16.35
Aerospace Software Engineering

Ada 95 – “some basics”
Cost study done in 1973-1974 determined that the US Department of Defense was spending $3 billion annually on software.

Evaluated 23 existing languages against the “Tinman” requirements:

- FORTRAN, COBOL, PL/I, HAL/S, TACPOL, CMS-2, CS-4, SPL/I, JOVIAL J3, JOVIAL J73, ALGOL 60, ALGOL 68, CORAL 66, Pascal, SUMULA 67, LIS, LTR, TRL/2, EUCLID, PDL2, PEARL, MORAL, EL/I

Augusta Ada Byron (1815-1852), Countess of Lovelace
A Small Ada Program - Mini

-- What is program supposed to be doing? Inputs/outputs?
-- Author(s):
-- Last updated:
-- Comments:

with Ada.Text_Io;

procedure Mini is
   -- declarations
   K    : constant Integer := 5;
   X, Y : Integer;

begin
   -- executable code
   Ada.Text_Io.Put (Item => "Hello world!");
   X := 10;
   Y := X * K; -- calculations
   Y := X + 1;
end Mini;

Reserved Words

- These identifiers are reserved for special significance in the language

- A *reserved word* must not be used as a declared identifier

<table>
<thead>
<tr>
<th>abort</th>
<th>else</th>
<th>new</th>
<th>select</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td>endif</td>
<td>not</td>
<td>separate</td>
</tr>
<tr>
<td>abstract</td>
<td>end</td>
<td>null</td>
<td>subtype</td>
</tr>
<tr>
<td>accept</td>
<td>entry</td>
<td>of</td>
<td>tagged</td>
</tr>
<tr>
<td>access</td>
<td>exception</td>
<td>or</td>
<td>task</td>
</tr>
<tr>
<td>aliased</td>
<td>exit</td>
<td>others</td>
<td>terminate</td>
</tr>
<tr>
<td>all</td>
<td>for</td>
<td>out</td>
<td>then</td>
</tr>
<tr>
<td>and</td>
<td>function</td>
<td></td>
<td>type</td>
</tr>
<tr>
<td>array</td>
<td>at</td>
<td>generic</td>
<td></td>
</tr>
<tr>
<td>begin</td>
<td>goto</td>
<td>package</td>
<td></td>
</tr>
<tr>
<td>body</td>
<td>if</td>
<td>pragma</td>
<td>until</td>
</tr>
<tr>
<td>case</td>
<td>in</td>
<td>private</td>
<td>use</td>
</tr>
<tr>
<td>constant</td>
<td>is</td>
<td>procedure</td>
<td></td>
</tr>
<tr>
<td>declare</td>
<td>limited</td>
<td>protected</td>
<td>when</td>
</tr>
<tr>
<td>delay</td>
<td>loop</td>
<td>range</td>
<td>while</td>
</tr>
<tr>
<td>delta</td>
<td>mod</td>
<td>record</td>
<td>xor</td>
</tr>
<tr>
<td>digits</td>
<td>rem</td>
<td>renames</td>
<td></td>
</tr>
<tr>
<td>do</td>
<td>requeue</td>
<td>return</td>
<td></td>
</tr>
<tr>
<td></td>
<td>reverse</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Finding the Value of a Coin Collection

- **Problem Specification**
  - Your little sister has been saving nickels and pennies for quite a while. Because she is getting tired of lugging her piggy bank with her whenever she goes to the store, she would like to trade in her collection for one-dollar banknotes and some change. To do this, she would like to know the value of her coin collection in dollars and cents.

- **Analysis**
  - To solve this problem, we must be given the count of nickels and pennies in the collection. The first step is to determine the total value of the collection in cents. Once we have this figure, we can do an integer division using 100 as the divisor to get the dollar value; the remainder of this division will be the loose change that she should receive. In the data requirements below, we list the total value in cents \((\text{TotalCents})\) as a program variable because it is needed as part of the computation process; it is not a required problem output.
Finding the Value of a Coin Collection

- Data Requirements and Formulas
  - Problem Inputs:
    Nickels : Natural (the number of nickels)
    Pennies : Natural (the number of pennies)
  - Problem Outputs:
    Dollars : Integer (the number of $s she should receive)
    Change : Integer (the loose change she should receive)
  - Additional Program Variables:
    TotalCents : Integer (the total number of cents)
  - Relevant Formulas
    One nickel equals 5 pennies
Finding the Value of a Coin Collection

