```
(* DIGITAL EQUIPMENT CORPORATION  CONFIDENTIAL AND PROPRIETARY *)
(* Last modified on Wed May  2  9:19:32 PDT 1990 by mhb *)
(*      modified on Mon Dec 12 15:46:07 PST 1988 by levin *)

MODULE Easter;
IMPORT FormsVBT, PaintExtras, VBT, Time, Trestle;
IMPORT ParseParams, Rd, Text, Wr;
FROM Stdio IMPORT stdin, stdout, stderr;

(* This program is a faithful transliteration of the one in section 8.2.1 of
   "Informal Introduction to Algol 68", by C. H. Lindsey and S. G. van der
   Meulen, North-Holland Publishing Co., 1971.

   That is, Roy Levin and Marc H. Brown claim no responsibility for its
   accuracy, coding style, etc. *)


VAR
  verbose, dialog: BOOLEAN;
  year, date, moon, paschal, easter: INTEGER;

CONST
  March21st = 31 + 28 + 21;
  (* The Gregorian calendar was introduced into various parts of the world at
     different dates. In Great Britain, the year was: *)
  GregoryStart = 1752; (* or whatever date you prefer *)

VAR
  century, leap, dominic, golden, lilius, epact, clavius: INTEGER;

PROCEDURE Mod(a: INTEGER; b: CARDINAL): CARDINAL;
  VAR t: CARDINAL;
  BEGIN
    IF a < 0 THEN
      t := (-a) MOD b;
      IF t = 0 THEN RETURN 0 ELSE RETURN b - t END;
    END;
    RETURN a MOD b
  END Mod;

PROCEDURE WeekDay(date: INTEGER): Text.T;
  BEGIN
    CASE Mod(date - dominic, 7) + 1 OF
    | 1: RETURN "Sunday";
    | 2: RETURN "Monday";
    | 3: RETURN "Tuesday";
    | 4: RETURN "Wednesday";
    | 5: RETURN "Thursday";
    | 6: RETURN "Friday";
    | 7: RETURN "Saturday";
    END;
    RETURN NIL
  END WeekDay;

PROCEDURE Month(date: INTEGER): Text.T;
  BEGIN
    IF date <= 31 THEN RETURN "January" END;
    date := date - 31;
    IF date <= 28 + leap THEN RETURN "February" END;
    date := date - 28 - leap;
    IF date <= 31 THEN RETURN "March" END;
    date := date - 31;
    IF date <= 30 THEN RETURN "April" END;
    date := date - 30;
    IF date <= 31 THEN RETURN "May" END;
    date := date - 31;
    IF date <= 30 THEN RETURN "June" END;
    date := date - 30;
    IF date <= 31 THEN RETURN "July" END;
    date := date - 31;
    IF date <= 31 THEN RETURN "August" END;
    date := date - 31;
    IF date <= 30 THEN RETURN "September" END;
    date := date - 30;
    IF date <= 31 THEN RETURN "October" END;
    date := date - 31;
    IF date <= 30 THEN RETURN "November" END;
    date := date - 30;
    RETURN "December"
  END Month;

PROCEDURE Day(date: INTEGER): Text.T;
  BEGIN
    IF date <= 31 THEN RETURN NumAsText(date) END;
    date := date - 31;
    IF date <= 28 + leap THEN RETURN NumAsText(date) END;
    date := date - 28 - leap;
    IF date <= 31 THEN RETURN NumAsText(date) END;
    date := date - 31;
    IF date <= 30 THEN RETURN NumAsText(date) END;
    date := date - 30;
    IF date <= 31 THEN RETURN NumAsText(date) END;
    date := date - 31;
    IF date <= 30 THEN RETURN NumAsText(date) END;
    date := date - 30;
    IF date <= 31 THEN RETURN NumAsText(date) END;
    date := date - 31;
    IF date <= 31 THEN RETURN NumAsText(date) END;
    date := date - 31;
    IF date <= 30 THEN RETURN NumAsText(date) END;
    date := date - 30;
    IF date <= 31 THEN RETURN NumAsText(date) END;
    date := date - 31;
    IF date <= 30 THEN RETURN NumAsText(date) END;
    date := date - 30;
    RETURN NumAsText(date)
  END Day;

PROCEDURE Compute();
  BEGIN
    century := year DIV 100;
      (* leap = 1 for a leap year, and 0 otherwise. *)
    leap :=
