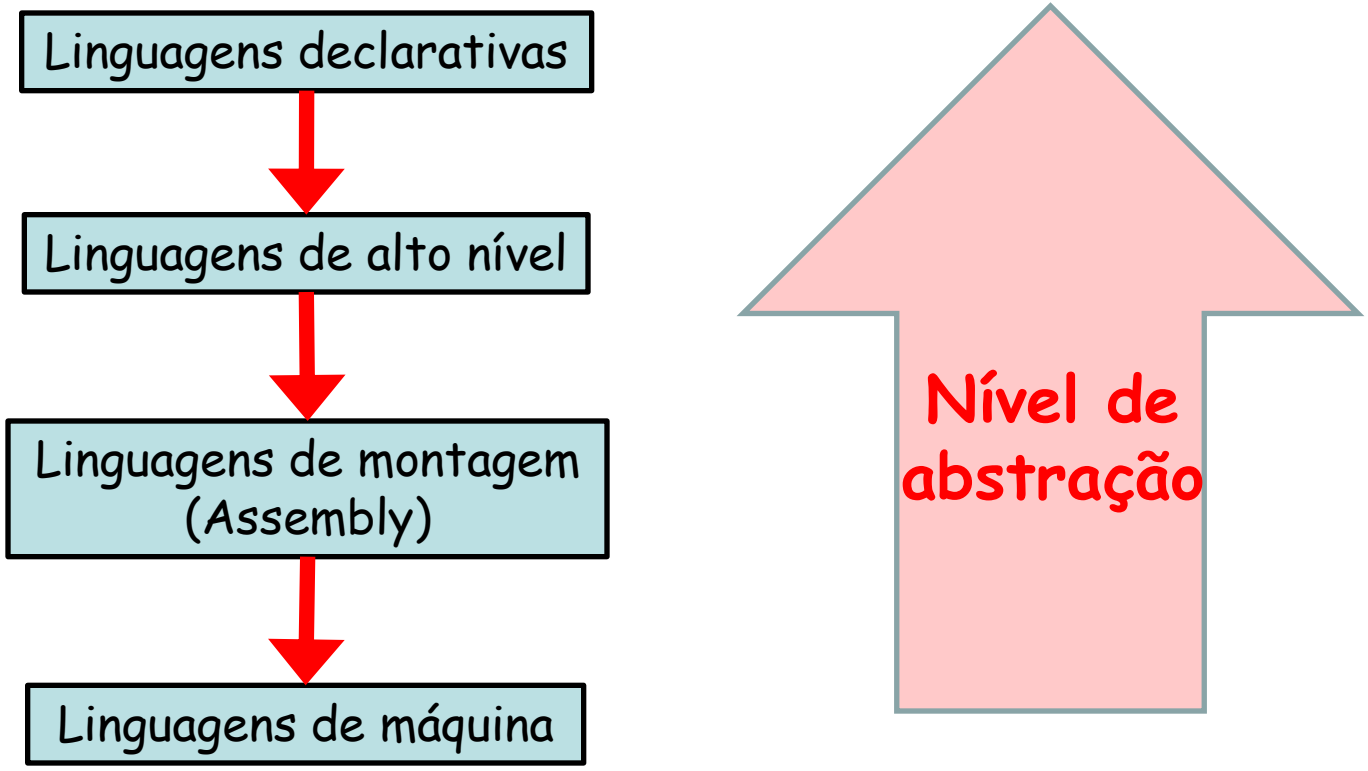
Computação I

Prof. Miguel Elias Mitre Campista

http://www.gta.ufrj.br/~miguel

Introdução ao Pascal

Níveis de Linguagens de Programação



Níveis de Linguagens de Programação

- Linguagens declarativas
  - Linguagens expressivas como a linguagem oral
    - Expressam o que fazer ao invés de como fazer
- Linguagens de alto nível
  - Linguagens típicas de programação
    - Permitem que algoritmos sejam expressos em um nível e estilo escrita fácil para leitura e compreensão
    - Possuem características de portabilidade já que podem ser transferidas de uma máquina para outra
- Linguagens de montagem e linguagem de máquina
  - Linguagens que dependem da arquitetura da máquina
    - Linguagem de montagem é uma representação simbólica da linguagem de máquina associada

Como um Programa é Executado?

- Linguagens de programação
  - São projetadas em função da facilidade na construção e confiabilidade dos programas
    - Quanto mais próximo a linguagem de programação estiver da forma de raciocínio humano, mais intuitivo se torna o programa e mais simples é a programação

```
#include <stdio.h>

main() {

    ENQUANTO condição satisfeita FAÇA

            execute ação 1;

    FIM DO ENQUANTO

    imprimir "Acabou";

}
```

Como um Programa é Executado?

- Entretanto, computadores não entendem a linguagem humana...
  - Computadores entendem sequências de 0's e 1's
    - Chamada de linguagem de máquina

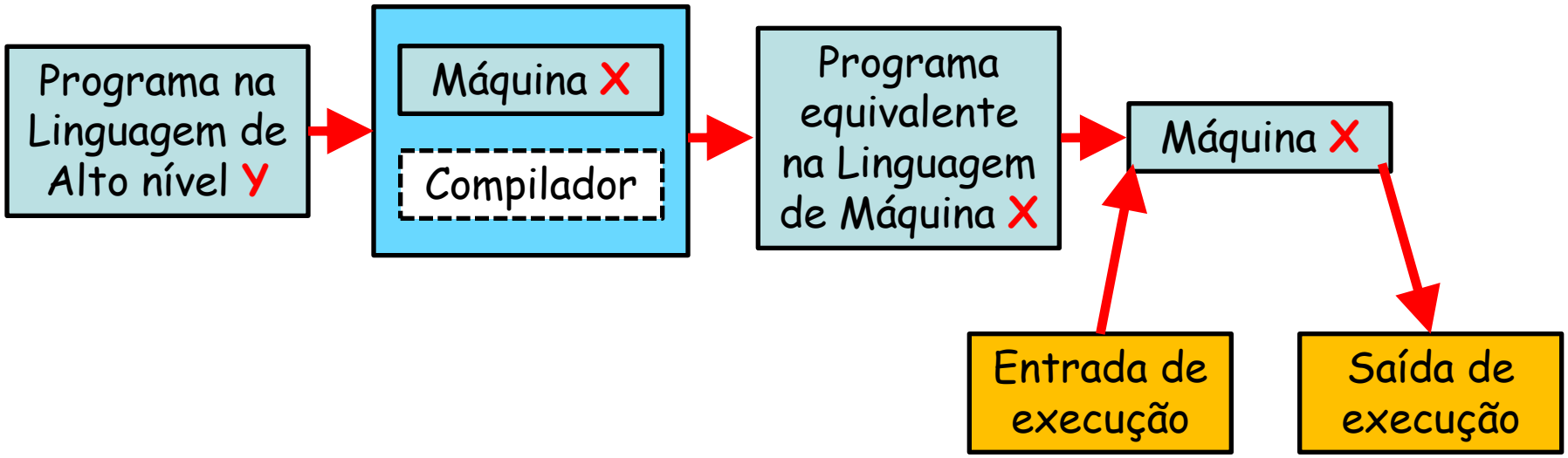
```
#include <stdio.h>
main() {
    ENQUANTO condição satisfeita FAÇA
            execute ação 1;
    FIM DO ENQUANTO
    imprimir "Acabou";
}
```

1	0	1	1	0
0	0	1	1	0
		...		
0	1	0	1	0
0	1	0	0	1

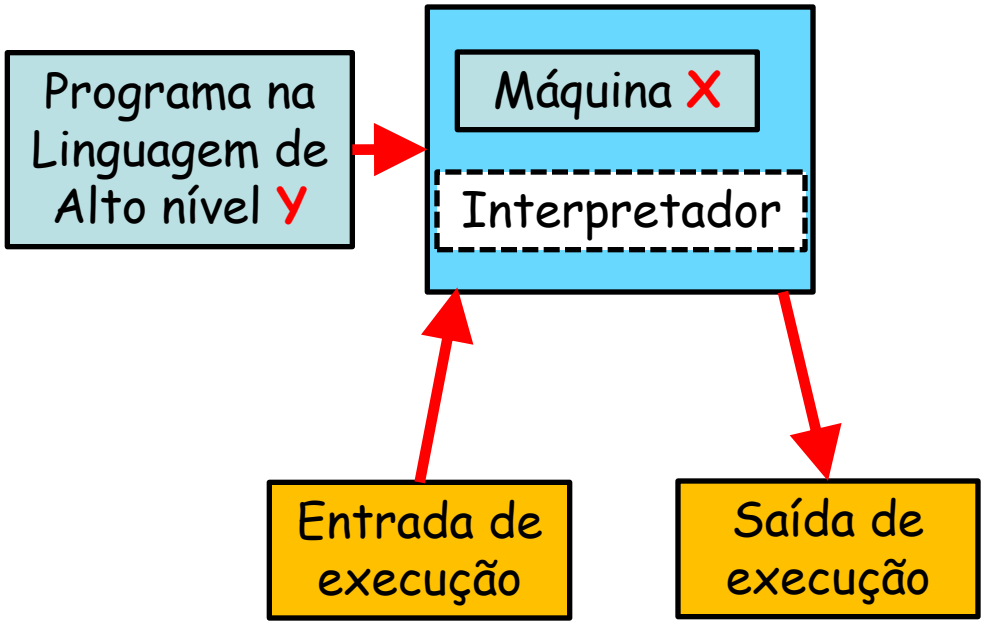
Níveis de Linguagem de Programação

- Existem duas maneiras para decodificar programas
  - Programa em linguagem de nível alto para programa em linguagem de nível baixo
    - Interpretação
    - Tradução

Programa Compilador



Programa Interpretador



Linguagem de Programação de Alto Nível

- Atualmente, há muitas linguagens de programação em alto-nível
  - C, C++, C#
  - Java
  - Perl, Python, Lua, Shell Script
  - Fortran, Cobol, **Pascal**

Histórico do Pascal

- Criado por Niklaus Wirth, na década de 60
  - Professor do departamento de informática da Escola Politécnica de Zurique (Suíça)
    - Objetivo era ensinar seus alunos a programar em PLI e ALGOL 60
  - Linguagem criada com objetivo de simplicidade para facilitar a compreensão

Tipos de Dados e Instruções Primitivas

- Estrutura de Dados
  - Representação da informação que ofereça facilidade de acesso e manipulação

Tipos de dados inteiros	Faixa de abrangência
shortint	De -128 até 127
integer	De -32.768 até 32.767
longint	De -2.147.483.648 até 2.147.483.647
byte	De 0 até 255
word	De 0 até 65.535

Tipos de Dados e Instruções Primitivas

- Estrutura de Dados
  - Representação da informação que ofereça facilidade de acesso e manipulação

Tipos de dados reais	Faixa de abrangência
real	De 2,9 E-39 até 1,7 E+38
single	De 1,5 E-45 até 3,4 E+38
double	De 5,0 E-324 até 1,7 E+308
extended	De 3,4 E-4.932 até 1,1 E+4.932
comp	De -9,2 E+18 até 9,2 E+18

Tipos de Dados e Instruções Primitivas

- Estrutura de Dados
  - Representação da informação que ofereça facilidade de acesso e manipulação

Tipos de dados caracteres (Devem vir sempre entre ' ')
string
char

Tipos de Dados e Instruções Primitivas

- Estrutura de Dados
  - Representação da informação que ofereça facilidade de acesso e manipulação

Tipos de dados lógicos
true
false

Estrutura de um Programa em Pascal

- Cabeçalho do programa
  - Área utilizada para fazer identificação de um programa
    - Uso de nome
      - Obs.: Nenhuma variável pode possuir o mesmo nome que o programa
      - Ex.: `program SOMA;`

```
program nome_do_programa;
```

Estrutura de um Programa em Pascal

- Área de declarações
  - Área utilizada para validar o uso de qualquer tipo de identificador que não seja pré-definido
    - **var ←**
    - uses
    - label
    - const
    - type
    - procedure
    - function

Estrutura de um Programa em Pascal

- Área de declarações
  - Área utilizada para validar o uso de qualquer tipo de identificador que não seja pré-definido
    - var
      - Ex.: var

        nome: string;
        idade: int;
        altura, peso: real;

```
var
      nome_variavel1: tipo1;
      nome_variavel2: tipo2;
      nome_variavel3, nome_variavel4: tipo3;
```

Estrutura do Programa em Pascal

- Corpo do programa
  - O programa propriamente dito em Pascal está escrito na área denominada **corpo do programa**
    - Área tem início com a instrução **begin** e término com a instrução **end**, seguida do símbolo ponto (.)
      - Ex.:
```
begin
    writeln(IDADE);
end.
```

```
begin
      <instruções>
end.
```

Estrutura do Programa em Pascal

- Comentário
  - Parte do programa ignorada pelo compilador
    - Serve para comentar o código inserido
      - Sentença comentada deve vir entre chaves ({ })
        - » Ex.: var
          { idade do usuário }
          idade: integer;

```
begin
      <instruções>
      {comentario}
end.
```

Estrutura do Programa em Pascal

- Entrada e saída de dados
  - Entrada padrão através do teclado
    - Função readln
      - Ex.: `readln(var);` → atribui o valor lido do teclado à variável **var**
  - Saída padrão através da tela
    - Função writeln
      - Ex.: `writeln(var);` → escreve o valor da variável **var** na tela

Primeiro Exemplo

- Escrever um programa que digite na tela 'Hello, world!'

