

Министерство высшего образования и науки РФ  
Национальный исследовательский университет «МЭИ»  
Институт радиотехники и электроники им. В.А. Котельникова  
Кафедра радиотехнических систем  
Аппаратура потребителей СРНС

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

Выполнила  
студентка 5 курса  
группы ЭР-15-17  
Солнцева М. К.  
Преподаватель:  
Корогодин И.В.

Москва 2022

## РЕФЕРАТ

Курсовой проект по теме «Расчет траектории движения спутника GPS по данным с демодулятора его сигнала» состоит из 39 страниц текстового документа, 15 рисунков, 3 приложения, 7 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.

## ОГЛАВЛЕНИЕ

РЕФРАТ.....	1
ОГЛАВЛЕНИЕ .....	2
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ.....	3
ВВЕДЕНИЕ.....	4
1 Обработка логов навигационного приемника.....	5
1.1 Задание первое.....	5
1.2 Разработка программы для обработки исходного файла и вывод в файл таблицы эфемерид.....	6
1.3 Сравнение результатов разработанной программы и программы RTKNAVI.....	8
Вывод.....	8
2 Моделирование траектории движения.....	9
2.1 Задание второе.....	9
2.2 Разработка программы расчета положения спутника .....	9
2.3 Результаты моделирования.....	11
2.4 Сравнение результатов моделирования с Trimble GNSS Planning Online .....	13
Вывод.....	15
3 Реализация модуля расчёта координат .....	15
3.1 Задание третье .....	15
3.2 Разработка программы расчета положения спутника .....	16
3.3 Потребляемая память и утечки .....	17
Вывод.....	17
Заключение .....	18
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	19
ПРИЛОЖЕНИЕ А.....	20
ПРИЛОЖЕНИЕ Б.....	26
ПРИЛОЖЕНИЕ В.....	28

## **ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

GPS	Global Positioning System
КА	Космический аппарат
НАП	Навигационная аппаратура потребителя
ГНСС	Глобальная навигационная спутниковая система
НКА	Навигационный космический аппарат
ИКД	Интерфейсный контрольный документ
СК	Система координат

## 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. Современная спутниковая навигация основывается на использовании принципа беззапросных дальномерных измерений между навигационными спутниками и потребителем. Это означает, что потребителю в составе навигационного сигнала передается информация о координатах спутников. Одновременно (синхронно) производятся измерения дальностей до навигационных спутников. Способ измерений дальностей основывается на вычислении временных задержек принимаемого сигнала от спутника по сравнению с сигналом, генерируемым НАП.

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 эфемерис. Данные эфемериса 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 эфемериса. Первый этап курсового проекта 4 нацелен на разработку модуля разбора навигационного сообщения до структуры эфемерид.

# 1 Обработка логов навигационного приемника

## 1.1 Задание первое

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.

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
100010111010101010101000101001011100001100101001011111000101010101010
```

Рисунок 1 – Структура одного сабфрейма одного НКА

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.

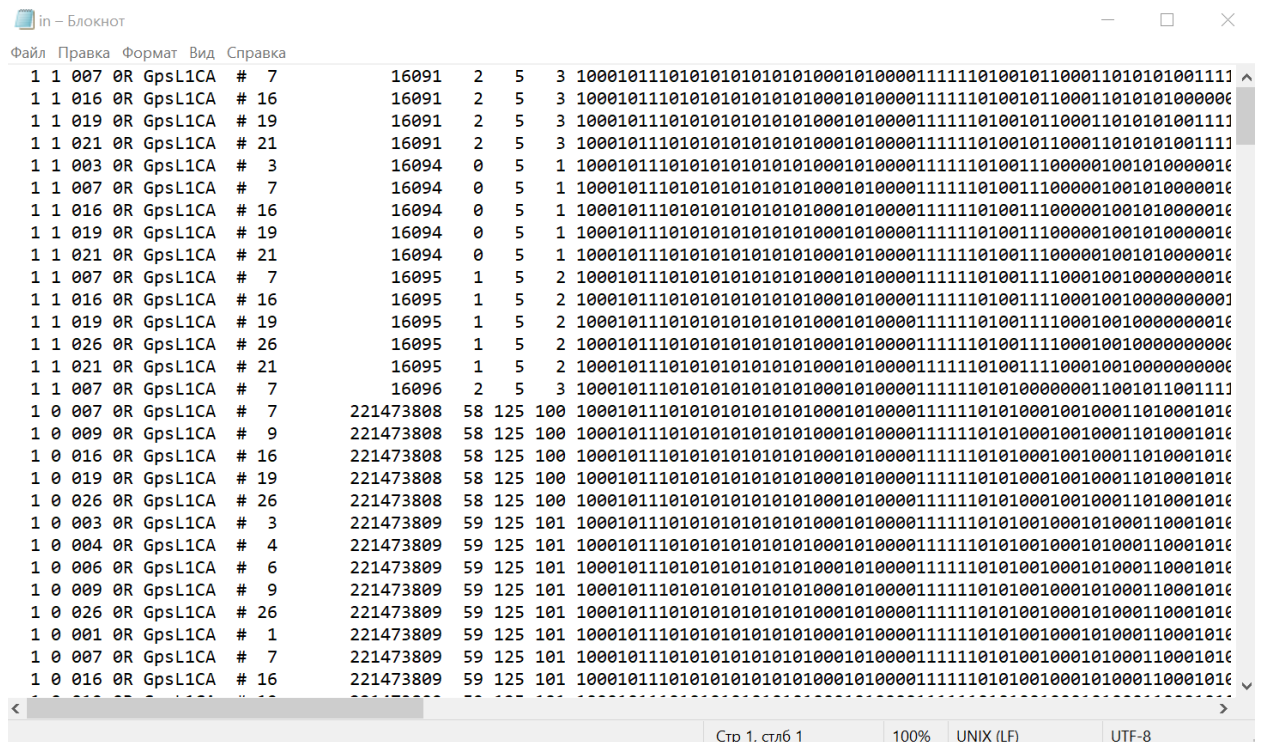
Требуется:

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 RTKLIV. Определить день и место проведения наблюдений, значения эфемерид для спутника согласно номеру варианта.
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 для обработки исходного файла и вывод в файл таблицы эфемерид

Исходные данные в файле in.txt имеют следующий вид:



