Funções

Assim como a série de Fibonacci existem outras seqüências definidas por recorrência, ou seja, onde um valor da seqüência é definido em termos de um ou mais valores anteriores, o que é denominado de relação de recorrência.

Exercício:

Estabeleça a relação de recorrência presente na seqüência abaixo e construa uma função recursiva que recebe a posição do elemento na série e retorna seu valor.

S = { 2, 4, 8, 16, 32 ... }

Tipos de Dados Definidos pelo Usuário

1. Enumerações

Numa enumeração podemos dizer ao compilador quais identificadores devem ser associados à seqüência numérica 0, 1, 2, 3, ... Sua forma geral é:

enum nome_do_tipo_da_enumeração
{lista_de_valores} lista_de_variáveis;

enum nome_do_tipo_da_enumeração
{lista_de_valores};

enum {lista_de_valores} lista_de_variáveis;

```
#include <stdio.h>
enum dias_da_semana {segunda, terca, quarta, quinta,
     sexta, sabado, domingo};
main ()
{
   enum dias_da_semana d1,d2;
   d1=segunda;
   d2=sexta;
   if (d1==d2)
   {
      printf ("\nO dia e o mesmo.");
   }
   else
   {
      printf ("\nSao dias diferentes.");
   }
}
```

```
#include <stdio.h>
enum dias_da_semana {segunda, terca, quarta, quinta,
     sexta, sabado, domingo};
main ()
{
   enum dias_da_semana d1,d2;
   d1=4;
   d2=sexta;
   if (d1==d2)
   {
      printf ("\nO dia e o mesmo.");
   }
   else
   {
      printf ("\nSao dias diferentes.");
   }
}
```

Tipos de Dados Definidos pelo Usuário

2. Estruturas

Uma estrutura agrupa várias variáveis numa só. A estrutura, então, serve para agrupar um conjunto de dados não similares, formando um novo tipo de dado.

Tipos de Dados Definidos pelo Usuário

2. Estruturas (continuação)

Para se criar uma estrutura usa-se o comando **struct**. Sua forma geral é:

```
struct nome_do_tipo_da_estrutura
{
  tipo_1 nome_campo1;
  tipo_2 nome_campo2;
  ...
  tipo_n nome_campon;
} variáveis_estrutura;
```

Tipos de Dados Definidos pelo Usuário

```
struct nome_do_tipo_da_estrutura
{
   tipo_1 nome_campo1;
   tipo_2 nome_campo2;
   ...
   tipo_n nome_campon;
};
struct
{
   tipo_1 nome_campo1;
   tipo_2 nome_campo2;
   ...
   tipo_n nome_campon;
} variáveis_estrutura;
```

Tipos de Dados Definidos pelo Usuário

2. Estruturas (continuação)

Vamos criar uma estrutura de endereço:

```
struct tipo_endereco
{
   char rua [50];
   int numero;
   char bairro [20];
   char cidade [30];
   char sigla_estado [3];
   long int CEP;
};
```

Tipos de Dados Definidos pelo Usuário

2. Estruturas (continuação)

Vamos agora criar uma estrutura chamada ficha_pessoal com os dados pessoais de uma pessoa:

```
struct ficha_pessoal
{
  char nome [50];
  long int telefone;
  struct tipo_endereco endereco;
};
```

```
#include <stdio.h>
#include <string.h>
/* aqui entrariam as declarações da struct's vistas
   anteriormente */
main ()
{
    struct ficha_pessoal ficha;
    strcpy (ficha.nome,"Luiz Osvaldo Silva");
    ficha.telefone=4921234;
    strcpy (ficha.endereco.rua,"Rua das Flores");
    ficha.endereco.numero=10;
    strcpy (ficha.endereco.bairro,"Cidade Velha");
    strcpy (ficha.endereco.cidade,"Belo Horizonte");
    strcpy (ficha.endereco.sigla_estado,"MG");
    ficha.endereco.CEP=31340230;
}
```

Tipos de Dados Definidos pelo Usuário

Exercício:

Construa um programa que manipule um vetor com 5 registros de alunos, onde cada registro possui informações referentes ao nome, data de nascimento, número de matricula, CPF e coeficiente de rendimento do aluno. A manipulação do vetor é feita através das seguintes funções: inicializar vetor, imprimir um determinado registro com base no valor do campo CPF e imprimir um determinado registro com base em sua posição no vetor. O programa não pode possuir variáveis globais e deve se utilizar de forma satisfatória das funções mencionadas.