

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»

Выполнил: студ. Бахолдин Н.В.

Группа: ЭР-15-17

Проверил: доц. Корогодин И.В.

Москва 2022

Оглавление

Цель работы	3
Дано	3
1. Обработка логов навигационного приемника	3
2. Моделирование и расчет траектории движения спутника	5
3. Реализация модуля расчета координат	10
3.1 Задание	10
3.2 Разработка алгоритма расчета положения спутника на языке C++	11
Вывод.....	12
Литература	13
Приложение.....	14
Приложение к п.1	14
Приложение к п.2.....	21
Приложение к п.3.....	24

Цель работы

Изучение особенностей сигналов спутников GPS для определения положения спутника по данным с демодулятора его сигнала L1 C/A. На первом этапе происходит моделирование модуля разбора навигационного сообщения до структуры эфемерид.

Дано

Файл “in.txt” с записанными в нем данными, зафиксированными навигационным приемником по сигналу GPS L1C/A. Файл содержит наблюдения псевдодальностей и прочих радионавигационных параметров, демодулированные и разобранные данные навигационного сообщения.

1. Обработка логов навигационного приемника

На первом этапе необходимо создать программу в среде C/ C++, выполняющую функции аналогичные модулю разбора навигационного сообщения. Листинг созданной части программы приведен в приложении.

На данном этапе реализован расчет всех параметров спутника.

```
Crs = -127.344
Dn = 2.35721e-007
M0 = 48.4748 [deg]
Cuc = -6.62543e-006
e = 0.00389213
Cus = 7.07805e-006
sqrtA = 5153.69
toe = 93584
Cic = -4.47035e-008
Omega0 = -50.1062 [deg]
Cis = 3.72529e-008
i0 = 55.7249 [deg]
Crc = 248.25
omega = 54.892 [deg]
omeDot = -4.5681e-007 [deg/s]
iDot = -1.71485e-008 [deg/s]
Tgd = 0 [sec]
toc = 93584
af2 = 0
af1 = -1.71667e-011
af0 = -0.000120343
WN = 149
IODC = 11
URA = 0
Health = 0
IOD2 = 11
IOD3 = 11
codeL2 = 1
L2P = 1
```

Рисунок 1 – Данные, полученные в ходе компиляции кода через командную строку

Таблица 1 – Данные, полученные в программе RTKNAVI

IODE	2827
IODC	11
URA	0
Toe	2022/02/14 01:59:44

Toc	2022/02/14 01:59:44
A (\sqrt{A})	26560504.265 (5153.69)
e	0.00389213
I0	55.72485
$\Omega 0$	-50.10616
ω	54.89203
M0	48.47481
Δn	2.3572e-07
Ωdot	-4.5681e-07
IDOT	-1.7149e-08
Af0	-120342.71
Af1	-0.0172
Af2	0.0
Cuc	-6.6254e-06
Cus	7.0781e-06
Crc	248.250
Crs	-127.344
Cic	-4.4703e-08
Cis	3.7253e-08
Code	001

Вывод: Данные, полученные в программе RTKNAVI совпадают с данными, полученными в ходе компиляции кода. Первый этап курсового проекта выполнен.

2. Моделирование и расчет траектории движения спутника

Задание

Эфемериды - параметры некоторой модели движения спутника. В разных ГНСС эти модели разные, а значит отличается и формат эфемерид, и алгоритмы расчета положения спутника.

Одна из самых простых и удобных моделей - в системе GPS.

Требуется реализовать на языке Matlab или Python функцию расчета положения спутника GPS на заданный момент по шкале времени UTC. В качестве эфемерид использовать данные, полученные на предыдущем этапе.

- Построить трехмерные графики множества положений спутника GPS с системным номером, соответствующим номеру студента по списку. Графики в двух вариантах: в СК ECEF WGS84 и соответствующей ей инерциальной СК. Положения должны соответствовать суточному интервалу на дне формирования наблюдений, определенном на предыдущем этапе. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал.
- Вывести значения координат спутника в файл out.txt в системе ECEF WGS 84 в виде строк: Секунда_от_начала_дня X Y Z
- Используя оценку местоположения с предыдущего этапа, построить Sky Plot за указанный временной интервал и сравнить результат с Trimble GNSS Planning Online.
- Оформить отчет по результатам этапа:
- Реализация в Matlab или Python (описание модели и её листинг)
- Таблица использованных эфемерид
- Трехмерные графики положений спутника в ECEF и ECI (не забудьте подписать оси, изобразите соответствующую Земле сферу в начале СК)
- Расчётный и полученный в GNSS Planning Online SkyView

Решение пункта 2

Запишем время системы в формате GPS. Дата: 14.02.2022 01:59:44. Формат времени GPS:

WN : TOW

WN – номер недели, начиная с 6 января 1980 года, по модулю 1024 (10 бит)

TOW – количество секунд от начала текущей недели

Для заданных даты и времени, формат времени GPS будет содержать следующее значение:

2197 : 93602

$\mu = 3.986005 \times 10^{14} \text{ meters}^3/\text{sec}^2$	WGS 84 value of the earth's gravitational constant for GPS user
$\dot{\Omega}_e = 7.2921151467 \times 10^{-5} \text{ rad/sec}$	WGS 84 value of the earth's rotation rate
$A = \left(\sqrt{A} \right)^2$	Semi-major axis
$n_0 = \sqrt{\frac{\mu}{A^3}}$	Computed mean motion (rad/sec)
$t_k = t - t_{oe}^*$	Time from ephemeris reference epoch
$n = n_0 + \Delta n$	Corrected mean motion
$M_k = M_0 + nt_k$	Mean anomaly
	Kepler's equation ($M_k = E_k - e \sin E_k$) may be solved for Eccentric anomaly (E_k) by iteration:
$E_0 = M_k$	– Initial Value (radians)
$E_j = E_{j-1} + \frac{M_k - E_{j-1} + e \sin E_{j-1}}{1 - e \cos E_{j-1}}$	– Refined Value, minimum of three iterations, (j=1,2,3)
$E_k = E_j$	– Final Value (radians)
$v_k = 2 \tan^{-1} \left(\sqrt{\frac{1+e}{1-e}} \tan \frac{E_k}{2} \right)$	True Anomaly (unambiguous quadrant)
$\Phi_k = v_k + \omega$	Argument of Latitude
$\delta u_k = c_{us} \sin 2\Phi_k + c_{uc} \cos 2\Phi_k$ $\delta r_k = c_{rs} \sin 2\Phi_k + c_{rc} \cos 2\Phi_k$ $\delta i_k = c_{is} \sin 2\Phi_k + c_{ic} \cos 2\Phi_k$	Argument of Latitude Correction Radius Correction Inclination Correction
	} Second Harmonic Perturbations
$u_k = \Phi_k + \delta u_k$	Corrected Argument of Latitude
$r_k = A(1 - e \cos E_k) + \delta r_k$	Corrected Radius
$i_k = i_0 + \delta i_k + (\text{IDOT}) t_k$	Corrected Inclination
$x_k' = r_k \cos u_k$ $y_k' = r_k \sin u_k$	} Positions in orbital plane.
$\Omega_k = \Omega_0 + (\dot{\Omega} - \dot{\Omega}_e) t_k - \dot{\Omega}_e t_{oe}$	Corrected longitude of ascending node.
$x_k = x_k' \cos \Omega_k - y_k' \sin \Omega_k$ $y_k = x_k' \sin \Omega_k + y_k' \cos \Omega_k$ $z_k = y_k' \sin i_k$	} Earth-fixed coordinates.

