Estruturas de Dados

2. Registros.

Um registro é uma variável especial que contém diversas outras variáveis normalmente de tipos diferentes, agrupadas em uma unidade. Em linguagem C, os registros são especificados pela palavra-chave *struct*.

As variáveis internas contidas pela struct são denominadas membros (ou campos) da struct.

Sintaxe:

```
struct <identificador>
{
   <listagem dos tipos e membros>;
};

struct <identificador> <variavel>;
```

Exemplo de declaração de uma struct

```
struct ficha_de_aluno
{
   char nome[50];
   char disciplina[30];
   float nota_prova1;
   float nota_prova2;
};

struct ficha_de_aluno aluno;
```

Neste exemplo criamos a *struct* **ficha_de_aluno**.

Depois de criar a *struct* precisamos criar a variável que vai utilizá-la.

Para tanto criamos a variável **aluno**, que será do tipo **ficha_de_aluno**.

```
struct ficha_de_aluno aluno;
```

Vamos ver agora um código completo, extraído de “http://linguagemc.com.br/struct-em-c/”:

```
#include <stdio.h>

int main(void)
{
   /*Criando a struct */
   struct ficha_de_aluno
   {
      char nome[50];
      char disciplina[30];
      float nota_prova1;
      float nota_prova2;
   };

   /*Criando a variável aluno
     que será do tipo struct ficha_de_aluno */
   struct ficha_de_aluno aluno;

   printf("\n-------- Cadastro de aluno ---------\n\n\n");

   printf("Nome do aluno ......: ");
   fflush(stdin);

   /* usaremos o comando fgets() para ler strings, no caso o nome
      do aluno e a disciplina
      fgets(variavel, tamanho da string, entrada)
      como estamos lendo do teclado a entrada é stdin
      (entrada padrão), porém em outro caso, a entrada tambem
      poderia ser um arquivo
   */

   fgets(aluno.nome, 40, stdin);

   printf("Disciplina ......: ");
   fflush(stdin);
   fgets(aluno.disciplina, 40, stdin);

   printf("Informe a 1a. nota ..: ");

   scanf("%f", &aluno.nota_prova1);

   printf("Informe a 2a. nota ..: ");
   scanf("%f", &aluno.nota_prova2);

   printf("\n\n ---- Imprimindo os dados da struct ----\n\n");
   printf("Nome ............: %s", aluno.nome);
   printf("Disciplina .....: %s", aluno.disciplina);
   printf("Nota da Prova 1 ...: %.2f\n" , aluno.nota_prova1);
   printf("Nota da Prova 2 ...: %.2f\n" , aluno.nota_prova2);

   return(0);
}
```

2.1 Vetores de Struct

Uma struct pode ser associada a vetores. Assim podemos criar um vetor de structs, da mesma forma que criamos um vetor de inteiros. Para isso, no momento da criação da variável, especificamos quantos elementos o vetor vai conter, como no exemplo abaixo, para armazenar os dados de uma classe com até 50 alunos.

Exemplo de declaração de uma struct

```
struct ficha_de_aluno
{
   char nome[50];
   char disciplina[30];
   float nota_prova1;
   float nota_prova2;
};

struct ficha_de_aluno aluno;

struct ficha_de_aluno turma[50];
```

Observe que o exemplo acima criou uma variável aluno, contendo um registro (uma estrutura) do tipo ficha_de_aluno, e uma outra variável chamada *turma*, que é um vetor com 50 registros (estruturas) do tipo ficha_de_aluno.

Para fazer referência ao campo `nome` do primeiro aluno, usa-se a notação `turma[0].nome`.

Obviamente o índice do vetor pode ser uma variável. O exemplo abaixo ilustra um trecho de código para listar todos os elementos do vetor *turma*, supondo que *NumAlunos* contém o número de alunos cadastrados.

Exemplo:

```
int i=0;
for (i=0; i < NumAlunos; i++)
{
     printf("\nNome: %s – Disciplina: %s - P1: %d – P2: %d",
            turma[i].nome, turma[i].disciplina,
            turma[i].nota_prova1, turma[i].nota_prova2);
}
```

2.2 Estruturas aninhadas

É possível usar em C uma estrutura dentro de outra estrutura. Isso é relativamente comum, quando existem cadastros com um conjunto de campos agrupados. Por exemplo, em um cadastro de alunos, podemos precisar armazenar o endereço de cada aluno, que é composto por uma série de informações. O mesmo conjunto de informações de endereço pode ser usado em cadastros de professores, servidores e fornecedores, por exemplo.

Exemplo de definição de estuturas aninhadas. A estrutura aluno contém uma estrutura ender.

```
struct ender
{
     char rua[50];
     int numero;
     char complemento[15];
     char bairro[30];
     char cidade[30];
     char UF[4];
     char CEP[10];
};

struct aluno
{
     char nome[50];
     int matricula;
     char nascimento[12];
     char sexo;
     char telefone[20];
     struct ender endereco;
};

struct aluno cadastro[50];
```