CI1055: Algoritmos e Estruturas de Dados I

Profs. Drs. Marcos Castilho Carmem Hara e Bruno Muller Jr

Departamento de Informática/UFPR

3 de setembro de 2020

Registros - Introdução

- Conceituação
- Registros como Encapsuladores
- Vetores com Registros

Conceituação

- Matrizes e vetores são estruturas de dados homogêneas (todos os elementos são de um mesmo tipo).
- Registros são estruturas de dados heterogêneas, ou seja, cada elemento pode abrigar grupos de dados de tipos diferentes.
- Exemplo: cadastro de um cliente de um banco;

```
type cliente = record
    nome     : string[50];
    fone     : longint;
    endereco : string;
    idade    : integer;
    rg       : longint;
    cpf      : qword;
end;
var r: cliente;
```

Conceituação

- Matrizes e vetores são estruturas de dados homogêneas (todos os elementos são de um mesmo tipo).
- Registros são estruturas de dados heterogêneas, ou seja, cada elemento pode abrigar grupos de dados de tipos diferentes.
- Exemplo: cadastro de um cliente de um banco;

```
type cliente = record
    nome     : string[50];
    fone     : longint;
    endereco : string;
    idade    : integer;
    rg       : longint;
    cpf      : qword;
end;
var r: cliente;
```

Conceituação

- Matrizes e vetores são estruturas de dados homogêneas (todos os elementos são de um mesmo tipo).
- Registros são estruturas de dados heterogêneas, ou seja, cada elemento pode abrigar grupos de dados de tipos diferentes.
- Exemplo: cadastro de um cliente de um banco;

```
type cliente = record
    nome      : string[50];
    fone      : longint;
    endereco  : string;
    idade     : integer;
    rg        : longint;
    cpf       : qword;
end;
var r: cliente;
```

Declaração

- Declaração:

```
type cliente = record
    nome     : string[50];
    fone     : longint;
    endereco : string;
    idade    : integer;
    rg       : longint;
    cpf      : qword;
end;
var r: cliente;
```

- Acesso:

```
r.nome     := 'Fulano de Tal';
r.fone     := 32145678;
r.endereco := 'Rua dos bobos, no 0';
r.idade    := 75;
r.rg       := 92346539;
r.cpf      := 11122233344;
```

Declaração

- Declaração:

```
type cliente = record
   nome     : string[50];
   fone     : longint;
   endereco : string;
   idade    : integer;
   rg       : longint;
   cpf      : qword;
end;
var r: cliente;
```

- Acesso:

```
r.nome     := 'Fulano de Tal';
r.fone     := 32145678;
r.endereco := 'Rua dos bobos, no 0';
r.idade    := 75;
r.rg       := 92346539;
r.cpf      := 11122233344;
```

Leitura

- Sem "with"

```
procedure le_reg (r: cliente)
begin
    read (r.nome);
    read (r.fone);
    read (r.endereco);
    read (r.idade);
    read (r.rg);
    read (r.cpf);
end;
```

- Com "with"

```
procedure le_reg (r: cliente)
begin
    with r do
    begin
        read (nome);
        read (fone);
        read (endereco);
        read (idade);
        read (rg);
        read (cpf);
    end;
end;
```

Leitura

- Sem "with"

```
procedure le_reg (r: cliente)
begin
    read (r.nome);
    read (r.fone);
    read (r.endereco);
    read (r.idade);
    read (r.rg);
    read (r.cpf);
end;
```

- Com "with"

```
procedure le_reg (r: cliente)
begin
    with r do
    begin
        read (nome);
        read (fone);
        read (endereco);
        read (idade);
        read (rg);
        read (cpf);
    end;
end;
```

Impressão

```
procedure imprime_reg (r: cliente)
begin
    with r do
    begin
        writeln (nome);
        writeln (fone);
        writeln (endereco);
        writeln (idade);
        writeln (rg);
        writeln (cpf);
    end;
end;
```