?

Primeiro Exemplo

```
program HELLO;

begin
        writeln('Hello, world!');
end.
```

Compilação
fpc -o<executavel> <codigo_fonte>

```
shell$>fpc -oex1 exemplo1.pas
```

Segundo Exemplo

- Escrever um programa que digite na tela 'Hello, ' seguido do seu nome

?

Segundo Exemplo

```
program HELLO;

var
        NOME: string;

begin
        { Solicita ao usuário a entrada do nome }
        writeln('Digite o seu nome, por favor');
        readln(NOME);

        writeln('Hello, ', NOME, '!');
end.
```

Terceiro Exemplo

- Escrever um programa que some três inteiros passados pelo teclado e imprima na tela o resultado da soma

?

Terceiro Exemplo

```
program SOMA;

var
        RESULTADO, P1, P2, P3: integer;

begin
        { Solicita ao usuário os três inteiros }
        writeln('Entre com as tres parcelas');

        { Lê os três inteiros do teclado }
        readln(P1);
        readln(P2);
        readln(P3);

        { Calcula a soma }
        RESULTADO := P1 + P2 + P3;

        {Imprime o resultado }
        writeln('O resultado eh: ', RESULTADO);
end.
```

Quarto Exemplo

- Escrever um programa que calcule o salário líquido de um trabalhador. Para isso, deve ser calculado o salário bruto como sendo o produto entre o valor da hora e o número de horas trabalhadas e, em seguida, calculado o INSS sobre o salário bruto para encontrar o salário líquido

?

Quarto Exemplo

```
program SALARIO;

var
        HT, VH, PD, TD, SB, SL: real;

begin
        write('Quantas horas de trabalho? ');
        readln(HT);

        write('Qual o valor da hora? ');
        readln(VH);

        write('Qual o percentual de desconto? ');
        readln(PD);

        SB := HT * VH;
        TD := (PD/100) * SB;
        SL := SB - TD;

        { Imprime os resultados }
        writeln('Salario bruto: ', SB);
        writeln('Total de desconto: ', TD);
        writeln('Salario líquido: ', SL);
end.
```

Quarto Exemplo

```
program SALARIO;

var
        HT, VH, PD, TD, SB, SL: real;

begin
        write('Quantas horas de trabalho? ');
        readln(HT);

        write('Qual o valor da hora? ');
        readln(VH);

        write('Qual o percentual de desconto? ');
        readln(PD);

        SB := HT * VH;
        TD := (PD/100) * SB;
        SL := SB - TD;

        { Imprime os resultados }
        writeln('Salario bruto: ', SB:7:2);
        writeln('Total de desconto: ', TD:7:2);
        writeln('Salario líquido: ', SL:7:2);
end.
```

Tomada de Decisão

- Desvio condicional simples
  - Tomada de decisão que pode gerar um desvio na execução do programa
    - Desvio depende da avaliação de uma sentença lógica em VERDADEIRO ou FALSO
      - Ex.: if (A > 0) then
        writeln('A > 0');

```
if (<condição>) then
     <instrução se verdadeiro>
```

Tomada de Decisão

- Desvio condicional simples
  - Tomada de decisão que pode gerar um desvio na execução do programa
    - Desvio depende da avaliação de uma sentença lógica em VERDADEIRO ou FALSO

```
if (<condição>) then
   begin
     <instrução1 se verdadeiro>
     <instrução2 se verdadeiro>
   end;
```

Tomada de Decisão

- Desvio condicional simples
  - Tomada de decisão que pode gerar um desvio na execução do programa
    - Desvio depende da avaliação de uma sentença lógica em VERDADEIRO ou FALSO
      - Ex.:

```
if (A > 0) then
    begin
        writeln('A > 0');
        A := B + C;
    end;
```

Tomada de Decisão

- Desvio condicional composto
  - Tomada de decisão que gera um desvio na execução do programa
    - Desvio depende da avaliação de uma sentença lógica em VERDADEIRO ou FALSO

```
if (<condição>) then
    begin
       <instrução1 se verdadeiro>
       <instrução2 se verdadeiro>
    end
else
    begin
       <instrução1 se verdadeiro>
       <instrução2 se verdadeiro>
    end;
```

Tomada de Decisão

- Desvio condicional composto
  - Tomada de decisão que gera um desvio na execução do programa
    - Desvio depende da avaliação de uma sentença lógica em VERDADEIRO ou FALSO
      - Ex.: if (A > 0) then

```
        begin
            writeln('A > 0');
            A := B + C;
        end
    else
        begin
            writeln('A < 0');
            A := B - C;
        end;
```

Não se pode colocar ";" antes do **else**!

Quinto Exemplo

- Escrever um programa que ordene duas variáveis inteiras

?

Quinto Exemplo

```
program ORDENA;

var
        TEMP, A, B: integer;

begin
        writeln('Entre com o valor de A:');
        readln(A);
        writeln('Entre com o valor de B:');
        readln(B);

        writeln('Sequencia: ', A, ' e ', B);

        if (A > B) then
                begin
                        TEMP := A;
                        A := B;
                        B := TEMP;
                        writeln('Sequencia ordenada: ', A, ' e ', B);
                end;
end.
```

Sexto Exemplo

- Escrever um programa que ordene duas variáveis inteiras, se elas já estiverem ordenadas, o programa avisa que não há nada para fazer

?

Sexto Exemplo

```
program ORDENA_V2;

var
        TEMP, A, B: integer;

begin
        writeln('Entre com o valor de A:');
        readln(A);
        writeln('Entre com o valor de B:');
        readln(B);

        writeln('Sequencia: ', A, ' e ', B);

        if (A > B) then
                begin
                        TEMP := A;
                        A := B;
                        B := TEMP;
                        writeln('Sequencia ordenada: ', A, ' e ', B);
                end
        else
                writeln('Sequencia jah ordenada: ', A, ' e ', B);
end.
```

Operadores Lógicos

- Ou
  - OR
    - Ex.: se (<condição1>) or (<condição2>) then
      sentença1;
- E
  - AND
    - Ex.: se (<condição1>) and (<condição2>) then
      sentença1;
- Negação
  - NOT
    - Ex.: se not (<condição1>) then
      sentença1;

Sétimo Exemplo

```
program TEMPERATURA;

var
        TEMP: real;

begin
        writeln('Qual a temperatura de hoje? ');
        readln(TEMP);

        if ( TEMP > 10 ) and ( TEMP <= 25 ) then
                writeln('Temperatura agradavel!')
        else
                if not (TEMP < 25) then
                        writeln('Muito calor!')
                else
                        writeln('Muito frio!');

end.
```

Repetição

- Utiliza o conceito de loop de programação
  - Repetição é realizada até que uma condição falhe
    - Modos: utilizando
      repeat-until
      while-do

```
repeat
      <instrução1 se verdadeiro>
      <instrução2 se verdadeiro>
until (<condição>);
```

Repetição

- Utiliza o conceito de loop de programação
  - Repetição é realizada até que uma condição falhe
    - Modos: utilizando
      repeat-until
      while-do

```
while (<condição>) do
     begin
          <instrução1 se verdadeiro>
          <instrução2 se verdadeiro>
     end;
```

Oitavo Exemplo

```
program FATORIAL;

var
        NUMBER, TEMP, RESULTADO: integer;

begin
        writeln('Entre com o numero inteiro:');
        readln(NUMBER);

        RESULTADO := NUMBER;
        TEMP := NUMBER;

        repeat
                TEMP := TEMP - 1;
                RESULTADO := RESULTADO * TEMP;
        until TEMP = 1;

        writeln('Fatorial de ', NUMBER, ' eh: ', RESULTADO);
end.
```

Nono Exemplo

```
program FATORIAL;

var
        NUMBER, TEMP, RESULTADO: integer;

begin
        writeln('Entre com o numero inteiro:');
        readln(NUMBER);

        RESULTADO := NUMBER;
        TEMP := NUMBER;

        while (TEMP > 1) do
                begin
                        TEMP := TEMP - 1;
                        RESULTADO := RESULTADO * TEMP;
                end;

        writeln('Fatorial de ', NUMBER, ' eh: ', RESULTADO);
end.
```

Repetição

- Utiliza o conceito de loop de programação
  - Repetição pode ser realizada com variável de controle
    - Modos: utilizando

      for-to //Loop com variável crescente

      for-downto //Loop com variável decrescente

```
for <variável> := <inicio> to <fim> do
     begin
          <instrução1 se verdadeiro>
          <instrução2 se verdadeiro>
     end;
```

Repetição

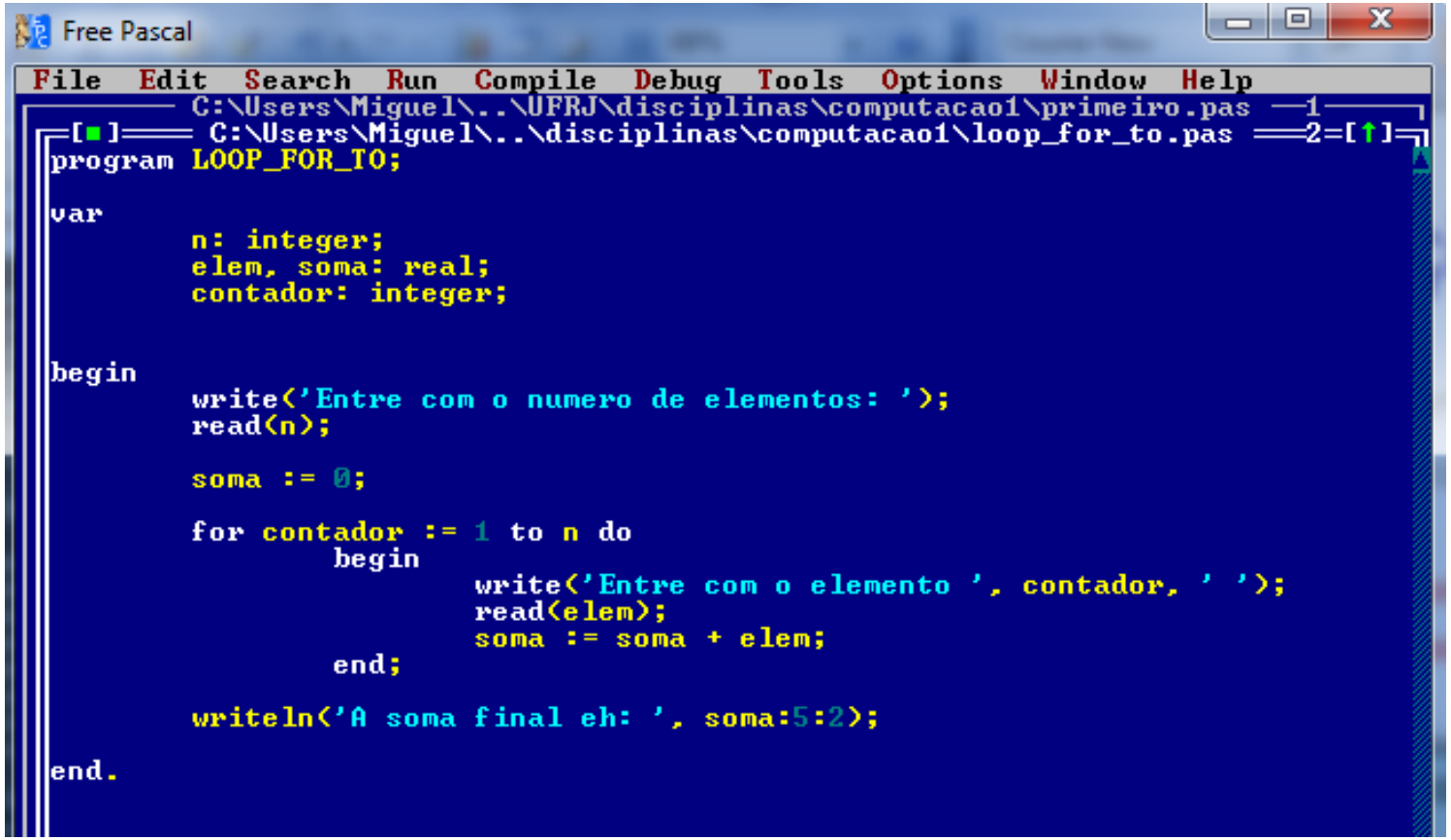
- Utiliza o conceito de loop de programação
  - Repetição pode ser realizada com variável de controle
    - Modos: utilizando

      for-to //Loop com variável crescente

      for-downto //Loop com variável decrescente

```
for <variável> := <inicio> downto <fim> do
     begin
          <instrução1 se verdadeiro>
          <instrução2 se verdadeiro>
     end;
```