Рисунок 2 – Алгоритм расчета координат спутника в СК ECEF WGS 84

Для пересчета СК из ECEF WGS 84 в ECI необходимо обратиться к ИКД GPS и использовать формулы перевода из пункта 20.3.3.4.3.3.2.

Результаты расчета:

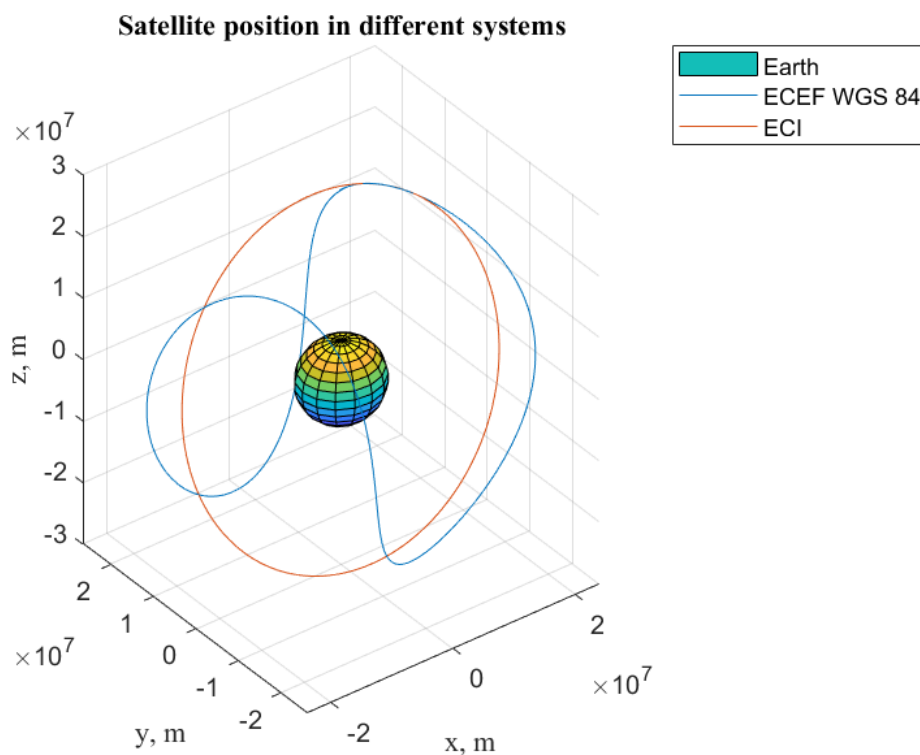


Рисунок 3 – Трехмерные графики положения спутника GPS в различных СК

По рассчитанным координатам были получены широта и долгота, которые использовались в сервисе GNSS Planing Online SkyView. Данные строились на интервале 24 часов от 02:00 14.02.2022 до 02:00 15.02.2022.

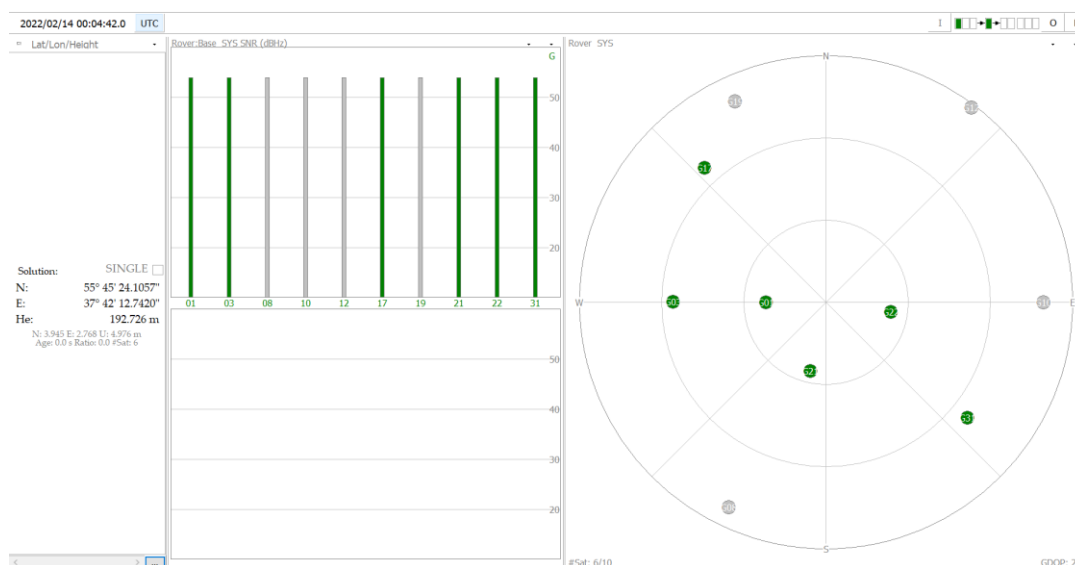


Рисунок 4 – Координаты приемника, полученные при помощи программы RTKNAVI

Settings

Latitude: °

Longitude: °

Height: m

Elevation cutoff: °

Day: Today

Start time: UTC +00:00

Period [hours]:

Time zone:

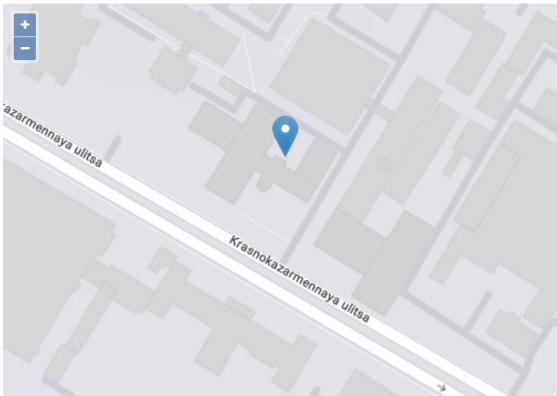


Рисунок 5 – Рисунок настроек в сервисе GNSS Planning Online SkyView

Были получены следующие результаты:

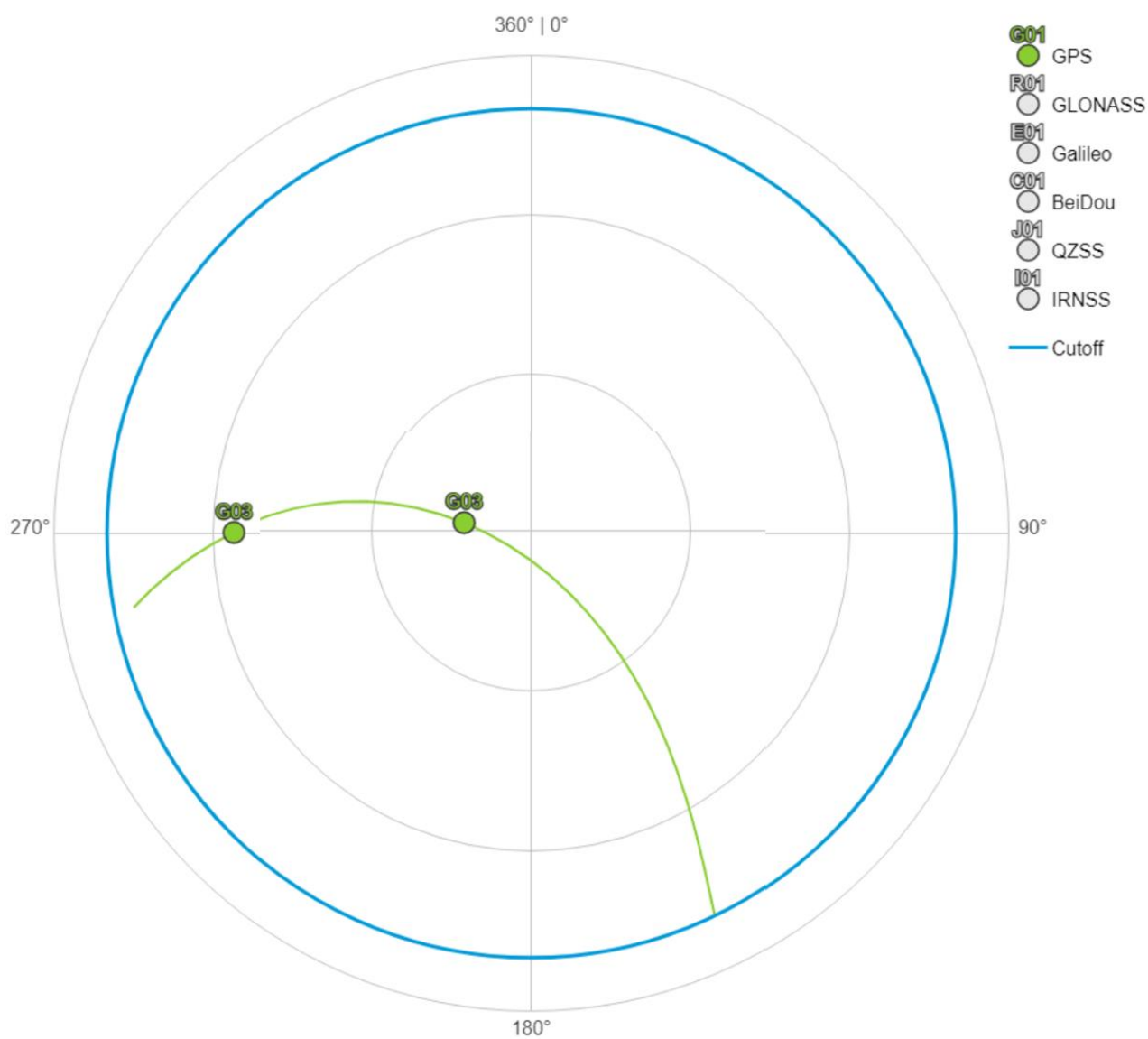


Рисунок 6 – Полученный график SkyView

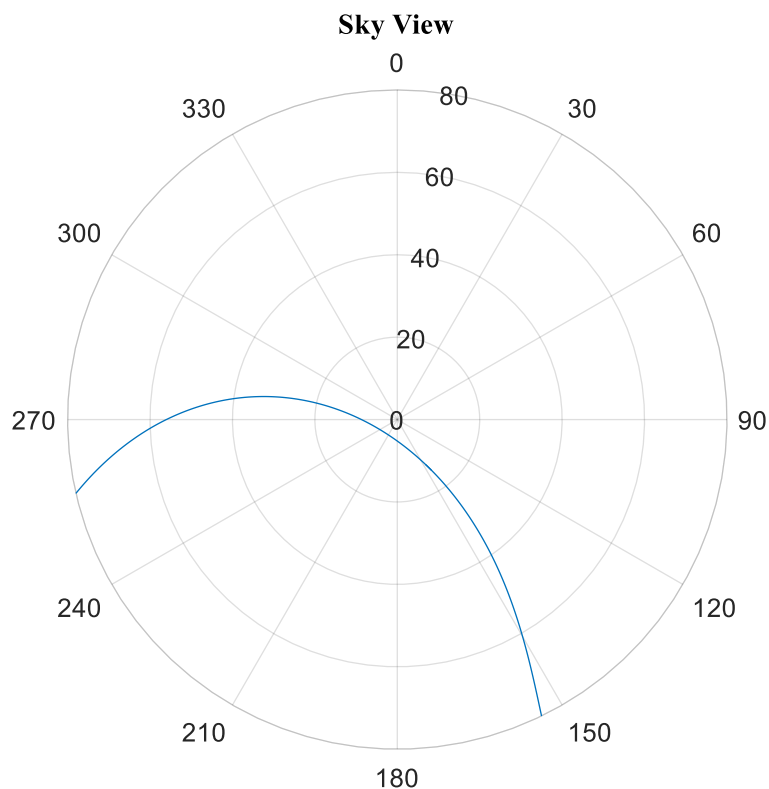


Рисунок 7 – График SkyView

Вывод

В результате выполнения второго этапа КР были получены трехмерные графики положения спутника GPS в различных СК и график SkyView. Из приведенных рисунков 6 и 7 следует, что расчет был проведен верно, траектория движения спутника рассчитана правильно. Согласно заданию, был выведен файл значений координат спутника в системе ECEF WGS 84 output_data.txt в виде строк: «Секунда_от_начала_дня X Y Z». Данный файл сохраняется в директории проекта.

3. Реализация модуля расчета координат

3.1 Задание

Требуется разработать на языке C/C++ функцию расчета положения спутника GPS на заданное время по шкале UTC, минимизируя время её исполнения и количество затрачиваемой оперативной памяти. Вызов функции не должен приводить к выбросу исключений или утечкам памяти при любом наборе входных данных.

Функция расчета положения спутника в Matlab/Python относительно проста, т.к. доступны библиотеки линейной алгебры и решения уравнений. Но при разработке встраиваемого ПО приходится сохранять лицензионную частоту, минимизировать вычислительную нагрузку и затраты памяти. Поэтому отобразить модель из Matlab/Python в прошивку приемника дословно, как правило, не получается. В рассматриваемом примере потребуется, как минимум, выполнить свою реализацию решения трансцендентного уравнения.

Программный модуль должен сопровождаться unit-тестами (например, используя Check):

- Тесты функции решения уравнения Кеплера
- Тест расчетного положения спутника в сравнении с Matlab/Python

Во время второго теста должно вычисляться и выводиться средняя длительность исполнения функции. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал (как на предыдущем этапе).

Требуется провести проверку на утечки памяти (например, с помощью утилиты valgrind).