Registros como Encapsuladores

- Por vezes, é interessante utilizar registros para encapsular informações correlatas;
- Por exemplo, vetor e seu tamanho: ao invés de passar dois parâmetros, basta o registro.

```
...
const MAX = 10000;
type vetor     = array [1..MAX] of real;
   tipo_vetor = record
                   tam    : integer;
                   dados  : vetor;
                end;
var v: tipo vetor;
...
procedure ler_vetor (var v: tipo_vetor );
var i : integer;
begin
   readln (v.tam); (* tamanho do vetor *)
   for i:= 1 to v.tam do
      readln (v.dados[i]) (* elementos do vetor *)
end;
```

Registros como Encapsuladores

- Por vezes, é interessante utilizar registros para encapsular informações correlatas;
- Por exemplo, vetor e seu tamanho: ao invés de passar dois parâmetros, basta o registro.

```
...
const MAX = 10000;
type vetor     = array [1..MAX] of real;
   tipo_vetor = record
                    tam   : integer;
                    dados : vetor;
                end;
var v: tipo vetor;
...
procedure ler_vetor (var v: tipo_vetor );
var i : integer;
begin
   readln (v.tam); (* tamanho do vetor *)
   for i:= 1 to v.tam do
      readln (v.dados[i]) (* elementos do vetor *)
end;
```

Vetores de Registros

- Sabemos como armazenar a informação de um cliente no banco.
- Para armazenar a informação de vários clientes, uma solução é criar um vetor de registros.
-

```
const MAX = 10000;
type cliente = record
                 nome     : string [50];
                 fone     : longint;
                 endereco : string;
                 idade    : integer;
                 rg       : longint;
                 cpf      : qword;
               end;
     bd = array [1..MAX] of cliente;

var  r     : cliente;
     v     : bd;        (* vetor com tam_v clientes! *)
     tam_v : integer;
```

Vetores de Registros

- Sabemos como armazenar a informação de um cliente no banco.
- Para armazenar a informação de vários clientes, uma solução é criar um vetor de registros.

```
const MAX = 10000;
type cliente = record
                 nome     : string [50];
                 fone     : longint;
                 endereco : string;
                 idade    : integer;
                 rg       : longint;
                 cpf      : qword;
               end;
     bd = array [1..MAX] of cliente;

var  r     : cliente;
     v     : bd;        (* vetor com tam_v clientes! *)
     tam_v : integer;
```

Vetores de Registros

- Sabemos como armazenar a informação de um cliente no banco.
- Para armazenar a informação de vários clientes, uma solução é criar um vetor de registros.
-

```
const MAX = 10000;
type cliente = record
                 nome     : string [50];
                 fone     : longint;
                 endereco : string;
                 idade    : integer;
                 rg       : longint;
                 cpf      : qword;
               end;
     bd = array [1..MAX] of cliente;

var  r     : cliente;
     v     : bd;        (* vetor com tam_v clientes! *)
     tam_v : integer;
```

Leitura em Vetores de Registros

- Supondo que as informações do banco de dados estejam em um arquivo, é possível carregá-lo para o programa:

```
procedure ler_cliente (var r : cliente ) ;
begin
   with r do
   begin
      readln (nome);
      readln (fone);
      readln (endereco);
      readln (idade);
      readln (rg);
      readln (cpf);
   end;
end;
procedure carregar_todos_clientes (var v: bd; var tam_v: integer) ;
begin
   readln (tam_v) ;
   for i:= 1 to tam_v do
      ler_cliente (v[i]) ;
end;
```

Leitura em Vetores de Registros

- Supondo que as informações do banco de dados estejam em um arquivo, é possível carregá-lo para o programa:

```
procedure ler_cliente (var r : cliente ) ;
begin
   with r do
   begin
      readln (nome);
      readln (fone);
      readln (endereco);
      readln (idade);
      readln (rg);
      readln (cpf);
   end;
end;
procedure carregar_todos_clientes (var v: bd; var tam_v: integer) ;
begin
   readln (tam_v) ;
   for i:= 1 to tam_v do
      ler_cliente (v[i]) ;
end;
```

