ФГБОУ ВО
Национальный исследовательский университет «МЭИ»
Институт радиотехники и электроники им. В.А. Котельникова
Кафедра радиотехнических систем

Курсовой проект

по дисциплине «Аппаратура потребителей спутниковых радионавигационных систем»

«Расчет траектории движения спутника GPS по данным с демодулятора его сигнала»

Группа: ЭР-15-17

ФИО студента: Танкина А.М.

ФИО преподавателя: Корогодин И.В.

Москва 2022

РЕФЕРАТ

Курсовой проект по теме «Расчет траектории движения спутника GPS по данным с демодулятора его сигнала» на первом этапе содержит 14 страниц текстового документа, 4 рисунка, 1 приложение, 3 использованных источника.

Цель проекта – разработка модулей разбора навигационного сообщения GPS и расчета положения спутника, предназначенных для использования в составе навигационного приемника.

В рамках курсового проекта были поставлены задачи:

- разработка модуля разбора символов навигационного сообщения;
- расчет положения КА в Matlab/Python и его проверка сторонними сервисами;
- реализация модуля расчета положения КА на C/C++ и его тестирование.

Конечная цель всего курсового проекта – получить библиотеку функций на Си++, позволяющую рассчитывать положение спутника GPS по данным с демодулятора его сигнала L1 C/A. На первом этапе был реализован модуль разбора навигационного сообщения до структуры эфемерид и проведено сравнение результатов со сторонней программой.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Спутниковая система навигации – комплексная электронно-техническая система, состоящая из совокупности наземного и космического оборудования, предназначенная для определения местоположения (географических координат и высоты), а также параметров движения (скорости и направления движения) для наземных, водных и воздушных объектов. Основными элементами спутниковой системы являются:

- Орбитальная группировка, состоящая из нескольких (от 2 до 30) спутников, излучающих специальные радиосигналы; наземная система управления и контроля, включающая блоки измерения текущего положения спутников и передачи на них полученной информации для корректировки информации об орбитах;
- Приемное клиентское оборудование, используемое для определения координат;
- Информационная радиосистема для передачи пользователям поправок, позволяющих значительно повысить точность определения координат.

Принцип работы спутниковых систем навигации основан на измерении расстояния от антенны на объекте (координаты которого необходимо получить) до спутников, положение которых известно с большой точностью. Таблица положений всех спутников называется альманахом, которым должен располагать любой спутниковый приемник до начала измерений. Обычно приемник сохраняет альманах в памяти со времени последнего выключения и если он не устарел – мгновенно использует его. Каждый спутник передает в своём сигнале весь альманах. Таким образом, зная расстояния до нескольких спутников системы, с помощью обычных геометрических построений, на основе альманаха, можно вычислить положение объекта в пространстве.

Навигационные спутники передают два вида данных — альманах и эфимерис. Данные эфимериса содержат очень точные корректировки параметров орбит и часов для каждого спутника, что требуется для точного

определения координат. Каждый навигационный спутник передает данные только своего собственного эфимериса. Первый этап курсового проекта нацелен на разработку модуля разбора навигационного сообщения до структуры эфемерид.

1 Обработка логов навигационного приемника

1.1 Задание

В неизвестной локации установлен навигационный приемник, принимающий сигналы GPS L1C/A и логирующий результаты этого приема в формате NVS BINR. Собранный на пятиминутном интервале файл приложен в архиве под именем BINR.bin. Файл содержит наблюдения псевдодальностей и прочих радионавигационных параметров, демодулированные и разобранные данные навигационного сообщения.

Для удобства студентов данные демодулятора продублированы в текстовый файл in.txt. Каждая строка файла содержит данные одного сабфрейма одного навигационного сигнала в формате:

1 0 013 0R GpsL1CA 13 212130404 29 125 53 1000101110...

где 13 - номер спутника, 212130404 - счетчик сабфреймов в сигнале, 53 - ID сабфрейма в навигационном сообщении, где в первых трех битах содержится номер сабфрейма в фрейме (5 в данном примере), а далее - номер фрейма в сообщении (6 в данном примере), 1000101110... символы с демодулятора в порядке возрастания времени слева направо.

