MINIAULA DE ALGORITMOS
REGISTROS

Prof. Ivanilton Polato
Departamento Acadêmico de Computação (DACOM–CM)
ipolato@utfpr.edu.br

UTFPR

Registros: o que são?

- São estruturas compostas heterogêneas que agregam informações de diferentes tipos em um lugar só
- Cada informação em um registro é chamada de campo
- Cada campo tem seu tipo:
  - *Simples: char, int, long, float, double*
  - *Complexo: strings, vetores, matrizes*
- Declarados através da instrução STRUCT
  - *Define um tipo de dados personalizado, a partir do qual são criadas variáveis, manipuladas pelo programa*

UTFPR

Registros: exemplo

```
struct CONTA{
    int codigo;
    char nome[51];
    long telefone;
    float saldo;
};

struct CONTA cliente;
```

- O código ao lado declara uma estrutura chamada CONTA.
- **CONTA é um tipo de dados** complexo, que reúne diversos campos: codigo, nome, saldo e telefone.
- A seguir, definimos a **variável** cliente, que é do tipo CONTA!
  - *Nesse caso, temos espaço para armazenar um cliente*

UTFPR

Registros: alternativamente!

```
struct CONTA{
    int codigo;
    char nome[51];
    long telefone;
    float saldo;
};

struct CONTA cliente;
```

=

```
struct CONTA{
    int codigo;
    char nome[51];
    long telefone;
    float saldo;
}cliente;
```

Registros: exemplo

- Não podemos acessar todos os campos de uma vez
- Devemos acessar individualmente cada campo, como se fosse uma variável comum
- Adicionamos o prefixo <nomeDaVariável> seguida de um '.'

```
cliente.codigo = 123;
strcpy(cliente.nome, "Peter Griffin");
cliente.telefone = 5551234;
cliente.saldo = 299.99;
```

UTFPR

Registros: exemplo

- Podemos usar a função ***scanf()***:

```
...
   printf("Código: ");
   scanf("%d", &cliente.codigo);
   printf("Nome: ");
   scanf("%[^\n]", cliente.nome);
   printf("Telefone: ");
   scanf("%ld", &cliente.telefone);
   printf("Saldo: ");
   scanf("%f", &cliente.saldo);
...
```

- Toda a entrada de dados é feita usando scanf(), solicitando as informações ao usuário
- Os campos são tratados como variáveis regulares
  - *Não se esqueça do* **&** *onde for necessário!*

UTFPR

Registros: exemplo

- Exibindo as informações armazenadas:

```
...
   printf("Código: %d", cliente.codigo);
   printf("Nome: %s", cliente.nome);
   printf("Telefone: %ld", cliente.telefone);
   printf("Saldo: %.2f", cliente.saldo);
...
```

UTFPR

Registros + Vetores

```
struct CONTA{
    int codigo;
    char nome[51];
    char telefone[15];
    float saldo;
};

struct CONTA clientes[5];
```

- Agora criamos um vetor clientes com 10 posições
  - *Cada posição é uma estrutura completa, com todos os campos*
- Cada posição no vetor tem que ser manipulada individualmente também!

UTFPR

Registros + Vetores: visualizando

```
struct CONTA{
    int codigo;
    char nome[51];
    char telefone[15];
    float saldo;
}clientes[5];
```

- Cada posição do vetor é uma variável do tipo CONTA, e possui todos os campos declarados no tipo.
- Devem ser acessados da mesma maneira, mas considerando seu índice no vetor.

	codigo nome telefone saldo	codigo nome telefone saldo	codigo nome telefone saldo	codigo nome telefone saldo	codigo nome telefone saldo
[]	0	1	2	3	4

UTFPR

Registros + Vetores: manipulação

- Acessar posições do vetor e seus campos individualmente!

```
...
    clientes[0].codigo = 123;
    strcpy(clientes[0].nome, "Peter Griffin");
    clientes[0].telefone = 5551234;
    clientes[0].saldo = 299.99;
    clientes[1].codigo = 456;
    strcpy(clientes[1].nome, "Stewie");
    clientes[1].telefone = 5556666;
    clientes[1].saldo = 0.99;
...
```

UTFPR

Registros + Vetores: manipulação

```
...
for(int i=0; i<5; i++){
   printf("Código: ");
   scanf("%d", &clientes[i].codigo);
   printf("Nome: ");
   scanf("%[^\n]", clientes[i].nome);
   printf("Telefone: ");
   scanf("%ld", &clientes[i].telefone);
   printf("Saldo: ");
   scanf("%f", &clientes[i].saldo);
}
...
```

- Assim como nos vetores de tipos simples, podemos utilizar estruturas de repetição para facilitar a manipulação do vetor completo de uma vez só!

UTFPR

Registros + Vetores: manipulação

- Exibindo as informações armazenadas:

```
...
   printf("Código: %d", clientes[0].codigo);
   printf("Nome: %s", clientes[0].nome);
   printf("Telefone: %ld", clientes[0].telefone);
   printf("Saldo: %.2f", clientes[0].saldo);
...
```

UTFPR

Registros: lembretes!

- Nunca manipule o tipo, e sim a variável criada a partir dele!

```
struct CONTA clientes[5];
```

- Cada campo deve ter seu tipo individual!
- Lembre-se que o índice pertence ao vetor!

Certo:

✅ `clientes[0].codigo = 123;`

Errado:

❌ `clientes.codigo[0] = 123;`

UTFPR