

ФГБОУ ВО
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ» »
ИНСТИТУТ РАДИОТЕХНИКИ И ЭЛЕКТРОНИКИ
НАПРАВЛЕНИЕ РАДИОЭЛЕКТРОННЫЕ СИСТЕМЫ И КОМПЛЕКСЫ
КАФЕДРА РАДИОТЕХНИЧЕСКИХ СИСТЕМ



Курсовой проект
по курсу «Аппаратура потребителей СРНС»

Выполнил студент:

Муратов Николай Сергеевич

группа: ЭР-15-17

Проверил:

к.т.н., доцент

Корогодин Илья Владимирович

Москва, 2022 г.

Содержание

1	Обработка логов навигационного приемника	3
1.1	Цель проекта	3
1.2	Задание	3
1.3	Основная часть	4
1.3.1	Теоретическая часть	4
1.3.2	Решение задачи	6
1.3.3	Приложение 1	9
1.4	Выводы	19
2	Моделирование траектории движения	20
2.1	Цель проекта	20
2.2	Задание	20
2.3	Основная часть	21
2.3.1	Пункт 1	21
2.3.2	Пункт 2	31
2.3.3	Пункт 3	33
2.4	Выводы	34
3	Реализация модуля расчета координат	35
3.1	Задание	35
3.2	Основная часть	36
3.2.1	Теоретическая часть	36
3.2.2	Решение задачи	36
3.2.3	Приложение 2	38
3.3	Выводы	51

Этап 1

Обработка логов навигационного приемника

1.1 Цель проекта

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 реализуем модуль разбора навигационного сообщения до структуры эфемерид, сравним результаты со сторонней программой.

1.2 Задание

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

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

1 0 013 0R GpsL1CA 13 212130404 29 125 53 100010111010...

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.

Муратов Николай Сергеевич: спутник №4.

Требуется:

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

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

1.3 Основная часть

1.3.1 Теоретическая часть

Целью передачи навигационных сообщений является возможность декодирования битового потока в параметры, с помощью которых можно определить положение устройства-приемника. Для определения своего местоположения требуется весь кадр сообщения. Но это не означает, что требуется непрерывный сбор данных. Это было бы очень неэффективно по времени и бесполезно. Как только данные были получены, они имеют срок службы, в течение которого они действительны. GPS эфемериды действительны в течение 2 часов, GPS альманах - в течение 24 часов. С момента получения этих данных, необходимо только постоянно обновлять время, коды дальности (и некоторые другие).

LNAV состоит из пяти подкадров. Подкадры 1-3 не содержат страниц, они декодируются сами по себе, поэтому они рассматриваются как наименьший декодируемый блок и включаются сами по себе. Подкадры 4-5 содержат определенные страницы. Каждая страница декодируется сама по себе, поэтому каждая страница рассматривается как наименьший декодируемый блок, но в данной работе нас интересуют лишь первые 3 подкадра.

Каждый подкадр состоит из 10 слов по 30 бит, то есть подкадр состоит из 300 символов. Одно слово может содержать как один, так и сразу несколько параметров. Но также существуют параметры разделенные на несколько слов (splitted parameters), пример их расположения можно увидеть на рис. 1.3.1.

Второй подкадр					
2	Preamble	Preamble	8	0	7
2	TLM_MSG	Telemetry_Message	14	8	21
2	ISF	Integrity_Status_Flag	1	22	22
2	Reserved	Reserved	1	23	23
2	P	Parity	6	24	29
2	MSG_TOW	Message_Time_Of_Week_Count	17	30	46
2	ALERT	Alert_Flag	1	47	47
2	ASF	Anti_Spoof_Flag	1	48	48
2	Sub_ID	Subframe_ID	3	49	51
2	t	Parity_Computation	2	52	53
2	P	Parity	6	54	59
2	IODE	E_IOD_Ephemeris	8	60	67
2	C_rs	E_Harmonic_Corr_rs	16	68	83
2	P	Parity	6	84	89
2	DEL_n	E_Mean_Motion_Diff	16	90	105
2	M_0	E_Mean_Anomaly_At_Ref_T	8	106	113
2	P	Parity	6	114	119
2	M_0	E_Mean_Anomaly_At_Ref_T	24	120	143
2	P	Parity	6	144	149
2	C_uc	E_Harmonic_Corr_uc	16	150	165
2	e	E_Eccentricity	8	166	173
2	P	Parity	6	174	179
2	e	E_Eccentricity	24	180	203
2	P	Parity	6	204	209
2	C_us	E_Harmonic_Corr_us	16	210	225
2	A_1/2	E_SQRT_Semi_Major_Axis	8	226	233
2	P	Parity	6	234	239
2	A_1/2	E_SQRT_Semi_Major_Axis	24	240	263
2	P	Parity	6	264	269
2	t_0e	E_Ref_T	16	270	285
2	FIF	Fit_Interval_Flag	1	286	286
2	AODO	Age_Of_Data_Offset	5	287	291
2	t	Parity_Computation	2	292	293
2	P	Parity	6	294	299

Рисунок 1.3.1 — LNAV Подкадр(Subframe) 2 структурирован в электронную таблицу. Строки, выделенные красным цветом являются разделенными параметрами

1.3.2 Решение задачи

Имеем входные данные:

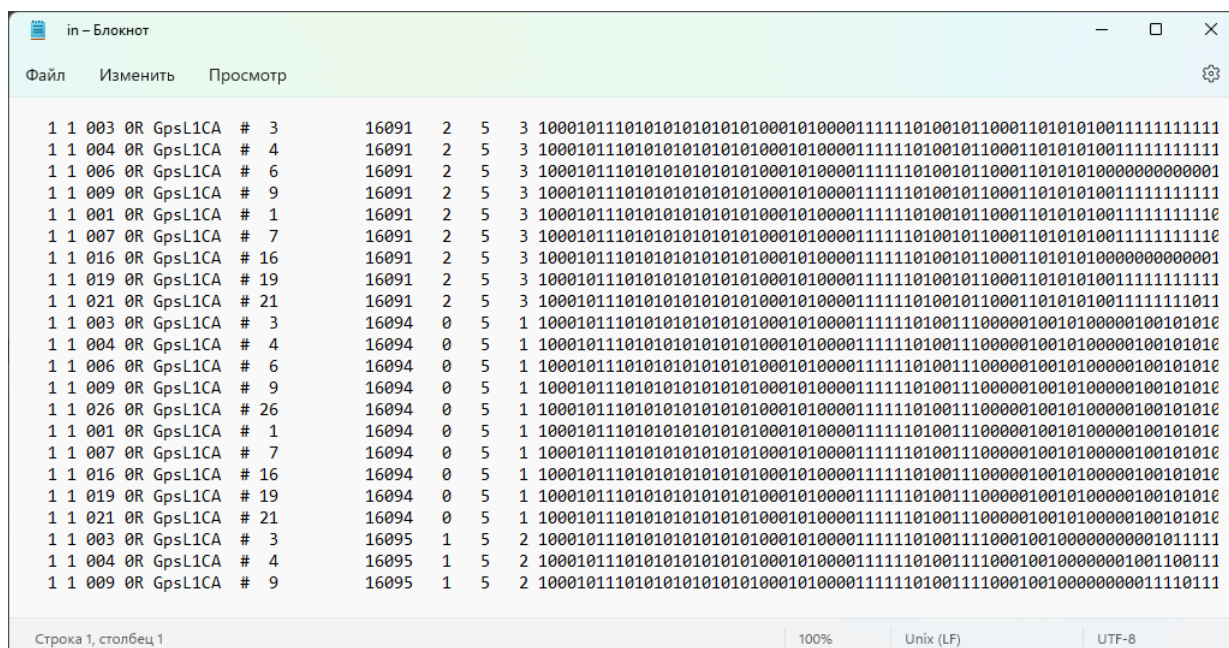
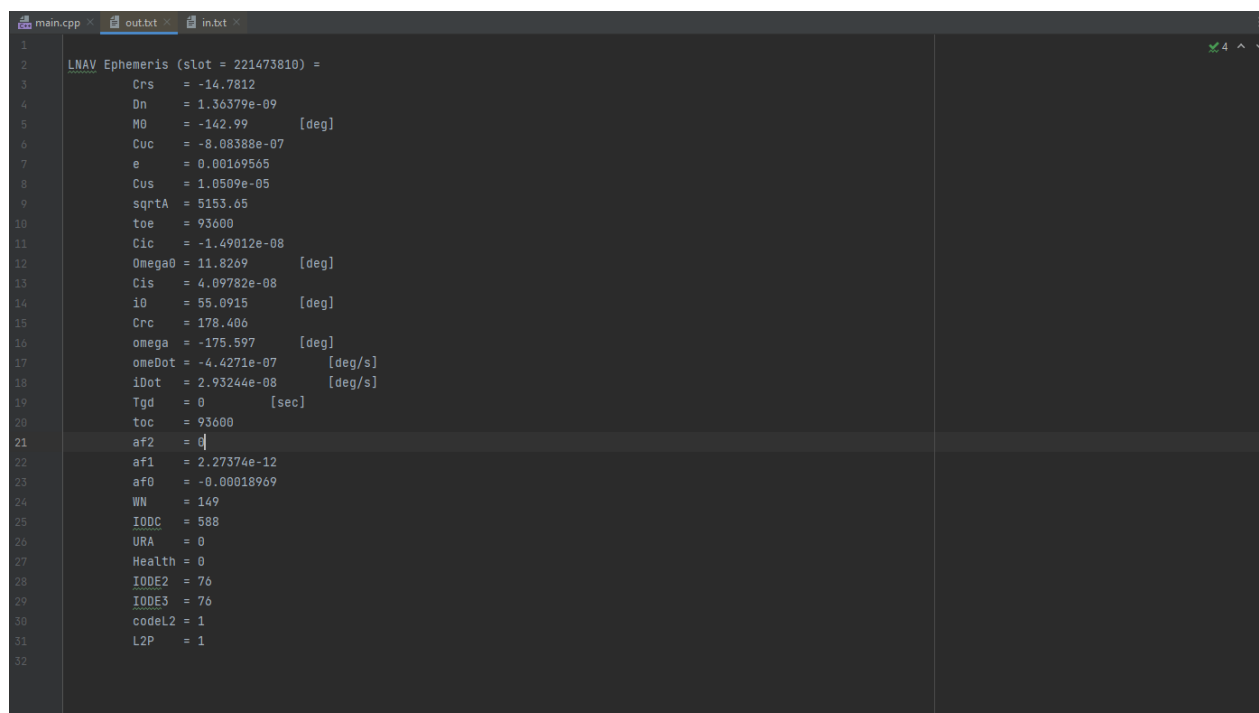


