Programme de transformation de Fourrier directe (FFT)

Le signal 'entrée est présenté sous forme de tableau de N (ici N=512 mais il est facile de changer ce nombre qui doit toujours être une puissance entière de 2) Le spectre est un tableau de N nombres complexes Il faut donc d'abord définir les types nécessaires .

```
Type
      Cplx= ARRAY[0..1] of real;          { nombre complexe  en partie réele et
      imaginaire}

      Tablo = ARRAY[0..511] of real;   { pour le signal d'entrée}
      TabloCplx = ARRAY[0..511] of Cplx ;    {pour le spectre }
```

Puis définir une arithmétique complexe si elle n'est pas disponible dans la version du logiciel utilisé

```
Function Puissance(N,P:integer):integer;      { Calcul des puissance de 2 }
var
i:integer;
Begin
  result:=1;
  For i:=1 to P do result:=result*N;
end;
{=================================================================
      Arithmétique complexe
=================================================================}
 Function ADDCplx(A,B:Cplx):Cplx;     { additionne deux complexes}
 Begin
 Result[0]:=A[0]+B[0];
 Result[1]:=A[1]+B[1];
 End;
{-----------------------------------------------------------------------}
 Function SousCplx(A,B:Cplx):Cplx;    { soustrait deux complexes }
 Begin
 Result[0]:=A[0]-B[0];
 Result[1]:=A[1]-B[1];
 End;
 {-----------------------------------------------------------------------}
 Function MULRCplx(A:Real;B:Cplx):Cplx;    { Multiplie un complexe par un réel }
 Begin
  Result[0]:=B[0]*A;
 Result[1]:=B[1]*A;
 End;
{-----------------------------------------------------------------------}
 Function MulCplx(A,B:cplx):cplx;        { Multiplie deux complexes }
 Begin
 Result[0]:=A[0]*B[0]-A[1]*B[1];
 Result[1]:=A[0]*B[1]+A[1]*B[0];
 End;
```

```
{-------------------------------------------------------------------------}
 Function ModuleCplx(A:cplx):Real;    { Calcule le module d'un complexe }
 Begin
 Result:=sqrt(A[0]*A[0]+A[1]*A[1]);
 End;
 {-------------------------------------------------------------------------}
 Function PhaseCplx(A:cplx):real;       { La phase  }
 Begin
 Result:=(arctan(A[1]/A[0]))*180/PI;
 End;
{-------------------------------------------------------------------------}
 Function RealTocplx(A:real):cplx;      { met réel sous forme de complexe  A+j0 }
 Begin
 Result[0]:=A;
 Result[1]:=0;
 End;
{-------------------------------------------------------------------------}
 Function DivCplx(A,B:cplx):Cplx;       { Divise deux complexes }
 var
 D:real;
 Begin
 D:=B[0]*B[0]+B[1]*B[1];
 Result[0]:=(A[0]*B[0]+A[1]*B[1])/D ;
 Result[1]:= (A[1]*B[0]-A[0]*B[1])/D ;
 End;
{-------------------------------------------------------------------------}
 Function DivCplxR(A:Cplx;B:Real):Cplx;      { Divise un complexe par un réel }
 Begin
 Result[0]:=A[0]/B;
 Result[1]:=A[1]/B;
 End;
{-------------------------------------------------------------------------}
Procedure ConstructionW;      { Construit le tableau des coefficients complexes
des papillons  FFT }
Var
 i:integer;
 Begin
  For i:=0 to 511 do  Begin
                      W[i,0]:=cos(2*PI*i/512);
                      W[i,1]:=-sin(2*PI*i/512);
                     End;
 End;
```

I l faut comme pour la transformée de Walsh effectuer un réarrangement (renversement digital) du tableau d'entrée TB

```
{-------------------------------------------------------------------------------------------------}
Procedure Renversement( Var TB:tablo);
{ Renversement digital  du tableau  de réels [TB]pour FFT et Walsh   }
Var
```

```
I,J,L,N:integer;
TT:real;
Begin
  Begin
    J:=1;
    N:=512;
    For i:=1 to N-1 do
      Begin
          If j>=i then
                Begin
              TT:=TB[j-1];
              TB[j-1]:=TB[i-1];
              TB[i-1]:=TT;
                end;
           L:= N div 2;
          While J>L do
          begin
              J:=J-L;
              L:=L div 2;
          end;
          J:=J+L;
        end;
  end;
end;
```

Puis les papillons de FFT qui fournissent le spectre complexe final .

```
{--------------------------------------------------------------------------}
Procedure FFT(Signal:Tablo;Var FFTSp:TabloCplx);
Var
   Puiss: ARRAY[0..10] of integer;
   P:cplx;
   M,NbrPtr,LNPtr,T,I,J,K,TR,NT,I0,INV:integer;
   SCplx:TabloCplx;
Begin
 ConstructionW;      { Le tableau de coefficients }
  LnPtr:=9;
  Puiss[0]:=1;
  For i:=1 to 10 do Puiss[i]:=Puiss[i-1]*2;
  For i:=0 to 511 do
                Begin
                  SCplx[i]:=RealToCplx(Signal[i]);
                End;
   For T:= (LNptr-1) DownTo 0 do
      Begin
       TR:=Puiss[LNPtr-T];
       NT:=Puiss[T];
       For j:=0 to (NT-1) do
           Begin
             I0:=J*TR;
```

```
            if TR>2 then
                Begin
                  INV:=TR div 2;
                  For k:= INV to TR-1 do  Begin
                                    M:=NT*(k-(TR div 2));
                        Scplx[I0+K]:=MulCplx(Scplx[I0+K],W[M]);
                                     End;
                end;
              For K:=0 to ((TR div 2)-1) do
                Begin
          p:=SCplx[I0+K];
          SCplx[I0+K]:=AddCplx(SCplx[I0+K],Scplx[I0+K+(TR div 2)]);
          Scplx[I0+K+(TR div 2)]:=SousCplx(p,Scplx[I0+K+(TR div 2)]);
                End;
          End;
     End;
     NbrPtr:=Puiss[LNPtr];
    For i:=0 to NbrPtr-1 do   FFTsp[i]:=DivCplxR(Scplx[i],Nbrptr);
End;
```