

Национальный Исследовательский Университет
«МЭИ»
Институт радиотехники и электроники им. В.А. Котельникова
Кафедра РТС

Курсовая работа
По дисциплине
«Аппаратура потребителей СРНС»
«Разработка модуля расчета координат спутника Beidou»

Студент: Ряшенцева В.И.

Группа: Эр-15-16

Преподаватель: Корогодин И.В.

Москва, 2021 г.

Исходные данные

Спутник №19 системы Beidou.

Этап 1. Использование сторонних средств

Заданному в задании спутнику 19 соответствует спутник со следующим номером:

Таблица 1

Спутник	PRN	ID	SCN
BEIDOU 3-M1	C19	2017-069A	43001

При помощи сервиса Celestrak можно получить изображение формы орбиты и положение спутника на ней. Изображения по требованию в задании необходимо получить на период 18:00 МСК 16 февраля 2021.

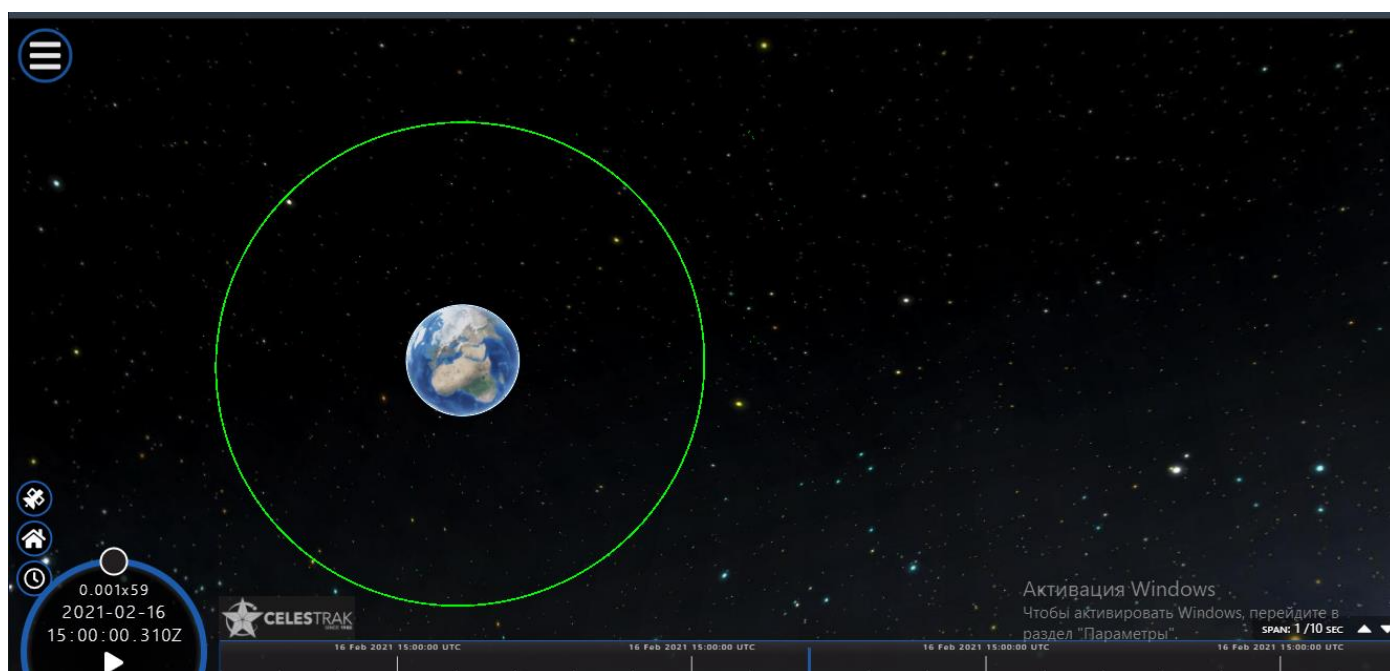


Рисунок 1 — Форма орбиты спутника

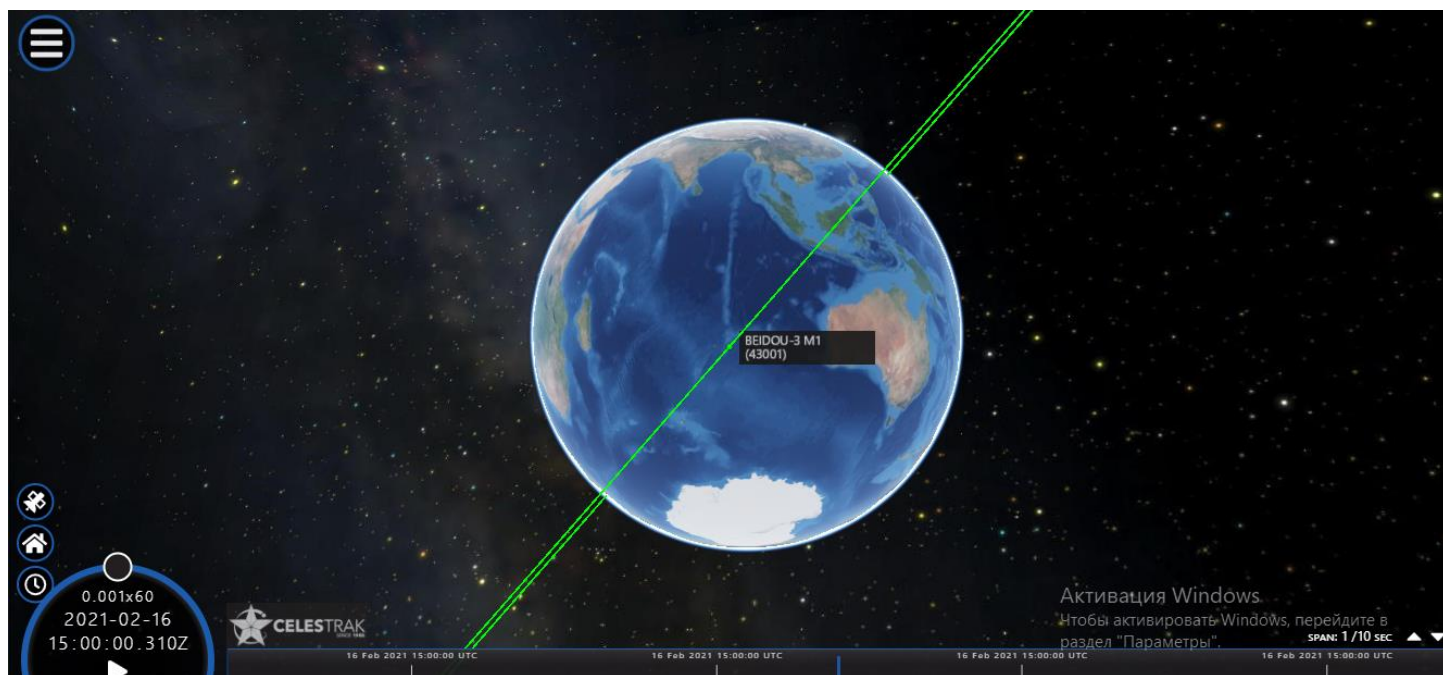


Рисунок 2 — Положение спутника на орбите

По данным Trimble GNSS Planning Online необходимо рассчитать диаграмму угла места и азимута спутника на интервал времени с 18:00 МСК 16 февраля до 06:00 МСК 17 февраля 2021 года. Расчет производится с учетом координат, соответствующих корпусу Е МЭИ.