Décimo Exemplo



Décimo Primeiro Exemplo

Free Pascal

File Edit Search Run Compile Debug Tools Options Window Help

C:\Users\Miguel\..\UFRJ\disciplinas\computacao1\primeiro.pas —1—

C:\Users\Miguel\..\disciplinas\computacao1\loop_for_to.pas —2—

C:\Users\..\disciplinas\computacao1\loop_for_downto.pas 3

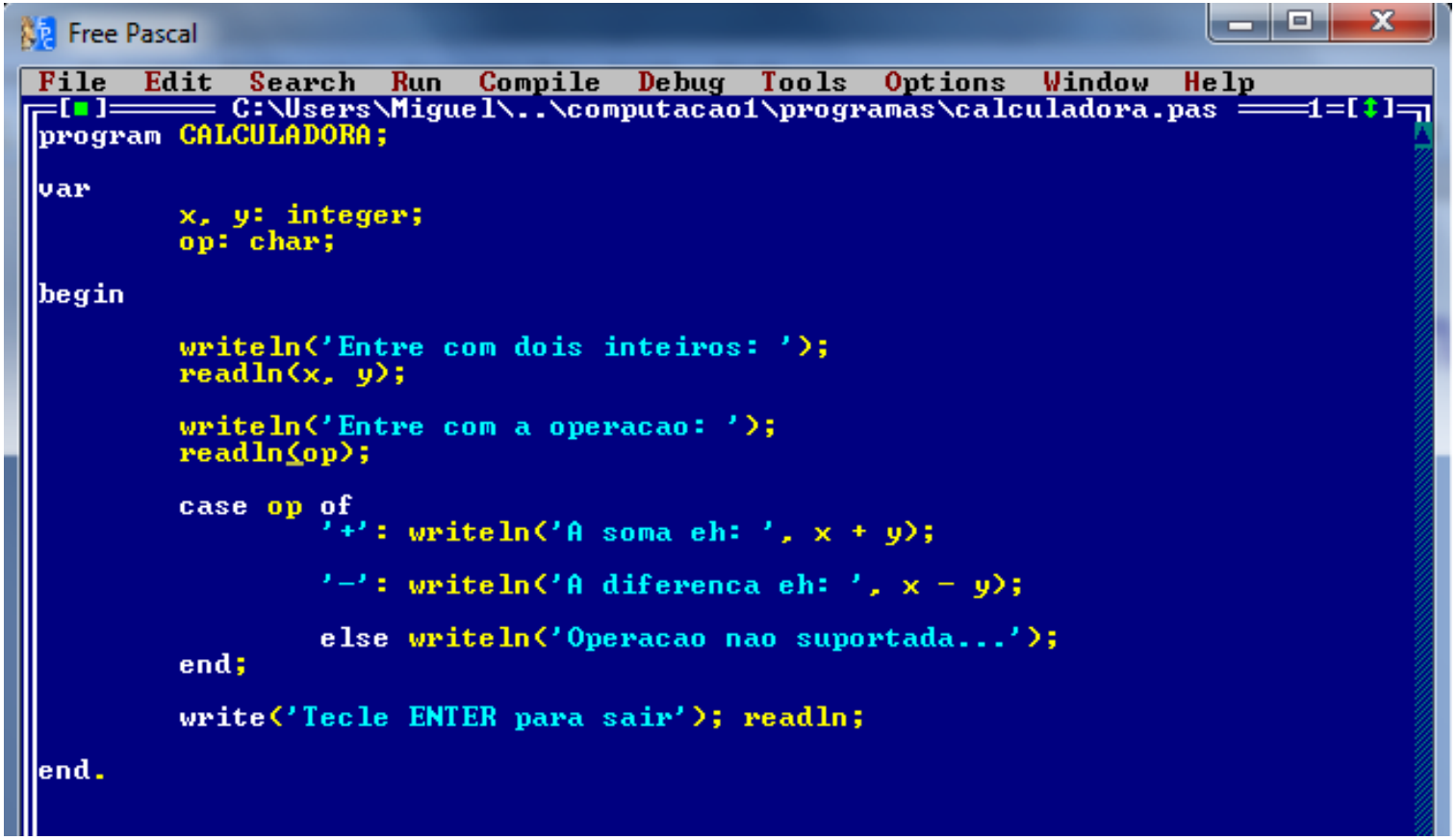
```
program LOOP_FOR_DOWNTO;

var
        n: integer;
        contador: integer;
        elem: real;
        soma: real;

begin
        write('Entre com o numero de elementos: ');
        read(n);

        soma := 0;
        for contador := n downto 1 do
                begin
                   write('Entre com o elemento ', n - contador + 1, ' ');
                   read(elem);
                   soma := soma + elem;
                end;

        writeln;
        writeln('A soma final eh: ', soma);

end.
```

Seleção

- Utiliza o conceito de seleção de sentença
  - Seleciona caso verdadeiro
    - Ex.: case var of
      1: <sentenças>;
      2: <sentenças>;
      else: <sentenças>;
      end;

```
case <variável> of
     valor1: <instrução1 se verdadeiro>
     valor2: <instrução2 se verdadeiro>
     else <instrução2 se verdadeiro>
end;
```

Décimo Segundo Exemplo



Seleção

```
case <variável> of
     valor1:
          begin
          <instrução1 se verdadeiro>
          end;
     valor2:
          begin
          <instrução2 se verdadeiro>
          end;
     else
          begin
          <instrução2 se verdadeiro>
          end;
end;
```

Estrutura de Dados Homogênea

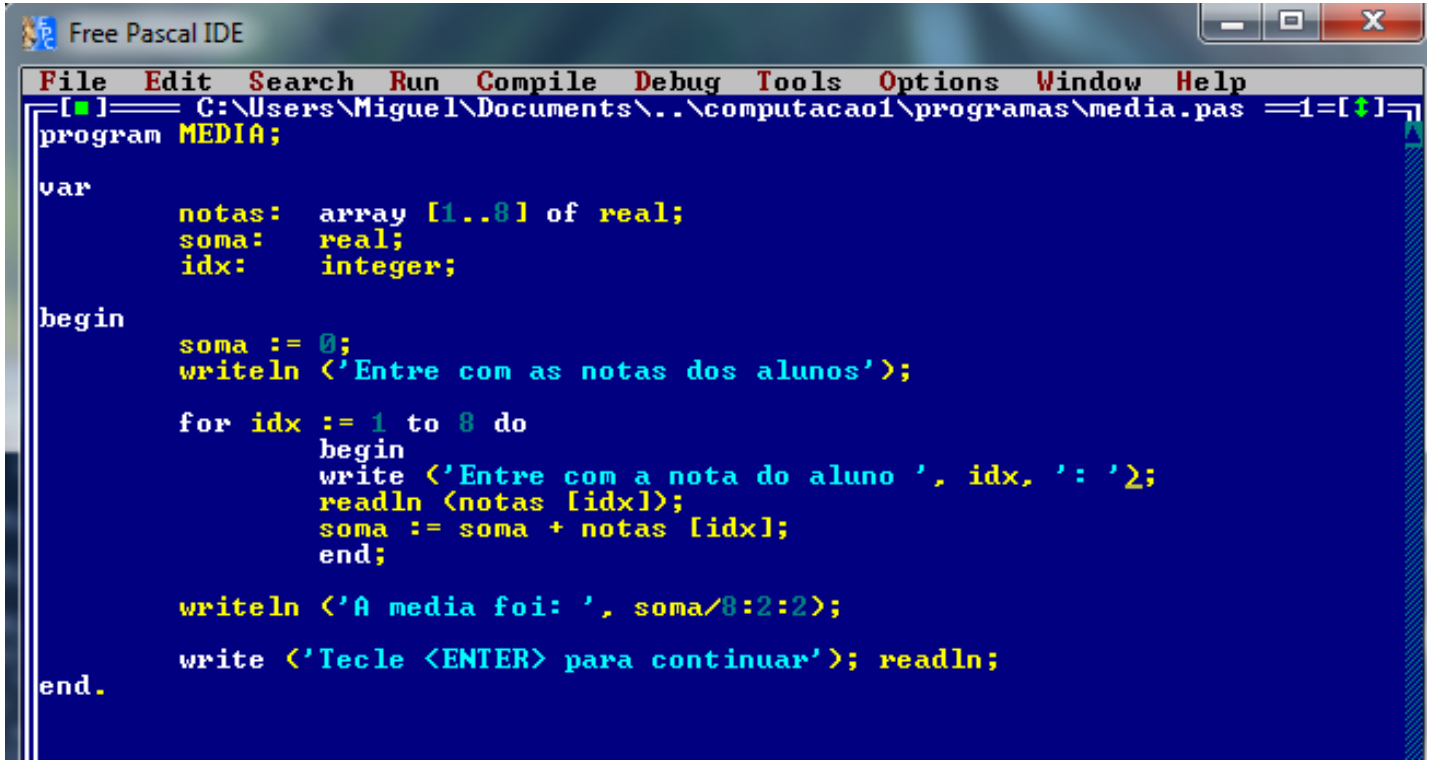
- Matrizes de uma dimensão ou vetores
  - Estruturas de dados que armazenam variáveis do mesmo tipo
    - Estrutura deve ser dimensionada antes do uso por constantes inteiras e positivas
    - Nomes dados às matrizes seguem as mesmas regras de nomenclatura de variáveis simples

Matrizes de uma Dimensão ou Vetores

- Uma matriz de uma dimensão ou vetor é representada por:
  - Nome
  - Tamanho (dimensão)
  - Tipo

```
<matriz>: array[<dimensão>] of <tipo de dados>;
```

