

Aula 10: Registros (Estruturas)

Professor(a): João Eduardo Montandon (103)

Virgínia Fernandes Mota (106)

jema.f.github.io

<http://www.dcc.ufmg.br/~virginiaferm>

INTRODUÇÃO A PROGRAMAÇÃO - SETOR DE INFORMÁTICA



Estruturas de dados heterogêneas

- Até agora vimos as estruturas de dados homogêneas: **vetores**, **matrizes** e **strings**.
- Nestas estruturas todos os elementos da estrutura são de tipos de dados primitivos: inteiro, real, caractere.
- No entanto, em muitos casos, necessitamos armazenar um conjunto de informações relacionadas, formado por diversos tipos de dado primitivos.
- Exemplo:
 - Endereço
 - Fichas com dados pessoais de um cliente
 - Fichas com dados de um produto

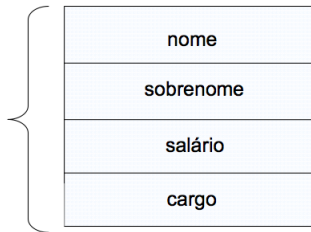
Variáveis Compostas Heterogêneas

- Quando uma determinada estrutura de dados for composta por **diversos tipos diferentes**, primitivos ou não, temos um conjunto heterogêneo de dados.
- Essas variáveis são chamadas de variáveis compostas heterogêneas.
- Estas variáveis compostas são chamadas de **estruturas** ou **registros** (ou *structs* em C).

Definição de Estrutura

- Uma estrutura pode ser definida como uma coleção de uma ou mais variáveis relacionadas (campos), onde **cada variável pode ser de um tipo distinto**.

empregado
(4 campos)



nome
sobrenome
salário
cargo

- Sintaxe para definir uma estrutura com **n** campos em C:

```
1 struct nome_estrutura {  
2     tipo1 identificador1;  
3     tipo2 identificador2;  
4     ...  
5     tipon identificadorn;  
6 };
```

- Em C, pode-se criar **m** variáveis de uma dada estrutura de duas formas:

```
struct nome_estrutura VAR_1, VAR_2,...,VAR_M;
```

ou

```
typedef struct nome_estrutura novo_nome_estrutura;  
novo_nome_estrutura VAR_1, VAR_2,...,VAR_M;
```

- Comando **typedef** é usado para definir uma novo nome para a estrutura.

- Exemplo de estrutura que armazenaria a matrícula e o nome de um funcionário.

```
1 //Definição
2 struct funcionario {
3     int matricula;
4     char nome[30];
5 };
6
7 //Declaração
8 struct funcionario f1;
```

Estruturas: Definição e Declaração

- Em geral, a definição de um tipo estrutura deve ficar **fora** do programa (principal) e de qualquer sub-rotina.
- A declaração de uma variável do tipo estrutura deve ficar **dentro** do programa (principal) e/ou **dentro** de qualquer sub-rotina.

- Campos ou membros de uma estrutura podem ser usados da mesma forma como as variáveis.
- Campos são acessados usando o operador de acesso ponto (.) entre o **nome da estrutura** e o **nome do campo**.
- Para modificar um campo de uma estrutura, basta usarmos novamente o operador (.).
`scanf("%d", &f1.matricula);`

Exemplo 1: Leitura dos dados da estrutura funcionário

```
1 #include <stdio.h>
2 struct funcionario {
3     int matricula;
4     char nome[30];
5 };
6
7 int main(){
8     struct funcionario f1;
9     scanf("%d", &f1.matricula);
10    gets(f1.nome);
11    puts("Informacoes armazenadas: \n");
12    printf("%d", f1.matricula);
13    puts(f1.nome);
14    return 0;
15 }
```

Exemplo 2: Estrutura para armazenar endereço

```
1 struct est_endereco {  
2     char rua[50];  
3     int numero;  
4     char bairro[20];  
5     char cidade[30];  
6     char sigla_estado[3];  
7     int cep;  
8 };  
9  
10 struct est_endereco end1;
```

Exemplo 3: Estrutura para armazenar endereço (outra maneira)

```
1 typedef struct est_endereco {  
2     char rua[50];  
3     int numero;  
4     char bairro[20];  
5     char cidade[30];  
6     char sigla_estado[3];  
7     int cep;  
8 }endereco;  
9  
10 endereco end1;
```

- typedef: Pode-se utilizar o modificador de tipo na criação da estrutura.
- A estrutura passará a ser referenciada pelo nome que aparece no final da definição.
- A criação de variáveis fica bastante facilitada dessa forma.