Local
Time:
2021-02-16 18:00 UTC +03:00

Satellite Selection

Change selection

Satellites: 1/131

System: active	Satellites	
	Selected	Healthy
GPS <input type="checkbox"/>	0	32
GLONASS <input type="checkbox"/>	0	23
Galileo <input type="checkbox"/>	0	22
BeiDou <input checked="" type="checkbox"/>	1	49
QZSS <input type="checkbox"/>	0	4

My Settings

Change settings

Time of almanac:	2021-02-16
Time zone:	UTC +03:00
Visible period:	2021-02-16 18:00 - 2021-02-17 06:00
Latitude:	N 55° 45' 24.1346"
Longitude:	E 37° 42' 11.4473"
Height:	500 m
Elevation cutoff:	10 °

Рисунок 3 — Настройки Trimble GNSS Planning Online

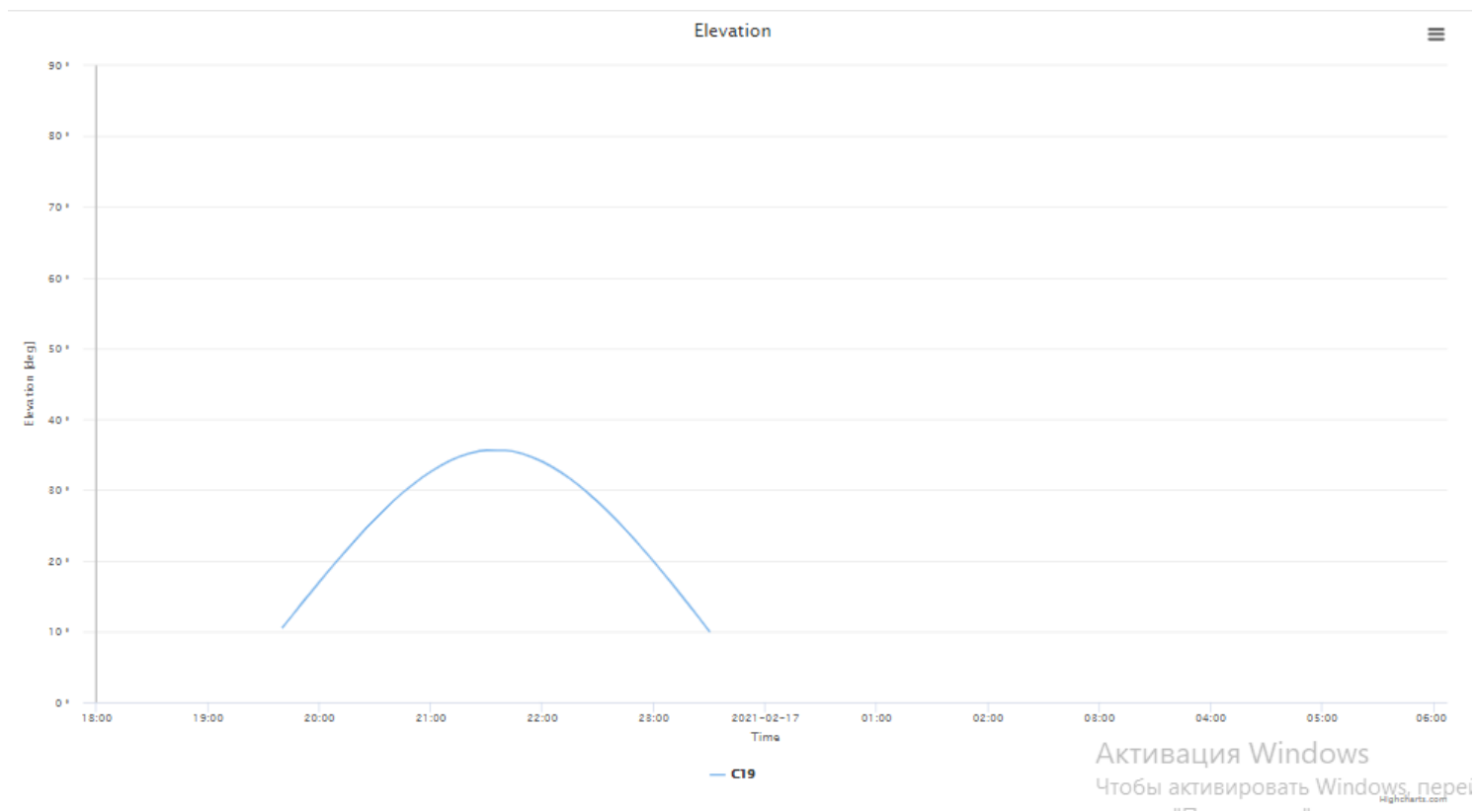


Рисунок 3 — Диаграмма угла места спутника

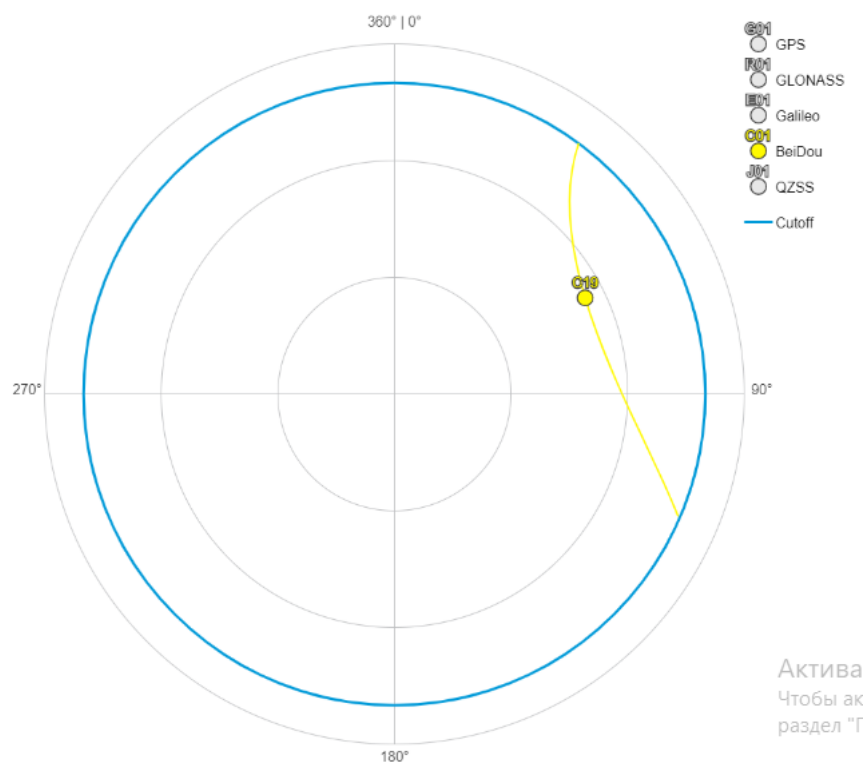


Рисунок 4 — Диаграмма угла азимута спутника

Ниже, на рисунке 5, приведены эфемериды спутника.