Требуется:

1. Разработать программу, обрабатывающую файл in.txt и выводящую в файл out.txt таблицу эфемерид для спутника согласно варианту в заданном формате.
2. Обработать файл BINR.bin с помощью программы RTKNAVI из состава RTKLIB. Определить день и место проведения наблюдений, значения эфемерид для спутника согласно номеру варианта (меню открывается в левом нижнем углу экрана по нажатию на квадрат)
3. Сравнить полученные таблицы
4. Оформить код программы и разместить на Github
5. Оформить отчет по этапу и разместить на Github

6. Завести Pull Request

Программа должна компилироваться gcc, все входные данные брать из in.txt, весь вывод осуществлять в out.txt.

1.2 Разработка программы обработки исходного файла и вывода таблицы эфемерид

Для решения поставленной задачи необходимо подгрузить в разрабатываемую программу файл исходных данных, содержимое которого представлено на рисунке 1.

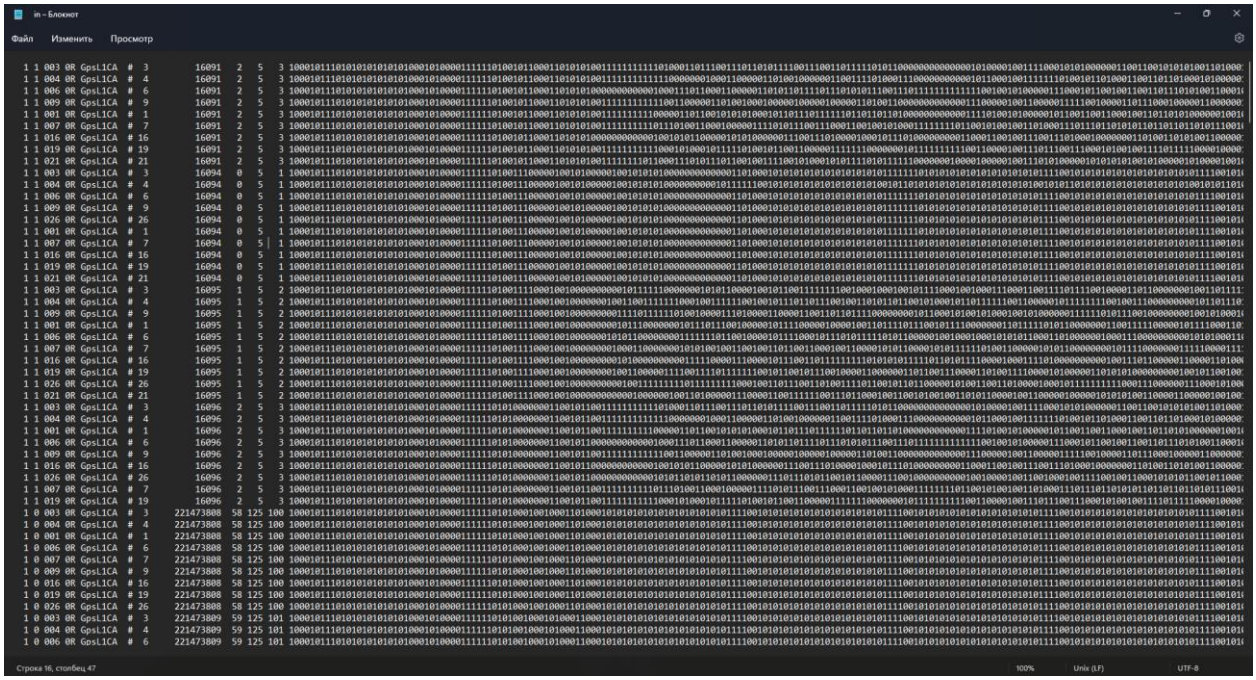

Рисунок 1 – Содержимое файла in.txt

Из этого файла нас интересуют данные, исходящие от девятого спутника. Информация об эфемеридах содержится в первых трех сабфреймах передаваемого сообщения, поэтому дальнейшая работа будет осуществляться со структурой WorkingSub, в которую производится извлечение этих данных.

Из полученной структуры декодируются параметры орбиты и ухода часов с учетом знаковости величины, ее формы представления, а также местоположения описывающих ее бит.

Также производится вывод полученных значений и сохранение их в требуемом формате в файл out.txt. На рисунке 2 представлены результаты работы программы, скомпилированной с помощью командной строки.

