CAPÍTULO 7

Registros (*Structs*)

CAPÍTULO 7

VARIÁVEIS COMPOSTAS HETEROGÊNEAS – REGISTROS (STRUCTS)

Registros são estruturas que podem agrupar diferentes informações gerando novos tipos de dados. Um registro é um conjunto de campos onde cada campo pode ser de um tipo de dado diferente. Por isso os registros são conhecidos como variáveis compostas heterogêneas. A diferença básica dos modelos de vetor e matriz para o modelo de registro é o fato de cada componente no registro poder ser de um tipo de dado diferente.

Por exemplo, uma ficha de cadastro de cliente tem a seguinte formação de campos:

Código do Cliente:
Nome do Cliente:
Endereço:
Saldo:

Esta ficha de cadastro pode ser usada em um programa como sendo um registro contendo cada um dos campos da ficha. Precisamos primeiramente definir em detalhes como é constituído o registro, especificando todos os campos e, depois, declarar uma ou mais variáveis do tipo registro definido. A seguir mostraremos como fazer esta definição.

SINTAXE DA VARIÁVEL COMPOSTA HETEROGÊNEA EM C++ - STRUCT

Em C/C++, um registro é chamado de **struct**, sendo portanto uma estrutura capaz de agrupar diferentes variáveis (os diferentes campos do registro), podendo estas serem de tipos iguais ou diferentes.

Em C/C++, uma **struct** pode ser declarado utilizando a seguinte sintaxe:

```
struct  nome_da_struct
{
        tipo1 campo1;
        tipo2 campo2;
            ...
        tipoN campoN;
};
```

sendo:

- nome_da_struct: nome válido associado ao novo tipo criado;
- campo1**,** campo2**,** ..., campoN são os nomes dos campos que irão compor a struct;
- **tipo1, tipo2, ..., tipoN:** são os tipos de dados associados a cada campo da struct.

Exemplo 1: a ficha de cadastro de cliente vista anteriormente pode ser declarada, como mostrado a seguir:

```
struct fichacad
{
        int codigo;
        char nome[30];
        char endereco[40];
        float saldo;
};
```

Observações:

1) Os campos de uma struct são sempre delimitados por chaves (**{ ... }**).
2) A declaração de uma struct é sempre finalizada com ponto e vírgula (**;**).
3) Ao criar uma struct, estamos na verdade criando um novo tipo de dado, que será utilizado na declaração de uma ou mais variáveis do tipo struct.

DECLARANDO UMA VARIÁVEL DO TIPO STRUCT

A declaração de uma variável do tipo struct pode ser feita de duas formas:
a) após criada a struct, esta é usada como o tipo da variável, na declaração dessa variável;
b) a variável já é declarada juntamente com a criação da struct.

Exemplo 2: declarando uma variável ficha como sendo do tipo fichacad, a struct do exemplo anterior:

(a)

```
struct fichacad
{
        int codigo;
        char nome[30];
        char endereco[40];
        float saldo;
};
fichacad ficha;
```

(b)

```
struct fichacad
{
        int codigo;
        char nome[30];
        char endereco[40];
        float saldo;
} ficha;
```

A variável criada (**ficha**) pode ser acessada normalmente como qualquer outra variável (pelo nome dela, mas é preciso informar o nome da variável e, separado por ponto (**.**), o nome do campo desejado.

SINTAXE: nomevar.nomedocampo

Exemplos:

Como atribuir um valor à uma posição (campo) do REGISTRO:

```
cin >> ficha.codigo;

cin.getline(ficha.nome, 30);

cin.getline(ficha.endereco, 30);
...
ficha.saldo = 100.0;
```

Como atribuir valores a um registro:

```
fichacad ficha = {1, “Jose da Silva”, “Rua 2, no. 55 – Abaeté”, 1200.00};
```

Como se mostra o elemento contido em uma posição (campo) do REGISTRO:

```
cout << ficha.saldo;
```