1	SatNum	19
2	toe (ms)	284400000.000
3	Crs (m)	-5.939062500000000000e+01
4	Dn (rad/ms)	4.05159716540537396e-12
5	M0 (rad)	2.09961820806318311e+00
6	Cuc (rad)	-2.80654057860374451e-06
7	e	8.22309288196265697e-04
8	Cus (rad)	5.74858859181404114e-06
9	sqrtA (sqrt (m))	5.28262378883361816e+03
10	Cic (rad)	6.42612576484680176e-08
11	Omega0 (rad)	-2.83355370614262747e-01
12	Cis (rad)	-7.59027898311614990e-08
13	i0 (rad)	9.65653770875602335e-01
14	Crc (m)	2.438437500000000000e+02
15	omega (rad)	-1.24505537862834337e+00
16	OmegaDot (rad/ms)	-7.01279211094012322e-12
17	iDot (rad/ms)	-2.57867884089573192e-13
18	Tgd (ms)	1.22000000000000000e+05
19	toc (ms)	2.84400000000000000e+08
20	af2 (ms/ms^2)	0.00000000000000000e+00
21	af1 (ms/ms)	1.37863054305853439e-11
22	af0 (ms)	7.30971693992614746e-01
23	URA	0
24	IODE	257
25	IODC	1
26	codeL2	0
27	L2P	0
28	WN	789

Рисунок 5 — Эфемериды спутника

Этап 2. Моделирование

На данном этапе работы необходимо реализовать на языке Matlab функцию расчета положения спутника № 19 СРНС Beidou на заданный момент по шкале времени UTC. В качестве эфемерид будут использоваться данные, полученные на предыдущем этапе.

Важно уточнить, что так как система Beidou унаследовала модель GPS, то расчеты будут производиться соответственно алгоритму из ИКД GPS [1].

Для начала необходимо разобраться с исходными данными. Как было сказано выше, эфемериды будут взяты с этапа 1 данной работы, но кроме этих данных, для расчета интервала прогноза необходимо учесть формат времени системы Beidou и рассчитать количество секунд от начала текущей недели.

Для этого запишем формат времени Beidou:

$$WN:SOW,$$

где WN — номер недели начиная с 1 января 2006 г., SOW — количество секунд от начала текущей недели.

Таким образом, необходимо рассчитать разницу между временами 16/02/2021 15:00:00 и 01/01/2006 00:00:00:

$$2021 - 2006 = 15 \text{ лет}$$

Из них 4 года является високосными (2008, 2012, 2016 и 2020) + 47 дней ($32 + 15 = 47$) + 15 часов 0 сек

– Временная разница в полных днях:

$$365 \cdot (15 - 4) + 366 \cdot 4 + 47 = 5526 \text{ дней}$$

– Временная разница в полных неделях:

$$\frac{5526}{7} = 789 \text{ недель}$$

– Количество дней после полных недель:

$$5526 - 789 \cdot 7 = 3 \text{ дня}$$

– Количество секунд от начала текущей недели:

$$3 \cdot 24 \cdot 3600 + 15 \cdot 3600 = 313200 \text{ сек}$$

– Количество секунд от начала текущей недели с учетом добавленных секунд [2]:

$$313200 + 4 = 313204 \text{ сек}$$

Ответ: 789:313204.

С учетом полученных данных и прогноза с 18:00 МСК 16 февраля до 06:00 МСК 17 февраля 2021 года (то есть временем прогноза, равным 43200 сек) построены трехмерные графики множества положений спутника Beidou в двух вариантах: в СК ECEF WGS84 (рисунок 6) и соответствующей ей инерциальной СК ECI (рисунок 7).

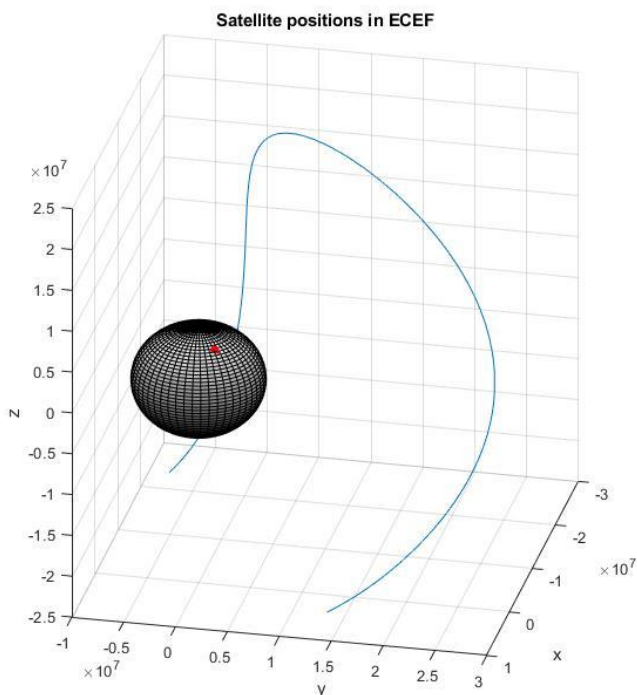


Рисунок 6 — Множество положений спутника Beidou в системе ECEF

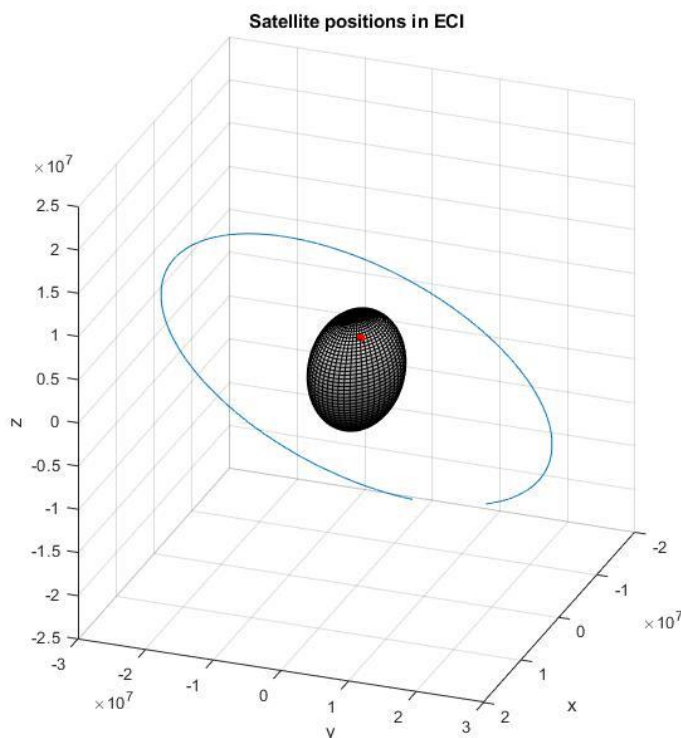


Рисунок 7 — Множество положений спутника Beidou в системе ECI

На рисунках 6 и 7 помимо траектории спутника так же изображена модель Земли с нанесенной на нее координатой антенны на крыше корпуса Е НИУ «МЭИ».

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

По полученным графикам видно, что за установленный интервал времени спутник не успевает полностью пройти всю свою траекторию. Экспериментально было установлено, что для этого необходимо задаться приблизительно в два раза большим интервалом.

Помимо траектории спутников в трехмерном виде было интересно получить эту траекторию в полярной системе координат и сравнить ее с результатом из Trimble GNSS Planning Online, изображенным на рисунке 4.

