UNIASSELVI

GRADUAÇÃO E PÓS

PROFESSOR JOSÉ PAULO | TURMA FLC15770BES

UNIASSELVI

#ADistanciaImporta

A **distância nunca foi tão importante** em um momento como esse. A **UNIASSELVI**, referência na metodologia EAD semipresencial, ajustou o seu modelo de ensino para que **todas as aulas e provas sejam totalmente virtuais.** Assim você não precisa sair de casa para estudar. ***Com atitudes como essa, você colabora para a integridade da saúde de todos.***

Ah, e fique atento: quando a situação se resolver, voltaremos para o nosso modelo semipresencial – a metodologia mais indicada para uma educação de qualidade. ***#ADistânciaImporta***

Tabela Verdade

CONDIÇÃO p	CONDIÇÃO q	p ∧ q (E)	p ∨ q (OU)	p ⊻ q (XOR)	p ↓ q (NOR)	p → q	p ↔ q	~p	~q
V	**V**	V	V	F	F	V	V	F	F
V	**F**	F	V	V	F	F	F	F	V
F	**V**	F	V	V	F	V	F	V	F
F	**F**	F	F	F	V	V	V	V	V

Conectivo e Estrutura Lógica		É verdadeira quando	É falsa quando
E	**p ∧ q**	as duas são verdadeiras	uma ou as duas sejam falsas
OU	**p ∨ q**	uma ou as duas sejam verdadeiras	as duas sejam falsas
SE ENTÃO	**p → q**	nas demais situações	**p** é verdadeira e **q** é falsa
SE E SOMENTE SE	**p ↔ q**	**p** e **q** tiverem valores lógicos iguais	**p** e **q** tiverem valores lógicos diferentes
NÃO	**~p**	**p** é falsa	**p** é verdadeira

UNIASSELVI
GRADUAÇÃO E PÓS

Conceitos de Proposições

- Normalmente, as proposições são representadas por letras minúsculas (t, v, d, s, r etc.).
- Nenhuma proposição pode assumir os dois valores de verdadeiro ou falso ao mesmo tempo. Ou a proposição é verdadeira, ou é falsa.

- Exemplos de proposições:
  - p:Turma de Bacharelado em Engenharia de Software.
  - q:10 > 12508.
- A forma de apresentação das sentenças acima, baseadas no raciocínio lógico, é representada da seguinte forma em estrutura de sentença:
  - VL(p) = V, indicando que o valor de p é VERDADEIRO
  - VL(q) = F, indicando que o valor de q é FALSO
- Existem três princípios básicos a que as proposições devem atender:
  - Princípio da Identidade: a proposição é verdadeira ou falsa.
  - Não contradição: a proposição não poderá ser verdadeira ou falsa ao mesmo tempo.
  - Terceiro excluído: não existem outros valores de juízo: somente verdadeiro e falso.
- Outra característica importante das proposições é que elas podem ser simples ou compostas:
  - Este é o primeiro período – SIMPLES.
  - Turma de Bacharelado em Engenharia de Software e Este é o primeiro período – COMPOSTA.

Casos Especiais

- Tautologia → Só existe se o resultado for sempre verdadeiro para uma proposição composta, independentemente dos seus valores lógicos .
- Ex: P:(p ∧ q) ⟶ (p V q)

p	q	p ∧ q	p v q	(p ∧ q) → (p V q)
V	V	V	V	V
V	F	F	V	V
F	V	F	V	V
F	F	F	F	V

- Contradição → Só existe se o resultado for sempre falso para uma proposição composta, independentemente dos seus valores lógicos.
- Ex: Considere p=q para a proposição P:p ↔ ~q

p	~q	p ↔ ~q
V	F	F
F	V	F

OBS:
Sempre que proposição composta não for uma tautologia e uma contradição, será uma contingência.

UNIASSELVI
GRADUAÇÃO E PÓS

Consistência de dados

- Consistir os dados significa validá-los, ou seja, verificar se os valores digitados como entrada são válidos ou não. Para isso, trabalhamos com as “críticas” ou mensagens de alerta, que serão exibidas ao usuário, referentes aos valores digitados que não atendam às condições estabelecidas. Por exemplo, quando desejamos calcular a nota de um aluno por média aritmética, solicitamos a entrada de 3 números válidos entre 0 e 10. Logo, o intervalo de valores permitidos varia de 0 a 10, não sendo aceitos números negativos e nem maiores do que 10.

- No caso de digitação errada ou proposital por parte do usuário, o algoritmo exibe uma mensagem de crítica, alertando-o para que digite de forma correta os valores solicitados.