Exemplo 4: Definição de uma estrutura onde um de seus campos é outra estrutura (endereco):

```
1 typedef struct est_ficha_pessoal {  
2     char nome[50];  
3     int telefone;  
4     endereco end;  
5 } ficha_pessoal;  
6  
7 ficha_pessoal ficha1, ficha2;
```

- Para acessarmos o campo telefone da variável ficha1 do tipo ficha_pessoal (tipo estrutura), devemos usar a seguinte sintaxe:
ficha1.telefone = 1234567;
- Para acessar os campos da estrutura interna (end), podemos fazer da seguinte forma:
ficha1.end.rua = "Rua das Flores";

Exemplo 5: Tipo estrutura possui vetores como um dos seus campos.

```
1 typedef struct est_ficha_pessoal {  
2     char nome[50];  
3     int telefone;  
4     endereco end;  
5 } ficha_pessoal;  
6  
7 ficha_pessoal ficha1, ficha2;
```

- O acesso a estes campos é feito da mesma maneira como acesso direto a um vetor.
ficha1.nome[1] = 'A';

Exemplo 6: Inicializando uma estrutura

```
1 typedef struct est_ficha_pessoal {  
2     char nome[50];  
3     int telefone;  
4 }ficha_pessoal = {"Zé das Couves", 1234567};  
5  
6 ficha_pessoal ficha1;
```

- Desta forma, a variável ficha1 inicializará com os campos preenchidos com "Zé das Couves" e 1234567.

- Uma das vantagens ao utilizarmos estruturas é a possibilidade de copiarmos toda a informação de uma estrutura para outra do mesmo tipo com uma atribuição simples:

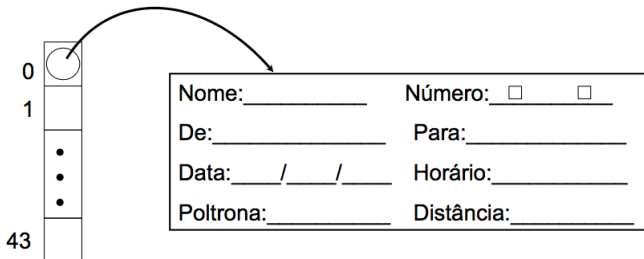
```
1 typedef struct coordenadas {  
2     int x;  
3     int y;  
4 }coordenadas;  
5  
6 coordenadas coord1, coord2;  
7 coord1.x = 10;  
8 coord1.y = 20;  
9 coord2 = coord1;
```


- Como qualquer outra variável, uma variável do tipo estrutura pode ser usada como parâmetro.
- Também, uma variável do tipo estrutura pode ser passada para uma subrotina por referência ou por valor.

- Pode-se criar vetores de estruturas como se criam vetores de tipos primitivos.
- Até o momento só fizemos menção a uma única instância da estrutura.
- é necessário possuir uma definição da estrutura antes de declarar um vetor de estrutura.

- Suponha que deseja-se manter um registro de informações relativas a passagens rodoviárias de todos lugares (poltronas) de um ônibus.
- Pode-se utilizar uma estrutura referente a cada poltrona (passagem) e para agrupar todas elas utiliza-se um vetor de estruturas.

- Um ônibus possui 44 lugares numerados de 0 a 43:



Vetores de estruturas

```
1 typedef struct reg_passagem{
2     char NOME[50];
3     int NUMERO;
4     char ORIGEM[20];
5     char DESTINO[20];
6     char DATA[8];
7     char HORARIO[5];
8     int POLTRONA;
9     float DISTANCIA;
10 }passagem;
11
12 passagem VET_PASSAGEM[44];
```

- Para acessar: `VET_PASSAGEM[3].NUMERO = 32;` // O campo numero da passagem da posição 3 recebe 32

Faça um programa que leia o nome e as 4 notas escolares de 8 alunos. Imprima a listagem dos alunos, com suas notas e a média das mesmas.

- Crie um procedimento para leitura, um procedimento para impressão e uma função para o cálculo da média.