Busca em Vetores de Registros

- É necessário especificar em qual campo do registro fazer a busca (CPF).
-

```
procedure busca_telefone (var v: bd; tam_v:integer; cpf_procurado : string);
var i : integer ;
begin
   i:= 1;
   while (i <= tam v) and (v[i].cpf <> cpf_procurado) do
      i := i+1;
   if i <= tam_v then
      writeln ('O telefone do cliente com CPF ' ,v[i].cpf, ' eh: ' ,v[i].fone)
   else
      writeln ('Cliente nao localizado na base.') ;
end;
```

- Crie uma função que retorna o índice que contém a informação desejada;

Busca em Vetores de Registros

- É necessário especificar em qual campo do registro fazer a busca (CPF).
-

```
procedure busca_telefone (var v: bd; tam_v:integer; cpf_procurado : string);
var i : integer ;
begin
   i:= 1;
   while (i <= tam v) and (v[i].cpf <> cpf_procurado) do
      i := i+1;
   if i <= tam_v then
      writeln ('O telefone do cliente com CPF ' ,v[i].cpf, ' eh: ' ,v[i].fone)
   else
      writeln ('Cliente nao localizado na base.') ;
end;
```

- Crie uma função que retorna o índice que contém a informação desejada;

Busca em Vetores de Registros

- É necessário especificar em qual campo do registro fazer a busca (CPF).
-

```
procedure busca_telefone (var v: bd; tam_v:integer; cpf_procurado : string);
var i : integer ;
begin
   i:= 1;
   while (i <= tam v) and (v[i].cpf <> cpf_procurado) do
      i := i+1;
   if i <= tam_v then
      writeln ('O telefone do cliente com CPF ' ,v[i].cpf, ' eh: ' ,v[i].fone)
   else
      writeln ('Cliente nao localizado na base.') ;
end;
```

- Crie uma função que retorna o índice que contém a informação desejada;

Ordenação em Vetores de Registros

- Para ordenar, é preciso especificar uma "chave";
- Por exemplo, o CPF:

```
procedure ordena_por_cpf (var v: bd; tam: integer);
var i, j , pos_menor : integer;
begin
   for i:= 1 to tam-1 do
   begin
      (* acha o menor elemento a partir de i *)
      pos_menor:= i;
      for j:= i+1 to tam do
         if v[j].cpf < v[pos_menor].cpf then
            pos_menor:= j;
      troca (bd, i, pos_menor); (* troca os elementos *)
   end;
end;
```

Ordenação em Vetores de Registros

- Para ordenar, é preciso especificar uma "chave";
- Por exemplo, o CPF:

```
procedure ordena_por_cpf (var v: bd; tam: integer);
var i, j , pos_menor : integer;
begin
   for i:= 1 to tam-1 do
   begin
      (* acha o menor elemento a partir de i *)
      pos_menor:= i;
      for j:= i+1 to tam do
         if v[j].cpf < v[pos_menor].cpf then
            pos_menor:= j;
      troca (bd, i, pos_menor); (* troca os elementos *)
   end;
end;
```

Troca Registros

- para trocar os conteúdos de dois registros, é necessário trocar cada campo (aux:=v[k] não funciona);

```
procedure troca (var v: bd; k,m: integer);
var aux: cliente;
begin
    with aux do
    begin
        nome     := v[k].nome;
        fone     := v[k].fone;
        endereco:= v[k].endereco;
        idade    := v[k].idade;
        rg       := v[k].rg;
        cpf      := v[k].cpf;
    end;
    with v[k] do
    begin
        nome     := v[m].nome;
        fone     := v[m].fone;
        endereco:= v[m].endereco;
        idade    := v[m].idade;
        rg       := v[m].rg;
        cpf      := v[m].cpf;
    end;
    with v[m] do
    begin
        nome     := aux.nome;
        fone     := aux.fone;
        endereco:= aux.endereco;
        idade    := aux.idade;
        rg       := aux.rg;
        cpf      := aux.cpf;
    end;
end;
```

- este material está no livro no capítulo 11, seções 11.1 e 11.3