Рисунок 1.3.2 — Входной файл in.txt

Задача решается в несколько следующих этапов:

- Открыть файл;
- Найти в потоке подкадров первые три, исходящие от необходимого спутника;
- Извлечь их в отдельную структуру данных;
- Достать из подкадров данные эфемерид (данный этап для некоторых переменный состоит из двух действий, т.к. необходимо не только выбрать необходимые биты, но и преобразовать их из дополнительного кода в прямой с помощью оператора побитового дополнения - это унарный оператор (работает только с одним операндом). Он обозначается символом \sim и меняет двоичные цифры 1 на 0 и 0 на 1;
- Вывести найденные параметры в терминал;
- Сохранить в файл out.txt

Листинг программы располагается в Приложении 2. Выходной файл out.txt:



```
1
2 LNAV Ephemeris (slot = 221473810) =
3   Crs   = -14.7812
4   Dn    = 1.36379e-09
5   M0    = -142.99      [deg]
6   Cus   = -8.08388e-07
7   e     = 0.00109565
8   Cus   = 1.0509e-05
9   sqrtA = 5153.65
10  toe    = 93600
11  Cic    = -1.49012e-08
12  Omega0 = 11.8269     [deg]
13  Cis    = 4.09782e-08
14  i0     = 55.0915     [deg]
15  Crc    = 178.406
16  omega  = -175.597    [deg]
17  omeDot = -4.4271e-07 [deg/s]
18  iDot   = 2.93244e-08 [deg/s]
19  Tgd    = 0           [sec]
20  toc    = 93600
21  af2    = 0
22  af1    = 2.27374e-12
23  af0    = -0.00018969
24  WN     = 149
25  IODC   = 588
26  URA    = 0
27  Health = 0
28  IODE2  = 76
29  IODE3  = 76
30  codeL2 = 1
31  L2P    = 1
32
```

Рисунок 1.3.3 — Выходной файл

Приведем данные, обработанные open-source программой RTKNAVI
рис. 1.3.5:

RTKNAVI ver.2.4.2: RTK Monitor															Close	
Nav GPS		All		Current												
SAT	PRN	Status	IDOE	IDOC	Acci	Health	Toe	Toc	Ttrans	A (m)	e	i0 (deg)	OMEGA0 (deg)	omega (deg)	M0 (deg)	deltan (deg/s)
G01	1	-	5911	23	0	00	2022/02/14 00:00:00	2022/02/14 00:00:00	2022/04/15 14:46:39	26560304.325	0.01137974	56.53057	-109.43551	50.51432	54.55674	2.4630E-07
G02	2	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G03	3	OK	2827	11	0	00	2022/02/14 01:59:44	2022/02/14 01:59:44	2022/04/15 14:46:39	26560504.265	0.00389213	55.72485	-50.10616	54.89203	48.47481	2.3572E-07
G04	4	OK	1953	588	0	00	2022/02/14 02:00:00	2022/02/14 02:00:00	2022/04/15 14:46:39	26560157.291	0.00169565	55.09149	11.82685	-175.5971E	-142.9901E	2.4548E-07
G05	5	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G06	6	OK	2210	86	0	00	2022/02/14 02:00:00	2022/02/14 02:00:00	2022/04/15 14:46:39	26560290.249	0.00269551	56.49500	-109.91291	-55.01959	133.64356	2.4669E-07
G07	7	OK	8995	35	0	00	2022/02/14 04:00:00	2022/02/14 04:00:00	2022/04/15 14:46:39	26560462.036	0.01554191	54.46659	70.20050	-131.2703E	136.52698	2.8489E-07
G08	8	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G09	9	OK	7710	30	0	00	2022/02/14 02:00:00	2022/02/14 02:00:00	2022/04/15 14:46:39	26559848.321	0.00225987	54.69112	8.79464	107.10939	-94.65112	2.5565E-07
G10	10	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G11	11	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G12	12	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G13	13	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G14	14	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G15	15	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G16	16	OK	5140	20	0	00	2022/02/14 04:00:00	2022/02/14 04:00:00	2022/04/15 14:46:39	26559618.285	0.01275745	55.59110	135.03970	40.56887	-7.13304	2.4325E-07
G17	17	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G18	18	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G19	19	OK	9766	38	0	00	2022/02/14 02:00:00	2022/02/14 02:00:00	2022/04/15 14:46:39	26560503.242	0.00897000	56.08838	-164.03953	115.03807	17.44108	2.3429E-07
G20	20	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G21	21	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G22	22	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G23	23	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G24	24	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G25	25	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G26	26	OK	2313	9	0	00	2022/02/14 02:00:00	2022/02/14 02:00:00	2022/04/15 14:46:39	26560614.536	0.00687526	53.77037	126.89030	21.18201	-13.00067	2.8919E-07
G27	27	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G28	28	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G29	29	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G30	30	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00
G31	31	OK	6168	24	0	00	2022/02/14 02:00:00	2022/02/14 02:00:00	2022/02/14 03:01:19	26560409.642	0.01045204	54.70826	71.24072	21.48581	50.45602	2.8242E-07
G32	32	-	-	-	0	00	-	-	-	0.000	0.00000000	0.00000	0.00000	0.00000	0.00000	0.0000E+00

Рисунок 1.3.4 — RTKNAVI результат 1/2

RTKNAVI ver.2.4.2: RTK Monitor																	Close
Nav GPS	All	Current															
deltan (deg/s)	OMEGA0dot (d)	IDOT (deg/s)	a0 (ns)	af1 (ns/s)	af2 (ns/s2)	TGD (ns)	BGD5a(ns)	BGD5b(ns)	Cuc(rad)	Cus(rad)	Crc(m)	Crs(m)	Cic(rad)	Cis(rad)	Code	Flag	
2.4630E-07	-4.8124E-07	-6.7735E-09	432151.8	-0.0095	0.0000	5.1	0.0	0.0	7.7300E-07	1.1530E-06	3.7197E+02	1.1719E+01	-1.1735E-07	1.1362E-07	1	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
2.3572E-07	-4.5681E-07	-1.7149E-08	-120342.7	-0.0172	0.0000	1.9	0.0	0.0	-6.6254E-06	7.0781E-06	2.4825E+02	-1.2734E+02	-4.4703E-08	3.7253E-08	1	0	
2.4548E-07	-4.4271E-07	2.9324E-08	-189689.9	0.0023	0.0000	-4.2	0.0	0.0	-8.0839E-07	1.0509E-05	1.7841E+02	-1.4781E+01	-1.4901E-08	4.0978E-08	1	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
2.4669E-07	-4.7165E-07	-7.1623E-10	200809.4	0.0119	0.0000	3.7	0.0	0.0	1.0580E-06	5.7183E-07	3.7962E+02	1.5906E+01	6.5193E-08	-1.3039E-08	1	0	
2.8489E-07	-4.5906E-07	7.3464E-09	310124.8	0.0027	0.0000	-11.2	0.0	0.0	3.5018E-07	5.3551E-06	2.7103E+02	1.0562E+01	-6.5193E-08	-2.8871E-07	1	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
2.5565E-07	-4.4940E-07	2.5354E-08	-354265.8	0.0016	0.0000	1.4	0.0	0.0	-1.2182E-06	1.0680E-05	1.7066E+02	-2.2969E+01	-2.4214E-08	1.0431E-07	1	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00	0.0	0.0000	0.0000	0.0	0.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0	0	
0.0000E+00	0.0000E+00	0.0000E+00															

1.3.3 Приложение 1

Листинг 1.1 — Программа декодирования подкадров навигационного сообщения GPS

```
1 // Made on Earth by Murathon.
2
3
4 #include <iostream>
5 #include <fstream>
6 #include <windows.h>
7 #include <string>
8 #include <stdlib.h>
9 #include <cmath>
10 #include <stdio.h>
11 #include <bitset>
12
13 #define P2_5      pow(2,-5)
14 #define P2_55     pow(2,-55)
15 #define P2_43     pow(2,-43)
16 #define P2_31     pow(2,-31)
17 #define P2_4      pow(2,4)
18 #define P2_19     pow(2,-19)
19 #define P2_29     pow(2,-29)
20 #define P2_33     pow(2,-33)
21 #define SemiCirc  180
22
23
24 using namespace std;
25
26 struct Ephemeris {
27     double    Crs;
28     double    Dn;
29     double    M0;
30     double    Cuc;
31     double    e;
32     double    Cus;
33     double    sqrtA;
34     uint32_t  toe;
35     double    Cic;
```

```

36     double    Omega0;
37     double    Cis;
38     double    i0;
39     double    Crc;
40     double    omega;
41     double    OmegaDot;
42     double    iDot;
43     int16_t    Tgd;
44     uint32_t    toc;
45     double    af2;
46     double    af1;
47     double    af0;
48     uint16_t    WN;
49     uint16_t    IODC;
50     uint32_t    URA;
51     uint32_t    Health;
52     uint16_t    IODE2;
53     uint16_t    IODE3;
54     bool        codeL2;
55     bool        L2P;
56     uint32_t    slot;
57 };
58
59 struct nssrSF {
60     uint32_t    slot;
61     string    sf1;
62     string    sf2;
63     string    sf3;
64 };
65
66 void parsStr(nssrSF *Rdata);
67 uint32_t pickSplitParam(string sf, uint16_t FS, int HMF, uint16_t
    FtS, int HMS);
68 void printEPH(Ephemeris* EPH);
69 void saveEPH(Ephemeris* EPH);
70 void decodeSF(Ephemeris* EPH, nssrSF *data);
71 int64_t pick32(string sf, int32_t FrmN, int HmR);
72
73
74 int main(void)

