Решение задачи 1

Вариант программы декобирования кода Хемминга. На вход программы поступает произволная последовательность из ‘0’ и ‘1’, на выходе программа возвращает декодированную последовательность из ‘0’ и ‘1’, причем если в исходной последовательности была допущена однократная ошибка, прогрмма ее исправляет и сообщает номер битого символа во входной последовательности.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
unsigned int maskf[] = {0x1,0x2
,0x4,0x8,0x10,0x20,0x40,0x80,0x100,0x200,0x400,0x800,0x1000,0x2000,0x4000,0x8000
,0x10000,0x20000,0x40000,0x80000,0x100000,0x200000,0x400000,0x800000,0x1000000};
void CharToBinaryArray(int &a, int l, char arr[]);
void AddControlBit(char arr[], int l, char arrc[], int lc);
int order(int a);//кол-во разрядов в двоичном представлении
void matrpreobr(char **preobr, int lrow, int lcol);
void decoder_hamming(char **preobr, int lrow, int lcol, char earr[], char
darr[]);
void cutcontrolbit(int lcol, char darr[], int l, char arr[]);
void HammingDecoderArr(char *esignal, int l_esignal, char *&signal, int
&l_signal);
void main(int argc, char* argv[]){
//---------------Decoder---------------------------------------//
        int l_esignal = strlen(argv[1]);//Длина входной последовательности
        int l_signal;
        char*esignal;
        char*signal = NULL;
        esignal = (char*)calloc(l_esignal+1,sizeof(char));
        strcpy(esignal,argv[1]);
        printf("esignal = %s     length = %d\n",esignal,l_esignal);
        HammingDecoderArr(esignal, l_esignal, signal, l_signal);
        printf(" signal = %s     length = %d\n",signal,l_signal);
//-------------------------------------------------------------//
}
void HammingDecoderArr(char *esignal, int l_esignal, char *&signal, int
&l_signal){
        int lrow = order(l_esignal);
        l_signal = l_esignal - lrow;
        signal = (char*)calloc(l_signal,sizeof(char));
        char *darr = new char[l_esignal];
        char **preobr = new char*[lrow];
        for(int i=0; i<lrow ; i++)
                preobr[i] = new char[l_esignal];
        matrpreobr(preobr, lrow, l_esignal);//Формируемматрицупреобразования
        decoder_hamming(preobr, lrow, l_esignal, esignal,
darr);//Декодированныйсигнал (исправленыошибки)
        cutcontrolbit(l_esignal, darr, l_signal, signal);
        delete [] darr;
        for(int i=0; i<lrow; i++)
                delete [] preobr[i];
        delete [] preobr;
}
void decoder_hamming(char **preobr, int lrow, int lcol, char earr[], char
darr[]){
        int k=0;
        strcpy(darr,earr);
        for(int i=0; i<lrow; i++){
                int sum =0;
                for(int j=0; j<lcol; j++)
                        sum += (preobr[i][j] - '0') * (earr[j] - '0');
                sum = sum%2;
```

```
                k += sum*(1<<i);
        }
        darr[k-1] = (darr[k-1]=='0') ? '1' : '0';
        printf("k=%d\n",k);
}
void matrpreobr(char **preobr, int lrow, int lcol){
        int i2=0,v;
        char arr[100];
        for(int i=1; i<=lcol; i++){//циклпостолбцам
                if((i&(i-1)) == 0){//степенидвойки
                        i2++;
                        v = 1<<(i2-1);
                        CharToBinaryArray(v, lrow, arr);
                        for(int j=0;j<lrow;j++) //циклпострокам
                                preobr[j][i-1] = arr[j];
                }
                else{//номераинформационныхбитов
                        CharToBinaryArray(i, lrow, arr);
                        for(int j=0;j<lrow;j++) //циклпострокам
                                preobr[j][i-1] = arr[j];
                }
        }