Exemplo 3: o código a seguir ilustra um exemplo simples de entrada e saída de dados fazendo uso de uma variável (**ficha**) do tipo struct (**fichacad**):

```
int main ( ){
        struct fichacad
        {
                int codigo;
                char nome[30];
                char endereco[40];
                float saldo;
        };
        fichacad ficha; //declaração da variável ficha

        //entrando com os dados da pessoa
        cout<<"Código :";
        cin>>ficha.codigo;
        cin.ignore();
        cout<<"Nome :";
        cin.getline(ficha.nome,30);
        cout<<"Endereço :";
        cin.getline(ficha.endereco,40);
        cout<<"Saldo :";
        cin>>ficha.saldo;

        //mostrando os dados da pessoa
        cout<<"\nDados da pessoa:" << endl;
        cout<<"Código: " <<ficha.codigo << endl;
        cout<<"Nome: " <<ficha.nome << endl;
        cout<<"Endereço: " <<ficha.endereco << endl;
        cout<<"Saldo: "<<ficha.saldo << endl;
        return 0;
}
```

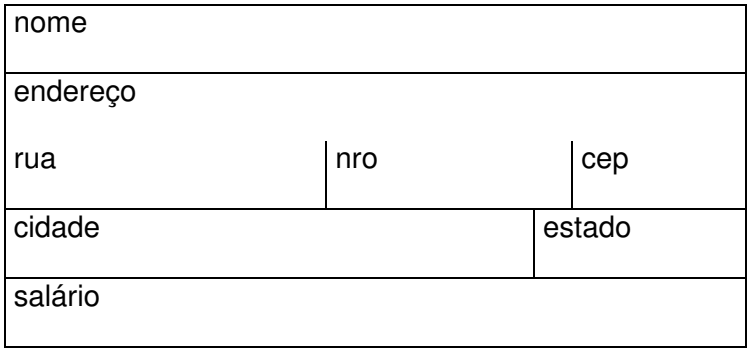
Exemplo 4: Deseja-se, para um dado aluno, ler sua matrícula (**int**) e suas duas notas (**float**) e, a seguir, mostrar sua média (média aritmética simples das duas notas) e sua condição: “APROVADO”, se média >= 60, ou “REPROVADO”, caso contrário.

```
int main ( )
{
        struct ficha                    //declaração da struct
        {
                int nmat;
                float np1, np2, md;
        } aluno;            //declaração da variável aluno junto com a struct

        //entrada de dados
        cout << "Matricula: ";
        cin >> aluno.nmat;
        cout << "Nota 1: ";
        cin >> aluno.np1;
        cout << "Nota 2: ";
        cin >> aluno.np2;

        aluno.md = (aluno.np1 + aluno.np2)/2;   // cálculo da média

        if (aluno.md >= 60)
                cout << "Media: "<<aluno.md<<" - APROVADO!";
        else {
                cout << "Aluno: "<<aluno.nmat;
                cout << "Media: "<<aluno.md<<" - REPROVADO!";
               }

        return 0;
}
```

DECLARANDO UMA STRUCT DENTRO DE STRUCT

O diagrama abaixo mostra um outro tipo de struct:

funcionario



A declaração desta struct pode ser feita da seguinte maneira:

```
struct ende             //struct ende
{
        char rua[40];
        int nro, cep;
};

struct ficha            //struct ficha
{
        char nome[40];
        ende endereco;      //declaração do campo endereço do tipo ende
        char cidade[20], estado[3];
        float salario;
} funcionario;                  //declaração da variável funcionario do tipo ficha
```

Percebe-se que na struct **ficha**, o campo **endereco** é declarado como sendo do tipo **ende**, que também é uma struct, que deve ser definida antes.

A leitura e atribuição de valores para os campos da variável **funcionario** e a saída desses pode ser feita, por exemplo, conforme indicado a seguir:

```
        :
cin.getline(funcionario.nome, 40);
cin>>funcionario.endereco.cep;
        :
funcionario.endereco.nro = 510;
        :
cout << "Nome: "<< funcionario.nome << endl;
cout << "Numero: "<< funcionario.endereco.nro << endl;
cout << "CEP: "<< funcionario.endereco.cep << endl;
        :
```

Exercício 7.1) Considere o registro a seguir e declare uma variável que contenha todos os campos indicados.

regpag

nome					
cpf				rg	
horas trabalhadas no semestre:					
1	2	3	4	5	6
salario					
fgts nos trimestres:					
1,1			1,2		
2,1			2,2		

Exercício 7.2) Se atribuirmos o nome “José da Silva”, o salário de R$ 1.200,00, o valor de 4.5 horas trabalhadas ao mês 4 e o valor de R$ 850,00 de FGTS ao segundo trimestre para o registro **regpag** acima, como ficariam estas atribuições?