- Design
  - Initial Algorithm
    1. Read in the count of nickels and pennies
    2. Compute the total value in cents
    3. Find the value in dollars and loose change
    4. Display the value in dollars and loose change
Finding the Value of a Coin Collection

- **Design**
  - **Refined Algorithm**
    1. Read in the count of nickels and pennies
    2. Compute the total value in cents
      1. TotalCents is 5 times Nickels plus Pennies
    3. Find the value in dollars and loose change
      1. Dollars is the integer quotient of TotalCents and 100
      2. Change is the integer remainder of TotalCents and 100
    4. Display the value in dollars and loose change
## Finding the Value of a Coin Collection

### Test Plan

<table>
<thead>
<tr>
<th>Test case</th>
<th>Nickles</th>
<th>Pennies</th>
<th>Reason</th>
<th>Expected output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30</td>
<td>77</td>
<td>Typical</td>
<td>$2.27</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>59</td>
<td>No nickels</td>
<td>$0.59</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>0</td>
<td>No pennies</td>
<td>$0.65</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>-5</td>
<td>Negative</td>
<td>?</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>No coins</td>
<td>$0.00</td>
</tr>
<tr>
<td>6</td>
<td>qwerty</td>
<td>4</td>
<td>Bad input</td>
<td>?</td>
</tr>
</tbody>
</table>
Finding the Value of a Coin Collection

```ada
with Ada.Text_Io;
with Ada.Integer_Text_Io;

procedure Coin_Collection is
  Pennies    : Natural; -- input, number of pennies
  Nickels    : Natural; -- input, number of nickels
  Dollars    : Natural; -- output, value in dollars
  Cents      : Natural; -- output, value in cents
  Totalcents : Natural;

begin -- coin_collection
  -- prompt user for number of nickels and pennies
  Ada.Text_Io.Put (Item => "how many nickels do you have? ");
  Ada.Integer_Text_Io.Get (Item => Nickels);
  Ada.Text_Io.Put (Item => "how many pennies do you have? ");
  Ada.Integer_Text_Io.Get (Item => Pennies);

  Totalcents := 5 * Nickels + Pennies; -- compute total value in cents
  Dollars := Totalcents / 100; -- find value in dollars and change
  Cents := Totalcents rem 100;

  -- display the value in dollars and change
  Ada.Text_Io.Put (Item => "Your collection is worth ");
  Ada.Integer_Text_Io.Put (Item => Dollars, Width => 1);
  Ada.Text_Io.Put (Item => " dollars and ");
  Ada.Integer_Text_Io.Put (Item => Cents, Width => 1);
  Ada.Text_Io.Put (" cents.");
  Ada.Text_Io.New_Line;

end Coin_Collection;
```
Testing

how many nickels do you have? 30
how many pennies do you have? 77
Your collection is worth 2 dollars and 27 cents.
Common Programming Errors

- Compilation errors, run-time errors (*exceptions* in Ada), and logic or algorithm errors.

- Compilation errors:
  - **Distance_with_Errors**
    - Finds distance traveled, given travel time and average speed
with Ada.Text_Io;
with Ada.Float_Text_Io;

procedure Distance_With_Errors is
  -- Finds distance, given travel time and average speed
  Howlong : Natural;
  Howfast : Float;
  Howfar : Natural;

begin
  -- distance_with_errors
  -- prompt user for hours and average speed
  Ada.Text_Io.Put (Item => "How long will you be driving (integer) ? ");
  Ada.Float_Text_Io.Get (Item => Howlong);
  Ada.Text_Io.Put (Item => "At what speed (miles per hour, integer) ?");
  Ada.Float_Text_Io.Get (Item => Howfast);

  -- compute distance driven
  Howfast := Howlong * Howfar;

  -- display result
  Ada.Text_Io.Put (Item => " you will travel about ");
  Ada.Float_Text_Io.Put (Item => Howfar);
  Ada.Text_Io.Put (Item => " miles");
  Ada.Text_Io.New_Line;
end Distance_With_Errors;
procedure Distance_With_Errors is
  -- Finds distance, given travel time and average speed
  Howlong : Natural;
  Howfast : Float;
  Howfar : Natural;
begin -- distance_with_errors
  -- prompt user for hours and average speed
  Ada.Text_Io.Put (Item => "How long will you be driving (integer) ? ");
  Ada.Float_Text_Io.Get (Item => Howlong);
  Ada.Text_Io.Put (Item => "At what speed (miles per hour, integer) ? ");
  Ada.Float_Text_Io.Get (Item => Howfast);
  -- compute distance driven
  Howfast := Howlong * Howfar;
  -- display result
  Ada.Text_Io.Put (Item => " you will travel about ");
  Ada.Float_Text_Io.Put (Item => Howfar);
  Ada.Text_Io.Put (Item => " miles");
  Ada.Text_Io.New_Line;
end Distance_With_Errors;
procedure Distance_With_Errors is