```

```

75 {
76     nssrSF data;
77     parsStr(&data);
78     cout << data.sf1 << endl << data.sf2 << endl << data.sf3 << endl
;
79     Ephemeris *EPH = (Ephemeris*) calloc(1, sizeof(Ephemeris));
80     decodeSF(EPH,&data);
81     printEPH(EPH);
82     saveEPH(EPH);
83     free(EPH);
84
85
86 }
87
88 void parsStr(nssrSF *Rdata)
89 {
90
91     SetConsoleOutputCP(CP_UTF8);
92     string path = "in.txt";
93
94     ifstream fin;
95     fin.open(path);
96
97     if(fin.is_open()) {
98
99         cout << "Файл " << "открыт." << endl;
100         while (!fin.eof()) {
101             int N = 3;
102             string rubber;
103             int nmbr_stlt;
104             uint32_t slot;
105             uint32_t subFrameNum;
106             string str;
107
108             uint32_t slot_SF1;
109             uint32_t slot_SF2;
110             uint32_t slot_SF3;
111
112             int mass[N];
113             for(int i=0;i<N;++i)

```

```

114         {
115             fin >> mass[i];
116         }
117         fin >> rubber >> rubber >> rubber;
118         fin >> nmbr_stlt >> slot >> rubber >> rubber >>
subFrameNum;
119         fin >> str;
120
121         if (nmbr_stlt == 4 and slot >= 604800/6){
122
123             if (subFrameNum == 1)
124             {
125                 slot_SF1 = slot;
126                 Rdata->sf1 = str;
127             }
128             else if (subFrameNum == 2)
129             {
130                 slot_SF2 = slot;
131                 Rdata->sf2 = str;
132             }
133             else if (subFrameNum == 3)
134             {
135                 slot_SF3 = slot;
136                 Rdata->sf3 = str;
137             }
138
139
140             if (slot_SF1 + 1 == slot_SF2 and slot_SF2 + 1 ==
slot_SF3) {
141                 Rdata->slot = slot_SF1;
142                 return;
143             }
144
145         }
146         //
147     }
148 }
149 else
150 {
151     cout << "Ошибка" открытия файла!!!" << endl;

```

```

152     }
153     fin.close();
154
155
156 }
157 void saveEPH(Ephemeris* EPH)
158 {
159     ofstream fout;
160     string path = "out.txt";
161     fout.open(path);
162     if(fout.is_open()) {
163
164         cout << "Выгружаю" << endl;
165         fout << endl << "LNAV Ephemeris (slot = " << EPH->slot << ")
=< endl;
166         fout << "\t\t Crs = " << EPH->Crs << endl;
167         fout << "\t\t Dn      = " << EPH->Dn << endl;
168         fout << "\t\t M0      = " << EPH->M0 << "\t\t[deg]" << endl;
169         fout << "\t\t Cuc = " << EPH->Cuc << endl;
170         fout << "\t\t e      = " << EPH->e << endl;
171         fout << "\t\t Cus = " << EPH->Cus << endl;
172         fout << "\t\t sqrtA = " << EPH->sqrtA << endl;
173         fout << "\t\t toe = " << EPH->toe << endl;
174         fout << "\t\t Cic = " << EPH->Cic << endl;
175         fout << "\t\t Omega0 = " << EPH->Omega0 << "\t\t[deg]" <<
endl;
176         fout << "\t\t Cis = " << EPH->Cis << endl;
177         fout << "\t\t i0      = " << EPH->i0 << "\t\t[deg]" << endl;
178         fout << "\t\t Crc = " << EPH->Crc << endl;
179         fout << "\t\t omega = " << EPH->omega << "\t\t[deg]" << endl
;
180         fout << "\t\t omeDot = " << EPH->OmegaDot << "\t\t[deg/s]"
<< endl;
181         fout << "\t\t iDot  = " << EPH->iDot << "\t\t[deg/s]" <<
endl;
182         fout << "\t\t Tgd = " << EPH->Tgd << "\t\t\t[sec]" << endl;
183         fout << "\t\t toc = " << EPH->toc << endl;
184         fout << "\t\t af2 = " << EPH->af2 << endl;
185         fout << "\t\t af1 = " << EPH->af1 << endl;
186         fout << "\t\t af0 = " << EPH->af0 << endl;

```

```

187     fout << "\t\t WN      = " << EPH->WN << endl;
188     fout << "\t\t IODC   = " << EPH->IODC << endl;
189     fout << "\t\t URA    = " << EPH->URA << endl;
190     fout << "\t\t Health  = " << EPH->Health << endl;
191     fout << "\t\t IODE2   = " << EPH->IODE2 << endl;
192     fout << "\t\t IODE3   = " << EPH->IODE3 << endl;
193     fout << "\t\t codeL2  = " << EPH->codeL2 << endl;
194     fout << "\t\t L2P    = " << EPH->L2P << endl;
195 }
196 else
197 {
198     cout << "Ошибка " открытия файла !!!" << endl;
199 }
200 fout.close();
201 cout << "Готово" !";
202 }
203
204 void printEPH(Ephemeris* EPH)
205 {
206
207     cout << endl << "LNAV Ephemeris (slot = " << EPH->slot << ") ="
<< endl;
208     cout << "\t\t Crs = " << EPH->Crs << endl;
209     cout << "\t\t Dn  = " << EPH->Dn << endl;
210     cout << "\t\t M0  = " << EPH->M0 << "\t\t[deg]" << endl;
211     cout << "\t\t Cuc = " << EPH->Cuc << endl;
212     cout << "\t\t e  = " << EPH->e << endl;
213     cout << "\t\t Cus = " << EPH->Cus << endl;
214     cout << "\t\t sqrtA = " << EPH->sqrtA << endl;
215     cout << "\t\t toe = " << EPH->toe << endl;
216     cout << "\t\t Cic = " << EPH->Cic << endl;
217     cout << "\t\t Omega0 = " << EPH->Omega0 << "\t\t[deg]" << endl;
218     cout << "\t\t Cis = " << EPH->Cis << endl;
219     cout << "\t\t i0  = " << EPH->i0 << "\t\t[deg]" << endl;
220     cout << "\t\t Crc = " << EPH->Crc << endl;
221     cout << "\t\t omega = " << EPH->omega << "\t\t[deg]" << endl;
222     cout << "\t\t omeDot = " << EPH->OmegaDot << "\t\t[deg/s]" <<
endl;
223     cout << "\t\t iDot  = " << EPH->iDot << "\t\t[deg/s]" << endl;
224     cout << "\t\t Tgd = " << EPH->Tgd << "\t\t\t[sec]" << endl;

```

```

225     cout << "\t\t toc = " << EPH->toc << endl;
226     cout << "\t\t af2 = " << EPH->af2 << endl;
227     cout << "\t\t af1 = " << EPH->af1 << endl;
228     cout << "\t\t af0 = " << EPH->af0 << endl;
229     cout << "\t\t WN = " << EPH->WN << endl;
230     cout << "\t\t IODC = " << EPH->IODC << endl;
231     cout << "\t\t URA = " << EPH->URA << endl;
232     cout << "\t\t Health = " << EPH->Health << endl;
233     cout << "\t\t IODE2 = " << EPH->IODE2 << endl;
234     cout << "\t\t IODE3 = " << EPH->IODE3 << endl;
235     cout << "\t\t codeL2 = " << EPH->codeL2 << endl;
236     cout << "\t\t L2P = " << EPH->L2P << endl;
237
238 }
239 int64_t pick32(string sf, int32_t FrmN, int HmR) {
240     int64_t ans = 0;
241     int64_t Rans = 0;
242     for (int i = FrmN; i < FrmN+HmR; i++) {
243         ans = (ans | ((sf[i - 1] == '1') ? 1 : 0));
244         cout << sf[i-1];
245         if (i < FrmN+HmR-1){
246             ans = ans<<1;
247         }
248
249     }
250     return ans;
251
252
253 }
254
255 int64_t compl2int(uint64_t ans, int HmZ){
256     int64_t Rans = 0;
257     if (HmZ == 8){
258         if (bool((1<<7) & ans)){
259             ans /= 0xFFFFFFFFFFFFFFF00;
260             Rans = ~(ans-1);
261             /*cout<< endl << bitset<64>(Rans).to_string() << endl;*/
262             return -Rans;
263         }
264

```

```

265 }
266 if (HmZ == 14){
267     if (bool((1<<13) & ans)){
268         ans |= 0xFFFFFFFFFFC000;
269         Rans = ~(ans-1);
270         return -Rans;
271     }
272
273 }
274 if (HmZ == 16){
275     if (bool((1<<15) & ans)){
276         ans |= 0xFFFFFFFFF0000;
277         Rans = ~(ans-1);
278         return -Rans;
279     }
280
281 }
282 if (HmZ == 22){
283     if (bool((1<<21) & ans)){
284         ans |= 0xFFFFFFFFC00000;
285         Rans = ~(ans-1);
286         return -Rans;
287     }
288
289 }
290 if (HmZ == 24){
291     if (bool((1<<23) & ans)){
292         ans |= 0xFFFFFFFFF000000;
293         Rans = ~(ans-1);
294         return -Rans;
295     }
296
297 }
298 if (HmZ == 32){
299     if (bool((1<<31) & ans)){
300         ans |= 0xFFFFFFFFF00000000;
301         Rans = ~(ans-1);
302         return -Rans;
303     }
304

```



```

305     }
306     return ans;
307 }
308 uint32_t pickSplitParam(string sf, uint16_t FS, int HMF, uint16_t
    FtS, int HMS) {
309     uint32_t ans = 0;
310
311     for (int i = FS; i < FS+HMF; i++) {
312         ans = (ans | ((sf[i-1] == '1')? 1 : 0)) << 1;
313     }
314     for (int i = FtS; i < FtS+HMS; i++) {
315         ans = ans | ((sf[i-1] == '1')? 1 : 0);
316         if (i < FtS+HMS-1){
317             ans = ans<<1;
318         }
319     }
320
321     return ans;
322 }
323 }
324
325 void decodeSF(Ephemeris* EPH, nssrSF *data){
326
327     EPH->slot = data->slot;
328
329     EPH->Crs = compl2int(pick32(data->sf2,69,16),16)*P2_5;
330
331     EPH->Dn = compl2int(pick32(data->sf2,91,16),16)*P2_43;
332
333     EPH->M0 = compl2int(pickSplitParam(data->sf2,107, 8, 121, 24)
    ,32)*P2_31*SemiCirc;
334
335     EPH->Cuc = compl2int(pick32(data->sf2,151,16),16)*P2_29;
336
337     EPH->e = pickSplitParam(data->sf2,167, 8, 181, 24) * P2_33;
338
339     EPH->Cus = compl2int(pick32(data->sf2,211,16),16)*P2_29;
340
341     EPH->sqrta = pickSplitParam(data->sf2,227, 8, 241, 24) * P2_19;
342