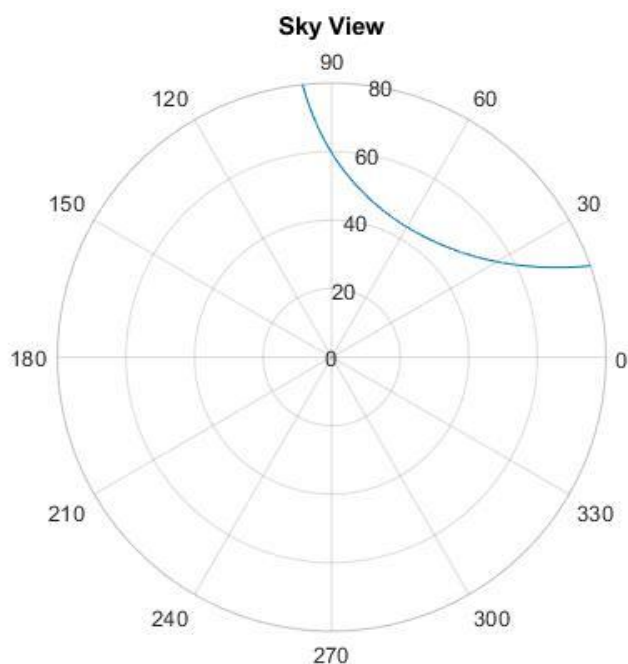


Рисунок 8 — Множество положений спутника Beidou в полярной системе координат

При сравнении результатов, изображенных на рисунках 4 и 8 важно учесть, что из-за особенностей алгоритмов Matlab, график в полярной системе координат строиться с установкой градусов против часовой стрелки и нулевым значением справа. Учитывая эти факторы видно, что результаты совпадают с небольшими расхождениями, которые могут быть связаны с тем, что при моделировании использовались одни эфемериды и то, что обрезка системы координат в Trimble GNSS Planning Online и Matlab могут отличаться.

Код программы в Matlab представлен в приложении.

Этап 3. Реализация

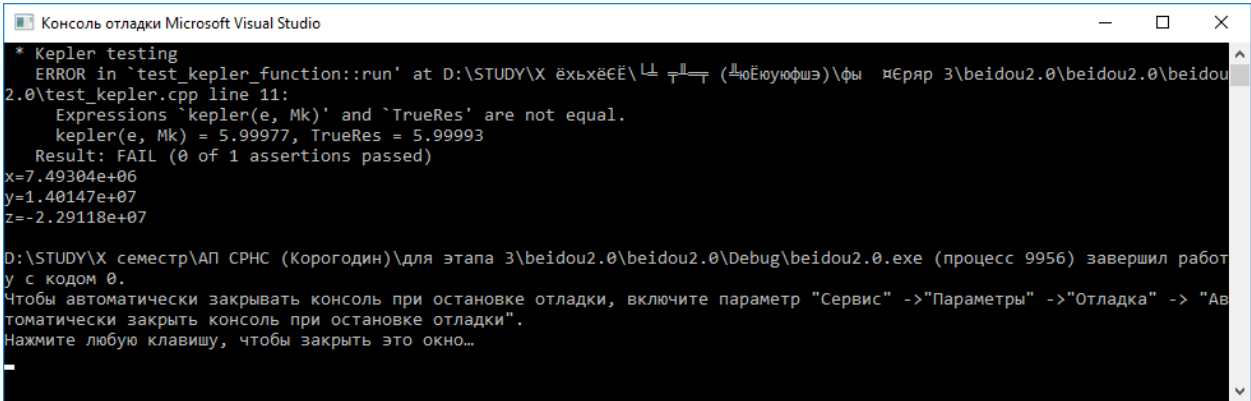
На данном этапе работы необходимо было разработать на языке C/C++ функцию расчета положения спутника Beidou на заданное время по шкале UTC, минимизируя время её исполнения и количество затрачиваемой оперативной памяти.

В качестве языка программирования был выбран язык C++, написание программы производилось в Visual studio 2019 (VS) (код в приложении).

Алгоритм расчета местоположения спутника включает в себя уравнение Кеплера. По условию задания необходимо было протестировать ее.

Решение уравнения Кеплера было выведено в отдельный программный модуль, для реализации теста решения данного уравнения была подключена OpenSource библиотека для Unit-тестирования Simple C++ testing library [3].

Для проверки корректной работы теста, были переприсвоены такие параметры как эксцентриситет и средняя аномалия (для получения заведомо ложных результатов). Результат работы теста показан на рисунке 9.



```
Консоль отладки Microsoft Visual Studio

* Kepler testing
ERROR in 'test_kepler_function::run' at D:\STUDY\X ёххёёё\11 ТТТ (ёюёюёфшэ)\фы ёёряр 3\beidou2.0\beidou2.0\beidou
2.0\test_kepler.cpp line 11:
  Expressions 'kepler(e, Mk)' and 'TrueRes' are not equal.
  kepler(e, Mk) = 5.99977, TrueRes = 5.99993
  Result: FAIL (0 of 1 assertions passed)
x=7.49304e+06
y=1.40147e+07
z=-2.29118e+07

D:\STUDY\X семестр\АП СРНС (Корогодин)\для этапа 3\beidou2.0\beidou2.0\Debug\beidou2.0.exe (процесс 9956) завершил работ
у с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Ав
томатически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

Рисунок 9 — Результат теста работы функции решения уравнения Кеплера

Далее необходимо было сравнить расчетного положения спутника в сравнении с Matlab с шагом 1 с. Данный этап был реализован путем сравнения массивов данными (результат работы кода, описанный в этапе 2 данного отчета) Matlab и C++. Для уменьшения количества открытых файлов в C++,

было принято решения сравнивать не координаты x , y , z , а дальность:

$$R = \sqrt{x^2 + y^2 + z^2}.$$

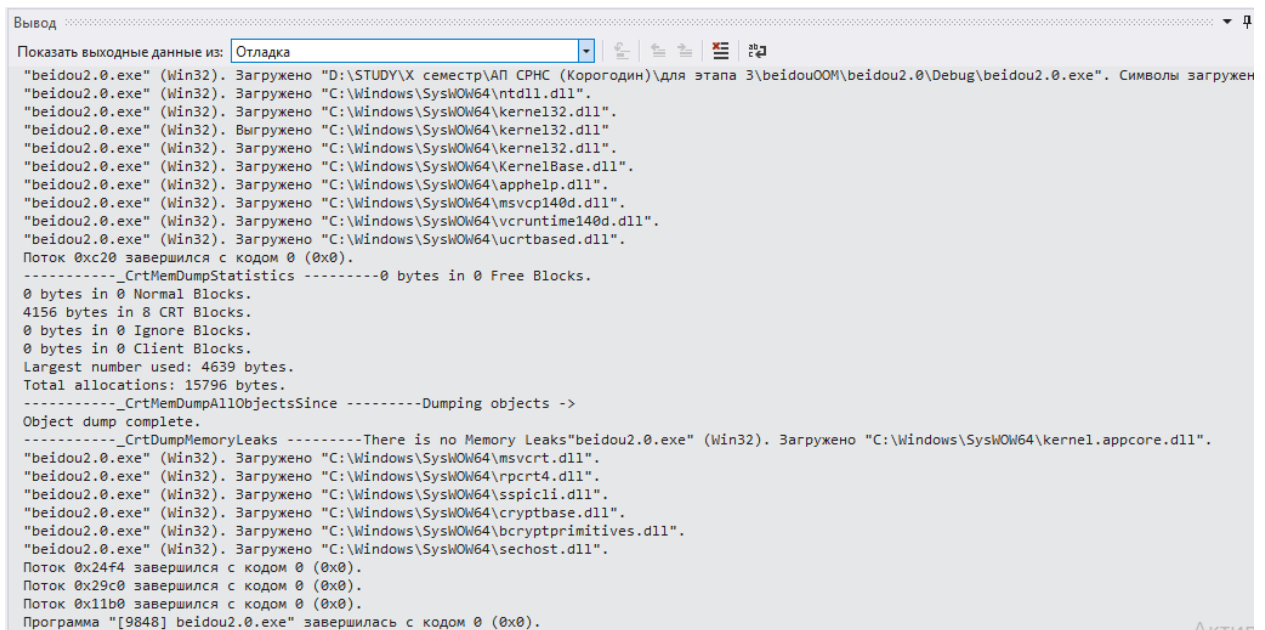
В ходе сравнения результирующих массивов Matlab и C++ была найден максимальное расхождение в значениях, которое составляет 3.16185 м.

Полученная разница обусловлена тем, что тип данных в Matlab и C++ имеют разную точность (количество байт, хранимых в памяти).

Для проверки наличия утечки памяти были использованы внутренние инструменты VS (подключение хедера `crtDBG.h`) в режиме debug.

С помощью функции `_CrtMemCheckpoint()` был сделан снэпшот утилизации памяти в момент запуска программы и после её выполнения. После чего оба снэпшета сравниваются посредством вызова функции `_CrtMemDifference()`.

Для того, чтобы убедиться, что вся память, выделенная приложением, была освобождена и не произошла утечка памяти с момента запуска программы, была вызвана функция `_CrtDumpMemoryLeaks()`. Данные по утечке памяти были выведены в окно отладки VS, как показаны на рисунке 10.



