16.070

Introduction to Computers & Programming

Ada II

Introduction to Straight-line programs

Prof. I. Kristina Lundqvist
Dept. of Aero/Astro, MIT

```
WITH package1;
WITH package2;
…
WITH package3;
PROCEDURE general_form IS

   declarations (variables, constants, etc.)

BEGIN --general_form

   program statement;
   …
   program statement;

END general_form;
```

```
WITH Ada.Text_IO;
WITH Ada.Integer_Text_IO;
PROCEDURE Distance IS
--------------------------
--| Finds distance travele
--| Author: Michael B. Fel
--| Last Modified: June 19
--------------------------
  HowLong : Natural;
  HowFast : Natural;
  HowFar  : Natural;

BEGIN -- Distance

  -- prompt user for hours and average speed
  Ada.Text_IO.Put
     (Item => "How many hours will you be driving (integer) ? ");
  Ada.Integer_Text_IO.Get (Item => HowLong); Ada.Text_IO.Skip_Line;
  Ada.Text_IO.Put
     (Item=>"At what average speed (miles per hour, integer) ? ");
  Ada.Integer_Text_IO.Get (Item => HowFast); Ada.Text_IO.Skip_Line;
```

```
-- compute distance driven
  HowFar := HowFast * HowLong;

  -- display results
  Ada.Text_IO.Put (Item => "You will travel about ");
  Ada.Integer_Text_IO.Put (Item => HowFar);
  Ada.Text_IO.Put (Item => " miles");
  Ada.Text_IO.New_Line;

END Distance;
```

Constants

- Data values that do not change their value
  - `CMPerInch: CONSTANT Float := 2.54;`
  - `Pi : CONSTANT Float := 3.1415926536;`
  - `Answer : CONSTANT Integer := 42;`
  - `MyName : CONSTANT String := “kristina”;`
  - `Name : CONSTANT Type := value; --Form`
- Benefits of using constants:
  - Easier to read
  - Only one line has to be modified
  - Ada statement that tries to change value of constant → compilation error

Variables

- Storing input data and computational results
  - `HowLong : Natural;`
  - `Initial1, Initial2 : Character;`
  - `Inches : Float;`
  - `Name : DataType;  --Form`
- Main pre-declared data types in Ada: integer, float, character, string, and boolean

- Integer (Natural, Positive)
  - positive or negative number with no decimal part
    - 123 -456 +789
    - ```
      Put ("The lowest integer value is: ");
      Put (Iteger'First); New_line;
      Put ("The highest integer value is: ");
      Put (Integer'Last); New_line;
      ```
- Float
  - Positive or negative number with decimal part
    - 123.4456 1.234e-4
- Character
  - Single character: Y/N answer
- String
  - Represent a sequence of characters as a single unit of data
- Boolean
  - Logical test
      ```

Strong Typing

- Cannot mix data types
  - 1 + 6
  - 4.2 / 7.0
  - 16 > 0.70
  - 12 * 3.0
- Mixed arithmetic
  - Float (16) > 0.70
  - 12 * Integer (3.0)

```
procedure Roundoff is   -- demonstrate rounding errors
   Zero                : constant       := 0.0;
   One                 : constant Float := 1.0;
   Ten_Thousand        : constant       := 10000;
   One_Ten_Thousandth : constant Float := One / Float (Ten_Thousand);
   The_Result          :          Float := Zero;

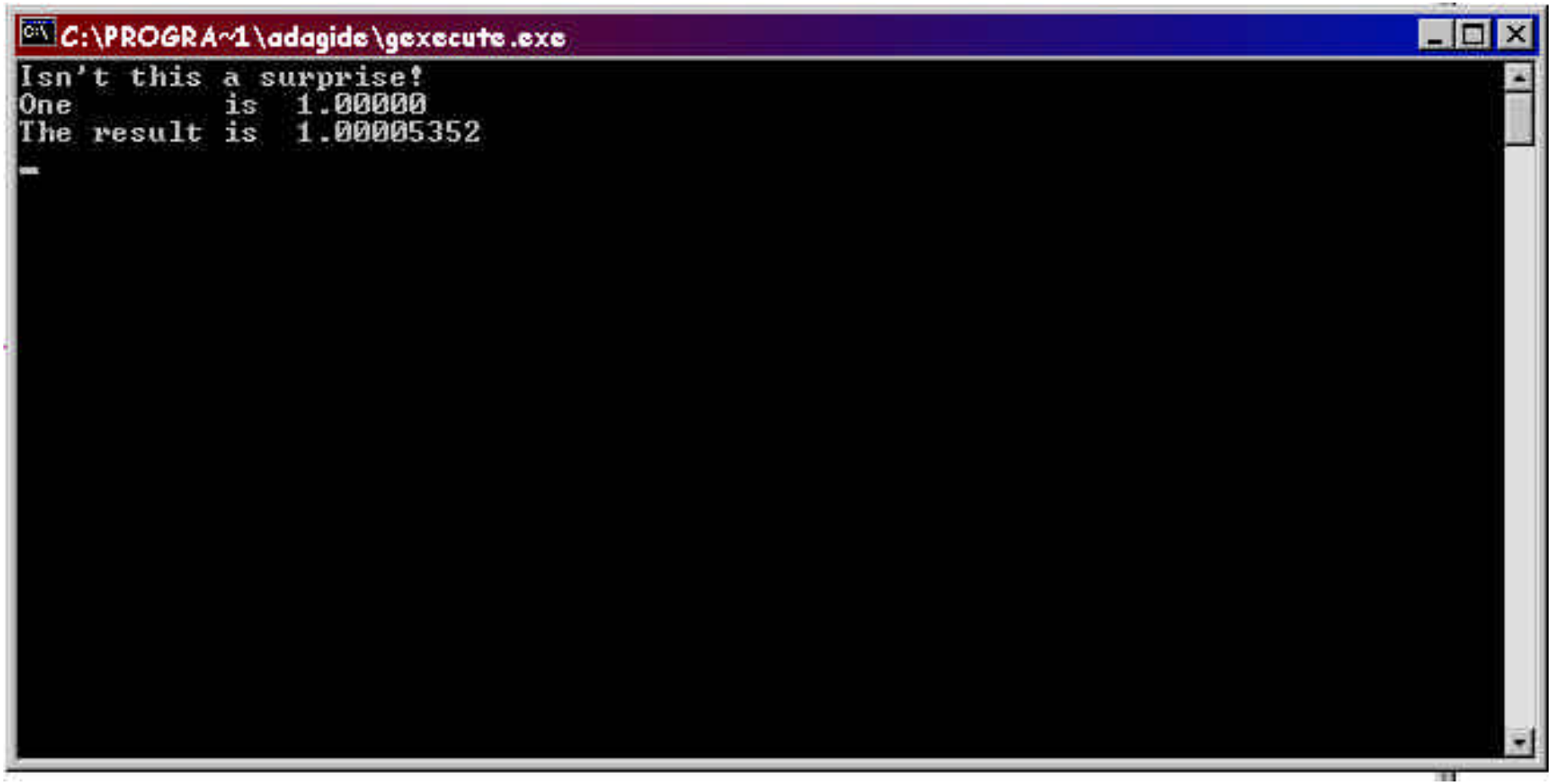
begin -- roundoff

   -- the_result is initially equal to zero

   for Counter in Integer range 1 .. Ten_Thousand loop
      The_Result := The_Result + One_Ten_Thousandth;
   end loop;

   -- mathematically, the_result is now equal to
   -- 1/10_000 * 10_000 = 1.0

   if (The_Result = One) then
      Put_Line("Isn't this what you expected!");
   else
      Put_Line("Isn't this a surprise!");
      Put("One        is "); Put(One, Exp => 0); New_Line;
      Put("The result is "); Put(The_Result, Aft => 8, Exp=>0);
      New_Line;
   end if;
end Roundoff;
```