        for(int j=0;j<lrow;j++)
                preobr[j][lcol] = 0;
}
void cutcontrolbit(int lcol, char darr[], int l, char arr[]){
        int j=0,i=1;
        while(i<=lcol){
                if(i&(i-1))
                        arr[j++] = darr[i-1];
                i++;
        }
        arr[j]=0;
}
void AddControlBit(char arr[], int l, char arrc[], int lc){
        int j=0,i=1;
        while(j<l){
                if((i&(i-1)) == 0)
                        arrc[i-1] = '0';
                else
                        arrc[i-1] = arr[j++];
                i++;
        }
        arrc[i-1]=0;
}
int order(int a){
        double x = (double)a;
        x = log(x)/log(2.0);
        return (int)x + 1;
}
void CharToBinaryArray(int &a, int l, char *arr){
        for(int i=0;i<l;i++)
                arr[i] = a & maskf[i] ? '1' : '0';
        arr[l]=0;
}
```

Пример программы, определяющий период во входной последовательности и среднеквадратичное отклонение периода от среднего, дополнительно решаются следующие задачи: определяется корреляционная функция стабильного сигнала (с постоянным периодом) и измеренным на макете, определяется отношение квадарата невязки между модельным и измеренным сигналом к енергии

модельного сигнала (т.е. определяется относительная энергия невязки между сигналами). Входные данные для программы содержатся в конфигурационном файле, имеющего следующий формат: кол-во последовательностей сигналов, длина кодовой последовательности (известна из предыдущей задачи), форма кодовой последовательности (последовательность из ‘0’ и ’1’), длительности переходных процессов (длительность переднего/заднего фронта сигнала, длительность смены фазы – переход от ‘0’ к ‘1’ и наоборот) – вводится для решения задачи в более общем случае, когда время переключяения фазы не бесконечно быстрое. На выходе программа возвращает для каждого канала средний период и СКО отклонения от среднего периода.

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include "Gears.h"
#define PI (3.14159265358979323846264338327795)
//Инициализация массивов
void GearsSignal::InitArray(){
        TP =
(double*)calloc(ngear,sizeof(double));//Периодсигналадлякаждойшестерни
        direction = (char*)calloc(ngear,sizeof(char));//Направлениевращения
        DT = (double*)calloc(ngear,sizeof(double));//Сдвигповремени
        Amp = (double*)calloc(ngear,sizeof(double));//Амплитудакор-функции
        Err = (double*)calloc(ngear,sizeof(double));//Ошибка
}
GearsSignal::GearsSignal(){
        ngear = 3;
        InitArray();
}
//Чтение конфигурационного файла
GearsSignal::GearsSignal(char *nameconfig){
        FILE*conf;
        conf = fopen(nameconfig,"r");//Работаемсконфигурационнымфайлом
        fscanf(conf,"%d",&ngear);//Строка №1 - Длина кодовой последовательности
        kd = (int*)calloc(ngear,sizeof(int));
        kod = (char**)calloc(ngear,sizeof(char*));
        InitArray();
        namefile = (char*)calloc(1000,sizeof(char));
        fscanf(conf,"%s",namefile);//Строка №2 имяфайласизмерениями
        for(int i=0;i<ngear;i++){
                fscanf(conf,"%d",&kd[i]);//Строка №3 - Длина кодовой
последовательности
                kod[i] = (char*)calloc(kd[i]+1,sizeof(char));
                fscanf(conf,"%s",kod[i]);//Строка №4 - форма кодовой
последовательности
                printf("kd[%d]=%d\n",i,kd[i]);
                printf("kod[%d]=%s\n",i,kod[i]);
        }
        fscanf(conf,"%lf",&tau1);//Строка №5 - длительность переднего фронта
(милисек)
        fscanf(conf,"%lf",&tau2);//Строка №6 - длительность смены фазы (милисек)
        fscanf(conf,"%lf",&tau3);//Строка №7 - длительность заднего фронта
(милисек)
        fscanf(conf,"%lf",&dt);//Строка №8 - время между отсчетами (милисек)
        fclose(conf);
}
//Анализ каналов – опредление энергии невязки между модельным и измеренным
сигналами
void GearsSignal::GearsAnaliz(){
        if(ReadFile(namefile)){
                CalcPeriod();
                SignalModel();
                CorAnaliz2('w');
                PrintArray();
```

```
                EnergyError();
        }
        else
                printf("analiz disable!\n");
}
void GearsSignal::EnergyError(){
        double dE,E,E1,E2;
        for(int k=0; k<ngear; k++){
                int i = (int)DT[k];//сдвиг
                int n = direction[k];//направление вращения
                dE = 0.0;//интеграл ошибок
                E1 = E2 = E  = 0.0;
                for(int j=0; j<I0[k]; j++) //Интегреируем
                        if(i+j>=0 && i+j<I){
                                E1 += A0[n][j] * A0[n][j];
                                E2 += A[k+1][i+j] * A[k+1][i+j];
                                E  += A0[n][j] * A[k+1][i+j];
                                dE += (A[k+1][i+j] - A0[n][j]) * (A[k+1][i+j] -
A0[n][j]);
                        }
                if(I0[k]>0){
                        Err[k] = 100.0*dE/E1;//Енергияневязки
                }
                else
                        Err[k] = -100.0;
        }
        for(int i=0;i<ngear;i++) printf("Err[%d]=%lf\n",i,Err[i]);
}
void GearsSignal::CorAnaliz2(char rw){
        double dtime[2],amp[2];
        double **akf;
        akf = (double**)calloc(ngear,sizeof(double*));
//      for(int i = 0; i<ngear; i++)
//              akf[i] = (double*)calloc(I,sizeof(double));
        for(int i = 0; i<ngear; i++){
                if(I0[i]>0){
                        akf[i] = CorAnaliz(A0[2*i],   I0[i], A[i+1], I, dtime[0],
amp[0], 'r');
                        free(akf[i]);
                        akf[i] = CorAnaliz(A0[2*i+1], I0[i], A[i+1], I, dtime[1],
amp[1], 'r');
                        free(akf[i]);
                        if(amp[0]>amp[1]){
                                printf("direction[%d]=forvard!\n",i);
                                DT[i] = dtime[0];
                                Amp[i] = amp[0];
                                direction[i] = 2*i;
                                akf[i] = CorAnaliz(A0[2*i],   I0[i], A[i+1], I,
dtime[0], amp[0], rw);
                        }
                        else{
                                printf("direction[%d]=back!\n",i);
                                DT[i] = dtime[1];
                                Amp[i] = amp[1];
                                direction[i] = 2*i+1;
                                akf[i] = CorAnaliz(A0[2*i+1], I0[i], A[i+1], I,
dtime[1], amp[1], rw);
                        }
                }
        }
        FILE*rez;
        rez=fopen("coranaliz_signal.dat","w");
        for(int j=0;j<I;j++){
                fprintf(rez,"%lf       ",dt*(double)j);
```

```
                for(int i = 0; i<ngear; i++){
                        fprintf(rez,"%lf       ",akf[i][j]);
                }
                fprintf(rez,"\n");
        }
        fclose(rez);
}
double* GearsSignal::CorAnaliz(double*&A1, int I1, double*&A2, int I2, double
&DT, double &AmpAkf, char rw){
        FILE*rez;
        if(rw == 'w'){
                rez=fopen("coranaliz_test_signal.dat","w");
                fprintf(rez,"dI E\n");
        }

        int i,imax;
        double E,E1,E2;
        double Emax=0.0;
        double *akf;
        akf = (double*)calloc(I2,sizeof(double));
        for(i=0;i<I2;i++){//Сдвиг первого сигнала относительно второго
                E1=E2=E=0.0;
                for(int j=0; j<I1; j++) //Интегреируем по разверте второго
сигнала
                        if(i+j>=0 && i+j<I2){
                                double v1 = A1[j];
                                double v2 = A2[i+j];
                                E1 += v1 * v1;
                                E2 += v2 * v2;
                                E  += v1 * v2;
                        }
                akf[i] = E;
                if(rw == 'w')
                        fprintf(rez,"%d%lf\n",i,E);
                if(E>Emax){
                        Emax=E;
                        imax=i;
                }
        }
        if(rw == 'w')
                fclose(rez);

        DT = (double)imax;
        AmpAkf = Emax;
        return akf;
}
void GearsSignal::control_period(){
        if(ReadFile(namefile))
                CalcPeriod();//Средний период для всех шестерней
        int k=1;
        int i=1,i2=0;
        char **shortkod;
        shortkod = (char**)calloc(ngear,sizeof(char*));
        for(int k=0;k<ngear;k++){
                shortkod[k] = (char*)calloc(kd[k]+1,sizeof(char));
        }
        for(int k=0;k<ngear;k++){
                int l=0;
                shortkod[k][l++] = kod[k][0];
                for(i=1;i<kd[k];i++){
                        if(kod[k][i]!=kod[k][i-1])
                                shortkod[k][l++]=kod[k][i];
                }
                shortkod[k][l]=0;
```

```
        }
        for(k=1;k<=3;k++){
                i=1,i2=0;
                while(A[k][i]*A[k][i-1]>0.0)
                        i++;
                i2=i;
                i++;
                int n=0,N=0;
                double per[1000]= {0.0};
                double Period,cko;
                while(i<I){
                        while(A[k][i]*A[k][i-1]>0.0 && i<I)
                                i++;
                        if(i>=I)
                                break;
                        n++;
                        if(n==strlen(shortkod[k-1])){
                                per[N] = (double)(i-i2);
                                i2 = i;
                                n  = 0;
                                N++;
                        }
                        i++;
                }
                Period = 0.0;
                for(int i=0; i<N; i++)
                        Period += per[i];
                Period = Period / ((double)N);
                cko = 0.0;
                for(int i=0; i<N; i++)
                        cko += (per[i] - Period)*(per[i] - Period);
                cko = sqrt(cko/((double)N));
                printf("Kanal=%d Numder period=%d Period=%1.1lf
CKO=%1.1lf\n",k,N,Period,cko);
        }
}
void GearsSignal::SignalModel(){//Модельсигнала - одинпериод
        int Jmax = 0;
        double Tadd = 0.0;//Уширение/сужениеимпульса
        I0 = (int*)calloc(ngear,sizeof(int));
        A0 = (double**)calloc(2*ngear,sizeof(double*));
        for(int j=0; j<ngear; j++){
                if(TP[j]>1.0){
        IzlImpAdd2(kd[j],kod[j],dt,Tadd,tau1,tau2,tau3,TP[j],A0[2*j],I0[j]);
                        A0[2*j+1] = (double*)calloc(I0[j],sizeof(double));
                        for(int i=0; i<I0[j]; i++)
                                A0[2*j+1][i] = A0[2*j][I0[j]-i-1];
                        if(I0[j]>Jmax)
                                Jmax = I0[j];
                }
                else
                        I0[j] = 0;
        }
        FILE*rez;
        rez= fopen("model_signal.dat","w");
        for(int i=0;i<Jmax;i++){
                fprintf(rez,"%lf       ",dt*(float)i);
                for(int j=0; j<ngear; j++){
                        if(i<I0[j])
                                fprintf(rez,"%lf       %lf
        ",A0[2*j][i],A0[2*j+1][i]);
                        else
                                fprintf(rez,"%lf       %lf     ",0.0,0.0);
                }
```

```
                fprintf(rez,"\n");
        }
        fclose(rez);
}
void GearsSignal::IzlImpAdd2(int kd,char *kod,double dt,double Tadd,double
tau1,double tau2,double tau3,double T,double *&A0,int &J0){
        int i,j,J,K3,k0,j0,jb;//Длительность сигнала в отсчетах
        int Np;//Длительность реализации в отсчетах - степень двойки
        int ll,l,m,m1,m2,m3,ml,k;
        int km[2][300];
        double amp,max,freq,t,tau,td,tl,Tl,x,y,ct,cT,T0;
        double Amp,st,sT,tc,E0;
        double *Xt;
        char c;
        ll=32000;
        Xt=(double *)calloc(ll,sizeof(double));
        k0=(int)((double)(T)/dt);//Длительность плоской вершины изл. имп.
        T0=T+0.0;
        T+=Tadd;//
//-------------------------------------------------------------//
        ll=0; m=0; m1=0;
        for(i=0;i<kd;i++){//Разбераем строку на отдельные чипы
                c=kod[i];
                k=atoi(&c);
                if(k==0) k=-1;
                if((i>0)&&(m1!=k)){
                        km[1][ll]=i-m;//Число элементарных элементов в чипе
                        km[0][ll]=m1;//Фазачипа
                        m=i;
                        ll++;
                }
                m1=k;
        }
        if(kd==1) {km[0][ll]=1; km[1][ll]=1;}
        else{km[0][ll]=k;       km[1][ll]=kd-m;}
//-------------------------------------------------------------//
        if(tau1<=tau3) tau=tau1;
        else    tau=tau3;
        if(tau>tau2/2) tau=tau2/2;//Ищемсамыйкороткийфронт
//---------------Формируем излученный импульс----------------//
        td=0.5;
        if(td>dt) td=dt;
        tl=T0/(double)(kd);//Длительность чипа в in mks
        tl=(double)((int)(tl/td))*td;//Длительность чипа Д/Б кратна шагу
        T0=(double)(kd)*tl;//Новая длительность импульса

        k=(int)(T0/td);
        j=0;
//----Создаем базу для фазаманипулированного импульса--------//
//----Т.е. огибающую на основе плоского импульса-------------//
        for(t=0;t<=T;t+=td){//Текущая длительность в mks
                j++;
//-------------------------------------------------------------//
                if(tau1+tau3<=T){//Импульс не фазаманипулированный и длиньше
длительности фронтов
                        if(t<=tau1){//Строим передний фронт когда все нормально
                                Xt[j]=(1.0-cos(PI*t/tau1))/2.0;
                        }
                        if((t>tau1)&&(t<=T-tau3)){
                                Xt[j]=1.0;
                        }
                        if(t>T-tau3){
                                Xt[j]=(1.0+cos(PI*(t-(T-tau3))/tau3))/2.0;
                        }
```

```
                }
//-------------------------------------------------------------//
                else{//Импульс не фазаманипулированный и короче длительности
фронтов
                        tau=T*tau1/(tau1+tau3);//Место встречи фронтов
                        if(t<=tau){
                                Xt[j]=(1.0-cos(PI*t/tau1))/2.0;
                        }
                        else{
                                Xt[j]=(1.0+cos(PI*(t-(T-tau3))/tau3))/2.0;
                        }
                }
//-------------------------------------------------------------//
        }
        jb=j;
//-------------------------------------------------------------//
//----Теперь наполняем импульс фазавой манипуляцией----------//
        if(kd>1){
                j=0;
                j=(int)(tau1/2/td);
                for(i=0;i<=ll;i++){//Циклпочипам
                        Tl=tl*(double)(km[1][i]);
                        amp=(double)(km[0][i]);
//-------------------------------------------------------------//
                        for(t=0;t<=Tl-td+0.000000001;t+=td){//Циклвнутричипа
                                j++;
                                if(j<=k){
                                        x=Xt[j];
//-------------------------------------------------------------//
                                        if(Tl>=tau2){//Если длительность чипа
больше фронта
                                                if(t<=tau2/2){//Переднийфронт
                                                        if(i>0){

        Xt[j]*=amp*sin(PI*t/tau2);//Форма фронта в момент переброса фазы в А-
квадратуре
                                                        }
                                                }
                                                if((t>tau2/2)&&(t<=Tl-tau2/2)){
                                                        Xt[j]*=amp;
        //Yt[j]=0;
                                                }
                                                if(t>Tl-tau2/2){//Заднийфронт
                                                        if(i<ll){

        Xt[j]*=amp*sin(PI*(t-(Tl-tau2))/tau2);//Форма фронта в момент переброса
фазы в А-квадратуре
                                                        }
                                                        if(i==ll){
                                                                Xt[j]*=amp;
                                                        }
                                                }
                                        }
//-------------------------------------------------------------//
                                        else{
                                                tau=Tl/2.0;//Место встречи фронтов
                                                if(t<=tau){
                                                        if(i>0){

        Xt[j]*=amp*sin(PI*t/tau2);//Форма фронта в момент переброса фазы в А-
квадратуре
                                                        }
                                                }
                                                else{
```

```
                                                        if(i<ll){

        Xt[j]*=amp*sin(PI*(t-(Tl-tau2))/tau2);//Форма фронта в момент переброса
фазы в А-квадратуре
                                                        }
                                                        if(i==ll){
                                                                Xt[j]*=amp;
                                                        }
                                                }
                                        }
//------------------------------------------------------------//
                                }//if(j<=k){
                                else{
                                        Xt[j]*=amp;
                                }
                        }//for(t=0;t<=Tl;t+=td){//Циклвнутричипа
                }//for(i=0;i<=ll;i++){//Циклпочипам
//------------------------------------------------------------//
                for(i=j+1;i<=jb;i++){//Циклвнутричипа
                        Xt[i]*=amp;
                }
//------------------------------------------------------------//
        }//if(kd>1){
//------------------------------------------------------------//
//----Теперь сплайнируем сигнал на отсчеты через dt----------//
        l=0;
        for(t=0;t<=T;t+=dt)
                l++;
        J0=l;//Длительность импульса в отсчетах
        A0 = (double*)calloc(J0,sizeof(double));
        l=0;
        for(t=0;t<=T;t+=dt){
                l++;
                m=(int)(t/td)+1;
                m1=m+1;
                if(m1<=jb){
                        A0[l]=Xt[m]+(Xt[m1]-Xt[m])*(t/td - (double)(m-1));
                        if(fabs(A0[l])>1) A0[l]=A0[l]/fabs(A0[l]);
                }
                else{
                        A0[l]=0;
                }
        }
//------------------------------------------------------------//
        free(Xt);
}
void GearsSignal::CalcPeriod(){
        int i,k,imax1,imax2;
        double **Emax,**EMAX;
        double E1,E2,E,Emax1,Emax2;
        Emax = (double**)calloc(ngear,sizeof(double*));
        EMAX = (double**)calloc(ngear,sizeof(double*));
        for(k=0; k<ngear; k++){//Циклпошестерням
                Emax[k] = (double*)calloc(I,sizeof(double));
                EMAX[k] = (double*)calloc(I/2,sizeof(double));
        }
        for(k=1; k<=ngear; k++){//Циклпошестерням
                for(i=0;i<I;i++){//Сдвиг
                        E1=E2=E=0.0;
                        for(int j=0; j<I; j++) //Интегреируем
                                if(i+j>=0 && i+j<I){
                                        E1 += A[k][j] * A[k][j];
                                        E2 += A[k][i+j] * A[k][i+j];
                                        E  += A[k][j] * A[k][i+j];
```

```
                                }
                        Emax[k-1][i] = E;
                }
                Emax1 = Emax[k-1][0];
                imax1 = imax2 = 0;
                Emax2 = 0.0;
                for(i=1;i<I-1;i++)
                        if(Emax[k-1][i]>Emax[k-1][i-1] && Emax[k-1][i]>Emax[k-
1][i+1]){
                                if(Emax2 < Emax[k-1][i]){
                                        imax2 = i;
                                        Emax2 = Emax[k-1][i];
                                }
                        }
                TP[k-1] = (double)(imax2 - imax1);
        }
        for(int i=0;i<ngear;i++) printf("TP[%d]=%lf\n",i,TP[i]);
        FILE*rez;
        rez = fopen("period_analiz.dat","w");
        for(int j=0;j<I;j++){
                fprintf(rez,"%lf       ",dt*(double)j);
                for(int i = 0; i<ngear; i++){
                        fprintf(rez,"%lf       ",Emax[i][j]);
                }
                fprintf(rez,"\n");
        }
        fclose(rez);
}
int GearsSignal::ReadFile(char *_namefile){
        FILE*file,*file2;
        int i=0;
        char str[200],ch;
        file = fopen(_namefile,"r");
        if(!file)
                return 0;
        fgets(str,'/n',file);
        while(!feof(file)){
                fgets(str,'/n',file);
                if(strlen(str)<5)
                        break;
                i++;
        }
        fclose(file);
        I = i;
        A = (double**)calloc(ngear+1,sizeof(double*));
        for(i=0;i<ngear+1;i++)
                A[i] = (double*)calloc(I,sizeof(double));
        file2 = fopen("gear_data.dat","w");
        file = fopen(_namefile,"r");
        fgets(str,'/n',file);
        i=0;
        while(!feof(file)){
                fgets(str,'/n',file);
                if(strlen(str)<5)
                        break;
                sscanf(str,"%lf %lf %lf
%lf",&A[0][i],&A[1][i],&A[2][i],&A[3][i]);
                for(int j=0; j<=ngear; j++){
                        if(j>0)
                                A[j][i] = 1.0 - 2.0*A[j][i];
                }
                fprintf(file2,"%lf     %lf     %lf
        %lf\n",A[0][i],A[1][i],A[2][i],A[3][i]);
                i++;
```

```
        }
        fclose(file);
        fclose(file2);
        return 1;
}
void GearsSignal::PrintArray(){
        FILE*rez;
        int Jmax = 0,k;
        int *DI;
        DI = (int*)calloc(ngear,sizeof(int));
        for(int i=0; i<ngear; i++)
                DI[i] = (int)DT[i];
        rez= fopen("new_signal.dat","w");
        fprintf(rez,"T A\n");
        for(int i=0;i<I;i++){
                fprintf(rez,"%d",i);
                for(int j=0; j<ngear; j++){
                        int di = i- DI[j];
                        k = direction[j];
                        if(di>=0 && di<I0[j])
                                fprintf(rez,"%lf       ",A0[k][di]);
                        else
                                fprintf(rez,"%lf       ",0.0);
                }
                fprintf(rez,"\n");
        }
        free(DI);
        fclose(rez);
}
```