```
in - Блокнот
Файл  Правка  Формат  Вид  Справка
1 1 007 0R GpsL1CA # 7 16091 2 5 3 1000101110101010101010001010001111110100101100011010101001111
1 1 016 0R GpsL1CA # 16 16091 2 5 3 1000101110101010101010001010001111110100101100011010101000000
1 1 019 0R GpsL1CA # 19 16091 2 5 3 1000101110101010101010001010001111110100101100011010101001111
1 1 021 0R GpsL1CA # 21 16091 2 5 3 1000101110101010101010001010001111110100101100011010101001111
1 1 003 0R GpsL1CA # 3 16094 0 5 1 1000101110101010101010001010001111110100111000001001010000010
1 1 007 0R GpsL1CA # 7 16094 0 5 1 1000101110101010101010001010001111110100111000001001010000010
1 1 016 0R GpsL1CA # 16 16094 0 5 1 1000101110101010101010001010001111110100111000001001010000010
1 1 019 0R GpsL1CA # 19 16094 0 5 1 1000101110101010101010001010001111110100111000001001010000010
1 1 021 0R GpsL1CA # 21 16094 0 5 1 1000101110101010101010001010001111110100111000001001010000010
1 1 007 0R GpsL1CA # 7 16095 1 5 2 1000101110101010101010001010001111110100111100010010000000010
1 1 016 0R GpsL1CA # 16 16095 1 5 2 1000101110101010101010001010001111110100111100010010000000010
1 1 019 0R GpsL1CA # 19 16095 1 5 2 1000101110101010101010001010001111110100111100010010000000010
1 1 026 0R GpsL1CA # 26 16095 1 5 2 1000101110101010101010001010001111110100111100010010000000010
1 1 021 0R GpsL1CA # 21 16095 1 5 2 1000101110101010101010001010001111110100111100010010000000010
1 1 007 0R GpsL1CA # 7 16096 2 5 3 100010111010101010101000101000111111010100000011001011001111
1 0 007 0R GpsL1CA # 7 221473808 58 125 100 1000101110101010101010001010001111110101000100100011010001010
1 0 009 0R GpsL1CA # 9 221473808 58 125 100 1000101110101010101010001010001111110101000100100011010001010
1 0 016 0R GpsL1CA # 16 221473808 58 125 100 1000101110101010101010001010001111110101000100100011010001010
1 0 019 0R GpsL1CA # 19 221473808 58 125 100 1000101110101010101010001010001111110101000100100011010001010
1 0 026 0R GpsL1CA # 26 221473808 58 125 100 1000101110101010101010001010001111110101000100100011010001010
1 0 003 0R GpsL1CA # 3 221473809 59 125 101 1000101110101010101010001010001111110101001000101000110001010
1 0 004 0R GpsL1CA # 4 221473809 59 125 101 1000101110101010101010001010001111110101001000101000110001010
1 0 006 0R GpsL1CA # 6 221473809 59 125 101 1000101110101010101010001010001111110101001000101000110001010
1 0 009 0R GpsL1CA # 9 221473809 59 125 101 1000101110101010101010001010001111110101001000101000110001010
1 0 026 0R GpsL1CA # 26 221473809 59 125 101 1000101110101010101010001010001111110101001000101000110001010
1 0 001 0R GpsL1CA # 1 221473809 59 125 101 1000101110101010101010001010001111110101001000101000110001010
1 0 007 0R GpsL1CA # 7 221473809 59 125 101 1000101110101010101010001010001111110101001000101000110001010
1 0 016 0R GpsL1CA # 16 221473809 59 125 101 1000101110101010101010001010001111110101001000101000110001010
<----->
```

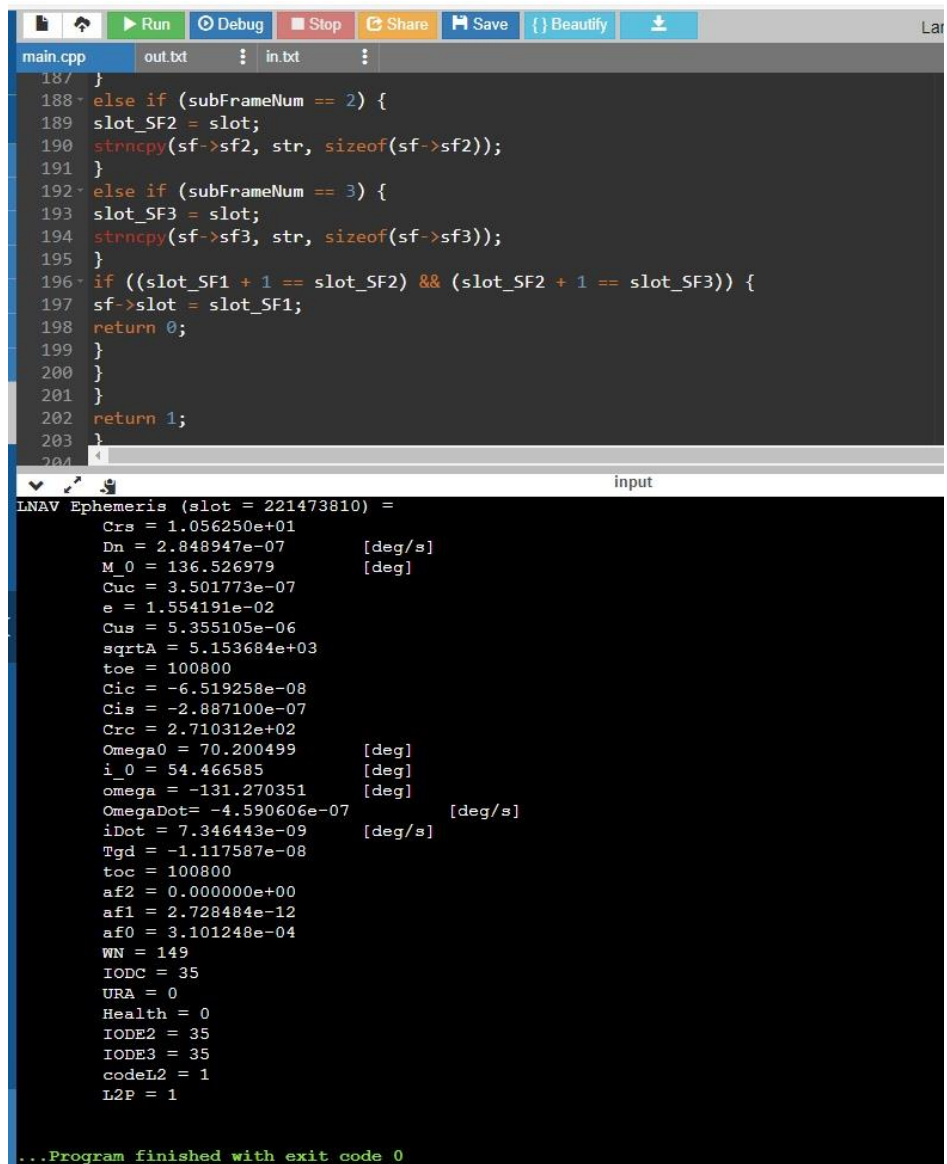
Рисунок 2 – Исходные данные файла in.txt

Согласно варианту задания необходимо обработать данные для 7 спутника, а именно: эфемеридная информация, содержащаяся в первых трёх сабфреймах структуры навигационного сообщения. В приложении А представлен листинг кода программы, реализующий обработку файла in.txt на языке C.

Функция в коде file2subFrames выделяет данные из первых трёх сабфреймов навигационного сообщения. Затем через функцию subFrames2Eph выделяет из сабфреймов необходимую эфемеридную информацию о Кеплеровских элементах орбиты, шкале времени часов спутника. Метод twoCompl2int вводит дополнение до двух, а именно позволяет получать данные из структуры сабфрейма с учётом знака. Функция printEmp выводит на экран таблицу эфемерид для заданного КА.

Сохранение таблицы эфемерид производится с помощью функции save в файл out.txt.

Скомпилируем программу с помощью онлайн-сервиса [Online C++ Compiler - online editor \(onlinegdb.com\)](https://onlinegdb.com) и выведем на экран полученную таблицу эфемерид:



The image shows a code editor window with a toolbar at the top containing icons for Run, Debug, Stop, Share, Save, Beautify, and a download icon. The editor has three tabs: 'main.cpp', 'out.txt', and 'in.txt'. The 'main.cpp' tab is active, displaying C++ code from line 187 to 204. The code includes conditional logic for 'subFrameNum' and a return statement. Below the code editor, there is an 'input' section with a dark background. It displays the output of the program, which is a table of LNAV Ephemeris data for a specific slot. The output includes various orbital parameters and their units, such as [deg/s] and [deg]. At the bottom of the output section, it states '...Program finished with exit code 0'.

```
187 }
188 else if (subFrameNum == 2) {
189     slot_SF2 = slot;
190     strncpy(sf->sf2, str, sizeof(sf->sf2));
191 }
192 else if (subFrameNum == 3) {
193     slot_SF3 = slot;
194     strncpy(sf->sf3, str, sizeof(sf->sf3));
195 }
196 if ((slot_SF1 + 1 == slot_SF2) && (slot_SF2 + 1 == slot_SF3)) {
197     sf->slot = slot_SF1;
198     return 0;
199 }
200 }
201 }
202 return 1;
203 }
204 }
```

input