C:\PROGRA~1\adagide\gexecute.exe
Isn't this a surprise!
One is 1.00000
The result is 1.00005352

Identifiers

1. P1
2. This_will_always_be_first_item_in_list
3. Car_1
4. First_car
5. i
6. N
7. Lecture_1
8. Lecture_2

Identifiers should be as meaningful as possible, without being verbose

Assignment

- Used to perform computations and save result in a variable
  - SquareYards := MetersToYards * SquareMeters;
  - NewX := X;
  - NewX := -X;
  - Sum := Sum + Item;

Input/Output Statements

- Get Procedure (Character)
  - `Ada.Text_IO.Get (Item => Initial);`
- Get Procedure (String)
  - `Ada.Text_IO.Get (Item => First_Name);`
- Get Procedure (Integer)
  - `Ada.Integer_Text_IO.Get (Item => How_Long);`
- Get Procedure (Floating Point)
  - `Ada.Float_Text_IO.Get (Item => Inches);`

- When prompting for values from a user, always follow `Get` with `Skip_Line`

Input/Output Statements

- Put Procedure (Character)
  - `Ada.Text_IO.Put (Item => Initial);`
- Put Procedure (String)
  - `Ada.Text_IO.Put (Item => First_Name);`
- New_Line Procedure
  - `Ada.Text_IO.New_Line (Spacing => 3);`
- Put Procedure (Integer)
  - `Ada.Integer_Text_IO.Put (Item => How_Long, Width => 5);`

234	Width: 4	• 234
234	Width: 6	• • • 234
-234	Width: 6	• • -234
234	Width: Len	• • • 234
234	Width: 1	234

Input/Output Statements

- Put Procedure (Floating Point)
  - `Ada.Float_Text_IO.Put`
    `(Item => Inches, Fore => 5, Exp => 0);`

Value	Fore	Aft	Exp	Displayed value
3.14159	2	2	0	• 3.14
3.14159	1	2	0	3.14
3.14159	2	5	0	• 3.14159
3.14159	1	3	2	3.142E+00
0.1234	1	2	0	0.12
- 0.006	1	2	2	-6.00E-3
- 0.006	4	3	0	• • -0.006

Expressions with several operators

- W, X, Y, Z are Integers. Let X=3, Y=4, Z=7
  - `W := X * Y + Z;` → `W = 19`
  - `W := X – Y + Z;` → `W = 6`
  - `W := X – Y – Z;` → `W = -8`
  - `W := X – (Y – Z);` → `W = 6`

  - `Pi * R ** 2` is equivalent `to pi*(R ** 2)` and not `Pi * R ** 2`

- Keep it simple
- Use a lot of parentheses

Errors

- Three main categories of programming errors:
  - Compilation errors
    - Syntax errors
    - Semantic errors
  - Run-time errors (exceptions)
    - Constraint error
    - Data error
  - Logic or algorithmic errors

. . .

- Fred Donovan: Save all your files in your catalogue

What / why pseudocode?

- **Pseudocode**: a kind of programming language. Its syntax and semantics are in general less strict, so that algorithms can be formulated at a higher, more abstract level.
- Ex: Sorting algorithm
  - while not at end of list
compare adjacent elements
if second is greater than first
switch them
get next two elements
if elements were switched
repeat for entire list
- Pseudocode cannot be compiled/executed, has no real formatting or syntax rules. It is one step towards producing the final code.
- Benefit of pseudocode
  - Enables programmer to concentrate on the algorithms without worrying about all the syntactic details of a particular programming language.