VOLTAR

Estruturas

Apresentar o conceito de estruturas (structs) e aplicar tal conceito utilizando a linguagem C.

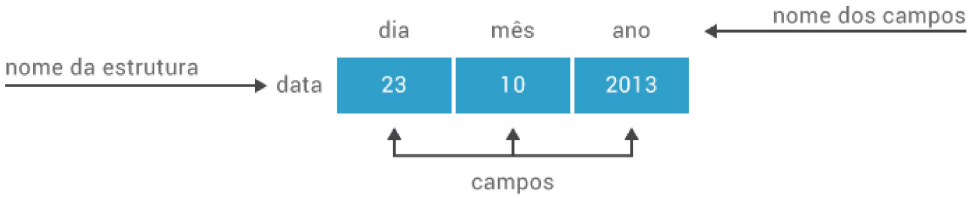
NESTE TÓPICO

Introdução

Até agora, vimos que os dados manipulados pelos programas são armazenados em variáveis simples, ou em um vetor, uma estrutura de dados homogênea, que permite armazenar uma coleção de dados do mesmo tipo.

Vamos agora estudar outra forma de manipulação de dados pelos programas, conhecida como estrutura de dados heterogênea. Este tipo de estrutura permite armazenar uma coleção de dados de tipos diferentes. Em linguagem C, essa forma de manipulação de dados é denominada estrutura ou registro (struct). Uma estrutura em C corresponde ao conceito de registro de algumas linguagens de programação, como o PASCAL.

Uma estrutura é uma coleção arbitrária de variáveis logicamente relacionadas agrupadas sob um único nome, de forma a facilitar a sua referência. As estruturas podem conter variáveis de qualquer tipo de dados válidos em C (tipos básicos, vetores, strings, ponteiros ou mesmo outras estruturas). As variáveis que fazem parte de uma estrutura são denominadas membros ou campose são identificadas por nomes. A figura a seguir apresenta a representação gráfica de uma estrutura.



Representação gráfica de uma estrutura em C

Fonte: Do próprio autor

Definição de estruturas

Antes de usarmos uma estrutura, precisamos defini-la. Definimos uma estrutura pelo seu nome e pelo conjunto de campos que ela contém. A forma geral de definição de uma estrutura em linguagem C é:

```
1.  struct <nome_estrutura> {
2.  <tipo> <nome_campo1>;
3.  <tipo> <nome_campo2>;
4.  ...
5.  <tipo> <nome_campoN>;
6.  };
```

Podemos, por exemplo, criar uma estrutura capaz de armazenar uma data.

```
1.  struct Data {
2.
3.      int dia;
4.      int mes;
5.      int ano;
6.
7.  };
```

No exemplo ilustrado, definimos uma estrutura denominada *Data*, que contém os campos *dia*, *mes*, *ano*, todos eles do tipo *inteiro*.

As estruturas devem ser definidas antes de qualquer função, isso é de maneira global. Isso permite que as funções tenham acesso a estrutura.

Declaração de variáveis do tipo estrutura

Na seção anterior, vimos como definir uma estrutura. Para a manipulação dos dados em uma estrutura é preciso declarar uma variável do tipo da estrutura. A forma geral de declaração de variáveis do tipo estrutura em linguagem C é:

```
1.  struct <nome_estrutura> <nome_variável>;
```

Por exemplo, vamos declarar uma variável para a estrutura criada da seção anterior.

```
1.  struct Data hoje;
```

Referenciando campos de uma estrutura

Uma vez declarada uma variável do tipo estrutura, podemos atribuir dados a seus campos. A forma geral na linguagem C é:

```
1.  <nome_variavel>.<nome_campo> = valor;
```

Por exemplo, podemos atribuir valores aos campos da variável do tipo estrutura declarada na seção anterior.

```
1.  hoje.dia = 20;
2.  hoje.mes = 10;
3.  hoje.ano = 2013;
```

Lendo dados para campos de uma estrutura

Imagine que temos a estrutura Data definida na seção anterior e criamos a variável *hoje*, que é do tipo desta estrutura. Queremos solicitar ao usuário que informe os dados para preencher essa variável, para isso vamos usar a função *scanf()*. Como em um vetor, não podemos acessar todos os dados ao mesmo tempo. Em uma estrutura, temos que acessar um campo de cada vez. O trecho de código a seguir solicita ao usuário para informar dados que preencham a variável hoje.

```
1.  // declaração de uma variável do tipo Data
2.  struct Data hoje;
3.  // leitura do campo dia
4.  printf ("Dia: ");
5.  scanf ("%d", &hoje.dia);
6.  // leitura do campo mês
7.  printf ("Mes: ");
8.  scanf ("%d", &hoje.mes);
9.  // leitura do campo ano
10. printf ("Ano: ");
11. scanf ("%d", &hoje.ano);
```