Décimo Terceiro Exemplo



Décimo Quarto Exemplo

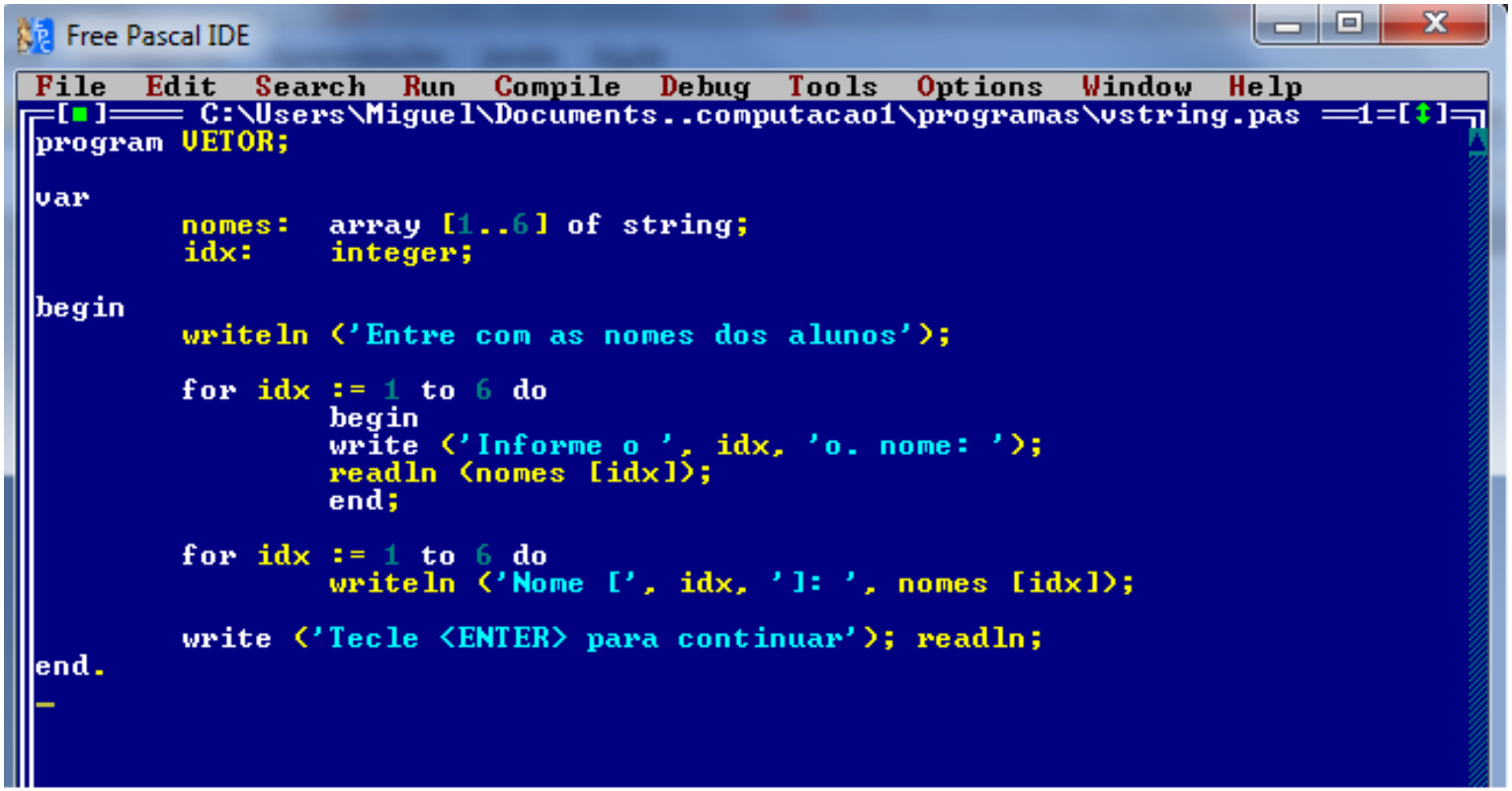
Free Pascal IDE

File Edit Search Run Compile Debug Tools Options Window Help

C:\Users\Miguel\Documents\..\computacao1\programas\media.pas

```
program VETOR;

var
        A, B:  array [1..12] of real;
        idx:    integer;

begin
        writeln ('Entre com as notas dos alunos');

        for idx := 1 to 12 do
                begin
                write ('Informe o ', idx, 'o. valor: ');
                readln (a [idx]);
                end;

        for idx := 1 to 12 do
                begin
                if ( idx mod 2 = 0) then
                        begin
                        b [idx] := 5 * a [idx];
                        end
                else
                        begin
                        b [idx] := 5 + a [idx];
                        end
                end;

        for idx := 1 to 12 do
                writeln (a [idx]:2:2, ' ', b [idx]:2:2);

        write ('Tecle <ENTER> para continuar'); readln;
end.
```

Décimo Quinto Exemplo



Décimo Sexto Exemplo

Free Pascal IDE

File Edit Search Run Compile Debug Tools Options Window Help

C:\Users\Miguel\Documents..computacao1\programas\vstring.pas —1

C:\Users\Miguel\..\computacao1\programas\vstring_ordena. ═2

```
program ORDENA_NOME;

var
        nomes   : array[1..6] of string;
        temp    : string;
        i,j     : integer;

begin
        writeln ('Entre com 6 nomes:');
        writeln;

        for i := 1 to 6 do
                begin
                readln (nomes [i]);
                end;

        for i := 1 to 6 do
                begin
                writeln ('Nome ', i, ': ', nomes [i]);
                end;

        {Ordenação do vetor de strings}
        for i := 1 to 5 do
                for j := i + 1 to 6 do
                        if (nomes [i] < nomes [j]) then
                                begin
                                temp := nomes [i];
                                nomes [i] := nomes [j];
                                nomes [j] := temp;
                                end;


        writeln;
        for i := 1 to 6 do
                begin
                writeln ('Nome Ordenado ', i, ': ', nomes [i]);
                end;

        writeln;
        writeln ('Tecle <ENTER> para continuar.'); readln;
end.
```

Décimo Sétimo Exemplo

Free Pascal

File Edit Search Run Compile Debug Tools Options Window Help

C:\Users\Miguel\..\computacao1\programas\vstring_busca.p

```
program BUSCA_NOME;

var
        nomes    : array[1..6] of string;
        temp     : string;
        i,j      : integer;

        busca    :string;
        achou    :boolean;

begin
        writeln ('Entre com 6 nomes:');
        writeln;

        for i := 1 to 6 do
                begin
                readln (nomes [i]);
                end;

        for i := 1 to 6 do
                begin
                writeln ('Nome ', i, ': ', nomes [i]);
                end;

        {Ordenação do vetor de strings}
        for i := 1 to 5 do
                for j := i + 1 to 6 do
                        if (nomes [i] < nomes [j]) then
                                begin
                                temp := nomes [i];
                                nomes [i] := nomes [j];
                                nomes [j] := temp;
                                end;

        writeln;
        for i := 1 to 6 do
                begin
                writeln ('Nome Ordenado ', i, ': ', nomes [i]);
                end;

        achou := false;
        i := 1;

        writeln ('Entre com o nome a ser buscado:');
        readln (busca);
```

Décimo Sétimo Exemplo

```
        {Busca do elemento no vetor}
        while ((achou = false) and (i <= 6)) do
                begin
                if (nomes [i] = busca) then
                        begin
                        achou := true;
                        end
                else
                        begin
                        i := i + 1;
                        end;
                end;

        if (achou = true) then
                begin
                writeln ('String encontrada na posicao ', i);
                end
        else
                begin
                writeln ('String nao encontrada...');
                end;

        writeln;
        writeln ('Tecle <ENTER> para terminar'); readln;
end.
```

72:37

Estrutura de Dados Homogênea

- Matrizes de mais de uma dimensão
  - Estruturas de dados que armazenam variáveis do mesmo tipo
    - Estrutura deve ser dimensionada antes do uso por constantes inteiras e positivas
    - Nomes dados às matrizes seguem as mesmas regras de nomenclatura de variáveis simples

Matrizes de Mais de uma Dimensão

- Uma matriz de mais de uma dimensão é representada por:
  - Nome
  - Tamanho de cada uma das suas dimensões
    - Caso possua duas: dimensão de linhas e colunas
  - Tipo

```
<matriz>:
     array[<dimensão linha>, <dimensão coluna>]
     of <tipo de dados>;
```

Décimo Oitavo Exemplo

C:\Users\Miguel\Documents..computacao1\programas\matrix.pas

```
program MATRIX;

var
        nomes    : array[1..6, 1..2] of string;
        temp_prim     : string;
        temp_ult      : string;
        i,j      : integer;

        busca    :string;
        achou    :boolean;

begin
        writeln ('Entre com primeiro e ultimo nomes de seis pessoas:');
        writeln;

        for i := 1 to 6 do
                begin
                write ('Primeiro nome de candidato ', i, ': ');
                readln (nomes [i, 1]);
                write ('Ultimo nome de candidato ', i, ': ');
                readln (nomes [i, 2]);
                end;

        for i := 1 to 6 do
                begin
                write ('Nome: ');
                for j := 1 to 2 do
                        begin
                        write (nomes [i, j], ' ');
                        end;
                writeln;
                end;

        {Ordenação do vetor de strings}
        for i := 1 to 5 do
                for j := i + 1 to 6 do
                        if (nomes [i, 1] > nomes [j, 1]) then
                                begin
                                temp_prim := nomes [i, 1];
                                temp_ult := nomes [i, 2];

                                nomes [i, 1] := nomes [j, 1];
                                nomes [i, 2] := nomes [j, 2];

                                nomes [j, 1] := temp_prim;
                                nomes [j, 2] := temp_ult;
                                end;
```

Décimo Oitavo Exemplo

```
        writeln;
        for i := 1 to 6 do
                begin
                write ('Nome Ordenado ');
                for j := 1 to 2 do
                        begin
                        write (nomes [i, j], ' ');
                        end;
                writeln;
                end;

        achou := false;
        i := 1;

        writeln;
        writeln ('Entre com o nome a ser buscado:');
        readln (busca);

        {Busca do elemento no vetor}
        while ((achou = false) and (i <= 6)) do
                begin
                if (nomes [i, 1] = busca) then
                        begin
                        achou := true;
                        end
                else
                        begin
                        i := i + 1;
                        end;
                end;

        if (achou = true) then
                begin
                writeln ('String encontrada na posicao ', i);
                end
        else
                begin
                writeln ('String nao encontrada...');
                end;

        writeln;
        writeln ('Tecle <ENTER> para terminar'); readln;
end.
```

92:22

Registros

- Estrutura de dados composta por dados de tipos diferentes
  - Matriz heterogênea
  - Declarada dentro do bloco `type`
    - Bloco `type` deve ser declarado antes de `var` porque o registro define tipo de dados

```
type
<nome_registro> = record
                    var1: <tipo var1>;
                    var2: <tipo var2>;
                    ...
                    varn: <tipo var n>;
                  end;
```

Registros

- Os registro podem ser usados como tipos

```
type
<nome_registro> = record
                    var1: <tipo var1>;
                    var2: <tipo var2>;
                    ...
                    varn: <tipo var n>;
                  end;
var
var_registro: <nome_registro>
```

Arrays de Registros

- Registros podem ser usados como tipos de arrays

```
type
<nome_registro> = record
                    var1: <tipo var1>;
                    var2: <tipo var2>;
                    ...
                    varn: <tipo var n>;
                  end;
var
var_array: array[1..N] of <nome_registro>
```

Arrays de Registros

- Registros podem conter como um de seus elementos um array

```
type
<nome_reg> = record
                var1: <tipo var1>;
                var2: array[1..4] of <tipo var2>;
                ...
                varn: <tipo var n>;
             end;
var
var_registro: array[1..N] of <nome_reg>
```

Décimo Nono Exemplo

Free Pascal IDE

File Edit Search Run Compile Debug Tools Options Window Help

C:\Users\Miguel\..\computacao1\programas\registros.pas 1=[↕]

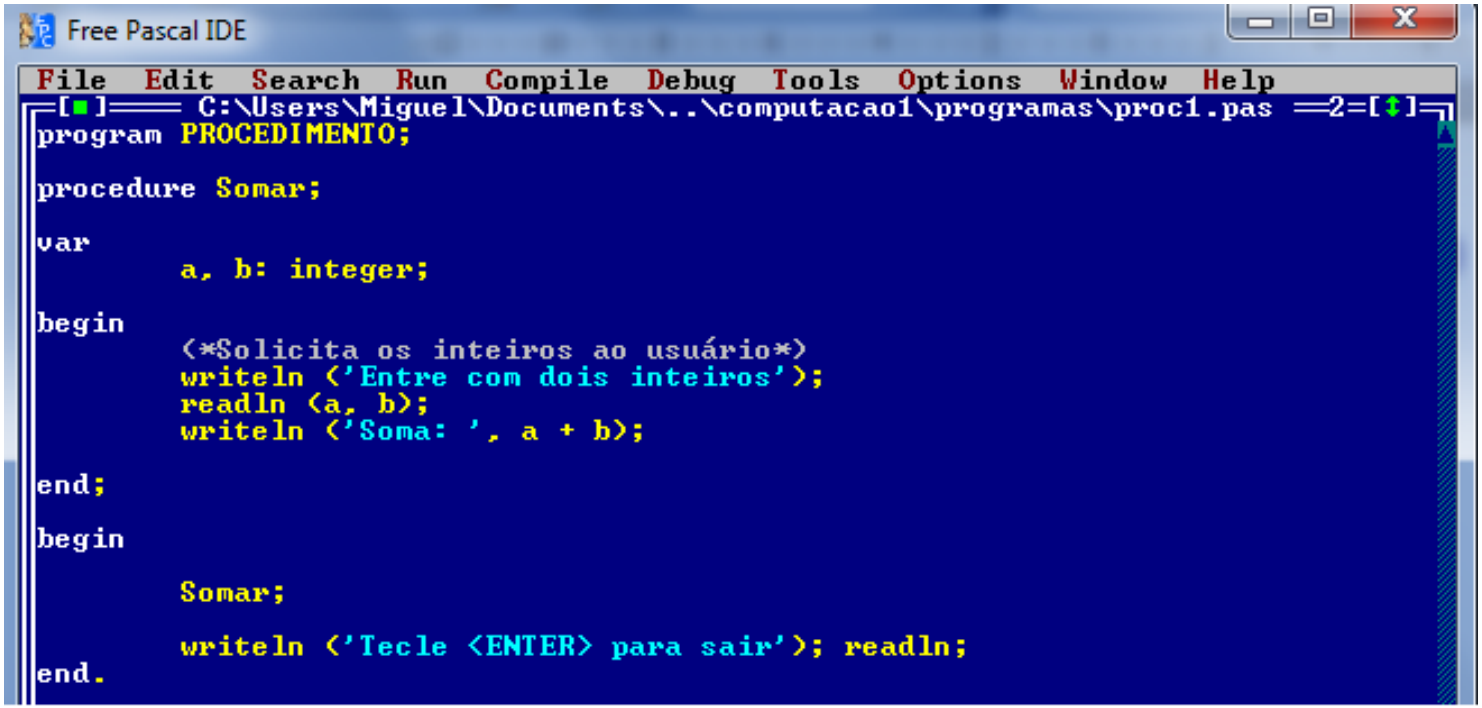
```
program REGISTROS;

type
        cadastro = record
                        nome:   string;
                        idade:  integer;
                   end;

const
        MAX = 3;

var
        i:      integer;
        {array de MAX cadastros}
        aluno:  array[1..MAX] of cadastro;
        {se fosse um cadastro somente...}
        {aluno: cadastro;}

begin
        for i := 1 to MAX do
                begin
                writeln ('Entre com nome e a idade do aluno ', i);
                readln (aluno[i].nome, aluno[i].idade);
                end;

        for i := 1 to MAX do
                writeln ('Aluno[', i, '] ', aluno[i].nome, aluno[i].idade);

        writeln ('Tecle <ENTER> para encerrar'); readln;

end.
```