```

```

343  EPH->toe = pick32(data->sf2,271,16)*pow(2,4);
344
345  EPH->Cic = compl2int(pick32(data->sf3,61,16),16)*P2_29;
346
347  EPH->Omega0 = compl2int(pickSplitParam(data->sf3,77, 8, 91, 24)
,32)*P2_31*SemiCirc;
348
349  EPH->Cis = compl2int(pick32(data->sf3,121,16),16)*P2_29;
350
351  EPH->i0 = compl2int(pickSplitParam(data->sf3,137, 8, 151, 24)
,32)*P2_31*SemiCirc;
352
353  EPH->Crc = compl2int(pick32(data->sf3,181,16),16)*P2_5;
354
355  EPH->omega = compl2int(pickSplitParam(data->sf3,197, 8, 211, 24)
,32)*P2_31*SemiCirc;
356
357  EPH->OmegaDot = compl2int(pick32(data->sf3,241,24),24)*P2_43*
SemiCirc;
358
359  EPH->iDot = compl2int(pick32(data->sf3,279,14),14)*P2_43*
SemiCirc;
360
361  EPH->Tgd = compl2int(pick32(data->sf1,197,8),8)*P2_31;
362
363  EPH->toc = compl2int(pick32(data->sf1,219,16),16)*P2_4;
364
365  EPH->af2 = compl2int(pick32(data->sf1,241,8),8)*P2_55;
366
367  EPH->af1 = compl2int(pick32(data->sf1,249,16),16)*P2_43;
368
369  EPH->af0 = compl2int(pick32(data->sf1,271,22),22)*P2_31;
370
371  EPH->WN = pick32(data->sf1,61,10);
372
373  EPH->IODC = pickSplitParam(data->sf1,83, 2, 211, 8);
374
375  EPH->URA = pick32(data->sf1,73,4);
376
377  EPH->Health = EPH->IODE2 = pick32(data->sf1,73,6);

```

```

378
379     EPH->IODE2 = pick32(data->sf2,61,8);
380
381     EPH->IODE3 = pick32(data->sf3,271,8);
382
383     EPH->codeL2 = pick32(data->sf1,71,2);
384
385     EPH->L2P = data->sf1[90];
386
387 }

```

1.4 Выводы

На первом этапе были получены эфемериды для дальнейшего написания функцию расчета положения спутника GPS на языке python.

Этап 2

Моделирование траектории движения

2.1 Цель проекта

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 реализуем модуль разбора навигационного сообщения до структуры эфемерид, сравним результаты со сторонней программой.

2.2 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.

Муратов Николай Сергеевич: спутник №4. Язык: Python.

Оформить отчет по результатам этапа:

- Реализация в Python;
- 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;
- Оформить отчет по этапу и разместить на Github;
- Завести Pull Request.

2.3 Основная часть

2.3.1 Пункт 1

Так как первой развернутой на орбите ГНСС была GPS (NAVSTAR), то и первой ГНСС шкалой времени стала GPS Time (TGPS), которая сейчас отличается от UTC на 18с. Эта величина называется leap second. Эпоха в шкале времени GPS определяется номером недели (GPS Week) и номером секунды в неделе. Начало отсчета этой шкалы приходится на ночь с субботы на воскресенье 6 января 1980 г. в 00:00 ч (UTC). Каждая новая неделя также начинается в ночь с субботы на воскресенье. Номер GPS недели передается в навигационном сообщении в 10-битном поле, по этой причине по прошествии 1024 недель счетчик обнуляется. Этот эффект назван GPS week number rollover и происходит каждые 19,7 лет. Первый сброс недели произошел 21 августа 1999г, второй – 6 апреля 2019г.

GPS представление времени:

Номер недели: номер секунды от начала недели с учетом мкс.

В предыдущем пункте мы нашли номер недели и номер секунды от начала недели:

2197:93600

Переведем в UTC:

14/02/2022 03:04:44

Напишем по алгоритму из ИКД программу на языке Python, рассчитывающую координаты заданного спутника на интервале суток.

Table 30-II. Broadcast Navigation User Equations (sheet 1 of 4)

Element/Equation	Description
$\mu = 3.986005 \times 10^{14} \text{ meters}^3/\text{sec}^2$	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
$A_0 = A_{REF} + \Delta A^*$	Semi-Major Axis at reference time
$A_k = A_0 + (\dot{A}) t_k$	Semi-Major Axis
$n_0 = \sqrt{\frac{\mu}{A_0^3}}$	Computed Mean Motion (rad/sec)
$t_k = t - t_{oc}^{**}$	Time from ephemeris reference time
$\Delta n_A = \Delta n_0 + \frac{1}{2} \Delta \dot{n}_0 t_k$	Mean motion difference from computed value
$n_A = n_0 + \Delta n_A$	Corrected Mean Motion
$M_k = M_0 + n_A t_k$	Mean Anomaly
$E_0 = M_k$	Kepler's equation ($M_k = E_k - e \sin E_k$) may be solved for Eccentric Anomaly (E_k) by iteration: – 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)
<p>* $A_{REF} = 26,559,710 \text{ meters}$</p> <p>** t is GPS system time at time of transmission, i.e., GPS time corrected for transit time (range/speed of light). Furthermore, t_k shall be the actual total difference between the time t and the epoch time t_{oc}, and must account for beginning or end of week crossovers. That is if t_k is greater than 302,400 seconds, subtract 604,800 seconds from t_k. If t_k is less than -302,400 seconds, add 604,800 seconds to t_k.</p>	

Table 30-II. Broadcast Navigation User Equations (sheet 2 of 4)

Element/Equation *	Description
$\Phi_k = \nu_k + \omega_n$ $\delta u_k = C_{us-n} \sin 2\Phi_k + C_{uc-n} \cos 2\Phi_k$ $\delta r_k = C_{rs-n} \sin 2\Phi_k + C_{rc-n} \cos 2\Phi_k$ $\delta i_k = C_{is-n} \sin 2\Phi_k + C_{ic-n} \cos 2\Phi_k$	Argument of Latitude Argument of Latitude Correction Radial Correction Inclination Correction <div style="float: right; text-align: right;"> } Second Harmonic } Perturbations </div>
$u_k = \Phi_k + \delta u_k$ $r_k = A_k(1 - e_n \cos E_k) + \delta r_k$ $i_k = i_{0-n} + (i_{0-n} - \text{DOT})t_k + \delta i_k$	Corrected Argument of Latitude Corrected Radius Corrected Inclination
$x_k' = r_k \cos u_k$ $y_k' = r_k \sin u_k$	Positions in orbital plane
$\dot{\Omega} = \dot{\Omega}_{\text{REF}} + \Delta \dot{\Omega} \quad ***$ $\Omega_k = \Omega_{0-n} + (\dot{\Omega} - \dot{\Omega}_e) t_k - \dot{\Omega}_e t_{os}$	Rate of Right Ascension 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 of SV antenna phase center
*** $\dot{\Omega}_{\text{REF}} = -2.6 \times 10^{-9}$ semi-circles/second.	

Имеем входные данные:

```

1  LNAV Ephemeris (slot = 221473810) =
2  Crs    = -14.7812
3  Dn     = 1.36379e-09
4  M0     = -142.99      [deg]
5  CUC    = -8.08388e-07
6  e      = 0.00149565
7  Cus    = 1.0509e-05
8  sqrtA  = 5153.05
9  toe    = 93600
10 C1c    = -1.49012e-08
11 Omega0 = 11.8269      [deg]
12 C1s    = 4.09782e-08
13 i0     = 55.0915      [deg]
14 Crc    = 178.406
15 omega  = -175.597     [deg]
16 omeDot = -4.4271e-07  [deg/s]
17 iDot   = 2.93244e-08  [deg/s]
18 Tgd    = 0            [sec]
19 toc    = 93600
20 af2    = 0
21 af1    = 2.27374e-12
22 af0    = -0.00018969
23 WN     = 149
24 IODE   = 588
25 URA    = 0
26 Health = 0
27 IODE2  = 70
28 IODE3  = 70
29 codeL2 = 1
30 L2P    = 1
31
32

```

Рисунок 2.3.1 — Входной файл in.txt

Листинг 2.1 — Программа декодирования подкадров навигационного сообщения GPS

```
1 # Made on Earth by Murathon
2 import math
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import pymap3d
6
7
8 class TakeCoordinate():
9
10     def __init__(self, t):
11         self.time = t
12         self.XYZ = np.zeros((1, 4))
13         self.XYZv = np.zeros((1, 4))
14         self.WGS_XYZ = np.zeros((1, 4))
15         self.WGS_XYZv = np.zeros((1, 4))
16
17     def coordinates(self):
18         for i in range(86400):
19             print(i)
20             self.XYZv[0, 0] = i
21             self.WGS_XYZv[0, 0] = i
22             t_k = self.time[i] - t_oe
23
24             if t_k > 302400:
25                 t_k -= 604800
26             if t_k < -302400:
27                 t_k += 604800
28
29             n_0 = math.sqrt(mu / A ** 3)
30             n = n_0 + Dn
31
32             E_old = M_0 + n * t_k
33
34             E_old = E_old
35             E_new = E_old + e_n * math.sin(E_old)
36             i = 1
37             while abs(E_new - E_old) > 1e-8:
```



```

38         E_old = E_new
39         E_new = E_old + e_n * math.sin(E_old)
40         i += 1
41         if (i > 10):
42             break
43         E_k = E_new
44
45         cosNu_k = (math.cos(E_k) - e_n) / (1 - e_n * math.cos(
E_k))
46         sinNu_k = (math.sqrt(1 - e_n ** 2) * math.sin(E_k)) / (1
- e_n * math.cos(E_k))
47
48         Nu_k = math.atan2(sinNu_k, cosNu_k)
49         Phi_k = Nu_k + omega
50
51         delta_u_k = Cus * math.sin(2 * Phi_k) + Cuc * math.cos(2
* Phi_k)
52         delta_r_k = Crs * math.sin(2 * Phi_k) + Crc * math.cos(2
* Phi_k)
53         delta_i_k = Cis * math.sin(2 * Phi_k) + Cic * math.cos(2
* Phi_k)
54
55         u_k = Phi_k + delta_u_k
56         r_k = A * (1 - e_n * math.cos(E_k)) + delta_r_k
57         i_k = i_0 + iDot * t_k + delta_i_k
58
59         x_k_shtrih = r_k * math.cos(u_k)
60         y_k_shtrih = r_k * math.sin(u_k)
61
62         Omega_k = Omega_0 + (Omega_dot - Omega_e_Dot) * t_k -
Omega_e_Dot * t_oe
63
64         x_k = x_k_shtrih * math.cos(Omega_k) - y_k_shtrih * math
.cos(i_k) * math.sin(Omega_k)
65         y_k = x_k_shtrih * math.sin(Omega_k) + y_k_shtrih * math
.cos(i_k) * math.cos(Omega_k)
66         z_k = y_k_shtrih * math.sin(i_k)
67         self.WGS_XYZv[0, 1] = x_k
68         self.WGS_XYZv[0, 2] = y_k
69         self.WGS_XYZv[0, 3] = z_k