Оформить отчет по результатам курсового проекта. В качестве первых двух глав использовать отчёты с предыдущих этапов, в третьей главе отразить результаты этого этапа:

1. Код реализации
2. Вывод тестов, включая анализ времени исполнения
3. Вывод проверок на утечку памяти
4. Вывод по этапу
5. Заключение по проекту

Программа должна компилироваться gcc и использовать в качестве входных данных in.txt с первого этапа. Результат должен записываться в out.txt в

строки формата, определенного на втором этапе. При тестировании должны сравниваться файлы out.txt второго и третьего этапов.

3.2 Разработка алгоритма расчета положения спутника на языке C++

Для того, чтобы успешно реализовать выполнение этого этапа, было принято решение продолжить код из пункта 1, так как на первом этапе был реализован программный модуль разбора навигационного сообщения. Одним из требований к КП является сравнение расчетов в двух независимых программах, которыми были выбраны C++ и Matlab.

В ходе было реализовано решение уравнения Кеплера, алгоритм расчета которого представлен на рисунке 8.

```
double M_k = ep->M0 + n * t_k;  
double E_0 = M_k;  
double E_k = 0;  
int k = 0;  
  
while (abs(E_0 - E_k) > 1e-8)  
{  
    E_k = E_0;  
  
    E_0 = E_0 + (M_k - E_0 + ep->e * sin(E_0)) / (1 - ep->e * cos(E_0));  
  
    k = k + 1;  
}
```

Рисунок 8 – Алгоритм решения уравнения Кеплера

Для того, чтобы достичь высокой точности, необходимо реализовать передачу и вывод чисел с высокой точностью (количеством знаков после запятой). Результат расчета предложенного алгоритма представлен на рисунке 9.

```
deviation = 5.4757e-06(m)  
calculation time = 1.195(sec)
```

Рисунок 9 – Результат работы программы

После реализации и отладки кода в программе CLion, была проведена работа по анализу объема потребляемой памяти и нахождения утечки памяти. Данные действия были проведены в программе Microsoft Visual Studio 2022. Результат анализа представлен на рисунке 10.

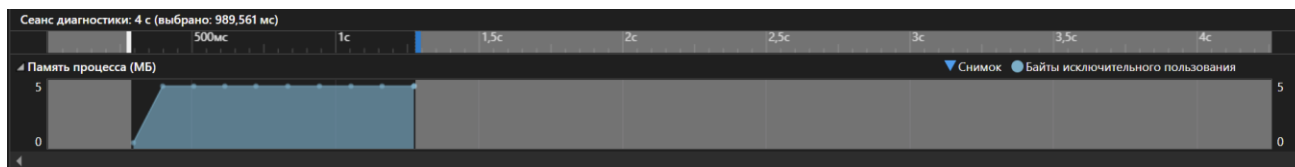


Рисунок 10 – Анализ объема потребляемой памяти

Из рисунка 10 следует, что объем потребляемой в ходе расчета памяти не превышает 5 Мб.

Вывод

На третьем этапе курсового проекта была написана программа на языке C++, что целесообразно при разработке встраиваемого ПО. Было показано, что реализация алгоритмов в C++ и Matlab не приводит к серьезному расхождению в результатах. Объем потребляемой памяти составил 4111,85 КБ, утечка памяти не была обнаружена.

Литература

1. https://gssc.esa.int/navipedia/index.php/GPS_and_Galileo_Satellite_Coordinates_Computation. [Электронный ресурс]. (дата обращения: 24.04.2022 20:00)
2. Interface Specification IS-GPS-200L, August 2020. [Электронный ресурс]. URL: <https://www.gps.gov/technical/icwg/IS-GPS-200L.pdf> (дата обращения: 24.04.2022).
3. <http://www.gnssplanningonline.com/#/skyplot>. [Электронный ресурс]. (дата обращения: 24.04.2022 20:00)
4. <https://planetcalc.ru/8829/>. [Электронный ресурс]. (дата обращения: 24.04.2022 20:00)
5. <http://www.gnssplanningonline.com/#/embedded?satellites=3,38,39,40,41,42,43,44,45,46,47,49,50,51,52,54,55,56,57,58,59,60,61,111,112,113,114,117,201,202,203,204,205,207,208,209,211,212,213,214,215,218,219,221,225,226,227,230,231,233,236,264,265,266,267,268,269,270,272,273,274,275,276,277,279,282,283,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,299,300,302,303,304,305,306,307,308,366,367,368,369,370,371,372,373,374,384,385,386,387,388,389&satSystems=GPS&restoreSats=0&target=settings&cutoffDeg=10&durationHours=24&utcTime=2022-02-14T00:00:00&hgt=200&lonDeg=37.7035394444&latDeg=55.7566960278>.

Приложение

Приложение к п.1

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

#define SF_2E5      pow(2,-5) //добавление  scale factor-a
#define SF_2E55     pow(2,-55)
#define SF_2E43     pow(2,-43)
#define SF_2E31     pow(2,-31)
#define SF_2E4      pow(2,4)
#define SF_2E19     pow(2,-19)
#define SF_2E29     pow(2,-29)
#define SF_2E33     pow(2,-33)
#define SemiCircles 180

using namespace std;

struct Ephemeris {
    float  Crs;
    float  Dn;
    float  M0;
    float  Cuc;
    float  e;
    float  Cus;
    float  sqrtA;
    uint32_t toe;
    float  Cic;
    float  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 sub {    // necessary subframes (string - массив char-ов), достаем первые 3 сабфрейма
    uint32_t slot;
    string sf1;
    string sf2;
    string sf3;
};

void sendStr(sub *Rslot); //объявление функций (нужно для дальнейшей работы main-a)
uint32_t razdel_param(string sf, uint16_t start1, int dlit1, uint16_t start2, int dlit2);
void printEph(Ephemeris* Eph);
void saveEph(Ephemeris* Eph);
void razborsubframes(Ephemeris* Eph, sub *slot);
int32_t sub2data(string sf, int32_t start, int dlit);

int main(void)
{
    sub slot; //этой командой создаем структуру slot и передаем ее дальше
    sendStr(&slot); //& - передача адреса
    cout << slot.sf1 << endl << slot.sf2 << endl << slot.sf3 << endl;
    Ephemeris *Eph = (Ephemeris*) calloc(1, sizeof(Ephemeris));
    razborsubframes(Eph, &slot); //декодирование эфемерид из даты
    printEph(Eph);
    saveEph(Eph);
    free(Eph); //освобождение памяти
}

void sendStr(sub *Rslot)
{
    SetConsoleOutputCP(CP_UTF8); // подключение русского языка в консоль
    string path = "in.txt"; //создание строки путь = название файла

    ifstream fin; //объявляем класс и в стрим (выдуманная функция fin)
    fin.open(path);

    if(fin.is_open()) {
        cout << "Файл открыт." << endl;
        while (!fin.eof()) { //пока не конец файла
            int N = 3;
            string unknown_data; //строка ненужных данных
            int nmbr_stlt; //номер спутника
            uint32_t slot;
            uint32_t subFrameNum;
            string str;

            uint32_t slot_SF1;
            uint32_t slot_SF2;
            uint32_t slot_SF3;

            int mass[N];
            for(int i=0; i<N; ++i)

```

```

    {
        fin >> mass[i]; //из файла закидываем некоторые 3 значения
    }
    fin >> unknown_data >> unknown_data >> unknown_data; //
    fin >> nmbr_stlt >> slot >> unknown_data >> unknown_data >> subFrameNum; //номер спутника -
    слот - 2 мусора - номер кадра
    fin >> str; //строка данных

    if (nmbr_stlt == 3 and slot >= 604800/6) { //задаем номер спутника

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

        }
        else if (subFrameNum == 3)
        {
            slot_SF3 = slot;
            Rslot->sf3 = str; //Rslot = real slot
        }

        if (slot_SF1 + 1 == slot_SF2 and slot_SF2 + 1 == slot_SF3) {
            Rslot->slot = slot_SF1;
            return;
        }