```
LNAV Ephemeris (slot = 221473810) =
  Crs = 1.056250e+01
  Dn = 2.848947e-07 [deg/s]
  M_0 = 136.526979 [deg]
  Cuc = 3.501773e-07
  e = 1.554191e-02
  Cus = 5.355105e-06
  sqrtA = 5.153684e+03
  toe = 100800
  Cic = -6.519258e-08
  Cis = -2.887100e-07
  Crc = 2.710312e+02
  Omega0 = 70.200499 [deg]
  i_0 = 54.466585 [deg]
  omega = -131.270351 [deg]
  OmegaDot = -4.590606e-07 [deg/s]
  iDot = 7.346443e-09 [deg/s]
  Tgd = -1.117587e-08
  toc = 100800
  af2 = 0.000000e+00
  af1 = 2.728484e-12
  af0 = 3.101248e-04
  WN = 149
  IODC = 35
  URA = 0
  Health = 0
  IODE2 = 35
  IODE3 = 35
  codeL2 = 1
  L2P = 1
...Program finished with exit code 0
```

Рисунок 3 – Результат компиляции программы

Запишем в файл out.txt таблицу эфемерид:



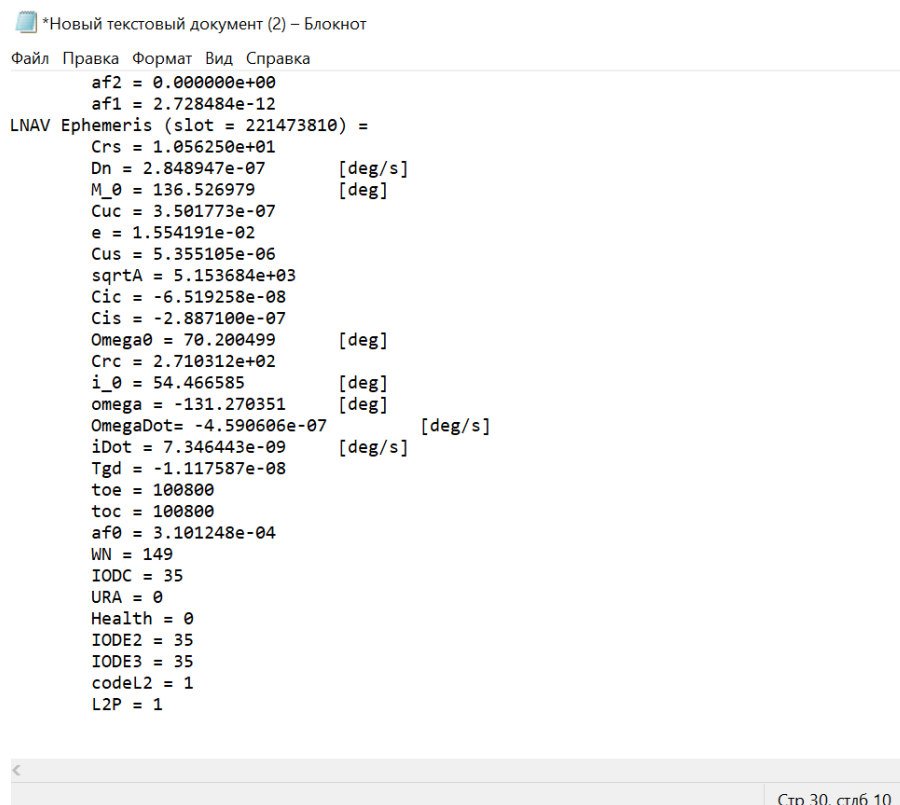


Рисунок 4 – Запись таблицы эфемерид в файл out.txt

### 1.3 Сравнение результатов разработанной программы и программы RTKNAVI

Эфемеридная информация также была представлена в файле BINR.bin. После обработки файла BINR.bin с помощью программы RTKNAVI из состава RTKLIV получим таблицу эфемерид и сравним с рисунком 4 п.1.2:

RTKNAVI ver.2.4.3 b34 (3): RTK Monitor																
Nav Data	GPS	ALL	Current 1													
SAT	PRN	Statu	IODE	IODC	URA	SVH	Toe	Toc	Ttrans	A (m)	e	i0 (°)	Ω0 (°)	ω (°)	M0 (°)	Δn (°/s)
G01	1	-	5911	23	0	000	2022/02/14 00:00:00	2022/02/14 00:00:00	-	26560304.325	0.01137974	56.53057	-109.43551	50.51432	54.55674	2.4630E-07
G02	2	-	-	0	000	-	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G03	3	OK	2827	11	0	000	2022/02/14 01:59:44	2022/02/14 01:59:44	2022/06/04 21:08:07	26560504.265	0.00389213	55.72485	-50.10616	54.89203	48.47481	2.3572E-07
G04	4	OK	1953	588	0	000	2022/02/14 02:00:00	2022/02/14 02:00:00	2022/06/04 21:08:07	26560157.291	0.00169565	55.09149	11.82685	-175.59716	-142.99015	2.4548E-07
G05	5	-	-	0	000	-	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G06	6	OK	2210	86	0	000	2022/02/14 02:00:00	2022/02/14 02:00:00	2022/06/04 21:08:07	26560290.249	0.00269551	56.49500	-109.91291	-55.01959	133.64356	2.4669E-07
G07	7	OK	8995	35	0	000	2022/02/14 04:00:00	2022/02/14 04:00:00	2022/06/04 21:08:07	26560462.036	0.01554191	54.46659	70.20050	-131.27035	136.52698	2.8489E-07
G08	8	OK	1028	4	0	000	2022/02/14 01:59:44	2022/02/14 01:59:44	-	26560852.794	0.00728541	55.29557	-171.29829	5.90629	-115.58396	2.4679E-07
G09	9	OK	7710	30	0	000	2022/02/14 02:00:00	2022/02/14 02:00:00	2022/06/04 21:08:07	26559848.321	0.00225987	54.69112	8.79464	107.10939	-94.65112	2.5565E-07
G10	10	OK	1182	46	0	000	2022/02/14 02:00:00	2022/02/14 02:00:00	-	26560464.021	0.00750119	55.71486	-50.26591	-144.08942	-1.69731	2.3953E-07
G11	11	-	-	0	000	-	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00

Рисунок 5 – Эфемериды, полученные из программы RTKNAVI

Комментарий к рисункам 4 – 5:

По полученным таблицам можно отметить, что эфемеридная информация для заданного спутника была выделена корректно.

#### Вывод:

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.

## **2 Моделирование траектории движения**

### **2.1 Задание второе**

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.

Требуется:

1. Реализовать в Matlab или Python (описание модели и её листинг)
2. Записать таблицу использованных эфемерид
3. Построить трехмерные графики положений спутника в ECEF и ECI (не забыть подписать оси, изобразить соответствующую Земле сферу в начале СК)
4. Построить расчётный и полученный в GNSS Planning Online SkyView

### **2.2 Разработка программы расчета положения спутника**

Из [2] запишем таблицу значений эфемерид из п.1.2 в соответствии с моделью GPS и воспользуемся константами из ИКД GPS.

Таблица 1 – Значения эфемерид в соответствии с моделью GPS

Crs	1.056250e+01
$\Delta n$	2.848947e-07
$M_0$	136.526979
Cuc	3.501773e-07
e	1.554191e-02

Cus	5.355105e-06
sqrtA	5.153684e+03
toe	-6.519258e-08
Cic	-2.887100e-07
Omega0	70.200499
Cis	2.710312e+02
i0	54.466585
Crc	-131.270351
omega	-4.590606e-07
OmegaDot	7.346443e-09
iDot	-1.117587e-08
Tgd	100800
toc	100800
af2	3.101248e-04
af1	149
af0	35
WN	0

Далее значения из табл.1 используются для расчета положения спутника GPS на заданный момент по шкале времени UTC на суточном интервале в системе координат ECEF WGS 84 и соответствующей ей инерциальной СК. Алгоритм расчёта координат КА взят из [2].

Для перевода в инерциальную СК расчёт проводится по (1):

