

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт: ИРЭ

Кафедра: Радиотехнических систем

Специальность:

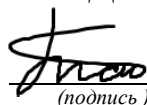
11.05.01 Радиоэлектронные системы и
комплексы

ОТЧЕТ по практике

**Наименование
практики:**

Производственная практика: научно-
исследовательская работа

СТУДЕНТ



(подпись)

/ Тасканов В.Е.

(Фамилия и инициалы)

Группа

ЭР-15-16

(номер учебной группы)

**ПРОМЕЖУТОЧНАЯ АТТЕСТАЦИЯ
ПО ПРАКТИКЕ**

ХОРОШО

(отлично, хорошо, удовлетворительно, неудовлетворительно,
зачтено, не зачтено)

/

Куликов Р.С.

/

(подпись)

(Фамилия и инициалы члена комиссии)

/

Шатилов А.Ю.

/

(подпись)

(Фамилия и инициалы члена комиссии)

**Москва
2021**

СПИСОК ИСПОЛНИТЕЛЕЙ

студент

Тасканов В.Е.

Содержание

ГЛАВА 1 ДОБАВЛЕНИЕ РАСЧЕТА КООРДИНАТ НС	5
1.1. Алгоритм расчета для ГНСС GPS	6
1.1.2. Алгоритм расчета координат	8
1.3. Алгоритм расчета для ГНСС ГЛОНАСС	10
1.3.2. Алгоритм расчета координат	12
1.4. Алгоритм расчета ионосферной погрешности	15
ГЛАВА 2 РЕАЛИЗАЦИЯ АЛГОРИТМОВ В ПРОГРАММЕ	18
2.1. Скачивание файла	18
2.2. Обработка файла	18
2.2.1. Обработка файла GPS	18
2.2.1. Обработка файла ГЛОНАСС	18
2.3. Расчет координат.....	19
2.4. Расчет времени	19
2.3. Изменение интерфейса программы.....	20
2.4. Необходимые файлы для сборки проекта	21
ГЛАВА 3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ	25
ЗАКЛЮЧЕНИЕ.....	31
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	31
ПРИЛОЖЕНИЕ	32

Введение

Спутниковые радионавигационные системы (СРНС) являются самыми точными системами по определению координат потребителя. Они стали важной частью в различных сферах нашей жизни. Наиболее распространенными являются системы ГЛОНАСС (Россия), GPS (США), Galileo (Евросоюз).

В 10 семестре стояла **цель работы** – откорректировать предыдущую программу, добавить функцию расчета координат НКА по данным альманахов систем ГЛОНАСС и GPS, добавить функцию расчета ионосферных погрешностей по углу возвышения спутника и текущему календарному времени года, и с учетом всех изменений произвести оценку координаты потребителя с учетом ошибок SISRE.

В рамках данной цели решаются следующие задачи:

1. Изменения предыдущего алгоритма определения координат НКА
2. Теоретическое изучение вносимых погрешностей от ионосферы
3. Добавления алгоритма расчета ионосферных погрешностей по углу возвышения спутника и текущему календарному времени года
4. Нахождения ошибки оценивания координаты потребителя с учетом ошибок SISRE и ионосферных погрешностей.

ГЛАВА 1 ДОБАВЛЕНИЕ РАСЧЕТА КООРДИНАТ НС

В данной работе изменим расчет координата НС, теперь они рассчитываются по данным альманахам соответствующих группировок для созвездий ГЛОНАСС и GPS.

Алгоритм расчета координат в программе будет следующий:

- Скачиваем файл с данными альманаха,
- Обработываем файл с помощью новой функции обработки,
- Рассчитываем координаты

Файл будет скачивать с сервера «информационно-аналитического центра координатно-временного и навигационного обеспечения, по адресу: «<ftp://ftp.glonass-iac.ru/MCC/ALMANAC/>», где далее следует выбор года и даты нужного альманаха. На сервере содержится два файла с разными расширениями – *agr* (для ГНСС GPS) и *agl* (для ГНСС ГЛОНАСС).

Скачав файл необходимо его оцифровать (перенести нужные данные в программу для реализации последующих алгоритмов).

Оцифровка файлов для GPS и ГЛОНАСС значительно отличается, поэтому разделим их на разные функции.

