LABORATÓRIO DE ALGORITMOS E ESTRUTURAS DE DADOS I

Prof. Caio César de Freitas Dantas

Registros

Declaração de um registro

Os registros em C são definidos pela palavra reservada struct

```
struct nome_da_estrutura{
    tipo_dado1 nome_dado1;
    tipo_dado2 nome_dado2;
            ...
};
```

Registros

Declaração de um registro

Os registros em C são definidos pela palavra reservada struct

```
struct conta{
      int nro_conta;
      char nome[50];
      float saldo;
};
```

Registros

- A partir da estrutura definida, o programa poderá considerar que existe um novo tipo de dado para ser utilizado, chamado conta.

- Esse tipo de dado é capaz de armazenar o número da conta, o nome do cliente e o saldo da conta.

```
struct conta{
    int nro_conta;
    char nome[50];
    float saldo;
};
```

Registros

Declaração de uma variável do tipo registro

- Para que uma estrutura possa ser utilizada é necessário fazer uma declaração do tipo de dado criado.

```
struct nome_da_estrutura nome_da_variável;
```

- Variável do tipo conta.

```
struct conta minha_conta;
```

Registros

- Já que essa estrutura representa um novo tipo de dado, todas as operações realizadas como os tipos predefinidos da linguagem também podem ser realizados por ela.

- Dessa maneira variáveis, vetores e matrizes também podem ser declaradas como sendo do tipo conta.

Registros

- A variável vet é um vetor de dez posições e, em cada posição, serão armazenados um número de conta, um nome e um saldo.

```
struct conta vet[10];
```

- A variável mat é uma matriz de com 5 linhas e 4 colunas, onde em cada célula serão armazenados um número de conta, um nome e um saldo.

```
struct conta mat[5][4];
```

Registros

Também é possível fazer a declaração da variável junto com a definição da estrutura, no mesmo bloco de comandos

```
struct conta{
      int nro_conta;
      char nome[50];
      float saldo;
}x[10][8];
```

Registros

Acesso a membros do registro

Depois que uma variável registro é declarada, o programa poderá manipular o seu conteúdo, ou seja, os valores armazenados em cada campo de sua estrutura.

Para isso utilizamos individualmente esses campos como se fossem variáveis normais. A sintaxe é:

```
variavel_registro.nome_do_campo;
```

Registros

Acesso a membros do registro

```
//utilizando variável da estrutura
struct conta minha_conta;
minha_conta.nro_conta = 200;
strcpy(minha_conta.nome,"Joao Carlos");
minha_conta.saldo = 1200.55;
```

Registros

Acesso a membros do registro

```
//utilizando vetor da estrutura
struct conta minha_conta[10];
minha_conta[2].nro_conta = 300;
strcpy(minha_conta[2].nome,"Marta Oliveira");
minha_conta[2].saldo = 910.30;
```

Registros

Acesso a membros do registro

```
//utilizando matriz da estrutura
struct conta minha_conta[5][5];
minha_conta[4][3].nro_conta = 10;
strcpy(minha_conta[4][3].nome,"Pedro Henrique");
minha_conta[4][3].saldo = 595.15;
```

Registros

Lendo e escrevendo Registros

A leitura dos campos de um registro deve ser feita campo a campo, como se fossem variáveis independentes. A mesma coisa vale para a escrita.

```
struct Aluno{
    int codigo;
    char nome[200];
    float nota1;
    float nota2;
        };
```

```
struct Aluno aluno_especial;
```

```
printf(" Digite o codigo do aluno especial: ");
scanf("%d", &aluno_especial.codigo);

printf(" Digite o nome do aluno especial: ");
scanf("%s", &aluno_especial.nome);

printf(" Digite a nota 1 do aluno especial: ");
scanf("%f", &aluno_especial.nota1);

printf(" Digite a nota 2 do aluno especial: ");
scanf("%f", &aluno_especial.nota2);
```

Registros

Atribuição de Registros

Podemos atribuir um registro a outro diretamente:

```
var1_registro = var2_registro;
```

Automaticamente é feito uma cópia de cada campo de var2_registro para var1_registro.

Registros

Atribuição de Registros

```
struct Aluno{
    int codigo;
    char nome[200];
    float nota1;
    float nota2;
        };
```

```
struct Aluno a, b;
```

```
// b recebe todo o conteúdo de a.
b = a;
```

```
printf(" Digite o codigo do aluno especial: ");
scanf("%d", &a.codigo);

printf(" Digite o nome do aluno especial: ");
scanf("%s", &a.nome);

printf(" Digite a nota 1 do aluno especial: ");
scanf("%f", &a.nota1);

printf(" Digite a nota 2 do aluno especial: ");
scanf("%f", &a.nota2);
```

Registros

Registros Aninhados

```
struct ALUNOS { // Registro Alunos
    char NOME[30];
    int MATRI;
    float N1;
    float N2;
}; // Fim registro Alunos
```

```
struct NOTAS {
        int N1;
        int N2;
        } ;

struct ALUNOS {
    char NOME[30];
    int MATRI;
    NOTAS N;
    };
```

Registros

1. O Tribunal Eleitoral necessita controlar os dados dos candidatos da última eleição. Para isso… Defina um tipo registro para o cadastro de candidatos a uma eleição composto dos seguintes campos: nome, endereço, partido, cargo e número de votos.

2. Faça um algoritmo que use uma variável do tipo registro criado e leia dados de um candidato para esta variável e os escreva.

3. Agora modifique o tipo registro para incluir o campo data da eleição que é um tipo registro composto dos campos dia, mês e ano.

4. Faça um algoritmo que use uma variável do tipo registro criado e leia e escreva os dados relativos ao dia da eleição.

5. E se quiséssemos guardar dados de 200 candidatos? Que estrutura de dados poderíamos usar? Faça um algoritmo que guarde a informação de 200 candidatos.

Registros

Crie uma estrutura representando os alunos de um determinado curso. A estrutura deve conter a matrícula do aluno, nome, nota da primeira prova, nota da segunda prova e nota da terceira prova.

a) Permita ao usuário entrar com os dados de 5 alunos.
b) Encontre o aluno com maior nota da primeira prova.
c) Encontre o aluno com maior média geral.
d) Encontre o aluno com menor média geral
e) Para cada aluno diga se ele foi aprovado ou reprovado, considerando o valor 6 para aprovação.

FIM!