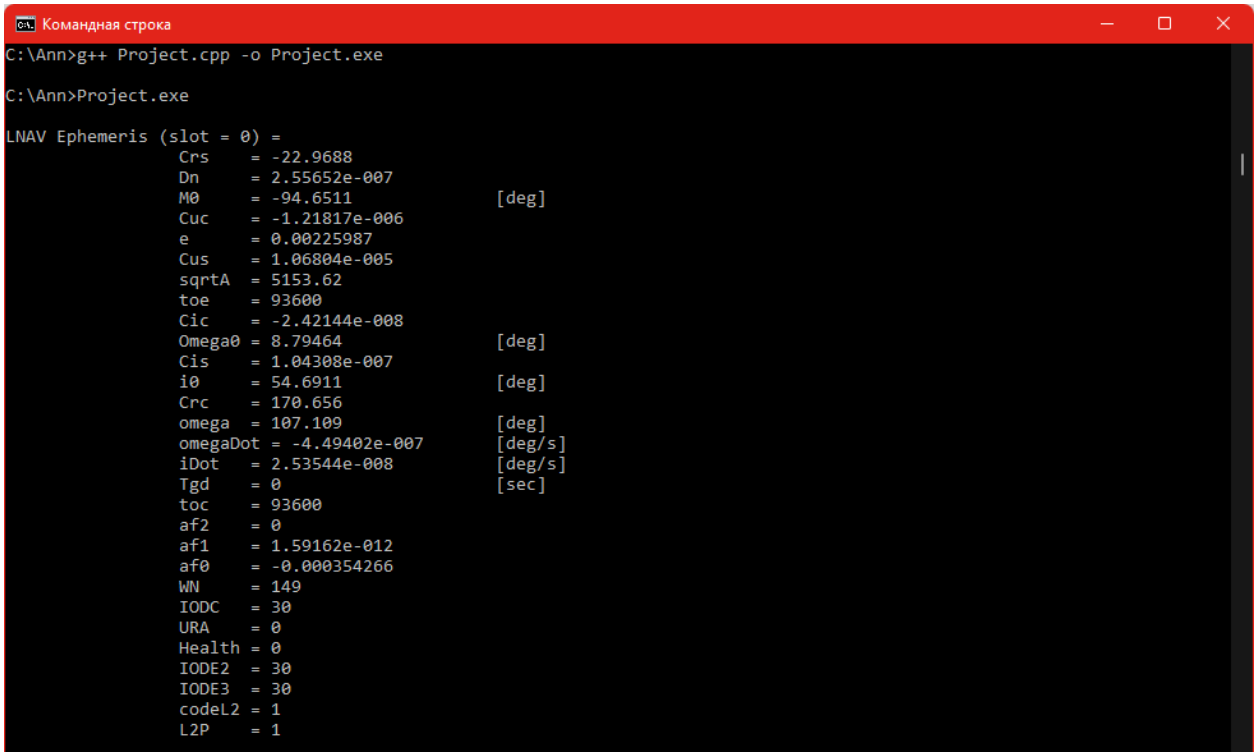

Рисунок 2 – Результаты работы программы

Листинг разработанной программы приведен в приложении.

1.3 Сравнение результатов с таблицей из программы RTKNAVI

Обработаем файл BINR.bin с помощью программы RTKNAVI из состава RTKLIB. Определим значения эфемерид для спутника №9 (рисунок 3) и сравним их с рассчитанными значениями (рисунок 4).

