Disciplina: Linguagem de Programação II
Professora: Caroline Mazetto Mendes

Aula 12 - 25/04/17

1) Registros

Um registro é uma estrutura de dados heterogênea, ou seja, formada por um conjunto de variáveis (campos) que podem ser de tipos diferentes, incluindo tipos compostos (vetores, matrizes e outros registros).

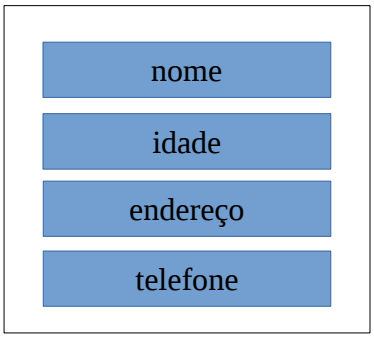

Figura 1. Registro que contém os campos nome, idade, endereço e telefone.

Na linguagem C, um registro pode ser representado por **struct**. Em geral, struct são usados para criar um novo tipo de dados composto. Um registro é declarado usando a palavra reservada **struct**, enquanto um novo tipo de dados é definido usando a palavra reservada **typedef**. Sintaxe:

```
typedef struct nome_do_registro {
        tipo1 nome_do_campo1;
        tipo1 nome_do_campo2;
        …
        tipoN nome_do_campoN;
}nome_do_tipo;
```

Um novo tipo de dados deve ser declarado após as diretivas, antecedendo a declaração das funções. O exemplo abaixo mostra a declaração de um struct chamado Pessoa, contendo os campos nome e idade. A partir desse struct, um novo tipo de dados, chamado TPessoa (abreviação para Tipo Pessoa), é definido.

```
typedef struct Pessoa {
        char nome[100];
        int idade;
}TPessoa;
```

Uma vez que um novo tipo de dados foi definido, pode-se declarar variáveis desse tipo. O exemplo abaixo mostra a declaração de uma variável chamada *pessoa* do tipo *TPessoa*:

```
TPessoa pessoa;
```

O acesso ao campo de um registro é realizado informando o nome da variável e nome do campo separado por um ponto, por exemplo:

```
pessoa.idade = 20;
```

Exemplo: Atribuir valores aos campos de um registro e imprimir os dados.

```
#include <stdio.h>
#include <string.h>
#define MAX_STRING 100

typedef struct Pessoa {
        char nome[MAX_STRING];
        int idade;
}TPessoa;

int main(void) {
        TPessoa pessoa;

        strcpy(pessoa.nome, “Maria”);
        pessoa.idade = 20;

        printf(“Dados da pessoa:\n”);
        printf(“nome = %s\n”, pessoa.nome);
        printf(“idade = %d\n”, pessoa.idade);
}
```

Código 1. Exemplo de atribuição de valores a um registro.

Exemplo: Preencher um registro com valores informados pelo usuário e imprimir os dados.

```
#include <stdio.h>
#include <string.h>
#define MAX_STRING 100

typedef struct Pessoa {
        char nome[MAX_STRING];
        int idade;
}TPessoa;

int main(void) {
        TPessoa pessoa;

        printf(“Entre com o nome:\n”);
        fgets(pessoa.nome, MAX_STRING-1, stdin);
        printf(“Entre com a idade:\n”);
        scanf(“%d”, &pessoa.idade);

        getchar();      /* para descartar o \n não lido pelo scanf() */

        printf(“Dados da pessoa:\n”);
        printf(“nome = %s”, pessoa.nome);
        printf(“idade = %d\n”, pessoa.idade);
}
```

Código 2. Exemplo de leitura de um registro.

Registro como parâmetro de funções

Pode-se passar um registro como parâmetro para uma função usando passagem de parâmetros por valor ou por referência, assim como realizado com variável simples.

Na passagem de parâmetros por valor, o acesso aos campos de um registro é realizado informando o nome do registro e nome do campo separado por um ponto. Já na passagem de parâmetros por referência, usa-se o operador -> ao invés do ponto.

Exemplo: Preencher um registro com valores informados pelo usuário e imprimir os dados.

```
#include <stdio.h>
#include <string.h>
#define MAX_STRING 100

typedef struct Pessoa {
	char nome[MAX_STRING];
	int idade;
}TPessoa;

void preenche_registro(TPessoa *pessoa) {

	printf(“Entre com o nome:\n”);
	fgets(pessoa->nome, MAX_STRING-1, stdin);
	printf(“Entre com a idade:\n”);
	scanf(“%d”, &pessoa->idade);

	getchar();
}

void imprime_registro(TPessoa pessoa) {

	printf(“Dados da pessoa:\n”);
	printf(“nome = %s”, pessoa.nome);
	printf(“idade = %d\n”, pessoa.idade);
}

int main(void) {
	TPessoa pessoa;

	preenche_registro(&pessoa);
	imprime_registro(pessoa);
}
```

Código 3. Programa que preenche e imprime registros usando funções.

2) Exercícios

- Implementar e testar os Códigos 1, 2 e 3.

3) Atividade Avaliativa 1

1. Faça um programa em C para ordenar dois cadastros pelo nome. Cada cadastro deve ser representado por um registro, contendo os campos nome e idade. O programa deve realizar a leitura de dois cadastros e imprimi-los em ordem crescente.

Para resolver o problema proposto, construa uma função para realizar a leitura de um cadastro e uma outra função para imprimir um cadastro. Adicionalmente, construa uma função para realizar a ordenação de dois cadastros. Essa função deve acionar (chamar) a função que imprime um cadastro.

2. Faça em programa em C para ordenar duas datas. Cada data deve ser representada por um registro, contendo os campos dia, mês e ano. O programa deve realizar a leitura de duas datas e imprimi-las em ordem crescente.

Para resolver o problema proposto, construa uma função para realizar a leitura de uma data e outra função para imprimir uma data. Adicionalmente, construa uma função para realizar a ordenação de duas datas. Essa função deve acionar (chamar) a função que imprime uma data.