1.1. Алгоритм расчета для ГНСС GPS

В файле с расширением – agr, содержатся альманахи, записанные в виде строк:

```

07 01 2020 00871
1 0 2087 405504 09 01 2020 59904.000 -0.25749207E-03 -0.10913936E-10 -0.24811015E-08
-0.22440493E+00 0.31148988E+00 0.24107289E+00 0.92692375E-02 0.51535981E+04 0.40129888E+00
07 01 2020 00872
2 0 2087 405504 09 01 2020 59904.000 -0.38242340E-03 -0.72759576E-11 -0.25465852E-08
-0.24777687E+00 0.30480462E+00 -0.53421688E+00 0.19700527E-01 0.51535654E+04 0.49287593E+00
07 01 2020 00873
3 0 2087 405504 09 01 2020 59904.000 -0.68664551E-04 -0.72759576E-11 -0.24156179E-08
0.10692251E+00 0.30714494E+00 0.24356914E+00 0.26187897E-02 0.51535933E+04 0.32618165E-01
07 01 2020 00874
4 1 2087 405504 09 01 2020 59904.000 -0.31471252E-04 -0.36379788E-11 -0.25320332E-08
0.45012069E+00 0.30539781E+00 -0.86515725E+00 0.45394897E-03 0.51536631E+04 0.80713654E+00
07 01 2020 00875
5 0 2087 405504 09 01 2020 59904.000 -0.57220459E-05 0.00000000E+00 -0.24483597E-08
0.97382665E-01 0.30297548E+00 0.25501275E+00 0.57921410E-02 0.51535308E+04 -0.68326879E+00
07 01 2020 00876
6 0 2087 405504 09 01 2020 59904.000 -0.17738342E-03 -0.10913936E-10 -0.24847395E-08
-0.22703457E+00 0.31136209E+00 -0.37527728E+00 0.17843246E-02 0.51536895E+04 0.50334632E+00
07 01 2020 00877
7 0 2087 405504 09 01 2020 59904.000 -0.18882751E-03 -0.72759576E-11 -0.24629117E-08
0.77744925E+00 0.30363160E+00 -0.77044582E+00 0.13245106E-01 0.51536509E+04 0.17412102E+00
07 01 2020 00878
8 0 2087 405504 09 01 2020 59904.000 -0.20027161E-04 0.00000000E+00 -0.26266207E-08
-0.56470180E+00 0.30871850E+00 -0.68337798E-01 0.49986839E-02 0.51535586E+04 -0.83613443E+00
07 01 2020 00879
9 0 2087 405504 09 01 2020 59904.000 -0.12874603E-03 -0.72759576E-11 -0.25574991E-08
0.43520999E+00 0.30307466E+00 0.54175544E+00 0.17147064E-02 0.51534614E+04 -0.74238789E+00
07 01 2020 00880
10 0 2087 405504 09 01 2020 59904.000 -0.20313263E-03 -0.14551915E-10 -0.23974280E-08
0.10593116E+00 0.30713922E+00 -0.86190462E+00 0.52723885E-02 0.51536206E+04 -0.24406230E+00
07 01 2020 00881
  
```

Рисунок 1 – Пример скаченного файла с расширением agr

Где, строка 1, соответствует:

1	Число получения альманаха
2	месяц получения альманаха
3	год получения альманаха
4	время получения альманаха от начала суток, с UTC

Строка 2

1	номер PRN
---	-----------

2	обобщенный признак здоровья (0 - здоров)
3	неделя GPS (альманаха) (номер недели полный)
4	время недели GPS, с (альманаха) (количество секунд от начала недели)
5	число
6	месяц
7	год
8	время альманаха, с
9	поправка времени КА GPS относительно системного времени, с,
10	скорость поправки времени КА GPS относительно системного времени, с/с
11	Ω_0 - скорость долготы узла, полуциклы/с, $\dot{\Delta\Omega}^{****}$

Строка 3

1	Ω_0 - долгота узла, полуциклы, Ω_{0-n}
2	i - наклонение, полуциклы, i_{0-n}
3	w - аргумент перигея, полуциклы, ω_n
4	E – эксцентриситет, e_n

5	SQRT(A) - корень из большой полуоси, $m \cdot 0.5$, $\sqrt{A_0}$
6	M_0 - средняя аномалия, полуциклы, M_{0-n}

1.1.2. Алгоритм расчета координат

Далее полученные значения подставляются в алгоритм расчета координат, который возьмем из ИКД GPS:

- 1.1.2.1. Определим время, отсчитываемое от опорной эпохи эфемерид:

$$t_k = t - t_{oc}$$

- 1.1.2.2. Определим среднее движение:

$$n_0 = \sqrt{\frac{\mu}{A_0^3}}$$

- 1.1.2.3. Определим скорректированное среднее движение:

$$n_A = n_0 + \Delta n$$

- 1.1.2.4. Определим среднюю аномалию:

$$M_k = M_0 + n_A \cdot t_k$$

- 1.1.2.5. Решим уравнение Кеплера минимум 3-мя итерациями и определим E_k :

$$M_k = E_k - e_n \cdot \sin(E_k) \Rightarrow E_k = M_k + e_n \cdot \sin(E_k)$$

- 1.1.2.6. Определим истинную аномалию:

$$v_k = \arctg \left(\frac{\sqrt{1-e_n^2} \sin(E_k)}{(\cos(E_k) - e_n)} \right)$$