      ORD(((year MOD 4 = 0) AND (year MOD 100 # 0)) OR (year MOD 400 = 0));
      (* To calculate the day of the week corresponding to any date, we
         associate with each year a Dominical Letter, whose position in the
         alphabet gives the date of the first Sunday in January. *)
    dominic :=
      7 - Mod(year + year DIV 4 - century + century DIV 4 - 1 - leap, 7);
      (* The moon revolves around the earth once every 29.530588 days. 235
         such lunations last just 1.5 hours less than 19 Julian years. The
         calendar is therefore based on a "Metonic" cycle of 10 years, each
         year in a cycle being allotted a "Golden Number" in the range 1 to
         19: *)
    golden := (year MOD 19) + 1;
      (* However, following this cycle indefinitely would introduce an error
         of approximately 0.43 days per century. There is therefore a
         correction which, for convenience, is only allowed to change at the
```

```
        end of a century: *)
    lilius := (* who is a not inherently meaningful identifier *)
      Mod(century - century DIV 4
          (* for the leap years omitted at the start of some centuries *) -
          (century - (century - 17) DIV 25) DIV 3 (* the 1.5 hours error *) -
          8,
          30);
      (* On the 1st of January of any year, the number of days since the last
         new moon is given by the "Epact": *)
    epact := Mod(11 * (golden - 1) - lilius, 30);
      (* If successive new moons were to occur every 30 days, then we should
         be able to associate with each date a unique epact, one less for
         each day modulo 30 (then that date would be a new moon in years with
         that epact). In fact, six times in the year (and once extra at the
         end of 19 years) we must have a lunation of only 29 days, whereupon
         the sequence of epacts slips back a day and some date will have two
         epacts listed against it. These dates have been carefully chosen (it
         is alleged) so as to minimize the deviation from the true moon. One
         of them occurs in February and so happenings in March occur exactly
         59 days after those in January (or 60 in a leap year, since the
         intercalary day, if any, in February is automatically added to the
         lunation in which it occurs). Therefore, there is a new moon on: *)
    moon    := 31 - epact + 59 + leap;
    paschal := moon + 13;
    IF paschal < March21st + leap THEN
        (* The fourteenth day of this moon falls before the Vernal Equinox
           and we want the next one. The next date with two epacts against it
           occurs in April, the critical epact being given by: *)
      IF golden > 11 THEN clavius := 26 ELSE clavius := 25 END;
      IF epact >= clavius THEN moon := moon + 30; ELSE moon := moon + 29; END;
      paschal := moon + 13;
    END;
    easter := paschal + 7 - Mod(paschal - dominic, 7);
  END Compute;

PROCEDURE ToLetter(n: INTEGER): Text.T;
  BEGIN
    RETURN Text.FromChar(CHR(n + ORD('A') - 1))
  END ToLetter;

PROCEDURE NumAsText(n: INTEGER): Text.T;
  BEGIN
    RETURN PaintExtras.IntToText(n)
  END NumAsText;

PROCEDURE YearChanged(
  v:         FormsVBT.T;
  eventName: Text.T;
  closure:   REFANY;
  eventTime: Time.Ticks
  );
  BEGIN
    Update(v);
  END YearChanged;

PROCEDURE Advance(
  v:         FormsVBT.T;
  eventName: Text.T;
  closure:   REFANY;
  eventTime: Time.Ticks
  );
  VAR
    delta, now: INTEGER;
  BEGIN
    now := easter;
