeiro com um valor diferente de 0, os demais serão automaticamente incrementados de 1.

```
enum tpLetras {A='a', B, C};   /* C='c', char convertido para int */
```

Exemplo 4: é possível inicializar valores diferentes

```
enum escapes {backspace='\b', tab='\t', newline='\n', return='\r'};
```

O que faz o programa abaixo?

```
enum tpMeses {JAN=1, FEV, MAR…...,DEZ};     /* DEZ vale 12 */
enum tpMeses mes;
char *nomeMes[ ] ={“  “, “Janeiro”, “Fevereiro”, … “Dezembro”);
for (mes =JAN; mes <=DEZ; mes++)
    printf (“%d = %s”, mes, nomeMes[mes]);
```

Exemplo 4: utilizando enum e typedef

```
typedef  enum {AZUL,VERMELHO} tpCor;
typedef  enum cores {AZUL,VERMELHO} tpCor;

/* tpCor é um tipo e não uma variável!!  */
tpCor cor;
```

Reescreva o programa abaixo usando o tipo enumerado, considerando o número de títulos

Argentina = 2, Italia = 4 e Brasil = 5

```
#include <stdio.h>
void mostrarRes(int quem);

int main(void)
 {
   int n;
   printf(“Digite o número de titulos: ”};
   scanf(“%d”,&n);
   mostrarRes(n)
 }

void mostrarRes(int quem)
{
    switch(quem)
     {
       case 5: printf( "Brasil invencível\n" );
       break;
       case 4: printf("Foi sorte\n")                    ;
       break;
       case 2: printf("Pobrecita\n");
       break;
       default: printf("Quem sera\n");
    }
}
```

```
#include <stdio.h>
void mostrarRes(int quem);

enum titulos {ARGENTINA=2, ITALIA=4, BRASIL};
int main(void)
 {
   int pais;
   scanf(“%d”,&pais);
   mostrarRes(pais)
 }

void mostrarRes(int quem)
{
    switch(quem)
     {
       case BRASIL: printf( "Brasil invencível\n" );
       break;
       case ITALIA: printf("Foi sorte\n")                    ;
       break;
       case ARGENTINA: printf("Pobrecita\n");
       break;
       default: printf("Quem sera\n");
    }
```

outra opção com typedef:

```
typedef enum  {ARGENTINA, ITALIA, ARGENTINA} titulos;
```

…...

```
void mostrarRes(titulos quem)
```