$$\begin{cases} x' = x \cos(\theta) - y \sin(\theta) \\ y' = x \sin(\theta) + y \cos(\theta) \\ z' = z \end{cases} \quad (1)$$

, где  $\theta = \Omega_c (t - t_0)$

Центр декартовой системы координат переносится в точку приёма (рисунок 6). Далее координаты КА относительно точки приёма пересчитываются в полярную систему координат по (2) из алгоритма ИКД.

$$\begin{cases} r = \sqrt{x^2 + y^2 + z^2} \\ \cos(\theta) = \frac{z}{\sqrt{x^2 + y^2 + z^2}} \\ \operatorname{tg}(\varphi) = \frac{y}{x} \end{cases} \quad (2)$$

Координаты приёмника в RTKNAVI:

Solution:	SINGLE <input type="checkbox"/>
N:	44° 09' 36.3261"
E:	39° 00' 13.0546"
He:	1.247 m
N: 5.081 E: 2.342 U: 4.863 m	
Age: 0.0 s Ratio: 0.0 #Sat: 6	

Рисунок 6 – Координаты приемника

Вывод значений координат производится в файл out.txt. В приложении Б представлен листинг кода, реализующий расчёт положения КА и вывод соответствующих графиков.

### 2.3 Результаты моделирования

На рисунке 7 представлена траектория движения КА №7 на интервале суток в СК ECEF WGS 84 с отмеченным местоположением приёмника. На рисунке 8 изображена траектория движения КА в СК, а также точка, в которой находится приёмник. На рисунке 8 изображён Sky Plot с учётом угла места в 10 град.

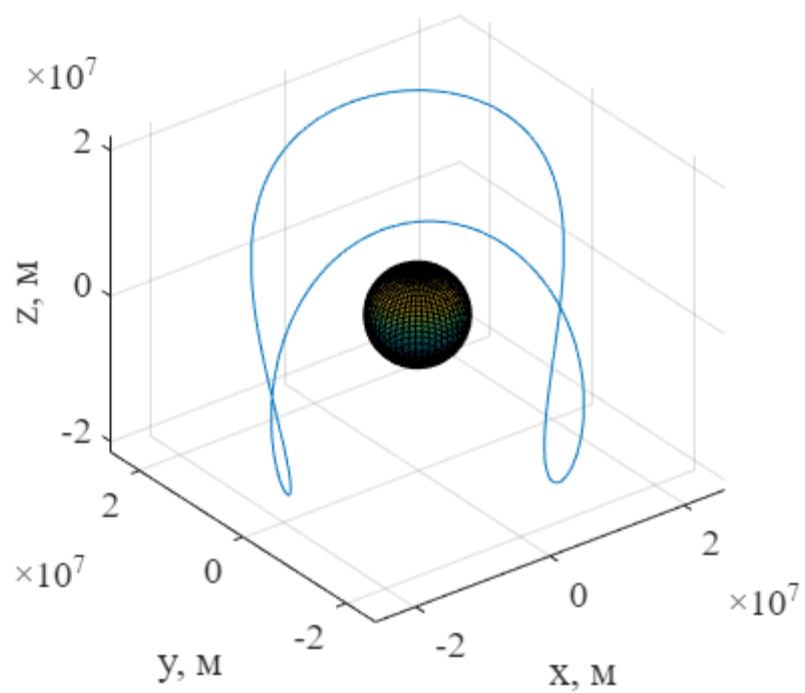


Рисунок 7 – Траектория движения КА в СК ECEF WGS84 и приёмник

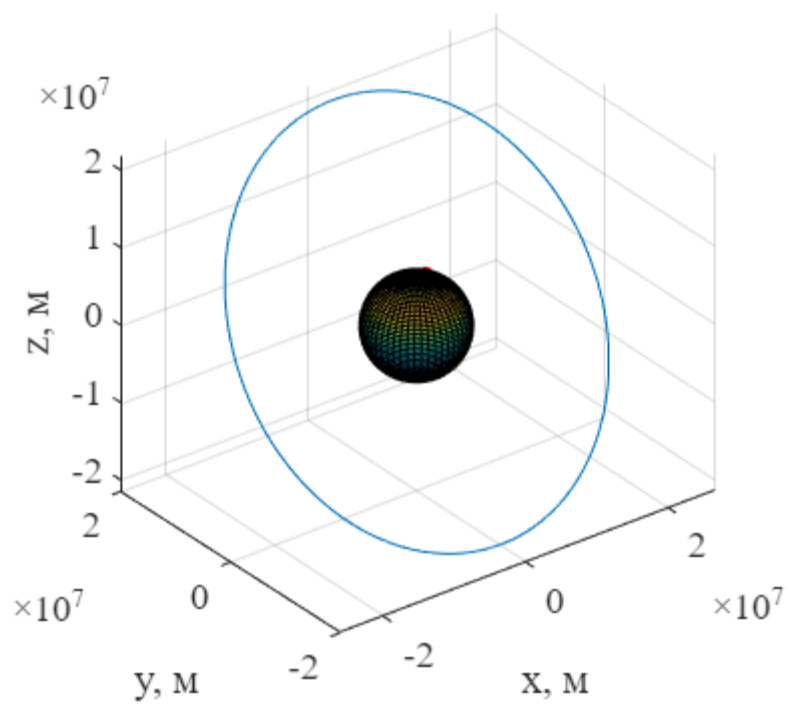


Рисунок 8 – Траектория движения КА в СК ECI и приёмник

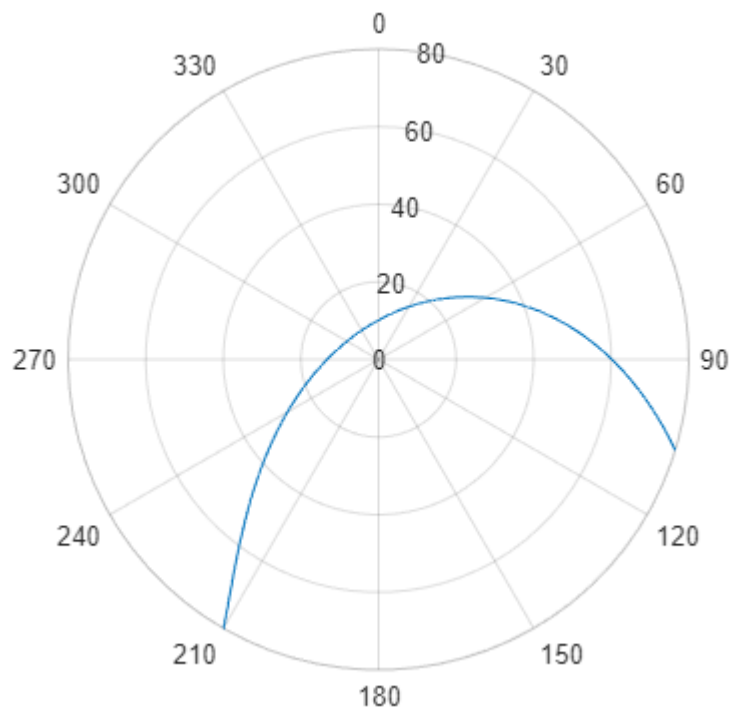


Рисунок 9 – Sky Plot

## 2.4 Сравнение результатов моделирования с Trimble GNSS Planning Online

Выставим необходимые параметры в Trimble GNSS Planning Online для построения SkyView на заданную дату и время:

Satellite Selection		
Change selection		
Satellites: 30/140		
System: active	Selected	Healthy
GPS <input checked="" type="checkbox"/>	30	30
GLONASS <input checked="" type="checkbox"/>	0	22
Galileo <input checked="" type="checkbox"/>	0	24
BeiDou <input checked="" type="checkbox"/>	0	48
QZSS <input checked="" type="checkbox"/>	0	5
IRNSS <input checked="" type="checkbox"/>	0	7

My Settings	
Time of almanac:	2022-02-14
Time zone:	UTC +00:00
Visible period:	2022-02-14 04:00 - 2022-02-15 04:00
Latitude:	N 44° 9' 36.3261"
Longitude:	E 39° 0' 13.0546"
Height:	1 m
Elevation cutoff:	10 °

Settings	
Latitude:	N 44° 9' 36.3261"
Longitude:	E 39° 0' 13.0546"
Height:	1,247 m
Elevation cutoff:	10 °
Day:	14.02.2022 Today
Start time:	04:00 UTC+00:00
Period [hours]:	24
Time zone:	(UTC) Coordinated Universal Time
<input checked="" type="button" value="Apply"/>	

Рисунок 10 — Данные для построения SkyView

На рисунках 11,12 построен SkyView КА №7 системы GPS.

На интервале времени 14.02.2022 04:00 - 15.02.2022 04:00 из точки приёма КА виден 2 раза.

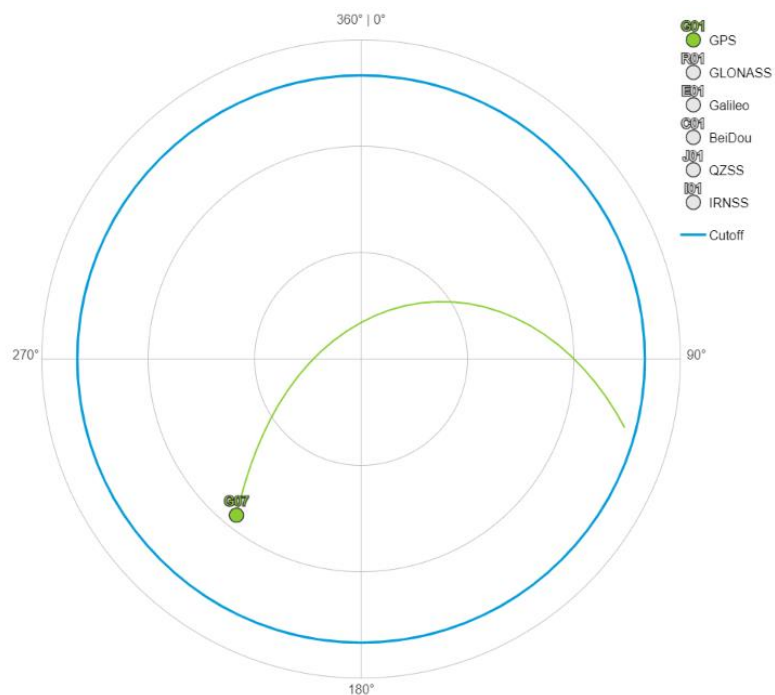


Рисунок 11 – Построение SkyView для первого пролёта КА

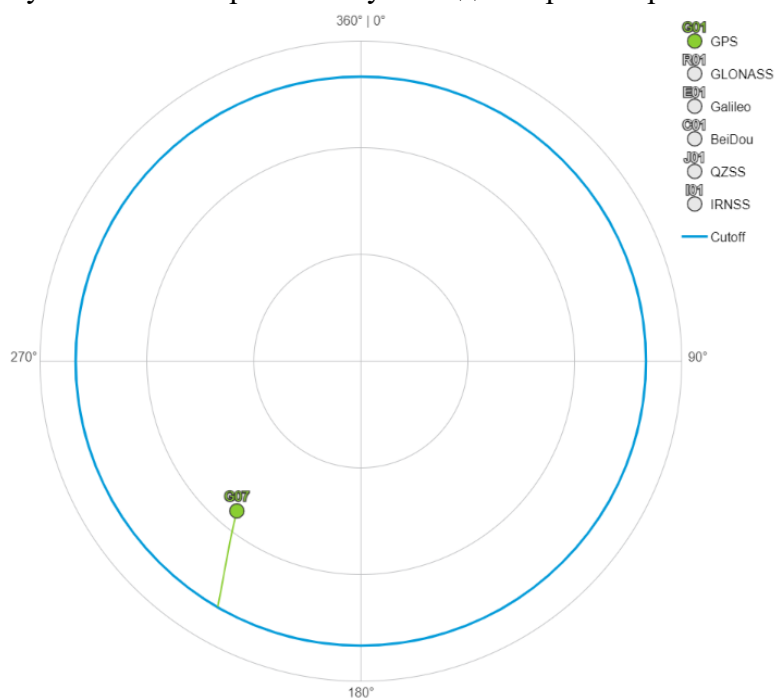


Рисунок 12 – Построение SkyView для второго пролёта КА

Комментарий к рис. 9,11,12:

Рассчитанные положения КА в результате моделирования совпадают с Trimble GNSS Planning Online

### ***Вывод:***

На втором этапе курсового проекта была разработана программа, выполняющая расчет положения спутника GPS, выводящая в файл out.txt значения координат спутника в заданном формате. При сравнении графиков на рисунках 8 и 10 замечается сходство рассчитанного SkyView и полученного с помощью Trimble GNSS Planning Online.

## ***3 Реализация модуля расчета координат***

### ***3.1 Задание третье***

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.

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.

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 программы расчета положения спутника***

Задача расчёта координат КА была решена в Matlab, код был перенесён на C++. В результате расчёта в Matlab и C++ возникла существенная погрешность расчёта координат, это связано с тем, что в ч.1. курсового проекта эфемеридные данные сохранялись с низкой точностью, что повлияло на расчёт координат в Matlab, тогда как в C++ при считывании данных из файла точность сохранялась. Длительность расчёта составила от 5 до 90 секунд из-за использования онлайн-сервиса. В связи с этим также наблюдали и большие затраты по памяти.

Расчёт местоположения КА осуществлялся при помощи уравнения Кеплера. Алгоритм расчета местоположения спутника включает в себя уравнение Кеплера. Уравнение Кеплера решается в виде трансцендентного уравнения. Алгоритм расчёта представлен на рисунке 13.

