UNISINOS - UNIVERSIDADE DO VALE DO RIO DOS SINOS
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS (C6/6) – Curso: Informática

PROGRAMAÇÃO I – AULA 10

Disciplina: Linguagem de Programação PASCAL
Professor responsável: ***Fernando Santos Osório***
Semestre: 2001/2
Horário: 63

E-mail: ***osorio@exatas.unisinos.br***
Web:
http://inf.unisinos.br/~osorio/prog1.html
Xerox : *Pasta 54* (Xerox do C6/6)

1. Registros – Comandos: TYPE e RECORD:

Variáveis estruturadas, compostas por um conjunto de variáveis:
- Estruturas heterogêneas = *Registros (RECORD)*
- Estruturas homogêneas = *Vetores (Array)*

Os registros permitem a criação de variáveis estruturadas que são compostas pelo agrupamento de 1 ou mais variáveis simples em uma estrutura única. Surge assim o conceito de grupo de variáveis, o ***registro***, onde cada elemento (variável) que compõe este grupo (registro) é chamado de ***campo do registro***. Os campos de um registro podem ser de um mesmo tipo ou de tipos diferentes, onde o exemplo típico de um registro é o de um grupo de dados usado para descrever o funcionário de uma empresa: nome, endereço, telefone, cargo e salário. Todas estas informações são agrupadas, permitindo manter "juntas" as informações referentes a um funcionário.

* SINTAXE: ***A declaração pode ser assim...***

```
<nome_da_variável> : RECORD
                        <campo1> : <tipo_da_variável> ;
                        <campo2> : <tipo_da_variável> ;
                        <campo3> : <tipo_da_variável> ;
                        ...
                        <campoN> : <tipo_da_variável> ;
                     END;
```

ou assim...

```
TYPE

 <nome_do_registro> = RECORD
                        <campo1> : <tipo_da_variável> ;
                        <campo2> : <tipo_da_variável> ;
                        <campo3> : <tipo_da_variável> ;
                        ...
                        <campoN> : <tipo_da_variável> ;
                     END;

VAR

   <nome_da_variável> = <nome_do_registro> ;
```

Usando registros, acesso a um campo específico...

<nome_da_variável> **.** <campo> := <valor>;

write (<nome_da_variável> **.** <campo>);

Observações:

- A principal vantagem de um registro é poder agrupar dados, dando um *nome único* para a estrutura resultante deste agrupamento. Assim podemos facilmente criar estruturas de dados do tipo "registro_de_aluno" e "registro_de_professor", onde temos em ambos uma variável chamada "nome". Deste modo, para o programador ficará muito fácil distinguir entre o nome que pertence ao professor e o nome que pertence ao aluno;

- Usualmente (e podemos dizer que na prática sempre será assim) vamos definir um tipo de dados novo para representar esta nossa estrutura do tipo registro. Primeiramente usaremos o comando TYPE para criar um novo tipo de dados (tipo do registro) e depois declaramos uma variável como sendo do tipo do registro previamente definido no Type.

Exemplos:

```
PROGRAM  Exemplo;

TYPE
   Reg_Funcionario =  RECORD
                         Nome      : String;
                         Sexo      : Char;
                         Endereco : String;
                         Telefone  : String;
                         Cargo     : String;
                         Salario    : Real;
                         Ano_Nascimento : Integer;
                       END;

   Reg_Aluno = RECORD
                    Nome : String;
                    Sexo   : Char;
                    Matricula : String;
                    GrauA : Real;
                    GrauB : Real;
                    GrauC : Real;
                    Media  : Real;
                    Situacao : Char;  { 'C' = Cursando, 'A'=Aprovado, 'R'=Repr. }
                 END;

VAR
   Prof_Matematica : Reg_Funcionario;
   Prof_Fisica : Reg_Funcionario;
   Aluno : Reg_Aluno;
   Melhor_Aluno : Reg_Aluno;
```

```
BEGIN
   Prof_Matematica.Nome := 'Pitágoras Fermat Lagrange';
   Prof_Fisica.Nome := 'Faraday do Aga';
   Prof_Fisica.Salario := 130.00 + indice_percentual_adicional (130.0, 0.46);
   Aluno.Nome := 'Dialético Festudantino';
   Melhor_Aluno.Nome := 'Carlos Duarte Filho';
   Melhor_Aluno.GrauA := 10.0;
   Melhor_Aluno.GrauB := 10.0;
   ...
   Melhor_Aluno.Media := (Melhor_Aluno.GrauA + Melhor_Aluno.GrauB ) / 2;
   WriteLn ('Média final do melhor aluno = ',Melhor_Aluno.Media:2:2);
   WriteLn ('Média final do aluno = ',Aluno.Media:2:2);
   ReadLn;
END.
```