```
Вывод
Показать выходные данные из: Отладка
"beidou2.0.exe" (Win32). Загружено "D:\STUDY\X семестр\АП СРНС (Корогодин)\для этапа 3\beidou00M\beidou2.0\Debug\beidou2.0.exe". Символы загружены
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\ntdll.dll".
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\kernel32.dll".
"beidou2.0.exe" (Win32). Выгружено "C:\Windows\SysWOW64\kernel32.dll".
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\kernel32.dll".
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\KernelBase.dll".
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\apphelp.dll".
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\msvcrt140d.dll".
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\vcruntime140d.dll".
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\ucrtbased.dll".
Поток 0xc20 завершился с кодом 0 (0x0).
-----_CrtMemDumpStatistics -----0 bytes in 0 Free Blocks.
0 bytes in 0 Normal Blocks.
4156 bytes in 8 CRT Blocks.
0 bytes in 0 Ignore Blocks.
0 bytes in 0 Client Blocks.
Largest number used: 4639 bytes.
Total allocations: 15796 bytes.
-----_CrtMemDumpAllObjectsSince -----Dumping objects ->
Object dump complete.
-----_CrtDumpMemoryLeaks -----There is no Memory Leaks
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\kernel.appcore.dll".
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\msvcrt.dll".
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\RPCRT4.dll".
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\SSPICLI.dll".
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\CRYPTBASE.dll".
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\BCRYPTPRIMITIVES.dll".
"beidou2.0.exe" (Win32). Загружено "C:\Windows\SysWOW64\SECHOST.dll".
Поток 0x24f4 завершился с кодом 0 (0x0).
Поток 0x29c0 завершился с кодом 0 (0x0).
Поток 0x11b0 завершился с кодом 0 (0x0).
Программа "[9848] beidou2.0.exe" завершилась с кодом 0 (0x0).
```

Рисунок 10 — Окно debug; leak memory

Таким образом, в ходе выполнения данного этапа была написана программа на языке C++, что целесообразно при разработке встраиваемого ПО. Кроме того, в процессе разработки производилось тестирование уравнения Кеплера, что позволяет быстро обнаружить ошибку при не верных исходных данных. Также по заданию необходимо было произвести сравнение результатов работы программы, на писанных на языках Matlab и C++, в ходе которого было установлено, что данные языки имеют различия в типах данных (в памяти хранится разное количество байт). Более того, интересно было протестировать программу на наличие утечки памяти, так как при ее наличии может произойти переполнении всей оперативной памяти устройства и т.п. В ходе данного тестирования утечка памяти не была обнаружена.

Заключение

В ходе выполнения данного курсового проекта было произведено ознакомление и изучения ИКД Beidou [1].

При помощи сторонних средств была построена орбита заданного в задании спутника Beidou, получена траектория его движения на заданный промежуток времени.

При помощи алгоритма, предоставленного в ИКД, были изучены все этапы нахождения местоположения спутника. Далее этот алгоритм был реализован на языке Matlab.

Заключительным этапом выполнения курсового проекта является факт разработки функции расчета местоположения спутника Beidou на языке C++. В ходе выполнения данного этапа было произведено сравнение результатов работы программы на языках Matlab и C++; обнаружен факт различия размера хранимых байт в памяти. Разработанный алгоритм протестирован на верность полученных результатов и отсутствие утечки памяти.

Список источников

1. Interface Specification IS-GPS-200L, August 2020. [Электронный ресурс]. URL: <https://www.gps.gov/technical/icwg/IS-GPS-200L.pdf> (дата обращения: 03.04.2021).
2. Дополнительная секунда [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Дополнительная_секунда (дата обращения: 03.04.2021).
3. Библиотека для Unit-тестирования C++ [Электронный ресурс]. URL: <https://github.com/mitya57/cxxunit> (дата обращения: 05.05.2021).

Приложение

```
clc
close all
%% ephemeris
e = 8.22309288*10^(-4); % eccentricity
mu = 3.986005*10^14; % WGS 84 value of the earth's gravitational constant
Omega_e = 7.2921151467*10^(-5); % WGS 84 value of the earth's rotation rate
toe = 284400 + 4; % time of ephemeris
A = (5.28262379*10^3)^2; % semi-major axis ^2
n0 = sqrt(mu/A^3); % computed mean motion
i0 = 0.96565377; % inclination
Om01 = 0.28335371; % longitude of the node
omega = -1.24505538; % perigee argument
M0 = 2.09961821; % mean anomaly
Dn = 4.05159717*10^(-9); % mean motion difference from computed value
n = n0 + Dn; % corrected mean motion
OmegaDot = -7.0127921109*10^(-9); % rate of right ascension
IDOT = -2.57867884*10^(-10); % rate of inclination angle
Crs = -59.390625; % amplitude of the sine harmonic correction term to the
orbit radius
Cuc = -2.80654058*10^(-6); % amplitude of the cosine harmonic correction term
to the argument of latitude
Cus = 5.74858859*10^(-6); % amplitude of the sine harmonic correction term to
the argument of latitude
Cic = 6.42612576*10^(-8); % amplitude of the cosine harmonic correction term
to the angle of inclination
Cis = -7.59027898*10^(-8); % amplitude of the sine harmonic correction term
to the angle of inclination
Crc = 243.84375; % amplitude of the cosine harmonic correction term to the
orbit radius
%%
for i = 1:43200
t = 313200 + 4 + i; % TOW
tk = t - toe; % time from ephemeris reference epoch % if tk is greater than
302400 seconds, subtract 604800 sec from tk.
% if tk is less than -302400 sec, add 604800 seconds to tk
Mk = M0 + n*tk; % mean anomaly
%% Kepler's equation may be solved by iteration
E = zeros(1,4);
E(1,1) = Mk;
    for j = 2:4
        E(1,j) = E(1,j - 1) + (Mk - E(1,j - 1) + e*sin(E(1,j - 1)))/(1 -
e*cos(E(1,j - 1))); % 3 iterations
        E(1,j-1) = E(1,j);
    end
E_k = E(1,4);
%%
nu_k = atan2((sqrt(1 - e^2)*sin(E_k))/(1 - e*cos(E_k)), (cos(E_k) - e)/(1 -
e*cos(E_k))); % true anomaly
Phi_k = nu_k + omega; % argument of latitude

delta_u_k = Cus*sin(2*Phi_k) + Cuc*cos(2*Phi_k); % argument of latitude
correction
delta_r_k = Crs*sin(2*Phi_k) + Crc*cos(2*Phi_k); % radius correction
delta_i_k = Cis*sin(2*Phi_k) + Cic*cos(2*Phi_k); % inclination correction

u_k = Phi_k + delta_u_k; % corrected argument of latitude
r_k = A*(1 - e*cos(E_k)) + delta_r_k; % corrected radius
i_k = i0 + delta_i_k + IDOT*tk; %corrected inclination
```