Ex:

```
...
nota:=-1;
Enquanto (nota<0) ou (nota>10) faca
   Escreva (Informe a nota do aluno, entre 0 e 10);
      Leia (nota);
      Se (nota<0) ou (nota>10) entao
         Escreva (“Nota inválida.”)
       FimSe;
   FimEnquanto;
...
```

UNIASSELVI
GRADUAÇÃO E PÓS

Modularização

- Um módulo é um grupo de comandos que constitui um trecho de algoritmo com uma função bem definida o mais independente possível das demais partes do algoritmo. Cada módulo, durante a execução do algoritmo, realiza uma tarefa específica da solução do problema e, para tal, pode contar com o auxílio de outros módulos do algoritmo. Desta forma, a execução de um algoritmo contendo vários módulos pode ser vista como um processo cooperativo.

- A construção de algoritmos usando de modulos possui uma série de vantagens:
  - Torna o algoritmo mais fácil de escrever e de ler.
  - Eleva o nível de abstração.
  - Economia de tempo, espaço e esforço.
- O módulo principal solicita a execução dos vários módulos em uma dada ordem, os quais, antes de iniciar a execução, recebem dados do módulo principal e, ao final da execução, devolvem o resultado de suas computações.

- O módulo principal solicita a execução dos vários módulos em uma dada ordem, os quais, são carregados na memória e antes de iniciar a execução, recebem dados do módulo principal, ao final da execução, devolvem o resultado de suas computações e são retirados da memória, liberando recursos de hardware.

Modularização

Exemplo:

```
início
        {declarações  globais}
         módulo principal;
        {declarações  locais}
         início
        {corpo do algoritmo principal}
    fim;
       {definições de outros módulos ou subalgoritmos}
 fim.
```

- Declarações globais: são variáveis, tipos e/ou constantes declaradas no início do algoritmo que podem ser utilizadas por qualquer módulo, inclusive pelo módulo principal.
- Módulo principal: é por onde se inicia a execução do algoritmo.
- Declarações locais: são variáveis, tipos e/ou constantes que só podem ser utilizadas dentro do módulo no qual foram declaradas.
- Corpo do algoritmo: conjunto de ações ou comandos.
- Definição dos módulos ou subalgoritmos: compreende um cabeçalho chamado módulo seguido de um nome, declarações locais e o corpo do subalgoritmo

- A interface de um módulo é definida em termos de parâmetros. Um parâmetro é um tipo especial de variável utilizado pelos módulos para receber ou comunicar valores de dados a outras partes do algoritmo. Existem três categorias de parâmetros:
  - parâmetros de entrada, que permitem que valores sejam passados para um módulo a partir do algoritmo que solicitou sua execução;
  - parâmetros de saída, que permitem que valores sejam passados do módulo para o algoritmo que solicitou sua execução;
  - parâmetros de entrada e saída, que permitem tanto a passagem de valores para o módulo quanto a passagem de valores do módulo para o algoritmo que solicitou sua execução.
- Quando criamos um módulo, especificamos o número e os tipos das variáveis correspondentes aos parâmetros que ele necessita, isto determina a interface do módulo.

Exemplo de algoritmo com módulos

```
Algoritmo media
                    nome      texto;
                    nota1     numerico;
                    nota2      numerico;
                    nota3      numerico;
                    Situacao  texto;
Início
    Início
          Leia(nome);
          Leia(nota1);
          Leia(nota2);
          Leia(nota3);
         --Chamada do módulo de cálculo
 calculo;
   Fim;
```

```
Módulo calculo;
   Início
         Media := (nota1+nota2+nota3) / 3;
         Se media >= 7
               Situacao := “Aprovado”;
          Senão
            Situacao := “Reprovado”;
          Fim-se
         Escreva(media,situacao);
    Fim;
Fim.
```

- Considere um aluno com três notas. Com base nisso, calcularemos a média deste aluno e exibiremos sua situação de acordo com a média obtida.
- Em linguagens de programação, os módulos ou subalgoritmos, são chamados de procedimentos ou funções.

UNIASSELVI
GRADUAÇÃO E PÓS

Funções e Procedimentos em Portugol

- São duas as formas de definição de módulos em Portugol e nas linguagens de programação.