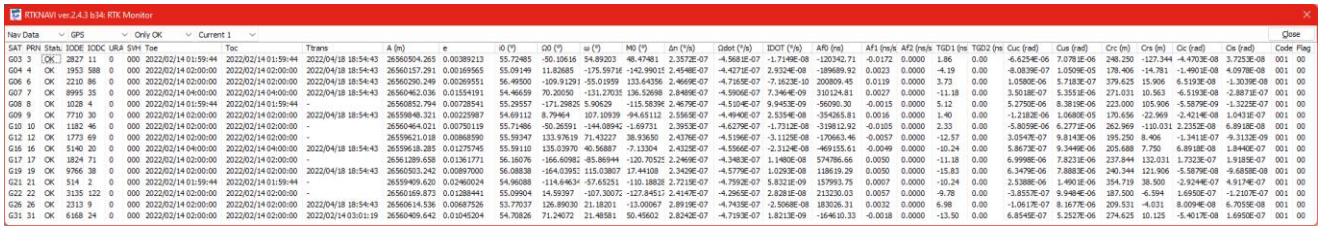

Рисунок 3 – Эфемериды из программы RTKNAVI

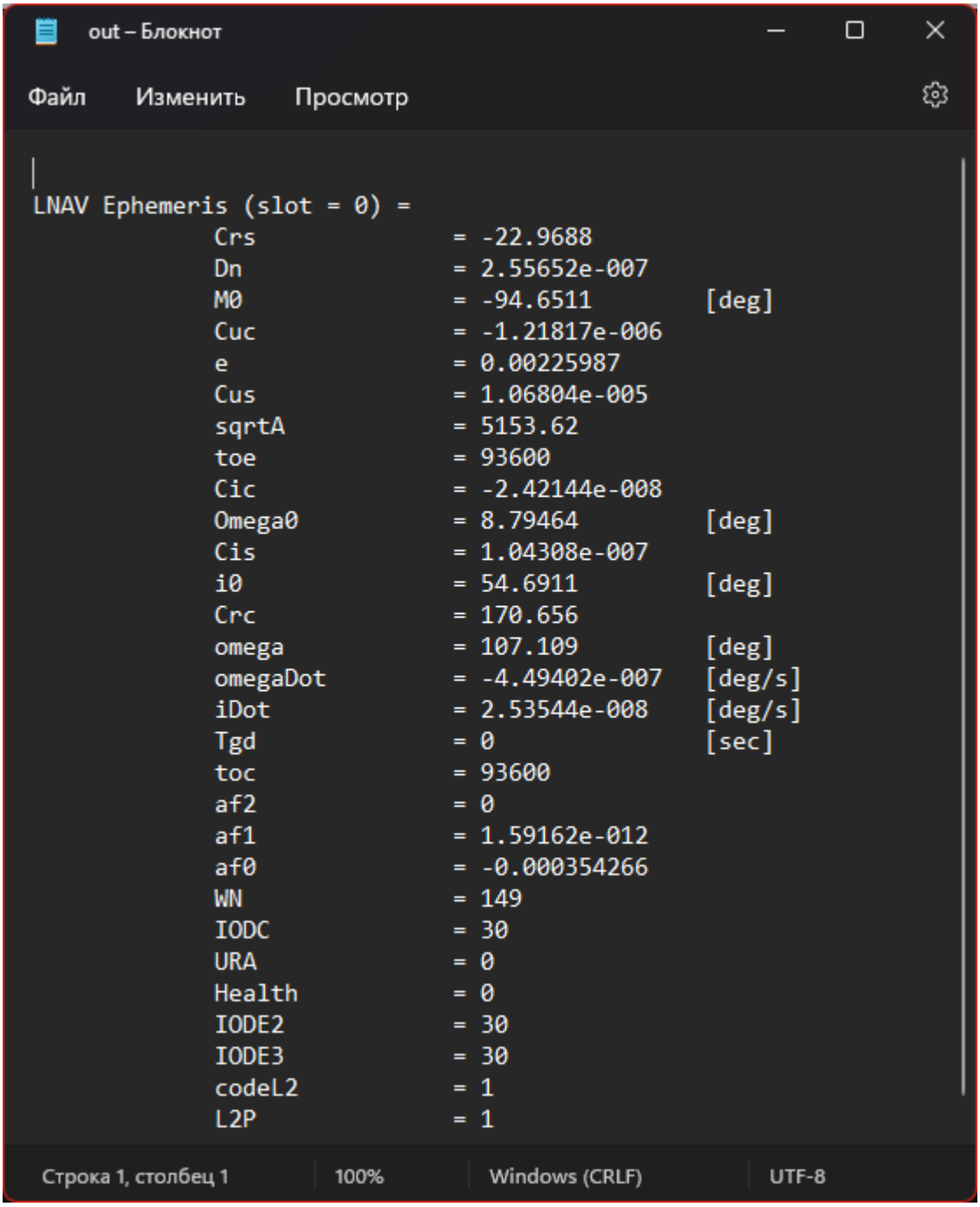

Рисунок 4 – Эфемериды разработанной программы

Из приведенных выше рисунков видно, что значения эфемерид, полученные сторонней программой, сходятся с результатами, полученными с помощью разработанной программы, что свидетельствует о ее корректной работе.