    }

}

}
else
{
    cout << "Ошибка открытия файла!!!" << endl;
}
fin.close();

}

int32_t sub2data(string sf, int32_t start, int dlit) {
    int32_t otvet = 0;
    int32_t Rotvet = 0;
    for (int i = start; i < start+dlit; i++) {
        otvet = (otvet | ((sf[i - 1] == '1') ? 1 : 0)); //перевод текста в цифру
        cout << sf[i-1];
        if (i < start+dlit-1){
            otvet = otvet<<1;
        }
    }

}

```



```

return otvet;

}

int32_t compl2int(uint32_t otvet, int dlit_sub){ //сюда передаем побитовое значение otvet и сколько было
бит и проверяем на дополнение до двух
int32_t Rotvet = 0;
if (dlit_sub == 8){
    if (bool((1<<7) & otvet)){
        otvet |= 0xFFFFF00;
        Rotvet = ~(otvet-1);
        /*cout<< endl << bitset<64>(Rotvet).to_string() << endl;*/
        return -Rotvet;
    }

}

if (dlit_sub == 14){
    if (bool((1<<13) & otvet)){
        otvet |= 0xFFFFC000;
        Rotvet = ~(otvet-1);
        return -Rotvet;
    }

}

if (dlit_sub == 16){
    if (bool((1<<15) & otvet)){//если 16й бит равен 1, тогда выполняем
        otvet |= 0xFFFF0000;//0xFFFF0000 - с 32 по 17й бит единицы (заполняем единицами с 16 го бита
        Rotvet = ~(otvet-1);
        return -Rotvet;
    }

}

if (dlit_sub == 22){
    if (bool((1<<21) & otvet)){
        otvet |= 0xFFC00000;
        Rotvet = ~(otvet-1);
        return -Rotvet;
    }

}

if (dlit_sub == 24){
    if (bool((1<<23) & otvet)){
        otvet |= 0xFF000000;
        Rotvet = ~(otvet-1);
        return -Rotvet;
    }

}

if (dlit_sub == 32){
    if (bool((1<<31) & otvet)){
        otvet |= 0x00000000;
        Rotvet = ~(otvet-1);
        return -Rotvet;
    }

}

```

```

    }
    return otvet;
}
uint32_t razdel_param(string sf, uint16_t start1, int dlit1, uint16_t start2, int dlit2) {
    uint32_t otvet = 0; //возвращаем 32 битное uint dlit1 - how much (выдумал)

    for (int i = start1; i < start1+dlit1; i++) { // преобразование строки в набор бит
        otvet = (otvet | ((sf[i-1] == '1')? 1 : 0)) << 1; // | - или, (функция вида true or false)
    }
    for (int i = start2; i < start2+dlit2; i++) {
        otvet = otvet | ((sf[i-1] == '1')? 1 : 0);
        if (i < start2+dlit2-1) { //чтобы последний раз не сдвигать влево после окончания
            otvet = otvet << 1;
        }
    }

    return otvet;
}

void razborsubframes(Ephemeris* Eph, sub *slot){

    Eph->slot = slot->slot;
    Eph->Crs = compl2int(sub2data(slot->sf2,69,16),16)*SF_2E5;
    Eph->Dn = compl2int(sub2data(slot->sf2,91,16),16)*SF_2E43*SemiCircles;
    Eph->M0 = compl2int(razdel_param(slot->sf2,107, 8, 121, 24),32)*SF_2E31*SemiCircles;
    Eph->Cuc = compl2int(sub2data(slot->sf2,151,16),16)*SF_2E29;
    Eph->e = razdel_param(slot->sf2,167, 8, 181, 24) * SF_2E33;
    Eph->Cus = compl2int(sub2data(slot->sf2,211,16),16)*SF_2E29;
    Eph->sqrtA = razdel_param(slot->sf2,227, 8, 241, 24) * SF_2E19;
    Eph->toe = sub2data(slot->sf2,271,16)*pow(2,4);
    Eph->Cic = compl2int(sub2data(slot->sf3,61,16),16)*SF_2E29;
    Eph->Omega0 = compl2int(razdel_param(slot->sf3,77, 8, 91, 24),32)*SF_2E31*SemiCircles;
    Eph->Cis = compl2int(sub2data(slot->sf3,121,16),16)*SF_2E29;
    Eph->i0 = compl2int(razdel_param(slot->sf3,137, 8, 151, 24),32)*SF_2E31*SemiCircles;
    Eph->Crc = compl2int(sub2data(slot->sf3,181,16),16)*SF_2E5;
    Eph->omega = compl2int(razdel_param(slot->sf3,197, 8, 211, 24),32)*SF_2E31*SemiCircles;
    Eph->OmegaDot = compl2int(sub2data(slot->sf3,241,24),24)*SF_2E43*SemiCircles;
    Eph->iDot = compl2int(sub2data(slot->sf3,279,14),14)*SF_2E43*SemiCircles;
    Eph->Tgd = compl2int(sub2data(slot->sf1,197,8),8)*SF_2E31;
    Eph->toc = compl2int(sub2data(slot->sf1,219,16),16)*SF_2E4;
    Eph->af2 = compl2int(sub2data(slot->sf1,241,8),8)*SF_2E55;
    Eph->af1 = compl2int(sub2data(slot->sf1,249,16),16)*SF_2E43;
    Eph->af0 = compl2int(sub2data(slot->sf1,271,22),22)*SF_2E31;
    Eph->WN = sub2data(slot->sf1,61,10);
    Eph->IODC = razdel_param(slot->sf1,83, 2, 211, 8);
    Eph->URA = sub2data(slot->sf1,73,4);
    Eph->Health = Eph->IODE2 = sub2data(slot->sf1,73,6);
    Eph->IODE2 = sub2data(slot->sf2,61,8);
    Eph->IODE3 = sub2data(slot->sf3,271,8);
    Eph->codeL2 = sub2data(slot->sf1,71,2);
    Eph->L2P = slot->sf1[90];
}

```

