16.070

Introduction to Computers & Programming

Ada
Display a message: “Hello there. We hope you enjoy studying Ada!”

Pseudo code: put (Hello there. We hope you enjoy studying Ada!)

WITH Ada.Text_IO; -- declare the package
PROCEDURE Hello IS

--| A very simple program; it just displays a greeting.
--| Author: Michael Feldman, The George Washington University
--| Last Modified: June 1998

BEGIN -- Hello

    Ada.Text_IO.Put(Item => "Hello there. ");
    Ada.Text_IO.Put(Item => "We hope you enjoy studying Ada!");
    Ada.Text_IO.New_Line;

END Hello;
Hello there. We hope you enjoy studying Ada!
WITH Spider;
PROCEDURE Walk_Line IS

-- Walk line with spider
-- Author: M. B. Feldman, The George Washington University
-- Last Modified: July 1998

BEGIN -- Walk_Line
  Spider.Start;
  Spider.Step;
  Spider.Step;
  Spider.Step;
  Spider.Step;
  Spider.Quit;
END Walk_Line;
A package is a way to encapsulate, or group, a set of related operations.

- Divided into 2 parts
  - Specification .ads
    “table of contents”
  - Body .adb
    actual program segments

- Standard Ada packages are “built in”, i.e., provided by the compiler
  - Standard (+, -, …, characters)
  - Character (Is_upper, To_Lower, …)
  - Numerics (Sqrt, Log, …)
  - Text_IO (get, put, open, …)
  - …
PROCEDURE Start IS

BEGIN

DrawRoom;
CurrentColumn := 10; -- these are in the spider's view
CurrentRow := 11;
Heading := North;
ChangeColor(NewColor => Green);
ShowSpider;
ShowDirection;
RandomSteps.Reset(Gen => GSteps);
RandomColors.Reset(Gen => GColors);
RandomDirections.Reset(Gen => GDirections);

END Start;
WITH Spider;

PROCEDURE Walk_Box IS

---------------------------------------
--| Walk 4 x 4 box with spider
--| Author: M. B. Feldman, The GWU
--| Last Modified: July 1998
---------------------------------------

BEGIN -- Walk_Box

Spider.Start;

Spider.Step;
Spider.Step;
Spider.Step;
Spider.Step;
Spider.TurnRight;
Spider.Step;
Spider.Step;
Spider.Step;
Spider.TurnRight;
Spider.Step;
Spider.Step;
Spider.Step;
Spider.TurnRight;
Spider.Quit;

END Walk_Box;
Algorithm with Single Loop

- Algorithm for drawing box
  1. Repeat steps a and b 4 times
     a. Take 3 steps forward
     b. Turn right
- A repetition is usually called a loop
- Use an Ada control structure called the FOR construct:
  ```ada
  WITH Spider;
  PROCEDURE Draw_Box_with_1_Loop IS
  ---------------------------------------------------------------
  --| Draw 4 x 4 box with spider - use loop
  --| Author: M. B. Feldman, The GWU
  --| Last Modified: July 1998
  ---------------------------------------------------------------
  BEGIN -- Draw_Box_with_1_Loop
    Spider.Start;
    Spider.ChangeColor(NewColor =>
                         Spider.Red);
    FOR Side IN 1..4 LOOP
      Spider.Step;
      Spider.Step;
      Spider.Step;
      Spider.TurnRight;
    END LOOP;
    Spider.Quit;
  END Draw_Box_with_1_Loop;
  ```
Reading and Writing Numbers

- Calculate the sum and product of two numbers
  - Algorithm:
    - Get the 2 numbers
      - Ask for the 2 numbers
      - Get number1
      - Get number2
    - Calculate and print the sum
      - Print “The sum is “
      - Print (number1 + number2)
    - Calculate and print the product
      - Print “The product is “
      - Print (number1 * number2)
  - Pseudo code
    Number1: number
    Number2: number