```

%% positions in orbital
x_k = r_k*cos(u_k);
y_k = r_k*sin(u_k);
%%
Omega_k = Om01 + (OmegaDot - Omega_e)*tk - Omega_e*toe; % corrected longitude
of ascending node
%% earth-fixed
x_k1 = x_k*cos(Omega_k) - y_k*cos(i_k)*sin(Omega_k);
y_k1 = x_k*sin(Omega_k) + y_k*cos(i_k)*cos(Omega_k);
z_k1 = y_k*sin(i_k);

x1_coord(1,i) = x_k1;
y1_coord(1,i) = y_k1;
z1_coord(1,i) = z_k1;
%% transfer
tetta = Omega_e*tk;
%%
x_k2 = x_k1*cos(tetta) - y_k1*sin(tetta);
y_k2 = x_k1*sin(tetta) + y_k1*cos(tetta);
z_k2 = z_k1;

x2_coord(1,i) = x_k2;
y2_coord(1,i) = y_k2;
z2_coord(1,i) = z_k2;
%% porar coordinates
Earth_radius = 6378136;
H = 500;
a = Earth_radius;
B = deg2rad(55.45241346);
N = a/sqrt((1-e^2*(sin(B))^2));
L = deg2rad(37.42114473);
Coord_x = (N+H)*cos(B)*cos(L);
Coord_y = (N+H)*cos(B)*sin(L);
Coord_z = ((1-e^2)*N+H)*sin(B);
p = sqrt((x_k1-Coord_x).^2+(y_k1-Coord_y).^2+(z_k1-Coord_z).^2);

tetta1 = asin((z_k1-Coord_z)/p);
phil = atan2((x_k1-Coord_x), (y_k1-Coord_y));

tetta1_i(1,i) = -tetta1*180/pi + 90;
phil_i(1,i) = -phil;
end
%% plot
% Earth with coord
[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;

figure (1)
subplot (1,2,1)
surf(x_Earth,y_Earth,z_Earth); hold on;
colormap ('gray');
plot3(Coord_x, Coord_y, Coord_z, 'r.','MarkerSize', 20 );
plot3(x1_coord(1,:), y1_coord(1,:), z1_coord(1,:));
xlabel('x');
ylabel('y');
zlabel('z');
title('Satellite positions in ECEF')

figure (2)
subplot (1,2,1)

```

```

surf(x_Earth,y_Earth,z_Earth); hold on;
colormap ('gray');
plot3(Coord_x, Coord_y, Coord_z, 'r.','MarkerSize', 20 );
plot3(x2_coord(1,:), y2_coord(1,:), z2_coord(1,:));
xlabel('x');
ylabel('y');
zlabel('z');
title('Satellite positions in ECI')

figure(3)
grid on;
polarplot(phi1_i(1,:), tetta1_i(1,:));
rlim([0 80]);
title('Sky View')

```

beidou2.0.cpp

```
#include <iostream>
#include <cmath>
#include "windows.h"
#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>
using namespace std;

long double kepler(long double e, long double Mk);
long double testing(long double e, long double Mk);

int main()
{
    _CrtMemState sOld;
    _CrtMemState sNew;
    _CrtMemState sDiff;
    _CrtMemCheckpoint(&sOld); //take a snapshot

    long double e, mu, Omega_e, toe, A, n0, i0, Om01, omega, M0, Dn, n, OmegaDot,
    IDOT, Crs, Cuc, Cus, Cic, Cis, Crc;
    long double t, tk, Mk, E_k, nu_k, Phi_k, delta_u_k, delta_r_k, delta_i_k, u_k,
    r_k, i_k;
    long double x_k, y_k, Omega_k, x_k1, y_k1, z_k1;

    e = 8.22309288E-4;
    mu = 3.986005E+14;
    Omega_e = 7.2921151467E-5;
    toe = 284404;
    A = pow(5.28262379E+3, 2);
    n0 = pow(mu / (pow(A, 3)), 0.5);
    i0 = 0.96565377;
    Om01 = 0.28335371;
    omega = -1.24505538;
    M0 = 2.09961821;
    Dn = 4.05159717E-9;
    n = n0 + Dn;
    OmegaDot = -7.0127921109E-9;
    IDOT = -2.57867884E-10;
    Crs = -59.390625;
    Cuc = -2.80654058E-6;
    Cus = 5.74858859E-6;
    Cic = 6.42612576E-8;
    Cis = -7.59027898E-8;
    Crc = 243.84375;

    t = 313204; // TOW
    tk = t - toe; // time from ephemeris reference epoch
    Mk = M0 + n * tk; // mean anomaly
    // Kepler's equation may be solved by iteration
    testing(e, Mk);
    E_k = kepler(e, Mk);

    nu_k = atan2((sqrt(1 - pow(e, 2)) * sin(E_k)) / (1 - e * cos(E_k)), (cos(E_k) - e)
/ (1 - e * cos(E_k))); // true anomaly
    Phi_k = nu_k + omega; // argument of latitude

    delta_u_k = Cus * sin(2 * Phi_k) + Cuc * cos(2 * Phi_k); // argument of latitude
correction
    delta_r_k = Crs * sin(2 * Phi_k) + Crc * cos(2 * Phi_k); // radius correction
```

```

    delta_i_k = Cis * sin(2 * Phi_k) + Cic * cos(2 * Phi_k); // inclination correction

    u_k = Phi_k + delta_u_k; // corrected argument of latitude
    r_k = A * (1 - e * cos(E_k)) + delta_r_k; // corrected radius
    i_k = i0 + delta_i_k + IDOT * tk; // corrected inclination

    // positions in orbital
    x_k = r_k * cos(u_k);
    y_k = r_k * sin(u_k);

    Omega_k = Om01 + (OmegaDot - Omega_e) * tk - Omega_e * toe; // corrected longitude
of ascending node

    // earth-fixed
    x_k1 = x_k * cos(Omega_k) - y_k * cos(i_k) * sin(Omega_k);
    y_k1 = x_k * sin(Omega_k) + y_k * cos(i_k) * cos(Omega_k);
    z_k1 = y_k * sin(i_k);
    //cout << E_k << endl;
    cout << "x=" << x_k1 << endl;
    cout << "y=" << y_k1 << endl;
    cout << "z=" << z_k1 << endl;

    _CrtMemCheckpoint(&sNew); //take a snapshot
    _CrtMemDifference(&sDiff, &sOld, &sNew);
    OutputDebugString(L"-----_CrtMemDumpStatistics -----");
    _CrtMemDumpStatistics(&sDiff);
    OutputDebugString(L"-----_CrtMemDumpAllObjectsSince -----");
    _CrtMemDumpAllObjectsSince(&sOld);
    OutputDebugString(L"-----_CrtDumpMemoryLeaks -----");
    if (_CrtDumpMemoryLeaks() == false)
    {
        OutputDebugString(L"There is no Memory Leaks");
    }

    return 0;
}

```

kepler.cpp

```
#include <math.h>

long double kepler(long double e, long double Mk)
{
    //long double E_k;
    long double E[] = { 0, 0, 0, 0 };
    E[0] = Mk;
    for (int j = 1; j < 4; j++)
    {
        E[j] = E[j - 1] + (Mk - E[j - 1] + e * sin(E[j - 1])) / (1 - e * cos(E[j - 1])); // 3 iterations
        E[j - 1] = E[j];
    }

    return E[3];
}
```

test_kepler.cpp

```
#include "testing.h"

