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

PROGRAMAÇÃO I – AULA 05

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. Comando WHILE / DO – Laço para repetição condicional de comandos
WHILE (Enquanto condição verdadeira) DO (Faça comando)

O comando WHILE repete (laço = loop) a execução de um conjunto de comandos enquanto a expressão condicional do comando for verdadeira. Inicialmente a expressão condicional é testada, se ela for verdadeira o comando que segue o "DO" é executado (e depois voltamos para um novo teste), e em caso contrário, se a expressão for falsa o comando não é executado e o laço de repetição termina. Ao terminar o comando WHILE, a execução segue normalmente à partir da próxima linha que segue o While/Do.

* SINTAXE:

```
WHILE <condição>
DO   <comando> ;
```

<condição> = Expressão condicional cujo resultado é um valor lógico (True ou False).
A expressão condicional é descrita de maneira análoga à <condição> de um comando IF.

<comandos> = Comando que é executado repetidas vezes enquanto a expressão condicional do While for verdadeira. O comando pode ser substituído por mais de um comando se usarmos um Begin/End.

Observações:

- Todo <comando> pode ser substituído por um BEGIN/END contendo múltiplos comandos no seu interior (da mesma forma como ocorre com os outros comandos do Pascal, como por exemplo no IF/THEN/ELSE). O While/Do, tendo ou não um Begin/End, sempre vai terminar por um ';'.

- Preste MUITA atenção nos Begin/End... o esquecimento de um deles pode causar grandes problemas (ex.: laço eterno).

- Certifique-se de que a expressão condicional do While será afetada pelo(s) comando(s) contidos dentro do laço (comandos descritos após o "do"). Caso a expressão não seja afetada, você irá criar um laço sem fim (loop infinito), pois jamais a expressão condicional mudará o seu estado.

- Certifique-se que as variáveis da expressão condicional usada no comando While tenham sido corretamente lidas ou inicializadas ANTES da execução do referido comando.

Variável Sinalizadora: "FLAG"

Um conceito/técnica interessante usado nas linguagens de programação é o emprego de uma variável para sinalizar algum tipo de evento, situação ou estado do programa. Esta variável que exerce a função de sinalizar alguma coisa é denominada de FLAG ("bandeira" em inglês).

Em muitas situações, o uso de *flags* pode simplificar bastante um programa. Podemos trabalhar com flags que são variáveis booleanas (verdadeiro/falso) ou mesmo podemos usar uma variável corrente do programa e através da atribuição de um valor especial para esta variável usá-la para sinalizar alguma situação.

Exemplos de uso de Flags:

```
Program Demonstra_Uso_de_Flags1;              { Flag: Booleano}
Var
   Resposta_Invalida:Boolean;
   Fim_Prog: string[3];
Begin
    Resposta_Invalida:=true;
    While (Resposta_Invalida)
   Do Begin
         Writeln('Deseja terminar o programa? ');
         ReadLn (Fim_Prog);
         If  (Fim_Prog = 'Sim') or (Fim_Prog = 'SIM') or (Fim_Prog = 'sim') or
            (Fim_Prog  = 'S')     or (Fim_Prog = 's')
         Then Resposta_Invalida:=false;
       End;
End.
```

```
Program Demonstra_Uso_de_Flags;               { Flag: Valor especial }
Var
   Fim_Prog: string[3];
Begin
    Fim_Prog:='X'
    While (Fim_Prog <> 'S')
   Do Begin
         Writeln('Deseja terminar o programa? ');
         ReadLn (Fim_Prog);
         If  (Fim_Prog = 'Sim') or (Fim_Prog = 'SIM') or (Fim_Prog = 'sim') or
            (Fim_Prog  = 'S')     or (Fim_Prog = 's')
         Then Fim_Prog:='S';
         If  (Fim_Prog = 'Não') or (Fim_Prog = 'NÃO') or (Fim_Prog = 'não') or
            (Fim_Prog  = 'N')    or (Fim_Prog = 'n')
         Then Fim_Prog:='N';
         If  (Fim_Prog <> 'S') and (Fim_Prog <> 'N')
         Then Fim_Prog:='X';
         If (Fim_Prog = 'X') Then Writeln('>> Resposta inválida!');
       End;
End.
```

Exemplos:

```
X : = 1;                        { Inicializa o contador        }
WHILE  X < 10                   { Faz uma contagem até 10 }
DO X := X + 1;
...

ReadLn (Nota);
WHILE  ( Nota < 0.0 )  OR  ( Nota  > 10.0 )
DO  ReadLn (Nota);
...

Resp := 'X';
WHILE ( Resp <> 'N' )  AND (  Resp <>'n' )
DO  BEGIN
        Writeln ('Hello World!');
        Write    ('Novamente (S/N) ? ');
        ReadLn (Resp);
    END;
...

ReadLn (Media);                 { Media => Usamos um valor especial para terminar  (-1)  }
WHILE  Media <> -1              { -1 serve como um sinalizador (flag) para terminar o laço }
DO  BEGIN
        IF  Media >= 6.0
        THEN  WriteLn ('Aprovado')
        ELSE   WriteLn ('Reprovado');
        ReadLn (Media);
    END;
...
```