    IF Text.Equal(eventName, "nextBtn") THEN delta := 1 ELSE delta := -1 END;
    REPEAT
      INC(year, delta);
      FormsVBT.PutInteger(v, "easterYear", year);
      Update(v);
    UNTIL now = easter;
  END Advance;

PROCEDURE Update(fv: FormsVBT.T);
  VAR
    month: Text.T;
  BEGIN
    year := FormsVBT.GetInteger(fv, "easterYear");
    Compute();
    FormsVBT.PutBoolean(fv, "leapYear", (leap = 1));
    IF leap = 1 THEN
      FormsVBT.PutText(
        fv, "dominicalLetter",
        Text.Cat(ToLetter(dominic), "|", ToLetter(Mod(dominic - 2, 7) + 1)))
    ELSE
      FormsVBT.PutText(fv, "dominicalLetter", ToLetter(dominic))
    END;
    FormsVBT.PutInteger(fv, "goldenNumber", golden);
    FormsVBT.PutInteger(fv, "epact", epact);
    FormsVBT.PutText(
      fv, "fullMoon", Text.Cat(WeekDay(paschal), " ",
                               Text.Cat(Month(paschal), " ", Day(paschal))));
    FormsVBT.PutInteger(fv, "easterYear", year);
    FormsVBT.PutText(
      fv, "easterDate", Text.Cat(Month(easter), " ", Day(easter)));
  END Update;

PROCEDURE RunDialog();
  VAR
    fv: FormsVBT.T;
    z:  VBT.T;
  BEGIN
    fv := FormsVBT.NewFromFile("Easter.fv");
    FormsVBT.AttachProc(fv, "easterYear", YearChanged, NIL);
    FormsVBT.AttachProc(fv, "nextBtn", Advance, NIL);
    FormsVBT.AttachProc(fv, "prevBtn", Advance, NIL);
    IF year # 0 THEN FormsVBT.PutInteger(fv, "easterYear", year); END;
    Update(fv);
    z := Trestle.Chassis(fv, "easter");
    Trestle.Install(z, "easter");
    Trestle.AwaitDelete(z);
  END RunDialog;

PROCEDURE RunStdout();
  BEGIN
    IF year <= GregoryStart THEN
      Wr.PrintF(
        stderr, "The Gregorian calendar was not introduced until %d\n",
        GregoryStart);
      HALT(2)
    END;
    Compute();
    IF verbose THEN
      Wr.PrintF(stdout, "For %d", year);
      IF leap = 1 THEN Wr.PutText(stdout, " (leap year)"); END;
      Wr.PutText(stdout, ":\n");
      Wr.PutText(stdout, "  Dominical Letter: ");
      Wr.PutText(stdout, ToLetter(dominic));
```

```
      IF leap = 1 THEN
        Wr.PutText(stdout, "|");
        Wr.PutText(stdout, ToLetter(Mod(dominic - 2, 7) + 1));
      END;
      Wr.PutChar(stdout, "\n");
      Wr.PrintF(stdout, "  Golden Number: %d\n", golden);
      Wr.PrintF(stdout, "  Epact: %d\n", epact);
      Wr.PrintF(stdout, "  Paschal Full Moon: %t, %t %t\n", WeekDay(paschal),
                Month(paschal), Day(paschal));
      Wr.PutText(stdout, "  Easter: ");
    END;
    Wr.PrintF(stdout, "%t %t\n", Month(easter), Day(easter));
  END RunStdout;

PROCEDURE Main();
  BEGIN
    year := 0;
    IF ParseParams.NumParameters = 1 THEN
      dialog := TRUE;
    ELSE
      ParseParams.BeginParsing(stderr);
      verbose := ParseParams.KeywordPresent("-v");
      dialog  := ParseParams.KeywordPresent("-d");
      ParseParams.UnparsedTail();
      IF ParseParams.NextParameter < ParseParams.NumParameters THEN
        TRY
          year := ParseParams.GetNextInt();
          IF year <= GregoryStart THEN
            Wr.PrintF(
              stderr, "The Gregorian calendar was not introduced until %d\n",
              GregoryStart);
            HALT(1)
          END;
          ParseParams.EndParsing();
        EXCEPT ParseParams.ScanFailed:
          Wr.PutText(stderr, "Usage: easter [[-d] [-v] year]\n");
          HALT(2)
        END;
      END;
    END;
    IF dialog THEN RunDialog() ELSE RunStdout() END
  END Main;

BEGIN Main() END Easter.
```