//#include "Unit.h"
long double kepler(long double e, long double Mk);

struct test_kepler_function : TestCase {
    void run(long double e, long double Mk) override {
        /*void test(double e, double Mk)
        {*/
            double TrueRes = kepler(8.22309288E-4, 6.00016);
            ASSERT_ALMOST_EQUAL(kepler(e, Mk), TrueRes);
        /*}
    }
};

REGISTER_TEST(test_kepler_function, "Kepler testing");

long double testing(long double e, long double Mk) {
    bool failfast = false;
    bool nocatch = false;

    int result = 0;
    for (const TestCaseInfo& info : storage) {
        std::cout << " * " << info.name << std::endl;
        TestCase* test_case = info.test_case;
        test_case->failfast = failfast;
        bool success = true;
        if (nocatch) {
            test_case->run(e, Mk);
        }
        else {
            try {
                test_case->run(e, Mk);
            }
            catch (const std::exception& e) {
                success = false;
                std::cerr << " " << E_ERROR("Exception occurred") << ": " << e.what()
<< std::endl;
            }
        }
        if (test_case->assertions_successful < test_case->assertions_total) {
            success = false;
        }
        if (success) {
            std::cout << "    Result: " << O_SUCCESS("SUCCESS")
<< " (" << test_case->assertions_total << " assertions passed)"
<< std::endl;
        }
        else {
            std::cout << "    Result: " << O_ERROR("FAIL")
<< " (" << test_case->assertions_successful << " of "
<< test_case->assertions_total << " assertions passed)"
<< std::endl;
            result = 1;
        }
        delete test_case;
    }
    return result;
}
```

testing.h

```
/* Simple C++ testing library.
 * Copyright: 2014-2017, Dmitry Shachnev.
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the
 * "Software"), to deal in the Software without restriction, including
 * without limitation the rights to use, copy, modify, merge, publish,
 * distribute, sublicense, and/or sell copies of the Software, and to
 * permit persons to whom the Software is furnished to do so, subject to
 * the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
 * in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
 * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
 * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
 * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

#ifndef CXXUNIT_TESTING_HPP
#define CXXUNIT_TESTING_HPP

#include <cfenv>
#include <cmath>
#include <cstdlib>
#include <cstring>
#include <algorithm>
#include <exception>
#include <iostream>
#include <string>
#include <vector>
#include "printing.h"
#ifdef __linux__
# include <signal>
# include <execinfo.h>
#endif

#ifdef _MSC_VER
# define __PRETTY_FUNCTION__ __FUNCTION__
#endif

#define REGISTER_TEST(cls, name) \
    static TestCaseProcessor _##cls({new cls(), name})

#define PRINT_ERROR() \
    std::cerr << " " << E_ERROR("ERROR") << " in `" << __PRETTY_FUNCTION__ << "' at " << \
    __FILE__ \
    << " line " << __LINE__ << ":" << std::endl;

#define ASSERT_TRUE(expression) \
    do { \
        auto _value = (expression); \
        if (!do_assert(_value)) { \
            PRINT_ERROR(); \
            std::cerr << " Expression `" #expression "' is not true." << std::endl; \
            std::cerr << " Value is " << repr(_value) << std::endl; \
            handle_failfast(); \
        } \
    }
```

```

    } \
} while (0)
#define ASSERT_FALSE(expression) \
do { \
    auto _value = (expression); \
    if (!do_assert(!_value)) { \
        PRINT_ERROR(); \
        std::cerr << "      Expression `" #expression "' is not false." << std::endl; \
        std::cerr << "      Value is " << repr(_value) << std::endl; \
        handle_failfast(); \
    } \
} while (0)
#define ASSERT_RELATION(e1, rel, e2) \
do { \
    auto _v1 = (e1); \
    auto _v2 = (e2); \
    if (!do_assert(_v1 rel _v2)) { \
        PRINT_ERROR(); \
        std::cerr << "      Expression `" #e1 " " #rel " " #e2 "' is not true." <<
std::endl; \
        std::cerr << "      " << #e1 " = " << repr(_v1) << ", " #e2 " = " << repr(_v2) <<
std::endl; \
        handle_failfast(); \
    } \
} while (0)
#define ASSERT_EQUAL(e1, e2) \
do { \
    auto _v1 = (e1); \
    auto _v2 = (e2); \
    if (!do_assert(_v1 == _v2)) { \
        PRINT_ERROR(); \
        std::cerr << "      Expressions `" #e1 "' and `" #e2 "' are not equal." <<
std::endl; \
        std::cerr << "      " << #e1 " = " << repr(_v1) << ", " #e2 " = " << repr(_v2) <<
std::endl; \
        handle_failfast(); \
    } \
} while (0)
#define ASSERT_ALMOST_EQUAL(e1, e2) \
do { \
    long double precision=0.00001; \
    auto _v1 = (e1); \
    auto _v2 = (e2); \
    if (!do_assert(std::fabs(_v1 - _v2) < precision)) { \
        PRINT_ERROR(); \
        std::cerr << "      Expressions `" #e1 "' and `" #e2 "' are not equal." <<
std::endl; \
        std::cerr << "      " << #e1 " = " << repr(_v1) << ", " #e2 " = " << repr(_v2) <<
std::endl; \
        handle_failfast(); \
    } \
} while (0)
#define ASSERT_STRINGS_EQUAL(e1, e2) \
do { \
    std::string _v1 = (e1); \
    std::string _v2 = (e2); \
    if (!do_assert(_v1 == _v2)) { \
        PRINT_ERROR(); \
        std::cerr << "      Strings `" #e1 "' (1) and `" #e2 "' (2) are not equal." <<
std::endl; \
        std::cerr << "      (1): '" << _v1 << "', " << std::endl; \
        std::cerr << "      (2): '" << _v2 << "' " << std::endl; \
        handle_failfast(); \
    } \
} \

```



```

    } while (0)
#define ASSERT_FLOATS_EQUAL(e1, e2) \
do { \
    auto _v1 = (e1); \
    auto _v2 = (e2); \
    if (!do_assert(compare_double(_v1, _v2))) { \
        PRINT_ERROR(); \
        std::cerr << "      Floating point numbers `" #e1 "` and `" #e2 "` are not equal." \
<< std::endl; \
        std::cerr << "      " << #e1 " = " << repr(_v1) << ", " #e2 " = " << repr(_v2) << \
std::endl; \
        handle_failfast(); \
    } \
} while (0)
#define ASSERT_THROWS(exception_class, expression) \
do { \
    bool caught = false; \
    try { \
        (void)(expression); \
    } catch (const exception_class &_exc) { \
        caught = true; \
        break; \
    } \
    if (!do_assert(caught)) { \
        PRINT_ERROR(); \
        std::cerr << "      Exception " << #exception_class << " not thrown." << std::endl; \
        handle_failfast(); \
    } \
} while(0)

template<typename T>
T repr(const T value) { return value; }

uint16_t repr(const uint8_t value) { return value; }
/*
bool compare_floats(float v1, float v2) {
    return (std::fabs(v1 - v2) * 100000.f <= std::min(fabs(v1), fabs(v2)));
}*/

bool compare_double(double v1, double v2) {
    return (std::fabs(v1 - v2) * 1000000000000000.f <= std::min(fabs(v1), fabs(v2)));
}

struct TestCase {
    unsigned assertions_total;
    unsigned assertions_successful;
    bool failfast;

    TestCase() :
        assertions_total(0),
        assertions_successful(0),
        failfast(false)
    {}

    bool do_assert(bool condition) {
        ++assertions_total;
        if (condition) {
            ++assertions_successful;
        }
        return condition;
    }

    void handle_failfast() {

```

```

        if (failfast) {
            std::cerr << "Exiting immediately because failfast = true." << std::endl;
            exit(1);
        }
    }

    virtual void run(long double e, long double Mk) = 0;
    virtual ~TestCase() {}
};

struct TestCaseInfo {
    TestCase* test_case;
    std::string name;
};

static std::vector<TestCaseInfo> storage;

struct TestCaseProcessor {
    TestCaseProcessor(TestCaseInfo info) {
        storage.push_back(info);
    }
};

