# Linguagem de Programação C

## Registros

Cristiano Lehrer

http://www.ybadoo.com.br/

# Conceito de registro (1/4)

- Vetores e matrizes:
  - Estruturas de dados homogêneas.
  - Armazenam vários valores, mas todos de um mesmo tipo.
- Problemas reais:
  - Temos coleções de dados que são de tipos diferentes.
  - Exemplo – ficha de cadastro de um cliente:
    - Nome: `string`
    - Endereço : `string`
    - Telefone: `long`
    - Salário: `float`
    - Idade: `int`

# Conceito de registro (2/4)

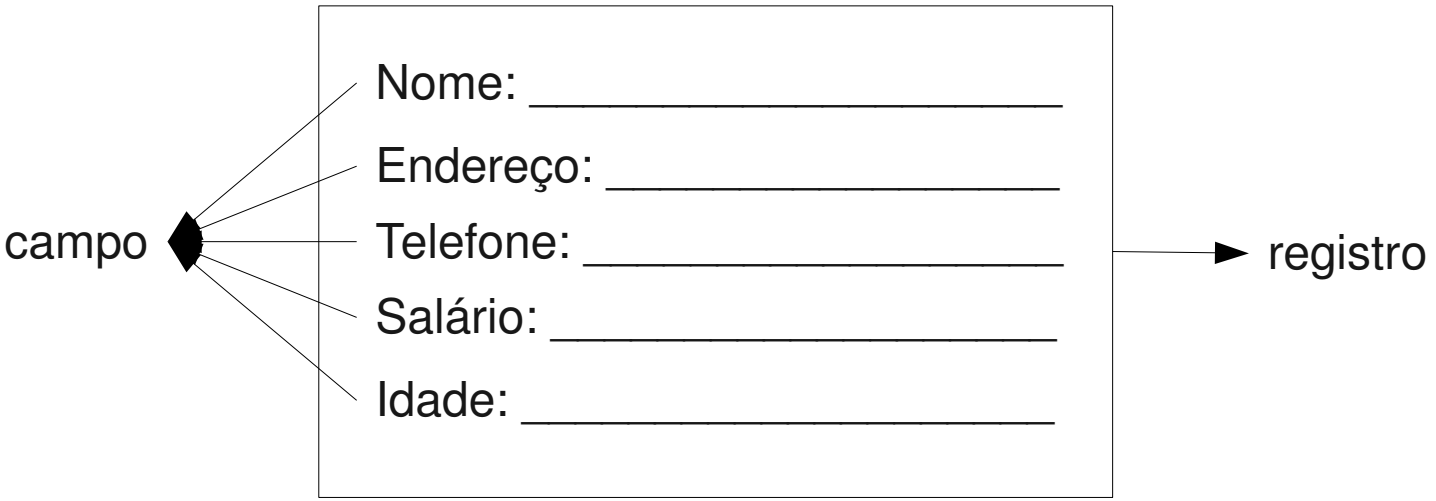
- Registro (ou `struct`):
  - Tipo de dado estruturado heterogêneo:
    - Coleção de variáveis referenciadas sobre um mesmo nome.
  - Permite agrupar dados de diferentes tipos numa mesma estrutura (ao contrário de matrizes que possuem elementos de um mesmo tipo):
    - Cada componentes de um registro pode ser de um tipo diferente.
    - Estes componentes são referenciados por um nome.

# Conceito de registro (3/4)

- Os elementos do registro:
  - São chamados de campos ou membros da `struct`.
- É utilizado para armazenar informações de um mesmo objeto.
- Exemplos:
  - Carro → cor, marca, ano, placa, chassi
  - Pessoa → nome, idade, endereço

# Conceito de registro (4/4)

- Campo (*field*):
  - Conjunto de caracteres com o mesmo significado.
  - Exemplo: nome
- Registro (*struct* ou *record*):
  - Conjunto de campos relacionados.
  - Exemplo: nome, endereço, telefone, salário e idade de uma pessoa.



# Sintaxe na Linguagem C (1/5)

- A palavra reservada `struct` indica ao compilador que está sendo criada uma estrutura.
- Uma estrutura deve ser declarada após incluir as bibliotecas e antes do `main`.

```
struct <identificador_struct> {
  tipo <nome_variável_campo1>;
  tipo <nome_variável_campo2>;
  ...
} <var1>, <var2>;
```

```
struct <identificador_struct> <var1>, <var2>;
```

# Sintaxe na Linguagem C (2/5)

- Se o compilador C for compatível com o padrão C ANSI:
  - Informação contida em uma `struct` pode ser atribuída a outra `struct` do mesmo tipo.
  - Não é necessário atribuir os valores de todos os elementos/campos separadamente.
  - Por exemplo:
    - `<var1> = <var2>;`
    - Todos os campos de `<var1>` receberão os valores correspondentes dos campos de `<var2>`.
  - Para acessar os campos da `struct`:
    - Utiliza-se o nome da variável `struct`, seguida de ponto, seguido do nome do campo.
    - Por exemplo:
      - `<var1>.<nome_variável_campo1>;`