Класс в котором объеденены методы анализа анализа измеренного сигнала (получен на макете спутника)

```
#ifndef _GEARS_
#define _GEARS_
struct Gears{//дляформышестерни
        int kd;
        char *kod;
};
class GearsSignal{
private:
        char *namefile;
        int *kd;//Длина кодовой последовательности
        char **kod;//Форма кодовой последовательности
        double tau1,tau2,tau3;//Длины фронтов модельного импульса
        double dt;//Расстояние между отсчетами
        int ngear;
        int I;//Кол-во отсчетов в измеренной сигнале
        double **A;//Измеренный сигнал (0-я колонка - время)
        double *TP;//Период сигнала для каждой шестерни
        int *I0;//Кол-во отсчетов в модельном импульсе
        double **A0;//Форма модельного сигнала
        char *direction;//Направление вращения
        double *DT;//Сдвиг по времени
        double *Amp;//Амплитуда кор-функции
        double *Err;//Невязка между измерениями и моделью
        void InitArray();
        int ReadFile(char *_namefile);//Чтениеданныхизфайла
        void CalcPeriod();//Определяем период сигнала для каждой шестерни
        void sort(double *A, int n);
        void SignalModel();//Модель сигнала - один период
        void IzlImpAdd(int kd,double dt,double Tadd,double tau1,double
tau2,double tau3,double T,double *&A0,int &J0);//Модельсигнала
```

```
        void IzlImpAdd2(int kd,char *kod,double dt,double Tadd,double
tau1,double tau2,double tau3,double T,double *&A0,int &J0);//Модельсигнала
        void CorAnaliz2(char rw);//Корреляционный анализ с моделью сигнала
        double* CorAnaliz(double*&A1, int I1, double*&A2, int I2, double &DT,
double &AmpAkf, char rw);
        void PrintArray();
        void EnergyError();
public:
        void calc_signal();
        void control_period();
        GearsSignal();
        GearsSignal(char *nameconfig);
        void GearsAnaliz();
};
#endif
```

Главная программа, из которой вызывается метод для анализа периодичности измеренного сигнала

```
#include<iostream>
#include <vector>
#include <math.h>
#include "Gears.h"
using namespace std;
void main(){
        GearsSignal gears("gears_config.dat");
        gears.control_period();
}
```

Задача 2.

Цель:

1. Получить навыки работы с помехоустойчивым кодированием, научиться использовать алгоритмы кодирования и декодирования сообщения, работа с байтами и битами, освоить методику пакетной передачи данных.
2. Получить навыки работы со статистическим анализом данных.

Результаты решения задачи могут быть использованы при решении задачи 5.

Постановка задачи. По ИК-каналу (со спутника на радар) передается телеметрическая информация (текстовый файл), отображающая техническое состояние спутника. Однако, в каждом канале присутствует шум и помехи, требующие организации помехозащитного кодирования передаваемых данных. Задачей команд является создание программы кодировщика, загружаемой на спутник, и программы декодировщика, действующей на стороне приемника (радара), позволяющей восстановить передаваемое сообщение.

Краткое описание подзадач:

1. Научиться работать со стендом спутник – радар. Запустить передачу данных со спутника посредством ИК канала и начать приём данных на «радаре».

2. Исследовать свойства шума, на основе данного исследования выбрать метод помехоустойчивого кодирования.
3. Разработать программу кодер (действует на стороне «спутника»).
4. Разработать программу декодер (действует на стороне «радара»).
5. Проанализировать RGB-последовательность, принятую со спутника. Провести сравнительный анализ между RGB-последовательностями, полученными на макете и принятыми со «спутника».
6. Определить характера повреждений на «спутнике», определяются шестерни, работающие неправильно. Заполнить диагностическую карту: определить по телеметрии периоды и СКО отклонения периодов от среднего на спутнике, определить характер повреждения шестерни – определить номера ошибочных элементов в коде Хэмминга на шестернях спутника.

Расчет баллов (30+10)

Расчет баллов задачи 2a (оценка эффективности кодера и декодера)

Поскольку изначально передаваемый файл является текстовым и табличным, в каждой строке несколько значений, для подсчета верно переданных данных использовалась стандартная утилита diff, которая сравнивает текстовые файлы, и перечисляет строки, в которых они отличаются.

Таким образом, параметром V, оценивающим точность передачи данных являлось количество строк (n), в которых файлы идентичны, деленное на общее количество строк в исходном файле (N):

$$V = \frac{n}{N}$$

Формула, по которой рассчитывается количество баллов (I) по точности передачи V, имеет вид:

$$I = M * V * 0{,}9,$$

где M — максимальное число основных баллов за задачу 2. Коэффициент 0,9 связан с долей основных баллов, отведенных на решение задачи 2a.

Расчет баллов задачи 2b (анализ полученных данных и заполнение диагностической карты)

Баллы за шестерню:

Правильно определен период и СКО – 0,5;

Правильно по телеметрии определена кодовая последовательность на шестерне спутника, определён битый элемент (чип) в последовательности – 0,5.