```

void saveEph(Ephemeris* Eph)
{
    ofstream fout;
    string path = "out.txt";
    fout.open(path);
    if(fout.is_open()) {

        fout << endl << "LNAV Ephemeris (slot = " << Eph->slot << ") =" << endl;
        fout << "\t\t Crs    = " << Eph->Crs << endl;
        fout << "\t\t Dn      = " << Eph->Dn << endl;
        fout << "\t\t M0       = " << Eph->M0 << "\t\t[deg]" << endl;
        fout << "\t\t Cuc     = " << Eph->Cuc << endl;
        fout << "\t\t e       = " << Eph->e << endl;
        fout << "\t\t Cus     = " << Eph->Cus << endl;
        fout << "\t\t sqrtA   = " << Eph->sqrtA << endl;
        fout << "\t\t toe     = " << Eph->toe << endl;
        fout << "\t\t Cic     = " << Eph->Cic << endl;
        fout << "\t\t Omega0    = " << Eph->Omega0 << "\t\t[deg]" << endl;
        fout << "\t\t Cis     = " << Eph->Cis << endl;
        fout << "\t\t i0       = " << Eph->i0 << "\t\t[deg]" << endl;
        fout << "\t\t Crc     = " << Eph->Crc << endl;
        fout << "\t\t omega    = " << Eph->omega << "\t\t[deg]" << endl;
        fout << "\t\t omeDot    = " << Eph->OmegaDot << "\t\t[deg/s]" << endl;
        fout << "\t\t iDot     = " << Eph->iDot << "\t\t[deg/s]" << endl;
        fout << "\t\t Tgd     = " << Eph->Tgd << "\t\t\t[sec]" << endl;
        fout << "\t\t toc     = " << Eph->toc << endl;
        fout << "\t\t af2     = " << Eph->af2 << endl;
        fout << "\t\t af1     = " << Eph->af1 << endl;
        fout << "\t\t af0     = " << Eph->af0 << endl;
        fout << "\t\t WN      = " << Eph->WN << endl;
        fout << "\t\t IODC    = " << Eph->IODC << endl;
        fout << "\t\t URA     = " << Eph->URA << endl;
        fout << "\t\t Health   = " << Eph->Health << endl;
        fout << "\t\t IODE2    = " << Eph->IODE2 << endl;
        fout << "\t\t IODE3    = " << Eph->IODE3 << endl;
        fout << "\t\t codeL2   = " << Eph->codeL2 << endl;
        fout << "\t\t L2P     = " << Eph->L2P << endl;
    }
    else
    {
        cout << "Ошибка открытия файла!!!" << endl;
    }
    fout.close();
    cout << "Успешно!";
}

```

```

void printEph(Ephemeris* Eph)
{
    cout << endl << "LNAV Ephemeris (slot = " << Eph->slot << ") =" << endl;
    cout << "\t\t Crs    = " << Eph->Crs << endl;
    cout << "\t\t Dn      = " << Eph->Dn << endl;
    cout << "\t\t M0       = " << Eph->M0 << "\t\t[deg]" << endl;
    cout << "\t\t Cuc     = " << Eph->Cuc << endl;
    cout << "\t\t e       = " << Eph->e << endl;
}

```

```

cout << "\t\t Cus      = " << Eph->Cus << endl;
cout << "\t\t sqrtA    = " << Eph->sqrtA << endl;
cout << "\t\t toe      = " << Eph->toe << endl;
cout << "\t\t Cic       = " << Eph->Cic << endl;
cout << "\t\t Omega0 = " << Eph->Omega0 << "\t\t[deg]" << endl;
cout << "\t\t Cis       = " << Eph->Cis << endl;
cout << "\t\t i0        = " << Eph->i0 << "\t\t[deg]" << endl;
cout << "\t\t Crc       = " << Eph->Crc << endl;
cout << "\t\t omega    = " << Eph->omega << "\t\t[deg]" << endl;
cout << "\t\t omeDot = " << Eph->OmegaDot << "\t\t[deg/s]" << endl;
cout << "\t\t iDot      = " << Eph->iDot << "\t\t[deg/s]" << endl;
cout << "\t\t Tgd       = " << Eph->Tgd << "\t\t\t[sec]" << endl;
cout << "\t\t toc       = " << Eph->toc << endl;
cout << "\t\t af2       = " << Eph->af2 << endl;
cout << "\t\t af1       = " << Eph->af1 << endl;
cout << "\t\t af0       = " << Eph->af0 << endl;
cout << "\t\t WN        = " << Eph->WN << endl;
cout << "\t\t IODC      = " << Eph->IODC << endl;
cout << "\t\t URA       = " << Eph->URA << endl;
cout << "\t\t Health    = " << Eph->Health << endl;
cout << "\t\t IODE2     = " << Eph->IODE2 << endl;
cout << "\t\t IODE3     = " << Eph->IODE3 << endl;
cout << "\t\t codeL2    = " << Eph->codeL2 << endl;
cout << "\t\t L2P       = " << Eph->L2P << endl;

```

```

}

```

Приложение к п.2

```
clc
close all
clear
%Константы из ИКД GPS
pi = 3.1415326535898;%отношение длины окружности к ее диаметру (Ratio of a circle)s
circumference to its diameter)
mu = 3.986004418e14;%геоцентрическая гравитационная постоянная
omegaE = 7.2921151467e-5;%средняя угловая скорость Земли
c = 299792458;%скорость света

%Параметры из эфемерид (пункт 1 КР)
i0 = deg2rad(55.7249);%градусов/номинальное наклонение орбиты относительно
экваториальной плоскости
A = 26560504.265 ;%большая полуось
n0 = sqrt(mu/A^3);%расчетное среднее движение (рад/с)
ecc = 0.00389213;%эксцентриситет
omega0 = deg2rad(-50.10616);%долгота восходящего узла
omegaDot = deg2rad(-4.5681e-07);%скорость изменения прямого восхождения
omega = deg2rad(54.89203);%аргумент перигея
M0 = deg2rad(48.47481);%ср. аномалия КА в контр. момент времени
af0 = -120342.71;%поправка времени КА
af1 = -0.0172;%поправка времени КА
t_0e = 93602;%опорное время

%Расчет элементов Кеплера
delta_n = deg2rad(2.3572e-07);
i_dot = deg2rad(-1.7149e-08);
C_uc = -6.6254e-06;
C_us = 7.0781e-06;
C_rc = 248.250;
C_rs = -127.344;
C_ic = -4.4703e-08;
C_is = 3.7253e-08;
n = n0+delta_n;%скорректированное среднее движение
%% Расчет элементов Кеплера (table 20-IV p.106) (ECEF WGS 84 system)
for i = 1:86400%(24 hours in sec, from 01:59:44 14th of February to 01:59:44 15th of
February )

    t = 93584+i+18; %TOW (добавляем 18 секунд координации)
    tk = t-t_0e;%время от опорной эпохи эфемерид

    if tk>302400%проверка условия
        tk = tk-604800;
    else
        tk = tk+604800;
    end

    Mk = M0+n*tk;%средняя аномалия

    E = zeros(1,4);
    E(1,1) = Mk;
    for j = 2:4
        E(1,j) = E(1,j - 1) + (Mk - E(1,j - 1) + ecc*sin(E(1,j - 1)))/(1 -
ecc*cos(E(1,j - 1))); % 3 iterations
        E(1,j-1) = E(1,j);
    end
    E_k = E(1,4);%эксцентрическая аномалия
```

