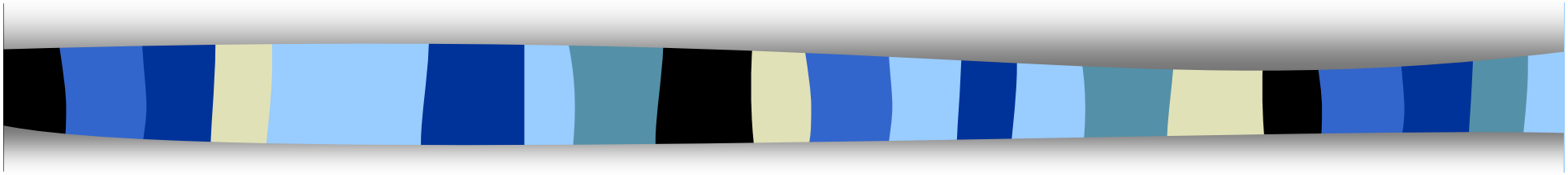
Universidade Federal do Amazonas
Faculdade de Tecnologia
Departamento de Eletrônica e Telecomunicações

Programando em Ada95



Lucas Cordeiro
lucascordeiro@ufam.edu.br

Instalação do compilador Ada95

- Verifique a versão do seu compilador gcc:

```
$gcc --version
```

- Baixe o pacote do gcc-gnat-`versão` do site: http://rpm.pbone.net/

- Utilize o comando:

```
$rpm -ivh nome-do-pacote ou
$yum -y install nome-do-pacote ou
$apt-get install nome-do-pacote
```

Hello World

- Considere o seguinte programa `hello.adb` escrito em Ada

```
with Ada.Text_IO; use Ada.Text_IO;
procedure Hello is
begin
  Put_Line("Hello World!");
end Hello;
```

- Use o seguinte comando para compilar este programa:

`$gcc -c hello.adb` (cria os arquivos `hello.ali` - *Ada Library Information* - e `hello.o` - *Object File*)

`$gnatbind hello` (cria os arquivos `b~hello.adb` e `b~hello.ads`)

`$gnatlink hello` (cria o executável `hello`)

ou simplesmente:

`$gnatmake hello.adb` (cria o executável `hello`)

Minimum (1)

```
-- Ada
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
procedure Minimum is
--numberX, NumberY : String(1..2);
IntegerX, IntegerY, Result : Integer;
function compute_minimum(X, Y : in Integer) return
Integer is
begin
  if X > Y then
    return Y;
  else
    return X;
  end if;
end compute_minimum;
```

Minimum (2)

```
begin
   Put_Line("Digite o valor de X: ");
--   Get(numberX);
   Get(IntegerX);
   Put_Line("Digite o valor de Y: ");
--   Get(numberY);
   Get(IntegerY);

--   IntegerX := Integer'Value(numberX);
--   IntegerY := Integer'Value(numberY);

   Result := compute_minimum(IntegerX, IntegerY);
   Put("O menor número é:");
   Put(Result);

end Minimum;
```

Quadratic (1)

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Float_Text_IO;
with Ada.Numerics.Generic_Elementary_Functions;
procedure Quadratic is
CoefA, CoefB, CoefC : Float;
Root1, Root2 : Float;
Result : Boolean;
procedure compute_quadratic (A, B, C : in Float;
                             R1, R2 : out Float;
                             Ok : out Boolean) is
  package Float_Functions is new
   Ada.Numerics.Generic_Elementary_Functions (Float);
   Z : Float;
   begin
     Put("Calculando as raizes...");
     New_Line(1);
     Z := B*B - 4.0*A*C;
     if Z < 0.0 or A = 0.0 then
       Ok := False;
       R1 := 0.0;          --valor arbitrário
```

Quadratic (2)