```

```

70         self.WGS_XYZ = np.vstack((self.WGS_XYZ, self.WGS_XYZv))
71
72         X = x_k * math.cos(Omega_k) + y_k * math.sin(Omega_k)
73         Y = -x_k * math.sin(Omega_k) + y_k * math.cos(Omega_k)
74         Z = z_k
75
76         self.XYZv[0, 1] = X
77         self.XYZv[0, 2] = Y
78         self.XYZv[0, 3] = Z
79         self.XYZ = np.vstack((self.XYZ, self.XYZv))
80         np.savetxt("XYZ.txt", self.XYZ)
81         np.savetxt("WGS_XYZ.txt", self.WGS_XYZ)
82
83
84 if __name__ == '__main__':
85
86     gps_pi = 3.1415926535898
87     toRad = gps_pi / 180
88     sec2rad = gps_pi / (3600 * 180)
89     min2rad = gps_pi / (60 * 180)
90     mu = 3.986005e14
91     Omega_e_Dot = 7.2921151467e-5
92     Omega_ref_Dot = -2.6e-9
93
94     # LNAV Ephemeris(slot=221473810) =
95     Crs = -14.7812
96     Dn = 1.36379e-09 * toRad
97     M_0 = -142.99 * toRad
98     Cuc = -8.08388e-07
99     e_n = 0.00169565
100    Cus = 1.0509e-05
101    A = 5153.65 ** 2
102    t_oe = 93600
103    Cic = -1.49012e-08
104    Omega_0 = 11.8269 * toRad
105    Cis = 4.09782e-08
106    i_0 = 55.0915 * toRad
107    Crc = 178.406
108    omega = -175.597 * toRad
109    Omega_dot = -4.4271e-07 * toRad

```

```

110 iDot = 2.93244e-08 * toRad
111 Tgd = 0
112 toc = 93600
113 af2 = 0
114 af1 = 2.27374e-12
115 af0 = -0.00018969
116 WN = 149
117 IODC = 588
118 URA = 0
119 Health = 0
120 IODE2 = 76
121 IODE3 = 76
122 codeL2 = 1
123 L2P = 1
124 begin_time = 86382
125 end_time = 172782
126 step = 1
127 # Создадим массив времени
128 t = np.arange(begin_time, end_time, step, int)
129 # Найдем координаты на интервале времени
130 moment1 = TakeCoordinate(t)
131 #moment1.coordinates()
132 moment1.XYZ = np.loadtxt("XYZ.txt")
133 moment1.WGS_XYZ = np.loadtxt("WGS_XYZ.txt")
134 # print(moment1.XYZ)
135 # Переведем в радианы координата приемника
136 N_gr = 44
137 N_min = 9
138 N_sec = 36.3261
139 N = N_gr * toRad + N_min * min2rad + N_sec * sec2rad
140
141 E_gr = 39
142 E_min = 00
143 E_sec = 13.0546
144 E = E_gr * toRad + E_min * min2rad + E_sec * sec2rad
145
146 H = 1.247
147
148 # найдем x
149 SKP = np.zeros((1, 3))

```

```

150     SKPF = np.zeros((1, 3))
151     # for i in range(86400):
152     #     Vrem = pymap3d.ecef2enu(moment1.WGS_XYZ[i, 1], moment1.
WGS_XYZ[i, 2], moment1.WGS_XYZ[i, 3], N, E, H, deg = False)
153     #     SKP[0, 0] = Vrem[0]
154     #     SKP[0, 1] = Vrem[1]
155     #     SKP[0, 2] = Vrem[2]
156     #     SKPF = np.vstack((SKPF, SKP))
157     #     print(i)
158     # np.savetxt("SKPF.txt", SKPF)
159     SKPF = np.loadtxt("SKPF.txt")
160     RFT = np.zeros((1, 3))
161     RFTF = np.zeros((1, 3))
162
163     for i in range(86400):
164         print(i)
165         if SKPF[i, 2] > 0:
166             RFT[0, 0] = math.sqrt(SKPF[i, 0] ** 2 + SKPF[i, 1] ** 2
+ SKPF[i, 2] ** 2)
167             RFT[0, 1] = math.acos(SKPF[i, 2] / RFT[0, 0])
168             if SKPF[i, 0] > 0:
169                 RFT[0, 2] = -math.atan(SKPF[i, 1] / SKPF[i, 0]) +
gps_pi / 2
170
171             elif ((SKPF[i, 0] < 0) and (SKPF[i, 1] > 0)):
172                 RFT[0, 2] = -math.atan(SKPF[i, 1] / SKPF[i, 0]) + 3
* gps_pi / 2
173
174             elif ((SKPF[i, 0] < 0) and (SKPF[i, 1] < 0)):
175                 RFT[0, 2] = -math.atan(SKPF[i, 1] / SKPF[i, 0]) -
gps_pi / 2
176         else:
177             RFT[0, 1] = np.nan
178             RFT[0, 0] = np.nan
179             RFT[0, 2] = np.nan
180             RFTF = np.vstack((RFTF, RFT))
181
182     plt.subplot(111, polar=True) # Полярная система координат
183
184

```

```

185 plt.plot(RFTF[0:86400, 2], RFTF[0:86400, 1]/toRad, lw=2)
186
187 plt.show()
188 Найдем# xyz приемника
189
190 X_RCV = 6400000 * math.cos(N)*math.cos(E)
191
192 Y_RCV = 6400000 * math.cos(N)*math.sin(E)
193
194 Z_RCV = 6400000*math.sin(N)
195
196
197 fig = plt.figure()
198 ax = fig.add_subplot(111, projection='3d')
199 m = 0
200 for i in range(85):
201     m += 1000
202     ax.scatter(moment1.WGS_XYZ[m, 1], moment1.WGS_XYZ[m, 2],
moment1.WGS_XYZ[m, 3], s=5)
203     print(X_RCV,Y_RCV,Z_RCV)
204     u = np.linspace(0, 2 * np.pi, 100)
205     v = np.linspace(0, np.pi, 100)
206
207     x = 6400000 * np.outer(np.cos(u), np.sin(v))
208     y = 6400000 * np.outer(np.sin(u), np.sin(v))
209     z = 6400000 * np.outer(np.ones(np.size(u)), np.cos(v))
210
211     ax.plot_surface(x, y, z, rstride=4, cstride=4, color='b')
212     ax.set_xlim3d(-3e7, 3e7)
213     ax.set_ylim3d(-3e7, 3e7)
214     ax.set_zlim3d(-3e7, 3e7)
215     plt.show()
216
217 fig = plt.figure()
218 ax = fig.add_subplot(111, projection='3d')
219 m = 0
220 for i in range(85):
221     m += 1000
222     ax.scatter(moment1.XYZ[m, 1], moment1.XYZ[m, 2], moment1.XYZ
[m, 3], s=5)

```

```

223
224     u = np.linspace(0, 2 * np.pi, 100)
225     v = np.linspace(0, np.pi, 100)
226
227     x = 6400000 * np.outer(np.cos(u), np.sin(v))
228     y = 6400000 * np.outer(np.sin(u), np.sin(v))
229     z = 6400000 * np.outer(np.ones(np.size(u)), np.cos(v))
230
231     ax.plot_surface(x, y, z, rstride=4, cstride=4, color='b')
232     ax.set_xlim3d(-3e7, 3e7)
233     ax.set_ylim3d(-3e7, 3e7)
234     ax.set_zlim3d(-3e7, 3e7)
235     plt.show()

```

2.3.2 Пункт 2

Построим трехмерные графики положения спутника на протяжении суток.