ЗАКЛЮЧЕНИЕ

На первом этапе курсового проекта была разработана программа, обрабатывающая данные демодулятора из файла in.txt и выводящая в файл out.txt таблицу эфемерид требуемого спутника.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2017. Структура и правила оформления отчета о научно-исследовательской работе.
2. Р. Лафоре Объектно-ориентированное программирование в C++. - 4-е изд. - Москва: Питер, 2004. - 923 с.
3. Interface Control Contractor: SAIC (GPS SEI) 200 N. Pacific Coast Highway, Suite 1800 El Segundo, CA 90245.

ПРИЛОЖЕНИЕ А

Ниже приведен листинг программы, разработанной в ходе выполнения первого этапа данного курсового проекта.

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h>
#include <cmath>
#include <stdio.h>

#define Semi_circles    180
#define SF1         pow(2,-5)
#define SF2         pow(2,-43)
#define SF3         pow(2,-31)
#define SF4         pow(2,-29)
#define SF5         pow(2,-33)
#define SF6         pow(2,-19)
#define SF7         pow(2,4)
#define SF8         pow(2,-55)

using namespace std;
struct Ephemeris
{
    float    Crs;
    float    Dn;
    float    M0;
    float    Cuc;
    float    e;
    float    Cus;
    float    sqrtA;
    uint32_t toe;
    float    Cic;
    double   Omega0;
    float    Cis;
    float    i0;
    float    Crc;
    float    omega;
    float    OmegaDot;
    float    iDot;
    int16_t  Tgd;
    uint32_t toc;
    float    af2;
    float    af1;
    float    af0;
    uint16_t WN;
    uint16_t IODC;
    uint32_t URA;
    uint32_t Health;
    uint16_t IODE2;
    uint16_t IODE3;
    bool     codeL2;
    bool     L2P;
    uint32_t slot;
};

struct WorkingSub
{
    string sf1;
    string sf2;
    string sf3;
    uint32_t slot;
};

void Extraction(WorkingSub *Subframes);
void sf2eph(Ephemeris* ep, WorkingSub *data);
int32_t  str2uint(string sf, int32_t Begin, int End);
void printEPH(Ephemeris* ep);
void saveEPH(Ephemeris* ep);

int main(void)
{
    WorkingSub data;
    Extraction(&data);
    Ephemeris *ep = (Ephemeris*) calloc(1, sizeof(Ephemeris));
    sf2eph(ep,&data);
    printEPH(ep);
    saveEPH(ep);
    free(ep);
}

void Extraction(WorkingSub *Subframes)
{
    string path = "in.txt";
    ifstream fin;
    fin.open(path);

    if(fin.is_open()) {

        while (!fin.eof()) {

            uint32_t slot;
            uint32_t slot_SF1 = 0;
            uint32_t slot_SF2 = 0;
            uint32_t slot_SF3 = 0;
            uint32_t subFrameNum;
            int svStr;
            int svNum = 9;
            string strSF;
            string useless;

            fin >> useless >> useless >> useless >> useless >> useless >> useless >> svStr >> slot >> useless >> useless >> subFrameNum >> strSF;

            if (svStr == svNum and slot>= 604800/6){

                if (subFrameNum == 1)
                {
                    slot_SF1 = slot;
                    Subframes->sf1 = strSF;
                }
                else if (subFrameNum == 2)
                {
                    slot_SF2 = slot;
                    Subframes->sf2 = strSF;

                }
                else if (subFrameNum == 3)
                {
```

```
                    slot_SF3 = slot;
                    Subframes->sf3 = strSF;
                }

                if (slot_SF1 + 1 == slot_SF2 and slot_SF2 + 1 == slot_SF3) {
                    Subframes->slot = slot_SF1;
                    return;
                }
            }
        }
    }
    else
    {
        cout << "Can't open" << endl;
    }
    fin.close();