За каждую неудачную попытку снимается 5% от максимальных 3 баллов, т.е. если команда приносит правильный ответ с первой попытки, то получает 100% (3 баллов), если первая попытка неудачная, но ответ верно найден со второй попытке команда получает 95% от 3 баллов, т.е. 2,85 и так далее.

Код решения задачи (блочное кодирование):

Программа осуществляетблочное кодирование - входной файл разбивается на блоки с четко заданной структурой. В принятом файле блоки оказываются перемешанными из-за периодического попадания спутника в «слепые» зоны радара. Программа декодер – находит целые блоки и далее выстраивает из них исходное сообщение, выставляя блоки согласно их номерам заданном на этапе кодирования.

Программа блочного кодерования:

```
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#ifndef __CODE_DECODE_H__
#define __CODE_DECODE_H__
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define DATA_BLOCK_LEN  400
#define CODE_BLOCK_LEN  (sizeof(unsigned long))
#define FULL_BLOCK_LEN  (DATA_BLOCK_LEN+CODE_BLOCK_LEN)
#define MAX_FILE_SIZE  20000000
#define HEADER_MARK "ABCDEFG"
#define FOOTER_MARK "HIJKLMN"
#define HEADER_MARK_LEN 7
#define FOOTER_MARK_LEN 7
typedef struct{
 char head[HEADER_MARK_LEN+1];
 unsigned int code_block;
 long block_len;
}
header_code_type;
typedef struct{
 char foot[FOOTER_MARK_LEN+1];
 unsigned int code_block;
// long block_len;
}
footer_code_type;
#define HEADER_BLOCK_LEN (sizeof(header_code_type))
#define FOOTER_BLOCK_LEN (sizeof(footer_code_type))
#endif
int main(int argn,char* argv[])
{
// FILE* tmpfile;
 int mode;
  mode=1;
 unsigned long code_block=0;
 long i,j,length;
 char* block;
 unsigned char* file_content;
 unsigned char* new_file_content;
 long file_len;
 long block_len;
 char srch_block[255];
 FILE* in;
 FILE* out;
 in=fopen(argv[1],"rb");
 out=fopen(argv[2],"wb");
 file_content=calloc(MAX_FILE_SIZE+1,sizeof(char));
 if(!file_content)
  {fprintf(stderr,"no memory\n");exit(1);}
 new_file_content=calloc(MAX_FILE_SIZE+1,sizeof(char));
```

```
 if(!new_file_content)
  {fprintf(stderr,"no memory\n");exit(1);}
 block=calloc(100000,/*DATA_BLOCK_LEN*2,*/sizeof(char));
 if(!block)
  {fprintf(stderr,"no memory\n");exit(1);}
 length=fread(file_content,sizeof(char),MAX_FILE_SIZE+1,in);
 if(length>MAX_FILE_SIZE)
  {
   fprintf(stderr,"file too long to process\n");exit(1);
  }
 long actual_len;
 header_code_type header_block;
 footer_code_type footer_block;
 file_len=0;
// for(i=0;i<length;)
  for(i=0;i<length;i++)
   {    memcpy((void*)(&header_block),(void*)(file_content+i),HEADER_BLOCK_LEN);
memcpy(&footer_block,file_content+i+HEADER_BLOCK_LEN+DATA_BLOCK_LEN,FOOTER_BLOCK
_LEN);
    if(memcmp((void*)(header_block.head),HEADER_MARK,HEADER_MARK_LEN)==0
&&
       memcmp((void*)(footer_block.foot),FOOTER_MARK,FOOTER_MARK_LEN)==0
&&
     footer_block.code_block==header_block.code_block
&&
     i+HEADER_BLOCK_LEN+DATA_BLOCK_LEN+FOOTER_BLOCK_LEN<length
     )
     {
        memcpy((void*)(block),file_content+i+HEADER_BLOCK_LEN,DATA_BLOCK_LEN);
memcpy(new_file_content+header_block.code_block*DATA_BLOCK_LEN,block,DATA_BLOCK_
LEN);
        block[DATA_BLOCK_LEN]=0;
        i+=HEADER_BLOCK_LEN;
        i+=DATA_BLOCK_LEN;
        i+=FOOTER_BLOCK_LEN;
        i--;
        file_len+=DATA_BLOCK_LEN;
        continue;
     }
   }
 fprintf(stderr,"flen:%ld\n",file_len);
 char *tmp1;
 long new_len=20000000;
for(i=file_len;i>=0;i--)
 if(new_file_content[i]!=0)
  break;
// fwrite(new_file_content,sizeof(char),file_len,out);
 fwrite(new_file_content,sizeof(char),i+1,out);
 fflush(out);
}
```

Программа декодирования – поиск блоков и сборка из них исходного сообщения

```
#ifndef __CODE_DECODE_H__
#define __CODE_DECODE_H__
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define DATA_BLOCK_LEN  400
#define CODE_BLOCK_LEN  (sizeof(unsigned long))
#define FULL_BLOCK_LEN  (DATA_BLOCK_LEN+CODE_BLOCK_LEN)
#define MAX_FILE_SIZE  20000000
```

```
#define HEADER_MARK "ABCDEFG"
#define FOOTER_MARK "HIJKLMN"
#define HEADER_MARK_LEN 7
#define FOOTER_MARK_LEN 7
typedef struct{
 char head[HEADER_MARK_LEN+1];
 unsigned int code_block;
 long block_len;
}
header_code_type;
typedef struct{
 char foot[FOOTER_MARK_LEN+1];
 unsigned int code_block;
// long block_len;
}
footer_code_type;
#define HEADER_BLOCK_LEN (sizeof(header_code_type))
#define FOOTER_BLOCK_LEN (sizeof(footer_code_type))
#endif
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
void init_header_code(header_code_type* header_code,footer_code_type*
footer_code)
{
 sprintf(header_code->head,HEADER_MARK);
 sprintf(footer_code->foot,FOOTER_MARK);
 header_code->code_block=0;
 header_code->block_len=0;
}
int main(int argn,char* argv[])
{
 char* tmp_file1;
 char* tmp_file2;
 long compr_len;
 unsigned long code_block=0;
 header_code_type header_code;
 footer_code_type footer_code;
 long i,j;
 char* block;
 int mode;
 if(argn>1)
  mode=atol(argv[1]);
 else
  mode=1;
 FILE* in;
 FILE* out;
 tmp_file1=calloc(sizeof(char),40000000);
 tmp_file2=calloc(sizeof(char),40000000);
 long file_len,block_len;
 block=calloc(DATA_BLOCK_LEN*2,sizeof(char));
 code_block=0;
 file_len=0;
 in=fopen(argv[1],"rb");
 out=fopen(argv[2],"wb");
 for(i=0;!feof(in);)
  {
   block_len=fread(block,sizeof(char),DATA_BLOCK_LEN,in);
   memcpy(tmp_file1+i,block,block_len);
   i+=block_len;
  }
 compr_len=20000000;
 memcpy(tmp_file2,tmp_file1,i);
 compr_len=i;
```

```
 file_len=0;
 init_header_code(&header_code,&footer_code);
 for(j=0;j<compr_len;j+=DATA_BLOCK_LEN)
  {
   block_len=DATA_BLOCK_LEN;
   if(j+DATA_BLOCK_LEN<=compr_len)
    memcpy(block,tmp_file2+j,DATA_BLOCK_LEN);
   else
    {
     block_len=compr_len-j;
     memcpy(block,tmp_file2+j,block_len);
    }
   file_len+=block_len;
   for(i=block_len;i<DATA_BLOCK_LEN;i++)
    block[i]=0;
   header_code.block_len=DATA_BLOCK_LEN;
   header_code.code_block=code_block;
   footer_code.code_block=code_block;
   code_block++;
   fwrite(&(header_code),sizeof(char),sizeof(header_code),out);
   fwrite(block,sizeof(char),DATA_BLOCK_LEN,out);
   fwrite(&(footer_code),sizeof(char),sizeof(footer_code),out);
   for(i=0;i<DATA_BLOCK_LEN;i++)
    block[i]=0;
  }
 code_block=-1;
 fprintf(stderr,"file len: %ld\n",file_len);
 memcpy(block,&file_len,sizeof(long));
 fwrite(block,sizeof(char),DATA_BLOCK_LEN,out);
 fwrite(&code_block,sizeof(char),CODE_BLOCK_LEN,out);
 fflush(out);
 fclose(out);
//copy result twice to compensate dead regions of satellite
 long len=0;
 out=fopen(argv[2],"rb");
 len=fread(tmp_file1,sizeof(char),40000000,out);
 fclose(out);
 out=fopen(argv[2],"wb");
 fwrite(tmp_file1,sizeof(char),len,out);
 fwrite(tmp_file1,sizeof(char),len,out);
 fclose(out);
}
```