```
290 double E_0 = M_k;  
291 double E_k = 0;  
292 int k = 0;  
293 while (1)  
294 {  
295     E_k = E_0;  
296     E_0 = E_0 + (M_k - E_k + ep->e * sin(E_0)) / (1 - ep->e * cos(E_0));  
297     if (abs(E_0 - E_k) < 1e-8){  
298         break;  
299     }  
300     k = k + 1;
```

Рисунок 13 – Реализация уравнения Кеплера

На рисунке 14 представлена максимальная разница в координатах и время выполнения разработанной программы, что и требовалось по заданию.

```

151 else {
152     printf(" Subframes not found\n ");
153 }
154 }
155 else {
156     printf(" Не могу открыть in.txt ");
157 }
158 return 0;
159 }
160 uint32_t str2uint(char *sf, int32_t start, int32_t stop) {
161     uint32_t ans = 0;
162     for (int i = start; i < stop; i++) {
163         bool bit = (sf[i - 1] == '1');
164         ans = ans | (bit << (stop - i - 1));
165     }
166     return ans;
167 }
168 uint32_t str2uint1(char *sf, int32_t start, int32_t stop, int32_t start1, int32_t stop1)
169 f

```

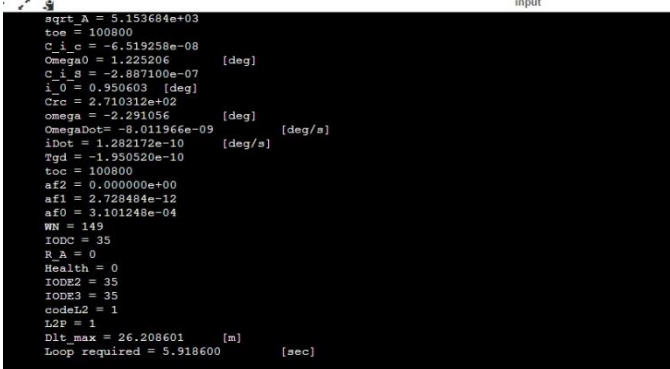


Рисунок 14 – Максимальная разница координат и время расчёта основной функции

### 3.3 Потребляемая память и утечки

В процессе отладки кода средний объем потребляемой памяти на расчёт программы составил 7,2 МБ. Величину получали путем сравнения значения используемой памяти до запуска кода и во время выполнения.

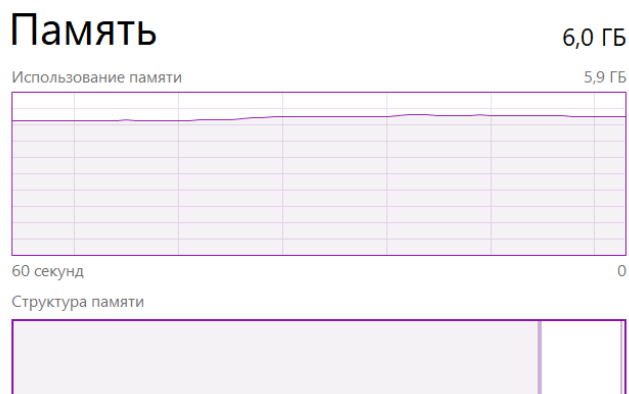


Рисунок 15 – Объём затрачиваемой памяти на алгоритм

#### Вывод:

На третьем этапе курсового проекта была написана программа на языке C++, что целесообразно при разработке встраиваемого ПО. Было обнаружено расхождение в результатах, полученных на 2 и 3 этапе курсового проекта, составляющее 26,2 м. Объем потребляемой памяти составил 7,2 МБ, утечка памяти не была обнаружена.

## *Заключение*

Разработана программа, обрабатывающая эфемеридные данные с КА системы GPS. Программа позволяет реализовать расчёт для любого КА, т.к. в эфемеридных данных заключена информация о всех КА в орбитальной группировке.

Разработан программный код в среде Matlab, который позволяет определять координаты КА в геоцентрической инерциальной системой координат, а также в системе координат, связанной с Землей. Перевод из одной системы координат в другую необходим для удобства, т.к., например для КА удобно использовать геоцентрическую инерциальную систему координат, а для приёмника легче рассчитывать координаты в системе координат, связанной с Землёй.


Построен Sky plot, для КА на интервале суток, отметим, что на интервале из точки приёма он был виден 2 раза, что совпало со SkyView.

Разработан программный модуль, обрабатывающий входной массив данных (эфемерид) и рассчитывающий координаты КА на интервале суток. Оценено время выполнения программы и оценена точность 2-х алгоритмов.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2017. Структура и правила оформления отчета о научноисследовательской работе.
2. [GPS Interface Specification IS-GPS-200, Revision M - May 2021](#) Interface Control Contractor: SAIC (GPS SEI) 200 M. Pacific Coast Highway, Suite 1800 El Segundo, CA 90245.
3. [Прикладной потребительский центр ГЛОНАСС \(glonass-iac.ru\)](#) Информационно-аналитический центр координатновременного и навигационного обеспечения.
4. <https://docs.microsoft.com/> Техническая документация Майкрософт.
5. Р. Лафоре Объектно-ориентированное программирование в C++. - 4-е изд. - Москва: Питер, 2004. - 923 с
6. [Планирование Trimble GNSS \(gnssplanningonline.com\)](#) (дата обращения: 04.06.2022)
7. [Online C++ Compiler - online editor \(onlinegdb.com\)](#) (дата обращения: 04.06.2022)

## ПРИЛОЖЕНИЕ А

 p11 – Блокнот

Файл Правка Формат Вид Справка

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <cmath>
#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)
#define sc 180
struct Ephemeris {
double Crs;
double Dn;
double M_0;
double Cuc;
double e;
double Cus;
double sqrtA;
uint32_t toe;
double Cic;
double Omega0;
double Cis;
double i_0;
double Crc;
double omega;
double OmegaDot;
double iDot;
double Tgd;
uint32_t toc;
double af2;
double af1;
double af0;
uint32_t WN;
uint16_t IODC;
uint8_t URA;
uint8_t Health;
```

```

uint16_t IODE2;
uint16_t IODE3;
bool codeL2;
bool L2P;
uint32_t slot;
};
const int32_t subFrameLength = 300;
struct SF1_3
{
uint32_t slot;
char sf1[subFrameLength + 1];
char sf2[subFrameLength + 1];
char sf3[subFrameLength + 1];
};
void printEmp(Ephemeris* ep);
int32_t file2subFrames(SF1_3* sf, FILE* fid, uint8_t svNum);
int32_t subFrames2Eph(Ephemeris* ep, SF1_3* subframes);
void save(Ephemeris* ep, FILE* fod);
int main(void
)

{
uint8_t svNum = 7;
FILE* fid = fopen("in.txt", "r");
FILE* fod = fopen("out.txt", "w");
if (fid != nullptr) {
SF1_3 subframes;
if (!file2subFrames(&subframes, fid, svNum)) {
Ephemeris *ep = (Ephemeris*)calloc(1, sizeof(Ephemeris));
if (!subFrames2Eph(ep, &subframes)) {
printEmp(ep);
}
}
else {
printf(" Cannot decode subframes\n ");
}
fclose(fid);
if (fod) {
save(ep, fod);
}
else

```

```

{
printf(" Cannot open out.txt ");
}
fclose(fod);
free(ep);
}
else {
printf(" Subframes not found\n ");
}
}
else {
printf(" Cannot open in.txt ");
}
return 0;
}
uint32_t str2uint(char *sf, int32_t start, int32_t stop) {
uint32_t ans = 0;
for (int i = start; i < stop; i++) {
bool bit = (sf[i - 1] == '1');
ans = ans | (bit << (stop - i - 1));
}
return ans;
}
uint32_t str2uint1(char *sf, int32_t start, int32_t stop, int32_t start1, int32_t stop1)
{
uint32_t ans = 0;
for (int i = start; i < stop; i++) {
bool bit = (sf[i - 1] == '1');
ans = (ans | bit) << 1;
}
for (int i = start1; i < stop1; i++) {
bool bit = (sf[i - 1] == '1');
ans = ans | bit;
if (i < stop1 - 1) {
ans = ans << 1;
}
}
return ans;
}

```

```

int32_t m = 0xFFFFFFFF;
if ((numBit < 32) && bool((1 << numBit - 1) & ans))
{
    ans |= m << numBit;
    return ~(~(ans - 1));
}
if (numBit == 32 && bool((1 << 31) & ans)) {
    return ~(~(ans - 1));
}
return ans;
}
int32_t subFrames2Eph(Ephemeris* ep, SF1_3* subframes) {
    ep->slot = subframes->slot;
    ep->WN = str2uint(subframes->sf1, 61, 71);
    ep->CrS = twoCompl2int(str2uint(subframes->sf2, 69, 85), 16) * sF1;
    ep->Cuc = twoCompl2int(str2uint(subframes->sf2, 151, 167), 16) * sF4;
    ep->toe = str2uint(subframes->sf2, 271, 287) * sF7;
    ep->toC = twoCompl2int(str2uint(subframes->sf1, 219, 235), 16) * sF7;
    ep->IODC = str2uint1(subframes->sf1, 83, 85, 211, 219);
    ep->URA = str2uint(subframes->sf1, 73, 75);
    ep->Health = str2uint(subframes->sf1, 73, 79);
    ep->IODE2 = str2uint(subframes->sf2, 61, 69);
    ep->IODE3 = str2uint(subframes->sf3, 271, 279);
    ep->codeL2 = bool(subframes->sf1[91]);
    ep->L2P = bool(subframes->sf1[91]);
    ep->e = str2uint1(subframes->sf2, 167, 175, 181, 205) * sF5;
    ep->af1 = twoCompl2int(str2uint(subframes->sf1, 249, 265), 16) * sF2;
    ep->af2 = twoCompl2int(str2uint(subframes->sf1, 241, 249), 8) * sF8;
    ep->af0 = twoCompl2int(str2uint(subframes->sf1, 271, 293), 22) * sF3;
    ep->Dn = twoCompl2int(str2uint(subframes->sf2, 91, 107), 16) * sF2 * sC;
    ep->M_0 = twoCompl2int(str2uint1(subframes->sf2, 107, 115, 121, 145), 32) * sF3 * sC;
    ep->Cus = twoCompl2int(str2uint(subframes->sf2, 211, 227), 16) * sF4;
    ep->sqrta = str2uint1(subframes->sf2, 227, 235, 241, 265) * sF6;
    ep->Cic = twoCompl2int(str2uint(subframes->sf3, 61, 77), 16) * sF4;
    ep->Omega0 = twoCompl2int(str2uint1(subframes->sf3, 77, 85, 91, 115), 32) * sF3 * sC;
    ep->Cis = twoCompl2int(str2uint(subframes->sf3, 121, 137), 16) * sF4;
    ep->i_0 = twoCompl2int(str2uint1(subframes->sf3, 137, 145, 151, 175), 32) * sF3 * sC;
    ep->Crc = twoCompl2int(str2uint(subframes->sf3, 181, 197), 16) * sF1;
    ep->omega = twoCompl2int(str2uint1(subframes->sf3, 197, 205, 211, 235), 32) * sF3 * sC;
    ep->OmegaDot = twoCompl2int(str2uint(subframes->sf3, 241, 265), 24) * sF2 * sC;
    ep->iDot = twoCompl2int(str2uint(subframes->sf3, 279, 293), 14) * sF2 * sC;
    ep->Tgd = twoCompl2int(str2uint(subframes->sf1, 197, 205, 8) * sF3;
    return 0;
}
int32_t file2subFrames(SF1_3* sf, FILE* fid, uint8_t svNum) {
    int32_t sth1, sth2, sth3, sth4, sth5;
    char str_OR[8];
    char str_GPSL1CA[12];
    char str_reh[8];
    char str[1000];
    uint32_t svStr;
    uint32_t slot;
    int32_t subFrameNum;
    uint32_t slot_SF1 = 0;
    uint32_t slot_SF2 = 0;
    uint32_t slot_SF3 = 0;
    int32_t readres = 0;
    while (readres != EOF)
    {
        svStr = 0;
        readres = fscanf(fid, "%d %d %d %s %s %s %u\t %u %d %d %d %s", &sth1, &sth2, &sth3, str_OR, str_GPSL1CA, str_reh, &svStr, &slot, &sth4, &sth5, &subFrameNum, str);
        if ((svStr == svNum) && (slot >= (604800 / 6))) {
            if (subFrameNum == 1) {
                slot_SF1 = slot;
                strncpy(sf->sf1, str, sizeof(sf->sf1));
            }
            else if (subFrameNum == 2) {
                slot_SF2 = slot;
                strncpy(sf->sf2, str, sizeof(sf->sf2));
            }
            else if (subFrameNum == 3) {
                slot_SF3 = slot;
                strncpy(sf->sf3, str, sizeof(sf->sf3));
            }
            if ((slot_SF1 + 1 == slot_SF2) && (slot_SF2 + 1 == slot_SF3)) {
                sf->slot = slot_SF1;
                return 0;
            }
        }
    }
}