}

void saveEPH(Ephemeris* ep)
{
    ofstream fout;
    string path = "out.txt";
    fout.open(path);
    if(fout.is_open()) {
        fout << endl << "LNAV Ephemeris (slot = " << ep->slot << ") =" << endl;
        fout << "\t\t Crs\t\t\t= " << ep->Crs << endl;
        fout << "\t\t Dn\t\t\t= " << ep->Dn << endl;
        fout << "\t\t M0\t\t\t= " << ep->M0 << "\t\t[deg]" << endl;
        fout << "\t\t Cuc\t\t\t= " << ep->Cuc << endl;
        fout << "\t\t e\t\t\t= " << ep->e << endl;
        fout << "\t\t Cus\t\t\t= " << ep->Cus << endl;
        fout << "\t\t sqrtA\t\t= " << ep->sqrtA << endl;
        fout << "\t\t toe\t\t\t= " << ep->toe << endl;
        fout << "\t\t Cic\t\t\t= " << ep->Cic << endl;
        fout << "\t\t Omega0\t\t= " << ep->Omega0 << "\t\t[deg]" << endl;
        fout << "\t\t Cis\t\t\t= " << ep->Cis << endl;
        fout << "\t\t i0\t\t\t= " << ep->i0 << "\t\t[deg]" << endl;
        fout << "\t\t Crc\t\t\t= " << ep->Crc << endl;
        fout << "\t\t omega\t\t= " << ep->omega << "\t\t[deg]" << endl;
        fout << "\t\t omegaDot\t\t= " << ep->OmegaDot << "\t[deg/s]" << endl;
        fout << "\t\t iDot\t\t\t= " << ep->iDot << "\t[deg/s]" << endl;
        fout << "\t\t Tgd\t\t\t= " << ep->Tgd << "\t\t\t[sec]" << endl;
        fout << "\t\t toc\t\t\t= " << ep->toc << endl;
        fout << "\t\t af2\t\t\t= " << ep->af2 << endl;
        fout << "\t\t af1\t\t\t= " << ep->af1 << endl;
        fout << "\t\t af0\t\t\t= " << ep->af0 << endl;
        fout << "\t\t WN\t\t\t= " << ep->WN << endl;
        fout << "\t\t IODC\t\t\t= " << ep->IODC << endl;
        fout << "\t\t URA\t\t\t= " << ep->URA << endl;
        fout << "\t\t Health\t\t= " << ep->Health << endl;
        fout << "\t\t IODE2\t\t= " << ep->IODE2 << endl;
        fout << "\t\t IODE3\t\t= " << ep->IODE3 << endl;
        fout << "\t\t codeL2\t\t= " << ep->codeL2 << endl;
        fout << "\t\t L2P\t\t\t= " << ep->L2P << endl;
    }
    else
    {
        cout << "Cant open" << endl;
    }
    fout.close();

}

void printEPH(Ephemeris* ep)
{
    cout << endl << "LNAV Ephemeris (slot = " << ep->slot << ") =" << endl;
    cout << "\t\t Crs    = " << ep->Crs <<  endl;
    cout << "\t\t Dn     = " << ep->Dn <<  endl;
    cout << "\t\t M0     = " << ep->M0 << "\t\t[deg]" << endl;
    cout << "\t\t Cuc    = " << ep->Cuc <<  endl;
    cout << "\t\t e = " << ep->e <<  endl;
    cout << "\t\t Cus    = " << ep->Cus <<  endl;
    cout << "\t\t sqrtA = " << ep->sqrtA <<  endl;
    cout << "\t\t toe    = " << ep->toe <<  endl;
    cout << "\t\t Cic    = " << ep->Cic <<  endl;
    cout << "\t\t Omega0    = " << ep->Omega0 << "\t\t[deg]" << endl;
    cout << "\t\t Cis    = " << ep->Cis <<  endl;
    cout << "\t\t i0     = " << ep->i0 << "\t\t[deg]" << endl;
    cout << "\t\t Crc    = " << ep->Crc <<  endl;
    cout << "\t\t omega = " << ep->omega << "\t\t[deg]" << endl;
    cout << "\t\t omegaDot = " << ep->OmegaDot << "\t[deg/s]" << endl;
    cout << "\t\t iDot   = " << ep->iDot << "\t\t[deg/s]" << endl;
    cout << "\t\t Tgd    = " << ep->Tgd << "\t\t\t[sec]" <<  endl;
    cout << "\t\t toc    = " << ep->toc <<  endl;
    cout << "\t\t af2    = " << ep->af2 <<  endl;
    cout << "\t\t af1    = " << ep->af1 <<  endl;
    cout << "\t\t af0    = " << ep->af0 <<  endl;
    cout << "\t\t WN     = " << ep->WN <<  endl;
    cout << "\t\t IODC   = " << ep->IODC <<  endl;
    cout << "\t\t URA    = " << ep->URA <<  endl;
    cout << "\t\t Health     = " << ep->Health <<  endl;
    cout << "\t\t IODE2 = " << ep->IODE2 <<  endl;
    cout << "\t\t IODE3 = " << ep->IODE3 <<  endl;
    cout << "\t\t codeL2     = " << ep->codeL2 <<  endl;
    cout << "\t\t L2P    = " << ep->L2P <<  endl;