```
      R1 := 0.0;          --valor arbitrário
      R2 := 0.0;
     return;          --retorna de um procedimento antes de
                      --alcançar o end
   end if;
   Ok := True;
     R1 := (-B + Float_Functions.Sqrt(Z)) / (2.0*A);
     R2 := (-B - Float_Functions.Sqrt(Z)) / (2.0*A);
   end compute_quadratic;

begin
     -- 3x²-7x+2=0 --> {1/3, 2}
     Put_Line("Calcula as raizes de 3x²-7x+2=0");
     CoefA := 3.0;
     CoefB := -7.0;
     CoefC := 2.0;
     compute_quadratic(CoefA,CoefB,CoefC,Root1,Root2,Result);
     Put_Line("Raizes:");
     Ada.Float_Text_IO.Put(Root1);
     Ada.Float_Text_IO.Put(Root2);
```

Quadratic (3)

```
   New_Line;
   --  -x²+4x-4=0 --> {2}
   Put_Line("Calcula as raizes de  -x²+4x-4=0");
   CoefA := -1.0;
   CoefB := 4.0;
   CoefC := -4.0;
   compute_quadratic(CoefA,CoefB,CoefC,Root1,Root2,Result);
   Put_Line("Raizes:");
   Ada.Float_Text_IO.Put(Root1);
   Put_Line("");
   Ada.Float_Text_IO.Put(Root2);

end Quadratic;
```

Inverte String

```
-------------------------------------------------------
-- PTR, Inverte String
-------------------------------------------------------

with Ada.Text_IO; use Ada.Text_IO;
procedure inverte_string is
  function Inverte(Item : String) return String is
    resultado : String(Item'Range);
  begin
    for i in Item'Range loop
      resultado(resultado'Last + Item'First - i) := Item(i);
    end loop;
  return resultado;
end Inverte;
begin
  Put_Line("Escreva um texto para inverter: ");
  Put_Line(Inverte(Get_Line));
end inverte_string;
```

Exercícios

- Escreva um programa em Ada que capture o valor do raio de uma esfera e exiba na tela o valor do volume e da área da superfície da esfera. Sabe-se que o volume da esfera é dado por $(4/3)\times\pi\times r^3$ e que a área da superfície é dada por $4\times\pi\times r^2$. Assuma $\pi=3.14159$
- Escreva um programa em Ada que leia dois vetores A, de tamanho *m* e B, de tamanho *n*, calcule o vetor C, soma de A e B
- Lidos dois strings A e B, verifique se o string B está contido em A
- Escreva um programa em Ada que leia um *string* e escreva quantas e quais são as vogais

Exercícios

- Escreva um programa para calcular a seqüência de Fibonacci dos termos inferiores a um número inteiro L qualquer usando Ada e C
- Escreva um programa em Ada que leia uma matriz quadrada e calcule a somatória dos elementos da diagonal principal

Soma Dois Vetores (1)

```
with Ada.Text_IO, Ada.Integer_Text_IO;
use Ada.Text_IO, Ada.Integer_Text_IO;
  procedure soma_vetor is
  SIZE : Constant Integer := 5;
  type VectorData is array(0 .. SIZE-1) of Integer;
  VetA, VetB, VetC : VectorData;
  function SomaVetores(vetorA, vetorB : in VectorData) return
   VectorData is
  vetorC : VectorData;
  begin
    for i in vetorA'Range loop
      vetorC(i) := vetorA(i) + vetorB(i);
    end loop;
    return vetorC;
  end SomaVetores;
```

Soma Dois Vetores (2)

```
procedure CarregaVetor(vetor : out VectorData) is
 begin
   for i in vetor'Range loop
     Put_Line("Digite o valor do proximo termo: ");
       Ada.Integer_Text_IO.Get(vetor(i));
    end loop;
    New_Line;
 end CarregaVetor;

procedure ImprimeVetor(vetor : in VectorData) is
  begin
      for i in vetor'Range loop
         Ada.Integer_Text_IO.Put(vetor(i));
      end loop;
      New_Line;
   end ImprimeVetor;
```

Soma Dois Vetores (3)

```
begin
   New_Line;
   Put_Line("CARREGA VETOR A:");
   CarregaVetor(VetA);
   ImprimeVetor(VetA);

   New_Line;
   Put_Line("CARREGA VETOR B:");
   CarregaVetor(VetB);
   ImprimeVetor(VetB);

   New_Line;
   Put_Line("SOMA DE VETORES:");
   VetC := SomaVetores(VetA,VetB);
   ImprimeVetor(VetC);
end soma_vetor;
```