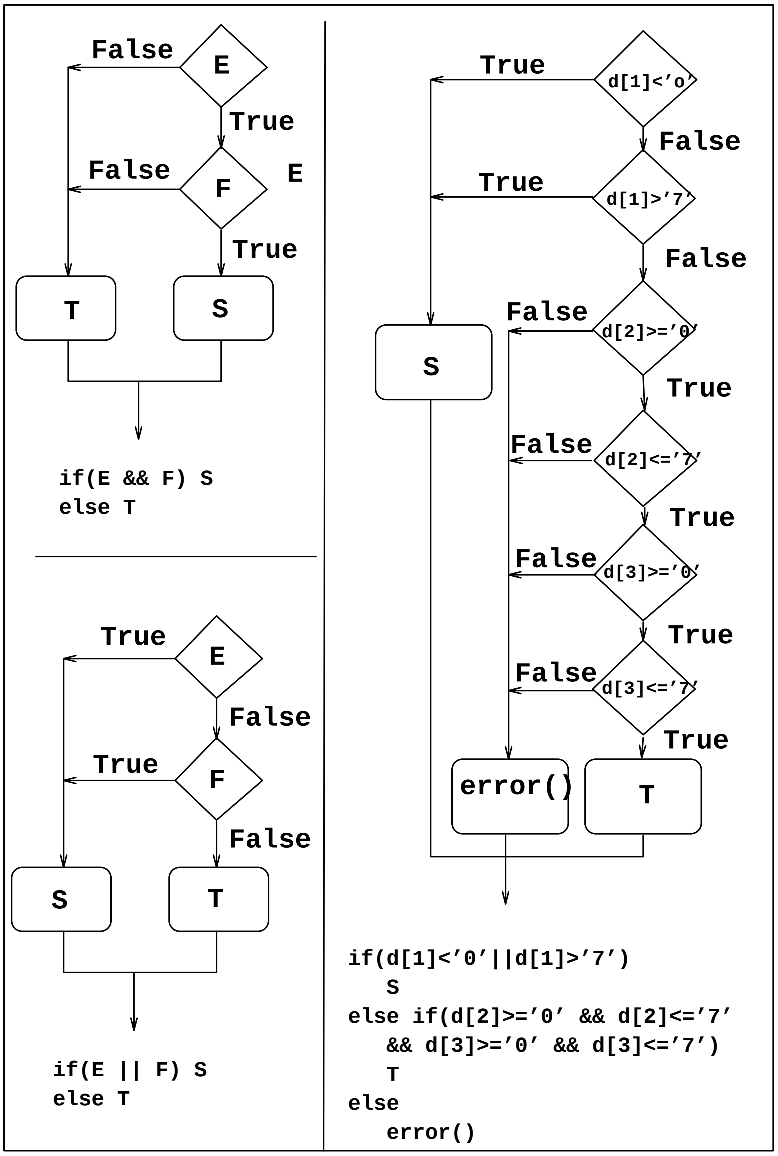
Programming Languages: Solutions to Homework 2

B. Mishra

October 23 1995

Problem 1. [Problems 3.6 (pp. 112) in Sethi: Programming Languages—Concepts and Constructs.]

Solution.



Problem 2. *An instant or interval of time expressed in hours, minutes and seconds can be represented by a record. For instance, one could define a type in Pascal as follows*

```
type HOUR     = 0..23;
type MINUTE   = 0..59;
type SECOND   = 0..59;
type TIME     = record hr: HOUR; min: MINUTE; sec: SECOND end;
```

Give the corresponding definition in ADA.

The following programs (with missing bodies) are supposed to add and subtract time intervals. Fill in the bodies:

```
function "+"(U,V: in TIME) return TIME is
   W: TIME;
begin
   --- MISSING BODY
return(W);
end "+";

function "-"(U,V: in TIME) return TIME is
   W: TIME;
begin
   --- MISSING BODY
return(W);
end "-";
```

Remark on the differences in style of programming as you go from Pascal to Ada.