Программа компилируется с -lm option

Варинат программы кодирования сигнала помехоустойчивым кодом Хемминга (8,4). Ипользуется (8,4) так как плотность помех изменяют не большего одного бит в байте

```
#include<iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
unsigned char mask[] = {128,64,32,16,8,4,2,1};
void read_file(const char name[], char*&buff, long &size);
void write_file(const char name[], char*&buff, long &size);

class Hamming
{
private:
        int order(int a);
```

```
        void CharToBinaryArray(char &c, char
arr[]);//Разложениебитовкодасимволавмассив
        char EncodedPartArray(char arr[],int
begin);//Кодированиечастибитовсимвола (сейчасэто 4 бита)
        char BinaryArrayToChar(char arr[]);//Массивбитовпреобразуемвсимвол
        char DecoderChar(char &e);//Декодируем отдельный символ
        char UnionChar(char e[]);//Объединение двух отдельных символов при
декодировании (так используется расширенный код Хэмминга)
        void Encoder(char &c, char e[]);//Кодирование расширенным кодом Хэмминга
(4,8) из одного символа получаем 2
        char Decoder(char e[]);//Декодирование пары символов и объединение в
один
public:
        unsigned int lencod;//Длина в битах закодированного сообщения
        unsigned int lenbit;//Длина в битах полезного сообщения
        unsigned int lenctr;//Кол-во контролльных бит
        Hamming();
        Hamming(unsigned int _lencod);
        void ENCODER(char*&buff, const long size, char*&buff_encod, long
&size_encod);
        void DECODER(char*&buff, long &size, char*&buff_encod, const long
size_encod);
};
void read_file(const char name[], char*&buff, long &size){
        FILE*inp;
        inp = fopen(name,"rb");
        if(!inp){
                printf("File %s not open!\n",name);
                return;
        }
        else{
                fseek(inp, 0, SEEK_END);                                        //
переместить внутренний указатель в конец файла
                size = ftell(inp);
                rewind(inp); //Устанавливаем указатель в конец файла
                buff = (char*)calloc(size,sizeof(char));
                if(buff == NULL){
                        printf("Error of memory\n");
                        return;
                }
                printf("size = %ld\n",size);
                long result = fread(buff,1,size,inp);
                printf("result = %d\n",result);
                if(result != size){
                        printf("Error of read\n");
                        return;
                }
                fclose(inp);
        }
}
void write_file(const char name[], char*&buff, long &size){
        FILE*out;
        out = fopen(name,"wb");
        if(!out){
                printf("File %s not open!\n",name);
                return;
        }
        else{
                long result = fwrite(buff,1,size,out);
                if(result != size){
                        printf("Error of write!\n");
                        return;
                }
                fclose(out);
```

```
        }
}
class CodFile{
private:
        int num;// кол-вобайтвблоке
        char *buff, *buff_encod;
        long size, size_encod;
        Hamming hamming;
        void ReadFile(const char name[], char*&_buff, long
&_size);//Чтениеданныхизфайлаисброссодержимоговбуфер buff
        void WriteFile(char name[], char*&_buff, long
&_size);//Записьсодержимого buff вфайл
public:
        CodFile(){}
        CodFile(const char *filename);//Заполнениебуфераданнымиизфайла
        void ENCODER(char*name_inp_file, char*name_out_file);
        void DECODER(char*name_inp_file, char*name_out_file);
        void ADDNOISE(char*name_inp_file, char*name_out_file);
        void WriteFile(char simbol, const char *name);//Записьсодержимого buff
вфайл
        ~CodFile();
};
void main(int argc, char *argv[]){
        if(argc<4){
                printf("Run file.exe E/D/N inpfile outfile\n");
                return;
        }
        CodFile codfile;
        if(argv[1][0] == 'E' || argv[1][0] == 'e'){
                codfile.ENCODER(argv[2],argv[3]);
        }
        else if(argv[1][0] == 'D' || argv[1][0] == 'd'){
                codfile.DECODER(argv[2], argv[3]);
        }
        else if(argv[1][0] == 'N' || argv[1][0] == 'n'){
                codfile.ADDNOISE(argv[2], argv[3]);
        }
        else{
                printf("Second param error (E|D)\n");
        }
}
CodFile::CodFile(const char*namefile){
        read_file(namefile,buff,size);
        puts(buff);
        printf("n=%d size = %d\n",strlen(buff),size);
}
void CodFile::DECODER(char*name_inp_file, char*name_out_file){
        read_file(name_inp_file,buff_encod,size_encod);
        hamming.DECODER(buff, size, buff_encod, size_encod);
        write_file(name_out_file,buff,size);
}
void CodFile::ENCODER(char*name_inp_file, char*name_out_file){
        read_file(name_inp_file,buff,size);
        hamming.ENCODER(buff,size,buff_encod,size_encod);
        write_file(name_out_file,buff_encod,size_encod);
}
void CodFile::ADDNOISE(char*name_inp_file, char*name_out_file){
        read_file(name_inp_file,buff_encod,size_encod);
        long i;
        for(i=0;i<size_encod;i++)
                buff_encod[i] = buff_encod[i] ^ mask[rand()%8];
        write_file(name_out_file,buff_encod,size_encod);
}
void CodFile::WriteFile(char simbol,const char *name){
```

```
        FILE*out;

        if(simbol =='D' || simbol =='d'){
                write_file(name,buff,size);
        }
        else if(simbol =='E' || simbol =='e'){
                write_file(name,buff_encod,size_encod);
        }
        else{
                printf("Parametr for write: D|d|E|e\n");
                return;
        }
}
CodFile::~CodFile(){
        delete [] buff;
        delete [] buff_encod;
}
Hamming::Hamming(){
        lencod = 7;//Общая длина
        lenbit = 4;//Полезная информация
}
void Hamming::DECODER(char*&buff, long &size, char*&buff_encod, const long
size_encod){
        long i=0,j=0;
        char e[2];
        size = size_encod / 2;
        buff = new char[size];

        while(size_encod>i){
                e[0] = buff_encod[i++];
                e[1] = buff_encod[i++];
                buff[j++] = Decoder(e);
        }
}
void Hamming::ENCODER(char*&buff, const long size, char*&buff_encod, long
&size_encod){
        long i=0,j=0;
        char e[2];
        size_encod = 2 * size;
        buff_encod = new char[size_encod];
        for(i=0;i<size;i++){
                Encoder(buff[i],e);
                buff_encod[j++]=e[0];
                buff_encod[j++]=e[1];
        }
}
char Hamming::Decoder(char e[]){
        char D,d[2];
        d[0] = DecoderChar(e[0]);//Декодируем
        d[1] = DecoderChar(e[1]);//Декодируем
        D = UnionChar(d);//Объединяем
        return D;
}
void Hamming::Encoder(char &c, char e[]){
        char arr[8];
        CharToBinaryArray(c,arr);
        e[0] = EncodedPartArray(arr,0);//КодируемХэмингом(4,7)
        e[1] = EncodedPartArray(arr,4);//КодируемХэмингом(4,7)
}
char Hamming::UnionChar(char d[]){
        char Data[8]={0};
        char data[2][8]={0};
        CharToBinaryArray(d[0],data[0]);
        CharToBinaryArray(d[1],data[1]);
```

```
        Data[0]=data[0][0];
        Data[1]=data[0][1];
        Data[2]=data[0][2];
        Data[3]=data[0][4];
        Data[4]=data[1][0];
        Data[5]=data[1][1];
        Data[6]=data[1][2];
        Data[7]=data[1][4];
        return BinaryArrayToChar(Data);
}
char Hamming::DecoderChar(char &e){
        char c,c1,c2,c3;
        char data[8]={0};
        CharToBinaryArray(e,data);
        c1=data[6]^data[4]^data[2]^data[0];
        c2=data[5]^data[4]^data[1]^data[0];
        c3=data[3]^data[2]^data[1]^data[0];
        c=c3*4+c2*2+c1;
        if(c)
                if(data[7-c]=='0')
                        data[7-c]='1';
                else
                        data[7-c]='0';
        return BinaryArrayToChar(data);
}
char Hamming::EncodedPartArray(char arr[],int begin){
        char data[8]={0};
        data[0]=arr[0+begin];
        data[1]=arr[1+begin];
        data[2]=arr[2+begin];
        data[4]=arr[3+begin];
        data[6]=data[0]^data[2]^data[4];
        data[5]=data[0]^data[1]^data[4];
        data[3]=data[0]^data[1]^data[2];
        char c = BinaryArrayToChar(data);
        return c;
}
void Hamming::CharToBinaryArray(char &c, char arr[]){
        for(int i=0;i<8;i++)
                arr[i] = c & mask[i] ? '1' : '0';
}
char Hamming::BinaryArrayToChar(char arr[]){
        char c=0;
        for(int i=0;i<8;i++)
                c |= (arr[i] - '0') ? mask[i] : 0;
        return c;
}
```

Задача 3.

Цель:

1. Научиться работать с программным интерфейсом управления «радаром» (API).
2. Получить навыки работы по проектированию предсказательных алгоритмов, в частности алгоритмов сопровождения движущихся объектов.

Результаты решения задачи могут быть использованы при решении задач 2 и 5.

Постановка задачи.

Необходимо разработать алгоритм сопровождения спутника радаром. Имитатор канала связи по легенде трека назван системой «радар-спутник» и состоит из подвижной каретки «спутник» и вращающейся вокруг вертикальной оси каретки «радар», рисунок 4. Каретка «спутник» может передвигаться вдоль рельсы (ось X) и имеет инфракрасный (ИК) передатчик. В зависимости от относительного отклонения спутника от оси наблюдения радара уровень шума в принимаемом канале меняется. Таким образом, моделируется отношение сигнал/шум и диаграмма направленности радара, плотность битового шума зависит от величины углового смещения ΔX, но нигде не превышает одного бита на байт. Для обеспечения наименьших уровней помех в принимаемом сигнале необходимо разработать алгоритм сопровождения спутника радаром.

Управление спутником осуществляется с помощью компьютера RaspberryPi, который получает необходимые для передачи данные от компьютера управления и передает их во время движения. Блок «Радар» оборудован компьютером RaspberryPi и цифровой видеокамерой, с помощью библиотеки OpenCV он распознает положение графического маркера «спутника» в поле зрения камеры и передает угловое положение маркера ΔX относительно центра камеры вдоль оси X на управляющий компьютер. Одновременно с этим, ИК-приемник «радара» принимает поток данных, передаваемый «спутником».

Краткое описание подзадач:

1. Необходимо разработать алгоритм сопровождения спутника радаром. Входными данными для программы является горизонтальное смещение спутника относительно оси радара ΔX. Для управления радаром используются команды, позволяющие регулировать скорость и направление вращения камеры.
2. Провести тестирование разработанной программы.
3. Использовать разработанную программу сопровождения спутника во время приема данных при решении задач 2 и 5, так как это позволяет значительно снизить уровень помех.