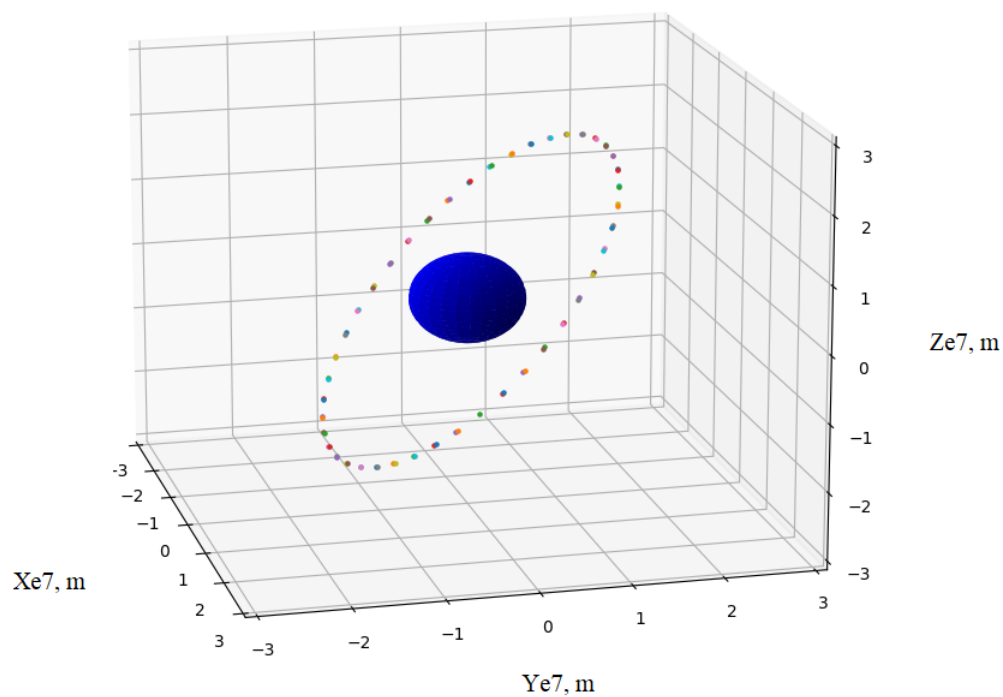


Рисунок 2.3.2 — Трехмерный график положения спутника GPS в ECEF WGS-84

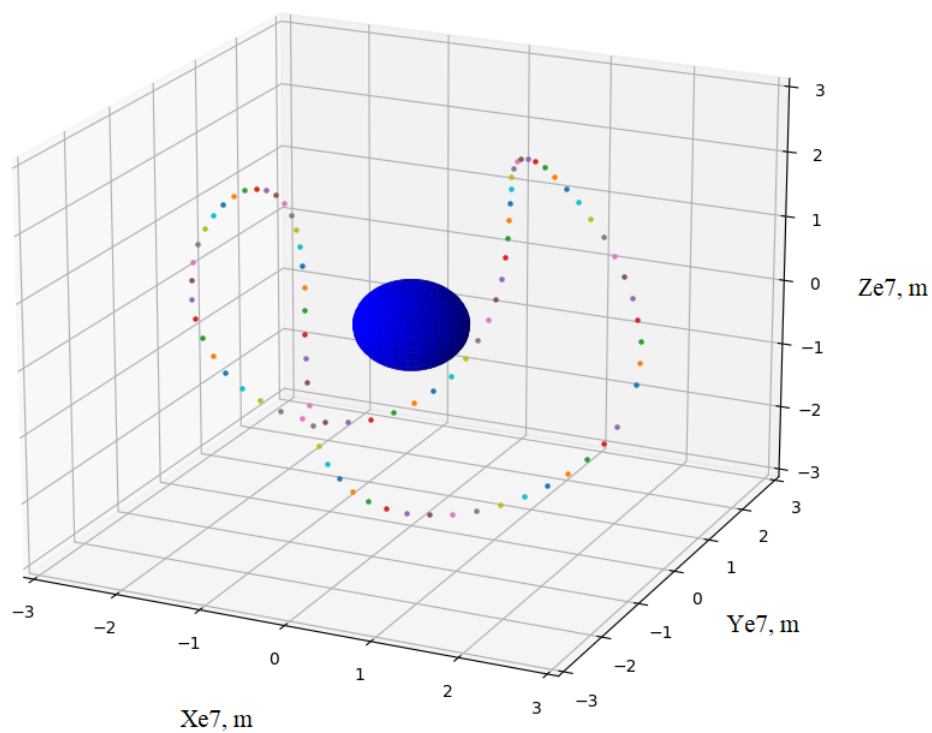


Рисунок 2.3.3 — Трехмерный график положения спутника GPS в инерциальной системе координат ECI

Переход из системы ECEF в систему ECI был осуществлен также согласно алгоритму из ИКД.

2.3.3 Пункт 3

Помимо траектории спутников в трехмерном виде получим эту траекторию в полярной системе координат(рис. 2.3.4) и сравним ее с результатом из Trimble GNSS Planning Online(рис. 2.3.5).

SkyView стоится относительно положения приемника из RTKNAVI.

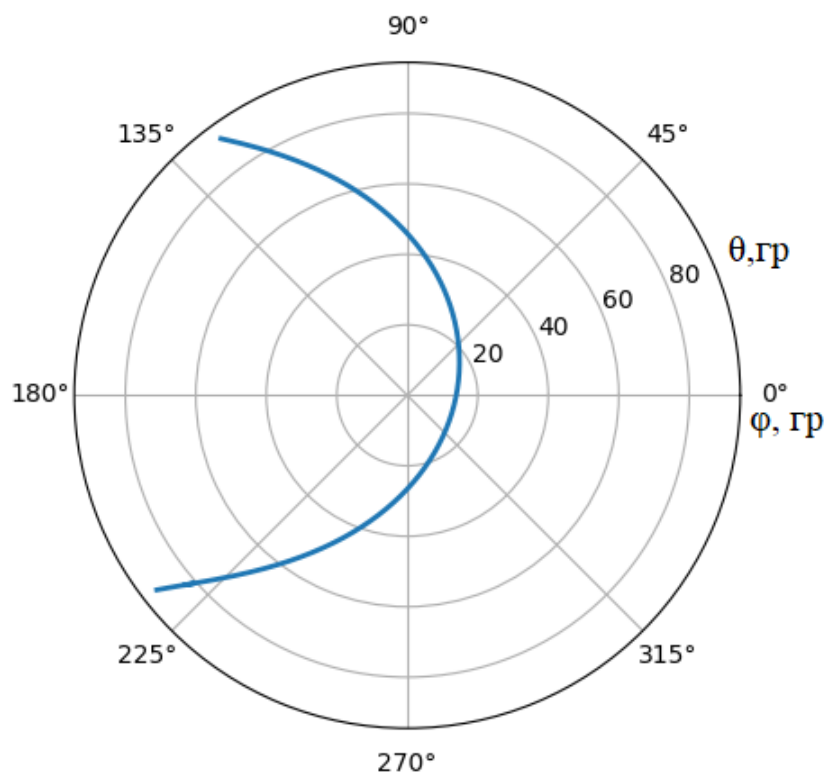


Рисунок 2.3.4 — Рассчитанный SkyPlot

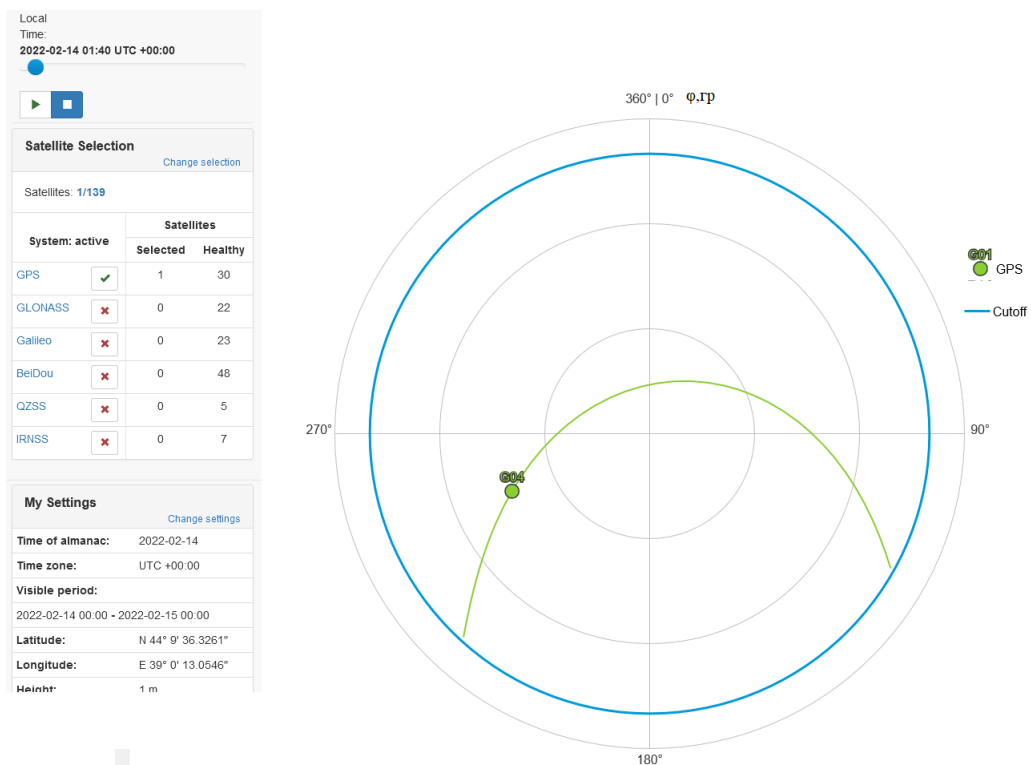


Рисунок 2.3.5 — Trimble SkyPlot

2.4 Выводы

На втором этапе по эфемеридам были рассчитаны положения спутника в разных системах координат, а также построен график SkyView.

Этап 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. 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:

- Код реализации
- Вывод тестов, включая анализ времени исполнения
- Вывод проверок на утечку памяти

- Вывод по этапу
- 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.

Работы по третьему этапу следует вести в директории libgpssvpos.

3.2 Основная часть

3.2.1 Теоретическая часть

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.