```

chislitel = sqrt(1-ecc^2)*sin(E_k)/(1-ecc*cos(E_k));
znamenatel = (cos(E_k)-ecc)/(1-ecc*cos(E_k));
nu = atan2(chislitel,znamenatel);%истинная аномалия
Fi = nu+omega;%аргумент широты
deltau = C_us*sin(2*Fi)+C_uc*cos(2*Fi);%аргумент скорректированной широты
deltar = C_rs*sin(2*Fi)+C_rc*cos(2*Fi);%радиальная коррекция
deltai = C_is*sin(2*Fi)+C_ic*cos(2*Fi);

uk = Fi+deltau;%скорректированный аргумент широты
rk = A*(1-ecc*cos(E_k))+deltar;%скорректированный радиус
ik = i0 +deltai + i_dot*tk;%скорректированный аргумент наклона
x_shtrih = rk*cos(uk);
y_shtrih = rk*sin(uk);%позиция в орбитальной плоскости

Omega = omega0 + (omegaDot - omegaE)*tk - omegaE*t_0e;%скорректированная широта
восходящего узла
x = x_shtrih*cos(Omega) - y_shtrih*cos(ik)*sin(Omega);
y = x_shtrih*sin(Omega) + y_shtrih*cos(ik)*cos(Omega);
z = y_shtrih*sin(ik);%координаты фазового центра антенны во время t

x_coord(1,i) = x;
y_coord(1,i) = y;
z_coord(1,i) = z;
%% ECI (Earth-Centered Inertial coordinate system 20.3.3.4.3.3.2 p.110)
theta = omegaE*tk;
x1 = x*cos(theta)-y*sin(theta);
y1 = x*sin(theta)+y*cos(theta);
z1 = z;

x1_coord(1,i) = x1;
y1_coord(1,i) = y1;
z1_coord(1,i) = z1;
%% Calculus for polar coordinates (for SkyView)
H = 200;%высота ПРМ
B = deg2rad(55.45241057);%широта (для ПРМ)
L = deg2rad(37.4212742);%долгота (для ПРМ)

[x(i), y(i), z(i)] = ecef2enu(x_coord(1,i), y_coord(1,i),
z_coord(1,i),B,L,H,wgs84Ellipsoid,'radians'); % Отображение координат спутника для ПРМ

if z(i) > 0
rho(i) = sqrt(x(i)^2 + y(i)^2 + z(i)^2);
theta_polar(i) = acos(z(1,i)/rho(i));
if x(i) > 0
phi_polar(i) = -atan(y(i)/x(i))+pi/2;
elseif (x(i)<0)&&(y(i)>0)
phi_polar(i) = -atan(y(i)/x(i))+3*pi/2;
elseif (x(i)<0)&&(y(i)<0)
phi_polar(i) = -atan(y(i)/x(i))-pi/2;
end
else theta_polar(i) = NaN;
rho_polar(i) = NaN;
phi_polar(i) = NaN;
end
end

%% Запись в файл
filename = "C:\Users\Nikita\OneDrive\Рабочий стол\Учеба\10 семестр\АП СРНС
(КП)\output_data.txt";

```

```

out = fopen(filename, 'w+'); % открытие файла на запись
for i = 1:86400
    fprintf(out, '%6.0f %10.8f %10.8f %10.8f \n ', i, x_coord(1,i), y_coord(1,i),
z_coord(1,i)); % запись в файл
end
fclose(out); % закрытие файла
%% Plots
[x_sphere, y_sphere, z_sphere] = sphere(15);
Earth_radius = 6378136;
x_Earth=Earth_radius*x_sphere;
y_Earth=Earth_radius*y_sphere;
z_Earth=Earth_radius*z_sphere;%рисую Землю

figure (1)
subplot (1,1,1)
surf(x_Earth,y_Earth,z_Earth);
hold on
grid on
colormap ('default')
plot3(x_coord(1,:), y_coord(1,:), z_coord(1,:))
plot3(x1_coord(1,:), y1_coord(1,:), z1_coord(1,:))
xlabel('x, m', 'FontName', 'Times New Roman')
ylabel('y, m', 'FontName', 'Times New Roman')
zlabel('z, m', 'FontName', 'Times New Roman')
title('Satellite position in different systems', 'FontName', 'Times New Roman')
legend ('Earth','ECEF WGS 84','ECI')
hold off

figure(2)
ax = polaraxes;
polarplot(ax, phi_polar, rad2deg(theta_polar))
ax.ThetaDir = 'clockwise';
ax.ThetaZeroLocation = 'top';%положение нуля тетты
rlim([0 80]);%ограничение по оси rho
title('Sky View', 'FontName', 'Times New Roman')

```

Приложение к п.3

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h>
#include <cmath>
#include <stdio.h>
#include <windows.h>
#include <ctime> //библиотека для отобр длит вычисл

#define Semi_circles 180
#define pi 3.1415326535898
#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 sub // структ рабочего фрейма
{
    string sf1;
    string sf2;
    string sf3;
```



```

    uint32_t slot;
};

struct Coordinates
{
    double x;
    double y;
    double z;
};

void sendStr(sub *Subframes); //выделение эфемерид
void scale_factor_use(Ephemeris* ep, sub *data); //преобраз эфемерид в необх типы
данных
int32_t convert_eph(string sf, int32_t Begin, int End); //преобраз эфемерид в
необх типы данных
void printEPH(Ephemeris* ep); //вывод в окно знач эфемерид
void saveEPH(Ephemeris* ep); //сохр знач эфемерид
void CalcCoord(Ephemeris* ep, uint32_t t, Coordinates *Position); //расчет коорд по
ИКД

int main(void)
{
    uint32_t begin = clock(); //ставим начало отсчета времени вычислений
    sub data;
    sendStr(&data);
    Ephemeris *ep = (Ephemeris*) calloc(1, sizeof(Ephemeris));
    scale_factor_use(ep, &data);
    printEPH(ep);
    saveEPH(ep);

    uint32_t start = 93584+1+18;
    uint32_t stop = 93602+86400+1;

    Coordinates Position;

    double** coord = new double*[3];

    for (int i = 0; i < 3; i++)
    {
        coord[i] = new double[stop - start];
    }

    double** coord_from_matlab = new double*[3];

    for (int i = 0; i < 3; i++)
    {
        coord_from_matlab[i] = new double[stop - start];
    }

    for (int t = start; t < stop; t++)
    {
        CalcCoord(ep, t, &Position);
        coord[0][t-start] = Position.x;
        coord[1][t-start] = Position.y;
        coord[2][t-start] = Position.z;
    }

    ifstream file("output_data3.txt");
    double sec;
    if (!file.is_open())