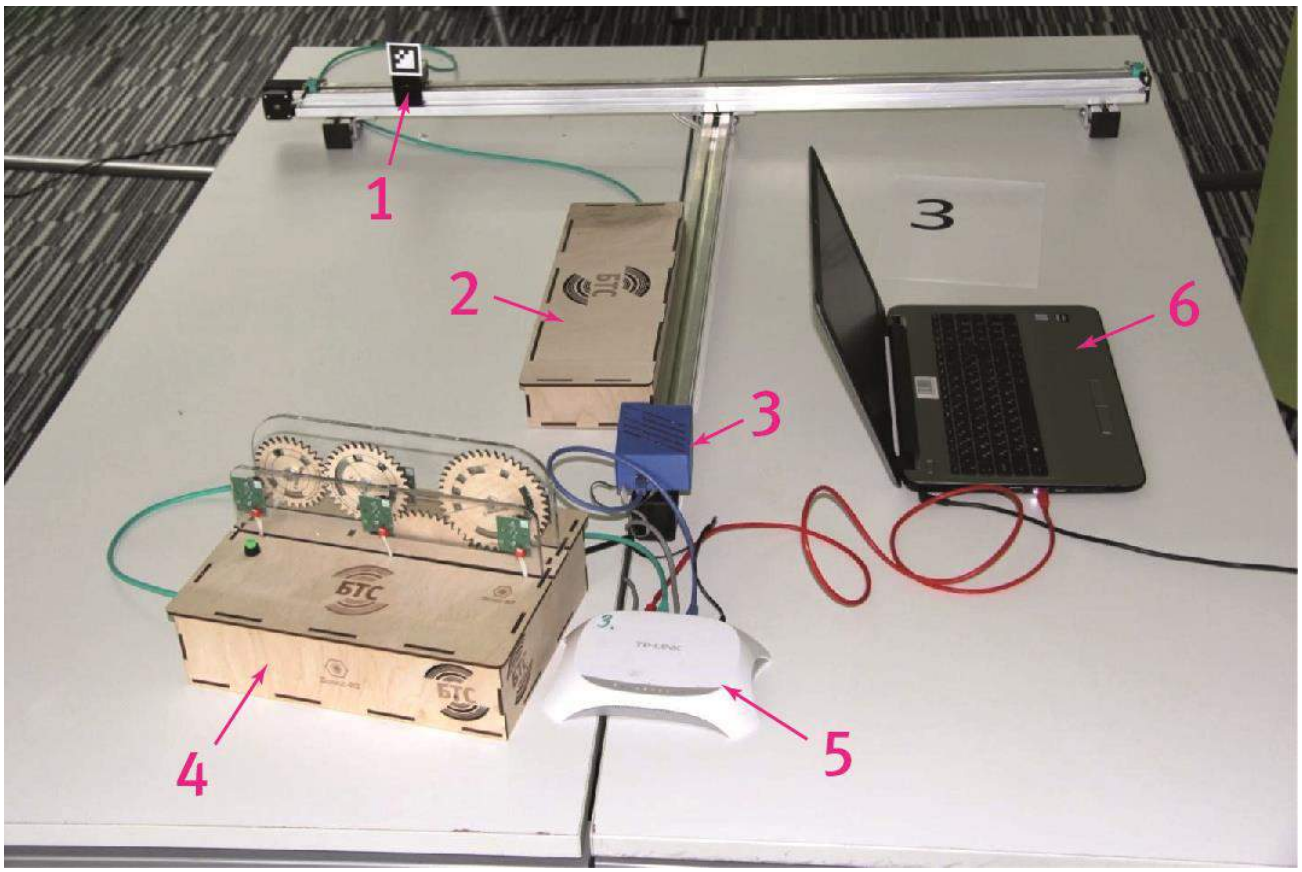

Рисунок 4. Общий вид стенда: 1 – подвижная передающая каретка «спутник» с графическим маркером, 2 – блок управления, 3 – подвижная приемная каретка «радар» с видеокамерой, 4 – макет с шестеренками, 5 – роутер, 6 – компьютер управления стендом.

Расчет баллов за задачу 3 (15 + 5).

По каждому 64-байтному блоку данных (условное обозначение - точка) рассчитывается уровень невязки сопровождения:

$$n_i = \{(dx) < 100 : \frac{(dx)}{100};\qquad (dx) > 100 : 1\},$$

при невязке больше 100 единиц (пикселов видеокамеры) этот уровень равен единице, при невязке меньше 100 единиц этот уровень уменьшается до нуля пропорционально невязке. Итоговая оценка невязки V является средней невязкой по всему объему переданных данных:

$$V = \frac{1}{N}\sum \quad n_i$$

При этом полный объем переданных данных составлял порядка 660 кБайт, или порядка 10500 64-байтных блоков (точек). Учитывая, что часть этих точек (порядка 15–20 процентов) передается в слепых зонах, где их невозможно принять, для полного сопровождения требуется статистика принятых данных не менее 8000 точек.

Таким образом, нами рассматривались результаты участников, в которых количество принятых точек составляло порядка 8000 и более, и рассматривалась средняя невязка сопровождения V, как

критерий точности сопровождения.

Расчет основных баллов (I) по вспомогательным (V) : I = M*(Vmin/V)/100, где M — максимальное число баллов за задачу (15). Vmin - наилучшая точность, достигнутая всеми командами

Вариант решения задачи сопровожденя радаром перемещающегося спутника.

```
import java.net.*;
import java.io.*;
import java.math.*;
class Tracking {
public static void main(String args[])
throws Exception{
Client client=new Client();
int speed=1;
int dir=1;
int dx;
int abs_dx;
client.start();
client.left(1);
Thread.sleep(1000);
client.right(speed);
for(int i=0;i<10;i=0) // in fact - infinite loop, to fix smart java compiler
{
    while(!client.readstatus())
     {
      System.out.println("not ready");
      Thread.sleep(100);
      client.readstatus();
     }
 dx=(int)client.getDx();
 abs_dx=Math.abs(dx);
// System.out.print("dx: "); System.out.println(dx);

 if(dx<500 && dx>-500)
 {
  speed=1;
  if(dx<30 && dx>-30)
   {
    client.stop();
//    System.out.println("stop");
   }
  else
   {
    speed=abs_dx/20;
//    System.out.print("move: ");
//    System.out.print(speed);
    if(dx<0)
     {
      client.right(speed);
//      System.out.println("right ");
     }
    if(dx>0)
     {
      client.left(speed);
//      System.out.println("left ");
     }
   }
  }
 else
```

```
  {
   speed=0;
//   System.out.println("DO NOTHING - CAN NOT SEE");
  }
//search if lost
 if(client.ifPositionRight())
  {
   speed=10;
   client.left(speed);
//   System.out.println("LOST, SEARCH LEFT");
  }
 if(client.ifPositionLeft())
  {
   speed=10;
   client.right(speed);
//   System.out.println("LOST, SEARCH RIGHT");
  }
 Thread.sleep(10);
}
 client.quit();
 client.stop();
}
}
```

Задача 4.

Цель:

1. Получить навыки работы с большими объемами данных.
2. Получить навыки работы с изображениями в растровом формате.
3. Познакомиться с печатью на 3D принтере.

Результаты решения задачи используются при решении задачи 5.

Постановка задачи. По заданной телеметрии (RGB-последовательности) построить цифровую модель шестерни в растровом формате (построить прорези и стенки в нужных положениях), так, чтобы при установки ее на макет спутника, можно было бы получить требуемую/заданную телеметрию (RGB-последовательности).

Краткое описание подзадач:

1. Задание на перепрошивку «спутника»: по заданной RGB-последовательности разработать новую модель шестерни, определить новые положения прорезей и стенок.
2. Построить цифровую модель шестерни, в растровом формате, с помощью которой можно получить требуемую телеметрию (RGB-последовательность) (пункт 1).
3. Цифровой образ специальным программным обеспечением переводится в формат stl (выполняют организаторы). Определяются возможные ошибки в цифровой модели. Если ошибок нет, то шестерня печатается на 3D принтере.
4. Проверить на макете напечатанную шестерню и провести сравнительный анализ между заданной RGB-последовательностью и полученной на макете с новой шестерней.

Расчет баллов(10 + 5)

Количество элементарных ошибок в цифровом образе шестерни не более 10, что обеспечивает возможность печати на 3D принтере.

Если шестерня напечатана верно – невязка с заданной RGB-последовательностью не превышает 10%, за задачу ставятся все 10 баллов. Если шестерня напечатана в отведенное под соревнование время добавляются бонусные 5 баллов.

Если невязка **Err** более 10% и менее 20%, то баллы начисляются так:

Баллы = 100*(0,2 – Err)

Если невязка более 20% цифровая модель не принимается.

Вариант программы для решения задачи 4. Сначала определяются параметры шестерни и убираются прорези в исходной шестерни, затем определяется угловое положение и размеры новых прорезей, которые далее «вырезаются» внаправлении по часовой. На вход программы подается кодовая последовательность из ‘0’ и ‘1’, которую должна реализовывать новая шестерня, на выходе программа возвращает файл, в котором сформирован образ шестерни в требуемом растровом формате. Кодрешения:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#pragma comment(linker, "/STACK:10000000000")
#define MAXLEN 6000
#define PI 3.14159265358979323846264338327795
#define PI2 6.28318530717958647692528676655900
#define rad (PI/180.0)
#define grad (180.0/PI)
signed char** arr;
long size_x,size_y;
int paint(int i, int j){
	if(arr[i][j] == 1)
		return 0;
	arr[i][j] = 1;
	return 1 + paint(i+1,j) + paint(i-1,j) + paint(i,j-1) + paint(i,j+1);
}
void windiws(char *gr,double wind[100][2],int &j){
	int i0,i,l = strlen(gr);
	char start;
	double df = PI2/(double)l;
	j = 0;
	for(i=1;i<l;i++){
		start = gr[i-1];
		i0 = i - 1;
		while(gr[i] == gr[i-1] && i<l)
			i++;
		if(gr[i-1] == '1'){
			wind[j][0] = df*(double)(i0);
			wind[j][1] = df*(double)(i);
			j++;
		}
	}
	if(i == l && gr[i-1] == '1'){
```