3.2.2 Решение задачи

Имеем входные данные:

Задача решается в несколько следующих этапов:

- Извлечь эфемериды;
- Заполнить массив положения спутника на протяжении суток;
- Сравнить с данными полученными на втором этапе;
- Найти время вычисления;
- Проверить программу на утечку данных;

3.2.3 Приложение 2

Листинг 3.1 — Программа 3 этапа

```
1 // Made on Earth by Murathon.
2
3
4 #include <iostream>
5 #include <fstream>
6 #include <windows.h>
7 #include <string>
8 #include <stdlib.h>
9 #include <cmath>
10 #include <stdio.h>
11 #include <bitset>
12
13 #define P2_5          pow(2,-5)
14 #define P2_55         pow(2,-55)
15 #define P2_43         pow(2,-43)
16 #define P2_31         pow(2,-31)
17 #define P2_4          pow(2,4)
18 #define P2_19         pow(2,-19)
19 #define P2_29         pow(2,-29)
20 #define P2_33         pow(2,-33)
21 #define SemiCirc      180
22
23
24 using namespace std;
25
26 struct Ephemeris {
27     double    Crs;
28     double    Dn;
29     double    M0;
30     double    Cuc;
31     double    e;
32     double    Cus;
33     double    sqrtA;
34     uint32_t  toe;
35     double    Cic;
36     double    Omega0;
```

```

37     double    Cis;
38     double    i0;
39     double    Crc;
40     double    omega;
41     double    OmegaDot;
42     double    iDot;
43     int16_t    Tgd;
44     uint32_t    toc;
45     double    af2;
46     double    af1;
47     double    af0;
48     uint16_t    WN;
49     uint16_t    IODC;
50     uint32_t    URA;
51     uint32_t    Health;
52     uint16_t    IODE2;
53     uint16_t    IODE3;
54     bool        codeL2;
55     bool        L2P;
56     uint32_t    slot;
57 };
58
59 struct nssrSF {
60     uint32_t    slot;
61     string    sf1;
62     string    sf2;
63     string    sf3;
64 };
65 struct XYZ {
66     double    xFin;
67     double    yFin;
68     double    zFin;
69 };
70
71 void parsStr(nssrSF *Rdata);
72 uint32_t pickSplitParam(string sf, uint16_t FS, int HMF, uint16_t
    FtS, int HMS);
73 void printEPH(Ephemeris* EPH);
74 void saveEPH(Ephemeris* EPH);
75 void calculateGps(Ephemeris* EPH, uint32_t k, XYZ *realPosition);

```

```

76 void decodeSF(Ephemeris* EPH, nssrSF *data);
77 int64_t pick32(string sf, int32_t FrmN, int HmR);
78
79
80 int main(void)
81 {
82     nssrSF data;
83     parsStr(&data);
84     cout << data.sf1 << endl << data.sf2 << endl << data.sf3 << endl
85     ;
86     Ephemeris *EPH = (Ephemeris*) calloc(1, sizeof(Ephemeris));
87     decodeSF(EPH,&data);
88     printEPH(EPH);
89     saveEPH(EPH);
90     uint32_t begin_time = 86382;
91     uint32_t end_time = 172782;
92     XYZ realPosition;
93     double** coord = new double*[3];
94     for (int i = 0; i < 3; i++) {
95         coord[i] = new double[end_time - begin_time];
96     }
97
98     double** coordPython = new double*[3];
99     for (int i = 0; i < 3; i++) {
100         coordPython[i] = new double[end_time - begin_time];
101     }
102     unsigned int start_time = clock();
103
104     for (uint32_t k = begin_time; k < end_time; k++) {
105
106         calculateGps(EPH, k, &realPosition);
107         coord[0][k-begin_time] = realPosition.xFin;
108         coord[1][k-begin_time] = realPosition.yFin;
109         coord[2][k-begin_time] = realPosition.zFin;
110
111     }
112     ifstream file("XYZ.txt");
113     double rub;
114     if (!file.is_open())

```



```

115     cout << "Файл" не может быть открыт!" << endl;
116 else {
117     for (int k = 0; k < end_time - begin_time; k++) {
118         file >> rub >> coordPython[0][k] >> coordPython[1][k] >>
coordPython[2][k];
119     }
120     file.close();
121 }
122
123 double delta_max = 0;
124 for (int i = 0; i < 3; i++) {
125     for (int k = 0; k < end_time - begin_time; k++) {
126
127         if (abs(coord[0][k] - coordPython[0][k]) > delta_max) {
128             delta_max = abs(coord[i][k] - coordPython[i][k]);
129
130         }
131
132     }
133
134 }
135 delete [] *coord;
136 delete [] coord;
137 delete [] * coordPython;
138 delete [] coordPython;
139 free(EPH);
140 unsigned int stop_time = clock();
141 unsigned int search_time = stop_time - start_time;
142
143
144
145     cout << endl << "Максимальная" разница координат м() = " << delta_max
<< endl;
146     cout << "Время" выполнения расчета с() = " << (double)search_time/
CLOCKS_PER_SEC << endl;
147
148
149
150 }
151 void calculateGps(Ephemeris* EPH, uint32_t k, XYZ *realPosition)

```

```

152 {
153     //cout << k << endl;
154     int32_t t_k = k - EPH->toe;
155
156     double Omega_e_Dot = 7.2921151467e-5;
157     double mu = 3.986005e14;
158
159     if (t_k > 302400) {
160         t_k -= 604800;
161     }
162     if (t_k < -302400) {
163         t_k += 604800;
164     }
165     double A = EPH->sqrtA*EPH->sqrtA;
166     double n_0 = sqrt(mu / pow(A, 3));
167     double n = n_0 + EPH->Dn;
168     double E_old = EPH->M0 + n * t_k;
169     double E_new = E_old + EPH->e * sin(E_old);
170     int i = 1;
171     while (abs(E_new - E_old) > 1e-8){
172         E_old = E_new;
173         E_new = E_old + EPH->e * sin(E_old);
174         i += 1;
175         if (i > 10){
176             break;
177         }
178     }
179
180     double E_k = E_new;
181     double cosNu_k = (cos(E_k) - EPH->e) / (1 - EPH->e * cos(E_k));
182     double sinNu_k = (sqrt(1 - pow(EPH->e,2)) * sin(E_k)) / (1 - EPH
->e * cos(E_k));
183
184     double Nu_k = atan2(sinNu_k, cosNu_k);
185     double Phi_k = Nu_k + EPH->omega;
186
187     double delta_u_k = EPH->Cus * sin(2 * Phi_k) + EPH->Cuc * cos(2
* Phi_k);
188     double delta_r_k = EPH->Crs * sin(2 * Phi_k) + EPH->Crc * cos(2
* Phi_k);

```

```

189     double delta_i_k = EPH->Cis * sin(2 * Phi_k) + EPH->Cic * cos(2
190     * Phi_k);
191
192     double u_k = Phi_k + delta_u_k;
193     double r_k = pow(EPH->sqrA,2) * (1 - EPH->e * cos(E_k)) +
194     delta_r_k;
195     double i_k = EPH->i0 + EPH->iDot * t_k + delta_i_k;
196
197     double x_k_shtrih = r_k * cos(u_k);
198     double y_k_shtrih = r_k * sin(u_k);
199
200     double Omega_k = EPH->Omega0 + (EPH->OmegaDot - Omega_e_Dot) *
201     t_k - Omega_e_Dot * (double)EPH->toe;
202
203     double x_k = x_k_shtrih * cos(Omega_k) - y_k_shtrih * cos(i_k) *
204     sin(Omega_k);
205     double y_k = x_k_shtrih * sin(Omega_k) + y_k_shtrih * cos(i_k) *
206     cos(Omega_k);
207     double z_k = y_k_shtrih * sin(i_k);
208
209     double X = x_k * cos(Omega_k) + y_k * sin(Omega_k);
210     double Y = -x_k * sin(Omega_k) + y_k * cos(Omega_k);
211     double Z = z_k;
212
213     realPosition->xFin = X;
214     realPosition->yFin = Y;
215     realPosition->zFin = Z;
216
217 }
218 void parsStr(nssrSF *Rdata)
219 {
220
221     SetConsoleOutputCP(CP_UTF8);
222     string path = "in.txt";
223
224     ifstream fin;
225     fin.open(path);
226
227     if(fin.is_open()) {

```

```

224
225     cout << "Файл " << "открыт." << endl;
226     while (!fin.eof()) {
227         int N = 3;
228         string rubber;
229         int nmbr_stlt;
230         uint32_t slot;
231         uint32_t subFrameNum;
232         string str;
233
234         uint32_t slot_SF1;
235         uint32_t slot_SF2;
236         uint32_t slot_SF3;
237
238         int mass[N];
239         for(int i=0;i<N;++i)
240         {
241             fin >> mass[i];
242         }
243         fin >> rubber >> rubber >> rubber;
244         fin >> nmbr_stlt >> slot >> rubber >> rubber >>
subFrameNum;
245         fin >> str;
246
247         if (nmbr_stlt == 4 and slot >= 604800/6){
248
249             if (subFrameNum == 1)
250             {
251                 slot_SF1 = slot;
252                 Rdata->sf1 = str;
253             }
254             else if (subFrameNum == 2)
255             {
256                 slot_SF2 = slot;
257                 Rdata->sf2 = str;
258
259             }
260             else if (subFrameNum == 3)
261             {
262                 slot_SF3 = slot;

```

```

263         Rdata->sf3 = str;
264     }
265
266     if (slot_SF1 + 1 == slot_SF2 and slot_SF2 + 1 ==
slot_SF3) {
267         Rdata->slot = slot_SF1;
268         return;
269     }
270
271 }
272 //
273 }
274 }
275 else
276 {
277     cout << "Ошибка" открытия файла!!!" << endl;
278 }
279 fin.close();
280
281
282 }
283 void saveEPH(Ephemeris* EPH)
284 {
285     ofstream fout;
286     string path = "out.txt";
287     fout.open(path);
288     if(fout.is_open()) {
289
290         cout << "Выгружаю" << endl;
291         fout << endl << "LNAV Ephemeris (slot = " << EPH->slot << ")
=" << endl;
292         fout << "\t\t Crs = " << EPH->Crs << endl;
293         fout << "\t\t Dn      = " << EPH->Dn << endl;
294         fout << "\t\t M0      = " << EPH->M0 << "\t\t[deg]" << endl;
295         fout << "\t\t Cuc = " << EPH->Cuc << endl;
296         fout << "\t\t e      = " << EPH->e << endl;
297         fout << "\t\t Cus = " << EPH->Cus << endl;
298         fout << "\t\t sqrtA = " << EPH->sqrtA << endl;
299         fout << "\t\t toe = " << EPH->toe << endl;
300         fout << "\t\t Cic = " << EPH->Cic << endl;