1.1. Comando WITH

Você já deve ter percebido como é desagradável ter que repetir todo o nome do registro a cada vez que queremos acessar ou modificar um dos campos do mesmo. Para simplificar esta tarefa, a linguagem Pascal tem um comando, chamado WITH, que nos permite indicar que à partir do Begin até o End deste comando vamos estar nos referenciando aos campos de um registro específico. Quando usamos o WITH indicamos o nome do registro e depois podemos acessar diretamente os campos sem ter que indicar novamente qual é o registro que estamos querendo acessar.

SINTAXE:

```
WITH <nome_da_variável_tipo_registro> DO <comando> ;
```

ou

```
WITH <nome_da_variável_tipo_registro> DO
BEGIN
  <comando> ;
  ...
  <comando> ;
END;
```

EXEMPLO: (baseado no uso dos registros do exemplo descrito anteriormente)

```
...
Aluno.Nome := 'Dilberto Futuro';
WITH  Aluno DO
Begin
    ReadLn (GrauA);                  { Com o With não preciso escrever : Aluno.GrauA }
    ReadLn (GrauB);
    Media := (GrauA + GrauB) / 2;
    WriteLn ('Média : ',Media:2:2);
    If  Media < 6.0
    Then ReadLn (GrauC);
End;
...
```

1.2. Estruturas RECORD com Campos Variantes (*Variant Fields*)

Quando criamos um conjunto de dados do tipo RECORD podemos ter *campos de variáveis que ocupam o mesmo espaço* na memória, mas que podem ser tratados diferentemente segundo a situação específica de um certo conjunto de dados.

Por exemplo, suponha que temos definido um registro que guarda as informações sobre uma pessoa, onde caso esta seja brasileira, queremos saber o estado onde nasceu, e caso seja estrangeira, queremos saber o país onde nasceu. Seria um desperdício de espaço em memória se mesmo para os brasileiros fosse guardada a informação extra sobre o país onde ele nasceu. Por isso podemos colocar as duas informações (estado e país estrangeiro) em um mesmo lugar, usando uma ou outra conforme a nacionalidade da pessoa.

Exemplo:

```
TYPE
   Reg_Pessoa = RECORD
                      Nome : String;
                      Sexo  :  Char;
                      Case Brasileiro : Boolean Of
                         True : ( Estado : string [30];
                                  Cidade : string[30]  ) ;
                         False: (  Pais :  string[40];
                                  Passaporte : String[20] );
                   END;
 VAR
    Pessoa : Reg_Pessoa;
```

Se a pessoa for brasileira (Pessoa.Brasileiro := True) poderemos então informar o estado (Pessoa.Estado := '...') e a cidade de onde esta pessoa é natural (Pessoa.Cidade := '...'). Caso a pessoa não seja brasileira (Pessoa.Brasileiro := False), então podemos informar o país de onde esta pessoa veio e o número de seu passaporte.

1.3. Comando RECORD usado na criação de dados Constantes

Veja o exemplo abaixo de como declarar registros, usando definições de enumerações e constantes ao mesmo tempo. O exemplo é bastante interessante devido a maneira clara como os dados são estruturados (exemplo tirado do Help do Turbo Pascal).

```
TYPE
   Month = (Jan, Feb, Mar, Apr, May, Jun, Jly, Aug, Sep, Oct, Nov, Dec);
   Date = record
             D : 1..31;
             M: Month;
             Y : 1900..1999;
          end;
CONST
    SomeDay: Date = (D: 2; M: Dec; Y: 1960);
```

EXERCÍCIOS – AULA 10

1. Faça a parte relativa a declaração de tipos de dados e variáveis de um programa, criando os registros de um sistema que precise descrever os seguintes dados:

   1.1. Alunos da Unisinos
   1.2. Produtos de um supermercado
   1.3. Músicas de um CD
   1.4. Carros de uma revendedora
   1.5. Modelos de microcomputadores

2. Faça um programa para uma loja de discos (CDs) com uma *sub-rotina de entrada de dados* (ler os dados de um único CD) e uma outra *sub-rotina de saída de dados* (escrever na tela estes dados). O programa principal deve conter apenas 4 linhas de comandos: executar a entrada de dados, seguido de um readln, após é executada a escrita dos dados na tela, terminando por um outro readln. Os dados referentes aos CDs são: nome do CD, nome do cantor/grupo, gravadora e preço. Use um REGISTRO para armazenar estas informações. Não use variáveis globais.

