Typical Errors

- Compilation errors
  - Syntax
  - Semantics

- Run-time errors
  - Exception

- Logic or Algorithm errors

- Propagation Errors

A Simple Ada Program

```ada
with Ada.Text_Io;
use Ada.Text_Io;

procedure Hello is
begin
  Put("Hello World");
end Hello;
```

Introduction to Computers and Programming

Prof. I. K. Lundqvist
Visibility Rules

- The **visibility rules** determine which declarations are visible and directly visible at each place within a program. The visibility rules apply to both explicit and implicit declarations.

- **Direct Visibility**
  - *immediate visibility*
  - *use-visibility*

---

For Loop

```plaintext
for <loop_control_variable> in <lower_bound>..<upper_bound> loop
  <loop_body>
end loop;
```

- `<loop_control_variable>`
  - This is the name of the "variable that controls the loop". The loop control variable is incremented by one each time through the loop.

- `<lower_bound>`
  - The initial value given to the loop control variable.

- `<upper_bound>`
  - The final value of the loop control variable. The loop body executes one more time when the loop control variable = upper bound, then the loop terminates.

- `<loop_body>`
  - The code that's executed each time through the loop.

---

Loop Demo [CQ1]

```plaintext
procedure Loop_Demo is
  -- Square_Integer : Integer;
begin
  for Square_Integer in 1 .. 5 loop
    Put (Item=>Square_Integer**2, Width => 3);
    New_Line;
  end loop;
end Loop_Demo;
```
The Program Output Appears As A, B or C? [CQ1]

A
1
4
9
16
25
B
0001
0004
0009
0016
0025
C
1
2
6
8
10

Concept Question 1
Single Loop

1. The program output appears as A
2. The program output appears as B
3. The program output appears as C
4. I still don’t understand loops ...

Procedure Nested_Loop_Demo is begin
-- Outer loop
for I in 2 .. 3 loop
-- Inner loop
for J in 1 .. 9 loop
Put (I);
Put (" *");
Put (J);
Put (" = ");
Put (I*J);
New_Line;
end loop;
end loop;
end Nested_Loop_Demo;

Concept Question 2
Nested Loops

1. The program goes through 2 iterations
2. The program goes through 9 iterations
3. The program goes through 18 iterations
4. The program goes through 27 iterations
5. I still don’t understand nested loops
• Type
  – Set of Values
  – Set of Operations

• Subtype
  – Defines a subset of the values associated with original type (base type)

Scalar Types

Discrete Types

Integer Types

Real Types

Floating Point

Composite Types

Enumeration Types

Fixed Point

Type Attributes

• TYPE’First, TYPE’Last, TYPE’Image (X)

```ada
with Ada.Text_IO;
procedure Print (A : Integer; P : Float) is
  type My_Int is range -100 .. 1_000_000;
  T : My_Int := My_Int’Last;
  type Day is (Mon, Tue, Wed, Thu, Fri, Sat, Sun);
  D : Day := Day’First;
  B : Integer := Integer’First;
begin
  Ada.Text_IO.Put (Integer’Image (A));
  Ada.Text_IO.Put (Float’Image (P));
end Print;
```

Enumerations

• An enumeration type is a sequence of ordered enumeration literals:

```ada
type State is (Off, Powering_Up, On);
```

• No arithmetic defined

```ada
S1, S2 : State;
S1 := Off;
S2 := Powering_Up;
S1 := S1 + S2; -- Illegal
```

• Can add/subtract one

```ada
State’Pred (S1)
State’Succ (S2)
```

function Compute (P, Q : Integer) return Integer is
  type My_Int is range -100 .. 1_000_000;
  T : My_Int := P + 1;
begin
  return (T + Q);
end Compute;
```
Functions

- **<function_header>**
  - contains the function name and parameters.
- **<local_variables>**
  - variables used in the function (but nowhere else).
- **<function_body>**
  - the code the function executes.
- **<function_name>**
  - the name of the function.

```plaintext
function <function name> (...
  return <data type>
is
begin
  if N <= 1 then
    return 1;
  else
    return N * Fact (N-1);
  end if;
end Fact;
```

Procedures

- **<procedure_header>**
  - contains the procedure name and parameters.
- **<local_variables>**
  - variables used in the procedure (but nowhere else).
- **<procedure_body>**
  - the code the procedure executes.
- **<procedure_name>**
  - the name of the procedure.

```plaintext
procedure Fact (N : Integer) return Integer is
begin
  if N <= 1 then
    return 1;
  else
    return N * Fact (N-1);
  end if;
end Fact;
```
Procedure Calls

No Parameters

<procedure name>

with Hello;
procedure Main is
begin
    Hello;
end Main;

With Parameters

<procedure name> ( ...

with Increment;
procedure Main is
    my_x : integer := 1;
    my_y : float := 2.0;
begin
    Increment(my_x, my_y);
end Main;

Arrays

- Single Dimension arrays A(I)
- Two dimensional arrays A(I,J)
- N dimensional array A(i_1, i_2, ..., i_n)

Loops can be used to access elements.

for I in 1 .. N loop
    for J in 1 .. N loop
        Put (B(I,J));
    end loop;
end loop;

Records

type My_Type_Record is
  record
    my_boolean : Boolean;
    my_integer : Integer;
    my_real : Float;
  end record;

Rec1 : my_type_record;
Rec2 : my_other_type_record;

  Rec2 := Rec1;

  Rec1.my_boolean := Rec2.my_boolean;
  Rec1.my_integer := Rec2.my_integer;
  Rec1.my_Real := Rec2.my_real;

JK

- CP Review Session (NOT required)
  - 7:30