```
MODULE FFT;


  (*
    Title: Fast-Fourier-Transformation, FFT
    LastEdit: 4th January 2012
    Author: Dr. Markus Bautsch, Berlin
    Programming Language: Component Pascal
    Reference: Elbert Oran, FFT-Anwendungen, Oldenbourg, München, Wien, 1997
  *)


  IMPORT

    Math, Out;


  TYPE

    COMPLEX* = RECORD re*, im*: REAL END;

    FFTArray* = POINTER TO ARRAY OF COMPLEX;

    FFTArray2D* = POINTER TO ARRAY OF FFTArray;

    CosSinArray = POINTER TO ARRAY OF REAL;

    BitrevArray = POINTER TO ARRAY OF INTEGER;


  VAR

    Bitrev: BitrevArray;
    Cos, Sin: CosSinArray;
    N, NU: INTEGER;
```

```
PROCEDURE amplitude* (c: COMPLEX): REAL;

VAR

    cRe, cIm: REAL;
    result: REAL;

BEGIN

    cRe := c.re;
    cIm := c.im;

    result := Math.Sqrt (cRe * cRe + cIm * cIm);

    RETURN result;

END amplitude;


PROCEDURE InitArrays ();

VAR

    bitset: SET;
    i, j, nu1: INTEGER;
    pi2N, arg: REAL;

BEGIN

    pi2N := 2 * Math.Pi () / N;
    i := N;
    REPEAT

        DEC (i);

        bitset := {};

        nu1 := 0;
        j := NU;
        REPEAT
            DEC (j);
            IF j IN BITS (i) THEN INCL (bitset, nu1) END;
            INC (nu1);
        UNTIL j = 0;

        Bitrev [i] := ORD (bitset);

        arg := pi2N * i;
        Cos [i] := Math.Cos (arg);
        Sin [i] := Math.Sin (arg);

    UNTIL i = 0;

END InitArrays;
```

```
PROCEDURE InitFFT (n : INTEGER);

VAR

    a: INTEGER;

BEGIN

    ASSERT (n >= 2);

    NU := 0;

    a := n;
    REPEAT
        ASSERT ((a MOD 2) = 0);
        a := a DIV 2;
        INC (NU);
    UNTIL a <= 1;

    N := n;

    NEW (Cos, N);
    ASSERT (Cos # NIL);

    NEW (Sin, N);
    ASSERT (Sin # NIL);

    NEW (Bitrev, N);
    ASSERT (Bitrev # NIL);

    InitArrays ();

END InitFFT;


PROCEDURE sort (H : FFTArray);

VAR

    i, l, k: INTEGER;
    temp: COMPLEX;

BEGIN

(* bit reversal: *)

    l := N - 1;

    k := 0;
    REPEAT

        INC (k);

        i := Bitrev [k];
        IF i > k THEN
            temp := H [k];
            H [k] := H [i];
            H [i] := temp;
        END;

    UNTIL k = l;

END sort;
```

```
(*
    FFT calculates the one-dimensional discrete fourier transformation H of a complex array h.
    The result is stored in the same array h because of efficiency reasons.
    The two halfs of the result are not exchanged, i.e. the DC-level of the fourier transform is contained in h [0] !
    res can be used as a scaling factor. If not used, set res to 1.
*)

PROCEDURE FFT* (h : FFTArray);

VAR

    i, l, k, kn2, n, n2, p: INTEGER;
    hc: COMPLEX;
    bitset: SET;
    cos, sin, tempRe, tempIm, hcre, hcim: REAL;

BEGIN

    ASSERT (h # NIL);

    n := LEN (h);
    IF N # n THEN InitFFT (n) END;

    h [0].re := 0.5 * (h [0].re + h [N-1].re);
    h [0].im := 0.5 * (h [0].im + h [N-1].im);

    n2 := N DIV 2;

    FOR l := NU - 1 TO 0 BY -1 DO

        bitset := {l};

        k := 0;
        WHILE k < N DO

            kn2 := k + n2;

            FOR i := 0 TO n2 - 1 DO

                p := Bitrev [k DIV ORD (bitset)];
                cos := Cos [p];
                sin := Sin [p];

                hc := h [kn2];
                hcre := hc.re;
                hcim := hc.im;
                tempRe := hcre * cos + hcim * sin;
                tempIm := hcim * cos - hcre * sin;

                hc := h [k];
                hcre := hc.re;
                hcim := hc.im;
                h [kn2].re := hcre - tempRe;
                h [kn2].im := hcim - tempIm;

                h [k].re := hcre + tempRe;
                h [k].im := hcim + tempIm;

                INC (k);
                INC (kn2);

            END;

            INC (k, n2);

        END;

        n2 := n2 DIV 2;

    END;

    sort (h);

END FFT;
```

```
PROCEDURE FFT2D* (h2: FFTArray2D);

VAR

    nx, ny: INTEGER;
    i, j: INTEGER;
    ht: FFTArray;

BEGIN

    ASSERT (h2 # NIL);
    ny := LEN (h2);
    ASSERT (ny >= 2);

    ASSERT (h2 [0] # NIL);
    nx := LEN (h2 [0]);
    ASSERT (nx >= 2);

    FOR j := 1 TO ny - 1 DO
        ASSERT (h2 [j] # NIL);
        ASSERT (LEN (h2 [j]) = nx);
    END;

    (* calc columns: *)

    FOR j := 0 TO ny - 1 DO FFT (h2 [j]) END;

    (* calc rows: *)

    NEW (ht, ny);
    ASSERT (ht # NIL);

    FOR i := 0 TO nx - 1 DO

        FOR j := 0 TO ny - 1 DO ht [j] := h2 [j, i] END;
        FFT (ht);
        FOR j := 0 TO ny - 1 DO h2 [j, i] := ht [j] END;

    END;

END FFT2D;
```

```
PROCEDURE ExampleCallCommand*;

VAR

    n, i, j: INTEGER;
    Example2DArray: FFTArray2D;

BEGIN

    (*
        Example calls, only for demonstration purposes
    *)

    n:= 8;

    NEW (Example2DArray, n);
    FOR j := 0 TO n - 1 DO
        NEW (Example2DArray [j], n);
        FOR i := 0 TO n-1 DO
            Example2DArray [j, i].re := Math.Sin (2 * Math.Pi () * i / n) * Math.Sin (4 * Math.Pi () * j / n) - 0.5;
            Example2DArray [j, i].im := 0;
        END;
    END;

    FFT2D (Example2DArray);

    FOR j := 0 TO n - 1 DO
        FOR i := 0 TO n - 1 DO
            Out.Real (amplitude (Example2DArray [j, i]), 20);
            Out.String (" ");
        END;
        Out.Ln;
    END;

END ExampleCallCommand;



BEGIN

    (*
        Initialisation of this module
    *)

    N := 0;
    NU := 0;
    Sin := NIL;
    Cos := NIL;
    Bitrev := NIL;

END FFT.
```

Result table of *Example2DArray* after the call of *ExampleCallCommand* (the four sub blocks have to be exchanged diagonally and row 0 and column 0 have been removed):

0,2	0,2	2,0	0,2	2,0	0,2	0,2
1,4	1,4	16,2	1,4	16,2	1,4	1,4
0,2	0,2	2,0	0,2	2,0	0,2	0,2
0,2	0,2	2,0	31,8	2,0	0,2	0,2
0,2	0,2	2,0	0,2	2,0	0,2	0,2
1,4	1,4	16,2	1,4	16,2	1,4	1,4
0,2	0,2	2,0	0,2	2,0	0,2	0,2