}

int32_t str2uint(string sf, int32_t Begin, int End)
{
    int32_t  ans = 0;
    for (int i = Begin; i < End; i++) {
        bool bit = (sf[i - 1] == '1');
        ans = ans | bit;
        if (i < End-1){
            ans = ans<<1;
        }
    }
    return ans;
}

int32_t compl2int(uint32_t ans, int Lenght)
```

```
{
    int32_t Result = 0;
    int32_t Mask = 0;
    if (Lenght < 32){
        if (bool((1<<Lenght-1) & ans)){
            for (int i = 0; i < 32 - Lenght + 1; i++) {
                Mask |= 0x80000000 >> i;
            }
            ans |= Mask;
            Result = ~(ans-1);
            return -Result;
        }
    }
    if (Lenght == 32){
        if (bool((1<<31) & ans)){
            Result = ~(ans-1);
            return -Result;
        }
    }
    return ans;
}

uint32_t splitstr2uint(string sf, uint16_t Begin, int End, uint16_t Contin, int End_of_Contin)
{
    uint32_t ans = 0;
    for (int i = Begin; i < End; i++) {
        bool bit = (sf[i - 1] == '1');
        ans = ans | bit;
        ans = ans<<1;
    }
    for (int i = Contin; i < End_of_Contin; i++) {
        bool bit = (sf[i - 1] == '1');
        ans = ans | bit;
        if (i < End_of_Contin-1){
            ans = ans<<1;
        }
    }
    return ans;
}

void sf2eph(Ephemeris* ep, WorkingSub *data)
{
    ep->slot = data->slot;

    ep->Crs = compl2int(str2uint(data->sf2,69,85),16)*SF1;

    ep->Dn = compl2int(str2uint(data->sf2,91,107),16)*SF2*Semi_circles;

    ep->M0 = compl2int(splitstr2uint(data->sf2,107, 115, 121, 145),32)*SF3*Semi_circles;

    ep->Cuc = compl2int(str2uint(data->sf2,151,167),16)*SF4;

    ep->e = splitstr2uint(data->sf2,167, 175, 181, 205)*SF5;

    ep->Cus = compl2int(str2uint(data->sf2,211,227),16)*SF4;

    ep->sqrtA = splitstr2uint(data->sf2,227, 235, 241, 265)*SF6;

    ep->toe = str2uint(data->sf2,271,287)*SF7;

    ep->Cic = compl2int(str2uint(data->sf3,61,77),16)*SF4;

    ep->Omega0 = compl2int(splitstr2uint(data->sf3,77, 85, 91, 115),32)*SF3*Semi_circles;

    ep->Cis = compl2int(str2uint(data->sf3,121,137),16)*SF4;

    ep->i0 = compl2int(splitstr2uint(data->sf3,137, 145, 151, 175),32)*SF3*Semi_circles;

    ep->Crc = compl2int(str2uint(data->sf3,181,197),16)*SF1;

    ep->omega = compl2int(splitstr2uint(data->sf3,197, 205, 211, 235),32)*SF3*Semi_circles;

    ep->OmegaDot = compl2int(str2uint(data->sf3,241,265),24)*SF2*Semi_circles;

    ep->iDot = compl2int(str2uint(data->sf3,279,293),14)*SF2*Semi_circles;

    ep->Tgd = compl2int(str2uint(data->sf1,197,205),8)*SF3;

    ep->toc = compl2int(str2uint(data->sf1,219,235),16)*SF7;

    ep->af2 = compl2int(str2uint(data->sf1,241,249),8)*SF8;

    ep->af1 = compl2int(str2uint(data->sf1,249,265),16)*SF2;

    ep->af0 = compl2int(str2uint(data->sf1,271,293),22)*SF3;

    ep->WN = str2uint(data->sf1,61,71);

    ep->IODC = splitstr2uint(data->sf1,83, 85, 211, 219);

    ep->URA = str2uint(data->sf1,73,75);

    ep->Health = ep->IODE2 = str2uint(data->sf1,73,79);

    ep->IODE2 = str2uint(data->sf2,61,69);

    ep->IODE3 = str2uint(data->sf3,271,279);

    ep->codeL2 = bool (data->sf1[91]);

    ep->L2P = bool (data->sf1[90]);
}
```