Exemplo

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 typedef struct ref_aluno{
4     char nome[40];
5     float nota[4];
6 }aluno;
7 float calculaMedia(float notas[], int n){
8     int i;
9     float media = 0.0;
10    for (i = 0; i < n; i++)
11        media += notas[i];
12    return media/(float)n;
13 }
14 void leVetorAlunos(aluno a[], int n){
15     int i, j;
16     for (i = 0; i < n; i++){
17         printf("Informe o nome do aluno \n");
18         gets(a[i].nome);
19         printf("Informe as quatro notas \n");
20         for (j = 0; j < 4; j++)
21             scanf("%f %c", &a[i].nota[j]);
22     }
23 }
```

Exemplo

```
1 void imprimeVetorAlunos(aluno a[], int n){
2     int i,j;
3     printf("\n***Alunos, Notas e Media***\n");
4     for (i = 0; i < n; i++){
5         printf("\n %20s ", a[i].nome);
6         for (j = 0; j < 4; j++)
7             printf("%.2f ", a[i].nota[j]);
8         printf("%.2f", calculaMedia(a[i].nota, 4));
9     }
10 }
11
12 int main(){
13     aluno alunos[8];
14     leVetorAlunos(alunos, 8);
15     imprimeVetorAlunos(alunos, 8);
16     return 0;
17 }
```


Alocação Dinâmica e Estruturas

- Para acessar os elementos de um registro através de um ponteiro, devemos primeiro acessar o registro e depois acessar o campo desejado.
- Os parênteses são necessários pois o operador `*` tem prioridade menor que o operador `.`

```
1 typedef struct ref_ponto{
2     float x;
3     float y;
4 }ponto;
5
6 ponto *ponteiro_ponto;
7 ponteiro_ponto = (ponto*) malloc (sizeof(ponto));
8 (*ponteiro_ponto).x = 4.0;
9 (*ponteiro_ponto).y = 3.4;
10
11 free(ponteiro_ponto);
```

- Para simplificar o acesso aos campos de um registro através de ponteiros, foi criado o operador \rightarrow
- Usando este operador acessamos os campos de um registro diretamente através do ponteiro.

```
1 typedef struct ref_ponto{
2     float x;
3     float y;
4 }ponto;
5
6 ponto *ponteiro_ponto;
7 ponteiro_ponto = (ponto*) malloc (sizeof(ponto));
8 ponteiro_ponto->x = 4.0;
9 ponteiro_ponto->y = 3.4;
10
11 free(ponteiro_ponto);
```

- A mesma idéia pode ser usada para alocação dinâmica de vetores de estruturas.
- O acesso a cada elemento irá ocorrer pelo uso do operador `.`

```
1 typedef struct ref_ponto{
2     float x;
3     float y;
4 }ponto;
5
6 ponto *ponteiro_ponto;
7 ponteiro_ponto = (ponto*) malloc (4*sizeof(ponto));
8 ponteiro_ponto[1].x = 4.0;
9 ponteiro_ponto[1].y = 3.4;
10
11 free(ponteiro_ponto);
```

Alocação Dinâmica e Estruturas

- E se elementos da estrutura forem do tipo ponteiro?
- Deve-se alocar e desalocar cada elemento!

```
1 typedef struct ref_ponto{
2     float *pontinhos;
3 }ponto;
4
5 int i;
6 ponto *ponteiro_ponto;
7 ponteiro_ponto = (ponto*) malloc (4*sizeof(ponto));
8
9 for (i = 0; i < 4; i++)
10     ponteiro_ponto[i].pontinhos = (float *) malloc (2*sizeof(float)
11     );
12
13 for (i = 0; i < 4; i++){
14     ponteiro_ponto[i].pontinhos[0] = i;
15     ponteiro_ponto[i].pontinhos[1] = 2*i;
16 }
17
18 for (i = 0; i < 4; i++)
19     free(ponteiro_ponto[i].pontinhos);
20 free(ponteiro_ponto);
```

1. Faça um programa para leitura, via teclado, dos dados de um aluno. Os dados a serem guardados na estrutura aluno são os seguintes: nome, curso, idade. Ao final, imprima estas informações na tela.
2. Considere a mesma estrutura definida anteriormente. Acrescente à estrutura um vetor com as notas das três provas feitas pelo aluno, calcule a sua média e diga se ele foi aprovado ou não (media \geq 60).
3. Altere o programa do exercício 2 para que ele leia as informações de N alunos. Imprima a média de cada aluno e a média geral da turma.
4. Acrescente um procedimento ao exercício 3 que ordene a estrutura de alunos em ordem crescente da média das notas. Ao final, imprima todas as informações na tela (ordenadas pela média das notas).

Na próxima aula...

Arquivos