```

```

301      fout << "\t\t Omega0 = " << EPH->Omega0 << "\t\t[deg]" <<
endl;
302      fout << "\t\t Cis = " << EPH->Cis << endl;
303      fout << "\t\t i0 = " << EPH->i0 << "\t\t[deg]" << endl;
304      fout << "\t\t Crc = " << EPH->Crc << endl;
305      fout << "\t\t omega = " << EPH->omega << "\t\t[deg]" << endl
;
306      fout << "\t\t omeDot = " << EPH->OmegaDot << "\t\t[deg/s]"
<< endl;
307      fout << "\t\t iDot = " << EPH->iDot << "\t\t[deg/s]" <<
endl;
308      fout << "\t\t Tgd = " << EPH->Tgd << "\t\t\t[sec]" << endl;
309      fout << "\t\t toc = " << EPH->toc << endl;
310      fout << "\t\t af2 = " << EPH->af2 << endl;
311      fout << "\t\t af1 = " << EPH->af1 << endl;
312      fout << "\t\t af0 = " << EPH->af0 << endl;
313      fout << "\t\t WN = " << EPH->WN << endl;
314      fout << "\t\t IODC = " << EPH->IODC << endl;
315      fout << "\t\t URA = " << EPH->URA << endl;
316      fout << "\t\t Health = " << EPH->Health << endl;
317      fout << "\t\t IODE2 = " << EPH->IODE2 << endl;
318      fout << "\t\t IODE3 = " << EPH->IODE3 << endl;
319      fout << "\t\t codeL2 = " << EPH->codeL2 << endl;
320      fout << "\t\t L2P = " << EPH->L2P << endl;
321  }
322  else
323  {
324      cout << "Ошибка" открытия файла!!!" << endl;
325  }
326  fout.close();
327  cout << "Готово!" ;
328 }
329
330 void printEPH(Ephemeris* EPH)
331 {
332
333      cout << endl << "LNAV Ephemeris (slot = " << EPH->slot << ") ="
<< endl;
334      cout << "\t\t Crs = " << EPH->Crs << endl;
335      cout << "\t\t Dn = " << EPH->Dn << endl;

```

```

336 cout << "\\t\\t M0 = " << EPH->M0 << "\\t\\t[deg]" << endl;
337 cout << "\\t\\t Cuc = " << EPH->Cuc << endl;
338 cout << "\\t\\t e = " << EPH->e << endl;
339 cout << "\\t\\t Cus = " << EPH->Cus << endl;
340 cout << "\\t\\t sqrtA = " << EPH->sqrtA << endl;
341 cout << "\\t\\t toe = " << EPH->toe << endl;
342 cout << "\\t\\t Cic = " << EPH->Cic << endl;
343 cout << "\\t\\t Omega0 = " << EPH->Omega0 << "\\t\\t[deg]" << endl;
344 cout << "\\t\\t Cis = " << EPH->Cis << endl;
345 cout << "\\t\\t i0 = " << EPH->i0 << "\\t\\t[deg]" << endl;
346 cout << "\\t\\t Crc = " << EPH->Crc << endl;
347 cout << "\\t\\t omega = " << EPH->omega << "\\t\\t[deg]" << endl;
348 cout << "\\t\\t omeDot = " << EPH->OmegaDot << "\\t\\t[deg/s]" <<
endl;
349 cout << "\\t\\t iDot = " << EPH->iDot << "\\t\\t[deg/s]" << endl;
350 cout << "\\t\\t Tgd = " << EPH->Tgd << "\\t\\t\\t[sec]" << endl;
351 cout << "\\t\\t toc = " << EPH->toc << endl;
352 cout << "\\t\\t af2 = " << EPH->af2 << endl;
353 cout << "\\t\\t af1 = " << EPH->af1 << endl;
354 cout << "\\t\\t af0 = " << EPH->af0 << endl;
355 cout << "\\t\\t WN = " << EPH->WN << endl;
356 cout << "\\t\\t IODC = " << EPH->IODC << endl;
357 cout << "\\t\\t URA = " << EPH->URA << endl;
358 cout << "\\t\\t Health = " << EPH->Health << endl;
359 cout << "\\t\\t IODE2 = " << EPH->IODE2 << endl;
360 cout << "\\t\\t IODE3 = " << EPH->IODE3 << endl;
361 cout << "\\t\\t codeL2 = " << EPH->codeL2 << endl;
362 cout << "\\t\\t L2P = " << EPH->L2P << endl;
363
364 }
365 int64_t pick32(string sf, int32_t FrmN, int HmR) {
366     int64_t ans = 0;
367     int64_t Rans = 0;
368     for (int i = FrmN; i < FrmN+HmR; i++) {
369         ans = (ans | ((sf[i - 1] == '1') ? 1 : 0));
370         cout << sf[i-1];
371         if (i < FrmN+HmR-1){
372             ans = ans<<1;
373         }
374

```

```

375     }
376     return ans;
377
378
379 }
380
381 int64_t compl2int(uint64_t ans, int HmZ){
382     int64_t Rans = 0;
383     if (HmZ == 8){
384         if (bool((1<<7) & ans)){
385             ans |= 0xFFFFFFFFFFFFF00;
386             Rans = ~(ans-1);
387             /*cout<< endl << bitset<64>(Rans).to_string() << endl;*/
388             return -Rans;
389         }
390
391     }
392     if (HmZ == 14){
393         if (bool((1<<13) & ans)){
394             ans |= 0xFFFFFFFFFFC000;
395             Rans = ~(ans-1);
396             return -Rans;
397         }
398
399     }
400     if (HmZ == 16){
401         if (bool((1<<15) & ans)){
402             ans |= 0xFFFFFFFFFF0000;
403             Rans = ~(ans-1);
404             return -Rans;
405         }
406
407     }
408     if (HmZ == 22){
409         if (bool((1<<21) & ans)){
410             ans |= 0xFFFFFFFFFC0000;
411             Rans = ~(ans-1);
412             return -Rans;
413         }
414

```



```

415     }
416     if (HmZ == 24){
417         if (bool((1<<23) & ans)){
418             ans |= 0xFFFFFFFF000000;
419             Rans = ~(ans-1);
420             return -Rans;
421         }
422
423     }
424     if (HmZ == 32){
425         if (bool((1<<31) & ans)){
426             ans |= 0xFFFFFFFF00000000;
427             Rans = ~(ans-1);
428             return -Rans;
429         }
430
431     }
432     return ans;
433 }
434 uint32_t pickSplitParam(string sf, uint16_t FS, int HMF, uint16_t
435 FtS, int HMS) {
436     uint32_t ans = 0;
437
438     for (int i = FS; i < FS+HMF; i++) {
439         ans = (ans | ((sf[i-1] == '1')? 1 : 0)) << 1;
440     }
441     for (int i = FtS; i < FtS+HMS; i++) {
442         ans = ans | ((sf[i-1] == '1')? 1 : 0);
443         if (i < FtS+HMS-1){
444             ans = ans<<1;
445         }
446     }
447
448     return ans;
449 }
450
451 void decodeSF(Ephemeris* EPH, nssrSF *data){
452     double toRad = 3.1415926535898/180;
453

```

```

454  EPH->slot = data->slot;
455
456  EPH->Crs = compl2int(pick32(data->sf2,69,16),16)*P2_5;
457
458  EPH->Dn = compl2int(pick32(data->sf2,91,16),16)*P2_43*toRad ;
459
460  EPH->M0 = compl2int(pickSplitParam(data->sf2,107, 8, 121, 24)
,32)*P2_31*SemiCirc*toRad ;
461
462  EPH->Cuc = compl2int(pick32(data->sf2,151,16),16)*P2_29;
463
464  EPH->e = pickSplitParam(data->sf2,167, 8, 181, 24) * P2_33;
465
466  EPH->Cus = compl2int(pick32(data->sf2,211,16),16)*P2_29;
467
468  EPH->sqrtA = pickSplitParam(data->sf2,227, 8, 241, 24) * P2_19;
469
470  EPH->toe = pick32(data->sf2,271,16)*P2_4;
471
472  EPH->Cic = compl2int(pick32(data->sf3,61,16),16)*P2_29;
473
474  EPH->Omega0 = compl2int(pickSplitParam(data->sf3,77, 8, 91, 24)
,32)*P2_31*toRad ;
475
476  EPH->Cis = compl2int(pick32(data->sf3,121,16),16)*P2_29;
477
478  EPH->i0 = compl2int(pickSplitParam(data->sf3,137, 8, 151, 24)
,32)*P2_31*SemiCirc*toRad ;
479
480  EPH->Crc = compl2int(pick32(data->sf3,181,16),16)*P2_5;
481
482  EPH->omega = compl2int(pickSplitParam(data->sf3,197, 8, 211, 24)
,32)*P2_31*SemiCirc*toRad ;
483
484  EPH->OmegaDot = compl2int(pick32(data->sf3,241,24),24)*P2_43*
SemiCirc*toRad ;
485
486  EPH->iDot = compl2int(pick32(data->sf3,279,14),14)*P2_43*
SemiCirc*toRad ;
487

```

```

488   EPH->Tgd = compl2int(pick32(data->sf1,197,8),8)*P2_31;
489
490   EPH->toc = compl2int(pick32(data->sf1,219,16),16)*P2_4;
491
492   EPH->af2 = compl2int(pick32(data->sf1,241,8),8)*P2_55;
493
494   EPH->af1 = compl2int(pick32(data->sf1,249,16),16)*P2_43;
495
496   EPH->af0 = compl2int(pick32(data->sf1,271,22),22)*P2_31;
497
498   EPH->WN = pick32(data->sf1,61,10);
499
500   EPH->IODC = pickSplitParam(data->sf1,83, 2, 211, 8);
501
502   EPH->URA = pick32(data->sf1,73,4);
503
504   EPH->Health = EPH->IODE2 = pick32(data->sf1,73,6);
505
506   EPH->IODE2 = pick32(data->sf2,61,8);
507
508   EPH->IODE3 = pick32(data->sf3,271,8);
509
510   EPH->codeL2 = pick32(data->sf1,71,2);
511
512   EPH->L2P = data->sf1[90];
513 }

```

3.3 Выводы

В данном этапе была реализована на языке C/C++ функция расчета положения спутника Beidou на заданное время по шкале UTC. Функция сопровождается тестами решения уравнения Кеплера, проверкой на утечки памяти и профилированием. Погрешность вычисления координат функцией и моделью составляет порядка 3,35e-8м среди трех осей, при использовании вещественного типа данных двойной точности double. Время выполнение функции составляет 1.111 с. Потребляемый объем оперативной памяти не превышает 4 МБ. Утечек памяти не обнаружено.

Литература

- [1] Interface Control Contractor: SAIC (GPS SEI) 200 N. Pacific Coast Highway,
Suite 1800 El Segundo, CA 90245