  - **Função:** é um módulo que produz um único valor de saída, que será usado em uma atribuição ou envolvido em alguma expressão, ou ainda exposto como uma saída para o usuário. Ela pode ser vista como uma expressão que é avaliada para um único valor, sua saída, assim como uma função em Matemática.
  - **Procedimento:** é um tipo de módulo usado para várias tarefas, não produzindo valores de saída, somente modificam o valor de uma variável do algoritmo que a chamou. As diferenças entre as definições de função e procedimento permitem determinar se um módulo para uma dada tarefa deve ser implementado como uma função ou procedimento. Nas definições acima, estamos considerando que a produção de um valor de saída por uma função é diferente da utilização de parâmetros de saída ou de entrada e saída. Veremos mais adiante que os parâmetros de saída e de entrada e saída, diferentemente do valor de saída produzido por uma função

Passagem de Parâmetros

- Quando construímos algoritmos de forma modularizada é muito comum precisar passar informações de um módulo para outro. Essas informações são denominadas parâmetros. Os parâmetros são responsáveis por estabelecer a comunicação entre os módulos. Um parâmetro que sai de um módulo é um valor de entrada para o módulo que está sendo chamado, sendo que a lógica do módulo é que vai definir a forma de manipulação da informação: se será somente para leitura ou se o valor será alterado. Caso está informação seja alterada no novo módulo, isto não significa que é alterada no módulo de origem, onde permanece inalterada. Um parâmetro também pode ser chamado de argumento (MANZANO, 2000). Existem dois tipos distintos de parâmetros:
- **Passagem de parâmetros por valor:** Consiste em copiar o valor das variáveis locais usadas na lógica e passá-las para outro módulo sem alterar informações originais.
- **Passagem de parâmetros por referência:** Neste caso, é feita a cópia do endereço da memória onde a variável está armazenada. Nesse mecanismo, outra variável ocupando outro espaço diferente na memória não armazena o dado em si, e sim o endereço onde ele se localiza na memória. Assim, todas as modificações efetuadas nos dados do parâmetro estarão sendo feitas no conteúdo original da variável, ou seja, as alterações ocorrerão de modo global para todas as passagens de parâmetro existentes no código.

Passagem de Parâmetros

Podemos identificar os dois tipos de situação em um mesmo algoritmo.
EXEMPLO 2: Dados dois valores positivos, calcular e exibir o resultado das operações aritméticas efetuadas utilizando passagem por parâmetros.

```
inicio
        módulo principal;
        real: x, y, res;    {variáveis locais}
        inicio
           leia(x);
           leia(y);
           se (x>0) e (y>0) então
                  inicio
                     calc(x,y,'+', res);
                     escreva(res);
                     calc(x,y,'-', res);
                     escreva(res);
                     calc(x,y,'*', res);
                     escreva(res);
                     calc(x,y,'/', res);
                     escreva(res);
                  fim;
        fim;
```

```
módulo calc(real:a,b;caracter:oper; ref real:re);
        inicio
           escolha oper
                  caso '+' : re ← a + b;
                  caso '-' : re ← a - b;
                  caso '*' : re ← a * b;
                  caso '/' : re ← a / b;
           fimEsco;
        fim;
fim.
```

a, **b** e **oper** são passados por parâmetro e **re** é passado por referência

UNIASSELVI
GRADUAÇÃO E PÓS

Passagem de Parâmetros

EXEMPLO 4: O mesmo do exemplo 2 modificado usando retorne. Observe que sempre irá retornar um único resultado de cada vez.

```
inicio
        modulo principal;
        real: x, y;    {variáveis locais}
         início
            leia(x);
            leia(y);
            se (x>0) e (y>0) entao
                    início
                       escreva(calc(x,y,'+'));
                       escreva(calc(x,y,'-'));
                       escreva(calc(x,y,'*'));
                       escreva(calc(x,y,'/'));
                    fim;
    fim;

    modulo calc(real:a,b;caracter:oper);
    início
       escolha oper
              caso '+' : retorne(a + b);
              caso '-' : retorne(a – b);
              caso '*' : retorne(a * b);
              caso '/' : retorne (a / b);
       fim;
    fim;
fim.
```

UNIASSELVI
GRADUAÇÃO E PÓS

Resumindo

- VARIÁVEIS GLOBAIS são as aquelas declaradas logo após o cabeçalho do programa (seu título) e antes do comando de início (Inicio) da execução do programa principal. Elas podem ser utilizadas dentro de todo o programa onde foram criadas.
- VARIÁVEIS LOCAIS são declaradas dentro dos subprogramas e só podem ser usadas dentro dos deles. Elas não podem ser usadas pelo programa principal.
- FUNÇÃO é forma de escrita cria um vínculo entre o programa principal e o subprograma e apresenta as seguintes características:
  - Toda função tem um nome.
  - Toda função pode ou não receber parâmetros ou argumentos de entrada.
  - Toda função retorna, obrigatoriamente, um valor de um único tipo de dado (data, texto ou número).
- PROCEDIMENTOS também precisam ter um nome definido e, caso exista, deve ocorrer também a passagem da lista de parâmetros como entrada para a lógica que será desenvolvida. Na prática, um procedimento se difere das funções porque podem não retornar nenhum, um ou mais de um valor, com tipos de dados distintos, por exemplo: poderia retornar um valor texto, um valor numérico e uma data, na mesma passagem de parâmetros de devolução ao módulo principal.