```

```

        cout << "Can't open" << endl;
    else {
        for (int t = 0; t < stop - start; t++)
        {
            file >> sec >> coord_from_matlab[0][t] >> coord_from_matlab[1][t] >>
coord_from_matlab[2][t];
        }
        file.close();
    }

    double deviation = 0;
    for (int i = 0; i < 3; i++)
    {
        for (int k = 0; k < stop - start; k++)
        {
            if (abs(coord[i][k] - coord_from_matlab[i][k]) > deviation)
            {
                deviation = abs(coord[i][k] - coord_from_matlab[i][k]);
            }
        }
    }

    delete[] *coord;
    delete[] coord;
    delete[] *coord_from_matlab;
    delete[] coord_from_matlab; //очистка

    uint32_t end = clock(); //конечного времени расчета
    uint32_t calculation_time = end - begin; //расчет времени исполнения

    cout << "deviation = " << deviation << "(m)" << endl;
    cout << "calculation time = " << (double)calculation_time/1000.0 << "(sec)"
<<endl;
    free(ep);
}

void CalcCoord(Ephemeris* ep, uint32_t t, Coordinates *Position)
{
    double Omega_e_dot = 7.2921151467e-5;
    double mu = 3.986005e14;
    double A = pow(ep->sqrta, 2);
    double n_0 = sqrt(mu / pow(A, 3));
    int32_t t_k = t - ep->toe;

    if (t_k > 302400)
    {
        t_k = t_k - 604800;
    }
    else if (t_k < -302400)
    {
        t_k = t_k + 604800;
    }

    double n = n_0 + ep->Dn;

    double M_k = ep->M0 + n * t_k;
    double E_0 = M_k;
    double E_k = 0;
    int k = 0;

    while (abs(E_0 - E_k) > 1e-8)

```

```

{
    E_k = E_0;

    E_0 = E_0 + (M_k - E_0 + ep->e * sin(E_0))/(1 - ep->e * cos(E_0));

    k = k + 1;
}

double nu_k = atan2((sqrt(1 - pow(ep->e,2)) * sin(E_k)/(1 - ep->e *
cos(E_k))), ((cos(E_k) - ep->e)/(1 - ep->e * cos(E_k))));
double Phi_k = nu_k + ep->omega;

double delta_u_k = ep->Cus * sin(2 * Phi_k) + ep->Cuc * cos(2 * Phi_k);
double delta_r_k = ep->CrS * sin(2 * Phi_k) + ep->Crc * cos(2 * Phi_k);
double delta_i_k = ep->Cis * sin(2 * Phi_k) + ep->Cic * cos(2 * Phi_k);
double u_k = Phi_k + delta_u_k;
double r_k = pow(ep->sqrta,2) * (1 - ep->e * cos(E_k)) + delta_r_k;
double i_k = ep->i0 + delta_i_k + ep->iDot * t_k ;

double x_k_sh = r_k * cos(u_k);
double y_k_sh = r_k * sin(u_k);

double Omega_k = ep->Omega0 + (ep->OmegaDot - Omega_e_dot) * t_k -
Omega_e_dot * ep->toe;

double x_k = x_k_sh * cos(Omega_k) - y_k_sh * cos(i_k) * sin(Omega_k);
double y_k = x_k_sh * sin(Omega_k) + y_k_sh * cos(i_k) * cos(Omega_k);
double z_k = y_k_sh * sin(i_k);

Position->x = x_k;
Position->y = y_k;
Position->z = z_k;
}

void sendStr(sub *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 = 3;
            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();
}

int32_t convert_eph(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 splitconvert_eph(string sf, uint16_t Begin, int End, uint16_t Contin,
int End_of_Contin)//когда параметр лежит в разных кадрах
{
    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 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\t= " << ep->Crs << endl;
        fout << "\t\t Dn\t\t\t\t= " << ep->Dn << endl;
        fout << "\t\t M0\t\t\t\t= " << ep->M0 << "\t\t[deg]" << endl;
        fout << "\t\t Cuc\t\t\t\t= " << ep->Cuc << endl;
        fout << "\t\t e\t\t\t\t\t= " << ep->e << endl;
        fout << "\t\t Cus\t\t\t\t= " << ep->Cus << endl;
        fout << "\t\t sqrtA\t\t\t= " << ep->sqrtA << endl;
        fout << "\t\t toe\t\t\t\t= " << ep->toe << endl;
        fout << "\t\t Cic\t\t\t\t= " << ep->Cic << endl;
        fout << "\t\t Omega0\t\t\t= " << ep->Omega0 << "\t\t[deg]" << endl;
        fout << "\t\t Cis\t\t\t\t= " << ep->Cis << endl;
        fout << "\t\t i0\t\t\t\t\t= " << ep->i0 << "\t\t[deg]" << endl;
        fout << "\t\t Crc\t\t\t\t\t= " << ep->Crc << endl;
        fout << "\t\t omega\t\t\t\t= " << ep->omega << "\t\t[deg]" << endl;
        fout << "\t\t omegaDot\t\t\t= " << ep->OmegaDot << "\t[deg/s]" << endl;
        fout << "\t\t iDot\t\t\t\t\t= " << ep->iDot << "\t[deg/s]" << endl;
        fout << "\t\t Tgd\t\t\t\t\t= " << ep->Tgd << "\t\t\t[sec]" << endl;
        fout << "\t\t toc\t\t\t\t\t= " << ep->toc << endl;
        fout << "\t\t af2\t\t\t\t\t= " << ep->af2 << endl;
        fout << "\t\t af1\t\t\t\t\t= " << ep->af1 << endl;
        fout << "\t\t af0\t\t\t\t\t= " << ep->af0 << endl;
        fout << "\t\t WN\t\t\t\t\t= " << ep->WN << endl;
        fout << "\t\t IODC\t\t\t\t\t= " << ep->IODC << endl;
        fout << "\t\t URA\t\t\t\t\t= " << ep->URA << endl;
        fout << "\t\t Health\t\t\t\t= " << ep->Health << endl;
        fout << "\t\t IODE2\t\t\t\t= " << ep->IODE2 << endl;
        fout << "\t\t IODE3\t\t\t\t= " << ep->IODE3 << endl;
        fout << "\t\t codeL2\t\t\t\t= " << ep->codeL2 << endl;
        fout << "\t\t L2P\t\t\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;
}

void scale_factor_use(Ephemeris* ep, sub *data)//yuet scale factor-a
{
    double deg2rad = pi / Semi_circles;

    ep->slot = data->slot;

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

    ep->Dn = compl2int(convert_eph(data->sf2,91,107),16)*SF2*Semi_circles*deg2rad;

    ep->M0 = compl2int(splitconvert_eph(data->sf2,107, 115, 121, 145),32)*SF3*Semi_circles*deg2rad;

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

```

```

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

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

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

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

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

ep->Omega0 = compl2int(splitconvert_eph(data->sf3,77, 85, 91,
115),32)*SF3*Semi_circles*deg2rad;

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

ep->i0 = compl2int(splitconvert_eph(data->sf3,137, 145, 151,
175),32)*SF3*Semi_circles*deg2rad;

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

ep->omega = compl2int(splitconvert_eph(data->sf3,197, 205, 211,
235),32)*SF3*Semi_circles*deg2rad;

ep->OmegaDot = compl2int(convert_eph(data-
>sf3,241,265),24)*SF2*Semi_circles*deg2rad;

ep->iDot = compl2int(convert_eph(data-
>sf3,279,293),14)*SF2*Semi_circles*deg2rad;

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

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

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

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

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

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

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

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

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

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

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

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

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

```