Procedimento

- Utilizado para modularizar o programa
  - Reuso reduz o tempo de codificação do programa
- Deve sempre ser declarado antes da função principal

```
procedure NOME;
var
      var1: <tipo1>
      var2: <tipo2>
begin
      código...
end;
```

Vigésimo Exemplo



Passagem de Parâmetro para Procedimento

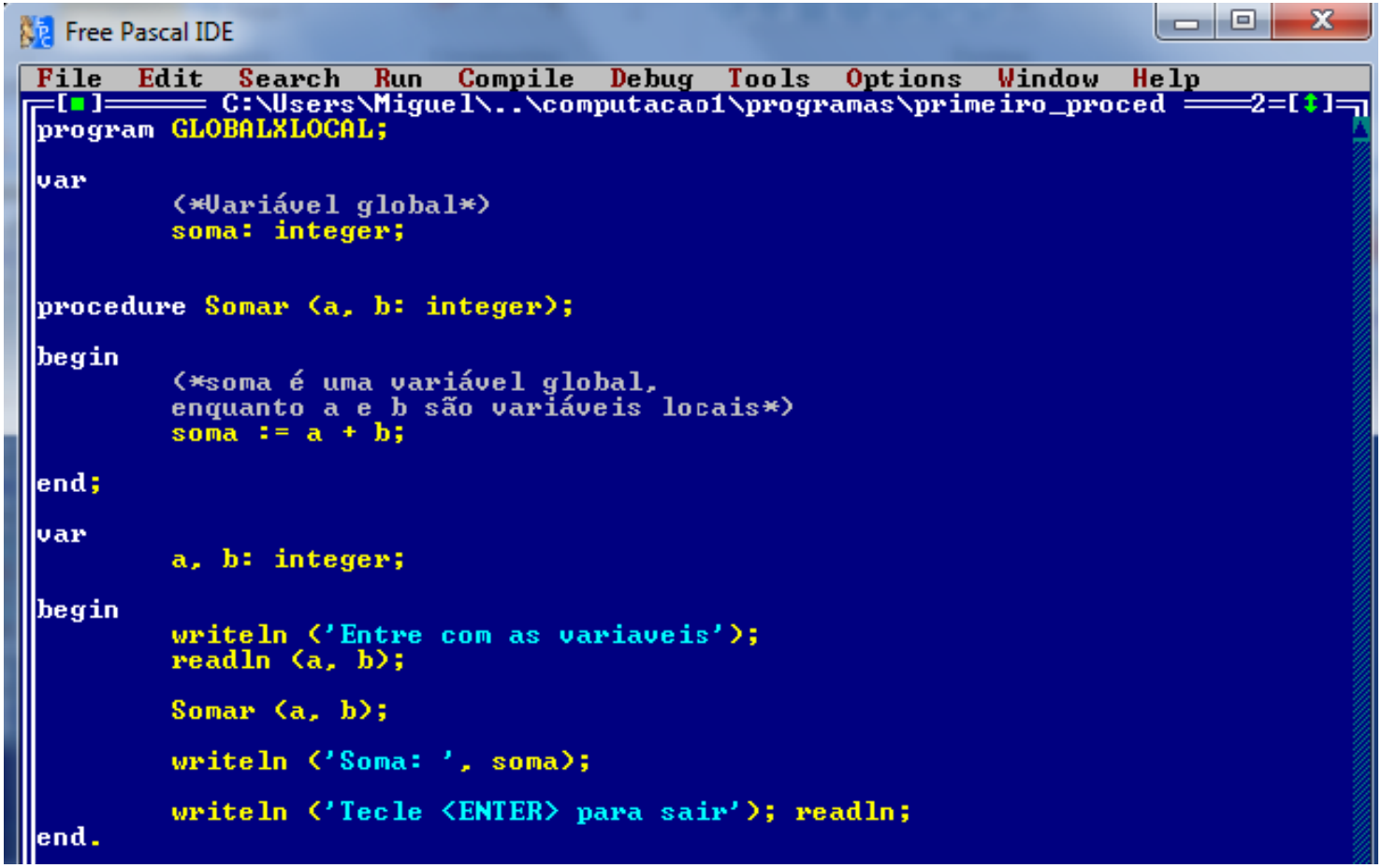
- Um procedimento pode receber uma lista de variáveis como entrada da função

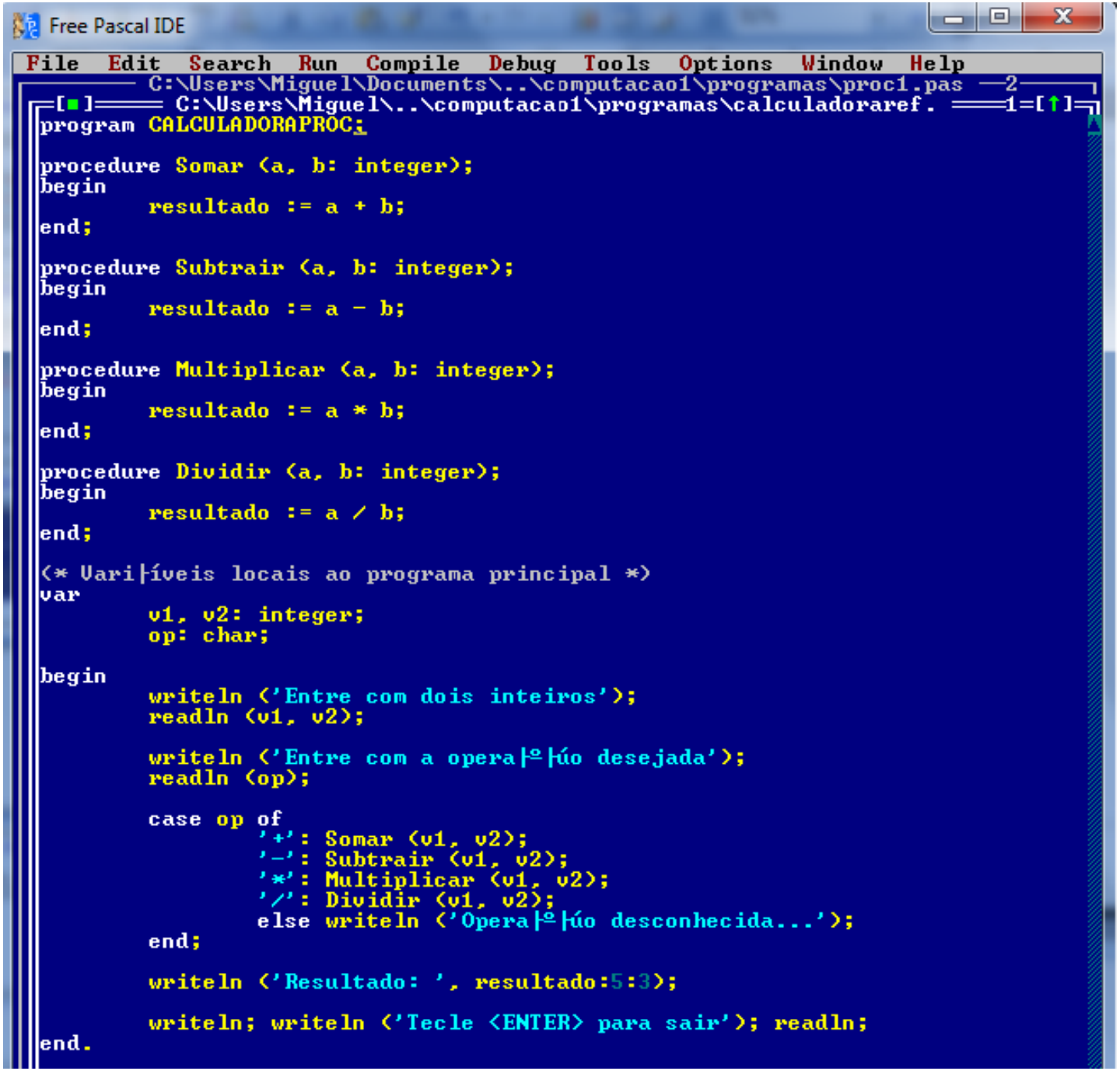
```
procedure NOME (<var1>: <tipo1>; <var2>: <tipo2> );
var
      var_local1: <tipo1>
      var_local2: <tipo2>
begin
      código...
end;
```

Variáveis Globais X Variáveis Locais

- Variáveis globais
  - Existem durante toda a execução do programa
    - Podem ser manipuladas em qualquer ponto do programa
- Variáveis locais
  - Existem durante a execução do procedimento/função
    - Podem ser manipuladas apenas dentro da função na qual foi declarada

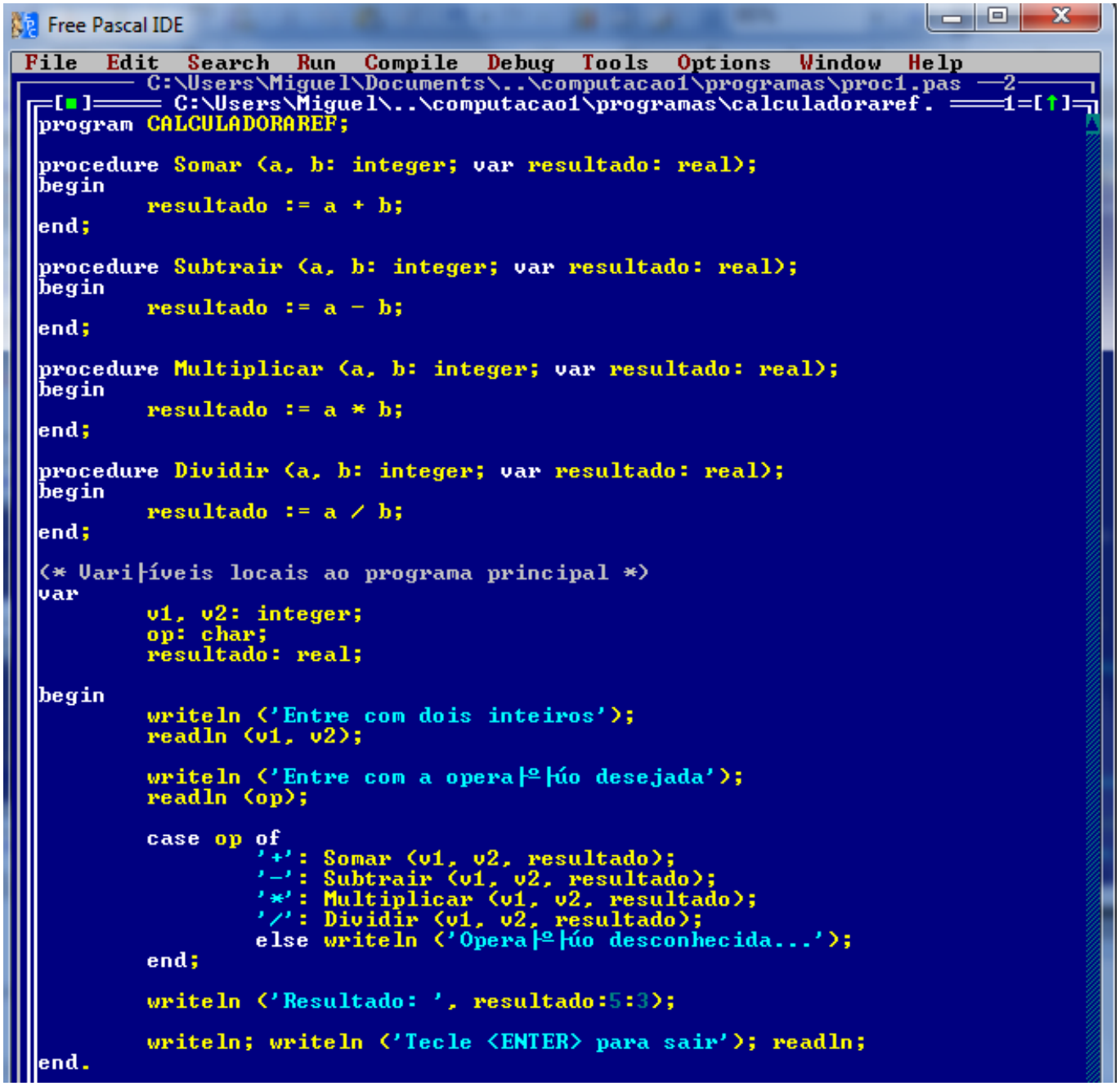
Vigésimo Primeiro Exemplo




Free Pascal IDE
File Edit Search Run Compile Debug Tools Options Window Help
C:\Users\Miguel\Documents\..\computacao1\programas\proc1.pas 2
C:\Users\Miguel\..\computacao1\programas\calculadoraref. 1
program CALCULADORAPROC;
procedure Somar (a, b: integer);
begin
resultado := a + b;
end;
procedure Subtrair (a, b: integer);
begin
resultado := a - b;
end;
procedure Multiplicar (a, b: integer);
begin
resultado := a * b;
end;
procedure Dividir (a, b: integer);
begin
resultado := a / b;
end;
(* Vari├íveis locais ao programa principal *)
var
v1, v2: integer;
op: char;
begin
writeln ('Entre com dois inteiros');
readln (v1, v2);
writeln ('Entre com a opera├º├úo desejada');
readln (op);
case op of
'+': Somar (v1, v2);
'-': Subtrair (v1, v2);
'*': Multiplicar (v1, v2);
'/': Dividir (v1, v2);
else writeln ('Opera├º├úo desconhecida...');
end;
writeln ('Resultado: ', resultado:5:3);
writeln; writeln ('Tecle <ENTER> para sair'); readln;
end.