CONTRA-Exemplos: *EXEMPLOS DE USO **INCORRETO** DO WHILE*

```
WHILE  Idade < 20                   { Laço Infinito – Idade não é alterada no laço! }
DO  Salario := Salario – Desconto;
...

X := 1.0;
WHILE  X < 10.0                     { Condição mal definida, X diminui... }
DO  X := X – 0.1;
...

Resp := '?';                        { Laço Infinito - Falta o Begin / End   }
WHILE  Resp <> 'S'
DO  Writeln ('Novamente (S/N) ? ');
    ReadLn (Resp);
...
```

EXERCÍCIOS – AULA 05

1. Faça um comando While equivalente ao descrito abaixo (que realize o mesmo tipo de procedimento e obtenha o mesmo resultado), mas sem utilizar o operador relacional "OR". Você pode usar outros operadores lógicos ou relacionais, exceto o "OR".

```
WHILE  ( Nota < 0.0 )  OR  ( Nota  > 10.0 )
DO  ReadLn (Nota);
```

2. Faça um programa que leia dois números inteiros, o primeiro é o valor inicial de um contador, e o segundo é o valor final do contador (testar se o valor inicial fornecido é inferior ao valor final). Usando o comando *While*, escreva na tela uma contagem que comece no primeiro número lido, escreva os números seguintes colocando sempre apenas um número em cada nova linha da tela, e terminando a contagem quando chegar ao valor final indicado.

3. Ler o nome de um aluno e suas duas notas A e B, e após calcular a média ponderada entre estas notas (A tem peso 1 e B tem peso 2). Repetir este procedimento para uma turma composta por cinco alunos, usando o comando *While*. Exemplo de tela de saída:

```
Entre com o nome do aluno: João da Silva
Entre com o grau A: 5.0
Entre com o grau B: 6.0
O aluno João da Silva tem uma média: 5.66
```

4. Baseado no programa anterior, faça um novo programa de maneira que possamos trabalhar com turmas compostas por um número variável de alunos. Após calcular e imprimir a média de um aluno, exibir uma mensagem perguntando ao usuário se existem mais alunos (resposta: sim / não). Se tiver mais alunos, continuar o procedimento de leitura das notas e o cálculo da média até que o usuário responda ' não' . Usar o comando *While* e gerar uma saída conforme o exemplo de tela de saída abaixo:

```
Entre com o nome do aluno: João da Silva
Entre com o grau A: 5.0
Entre com o grau B: 6.0
O aluno João da Silva tem uma média: 5.66
Continuar (sim/não) ? sim
```

5. Baseado no programa anterior, faça um novo programa de maneira a validar as notas fornecidas pelo usuário (notas devem ser valores positivos entre 0.0 e 10.0). Indicar ao usuário se a nota fornecida é inválida e pedir para fornecer uma nova nota, repetindo este processo até que o usuário informe uma nota correta. Usar um laço *While* na leitura das nota, e gerar uma saída conforme o exemplo de tela de saída abaixo:

```
Entre com o nome do aluno: João da Silva
Entre com o grau A: 15.3
ERRO: Nota inválida! Digite novamente a nota.
Entre com o grau A: 5.0
Entre com o grau B: 6.0
O aluno João da Silva tem uma média: 5.66
Continuar (sim/não) ? não
```

6. Fazer um programa que calcule e imprima o fatorial de um número fornecido pelo usuário, usando o comando *While*. Repetir a execução do programa tantas vezes quantas o usuário quiser. Lembre-se que o resultado do cálculo de um fatorial pode ser um número "grande" (Exemplo: Fatorial de 8 = 40320). Exemplo de tela de saída:

Entre com um número: *5* O fatorial de **5** é **120** Outro número (sim/não) ? *não*

7. Escrever um programa que calcule todos os números inteiros divisíveis por um certo valor indicado pelo usuário, e compreendidos em um intervalo também especificado pelo usuário. O usuário deve entrar com um primeiro valor correspondente ao divisor e após ele vai fornecer o valor inicial do intervalo, seguido do valor final deste intervalo. Usar o comando *While*. Exemplo de tela de saída:

Entre com o valor do divisor: *3* Início do intervalo: *17* Final do intervalo: *29* Números divisíveis por **3** no intervalo de **17** à **29**: **18 21 24 27**

8. Faça um programa para o "jogo de adivinhar um número". O computador deve sortear um número entre 0 e 100 e pedir para o usuário tentar adivinhar este número. O usuário vai dizer o seu palpite, e o computador deve responder, se ele é maior ou menor que o número que ele sorteou. O programa termina somente quando o usuário acertar exatamente qual o número que o computador tinha sorteado, escrevendo uma mensagem de felicitações para o nosso usuário e indicando o número total de tentativas feitas. Dica: para gerar um número qualquer entre 0 e 100, use um comando como o deste exemplo indicado logo a seguir. Exemplo: numero_sorteado := random (100); { 0 <= numero_sorteado < 100 }

9. Faça um programa para o "jogo de adivinhar um número", mas invertendo os papéis desta vez. O computador que vai tentar adivinhar um número escolhido pelo usuário. O usuário deve escolher um número e para cada número apresentado pelo computador, responder se ele acertou, ou se o número apresentado é maior que o escolhido, ou se ele é menor que o escolhido. O programa termina quando o usuário responder que o computador acertou.

10. Faça um programa que obtenha e exiba na tela todos os números primos de 1 até 150. Os números primos são aqueles que só são divisíveis por 1 e por eles mesmos (exemplo: 1, 3, 5, 7, ...).