```



```

}
}
}
return 1;
}
void printEmp(Ephemeris* ep)
{
printf("LNAV Ephemeris (slot = %u) = \n", ep->slot);
printf("\tCrs = %e \n", ep->Crs);
printf("\tDn = %e \t[deg/s] \n", ep->Dn);
printf("\tM_0 = %f \t[deg] \n", ep->M_0);
printf("\tCuc = %e \n", ep->Cuc);
printf("\te = %e \n", ep->e);
printf("\tCus = %e \n", ep->Cus);
printf("\tsqrtA = %e \n", ep->sqrtA);
printf("\ttoe = %u \n", ep->toe);
printf("\tCic = %e \n", ep->Cic);
printf("\tCis = %e \n", ep->Cis);
printf("\tCrc = %e \n", ep->Crc);
printf("\tOmega0 = %f \t[deg] \n", ep->Omega0);
printf("\ti_0 = %f \t[deg] \n", ep->i_0);
printf("\tomega = %f \t[deg] \n", ep->omega);
printf("\tOmegaDot = %e \t[deg/s] \n", ep->OmegaDot);
printf("\tiDot = %e \t[deg/s] \n", ep->iDot);
printf("\tTgd = %e \n", ep->Tgd);
printf("\ttoc = %u \n", ep->toc);
printf("\taf2 = %e \n", ep->af2);
printf("\taf1 = %e \n", ep->af1);
printf("\taf0 = %e \n", ep->af0);
printf("\tWN = %u \n", ep->WN);
printf("\tIODC = %u \n", ep->IODC);
printf("\tURA = %u \n", ep->URA);
printf("\tHealth = %u \n", ep->Health);
printf("\tIODE2 = %u \n", ep->IODE2);
printf("\tIODE3 = %u \n", ep->IODE3);
printf("\tcodeL2 = %u \n", ep->codeL2);
printf("\tL2P = %u \n", ep->L2P);
}

```

```

void save(Ephemeris* ep, FILE* fod)
{
    fprintf(fod, "\taf2 = %e \n", ep->af2);
    fprintf(fod, "\taf1 = %e \n", ep->af1);
    fprintf(fod, "LNAV Ephemeris (slot = %u) = \n", ep->slot);
    fprintf(fod, "\tCrs = %e \n", ep->Crs);
    fprintf(fod, "\tDn = %e \t[deg/s] \n", ep->Dn);
    fprintf(fod, "\tM_0 = %f \t[deg] \n", ep->M_0);
    fprintf(fod, "\tCuc = %e \n", ep->Cuc);
    fprintf(fod, "\te = %e \n", ep->e);
    fprintf(fod, "\tCus = %e \n", ep->Cus);
    fprintf(fod, "\tsqrtA = %e \n", ep->sqrtA);
    fprintf(fod, "\tCic = %e \n", ep->Cic);
    fprintf(fod, "\tCis = %e \n", ep->Cis);
    fprintf(fod, "\tOmega0 = %f \t[deg] \n", ep->Omega0);
    fprintf(fod, "\tCrc = %e \n", ep->Crc);
    fprintf(fod, "\ti_0 = %f \t[deg] \n", ep->i_0);
    fprintf(fod, "\tomega = %f \t[deg] \n", ep->omega);
    fprintf(fod, "\tOmegaDot = %e \t[deg/s] \n", ep->OmegaDot);
    fprintf(fod, "\tiDot = %e \t[deg/s] \n", ep->iDot);
    fprintf(fod, "\tTgd = %e \n", ep->Tgd);
    fprintf(fod, "\ttoe = %u \n", ep->toe);
    fprintf(fod, "\ttoc = %u \n", ep->toc);
    fprintf(fod, "\taf0 = %e \n", ep->af0);
    fprintf(fod, "\tWN = %u \n", ep->WN);
    fprintf(fod, "\tIODC = %u \n", ep->IODC);
    fprintf(fod, "\tURA = %u \n", ep->URA);
    fprintf(fod, "\tHealth = %u \n", ep->Health);
    fprintf(fod, "\tIODE2 = %u \n", ep->IODE2);
    fprintf(fod, "\tIODE3 = %u \n", ep->IODE3);
    fprintf(fod, "\tcodeL2 = %u \n", ep->codeL2);
    fprintf(fod, "\tL2P = %u \n", ep->L2P);
}

```

## ПРИЛОЖЕНИЕ Б

```
clc
clear all
close all hidden
close all force
set(0,'defaultFontSize', 14)
set(0,'defaultAxesFontSize', 14)
set(0,'defaultAxesFontName','Times')
set(0,'defaultTextFontName','Times')
set(0,'defaultFigurePaperPositionMode','auto')
set(0,'defaultFigurePaperType','<custom>')
set(0,'defaultFigurePaperSize',[12, 9]')
format longE
pi= 3.1415926535898; % Отношение длины окружности к ее диаметру
mu = 3.986004418*1e14; % Геоцентрическая гравитационная постоянная
omega_E = 7.2921151467*1e-5; % Средняя угловая скорость Земли
c = physconst('LightSpeed'); % Скорость света
Earth_radius = physconst('EarthRadius'); % Радиус Земли
%% Эфемериды в соответствии с моделью GPS
C_rs = 1.056250e+01; % Амплитуда поправочного члена синусоидальной гармоник
delta_n = deg2rad(2.848947e-07); % Среднее отклонение движения от вычисленного
M_0 = deg2rad(136.526979); % Средняя аномалия в контрольный момент времени
C_u_c = 3.501773e-07; % Амплитуда поправочного члена косинусной гармоник к
e_c_c = 1.554191e-02; % Эксцентриситет
C_u_s = 5.355105e-06; % Амплитуда поправочного члена синусоидальной гармоник
aSqRoot = 5.153684e+03; % Поправка к большой полуоси
t_oe = 100800; % Опорное время эфемерид
C_ic = -6.519258e-08 ; % Амплитуда поправочного члена косинусной гармоник к
Omega0 = deg2rad(70.200499); % Долгота восходящего узла
C_is = -2.887100e-07; % Амплитуда поправочного члена синусоидальной гармоник
i0 = deg2rad(54.466585);
C_rc = 2.056875e+002;
omega = deg2rad(-131.270351); % Аргумент перигея
OmegaDot = deg2rad(-4.590606e-07); % Скорость прямого восхождения
iDot = deg2rad(7.346443e-09); % Скорость угла наклона
%% Расчёт Кеплеровских элементов орбиты
A = aSqRoot^2; % Большая полуось
n_0 = sqrt(mu/A^3); % Расчетное среднее движение
%% Алгоритм расчёта координат
leapSeconds = 18; % Секунды координации
daySecondsCount = 86400;
for ind = 1:daySecondsCount
    t = daySecondsCount + leapSeconds + ind; % Количество секунд от начала
    %текущей недели
    t_k = t - t_oe; % Время от опорной эпохи эфемерид
    if t_k > 302400
        t_k = t_k - 604800;
    elseif t_k < -302400
        t_k = t_k + 604800;
    end
    n = n_0 + delta_n; % Скорректированное среднее движение
    M_k = M_0 + n * t_k; % Средняя аномалия
    E_0 = M_k;
    E_k = 0;
```

```

count = 0;
while true
    E_k = E_0;
    E_0 = E_0 + (M_k - E_0 + e_c_c * sin(E_0))/(1 - e_c_c * cos(E_0));
    if abs(E_0-E_k) < 1e-8
        break
    end
    count = count + 1;
end
nu=atan2((sqrt(1-e_c_c^2)* sin(E_k)/(1 - e_c_c*cos(E_k))), ((cos(E_k)-e_c_c)/(1 -
e_c_c * cos(E_k)))); % Истинная аномалия
Phi = nu + omega; % Аргумент широты
delta_u = C_u_s*sin(2*Phi) + C_u_c*cos(2*Phi); % Аргумент поправки на широту
delta_r = C_rs*sin(2*Phi) + C_rc*cos(2*Phi); % Коррекция радиуса
delta_i = C_is*sin(2*Phi) + C_ic*cos(2*Phi); % Коррекция наклона
u_k = Phi + delta_u; % Скорректированный аргумент широты
r = A * (1 - e_c_c * cos(E_k)) + delta_r; % Скорректированный радиус
i_corr = i0 + delta_i + iDot * t_k; % Скорректированное наклонение
x_shtr = r * cos(u_k);
y_shtr = r * sin(u_k);
Omega_corr = Omega0 + (OmegaDot - omega_E) * t_k - omega_E * t_oe; %
%Скорректированная широта восходящего узла
x(ind) = x_shtr * cos(Omega_corr) - y_shtr * cos(i_corr) * sin(Omega_corr); %
%Координаты КА
y(ind) = x_shtr * sin(Omega_corr) + y_shtr * cos(i_corr) * cos(Omega_corr);
z(ind) = y_shtr * sin(i_corr);
% Инерциальная СК
Theta = omega_E * t_k;
x_ics(ind) = x(ind) * cos(Theta) - y(ind) * sin(Theta);
y_ics(ind) = x(ind) * sin(Theta) + y(ind) * cos(Theta);
z_ics(ind) = z(ind);
% Вычисление координат приёмника
lat = deg2rad(44.09363261); % Широта
lon = deg2rad(39.00130546); % Долгота
H = 1.247; % Высота
x_point = (Earth_radius + H)*cos(lat)*cos(lon); % Расчёт координат приёмника
y_point = (Earth_radius + H)*cos(lat)*sin(lon);
z_point = (Earth_radius + H)*sin(lon);
[xRezult(ind), yRezult(ind), zRezult(ind)] =
ecef2enu(x(ind),y(ind),z(ind),lat,lon,H,wgs84Ellipsoid,'radians'); %
%Отображение координат КА относительно передатчика
% Полярная СК
if zRezult(ind) > 0
    rho(ind) = norm([xRezult(ind),yRezult(ind),zRezult(ind)]);
    theta(ind) = acos(zRezult(1,ind)/rho(ind));
    if xRezult(ind) > 0
        phi(ind) = -atan(yRezult(ind)/xRezult(ind))+pi/2;
    elseif (xRezult(ind)<0)&&(yRezult(ind)>0)
        phi(ind) = -atan(yRezult(ind)/xRezult(ind))+3*pi/2;
    elseif (xRezult(ind)<0)&&(yRezult(ind)<0)
        phi(ind) = -atan(yRezult(ind)/xRezult(ind))-pi/2;
    end
else
    rho(ind) = nan;
    theta(ind) = nan;

