# Introduction to Computers and Programming

Prof. I. K. Lundqvist

Reading: B: 156-171; FK: 34-61

Lecture 2
Sept 8 2003

## In reply to 


- Palindrome example
- The von Neumann model
- Why is Ada good for mission critical applications?
- What kind of tests?
- How will Ada integrate into rest of unified?

- How to make CP more challenging?

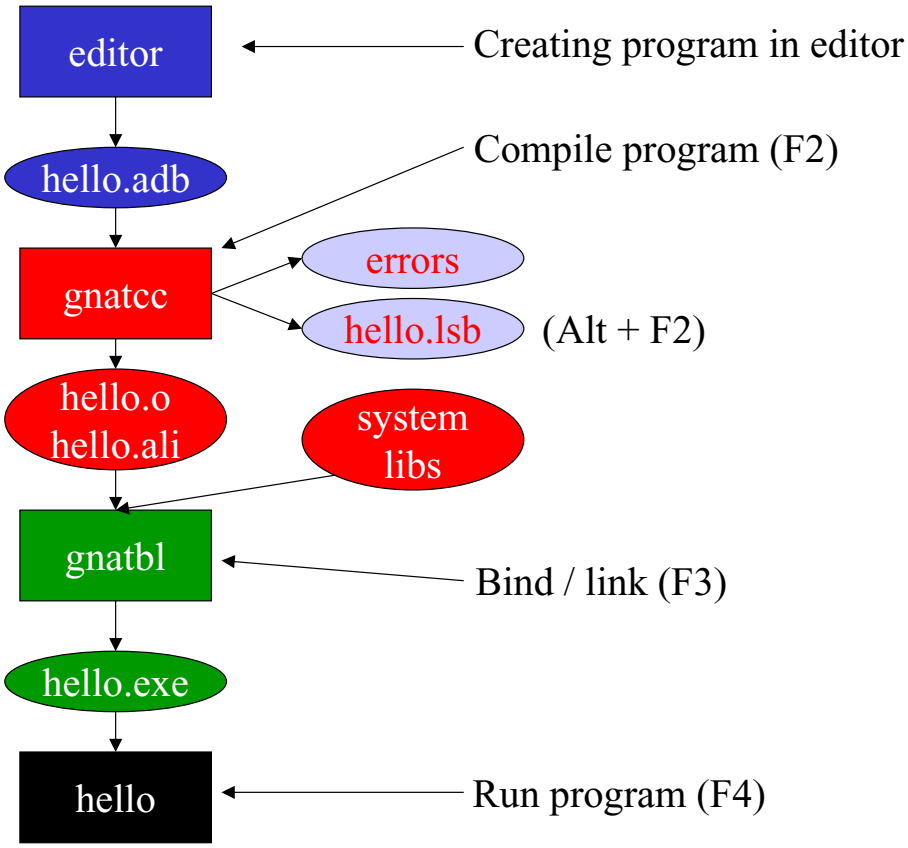
## CP homework vs XXX?

1. I want to do the regular homework problems in CP

2. I feel confident in my programming knowledge and would like to skip introductory Ada homework exercises and get something more challenging to do!

## Today

- Components of the “hello world” program
- Step-by-step compilation exercise
- Creating the listing file using AdaGIDE
- Programming style
- The Feldman Spider Adventure




# hello.adb [p. 33]

```
WITH Ada.Text_IO;
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.lsb – Listing files

```
GNAT 3.14p  (20010503) Copyright 1992-2001 Free Software Foundation, ..

Compiling:
c:\docume~1\kristina\mydocu~1\underv~1\unifie~1\adakod\hello.adb
(source file time stamp: 1998-09-13 21:04:32)

     1. WITH Ada.Text_IO;
     2. PROCEDURE Hello IS
     3. ------------------------------------------------------------
     4. --| A very simple program; it just displays a greeting.
     5. --| Author: Michael Feldman, The George Washington University
     6. --| Last Modified: June 1998
     7. ------------------------------------------------------------
     8. BEGIN - Hello
     9.
    10.   Ada.Text_IO.Put(Item => "Hello there. ");
    11.   Ada.Text_IO.Put(Item => "We hope you enjoy studying Ada!");
    12.   Ada.Text_IO.New_Line;
    13.
    14. END Hello;
    15.