    Put (give me two whole numbers)
    Get (number1)
    Get (number2)

    Put (the sum of the numbers is )
    Put (number1 + number2)

    Put (the product of the numbers is )
    Put (number1 * number2)
sum_prod

-- sum_prod - sum and product, Skansholm #2.4.2

with Text_Io; -- specify packages we depend on
use Text_Io;
procedure Sum_Prod is
  -- declare integer I/O package
  package Int_Io is new Text_Io.Integer_Io(Integer);
  use Int_Io;
  -- declare any constants and variables required
  Number1, Number2 : Integer; -- numbers used
begin -- sum_prod
  -- ask user for numbers and read them
  Put_Line("Give me two whole numbers!");
  Get(Number1);
  Get(Number2);
  -- display sum and product of numbers
  Put("The sum of the numbers is:");
  Put(Number1+Number2);
  New_Line;
  Put("The product of the numbers is:");
  Put(Number1*Number2);
  New_Line;
end Sum_Prod;
Give me two whole numbers!
4
5
The sum of the numbers is: 9
The product of the numbers is: 20

Program complete.
Press <enter> to exit.
Alternative solution

```plaintext
with Text_Io; use Text_Io;

procedure Sum_Prod is
    -- declare integer I/O package
    package Int_Io is new Text_Io.Integer_Io( Integer );
    use Int_Io;

    -- declare any constants and variables required
    Number1, Number2, Total, Product : Integer;

    begin -- sum_prod
        -- ask user for numbers and read them
        Put ( "Please enter the first number ");
        Get ( Number1 );   Skip_Line;
        Put ( "Please enter the second number ");
        Get ( Number2 );   Skip_Line;

        -- display sum and product of numbers
        Total := Number1 + Number2;
        Product := Number1 * Number2;
        Put("The sum of the numbers is:" );
        Put(Total, Width=>7);   New_Line;
        Put("The product of the numbers is:" );
        Put(Product, Width=>3);   New_Line;
    end Sum_Prod;
```

Layout conventions
- one statement (thought) per line
- break long lines into readable segments
- indent lines to show different parts of the program
- blank lines separate parts of the program
- comments/header to help readers understand the program
Comments

- Lines starting with -- are ignored by the compiler. Only there to help someone reading the program.

- Good comments:
  - are always correct and up to date
  - conform to usual conventions of prose
  - provide information not immediately obvious
  - describe the intended effect of (part of) the program

- Minimum comments in any program:
  - the name of the program (name)
  - who wrote it and when (author & date written)
  - description of what the program does (purpose)
  - description of any constants or variables
  - description of purpose of each segment of code
  - assumptions made (precondition / postcondition)
Please enter the first number?
Please enter the second number 8
The sum of the numbers is: 15
The product of the numbers is: 56

Program complete.
Press <enter> to exit.
- Good programs
  - Meet specification
    - Verifiable, dependable
    - Correct
  - Natural
    - Abstraction, modularization, encapsulation
  - Efficient
  - Readable/”elegant”
Program structure and layout

---
-- sum_prod - sum and product ...
-- author: Joe B
-- last modified: 2/9/03
---

with ...;
use ...;

procedure program_name is
    declare I/O packages
    declare constants & variables used
begin  -- program_name
    statements
end  program_name ;
Reserved words

- Reserved words cannot be used for any other purpose than their special meaning.
  - abort abs accept access all and array at begin body case constant declare delay delta digits else elsif end entry exception exit for function generic goto if in is limited loop mod new not null of or others out package pragma private procedure raise range record rem renames return reverse select separate subtype task terminate then type use when while with xor

- Pre-defined words have standard pre-defined meaning. Their meaning can be changed ... with care ... but don't!
  - BOOLEAN CHARACTER CLOSE CREATE DELETE FALSE FLOAT GET INTEGER NATURAL NEW_LINE OPEN PUT PUT_LINE POSITIVE READ RESET SKIP_LINE STRING TEXT_IO TRUE WRITE