Passagem Parâmetro por Valor X por Referência

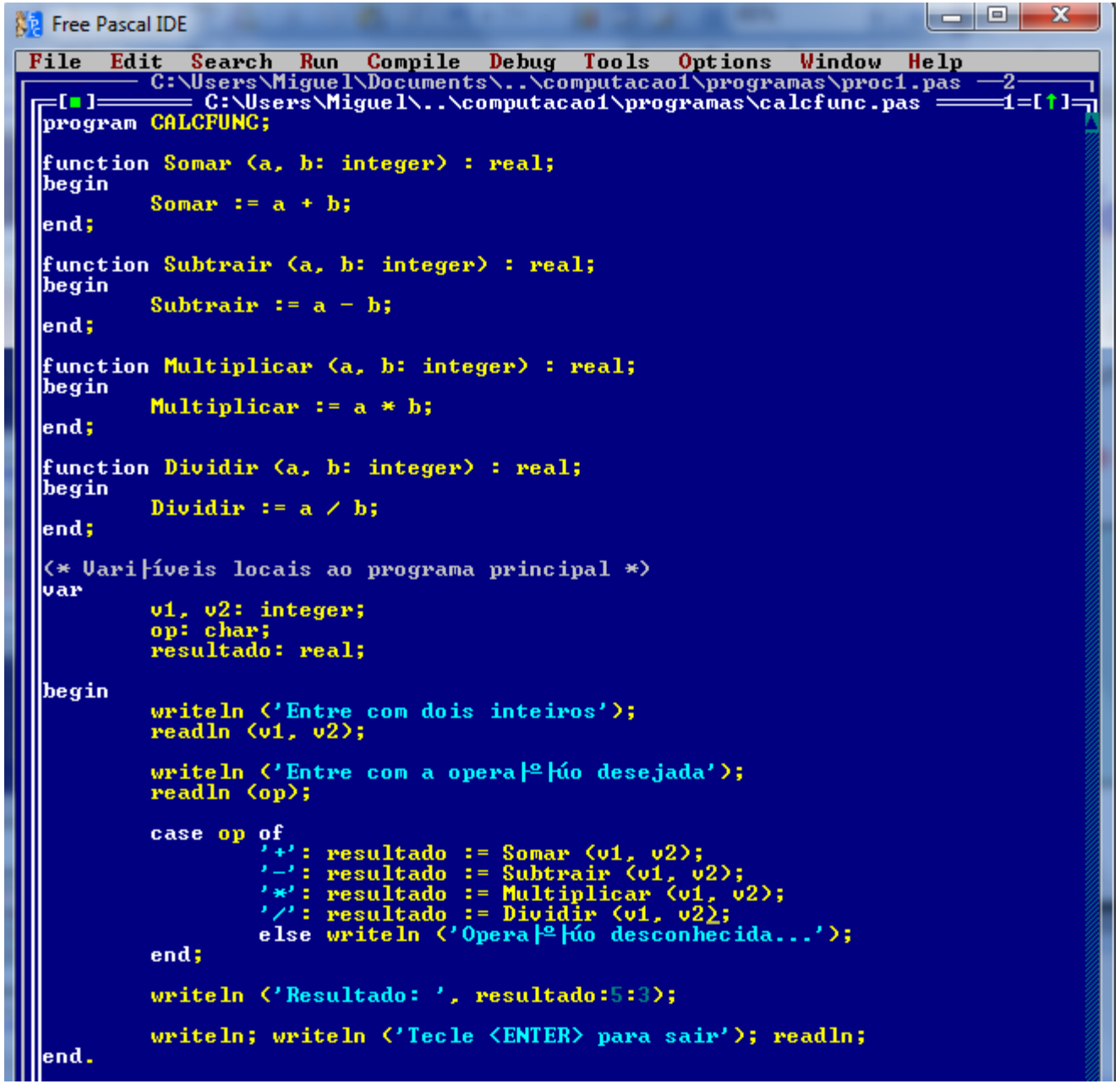
- Passagem de parâmetro por valor
  - Valor da variável é passada para função e é usada para inicializar uma variável local definida como um argumento da função
    - Após o término da execução da função, a variável é desalocada e o valor é perdido
- Passagem de parâmetro por referência
  - Endereço da variável é passada para função e é usada como referência para a posição da variável em memória
    - Após o término da execução da função, a variável é é alterada


Free Pascal IDE
File Edit Search Run Compile Debug Tools Options Window Help
C:\Users\Miguel\Documents\..\computacao1\programas\proc1.pas 2
C:\Users\Miguel\..\computacao1\programas\calculadoraref. 1
program CALCULADORAREF;
procedure Somar (a, b: integer; var resultado: real);
begin
resultado := a + b;
end;
procedure Subtrair (a, b: integer; var resultado: real);
begin
resultado := a - b;
end;
procedure Multiplicar (a, b: integer; var resultado: real);
begin
resultado := a * b;
end;
procedure Dividir (a, b: integer; var resultado: real);
begin
resultado := a / b;
end;
(* Vari├íveis locais ao programa principal *)
var
v1, v2: integer;
op: char;
resultado: real;
begin
writeln ('Entre com dois inteiros');
readln (v1, v2);
writeln ('Entre com a opera├º├úo desejada');
readln (op);
case op of
'+': Somar (v1, v2, resultado);
'-': Subtrair (v1, v2, resultado);
'*': Multiplicar (v1, v2, resultado);
'/': Dividir (v1, v2, resultado);
else writeln ('Opera├º├úo desconhecida...');
end;
writeln ('Resultado: ', resultado:5:3);
writeln; writeln ('Tecle <ENTER> para sair'); readln;
end.

Function

- Cumpre papel semelhante aos das procedures
  - Entretanto, retorna sempre um valor de um tipo pré-determinado
    - Valor de retorno é retornado no próprio nome da function

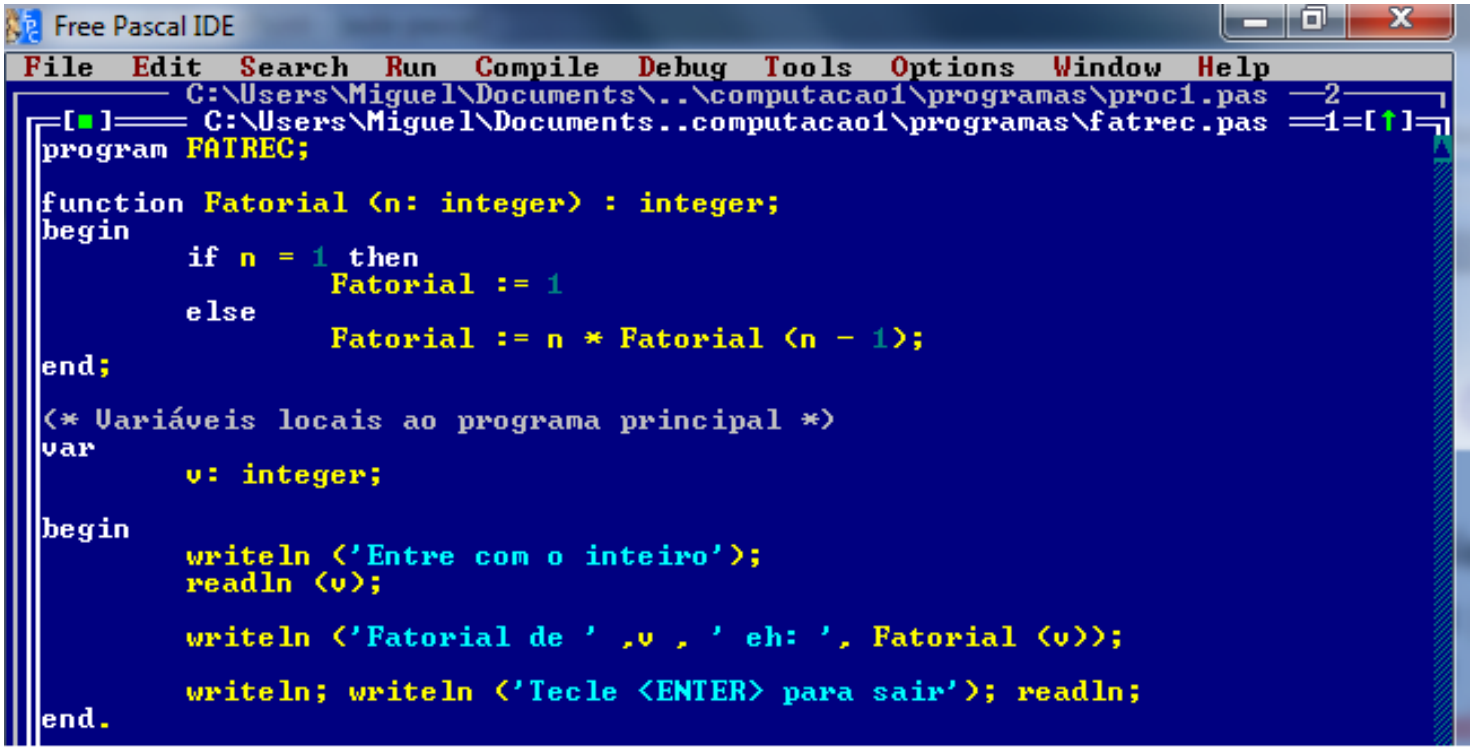
```
function NOME (<variáveis>) : <tipo_var_retorno>;
var
      var_local1: <tipo1>
      var_local2: <tipo2>
begin
      código...
end;
```


Free Pascal IDE
File Edit Search Run Compile Debug Tools Options Window Help
C:\Users\Miguel\Documents\..\computacao1\programas\proc1.pas 2
C:\Users\Miguel\..\computacao1\programas\calcfunc.pas 1
program CALCFUNC;
function Somar (a, b: integer) : real;
begin
Somar := a + b;
end;
function Subtrair (a, b: integer) : real;
begin
Subtrair := a - b;
end;
function Multiplicar (a, b: integer) : real;
begin
Multiplicar := a * b;
end;
function Dividir (a, b: integer) : real;
begin
Dividir := a / b;
end;
(* Vari├íveis locais ao programa principal *)
var
v1, v2: integer;
op: char;
resultado: real;
begin
writeln ('Entre com dois inteiros');
readln (v1, v2);
writeln ('Entre com a opera├º├úo desejada');
readln (op);
case op of
'+': resultado := Somar (v1, v2);
'-': resultado := Subtrair (v1, v2);
'*': resultado := Multiplicar (v1, v2);
'/': resultado := Dividir (v1, v2);
else writeln ('Opera├º├úo desconhecida...');
end;
writeln ('Resultado: ', resultado:5:3);
writeln; writeln ('Tecle <ENTER> para sair'); readln;
end.