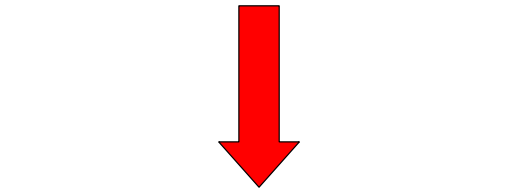

 15 lines: No errors
```

# Common Programming Errors - *bugs*

- Compilation errors
- Run-time errors
- Logic or algorithmic errors

# Compilation Errors

- Syntax errors
  - Fatal error that has to be fixed before code can be compiled
- Semantic errors
  - Inconsistency in the use of values, variables, packages, …



- Propagation errors



Correct the first **one** or **two** errors in program, then **recompile**

# Run-time Errors

- Detected during execution of a program
- Called **exception** in Ada
- In Ada we have a way of predicting the occurrence of exceptions and prevent the computer from halting
  - Exception handling

## Logic / Algorithm Errors

- Developing an incorrect algorithm for solving a problem
- Incorrect translation of a correct algorithm

The computer does only what you tell it to do, not what you meant to tell it to do … (*GIGO*)

## Comments, headers, and programming style

- Good programming style:

**Communication**□

- Good style leads to programs that are:
  - Understandable, readable, reusable, efficient, easy to develop and debug

# Comments, headers, and programming style

- Comments start with "--" and are ignored by the compiler

- ```
  ----------------------------------
  -- program name: my_first_program
  -- programmer: Jane B
  -- usage:
  -- compile:
  -- system:
  -- date: started 9/5/03
  --          phase 1 complete 9/8/03
  -- bugs:
  -- description:
  ----------------------------------
  ```

# Adventures of the Spider "introduction to algorithms"

- Simple picture-drawing creature – The Spider
  - Algorithmic constructs (control structures and parameters)
  - Ada packages
  ```

# Straight-Line Algorithms

- Program 2.3 – The Spider walks a line
- Program 2.5 – Spider commands with parameters
  - `TYPE Directions IS (North, East, South, West);`
  - `TYPE Colors IS (Red, Green, Blue, Black, None);`
    - Enumeration types: provides lists of values
  - `PROCEDURE Face (WhichWay: IN Directions);`

    ```
    -- Pre:  WhichWay has been assigned a value
    -- Post: Spider turns to face the given direction.
    ```

    - `Spider.Face(WhichWay => Spider.West);`
  - `PROCEDURE ChangeColor (NewColor: Colors);`

    ```
    -- Pre:  NewColor has been assigned a value
    -- Post: Spider leaves its tracks in the new color
    ```

    - `Spider.ChangeColor(NewColor => Spider.Red);`

# Algorithm with single loop

- Algorithm for drawing a box:
  - Repeat steps 1 and 2 four times
    1. Take three steps forward
    2. Turn right
  - A repetition usually called a **loop**□

```
FOR Side IN 1..4 LOOP
…

END LOOP;
```

# Algorithm with nested loop

- Algorithm for drawing a box:
  - Repeat steps 1 through 3 four times
    1. Choose a color
    2. Repeat step 2.1 three times
       2.1 Take one step forward
    3. Turn right

```
- FOR Side IN 1..4 LOOP
    Spider.ChangeColor(Spider.RandomColor);
    FOR Count IN 1..5 LOOP
      Spider.Step;
    END LOOP;
    Spider.TurnRight;
  END LOOP;
```

# Run-time error

```
WITH Spider;
PROCEDURE Spider_Crash IS

BEGIN -- Spider_Crash

  Spider.Start;
  Spider.ChangeColor(NewColor => Spider.Red);

  FOR Count IN 1..12 LOOP
    Spider.Step;
  END LOOP;

  Spider.Quit;

END Spider_Crash;
```

# Conditional execution

```
FUNCTION AtWall RETURN Boolean;
-- Pre:  None
-- Post: Return True if the spider is standing
--       next to a wall
```

```
IF Spider.AtWall THEN
  EXIT;
END IF;
```

# Concept Question

1. The Code will have no errors.
2. The Code will have compilation errors.
3. The Code will have logical errors.