- 1.1.2.7. Определим скорректированный радиус орбиты спутника:

$$A_k = A_0 + \left(\dot{A} \right) t_k$$

$$r_k = A_k (1 - e_n \cos(E_k)) + \cancel{\delta r_k}$$

1.1.2.8. Определим аргумент широты:

$$\Phi_k = v_k + \omega$$

$$u_k = \Phi_k + \cancel{\delta u_k}$$

1.1.2.9. Определим координаты НС в орбитальной плоскости:

$$\begin{cases} x'_k = r_k \cdot \cos(u_k) \\ y'_k = r_k \cdot \sin(u_k) \end{cases}$$

1.1.2.10. Определим скорректированную долготу восходящего узла Ω_k определяется из соотношения:

$$\dot{\Omega} = \dot{\Omega}_{REF} + \cancel{\Delta \dot{\Omega}}$$

$$\Omega_k = \Omega_{0-n} + \left(\dot{\Omega} - \dot{\Omega}_e \right) t_{oe}$$

1.1.2.11. Определим скорректированное наклонение орбиты спутника

$$i_k = i_{0-n} + (\cancel{i_{0-n} - DOT}) t_k + \cancel{\delta i_k}$$

1.1.2.12. Определим координаты НС в геоцентрической системе координат:

$$\begin{cases} x_k = x'_k \cos \Omega_k - y'_k \cos i_k \sin \Omega_k \\ y_k = x'_k \sin \Omega_k + y'_k \cos i_k \cos \Omega_k \\ z_k = y'_k \sin i_k \end{cases}$$

1.3. Алгоритм расчета для ГНСС ГЛОНАСС

В файле с расширением – agl, содержатся альманахи, записанные в виде строк:

```

MSCI_200106 – Блокнот
Файл  Правка  Формат  Вид  Справка
01 01 2020 75600
1 1 1 31 12 2019 0.517487500E+04 0.000000000E+00 0.000000000E+00 -0.534057617E-04
0.4236565E+00 0.6714821E-02 0.1851501E+00 0.3643036E-03 -0.2656244E+04 0.8544922E-03
01 01 2020 75601
2 -4 1 31 12 2019 0.111991563E+05 0.000000000E+00 0.000000000E+00 -0.404357910E-03
0.2906666E+00 0.9313583E-02 -0.7553711E+00 0.1609802E-02 -0.2656010E+04 0.1037598E-02
01 01 2020 75602
3 5 1 31 12 2019 0.162281563E+05 0.000000000E+00 0.000000000E+00 -0.381469727E-05
0.1726351E+00 0.8380890E-02 -0.7248840E+00 0.1530647E-02 -0.2656043E+04 0.1037598E-02
06 01 2020 1803
4 6 1 05 01 2020 0.351999375E+05 0.000000000E+00 0.000000000E+00 -0.343322754E-04
-0.2965450E+00 0.1013565E-01 -0.5179749E+00 0.6799698E-03 -0.2656115E+04 0.1708984E-02
01 01 2020 75604
5 1 1 31 12 2019 0.262059375E+05 0.000000000E+00 0.000000000E+00 -0.419616699E-04
-0.5873966E-01 0.1023102E-01 -0.9271851E+00 0.4472733E-03 -0.2656137E+04 0.1098633E-02
01 01 2020 75605
6 -4 1 31 12 2019 0.317489688E+05 0.000000000E+00 0.000000000E+00 -0.167846680E-03
-0.1937952E+00 0.6618500E-02 0.7283020E+00 0.7905960E-03 -0.2656008E+04 0.1220703E-02
01 01 2020 75606
7 5 1 31 12 2019 0.366605000E+05 0.000000000E+00 0.000000000E+00 0.305175781E-04
-0.3013716E+00 0.8426666E-02 -0.7244873E+00 0.1291275E-02 -0.2656018E+04 0.1159668E-02
01 01 2020 75607
8 6 1 31 12 2019 0.920250000E+03 0.000000000E+00 0.000000000E+00 -0.114440918E-04
0.5282898E+00 0.8465767E-02 -0.5527344E+00 0.1614571E-02 -0.2656029E+04 0.9155273E-03
06 01 2020 1808
9 -2 1 05 01 2020 0.193098437E+05 0.000000000E+00 0.000000000E+00 -0.106811523E-03
0.7387104E+00 0.5420685E-02 -0.8858643E+00 0.1379013E-02 -0.2655211E+04 -0.1892090E-02
01 01 2020 75609
10 -5 1 31 12 2019 0.119666250E+05 0.000000000E+00 0.000000000E+00 0.534057617E-04
0.9348469E+00 0.9264946E-02 -0.4678345E-01 0.1546860E-02 -0.2689875E+04 -0.6103516E-04
01 01 2020 75610