```

```

    phi(ind) = nan;
    end
end
%% Вывод значений координат в файл out.txt
outTxt = fopen('out.txt', 'w');
formatSpec = '%1.0f %10.8f %10.8f %10.8f\n';
for indCount = 1:length(x)
    fprintf(outTxt, formatSpec, indCount, x(indCount), y(indCount), z(indCount));
end
fclose(outTxt);
type out.txt

%% Расчёт для построения Земли
[x_sphere, y_sphere, z_sphere] = sphere(50);
x_Earth = Earth_radius * x_sphere;
y_Earth = Earth_radius * y_sphere;
z_Earth = Earth_radius * z_sphere;
%% Построение графиков
figure(1)
subplot(1,1,1)
surf(x_Earth, y_Earth, z_Earth)
hold on
plot3(x, y, z)
plot3(x_point, y_point, z_point, 'o', 'Color', 'r', 'MarkerSize', 5, 'MarkerFaceColor', 'r') % Построение точки
xlabel('x, м')
ylabel('y, м')
zlabel('z, м')
daspect([1,1,1])

figure (3)
axes = polaraxes;
polarplot(axes, phi, rad2deg(theta))
axes.ThetaDir = 'clockwise';
axes.ThetaZeroLocation = 'top';
rlim([0 80]) % Учёт угла места

```

## ПРИЛОЖЕНИЕ В

```

#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <cmath>
#include <time.h>
#include <ctime>
#include <iostream>
#include <fstream>
#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)
constexpr auto sc = 180;
constexpr auto pi = 3.1415326535898;
struct Ephemeris {
double Crs;
double Dn;
double M_0;
double C_u_c;
double e;
double C_u_s;
double sqrt_A;
double af2;
double af1;
double af0;
double C_i_c;
double Omega0;
double C_i_S;
double i_0;
double Crc;
double omega;
double OmegaDot;
double iDot;
double Tgd;
uint32_t toc;
uint32_t toe;
uint32_t WN;
uint16_t IODC;
uint8_t R_A;
uint8_t Health;
uint16_t IODE2;
uint16_t IODE3;
bool codeL2;
bool L2P;
uint32_t slot;
};
const int32_t subFrameLength = 300;
struct SF1_3 {
uint32_t slot;
char sf1[subFrameLength + 1];

```

```

char sf2[subFrameLength + 1];
char sf3[subFrameLength + 1];
};
struct Coord
{
double x;
double y;
double z;
};
void printEmp(Ephemeris* ep);
int32_t filesubtFr(SF1_3* sf, FILE* fid, uint8_t svNum);
int32_t subFramesTEph(Ephemeris* ep, SF1_3* subframes);
void save(Ephemeris* ep, FILE* fod);
void Coordinate_calculation(Ephemeris* ep, uint32_t t, Coord *Position);
int main(void)
{
uint32_t begin_time = clock();
uint8_t svNum = 7;
FILE* fid = fopen("in.txt", "r");
FILE* fod = fopen("out.txt", "w");
FILE* fopenfile = fopen("coords.txt", "r");
if (fid != nullptr) {
SF1_3 subframes;
if (!filesubtFr(&subframes, fid, svNum)) {
Ephemeris *ep = (Ephemeris*)calloc(1, sizeof(Ephemeris));
if (!subFramesTEph(ep, &subframes)) {
printEmp(ep);
}
else {
printf(" Cannot decode subframes\n ");
}
fclose(fid);
if (fod) {
save(ep, fod);
}
else
{
printf(" не могу открыть out.txt ");
}
fclose(fod);
uint32_t start = 86400 + 18 + 1; //

```

```

uint32_t stop = 86399 + start;
Coord Position;
double** Array_coord = new double*[3];
for (int i = 0; i < 3; i++)
{
    Array_coord[i] = new double[stop - start];
}
double** Array_coord_matlab = new double*[3];
for (int i = 0; i < 3; i++)
{
    Array_coord_matlab[i] = new double[stop -
start];
}
for (int t = start; t < stop; t++)
{
    Coordinate_calculation(ep, t, &Position);
    Array_coord[0][t - start] = Position.x;
    Array_coord[1][t - start] = Position.y;
    Array_coord[2][t - start] = Position.z;
}
std::ifstream file("coords.txt");
double sec;
if (!file.is_open())
    std::cout << "Не могу открыть" << std::endl;
else {
    for (int t = 0; t < stop - start; t++)
    {
        file >> sec >> Array_coord_matlab[0][t]
>> Array_coord_matlab[1][t] >> Array_coord_matlab[2][t];
    }
    file.close();
}
double Dlt_max = 0;
for (int i = 0; i < 3; i++)
{
    for (int k = 0; k < stop - start; k++)
    {
        if (abs(Array_coord[i][k] - Array_coord_matlab[i][k]) > Dlt_max)
        {
            Dlt_max = abs(Array_coord[i][k] - Array_coord_matlab[i][k]);
        }
    }
}

```



```

}
}
delete[] * Array_coord;
delete[] Array_coord;
delete[] * Array_coord_matlab;
delete[] Array_coord_matlab;
uint32_t end_time = clock();
uint32_t calculation_time = end_time - begin_time;
printf("\tDlt_max = %f \t[m] \n", Dlt_max);
printf("\tLoop required = %f \t[sec] \n",
double(calculation_time)/1e3);
free(ep);
}
else {
printf(" Subframes not found\n ");
}
}
else {
printf(" Не могу открыть in.txt ");
}
return 0;
}
uint32_t str2uint(char *sf, int32_t start, int32_t stop) {
uint32_t ans = 0;
for (int i = start; i < stop; i++) {
bool bit = (sf[i - 1] == '1');
ans = ans | (bit << (stop - i - 1));
}
return ans;
}
uint32_t str2uint1(char *sf, int32_t start, int32_t stop, int32_t start1, int32_t
stop1)
{
uint32_t ans = 0;
for (int i = start; i < stop; i++) {
bool bit = (sf[i - 1] == '1');
ans = (ans | bit) << 1;
}
for (int i = start1; i < stop1; i++) {
bool bit = (sf[i - 1] == '1');
ans = ans | bit;
}
}

```

```

if (i < stop1 - 1) {
ans = ans << 1;
}
}
return ans;
}

```

```

int32_t twoCompl2int(uint32_t ans, int numBit) { // twos-complement func
int32_t m = 0xFFFFFFFF;
if ((numBit < 32) && bool((1 << numBit - 1) & ans))
{
ans |= m << numBit;
return -(~(ans - 1));
}
if (numBit == 32 && bool((1 << 31) & ans)) {
return -(~(ans - 1));
}
return ans;
}

int32_t subFramesTEph(Ephemeris* ep, SF1_3* subframes) {
double dg_rd = pi / sc;
ep->slot = subframes->slot;
ep->WN = str2uint(subframes->sf1, 61, 71);
ep->Crs = twoCompl2int(str2uint(subframes->sf2, 69, 85), 16) * sF1;
ep->C_u_c = twoCompl2int(str2uint(subframes->sf2, 151, 167), 16)*sF4;
ep->toe = str2uint(subframes->sf2, 271, 287)*sF7;
ep->toc = twoCompl2int(str2uint(subframes->sf1, 219, 235), 16) * sF7;
ep->IODC = str2uint1(subframes->sf1, 83, 85, 211, 219);
ep->R_A = str2uint(subframes->sf1, 73, 75);
ep->Health = str2uint(subframes->sf1, 73, 79);
ep->IODE2 = str2uint(subframes->sf2, 61, 69);
ep->IODE3 = str2uint(subframes->sf3, 271, 279);
ep->codeL2 = bool(subframes->sf1[91]);
ep->L2P = bool(subframes->sf1[91]);
ep->e = str2uint1(subframes->sf2, 167, 175, 181, 205) * sF5;
ep->af1 = twoCompl2int(str2uint(subframes->sf1, 249, 265), 16) * sF2;
ep->af2 = twoCompl2int(str2uint(subframes->sf1, 241, 249), 8) * sF8;
ep->af0 = twoCompl2int(str2uint(subframes->sf1, 271, 293), 22) * sF3;
ep->Dn = twoCompl2int(str2uint(subframes->sf2, 91, 107), 16) * sF2 * sc
*dg_rd;