Acessando dados de uma estrutura

Uma vez preenchida uma estrutura, queremos manipular os seus elementos ou apresentá-los na tela. Utilizando a variável hoje preenchida na seção anterior, vamos apresentar seus dados na tela usando a função *printf()*. Analise o seguinte trecho de código.

```
// imprime o valor do campo dia
printf ("Dia = %d\n", hoje.dia);
// imprimi o valor do campo mes
printf ("Mês = %d\n", hoje.mes);
// imprime o valor do campo ano
printf ("Ano = %d\n", hoje.ano);
```

Exemplo 1

A seguir, apresentamos um exemplo completo de um programa em linguagem C que trabalha com estruturas. O programa cria uma estrutura denominada Data que contém os campos dia, mes, ano; lê um valor para cada um dos campos da estrutura e os apresenta na tela.

```
#include <stdio.h>
#include <stdlib.h>
// definição da estrutura Data
struct Data {
     int dia, mes, ano;
};

main () {
      // declaracao de uma variavel da estrutura
       struct Data umaData;
       // leitura do campo dia
       printf ("Dia: ");
       scanf ("%d", &umaData.dia);
       // leitura do campo mes
       printf ("Mês: ");
       scanf ("%d", &umaData.mes);
       // leitura do campo ano
       printf ("Ano: ");
       scanf ("%d", &umaData.ano);
       // impressão da tada informada
       printf ("\n*** Data Informada ***\n");
       printf ("%d / %d / %d\n\n", umaData.dia, umaData.mes, umaData.ano);
       system ("PAUSE");
}
```

Exemplo 2

Vamos estudar outro exemplo de utilização de estruturas. Um aluno possui um código (inteiro), um nome e três notas. O seguinte programa lê os dados de um aluno, apresenta-os na tela juntamente com a média aritmética de suas notas.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// definição da estrutura aluno
struct Aluno {
      int codigo;
      // o nome é representado por uma string
      char nome[80];
      // as notas são armazenadas em um vetor
      float notas[3];
};

main () {
      // declaração da variável da estrutura Aluno
      struct Aluno umAluno;
      // leitura do código
      printf ("Codigo: ");
      scanf ("%d", &umAluno.codigo);
      getchar();
      // leitura do nome. Usamos o comando gets() para ler a string
      printf ("Nome: ");
      gets (umAluno.nome);
      // leitura das notas do aluno. Percorremos o vetor com um for
      int i;
      for (i = 0; i < 3; i++) {
            printf ("Nota %d: ", i + 1);
            scanf ("%f", &umAluno.notas[i]);
      }
      float soma = 0;
      // calculo da media do aluno
      for (i = 0; i < 3; i++) {
            soma += umAluno.notas[i];
      }
      printf ("\n*** Dados do Aluno ***\n");
      printf ("Codigo: %d\n", umAluno.codigo);
      printf ("Nome: ");
      puts (umAluno.nome);
      printf ("Notas: %.1f %.1f %.1f\n", umAluno.notas[0], umAluno.notas[1], umAluno.notas[2]);
      printf ("Media: %.1f\n\n", soma / 3);
      system ("PAUSE");
}
```

Exemplo 3

Neste exemplo, é declarada uma estrutura chamada "livro", que contém os seguintes dados: título, autor, editora e ano de publicação. O programa faz a leitura dos valores para os campos da estrutura e depois os apresenta na tela.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// definiçao da estrutura Livro
struct Livro {
 char titulo[80];
 char autor[80];
 char editora[40];
 int ano;
};
main () {
     // declaracao de uma variavel do tipo Livro
     struct Livro liv;
     // leitura dos dados do livro
     printf ("*** Informe os dados do livro ***\n");
     printf ("Titulo: ");
     gets (liv.titulo);
     printf ("Autor: ");
     gets (liv.autor);
     printf ("Editora: ");
     gets (liv.editora);
     printf ("Ano de Publicacao: ");
     scanf ("%d", &liv.ano);
     // impressao dos dados do livro
     printf ("\n*** Dados do Livro ***\n");
     printf ("Titulo: %s\n", liv.titulo);
     printf ("Autor: %s\n", liv.autor);
     printf ("Editora: %s\n", liv.editora);
     printf ("Ano de Publicacao: %d\n\n", liv.ano);  Prática de Programação
     system ("PAUSE");
}
```

Inicialização de estruturas

Assim como vetores, estruturas podem ser inicializadas no momento de sua declaração. Para isso, basta fornecer os valores iniciais de seus campos entre chaves e separados por vírgula. Os valores devem seguir a ordem em que os campos foram declarados na estrutura.