VETOR DE REGISTROS (Vetor de Structs)

Note que podemos também declarar vetores ou matrizes do tipo registro, fazendo com que cada elemento do vetor ou matriz seja composto dos campos do registro. Vejamos o exemplo seguinte:

```
struct cli
{
        int cod;
        char nome[30];
        char ende[40];
        float saldo;
} cliente[100];
```

De acordo com a declaração acima, teremos um vetor chamado **cliente**, do tipo cli, contendo 100 posições. Assim, cada posição do vetor possui quatro campos: codigo, nome, endereço e saldo.

Coloque o nome "'Fulano de Tal" na 3ª. posição do vetor de registros **cliente:**

cliente[2].nome = "Fulano de Tal";

	cod	nome	ende	saldo
0				
1				
2		Fulano de Tal		
3				
4				
:				
:				
98				
99				

Exercício 7.3) Escreva o trecho de programa para ler os dados de cadastro dos 100 clientes e armazená-los no vetor **cliente**.

Exercício 7.4) Escreva o trecho de programa que calcule e escreva o total dos saldos de todos os clientes armazenados (suponha que todo o vetor está preenchido).

Exercício 7.5) Escreva o trecho de programa que mostre todos os clientes (código e nome) cujo saldo seja maior que 1500.

Exercício 7.6) Declare uma variável que represente um cheque bancário e que possa armazenar informações de até 500 cheques.

Exercício 7.7) Declare uma variável que represente uma ficha de cadastro em uma loja de eletrodomésticos e que possa armazenar informações de até 1000 clientes.

Exercícios propostos do Capítulo 7:

P7.1) Faça um programa que leia os dados de 100 pessoas segundo o registro abaixo e forneça as seguintes estatísticas:

a) porcentagem de pessoas casadas;
b) idade média da população;
c) idade do homem mais velho.

Faça crítica para o sexo (1-Masc e 2-Fem) e para o estado civil (C-S-V-D).

nome		
sexo	estado civil	idade

P7.2) A prefeitura de uma cidade fez uma pesquisa entre os seus habitantes (N), coletando dados sobre o CPF, o salário, a idade e o número de filhos. Faça um programa que leia esses dados segundo o registro abaixo, calcule e mostre:

a) a média do número de filhos;
b) qual o valor do salário e do CPF da pessoa mais velha.

O número de habitantes (N) é fornecido e deve ser criticado

cpf		
salário	idade	número de filhos

P7.3) Faça um programa que leia um número indeterminado de informações segundo o registro abaixo:

matrícula	nome	nota

Calcule e imprima:

a) a média da turma;
b) o nome do aluno que obteve a menor nota.

Invente um FLAG.

P7.4) Considerando a tabela abaixo:

a) Faça um programa para criar um registro com tal formato, e cadastrar os elementos da tabela.
b) Supondo o vetor já preenchido com as informações, adicione um trecho no mesmo programa que faça a consulta: o usuário fornece o código do cargo (de 1 a 17), o programa mostra o nome e o salário pertencente a este cargo e também mostra os cargos (código e nome do cargo) cujos salários são superiores ao do cargo fornecido.

código	cargo	salário
1	Analista de salários	1.500,00
2	Auxiliar de contabilidade	1.100,00
3	Contador	2.000,00
4	Gerente	3.000,00
:	:	:
:	:	:
17	Secretária	850,00

P7.5) Uma determinada biblioteca possui obras de ciências exatas, ciências humanas e ciências biomédicas, totalizando 1500 volumes, sendo 500 volumes de cada área. O proprietário resolveu informatizá-la e para tal agrupou as informações sobre cada livro do seguinte modo:

Código da Catalogação:	Doado:
Nome da Obra:	
Nome do Autor:	
Editora:	No. de Páginas:

a) Declare tal estrutura e que reúna/entre com todas as informações de todas as áreas em vetores distintos para cada área.

b) Elabore um trecho de programa que faça a seguinte consulta: o usuário fornece o código da obra e a área; existindo tal livro, o programa mostra seus campos; caso contrário, envia uma mensagem de inexistência. A consulta se repete até que o usuário digite o código da obra igual a –1 (FLAG).

c) Escreva um trecho de programa que liste todas as obras das áreas de exatas e biomédicas que representam livros doados.