#ifdef __linux__
void signal_handler(int signum) {
    std::cerr << E_ERROR("Signal occurred") << ": " << strsignal(signum) << std::endl;
    /* Print the backtrace */
    void* buffer[10];
    size_t size = backtrace(buffer, 10);
    backtrace_symbols_fd(buffer, size, STDERR_FILENO);
    /* Now call the default handler */
    signal(signum, SIG_DFL);
    exit(128 + signum);
}
#endif

void print_help(const char* command_name) {
    std::cout << "Usage: " << command_name << " [-f] [-n]" << std::endl;
    std::cout << std::endl;
    std::cout << " -f, --fail-fast: Exit after first failure" << std::endl;
    std::cout << " -n, --no-catch: Do not catch C++ exceptions (useful for debugging)"
<< std::endl;
}

/*int main(int argc, char** argv) {
    bool failfast = false;
    bool nocatch = false;

    for (int i = 1; i < argc; ++i) {
        char* arg = argv[i];
        if (!strcmp(arg, "--fail-fast") || !strcmp(arg, "-f")) {
            failfast = true;
        }
        else if (!strcmp(arg, "--no-catch") || !strcmp(arg, "-n")) {
            nocatch = true;
        }
        else {
            print_help(argv[0]);
            return 0;
        }
    }
}

#ifdef __USE_GNU
feenableexcept(FE_DIVBYZERO | FE_INVALID | FE_OVERFLOW | FE_UNDERFLOW);

```

```

#endif
#ifdef __linux__
    signal(SIGSEGV, signal_handler);
#endif

    int result = 0;
    for (const TestCaseInfo& info : storage) {
        std::cout << " * " << info.name << std::endl;
        TestCase* test_case = info.test_case;
        test_case->failfast = failfast;
        bool success = true;
        if (nocatch) {
            test_case->run();
        }
        else {
            try {
                test_case->run();
            }
            catch (const std::exception& e) {
                success = false;
                std::cerr << " " << E_ERROR("Exception occurred") << ": " << e.what()
<< std::endl;
            }
        }
        if (test_case->assertions_successful < test_case->assertions_total) {
            success = false;
        }
        if (success) {
            std::cout << "    Result: " << O_SUCCESS("SUCCESS")
                << " (" << test_case->assertions_total << " assertions passed)"
                << std::endl;
        }
        else {
            std::cout << "    Result: " << O_ERROR("FAIL")
                << " (" << test_case->assertions_successful << " of "
                << test_case->assertions_total << " assertions passed)"
                << std::endl;
            result = 1;
        }
        delete test_case;
    }
    return result;
}
*/
#endif /* CXXUNIT_TESTING_HPP */

```

time.cpp

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

using namespace std;

long double kepler(long double e, long double Mk);
//double testing(double e, double Mk);

int main()
{
    long double e, mu, Omega_e, toe, A, n0, i0, Om01, omega, M0, Dn, n, OmegaDot,
    IDOT, Crs, Cuc, Cus, Cic, Cis, Crc;
    long double tk, Mk, E_k, nu_k, Phi_k, delta_u_k, delta_r_k, delta_i_k, u_k, r_k,
    i_k, maxres;
    long double x_k, y_k, Omega_k, x_k1, y_k1, z_k1;

    int t;
    e = 8.22309288E-4;
    mu = 3.986005E+14;
    Omega_e = 7.2921151467E-5;
    toe = 284404;
    A = pow(5.28262379E+3, 2);
    n0 = pow(mu / (pow(A, 3)), 0.5);
    i0 = 0.96565377;
    Om01 = 0.28335371;
    omega = -1.24505538;
    M0 = 2.09961821;
    Dn = 4.05159717E-9;
    n = n0 + Dn;
    OmegaDot = -7.0127921109E-9;
    IDOT = -2.57867884E-10;
    Crs = -59.390625;
    Cuc = -2.80654058E-6;
    Cus = 5.74858859E-6;
    Cic = 6.42612576E-8;
    Cis = -7.59027898E-8;
    Crc = 243.84375;

    long double* x1_coord = new long double[43200];
    long double* y1_coord = new long double[43200];
    long double* z1_coord = new long double[43200];

    long double* R = new long double[43200];
    long double* Rmc = new long double[43200];
    long double* res = new long double[43200];

    for (int i = 0; i < 43200; i++)
    {
        t = 313204 + i; // TOW
        tk = t - toe; // time from ephemeris reference epoch
        Mk = M0 + n * tk; // mean anomaly
        // Kepler's equation may be solved by iteration
        //testing(e,Mk);
        E_k = kepler(e, Mk);
```

```

        nu_k = atan2((sqrt(1 - pow(e, 2)) * sin(E_k)) / (1 - e * cos(E_k)),
(cos(E_k) - e) / (1 - e * cos(E_k))); // true anomaly
        Phi_k = nu_k + omega; // argument of latitude

        delta_u_k = Cus * sin(2 * Phi_k) + Cuc * cos(2 * Phi_k); // argument of
latitude correction
        delta_r_k = Crs * sin(2 * Phi_k) + Crc * cos(2 * Phi_k); // radius
correction
        delta_i_k = Cis * sin(2 * Phi_k) + Cic * cos(2 * Phi_k); // inclination
correction

        u_k = Phi_k + delta_u_k; // corrected argument of latitude
        r_k = A * (1 - e * cos(E_k)) + delta_r_k; // corrected radius
        i_k = i0 + delta_i_k + IDOT * tk; // corrected inclination

        // positions in orbital
        x_k = r_k * cos(u_k);
        y_k = r_k * sin(u_k);

        Omega_k = Om01 + (OmegaDot - Omega_e) * tk - Omega_e * toe; // corrected
longitude of ascending node

        // earth-fixed
        x_k1 = x_k * cos(Omega_k) - y_k * cos(i_k) * sin(Omega_k);
        y_k1 = x_k * sin(Omega_k) + y_k * cos(i_k) * cos(Omega_k);
        z_k1 = y_k * sin(i_k);

        x1_coord[i] = x_k1;
        y1_coord[i] = y_k1;
        z1_coord[i] = z_k1;

        R[i] = sqrt(pow(x1_coord[i], 2) + pow(y1_coord[i], 2) + pow(z1_coord[i],
2));

        //cout << R[i] << endl;

    }
    /*
#pragma warning(disable : 4996)
    FILE* fin;
    fin = fopen("Data1.txt", "r");
    if (fin == NULL)
    {
        puts("Файл input.txt не найден!");
    }
    else
    {
        int k;
        double N;
        k = 0;
        while (fscanf(fin, "%d", &N) != EOF)
        {
            Rmc[k] = N;
            k++;
            cout << N << endl;
        }
    }
    */

    ifstream f;
    f.open("Data1.txt");
    //double N;
    maxres = 0;
    for (int i = 0; i < 43200; i++)

```

```

{
    f >> Rmc[i];
    //cout << Rmc[0]<<endl;
    res[i] = fabs(R[i] - Rmc[i]);
    //cout << res[i] << endl;
    if (maxres < res[i])
        maxres = res[i];
}

cout << maxres << endl;

delete[] R;
delete[] Rmc;
delete[] res;
delete[] x1_coord;
delete[] y1_coord;
delete[] z1_coord;
return 0;
}

```