O exemplo a seguir apresenta a inicialização da estrutura Data.

```c
// definição da estrutura
struct Data {
    int dia, mes, ano;
};
...
// declaração e inicialização da variável hoje
struct Data hoje = {21, 10, 2013};
```

Os valores são atribuídos aos campos da estrutura *hoje* na ordem em que são fornecidos: o campo *dia* recebe o valor 21, o *mes*, 10 e o *ano*, 2013.

Estruturas aninhadas

É possível criar um tipo de estrutura em que um ou mais de seus campos também sejam estruturas, desde que os tipos de tais estruturas tenham sido previamente declarados no programa. Isso é conhecido como estrutura aninhada. A seguir, apresentamos um exemplo deste tipo de estrutura.

```
struct Data {
    int dia, mes, int ano;
};

struct Pessoa {
    char nome[50];
    struct Data dataNasc;
};
```

Note que o campo *dataNasc* é uma estrutura aninhada. Para isso, é necessário que a estrutura *Data* tenha sido definida antes da estrutura *Pessoa*. A próxima figura apresenta a representação gráfica da estrutura *Pessoa*.

Representação gráfica da estrutura Pessoa

Ao acessar campos em estruturas aninhadas, podemos usar o operador de seleção de campo (.) quantas vezes forem necessárias. A seguir, apresentamos um exemplo de acesso aos campos de uma variável do tipo da estrutura*Pessoa*.

```
// declaração da variável da estrutura Pessoa
struct Pessoa umaPessoa;
// atribuição de um nome
strcpy (umaPessoa.nome, "Joao");
// acesso aos campos da estrutura aninhada
umaPessoa.dataNasc.dia = 15;
umaPessoa.dataNasc.mes = 5;
umaPessoa.dataNasc.ano = 1976;
```

Para acessar os campos *dia*, *mes* e *ano*, primeiro temos que acessar o campo *dataNasc*. Por exemplo, seria um erro escrever *umaPessoa.dia*, já que a variável *umaPessoa* é do *tipoPessoa*, e esse tipo não tem nenhum campo chamado *dia*.

Podemos também declarar e inicializar uma estrutura aninhada. Veja o exemplo que usa a estrutura aninhada definida anteriormente:

```
// declaração e inicialização da variável da estrutura Pessoa
struct Pessoa umaPessoa = {"Joao", {15, 5, 1976}};
```

Exemplo 4

A seguir, apresentamos um exemplo completo de um programa em linguagem C que trabalha com estruturas aninhadas. Uma pessoa tem um nome, um endereço e uma data de nascimento. O endereço tem rua, número e bairro. No programa, vamos criar três estruturas *Pessoa*, *Endereco* e *Data*. O programa lê os dados de uma pessoa e depois os apresenta na tela.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// definição da estrutura Data
struct Data {
      int dia, mes, ano;
};
// definição da estrutura Endereco
struct Endereco {
      char rua[100];
      char numero[10];
      char bairro[30];
};
// definição da estrutura Pessoa
struct Pessoa {
      char nome[80];
      struct Endereco endereco;
      struct Data dataNasc;
};
main () {
      // declaração da variavel do tipo Pessoa
      struct Pessoa umaPessoa;
      // leitura do nome
      printf ("Nome: ");
      gets (umaPessoa.nome);
      // leitura do endereco
      printf ("Rua: ");
      gets (umaPessoa.endereco.rua);
      printf ("Numero: ");
      gets (umaPessoa.endereco.numero);
      printf ("Bairro: ");
      gets (umaPessoa.endereco.bairro);
      // leitura da data de nascimento
      printf ("Dia: ");
      scanf ("%d", &umaPessoa.dataNasc.dia);
      printf ("Mes: ");
      scanf ("%d", &umaPessoa.dataNasc.mes);
      printf ("Ano: ");
      scanf ("%d", &umaPessoa.dataNasc.ano);
      // impressao dos dados da pessoa
      printf ("\n\n*** Dados da Pessoa *** \n");
      printf ("Nome: ");
      puts (umaPessoa.nome);
      printf ("Endereco: %s, ", umaPessoa.endereco.rua);
      printf ("%s - ", umaPessoa.endereco.numero);
      printf ("%s\n", umaPessoa.endereco.bairro);
      printf ("Data de Nasscimento: %d/", umaPessoa.dataNasc.dia);
      printf ("%d/", umaPessoa.dataNasc.mes);
      printf ("%d\n\n", umaPessoa.dataNasc.ano);
      system ("PAUSE");
}
```

Exemplo 5

Neste exemplo, é definida uma estrutura para armazenar os dados de um voo (cidades de origem e destino, datas e horários de partida e chegada). Para armazenar as datas e horários, uma estrutura denominada "DataHorario" é criada e utilizada como uma estrutura aninhada à estrutura "voo". O programa faz a leitura dos dados do voo e, por fim, os apresenta na tela.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// definicao da estrutura DataHorario
struct DataHorario {
 int dia, mes, ano;
 int hora, minuto;
};
struct Voo {
 char origem[80];
 char destino[80];
 struct DataHorario partida;
 struct DataHorario chegada;
};
main () {
    // declaração da variavel do tipo Voo
    struct Voo voo;
    // leitura dos dados do voo
    printf ("Cidade de Origem: ");
    gets (voo.origem);
    printf ("Cidade de Destino: ");
    gets (voo.destino);
    printf ("Dia da Partida: ");
    scanf ("%d", &voo.partida.dia);
    printf ("Mes da Partida: ");
    scanf ("%d", &voo.partida.mes);
    printf ("Ano da Partida: ");
    scanf ("%d", &voo.partida.ano);
    printf ("Hora da Partida: ");
    scanf ("%d", &voo.partida.hora);
    printf ("Minutos da Partida: ");
    scanf ("%d", &voo.partida.minuto);
    printf ("Dia da Chegada: ");
    scanf ("%d", &voo.chegada.dia);
    printf ("Mes da Chegada: ");
    scanf ("%d", &voo.chegada.mes);
    printf ("Ano da Chegada: ");
    scanf ("%d", &voo.chegada.ano);
    printf ("Hora da Chegada: ");
    scanf ("%d", &voo.chegada.hora);
    printf ("Minutos da Chegada: ");
    scanf ("%d", &voo.chegada.minuto);

   // impressao dos dados do voo
    printf ("\n*** Dados do Voo ***\n");
    printf ("Cidade de Origem: %s\n", voo.origem);
    printf ("Cidade de Destino: %s\n", voo.destino);
     printf ("Data/Hora da Partida: %d/%d/%d %d:%d\n", voo.partida.dia, voo.partida.m
es, voo.partida.ano, voo.partida.hora, voo.partida.minuto);
     printf ("Data/Hora da Chegada: %d/%d/%d %d:%d\n", voo.chegada.dia, voo.chegada.m
es, voo.chegada.ano, voo.chegada.hora, voo.chegada.minuto);
    system ("PAUSE");
}
```

