Registro em C (struct)

Disciplina: PROGRAMAÇÃO II

Prof. Jean Eduardo Glazar
Curso de Sistemas de Informação
Campus Colatina-ES

INSTITUTO FEDERAL
Espírito Santo

Definição

- **Registro** é uma estrutura de dados heterogênea, ou seja, agrupa diversas variáveis, que podem ser de tipos diferentes, em somente uma variável.

- Os registros são uma forma de declaração de **tipos de dados** definidos pelo usuário, ou seja, uma vez definida uma estrutura, podemos declarar variáveis daquela estrutura.

Declaração de Registros

```
struct <nome_do_tipo_do_registro> {
        <tipo1> variável_1;
        <tipo2> variável_2;
        . . .
        <tipoN> variável_N;
};
```

Declaração de variável do tipo Registro

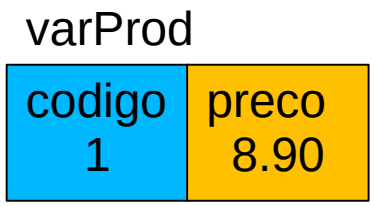
Na declaração anterior foi criado um **novo tipo** em nosso programa, portanto, pode-se agora **declarar variáveis** desse tipo. A declaração de variáveis para o tipo struct é:

struct <nome_do_tipo_do_registro> variável;

Tipo do registro declarado anteriormente

Registros – exemplo

```
struct Produto {
        int codigo;
        float preco;
};
int main() {
        struct Produto varProd;
        varProd.codigo = 1;
        varProd.preco = 8.90;
        printf("%d - R$ %.2f", varProd.codigo, varProd.preco);
}
```



Registros – outros exemplos

```
struct Aluno {
        int matricula;
        char nome[100];
        float coeficiente;
};

struct Cliente {
        int codigo;
        char nome[100];
        char telefone[50];
        char cpf[14];
};
```

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

struct Arquivo {
        char nome[100];
        char extensao[3];
        int tamanho;
};
```

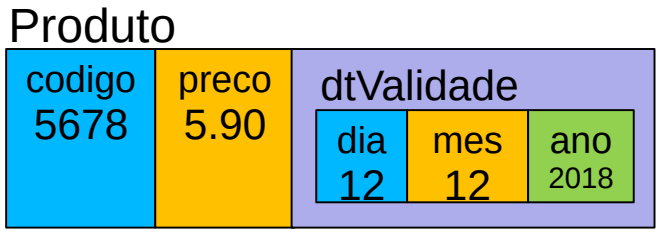
Registros – exemplo cont.

```
int main() {
        struct Aluno a;
        struct Cliente cli;
        struct Data dtNasc;
        struct Arquivo arq;

        a.matricula = 1234;
        strcpy(cli.nome, "Cicrano");
        dtNasc.dia = 13;
        arq.tamanho = 1024;
}
```

Structs aninhadas

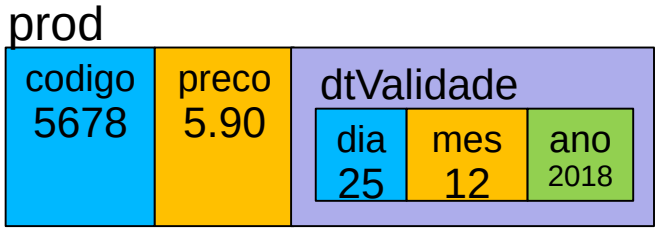
- Um registro pode fazer parte de outro registro. Exemplo:

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

struct Produto {
        int codigo;
        float preco;
        struct Data dtValidade;
};
```



Structs aninhadas

```
int main() {
      struct Produto prod;
      prod.codigo = 5678;
      prod.preco = 5.90;
      prod.dtValidade.dia = 25;
      prod.dtValidade.mes = 12;
      prod.dtValidade.ano = 2018;
      ...
}
```



Diversas variáveis

- Pode-se criar quantas variáveis desejar para cada tipo struct.

```
int main() {
        struct Aluno a1, a2, a3;

        a1.matricula = 1234;
        a2.matricula = 5678;
        a3.matricula = 9090;
        . . .
}
```

a1

matricula	nome	coef.
1234	“Bia”	8.90

a2

matricula	nome	coef.
5678	“Amy”	7.90

a3

matricula	nome	coef.
9090	“Ana”	8.50

Comando: typedef

- O comando ***typedef*** permite ao programador definir um novo nome para um determinado tipo. Sua forma geral é:

```
typedef nome_antigo novo_nome;
```

- Como exemplo vamos dar o nome de **inteiro** para o tipo **int**:

```
typedef int inteiro;
```

- Agora podemos declarar o tipo inteiro.

```
int main() {
    inteiro a, x, idade;
    ...
}
```

Comando: typedef

- O comando ***typedef*** também pode ser utilizado para dar nome a tipos complexos, como as estruturas (structs):

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

```
struct Produto {
        int codigo;
        float preco;
        Data dtValidade;
};
typedef struct Produto Produto;
```

```
int main() {
        Produto prod;
        ...
}
```

Passagem de Structs por parâmetro

- Segue a mesma lógica das passagens de variáveis por valor e por referência. Exemplo por valor:

```
void imprimir ( Aluno pa ) {
        printf("%d - %s \n", pa.matricula, pa.nome);
}

int main() {
        Aluno a;
        a.matricula = 1234;
        strcpy (a.nome, "Cicrano");
        imprimir(a);
}
```

Passagem de Structs por parâmetro

- Exemplo de passagem de parâmetro por referência:

```
void lerAluno ( Aluno *pa ) {
        printf("Matricula: ");
        scanf("%d", &(*pa).matricula );
}

int main() {
        Aluno a;
        lerAluno(&a);
        imprimir(a);
}
```

pa->

Quando temos ponteiro para struct.

```
scanf("%d", &pa->matricula );
```

Vetor de structs

```
int main() {
      Aluno alunos[10];
      int i;
      for (i=0; i < 10; i++) {
          printf("Nome: ");
          scanf("%[^\n]s", &alunos[i].nome);
          printf("Matricula: ");
          scanf("%d", &alunos[i].matricula);
          printf("Coeficiente: ");
          scanf("%f", &alunos[i].coeficiente);
      }
      ...
}
```

alunos

	matricula	nome	coef.
0	1234	“Bia”	8.90
1	5678	“Amy”	7.90
2	9090	“Ana”	8.50
...	...	...	...

INSTITUTO
FEDERAL
Espírito Santo