3. Faça um programa para uma loja de computadores com uma *sub-rotina de entrada de dados* (ler os dados de um único modelo de computador) e uma outra *sub-rotina de saída de dados* (escrever na tela estes dados). O programa principal deve conter apenas 4 linhas de comandos: executar a entrada de dados, seguido de um readln, após é executada a escrita dos dados na tela, terminando por um outro readln. Defina precisamente os dados referentes ao computador, e use um REGISTRO para armazenar as informações. Não use variáveis globais.

4. Faça um programa para o controle das notas de um aluno, similar aos programas vistos anteriormente. Use funções, procedures e *registros.* Os registros devem descrever os dados relativos ao aluno e deve ser criada uma *sub-rotina para o cálculo da médi*a *ponderada* (NotaA=peso1, NotaB=peso2). Ler o nome do aluno e suas duas notas A e B, e após calcular a média ponderada entre estas notas. Determinar se o aluno precisa recuperar alguma das notas para ficar com média igual ou superior à 6.0. Repetir este procedimento para uma turma composta por cinco alunos. Sub-rotinas a serem implementadas: Le_Dados, Calcula_Media, Calcula_Nota_Recuperacao, Exibe_Dados.

5. Crie um programa baseado no programa anterior, contendo um menu com as seguintes opções:

   (1) Entrar com o nome, número de matrícula e as notas GA e GB de um aluno;
   (2) Calcular a média do aluno e exibir na tela o nome, número de matrícula e a média. Usar uma média ponderada com pesos 1 e 2 respectivamente para o GA e GB;
   (3) Mostrar os dados do aluno e a situação deste aluno: aprovado ou reprovado, sabendo que a média para a aprovação deve ser superior ou igual à 6.0;
   (4) Indicar a nota mínima que o aluno reprovado necessita tirar caso ele substitua o GA;
   (5) Indicar a nota mínima que o aluno reprovado necessita tirar caso ele substitua o GB;
   (6) Opção escolhida caso o usuário deseje sair do programa e terminar a execução.

   Cada opção do programa deve ser implementada por uma sub-rotina (menos a última opção), sendo que o programa *não deve* possuir variáveis globais, bem como *deve ter* um registro declarado para armazenar os dados referentes ao aluno.

6. Faça uma sub-rotina que receba como entrada 2 registros do tipo "Reg_Pizza" e troque o conteúdo dos dois registros entre si: chame esta sub-rotina de "troca_pizza".

7. Faça um programa que crie 2 registros com 2 campos: nome e idade. Leia estes dados referentes a duas pessoas e crie uma sub-rotina que receba os dois registros como parâmetros e retorne ambos ordenados (colocar no primeiro os dados daquela pessoa que possui a menor idade e no segundo aquela que possui a maior idade). Se necessário troque o conteúdo entre os dois registros. Exiba os registros na tela na ordem em que foram lidos, de forma a verificar se foram realmente bem ordenados.

8. Crie um registro que contenha o nome, CPF e salário de uma pessoa, declarando depois duas variáveis Func_1 e Func_2 que contenham estas informações descritas acima sobre os funcionários. Faça um programa para testar os seguintes procedimentos:

   - Ler os dados referentes ao Func_1;
   - Exibir os dados de um funcionários (é possível exibir todo o registro de uma só vez ?);
   - Copiar os dados do registro do Func_1 para o registro do outro funcionário (é possível atribuir todos os dados de um funcionário para o registro do outro de uma só vez ?);
   - Reajustar o valor do salário do funcionário Func_2 em 10%;
   - Exibir os dados de cada um dos dois funcionários, Func_1 e Func_2;
   - Calcular a média de salário dos funcionários, usando uma função para isto (é possível passar como parâmetro para a função apenas o valor do salário dos funcionários?).