Passagem de estruturas como parâmetro para funções

Estruturas podem ser passadas como parâmetros para funções da mesma forma que variáveis simples. Para isso, declaramos na assinatura da função um parâmetro do tipo da estrutura que queremos passar. A forma geral é a seguinte:

```
<tipo_retorno><nome_funcao> (struct<nome_estrutura><nome_var>)
```

Por exemplo, imagine que temos uma estrutura denominada *Pessoa* e queremos passá-la como parâmetro para a função *imprimirPessoa*. A assinatura da função fica da seguinte forma:

void imprimirPessoa (struct Pessoa p)

Na chamada da função *imprimirPessoa*, temos que passar como argumento uma variável do tipo *Pessoa*. Analise o trecho de código seguinte:

```
// declaração da tabela do tipo Pessoa
struct Pessoa pessoa;
...
// passagem da tabela como parâmetro para a função imprimirPessoa
imprimirPessoa (pessoa);
```

Exemplo 6

Vamos criar um programa que define uma estrutura “Pessoa” com os campos *nome* e *idade*; ler valores para os campos da estrutura e depois chamar a função *imprimirPessoa*, que apresenta os dados na tela. Note que a estrutura*Pessoa* deve ser definida antes da função que a usa como parâmetro.

```
#include <stdio.h>
#include <stdlib.h>
// definiçao da estrutura Pessoa
struct Pessoa {
      char nome[80];
      int idade;
};
// definição da função imprimirPessoa
void imprimirPessoa (struct Pessoa p) {
      printf ("%s - %d", p.nome, p.idade);
}
main () {
      // declaração da variável do tipo Pessoa
      struct Pessoa pessoa;
      // leitura dos dados de uma Pessoa
      printf ("Nome: ");
      gets (pessoa.nome);
      printf ("Idade: ");
      scanf ("%d", &pessoa.idade);
      printf ("\n");
      // chamada da funçao imprimirPessoa, passando a estrutura comoargumento
      imprimirPessoa (pessoa);
      printf ("\n\n");
      system ("PAUSE");
}
```

Quiz

Exercício

Estruturas

INICIAR

Quiz

Exercício Final

Estruturas

INICIAR

Referências

MIZRAHI, V. V. *Treinamento em linguagem* C, São Paulo: Pearson, 2008.

SCHILDT, H. C — *Completo e Total*. São Paulo: Pearson, 2006.

Avalie este tópico

ANTERIOR
Pesquisa em vetores

Índice

Biblioteca (https://www.uninove.br/conheca-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/)
Portal Uninove (http://www.uninove.br)
Mapa do Site

Ajuda? (https://ava.unidCurso=)