```
/*
Ce code calcul le transformé discret de Fourrier d'une série de 50 000 nombres
en utilisant des float et des double. Sur mon P4 2.4GHz avec 768Mo de RAM le
calcul prend 399.5 secondes (avec des variations d'un dixième de seconde selon
les exécution, mais dans les deux sens pour les deux calculs) que l'on utilise
des double ou des float.

je précise juste que le code du dft n'est pas de moi mais vient de là :
http://www.immersive.com/marc/dft80x87.html

On peut peut-être en conclure qu'il n'y a pas de gain de temps à utiliser des float plutot
que des double, mais je ne suis pas sûr d'être qualifié pour faire cette analyse.
*/

#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
#include <conio.h>


#define NB 50000
#define TYPE_DOUBLE
#define PRINT

#ifndef TYPE_DOUBLE
#define TYPE_FLOAT
#endif

#ifdef TYPE_DOUBLE
#define D_PREC  // For double-precision (double) arithmetic
#else REMOVE
#define S_PREC  // For single-precision (float) arithmetic
#endif

#if defined(S_PREC) && defined(D_PREC)
# error S_PREC and D_PREC cannot both be defined
#endif

/* If single precision, Real corresponds to a 4-byte floating-point
 * quantity, and FWORD is defined as 'dword' for the assembly language
 * routines.
 * If double precision, Real corresponds to an 8-byte floating-point
 * quantity, and FWORD is defined as 'qword' for the assembly language
 * routines.
 */
#ifdef  S_PREC
typedef float   Real;
#define FWORD   dword
#define FWSIZE  (4)
#else
typedef double Real;
#define FWORD   qword
#define FWSIZE  (8)
#endif

/* "Whole" must be typedef'd to a 32-bit quantity */
typedef long Whole;


void dft(Real t[], Real f[], Whole serlen)
{
    __asm {
        mov edi, f          // Init freq series ptr
        fldpi
        fadd    st, st(0)       // Put 2*pi on the stack
        fild    dword ptr serlen    // Put series length on stack
        fdivp   st(1), st       // Leave radians per sample in st(0)
        fldz                // Push 1st angle accum
        mov ebx, serlen     // ebx is loop counter for freq series
    outer:
        mov esi, t          // Point at time series
        fldz                // Push the real freq component
        fldz                // Push the imaginary freq component
        fldz                // Push 2nd angle accum
        mov ecx, serlen     // Init inner loop count
```

```
    inner:
        fld st(0)           // Push copy of 2nd angle accum
        fcos                // Get cos(angle)
        fmul    FWORD ptr [esi]     // Multiply by t[j]
        faddp   st(3), st       // Add cos(angle)*t[j] to real freq component
        fld st(0)
        fsin
        fmul    FWORD ptr [esi]
        faddp   st(2), st       // Add sin(angle)*t[j] to imaginary freq component
        fadd    st, st(3)       // Increment 2nd angle accum
        add esi, FWSIZE     // Increment time series pointer
        sub ecx, 1
        jne inner
        fstp    st(0)           // Drop the 2nd angle accum
        fild    dword ptr serlen
        fdivp   st(1), st
        fstp    FWORD ptr [edi+FWSIZE]
        fild    dword ptr serlen
        fdivp   st(1), st
        fstp    FWORD ptr [edi]
        fadd    st, st(1)       // Increment the 1st angle accum
        add edi, FWSIZE*2
        sub ebx, 1
        jne outer
        fstp    st(0)           // Pop 1st angle accum
        fstp    st(0)           // Pop radians per sample
        fwait
    }
}

void main()
{
  __int64 Freq,Start,End;
  double duree;
  Real t[NB],f[NB];
  Whole qt=NB-1;
  int i;
  QueryPerformanceFrequency((LARGE_INTEGER*)&Freq);
  QueryPerformanceCounter((LARGE_INTEGER*)&Start);
  srand((unsigned int)(Start&0xffffffff));
  for(i=0;i<NB;i++)
  {
    t[i]=(Real)rand()/RAND_MAX*100.0+.1;
    f[i]=(Real)rand()/RAND_MAX*100.0+.1;
#ifdef PRINT
    printf("%g %g\n",t[i],f[i]);
#endif
  }
  SetThreadPriority(GetCurrentThread(),THREAD_PRIORITY_HIGHEST);
  QueryPerformanceCounter((LARGE_INTEGER*)&Start);
  dft(t,f,qt);
  QueryPerformanceCounter((LARGE_INTEGER*)&End);
  SetThreadPriority(GetCurrentThread(),THREAD_PRIORITY_NORMAL);
  duree=(double)(End-Start)/(double)Freq;
  printf("\n");
#ifdef PRINT
  for (i=0;i<NB;i++)
    printf("%g %g\n",t[i],f[i]);
#endif
  printf("\n%g",duree);
  getch();
}
```