Лабораторна робота № 4
ШВИДКОГО ПЕРЕТВОРЕННЯ ФУР'Є (ШПФ, FFT)

Завдання: Реалізувати пряме та обернене дискретне перетворення Фур'є та на мові програмування С++.

Програма на С для двійкового ШПФ з часовою децимації

Наведена нижче програма обчислення ШПФ являє собою реалізацію на С двійкового ШПФ з часовою децимацією. Програма обчислює ДПФ і ОДПФ дискретної в часі послідовності згідно з визначенням. Програма складається з головної функції `dftf.c` і трьох функцій: `fft()`, `read_data()` і `save_data()`. Як і при обчисленні безпосередньо ДПФ, всі функції записані в окремих файлах і об'єднуються при компіляції за допомогою виклику (`include`) з головної функції. Функції `read_data()` і `save_data()` використовуються для читання даних і запису перетворених даних у файл. Ці два файли ідентичні файлам, які застосовувалися для прямого обчислення ДПФ. Головна програма `dftf.c` і функція `fft()` наведені в програмах 1 і 2 відповідно.

Використання до будь-якого прикладу програми для обчислення ШПФ дає такі ж результати, що і програма для прямого обчислення ДПФ. Довести це твердження пропонується студентам самостійно.

Програма 1. Основна функція `dftf.c` для обчислення ДПФ за допомогою ШПФ з часовою децимацією

```
/*-------------------------------------------*/
/* Програма для розахунку коефіцієнтів ДПФ ЧД */
/*         з допомогою ШПФ з ЧД               */
/*    використовуємо 3 інші функції           */
/*-------------------------------------------*/
#include "dspl.h"
#include "dft.h"
main()
{
    extern long npt;
    extern int inv;
    printf("виберіть тип перетворення\n");
    printf("\n");
    printf("0 для прямого ДПФ\n");
    printf("1 для оберненого ДПФ\n");
    scanf("%d",&inv);
    read_data();
    fft();
    save_data();
    exit();
}
#include "fft.c";
#include "rdata.c";
#include "sdata.c";
```

Програма 2. Реалізація на мові С двійкового алгоритма ШПФ з часовою децимацією.

```
/*-------------------------------------------------*/
```

```
/*               файл fft.c                                  */
/*     функція розраховує ДПФ послідовності                 */
/*        з допомогою двійкового ШПФ                        */
/*---------------------------------------------------------*/
void fft()
{
    int sign;
    long m,irem,l,le,le1,k,ip,i,j;
    double ur,ui,wr,wi,tr,ti,temp;
    extern long npt;
    extern complex x[size];
/* перестановка із заміщенням даних, що обумовлена
оберненим порядком бітів */
    j=1;
    for(i=1;i<npt;++i) {
        if(i<j) {
            tr=x[j].real;ti=x[j].imag;
            x[j].real=x[i].real;
            x[j].imag=x[i].imag;
            x[i].real=tr;x[i].imag=ti;
            k=npt/2;
            while(k<j) {
                j=j-k;
                k=k/2;
                }
            }
            else{
                k=npt/2;
                while(k<j)
                {
                    j=j-k;
                    k=k/2;
                }
                }
            j=j+k;
    }
    /* рахуємо кількість каскадів: m=log2(npt) і обираємо ШПФ
    або ОШПФ */
        m=0;irem=npt;
        while(irem>1) {
            irem=irem/2;
            m=m+1;
        }
        if(inv==1)
            sign=1;
        else
            sign=-1;
    /* рахуємо ШПФ для кожного каскаду*/
        for(l=1;l<=m,++l) {
            le=pow(2,l);
            le1=le/2;
            ur=1.0; ui=0;
            wr=cos(pi/le1);
            wi=sign*sin(pi/le1);
            for(j=1;j<=le1;++j){
                i=j;
                while(i<=npt){
```

```
                    ip=i+le1;
                    tr=x[ip].real*ur-x[ip].imag*ui;
                    ti=x[ip].imag*ur+x[ip].real*ui;
                    x[ip].real=x[i].real-tr;
                    x[ip].imag=x[i].imag-ti;
                    x[i].real=x[i].real+tr;
                    x[i].imag=x[i].imag+ti;
                    i=i+le;
                }
                temp=ur*wr-ui*wi;
                ui=ui*wr+ur*wi;
                ur=temp;
            }
        }
/* якщо потрібно знайти ШПФ, то кожен коефіцієнт ділимо на на npt */
        if(inv=-1) {
            for(i=1;i<=npt;++i) {
               x[i].real=x[i].real/npt;
               x[i].imag=x[i].imag/npt;
        }
    }
}
```