For Extra Credit

How would you change it so that depending on a tag, the time could be in 12 Hrs (AM/PM) notation or in 24 Hrs notation as above?

Solution.

```
package pkg_definition is
  type type_tag is (twelve, twenty_four);
  type type_ampm is (am, pm);
  type type_time(tag: type_tag) is record
    minute: integer range 0..59;
    second: integer range 0..59;
    case tag is
      when twelve =>
        ampm: type_ampm;
        hour12: integer range 0..11;
      when twenty_four =>
        hour24: integer range 0..23;
```

```
    end case;
  end record;

  subtype type_time12 is type_time(twelve);
  subtype type_time24 is type_time(twenty_four);
end pkg_definition

--  ....

with text_io;
use text_io;
package int_io is new integer_io(integer);


with text_io;
use text_io;
package real_io is new float_io(float);

--  ....

with pkg_definition;
with text_io; with int_io;
package pkg_utility is
  use pkg_definitions;
  function ``+´´(u,v: in type_time) return type_time;
end pkg_utility;

with pkg_definition;
with text_io; use text_io;
with int_io; use int_io;
package body pkg_utility is
  use pkg_definitions;
  function ``+´´(u,v: in type_time) return type_time is
    w24:type_time(twenty_four)
    w12:type_time(twelve)
    hr, min, sec: integer;
  begin
    if u.tag=twenty_four and v.tag=twenty_four then
      hr := u.hour24 + v.hour24;
      min := u.minute + v.minute;
      sec := u.second + v.second;
      w24.second := sec mod 60;
      w24.minute := (sec/60 + min) mod 60;
      w24.hour24 := ((sec/60 + min)/60 + hr) mod 24;

      return w24;
    elsif u.tag=twelve and v.tag=twelve then
      hr := u.hour12 + v.hour12;
```

```
      if u.ampm = pm then
         hr := hr + 12;
      end if;
      if v.ampm = pm then
         hr := hr + 12;
      end if;
      min := u.minute + v.minute;
      sec := u.second + v.second;
      w12.second := sec mod 60;
      w12.minute := (sec/60 + min) mod 60;
      hr := ((sec/60 + min)/60 + hr) mod 24;
      if hr>11 then
        w12.ampm := pm;
        w12.hour12 := hr mod 12;
      else
        w12.ampm := am;
        w12.hour12 := hr;
      end if;

      return w12;
    end if;
  end ``+´´;
end pkg_utility;
```

Problem 3. *We have discussed before that, in PASCAL, it is impossible to write a single search procedure that can search for a given character within a given string. That is because strings of different lengths have different types in PASCAL. The idea of a "generic" procedure can be used to solve this problem in ADA. Write an ADA program to search for an item within an array; the procedure should be generic with respect to length of the array and type of the item.*

Write a short discussion of the advantages and disadvantages of this approach.

Hint: *Consider what happens if you use the generic procedure to search strings of many different lengths in the same program.*

Solution. An Ada program to search for an item over an array:

```
generic
  n: POSITIVE;
  type ITEM is private;
  type SEARCH_ARRAY is array(1..n) of ITEM;
function SEARCH(S: in SEARCH_ARRAY;
                IT: in ITEM) return POSITIVE is
  J: POSITIVE;
begin
  FOUND := FALSE;
```

```
    for J in S´RANGE loop
      exit when (S(J) = IT);
    end loop;
    return J;
  end SEARCH
```

In standard Pascal, one gets around the problem of a lack of dynamic array types by writing separate procedure for each index type and bounds, or by making the program general enough to take care of all index bounds. In Ada, with a powerful abstraction mechanism, the compiler can do the same thing by instantiating a copy of the procedure for each instance of the index type and bound that will be used. This has the advantage of saving the programmer from a repetitive and error-prone task and of making the program more readable.

However, this is at the expense of a rather larger object program and a more complicated procedure invocation.