  Howlong : Natural;
  Howfast : Float;
  Howfar : Natural;

begin -- distance_with_errors
  -- prompt user for hours and average speed
  Ada.Text_Io.Put (Item => "How long will you be driving (integer) ? ");
  Ada.Float_Text_Io.Get (Item => Howlong);

  Ada.Text_Io.Put (Item => "At what speed (miles per hour, integer) ?");
  Ada.Float_Text_Io.Get (Item => Howfast);

  -- compute distance driven
  Howfast := Howlong * Howfar;

  Ada.Text_Io.Put (Item => " you will travel about ");
  Ada.Float_Text_Io.Put (Item => Howfar);

  Ada.Text_Io.Put (Item => " miles");
  Ada.Text_Io.New_Line;

end Distance_With_Errors;
Exceptions (Run-Time Errors)

- How many nickels do you have? 13
  How many pennies do you have? -5
  - CONSTRAINT ERROR
    - Variables out of range in their programs

- How many nickels do you have? qwerty
  - Ada.IO_EXCEPTIONS.DATA_ERROR
    - Input/output exception
Control Structures

- So far: straight-line algorithms
- Control structures: if, for, while

```ada
if X >= 0.0 then
    Ada.Text_Io.Put (Item => "Positive");
else
    Ada.Text_Io.Put (Item => "Negative");
end if;

if X > 0.0 then
    Posprod = Posprod * X;
    Countpos = Countpos + 1;
end if;
```
Ada.Text_IO.Put(Item => "Hello there. ");
Ada.Text_IO.Put(Item => "Hello there. ");
Ada.Text_IO.Put(Item => "Hello there. ");
Ada.Text_IO.Put(Item => "Hello there. ");
Ada.Text_IO.Put(Item => "Hello there. ");

-- Can Be Written More Concisely As
for Count in 1..5 loop
    Ada.Text_IO.Put(Item => "Hello there. ");
end loop;

for Counter in Min .. Max loop
    Ada.Integer_Text_IO.Put (Item => I,
                              Width => 5);
    Ada.Text_IO.New_Line;
end loop;
While

Power := 1;
while Power < 10000 loop
   Ada.Integer_Text_Io.Put (Item => Power,
                           Width => 5);
   Power = Power * 2;
end loop;

Power := 1;
loop
   exit when Power >= 10000;
   Ada.Integer_Text_Io.Put (Item => Power,
                           Width => 5);
   Power = Power * 2;
end loop;
AdaGIDE

```
with Ada.Text_Io;
use Ada.Text_Io;
procedure hello is
begin
   put line("Hello");
end hello;
```

Compiling...

hello.adb:9:21: missing ";;"
Completed.
The GNU Visual Debugger

Toolbar
Canvas
Highlighted Source
Explorer
Breakpoint
Debugger Window

Dereference
Record type
Unknown value
Current line

Type String Access is access String:
Type Array Of String is array (1 .. 2) of String Access:
Type Null Record is null record;
Type My_Record is record:
Field1 : Access_Type;
Type My_Record Access is access all My_Record:
Type My_Record Array is array (1 .. 100) of My_Record:
break parse adb 105
graph display Tcb
break parse adb 105
graph display y
file parse adb.line 105.

file parse adb. line 105.
- Problem set 1 due 9/25/02, 3pm
- Email addresses:
  - 16.35-students@mit.edu
  - 16.35-staff@mit.edu
- http://web.mit.edu/16.35/www
- Class project
  - X-38 Integrated Test Plan
  - X-38 Software Development Plan
  - Early Flight Control System - Statement of Work
  - Software Development Plan
  - Software Test Plan
Project Groups

- Mandic, Milan
  Nyenke, Chinwe P
  da Silva, Lanya M
  Riedel, Robin

- Modisette, James M
  Guzman, Bryan J
  Utter, Darlene A
  Bly, Elizabeth
  Kambouchev, Nayden

- Phifer, Gabriel S
  Sidelnik, Nicholas
  Ouellette, Joshua T
  Chang, Catherine W
  Stringfellow, Margaret

- Minogue, Kenneth
  Broniatowski, David
  Qu, Shen
  Guevara, Gerardo