Recursividade

- Uma função pode chamar a mesma função para um problema reduzido
  - As chamadas são realizadas até que o problema seja mínimo
    - Caso base

Vigésimo Quinto Exemplo



Units

- Arquivo em Pascal (*.pas) utilizado para pré-programar procedimentos e funções
  - Biblioteca
    - Muitas já são padrão do Pascal (Ex.: crt)
    - Outras podem ser criadas pelo programador
  - Sintaxe
    - **unit**
      - Define o nome da unit e deve ser o mesmo nome do arquivo
    - **Interface**
      - Define a interface dos procedimentos e funções
    - **Implementation**
      - Define a implementação de cada um dos procedimentos e funções da interface

exemplo.pas

```
unit exemplo;

interface
      function NOMEFUNC (<variáveis>):<tipo_var_retorno>;
      procedure NOMEPROC (<variáveis>);

implementation
      function NOMEFUNC (<variáveis>):<tipo_var_retorno>;
      var
                        ...
      begin
                        ...
      end;
      procedure NOMEPROC (<variáveis>);
      var
                        ...
      begin
                        ...
      end;
end.
```

Units

usaUnit.pas

```
program USAUNITS;

uses exemplo;

var
                              ...
begin
                              ...
end.
```

Exemplo de Uso de Unit

```
unit fatUnit;

interface

function Fatorial (n: integer) : integer;

implementation

function Fatorial (n: integer) : integer;
begin
        if n = 1 then
                {begin}
                Fatorial := 1
                {end}
        else
                {begin}
                Fatorial := n * Fatorial (n - 1);
                {end;}
end;

end.
```

Exemplo de Uso de Unit

```
program FATREC;

uses fatUnit;

(* Variáveis locais ao programa principal *)
var
        v: integer;

begin
        writeln ('Entre com o inteiro');
        readln (v);

        writeln ('Fatorial de ' ,v , ' eh: ', Fatorial (v));

        writeln; writeln ('Tecle <ENTER> para sair'); readln;
end.
```

Arquivos

- Permite escrever e ler dados da memória secundária
  - Operações principais
    - Assign (<variável>, <arquivo>)
      - Associa o nome lógico de um arquivo ao arquivo físico, o parâmetro <variável> é a indicação da variável do tipo arquivo e <arquivo> é o nome do arquivo a ser manipulado
    - Rewrite (<variável>)
      - Cria um arquivo para uso, utilizando o nome associado ao parâmetro <variável>. Caso o arquivo já exista, esta instrução o apaga para criá-lo novamente
    - Reset (<variável>)
      - Abre um arquivo existente, colocando-o disponível para leitura e escrita, utilizando o nome associado ao parâmetro <variável>.

Arquivos

- Permite escrever e ler dados da memória secundária
  - Operações principais
    - Write (<variável>, <dado>)
      - Escreve a informação <dado> no arquivo indicado
    - Read (<variável>, <dado>)
      - Lê a informação <dado> no arquivo indicado pela <variável>
    - Close (<variável>)
      - Fecha um arquivo em uso dentro de um programa. Nenhum programa deve ser encerrado sem antes fechar os arquivos abertos

Arquivos de Texto

- Cria-se variável do tipo **text**

```
program ARQUIVO;
var
        arquivo: text
```

- A variável é, então, associada a um nome de arquivo

```
program ARQUIVO;
var
        arquivo: text
begin
        assign (arquivo, 'arquivo.txt');
        <sentenças>...
        close (arquivo);
end;
```

- Depois o programa é escrito manipulando a variável...

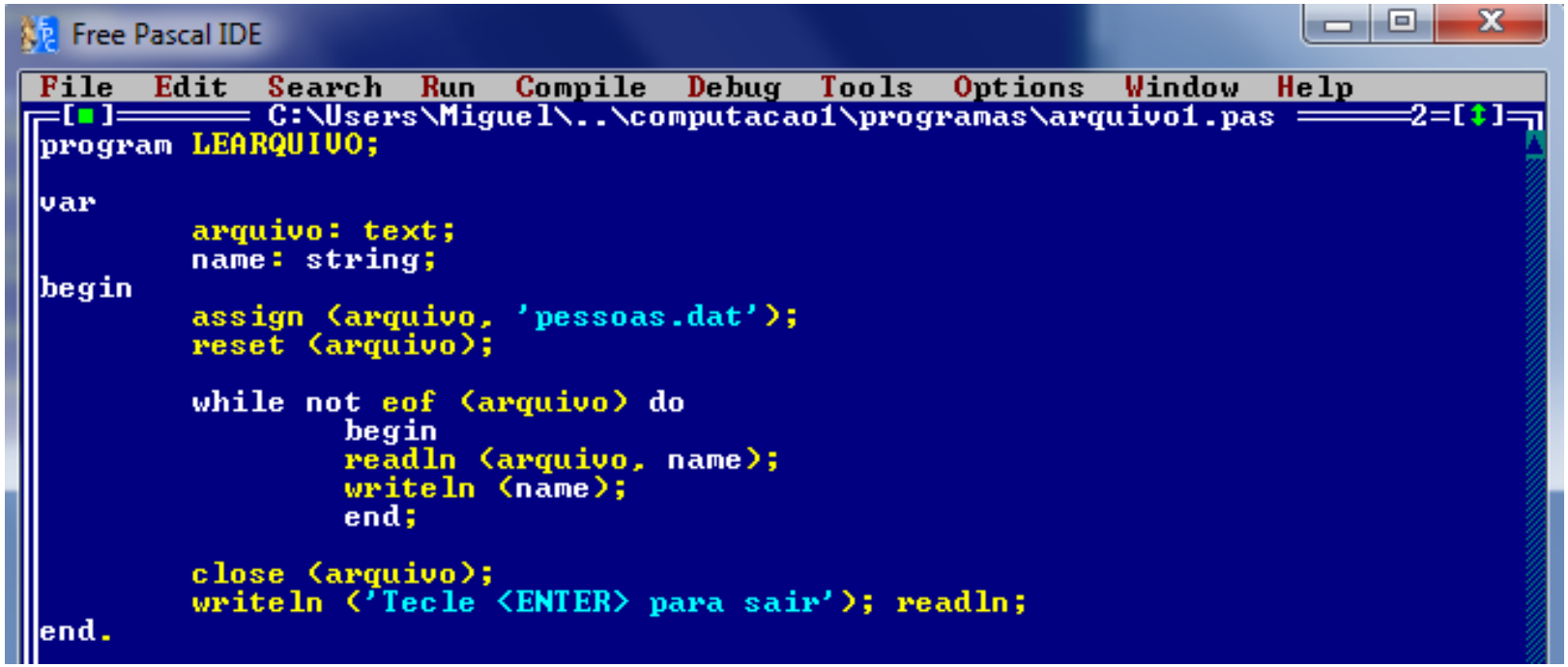
Arquivos de Texto

Exemplo de Escrita

C:\Users\Miguel\..\computacao1\programas\arqsimp.pas

```
program ESCREVEARQUIVO;

var
        arquivo : text;
        nome: string;
        continua: char;

begin
        assign (arquivo, 'pessoas.txt');
        rewrite (arquivo);

        repeat
                write ('Digite o Nome: ');
                readln (nome);

                writeln (arquivo, nome);

                writeln ('Deseja continuar (s/n)? ');
                readln (continua);

        until (continua = 'N') or (continua = 'n');

        close (arquivo);
end.
```

Exemplo de Leitura



Exemplo de Leitura com Registro

Free Pascal IDE

File Edit Search Run Compile Debug Tools Options Window Help

C:\Users\Miguel\Documents..computacao1\programas\arquivo.pas

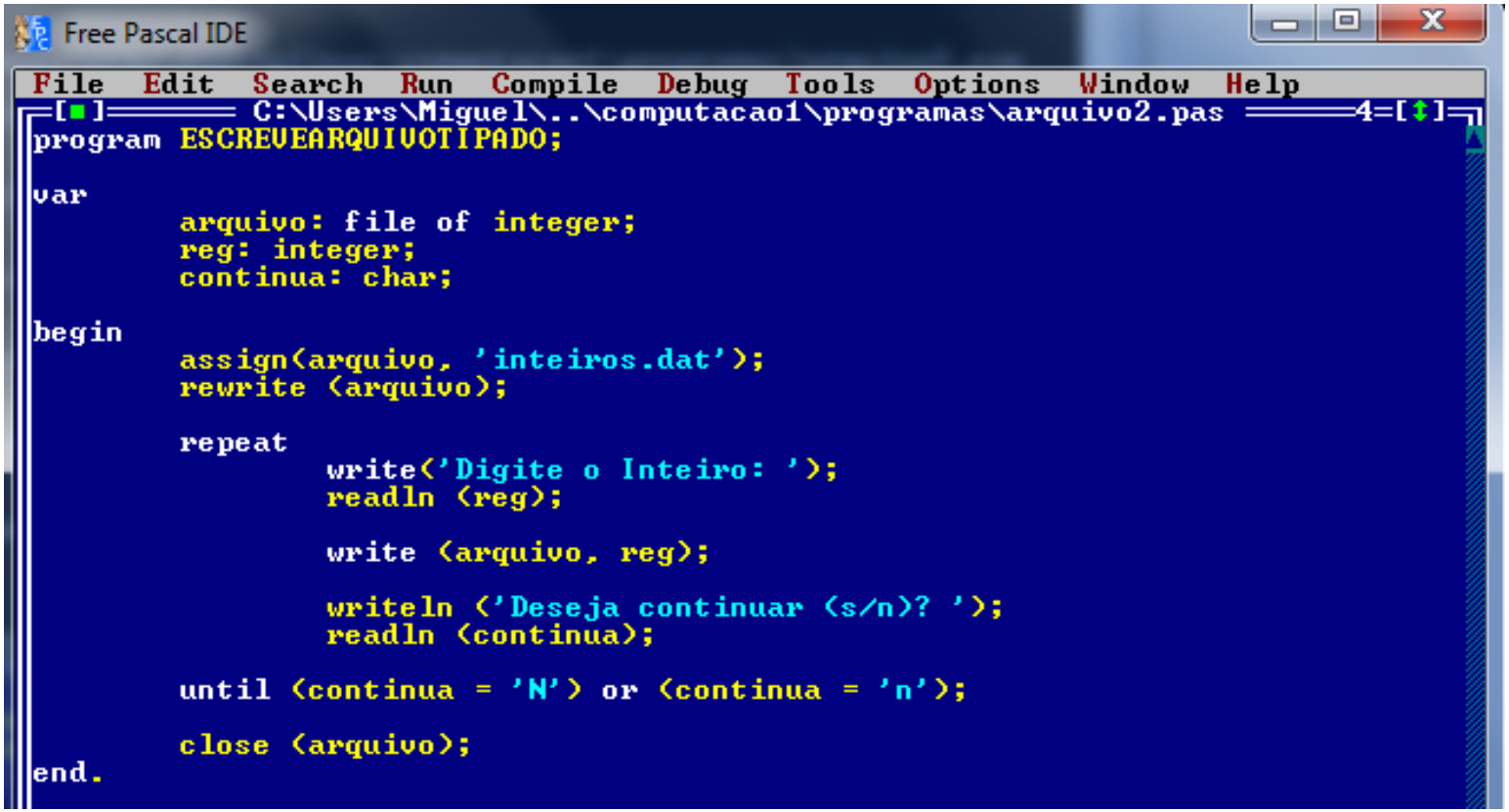
```
program ESCREVEARQUIVO;

type
        registro = record
                nome: string;
                idade: integer;
        end;

var
        arquivo : text;
        reg: registro;
        continua: char;

begin
        assign (arquivo, 'pessoas.txt');
        rewrite (arquivo);

        repeat
                write ('Digite o Nome: ');
                readln (reg.nome);
                write ('Digite o Idade: ');
                readln (reg.idade);

                write (arquivo, reg.nome);
                write (arquivo, ' ');
                writeln (arquivo, reg.idade);

                writeln ('Deseja continuar (s/n)? ');
                readln (continua);

        until (continua = 'N') or (continua = 'n');

        close(arquivo);
end.
```