Vetores e Matrizes

- São estruturas de dados muito simples que podem nos ajudar muito quando temos muitas variáveis do mesmo tipo em um algoritmo, em que cada item do vetor (ou matriz) é acessado por um número chamado de índice.
- Vetor (array unidimensional) é uma variável que armazena várias variáveis do mesmo tipo. Voltando à metáfora das “gavetas”, um Array é uma “gaveta” da memória em que se pode alocar um conjunto de vaiáveis do mesmo tipo.

i1	i2	i3	i4	i5

- Matriz (array multidimensional) é um vetor de vetores. É como se ela fosse um vetor em que cada posição do seu índice pudesse alojar outro vetor.

[i,j]	i1	i2	i3	i4
j1	[1,1]	[2,1]	[3,1]	[4,1]
j2	[1,2]	[2,2]	[3,2]	[4,2]
j3	[1,3]	[2,3]	[3,3]	[4,3]

UNIASSELVI
GRADUAÇÃO E PÓS

Vetores e Matrizes

- Imagine que é preciso armazenar as notas AV1 de 100 alunos para, depois, calcular a média delas. Isso pode ser feito declarando 100 variáveis distintas, ou usando uma estrutura de repetição com a junção de uma variável de leitura e um acumulador de repetição, formando os índices de um Array. Na verdade serão criadas as 100 vairáveis, mas sem precisar nomear uma a uma. Um Array (vetor) é um conjunto de variáveis de mesmo tipo, referenciadas por um nome comum e indexadas. O acesso a cada uma das variáveis no processo de indexação consiste em atribuir um índice para acessar cada uma das variáveis. Tudo feito com muita facilidade!
- Caso fosse preciso associar as notas AV1, AV2, AV3 e AV4 de cada um desses alunos, para calcular suas médias individuais, isto poderia ser feito com uma matriz. Para cada posição de índice do vetor principal (i), seria associado um novo vetor (j) de 4 posições. O número do índice do vetor principal poderia também ser relacionado ao número do diário para associar aos respectivos alunos.

Exemplo de Vetor (em Pascal)

```
Program  MediaAV1;
var
        AV1: array[1..100] of real;
        soma, media : real;
        i : integer;
begin
        soma := 0;
        for i := 1 to 100 do
                begin
                        WriteIn (‘Informe a nota AV1 do aluno ‘, i, ‘: ‘);
                        read (AV1[i]);
                        soma := soma +AV1[i];
                end;
        media := soma/i;
        writeln('O valor da media das notas é: ',media:2:2);
        Readkey ();
end.
```

UNIASSELVI
GRADUAÇÃO E PÓS

Exemplo de Matriz (em Pascal)

```
Program  MediaAV1;
var
        AV1: array[1..100, 1..4] of real;
        soma, media : real;
        i,j : integer;
begin
        soma := 0;
        for i := 1 to 100 do
                begin
                    for j:=1 to 4 do
                        begin
                            Writeln (‘Informe a nota AV’, j,’ do aluno ‘, i, ‘: ‘);
                            read (AV[i,j]);
                            soma := soma +AV[i,j];
                        end;
                        media := soma/j;
                          writeln('O valor da média das notas do aluno ‘, i, ‘é: ',media:2:2);
        end;
          Readkey ();
end.
```

UNIASSELVI
GRADUAÇÃO E PÓS

Vetores e Matrizes

- São estruturas de dados muito simples que podem nos ajudar muito quando temos muitas variáveis do mesmo tipo em um algoritmo, em que cada item do vetor (ou matriz) é acessado por um número chamado de índice.

- Vetor (array unidimensional) é uma variável que armazena várias variáveis do mesmo tipo. Voltando à metáfora das “gavetas”, um Array é uma “gaveta” da memória em que se pode alocar um conjunto de vaiáveis do mesmo tipo.
- Matriz (array multidimensional) é um vetor de vetores. É como se ela fosse um vetor em que cada posição do seu índice pudesse alojar outro vetor.

Muito Obrigado!
Vejo vocês na próxima segunda-feira!

e-mail:
professorjosepauloviana@gmail.com

UNIASSELVI
GRADUAÇÃO E PÓS