Задача 1. Скористайтеся програмою прямого обчислення ШПФ і знайдіть коефіцієнти такої дискретної в часі послідовності:

$$x(n)=\{5, 7, 3, 3, 0, 0, 15, 32, 13, 7\}$$

Програма 3. Ще один спосіб обчислення ШПФ

```
/*-----------------------------------------*/
/*              Лістинг програми ШПФ       */
/*-----------------------------------------*/
#include <stdio.h>
#include <math.h>
#include <time.h>
SHPF(x,y,N,I)           /*Процедура ШПФ */
register float *x,*y;   /*x,y-вхідні масиви даних*/
register int N,I;      /*розмірністю I=1 для ШПФ  I=-1 для ОШПФ */
{
    register float c,s,t1,t2,t3,t4,u1,u2,u3;
    register int i,j,p,l,L,M,M1,K;
    L=N;
    M=N/2;
    M1=N-1;
    while(L>=2){
    l=L/2; u1=1.; u2=0.; t1=PI/(float)l;
    c=cos(t1); s=(-1)*I*sin(t1);
    for(j=0; j<l;j++)
    {
        for(i=j;i<N;i+=L)
        {
            p=i+l;
            t1=*(x+i)+*(x+p);
            t2=*(y+i)+*(y+p);
            t3=*(x+i)-*(x+p);
            t4=*(y+i)-*(y+p);
```

```
            *(x+p)=t3*u1-t4*u2;
            *(y+p)=t4*u1+t3*u2;
            *(x+i)=t1; *(y+i)=t2;
        }
        u3=u1*c-u2*s;
    u2=u2*c+u1*s; u1=u3;
}
L/=2;
}
j=0;
for(i=0;i<M1;i++)
{
    if(i>j)
    {
        t1=*(x+j); t2=*(y+j);
        *(x+j)=*(x+i); *(y+j)=*(y+i);
        *(x+i)=t1; *(y+i)=t2;
    }
    K=M;
    while(j >=K)
    {
        j-=K;K/=2;
    }
    j+=K;
    }
}
sinsignal(PiFiAiN) /*моделювання вхідного сигналу*/
                    /*в формі синусоїди/*
float *P,F,A;      /*P-масив сигналу розмірності N*/
int N;             /*F-частота сигналу, */
                   /*A-амплітуда сигналу*/
{
    register int i;
    register float r,re,re1,im,im1;
        re=cos(2.*PI*F/(float)N);
        im=sin(2.*PI*F/(float)N);
        re1=A;im1=0.;
        for(i=0;i< N;i++)
            {
                *(P+i)=re1;r=re1;
                re1=r*re-im1*im;
                im1=im1*re+r*im;
            }
}
main()
{
    int j,N;
    float *x,*y,F,A,Re,Im;
    printf("\t\t N :"); scanf("%d",&N);
    printf("\t\t F(gc):"); scanf("%f",&F);
    printf("\t\t A :"); scanf("%f",&A);
        x=(float*)calloc(N,sizeof(float));
        y=(float*)calloc(N,sizeof(float));

        sinsignal(x,F,A,N);

        for(j=0;j < N;j++) printf(" X[%d] - %.1f \n",j,*(x+j));
```

```
        SHPF(x,y,N,1);

        for(j=0;j < N/2;j++)
        {
            Re=*(x+j);
            Im=*(y+j);
            A=2.*sqrt(Re*Re+Im*Im)/(float)N;
            printf(" X[%d] - %d \n",j,(int)A);

        free(x); free(y);
}

Тестовий приклад
N = 1024
F = 100
A = 100

Вхідний масив :
X[0]= 100
X[1]= 81.8
X[2]= 33.7
X[3]= - 26.7
X[4]= - 77.7
X[5]= - 99.7

... ... ... ... ...
X[1018]= - 85.8
X[1019]= - 99.7
X[1020]= - 77.3
X[1021]= - 26.7
X[1022]= 33.7
X[1023]= 81.8

Вихідний масив:
X[0]= 0
X[1]= 0
X[2]= 0
X[3]= 0
X[4]= 0

...... ... ... ...
X[100]= 100

... ... ... ... ...
X[508]= 0
X[507]= 0
X[508]= 0
X[509]= 0
X[510]= 0
X[511]= 0
```

Cooley J.W, and Tukey J.W. (1965) An algorithm for the machine calculation of complex Fourier series. Mathe¬matics Computation, 19 (90), April, 297–301.
IEEE (1979) Programs for Digital Signal Processing. New York: IEEE Press.