# Sintaxe na Linguagem C (3/5)

- Por exemplo, um `struct endereco` que guarda os elementos nome, rua, cidade, estado e cep.

```
struct endereco {
  char nome[30];
  char rua[40];
  char cidade[20];
  char estado[2];
  long int cep;
};
```

- Foi feita apenas a declaração da `struct`, ainda não foi criada nenhuma variável da `struct endereco`.
- O comando para declarar uma variável com esta `struct` é:

```
struct endereco info_end;
```

# Sintaxe na Linguagem C (4/5)

- Já vimos que para acessar os membros de uma `struct` deve-se usar `nome_variável.nome_membro`
- Portanto, considerando o exemplo anterior:
  - Para inicializar o `cep` da variável `info_end` que é uma variável da `struct endereco` se faria:

```
info_end.cep = 123456;
```

  - Para obter o nome da pessoa e colocar na *string* `nome` da `struct` poderia utilizar:

```
gets(info_end.nome);
```

  - Para percorrer toda a *string* `rua` seria:

```
for(int i = 0; i < 40; i++)
  printf("%c", info_end.rua[i]);
```

# Sintaxe na Linguagem C (5/5)

```
struct aluno {
  char nome[40];
  float A1;
  float A2;
  float A3;
  int faltas;
};

int main()
{
  struct aluno fulano, ciclano;
  fulano.P1 = 9.5;
  fulano.P2 = 8.5;
  fulano.P3 = 9.0;
  fulano.faltas = 4;
  ciclano = fulano;
}
```

# Vetor de Struct (1/2)

- O uso mais comum de `struct` é em vetores.
- Para declarar um vetor de `struct`:
  - Define-se a `struct`.
  - Declara-se o vetor do tipo `struct` criado.
- Exemplo:
  - `struct aluno Turma380[28];`
  - `struct endereco vetorEndAmigos[100];`

# Vetor de Struct (2/2)

- Para manipular os dados do vetor, devem ser fornecidos o índice e o campo.

```
strcpy(Turma380[0].nome, "Fulano");

Turma380[0].P1 = 9.5;

Turma380[0].P2 = 8.5;

Turma380[0].P3 = 9.0;

Turma380[0].faltas = 4;
```

# Arquivos e struct

- Para gravar um `struct` num arquivo binário, utilize o comando:

```
fwrite(&fulano, sizeof(struct aluno), 1, file)
```

- Para ler um `struct` de um arquivo binário, utilize o comando:

```
fread(&fulano, sizeof(struct aluno), 1, file)
```

# Posicionamento do Cursor

- Em todo arquivo aberto há um *cursor* indicando a posição onde ocorrerá a próxima leitura e/ou escrita.
- Todas as operações de leitura e escrita avançam o *cursor* após ler/escrever os dados.

```
int fseek(FILE *file, long distancia, int origem)
```

  - Move o cursor do arquivo `file` para a posição `distancia` relativa a alguma `origem`.
  - A `origem` deve ser uma destas 3 constantes:
    - `SEEK_SET` – início do arquivo.
    - `SEEK_CUR` – posição atual.
    - `SEEK_END` – fim do arquivo.
  - `fseek` retorna `0` em caso de sucesso e `-1` em caso de erro.

```
long ftell(FILE *file)
```

- Retorna a posição atual do cursor no arquivo `file`.

# Exemplo (1/4)

```
#include<stdio.h>
#include<stdlib.h>

#define MAX 4

int main (){
  FILE *pa;
  char nome[40];
  char linha[80];

  struct pessoa {
    char nome[40];
    int ano;
  } turma [MAX], back[MAX];

  int i;
```

# Exemplo (2/4)

```
for (i=0; i<MAX; i++) {
  puts("Nome ? ");
  fgets(turma[i].nome, 40, stdin);
  puts("Ano  ? "); fgets(linha, 80, stdin);
  sscanf(linha, "%d", &turma[i].ano);
}

puts ("\nImprimindo\n");
for (i=0; i<MAX; i++) {
  printf("Nome = %s\n", turma[i].nome);
  printf("Ano  = %d\n\n", turma[i].ano);
}
```

# Exemplo (3/4)

```
puts("\nGravando\n");
puts("Qual o nome do arquivo?"); fgets(nome, 40, stdin);

if (( pa = fopen(nome, "w+b")) == NULL ) {
  puts("Arquivo nao pode ser aberto");
  exit(1);
}

for (i=0; i<MAX; i++) {
  if (fwrite( &turma[i], sizeof (struct pessoa), 1, pa) != 1)
    puts("Erro na escrita.");
}

fseek(pa, 0, SEEK_SET);
```

# Exemplo (4/4)

```
  for (i=0; i<MAX; i++) {
    if (fread(&back[i], sizeof (struct pessoa), 1, pa) != 1) {
      puts("Erro na escrita.");
      if (feof(pa)) break;
        puts("Erro na leitura.");
    }
  }

  puts("Imprimindo o vetor lido.");

  for (i=0; i<MAX; i++) {
    printf("Nome = %s\n", back[i].nome);
    printf("Ano  = %d\n\n", back[i].ano);
  }

  exit(0);
}
```