Modularidade e Estruturas de Dados

Objetivos:

- Introduzir noção de procedimentos
- Iniciar estudo sobre estruturas de dados

Exercício

• Ler trincas de números inteiros (a,b,c) até que seja lido a trinca (0,0,0). Para cada trinca, se a for diferente de zero calcular as raízes reais da equação do segundo grau $ax^2 + bx + c = 0$. Imprimir mensagem adequada se não houver raízes reais. Use 3 funções: uma para o calculo do discriminante e outras duas para o cálculo das duas raízes.

Procedimentos

• Procedimentos são subprogramas que não devolvem um valor de retorno, sendo semanticamente como novos comandos

• Os conceitos de variáveis globais ou locais bem como de passagem de parâmetros por valor ou por referência se dão exatamente do mesmo modo que no caso das funções

Procedimentos

- Exemplo: trocar duas variáveis

```
program exemplo;
var a,b: integer;
begin
        read (a,b);
        troca (a,b);
        writeln (a,b);
end.
```

Procedimentos

- Exemplo: trocar duas variáveis

```
procedure troca (var a,b: integer);
var temp: integer;
begin
        temp:= a;
        a:= b;
        b:= temp;
end;
```

Brincando com os parâmetros

- O que ocorreria se a passagem fosse por valor?

```
procedure troca (    a,b: integer);
var temp: integer;
begin
        temp:= a;
        a:= b;
        b:= temp;
end;
```

Brincando com os parâmetros

- Vejam o que ocorre se:

procedure troca (var a: integer; b: integer);

procedure troca (a: integer; var b: integer);

Procedure troca;

Uso de temp como variável global

Estruturas de dados

- Até agora vimos um uso muito limitado da memória do computador, através do uso de variáveis “simples”

- Estruturas de dados permitem organizar melhor os dados na memória de maneira a permitir que os algoritmos sejam mais eficientes quando exploram esta organização

Estruturas de dados

- O estudo de estruturas de dados permeia todo um curso de computação

- Neste semestre veremos uma das estruturas de dados mais elementares, que são os vetores, tanto na sua forma unidimensional quanto multidimensional

Vetores

- Vetores, ou arrays unidimensionais, são basicamente um conceito que permite utilizar grandes partes da memória através de um único nome

- Com uso sistemático e criterioso por parte do programador é possível explorar este recurso de maneira a obter tanto eficiência quanto modularidade

Vetores

- Declaração:

Var V: ARRAY [<ini>..<fim>] of <tipo>;

<ini> e <fim> são constantes inteiras
<tipo> é qualquer outro tipo já definido

Vetores

- Exemplos:

```
Var V: ARRAY [1..10] of real;
    W: ARRAY [-10..-1] of integer;
    B: ARRAY [2..11] of boolean;
```

Motivação

- Ler vários números até encontrar um zero, e imprimir a soma deles

Motivação

```
Begin
    Soma := 0;
    Read (n);
    While n <> 0 do
    Begin
        Soma := Soma + n;
        Read (n);
    End;
    Writeln (Soma);
End.
```