```
                wind[j][0] = df*(double)(i-1);
                wind[j][1] = df*(double)(i);
                j++;
        }
}
void create_wind(double wind[], double R1, double R2, double x0, double y0){
        double r,fi,dfi = 1.0/R2/2.0;
        for(r=R1;r<=R2;r+=0.25){
                for(fi = wind[0]; fi<=wind[1]; fi += dfi){
                        int i = (int)(x0 + r*sin(fi) + 0.5); //номерстолбца
                        int j = (int)(y0 - r*cos(fi) + 0.5); //номерстроки
                        arr[i][j] = 0;
                }
        }
}
void found_param(double Rg,double Rcod,double x0,double y0,double &R1,double
&R2){
        int k=0;
        double fi,fimid,dfi = 0.1*rad;
        double fil[2];
        for(fi = 0.0; fi <= PI2; fi+=dfi){
                int i = (int)(x0 + Rcod*sin(fi) + 0.5); //номерстолбца
                int j = (int)(y0 - Rcod*cos(fi) + 0.5); //номерстроки
                if(arr[i][j] == 0){
                        if(!k) fil[0] = fi;
                        fil[1] = fi;
                        k++;
                }
                if(arr[i][j] == 1 && k)
                        break;
        }
        fimid = (fil[0]+fil[1])/2.0;
        k = 0;
        double arrold = arr[(int)(y0 + 0.5)][(int)(x0 + 0.5)];
        for(double r = 1; r<Rg; r++){
                int i = (int)(x0 + r*sin(fimid) + 0.5); //номерстолбца
                int j = (int)(y0 - r*cos(fimid) + 0.5); //номерстроки
                if(arr[i][j] != arrold){
                        k++;
                        if(k == 2)
                                R1 = r;
                        if(k == 3)
                                R2 = r;
                }
                arrold = arr[i][j];
        }
}
void main(int argn,char* argv[]){
        FILE*stream,*newgear;
        char*string,*newstring;
        double wind[100][2];
        char cod[100];
        int lcod;
        long x,y,a1,a4;
        string=(char*)calloc(MAXLEN,sizeof(char));
        newstring=(char*)calloc(MAXLEN,sizeof(char));
        if(argn<4){
                fprintf(stderr,"usage: %s inputfile output cod\n",argv[0]);
                exit(1);
        }
        stream=fopen(argv[1],"rt");
        strcpy(newstring,argv[2]);
        strcpy(cod,argv[3]);
        windiws(cod,wind,lcod);
```

```
        fscanf(stream,"%ld%ld%ld%ld",&a1,&size_x,&size_y,&a4);
        arr=(signed char**)calloc(MAXLEN,sizeof(signed char*));
        for(y=0;y<size_y && y<MAXLEN;y++)
                arr[y]=(signed char*)calloc(MAXLEN,sizeof(signed char));
        for(y=0;y<size_y;y++){
                fscanf(stream,"%s",string);
                for(x=0;x<size_x;x++){
                        arr[y][x] = (string[x]=='0') ? 0 : 1;
            }
        }
        fclose(stream);
        double x0 = ((double)size_x)/2.0;
        double y0 = ((double)size_y)/2.0;
        double Rg = (x0 + y0)/2.0;
        double R1,R2,Rcod = Rg*3.0/5.0;
        double fi,fimid,dfi = 0.1*rad;
        found_param(Rg,Rcod,x0,y0,R1,R2);
        for(double fi = 0.0; fi <= PI2; fi+=dfi){
                int i = (int)(x0 + Rcod*sin(fi) + 0.5); //номерстолбца
                int j = (int)(y0 - Rcod*cos(fi) + 0.5); //номерстроки
                if(i>=0 && i<size_x && j>=0 && j<size_y)
                        if(arr[i][j] == 0)
                                printf("%d\n",paint(i,j));
        }
        for(int i = 0; i<lcod; i++)
                create_wind(wind[i],R1,R2,x0,y0);
        newgear = fopen(newstring,"wb");
        char ch[1],s[1];
        ch[0] = 10;
        char st[20],str[20];
        itoa(a1,st,10); strcpy(str,st); strcat(str," ");
        itoa(size_x,st,10); strcat(str,st); strcat(str," ");
        itoa(size_y,st,10); strcat(str,st); strcat(str," ");
        itoa(a4,st,10); strcat(str,st);
        fwrite(str,strlen(str),1,newgear);
        fwrite(ch,1,1,newgear);
        for(y=0;y<size_y;y++){
                for(x=0;x<size_x;x++){
                        s[0] = (arr[y][x]==0) ? '0' : '1';
                        fwrite(s,1,1,newgear);
            }
                fwrite(ch,1,1,newgear);
        }
        fclose(newgear);
        printf("OK!\n");
}
```

Задача 5.

Цель:

1. Получить навыки работы с беспотерьной передачей данных в условиях канала с потерями.
2. Получить навыки работы с компрессией данных

Задача является финальным испытанием, для успешного решения необходимо решить четыре предыдущих задачи.

Постановка задачи.

Требуется передать образ новой шестерни на спутник в условиях канала с потерями и большого объема исходных данных и медленного канала передачи. Для этого необходимо написать шифратор и дешифратор, которые смогут преобразовать данные таким образом, чтобы их эффективно передать через данный канал связи. Необходимо заметить, что данные о шестерне в теле дешифратора хранить нельзя, и одна и та же пара шифратор-дешифратор должна иметь возможность передавать образы различных шестеренок. Следует заметить, что решение задачи «в лоб», без детального анализа задачи возможно, но крайне неэффективно, поскольку потребует 5–6 часов работы комплекса, что в условиях ограничения олимпиады по времени приведет к проигрышу.

Краткое описание подзадач:

1. Проанализировать возможность беспотерьной передачи большого объема данных через канал с существенными потерями

2. Разработать алгоритм беспотерьной передачи большого объема данных, включающий всебя контроль целостности передаваемых данных, защиту от шумов, эффективное сопровождениеспутника радаром для минимизации шумов и максимально эффективное сжатие данных.

Расчет баллов (30 + 10)

При расчете баллов за задачу 5 оценивалось число верно переданных байтов файла, по аналогии с оценкой задачи 2.

Параметром V, оценивающим точность передачи данных являлось количество байтов (n), в которых файлы идентичны, деленное на общее количество строк в исходном файле (N):

$$V = \frac{n}{N}$$

Поскольку основные баллы не присваивались (задача полностью не была решена ни одной командой), бонусные баллы рассчитывались, как

$I = M * V$

где M - максимальное число бонусных баллов за задачу.

Вариант программы, реализующий решение задачи 5. Алгоритм решения: сначала исходный образ шестерни заданный в растровом формате определенного формата анализируется, в результате чего исходному образу шестерни ставится в соответствие набор определенных параметров: радиус отверстия под вал, внутренний и внешний радиусы прорезей, внешний радиус, форма зубца. Далее эти параметры объемом примерно в 100 байт многократно передаются на спутник, в так как данный объем очень мал, то при многократной передаче такой пакет может быть с вероятность очень близкой к едениуе передан на спутник без искажений. На спутнике действует программа декодер, на вход которой поступает блок параметров шестерни, на основе которого создается полный растровый образ исходной/передаваемой шестерни.

Программа кодирующая шестерню – определение базовых параметров.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define MAXLEN 6000
#define CODELEN 10000
#define REPEATS 100
#define FILELEN 5714
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define HEADER_MARK "BEGIN"
#define FOOTER_MARK "END"
#define HEADER_MARK_LEN strlen(HEADER_MARK)
#define FOOTER_MARK_LEN strlen(FOOTER_MARK)
#include <sys/resource.h>
signed char** arr;
long size_x,size_y;
//increase stack size - necessary to use this function if default stack defined
by system administrtor is too small.
//Big stack is required to fill_holes() algorithm can work properly at large
arrays
//or use "ulimit -s unlimited" shell command instead
//or fix /etc/securty/limits.conf instead
int inc_stack_size()
{
    const rlim_t kStackSize = 64L * 1024L * 1024L;   // min stack size = 64 Mb
    struct rlimit rl;
    int result;
    result = getrlimit(RLIMIT_STACK, &rl);
    if (result == 0)
    {
        if (rl.rlim_cur < kStackSize && rl.rlim_cur>=0)
        {
            rl.rlim_cur = -1;
            result = setrlimit(RLIMIT_STACK, &rl);
            if (result != 0)
             { fprintf(stderr, "Error increasing stack\n"); }
        }
    }
    return 0;
}
```

```
//fill holes aglorithm
int fill_holes(long x,long y,int src,int dest)
 {
  long x1,y1;
  if(x>=size_x || x<0)
   return;
  if(y>=size_y || y<0)
   return;
  arr[y][x]=dest;
  if(x>0 && arr[y][x-1]==src)
   fill_holes(x-1,y,src,dest);
  if(x<size_x-1 && arr[y][x+1]==src)
   fill_holes(x+1,y,src,dest);
  if(y>0 && arr[y-1][x]==src)
   fill_holes(x,y-1,src,dest);
  if(y<size_y-1 && arr[y+1][x]==src)
   fill_holes(x,y+1,src,dest);
 }
//print shape algorithm
int print_she(FILE* strout)
 {
  long x,y;
  for(y=0;y<size_y;y+=1)
  {
    for(x=0;x<size_x;x+=1)
     fprintf(strout,"%d",(int)arr[y][x]);
    fprintf(strout,"\n");
   }
 }
main(int argn,char* argv[])
{
 char* src;
 char* record;
 char* string;
 char* outstr;
 FILE* stream;
 FILE* strout;
 long x,y;
 long a1,a4;
 long i,s;


char rec_c[10240];
char rec_h[10240];
char rec_t[10240];
int val_t[10240];
long xmid,ymid,r;
 src=calloc(FILELEN*REPEATS*2,sizeof(char));
 record=calloc(FILELEN+10,sizeof(char));
 string=calloc(MAXLEN,sizeof(char));
 outstr=calloc(CODELEN,sizeof(char));
inc_stack_size();
 if(argn<3)
  {
   fprintf(stderr,"usage: %s filein fileout\n",argv[0]);
   exit(1);
  }
 stream=fopen(argv[1],"rt");
 strout=fopen(argv[2],"wt");
 arr=calloc(MAXLEN,sizeof(signed char*));
 for(y=0;y<MAXLEN;y++)
  arr[y]=calloc(MAXLEN,sizeof(signed char));
 for(y=0;y<size_y;y++)
   for(x=0;x<size_x;x++)
```

```
    arr[y][x]=0;
 long length;
 length=fread(src,sizeof(char),FILELEN*REPEATS*2,stream);
 char hdr[HEADER_MARK_LEN+1];
 char ftr[FOOTER_MARK_LEN+1];
  for(i=0;i<length;i++)
   {
   //get header and footer positions
    memcpy((void*)(hdr),(void*)(src+i),HEADER_MARK_LEN);
    hdr[HEADER_MARK_LEN]=0;
    memcpy((void*)(ftr),(void*)(src+i+FILELEN-FOOTER_MARK_LEN),FOOTER_MARK_LEN);
    ftr[FOOTER_MARK_LEN]=0;
    memcpy((void*)(record),(void*)(src+i+HEADER_MARK_LEN),FILELEN-
FOOTER_MARK_LEN-HEADER_MARK_LEN);

    //are footer and header correct?
    if(memcmp((void*)(hdr),HEADER_MARK,HEADER_MARK_LEN)==0
&&
       memcmp((void*)(ftr),FOOTER_MARK,FOOTER_MARK_LEN)==0
&&
     i+FILELEN<length
     )
     {
//read parameters if received correctly
#define POS_C 6
#define LEN_C 37
        record[FILELEN]=0;
        if(src[i+POS_C]=='C' && src[i+POS_C+LEN_C]=='C')
         {
          memcpy((void*)(rec_c),(void*)(src+i+POS_C+1),LEN_C-1);
          rec_c[LEN_C-2]=0;
sscanf(rec_c,"%ld%ld%ld%ld%ld%ld%ld",&a1,&size_x,&size_y,&a4,&xmid,&ymid,&r);
         }
//read code if received correctly
#define POS_H 45
#define LEN_H (677-45)
        if(src[i+POS_H]=='H' && src[i+POS_H+LEN_H]=='H')
         {
          memcpy((void*)(rec_h),(void*)(src+i+POS_H+1),LEN_H-1);
          rec_h[LEN_H-2]=0;
          long j;
         }
//read teeth if received correctly
#define POS_T 679
#define LEN_T (5709-679)
long old_x,old_y;
long x1,y1;
        if(src[i+POS_T]=='T' && src[i+POS_T+LEN_T]=='T')
         {
          memcpy((void*)(rec_t),(void*)(src+i+POS_T+1),LEN_T-1);
          rec_t[LEN_T-2]=0;
          long j;
            double R,phi=0;
//fill shape by teeth
          for(j=0;j<LEN_T/4;j++,phi+=0.005)
           {
            sscanf(rec_t+j*4,"%d",&(val_t[j]));
            R=2400.0+val_t[j];
            if(j>0)
            {
             old_x=x;
             old_y=y;
             x=xmid+(long)(sin(phi)*R);
             y=ymid+(long)(cos(phi)*R);
```

```
              double dx,dy;
              long xn,yn;
              dx=(double)(x-old_x)/1000.0;
              dy=(double)(y-old_y)/1000.0;
              for(s=0;s<=1000;s++)
               {
                xn=(long)((double)old_x+dx*(double)s);
                yn=(long)((double)old_y+dy*(double)s);
               for(x1=xn-2;x1<xn+2;x1++)
                for(y1=yn-2;y1<yn+2;y1++)
                  {
                   arr[y1][x1]=1;
                  }
               }
              }
              else
              {
               x=xmid+(long)(sin(phi)*R);
               y=ymid+(long)(cos(phi)*R);
              }
             }
              R=2400.0+val_t[0];
              old_x=x;
              old_y=y;
              x=xmid+(long)(sin(phi)*R);
              y=ymid+(long)(cos(phi)*R);
              double dx,dy;
              long xn,yn;
              dx=(double)(x-old_x)/1000.0;
              dy=(double)(y-old_y)/1000.0;
              for(s=0;s<=1000;s++)
               {
                xn=(long)((double)old_x+dx*(double)s);
                yn=(long)((double)old_y+dy*(double)s);
                for(x1=xn-2;x1<xn+2;x1++)
                 for(y1=yn-2;y1<yn+2;y1++)
                   {
                    arr[y1][x1]=1;
                   }
               }
           }
         break;
      }
    }
//fill shape
 fill_holes(xmid,ymid,0,1);
//draw central hole
 for(x=xmid-r;x<=xmid+r;x++)
  for(y=ymid-r;y<=ymid+r;y++)
   if((x-xmid)*(x-xmid)+(y-ymid)*(y-ymid)<r*r)
    arr[y][x]=0;
//draw code
double r1;
double phi;
double old_x,old_y;
double dx,dy;
long x1,y1,j;
r1=1700;
//draw tangential lines
for(r1=1621;r1<2036;r1+=2036-1621-1)
{
 for(j=0,phi=0.;phi<3.14159*2.;phi+=0.01,j++)
  {
    if(j>0)
```

```
     {
      old_x=x;
      old_y=y;
     }
    x=xmid+(long)((float)r1*sin(phi));
    y=ymid+(long)((float)r1*cos(phi));
    if(j==0)
     {
      old_x=x;
      old_y=y;
     }
    double dx,dy;
    long xn,yn;
    dx=(double)(x-old_x)/1000.0;
    dy=(double)(y-old_y)/1000.0;
    for(s=0;s<=1000;s++)
     {
      xn=(long)((double)old_x+dx*(double)s);
      yn=(long)((double)old_y+dy*(double)s);
      for(x1=xn-5;x1<xn+5;x1++)
       for(y1=yn-5;y1<yn+5;y1++)
         {
          arr[y1][x1]=rec_h[j+1]-48;
         }
     }
  }
}
//draw radial lines
for(r1=1621;r1<2036;r1+=10)
{
 for(j=0,phi=0.;phi<3.14159*2.;phi+=0.01,j++)
  {
    if(j>0)
     {
      old_x=x;
      old_y=y;
     }
    x=xmid+(long)((float)r1*sin(phi));
    y=ymid+(long)((float)r1*cos(phi));
    if(j==0)
     {
      old_x=x;
      old_y=y;
     }
    if((rec_h[j+1]!=rec_h[j+2] || rec_h[j+1]!=rec_h[j]) && (rec_h[j+1]-48==0))
    {
    double dx,dy;
    long xn,yn;
    dx=(double)(x-old_x)/1000.0;
    dy=(double)(y-old_y)/1000.0;
    for(s=0;s<=1000;s++)
     {
      xn=(long)((double)old_x+dx*(double)s);
      yn=(long)((double)old_y+dy*(double)s);
      for(x1=xn-5;x1<xn+5;x1++)
       for(y1=yn-5;y1<yn+5;y1++)
         {
          arr[y1][x1]=0;
         }
     }
    }
  }
}
//fill code regions
```

```
 r1=1700;
 for(j=0,phi=0.;phi<3.14159*2.;phi+=0.01*2,j+=2)
  {
    x=xmid+(long)((float)r1*sin(phi));
    y=ymid+(long)((float)r1*cos(phi));
    if(rec_h[j+1]-48==0)
     fill_holes(x,y,1,0);
  }
//print shape
 fprintf(strout,"%ld %ld %ld %ld\n",a1,size_x,size_y,a4);
 print_she(strout);
 fclose(strout);
}
```