```

```

ep->M_0 = twoCompl2int(str2uint1(subframes->sf2, 107, 115, 121, 145),
32)
* sF3 * sc * dg_rd;
ep->C_u_s = twoCompl2int(str2uint(subframes->sf2, 211, 227), 16) * sF4;
ep->sqrt_A = str2uint1(subframes->sf2, 227, 235, 241, 265) * sF6;
ep->C_i_c = twoCompl2int(str2uint(subframes->sf3, 61, 77), 16) * sF4;
ep->Omega0 = twoCompl2int(str2uint1(subframes->sf3, 77, 85, 91, 115),
32) * sF3 * sc * dg_rd;
ep->C_i_S = twoCompl2int(str2uint(subframes->sf3, 121, 137), 16) * sF4;
ep->i_0 = twoCompl2int(str2uint1(subframes->sf3, 137, 145, 151, 175), 32)
*sF3 * sc * dg_rd;
ep->Crc = twoCompl2int(str2uint(subframes->sf3, 181, 197), 16) * sF1;
ep->omega = twoCompl2int(str2uint1(subframes->sf3, 197, 205, 211,
235),32) * sF3 * sc * dg_rd;
ep->OmegaDot = twoCompl2int(str2uint(subframes->sf3, 241, 265), 24)
*sF2 * sc * dg_rd;
ep->iDot = twoCompl2int(str2uint(subframes->sf3, 279, 293), 14) * sF2 *
sc* dg_rd;
ep->Tgd = twoCompl2int(str2uint(subframes->sf1, 197, 205), 8) * sF3
*dg_rd;
return 0;
}
int32_t filesubtFr(SF1_3* sf, FILE* fid, uint8_t svNum) {
int32_t sth1, sth2, sth3, sth4, sth5;
char str_0R[8];
char str_GPSL1CA[12];
char str_reh[8];
char str[1000];
uint32_t svStr;
uint32_t slot;
int32_t subFrameNum;
uint32_t slot_SF1 = 0;
uint32_t slot_SF2 = 0;
uint32_t slot_SF3 = 0;
int32_t readres = 0;
while (readres != EOF)
{
svStr = 0;
readres = fscanf(fid, "%d %d %d %s %s %s %u\t %u %d %d %d %s",
&sth1, &sth2, &sth3, str_0R, str_GPSL1CA, str_reh, &svStr, &slot, &sth4,
&sth5,

```

```

&subFrameNum, str);
if ((svStr == svNum) && (slot >= (604800 / 6))) {
if (subFrameNum == 1) {
slot_SF1 = slot;
strncpy(sf->sf1, str, sizeof(sf->sf1));
}
else if (subFrameNum == 2) {
slot_SF2 = slot;
strncpy(sf->sf2, str, sizeof(sf->sf2));
}
else if (subFrameNum == 3) {
slot_SF3 = slot;
strncpy(sf->sf3, str, sizeof(sf->sf3));
}
if ((slot_SF1 + 1 == slot_SF2) && (slot_SF2 + 1 == slot_SF3))
{
sf->slot = slot_SF1;
return 0;
}
}
}
return 1;
}

void Coordinate_calculation(Ephemeris* ep, uint32_t t, Coord *Position)
{
double Omega_e_dot = 7.2921151467e-5;
double mu = 3.986005e14;
double A = pow(ep->sqr_A, 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->M_0 + n * t_k;
double E_0 = M_k;

```

```

double E_k = 0;
int k = 0;
while (1)
{
E_k = E_0;
E_0 = E_0 + (M_k - E_0 + ep->e * sin(E_0)) / (1 - ep->e * cos(E_0));
if (abs(E_0 - E_k) < 1e-8){
break;
}
k = k + 1;
}
double nu = 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 = nu + ep->omega;
double delta_u = ep->C_u_s * sin(2 * Phi) + ep->C_u_c * cos(2 * Phi);
double delta_r = ep->C_r_s * sin(2 * Phi) + ep->C_r_c * cos(2 * Phi);
double delta_i = ep->C_i_s * sin(2 * Phi) + ep->C_i_c * cos(2 * Phi);
double u_k = Phi + delta_u;
double r = pow(ep->sqrt_A, 2) * (1 - ep->e * cos(E_k)) + delta_r;
double i_corr = ep->i_0 + delta_i + ep->iDot * t_k;
double x_shtr = r * cos(u_k);
double y_shtr = r * sin(u_k);
double Omega_corr = ep->Omega0 + (ep->OmegaDot - Omega_e_dot) * t_k
- Omega_e_dot * ep->toe;
double x_k = x_shtr * cos(Omega_corr) - y_shtr * cos(i_corr)
*sin(Omega_corr);
double y_k = x_shtr * sin(Omega_corr) + y_shtr * cos(i_corr)
*cos(Omega_corr);
double z_k = y_shtr * sin(i_corr);
Position->x = x_k;
Position->y = y_k;
Position->z = z_k;
}
void printEmp(Ephemeris* ep)
{
printf("LNAV Ephemeris (slot = %u) = \n", ep->slot);
printf("\tCrS = %e \n", ep->CrS);
printf("\tDn = %e \t[deg/s] \n", ep->Dn);
printf("\tM_0 = %f \t[deg] \n", ep->M_0);
printf("\tC_u_c = %e \n", ep->C_u_c);
printf("\te = %e \n", ep->e);
}

```

```

printf("\tC_u_s = %e \n", ep->C_u_s);
printf("\tsqrt_A = %e \n", ep->sqrt_A);
printf("\ttoe = %u \n", ep->toe);
printf("\tC_i_c = %e \n", ep->C_i_c);
printf("\tOmega0 = %f \t[deg] \n", ep->Omega0);
printf("\tC_i_S = %e \n", ep->C_i_S);
printf("\ti_0 = %f \t[deg] \n", ep->i_0);
printf("\tCrc = %e \n", ep->Crc);
printf("\tomega = %f \t[deg] \n", ep->omega);
printf("\tOmegaDot= %e \t[deg/s] \n", ep->OmegaDot);
printf("\tiDot = %e \t[deg/s] \n", ep->iDot);
printf("\tTgd = %e \n", ep->Tgd);
printf("\ttoc = %u \n", ep->toc);
printf("\taf2 = %e \n", ep->af2);
printf("\taf1 = %e \n", ep->af1);
printf("\taf0 = %e \n", ep->af0);
printf("\tWN = %u \n", ep->WN);
printf("\tIODC = %u \n", ep->IODC);
printf("\tR_A = %u \n", ep->R_A);
printf("\tHealth = %u \n", ep->Health);
printf("\tIODE2 = %u \n", ep->IODE2);
printf("\tIODE3 = %u \n", ep->IODE3);
printf("\tcodeL2 = %u \n", ep->codeL2);
printf("\tL2P = %u \n", ep->L2P);
}
void save(Ephemeris* ep, FILE* fod)
{
fprintf(fod,"LNAV Ephemeris (slot = %u) = \n", ep->slot);
fprintf(fod,"\tCrs = %e \n", ep->Crs);
fprintf(fod,"\tDn = %e \t[deg/s] \n", ep->Dn);
fprintf(fod,"\tM_0 = %f \t[deg] \n", ep->M_0);
fprintf(fod,"\tC_u_c = %e \n", ep->C_u_c);
fprintf(fod,"\te = %e \n", ep->e);
fprintf(fod, "\tC_u_s = %e \n", ep->C_u_s);
fprintf(fod, "\tsqrt_A = %e \n", ep->sqrt_A);
fprintf(fod, "\ttoe = %u \n", ep->toe);
fprintf(fod, "\tC_i_c = %e \n", ep->C_i_c);
fprintf(fod, "\tOmega0 = %f \t[deg] \n", ep->Omega0);
fprintf(fod, "\tC_i_S = %e \n", ep->C_i_S);
fprintf(fod, "\ti_0 = %f \t[deg] \n", ep->i_0);
fprintf(fod, "\tCrc = %e \n", ep->Crc);

```

```

fprintf(fod, "\\tomega = %f \\t[deg] \\n", ep->omega);
fprintf(fod, "\\tOmegaDot= %e \\t[deg/s] \\n", ep->OmegaDot);
fprintf(fod, "\\tiDot = %e \\t[deg/s] \\n", ep->iDot);
fprintf(fod, "\\tTgd = %e \\n", ep->Tgd);
fprintf(fod, "\\ttoc = %u \\n", ep->toc);
fprintf(fod, "\\taf2 = %e \\n", ep->af2);
fprintf(fod, "\\taf1 = %e \\n", ep->af1);
fprintf(fod, "\\taf0 = %e \\n", ep->af0);
fprintf(fod, "\\tWN = %u \\n", ep->WN);
fprintf(fod, "\\tIODC = %u \\n", ep->IODC);
fprintf(fod, "\\tR_A = %u \\n", ep->R_A);
fprintf(fod, "\\tHealth = %u \\n", ep->Health);
fprintf(fod, "\\tIODE2 = %u \\n", ep->IODE2);
fprintf(fod, "\\tIODE3 = %u \\n", ep->IODE3);
fprintf(fod, "\\tcodeL2 = %u \\n", ep->codeL2);
fprintf(fod, "\\tL2P = %u \\n", ep->L2P);
}

```