Arquivos com Tipo Definido

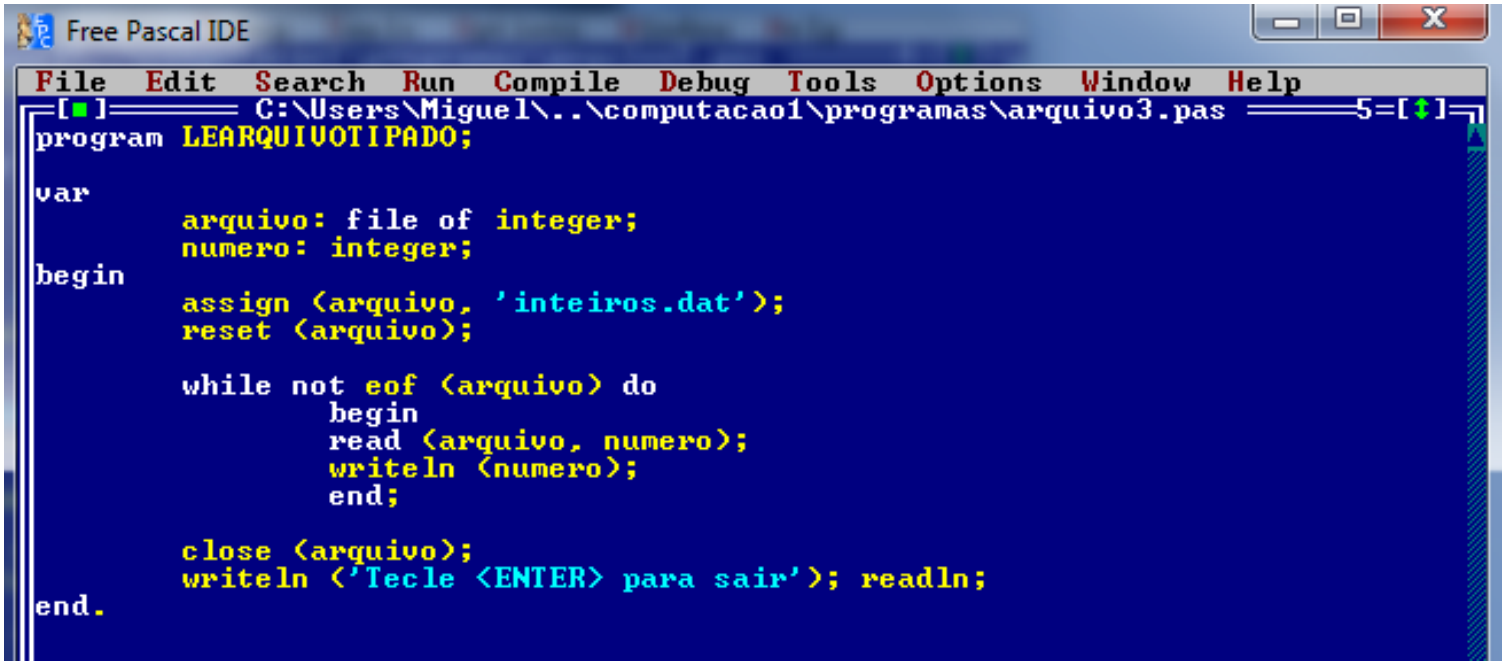
- Arquivos denominados em Pascal como arquivos tipados
  - Arquivos do tipo **binário**, diferentes dos arquivos de texto
    - Operações de leitura e escrita são mais rápidas

```
program ARQUIVOTIPADO;
var
      arquivo: file of integer;
begin
      assign (arquivo, 'arquivo.bin');
      <sentenças>...
      close (arquivo);
end;
```

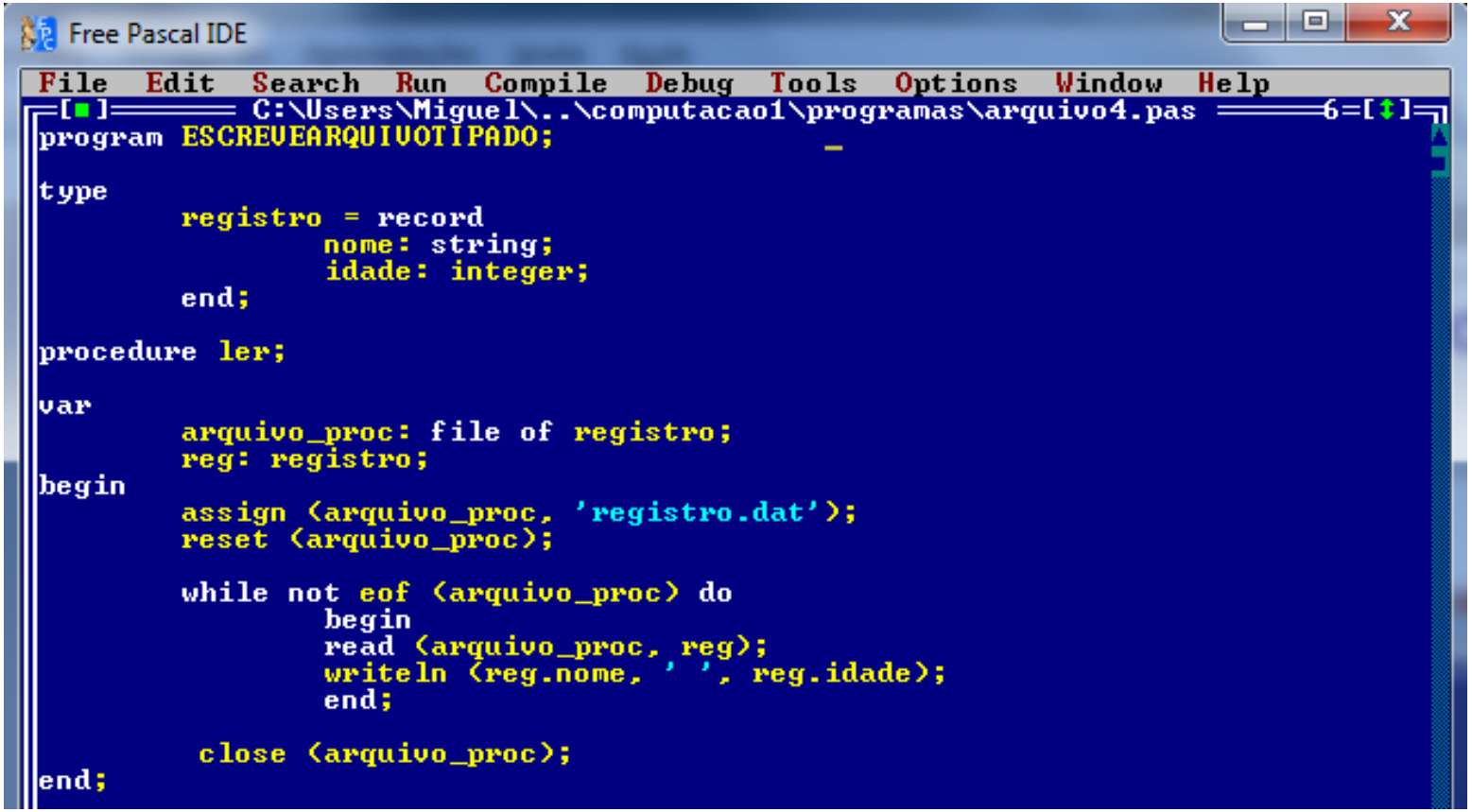
Exemplo de Escrita com Tipo Definido



Exemplo de Leitura com Tipo Definido



Exemplo de Escrita e Leitura com Tipo Definido



Exemplo de Escrita e Leitura com Tipo Definido

```
var
        arquivo : file of registro;
        reg: registro;
        continua: char;

begin
        assign(arquivo, 'registro.dat');
        {$I-}
        reset (arquivo);
        {$I+}
        if (IORESULT = 2) then
                begin
                rewrite (arquivo);
                write (arquivo, reg);
                seek (arquivo, filepos (arquivo) - 1);
                end
        else
                seek (arquivo, filesize (arquivo));

        repeat
                write ('Digite o Nome: ');
                readln (reg.nome);
                write ('Digite o Idade: ');
                readln (reg.idade);

                write (arquivo, reg);

                writeln ('Deseja continuar (s/n)? ');
                readln (continua);

        until (continua = 'N') or (continua = 'n');

        close (arquivo);

        ler;

end.
```

Ponteiros

Exemplo de Ponteiro

```
program POINTER1;

procedure enviaInteiro (var a: integer);
var
        x: ^integer;
begin
        new (x);
        x^ := a;
        writeln ('x: ', x^);
        dispose (x);
end;

var
        ptrNome: ^string;
        ptrInt: ^integer;

begin

        writeln  ('Escreva um nome: ');
        new (ptrNome);
        ptrNome^ := 'Miguel';
        writeln (ptrNome^);
        dispose (ptrNome);

        writeln ('Escreva um inteiro: ');
        GetMem (ptrInt, 4);
        readln (ptrInt^);
        writeln (ptrInt^);
        enviaInteiro (ptrInt^);
        freeMem (ptrInt);
        write ('Tecle ENTER para terminar'); readln;

end.
```

Exemplo de Ponteiro

```
program POINTER2;

uses
        Crt;

type
        ptrMat = ^matriz;
        matriz = array [1..10] of integer;

var
        a: ptrMat;
        i, j, n, x: integer;

begin
        clrscr;
        writeln ('Informe o numero de elementos:');
        readln (n);
        writeln;
        GetMem (a, n * SizeOf (integer));
        for i := 1 to n do
                begin
                write ('Entre o ', i, 'o. elemento: ');
                readln (a^[i]);
                end;
        for i := n - 1 downto 1 do
                for j := 1 to i do
                        if a^[j] > a^[j + 1] then
                                begin
                                x := a^[j];
                                a^[j] := a^[j + 1];
                                a^[j + 1] := x;
                                end;
        clrscr;
        writeln ('Classificacao de ', n, ' elementos');
        writeln;
        for i := 1 to n do
                writeln (a^[i]);
        writeln;
        FreeMem (a, n * SizeOf (integer));
        writeln ('Tecle ENTER para terminar'); readkey;
end.
```

Exemplo de Ponteiro

```
program LISTAPOINTER1;

uses
        Crt;

type
        lista = ^tabela;
        tabela = record
                elemento: real;
                prox: lista;
                end;

var
        a, primeiro_elem, segundo_elem, lista_saida: lista;
        i: longint;
        soma, media: real;
        n: string;
        codigo: integer;

begin
        soma := 0;
        i := 1;

        clrscr;
        writeln ('Matriz Dinamica');
        writeln;

        primeiro_elem := nil;

        repeat
                if primeiro_elem = nil then
                        begin
                        new (a);
                        write ('Informe o ', i, 'o. valor: ');
                        readln (n);
                        if n <> '*' then
                                begin
                                val (n, a^.elemento, codigo);
                                primeiro_elem := a;
                                a^.prox := nil;
                                i := i + 1;
                                end;
                        end
```

Exemplo de Ponteiro

```
                else
                        begin
                        segundo_elem := a;
                        new (a);
                        write ('Informe o ', i, 'o. valor: ');
                        readln (n);
                        if n <> '*' then
                                begin
                                val (n, a^.elemento, codigo);
                                segundo_elem^.prox := a;
                                a^.prox := nil;
                                i := i + 1;
                                end;
                        end;
        until n = '*';
        lista_saida := primeiro_elem;
        writeln;
        writeln ('A lista eh: ');
        while lista_saida <> nil do
                begin
                writeln (lista_saida^.elemento:6:2);
                soma := soma + lista_saida^.elemento;
                lista_saida := lista_saida^.prox;
                end;
        media := soma/(i - 1);
        writeln;
        writeln ('Media = ', media:6:2);
        writeln;
        write ('Tecle qualquer tecla para encerrar.');
        readkey;
end.
```

Exemplo de Ponteiro

```
program LISTAPOINTER2;

uses
        Crt;

type
        lista = ^dados;
        dados = record
                nome: string;
                salario: real;
                tempo: integer;
                prox: lista;
                end;

var
        cadfunc, atual, list_cf: lista;
        ent_nome: string;
        ent_salario: real;
        ent_tempo: integer;
        i, linha: longint;
        resp: char;

begin
        i := 1;
        cadfunc := nil;

        repeat
                clrscr;
                writeln ('cadfunc');
                writeln;
                new (cadfunc);
                writeln ('Entre o ', i, 'o. registro');
                writeln;
                write ('Nome.............: '); readln (ent_nome);
                write ('Salario..........: '); readln (ent_salario);
                write ('Tempo de Servico...: '); readln (ent_tempo);
                cadfunc^.nome := ent_nome;
                cadfunc^.salario := ent_salario;
                cadfunc^.tempo := ent_tempo;

                if (list_cf = nil) or (ent_nome < list_cf^.nome) then
                        begin
                        cadfunc^.prox := list_cf;
                        list_cf := cadfunc;
                        end
```

Exemplo de Ponteiro

```
                else
                        begin
                        atual := list_cf;
                        while (atual^.prox <> nil) and
                                (ent_nome > atual^.prox^.nome) do
                                atual := atual^.prox;
                        cadfunc^.prox := atual^.prox;
                        atual^.prox := cadfunc;
                        end;
                i := i + 1;
                writeln;
                writeln ('Deseja continuar [S]im ou [N]ao: ');
                readln (resp);
        until upcase(resp) = 'N';
        clrscr;
        i := 1;
        linha := 3;
        gotoxy ( 1, 1); write ('Reg#');
        gotoxy ( 6, 1); write ('Nome');
        gotoxy (46, 1); write ('Salario');
        gotoxy (61, 1); write ('Tempo de Servico');
        while list_cf <> nil do
                begin
                gotoxy ( 1, linha); write (i:4);
                gotoxy ( 6, linha); write (list_cf^.nome);
                gotoxy (46, linha); write (list_cf^.salario:6:2);
                gotoxy (61, linha); write (list_cf^.tempo);
                list_cf := list_cf^.prox;
                linha := linha + 1;
                i := i + 1;
        end;
        writeln; writeln;
        writeln ('Tecle qualquer coisa para encerrar'); readkey;
end.
```