Программа, восстанавливающая форму шестерни на основе базовых паратров

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define MAXLEN 6000
#define CODELEN 10000
#define REPEATS 100
#define FILELEN 5714
signed char** arr;
 long size_x,size_y;
main(int argn,char* argv[])
{
 char* string;
 char* outstr;
 FILE* stream;
 FILE* streamout;
 long x,y;
 long a1,a4;
 string=calloc(MAXLEN,sizeof(char));
 outstr=calloc(CODELEN,sizeof(char));
 if(argn<3)
  { fprintf(stderr,"usage: %s filein fileout\n",argv[0]);exit(1); }
 stream=fopen(argv[1],"rt");
 streamout=fopen(argv[2],"wt");
 fscanf(stream,"%ld%ld%ld%ld",&a1,&size_x,&size_y,&a4);
 arr=calloc(MAXLEN,sizeof(signed char*));
 for(y=0;y<size_y && y<MAXLEN;y++)
  arr[y]=calloc(MAXLEN,sizeof(signed char));
 for(y=0;y<size_y;y++)
  {
   fscanf(stream,"%s",string);
   for(x=0;x<size_x;x++)
    {
     (arr[y])[x]=((string[x]=='0')?0:1);
    }
  }
 long xmid,ymid,r;
 float phi;
 xmid=size_x/2+4;
 ymid=size_y/2+4;
 double xavrg=0,yavrg=0,n=0;
 long r1;
//calculate center
 for(phi=0.;phi<3.14159*2.;phi+=0.005)
  {
   for(r1=0;r1<300;r1++)
    {
```

```
      x=xmid+(long)((float)r1*sin(phi));
      y=ymid+(long)((float)r1*cos(phi));
      if(arr[y][x]==0)
       {
        xavrg+=(double)x;
        yavrg+=(double)y;
        n+=(double)1;
       }
    }
  }
 xmid=(long)(xavrg/n);
 ymid=(long)(yavrg/n);
//save paramters
sprintf(outstr,"BEGIN\n");
sprintf(outstr+strlen(outstr),"C %4ld %4ld %4ld %4ld %4d %4d
",a1,size_x,size_y,a4,xmid,ymid);
 r=1700;
//get central hole radius
long max_r=0;
 for(phi=0.;phi<3.14159*2.;phi+=0.005)
  {
   for(r1=0;r1<r;r1++)
    {
      x=xmid+(long)((float)r1*sin(phi));
      y=ymid+(long)((float)r1*cos(phi));
      if(arr[y][x]==1)
       {
        if(r1-1>max_r)
         {
          max_r=r1-1;
         }
        break;
       }
    }
  }
//save radius
sprintf(outstr+strlen(outstr),"%4d C\n",max_r);
sprintf(outstr+strlen(outstr),"H ");

//save code shape
 for(phi=0.;phi<3.14159*2.;phi+=0.01)
  {
      x=xmid+(long)((float)r*sin(phi));
      y=ymid+(long)((float)r*cos(phi));
      sprintf(outstr+strlen(outstr),"%1d",(int)(arr[y][x]));
  }
sprintf(outstr+strlen(outstr)," H\nT ");

//save tooth shape
 for(phi=0.;phi<3.14159*2.;phi+=0.005)
  {
   for(r1=size_x/2;r1>r;r1--)
    {
      x=xmid+(long)((float)r1*sin(phi));
      y=ymid+(long)((float)r1*cos(phi));
      if(x>=0 && x<size_x &&
         y>=0 && y<size_y &&
         arr[y][x]==1)
       {
        sprintf(outstr+strlen(outstr),"%3d ",(r1 - 2400));
        break;
       }
    }
  }
```

```
sprintf(outstr+strlen(outstr),"T\nEND");
long i;
//copy multiple
for(i=0;i<REPEATS;i++)
 fprintf(streamout,"%s",outstr);
fclose(streamout);
}
```