```

Рисунок 2 – Пример скаченного файла с расширением agl

Где, строка 1, соответствует:

1	Число получения альманаха
2	месяц получения альманаха
3	год получения альманаха
4	время получения альманаха от начала суток, с UTC

Строка 2

1	номер КА в группировке
2	номер частотного слота (-7 - 24)

3	признак здоровья по альманаху (0 - 1)
4	число
5	месяц
6	год
7	время прохождения первого узла, на которое все дано, с
8	поправка ГЛОНАСС-UTC, с
9	поправка GPS-ГЛОНАСС, с
10	поправка времени КА ГЛОНАСС относительно системного времени, с

Строка 3

1	Lam - долгота узла, полуциклы
2	dI - коррекция наклона, полуциклы
3	w - аргумент перигея, полуциклы
4	E - эксцентриситет
5	dT - поправка к драконическому периоду, с
6	dTT - поправка к драконическому периоду, с/виток

1.3.2. Алгоритм расчета координат

Далее полученные значения подставляются в алгоритм расчета координат, который возьмем из ИКД ГЛОНАСС:

1.3.2.1. Определяется интервал прогноза в секундах:

$$\Delta t_{\text{пр}} = \Delta N_A \cdot 86400 + (t_i - t_{\lambda_A}),$$

$$\Delta N_A = \begin{cases} N - N_A - \left\langle \left\langle \frac{N - N_A}{1461} \right\rangle \right\rangle \cdot 1461 & \text{если } N_4 \neq 27, \\ N - N_A - \left\langle \left\langle \frac{N - N_A}{1460} \right\rangle \right\rangle \cdot 1460 & \text{если } N_4 = 27; \end{cases}$$

Где:

N – календарный номер суток внутри четырехлетнего периода, начиная с високосного года, на которых находится заданный момент времени t_i в секундах по шкале МДВ;

N_A – календарный номер суток по шкале МДВ внутри четырехлетнего интервала, передаваемый НКА в составе неоперативной информации;

$\langle \langle x \rangle \rangle$ – вычисление целого, ближайшего к x .

1.3.2.2. Рассчитывается количество целых витков W на интервале прогноза:

$$W = \left\langle \frac{\Delta t_{\text{пр}}}{T_{\text{ср}} + \Delta T_A} \right\rangle,$$

где $\langle x \rangle$ выделение целой части x ;

1.3.2.3. Определяется текущее наклонение:

$$i = \left(\frac{i_{\text{ф}}}{180^\circ} + \Delta i_A \right) \cdot \pi \text{ рад},$$

1.3.2.4. Определяются средний драконический период на витке W+1 и среднее движение:

$$T_{\text{др}} = T_{\text{ср}} + \Delta T_A + (2W + 1) \cdot \Delta \dot{T}_A,$$

$$n = 2\pi / T_{\text{др}},$$

1.3.2.5. Методом последовательных приближений $m = 0, 1, 2 \dots$ рассчитывается большая полуось орбиты a :

$$a^{(m+1)} = \sqrt[3]{\left(\frac{T_{\text{оск}}^{(m)}}{2\pi}\right)^2 \cdot GM};$$

$$p^{(m+1)} = a^{(m+1)}(1 - (\varepsilon_A)^2);$$

$$T_{\text{оск}}^{(m+1)} = \frac{T_{\text{др}}}{1 - \frac{3}{2} \cdot J_2^0 \left(\frac{a_e}{p^{(m+1)}}\right)^2 \left[\left(2 - \frac{5}{2} \cdot \sin^2 i\right) \cdot \frac{(1 - (\varepsilon_A)^2)^{3/2}}{(1 + \varepsilon_A \cdot \cos(\omega_A \pi))^2} + \frac{(1 + \varepsilon_A \cdot \cos(\omega_A \pi))^3}{1 - (\varepsilon_A)^2} \right]},$$

1.3.2.6. Определяются текущие значения долготы восходящего узла орбиты и аргумента перигея с учетом их векового движения под влиянием сжатия Земли:

$$\lambda = \lambda_A \cdot \pi - \left\{ \omega_3 + \frac{3}{2} J_2^0 \cdot n \cdot \left(\frac{a_e}{p}\right)^2 \cos i \right\} \Delta t_{\text{пр}};$$

$$\omega = \omega_A \cdot \pi - \frac{3}{4} J_2^0 n \left(\frac{a_e}{p}\right)^2 (1 - 5 \cos^2 i) \cdot \Delta t_{\text{пр}},$$

1.3.2.7. Рассчитывается значение средней долготы на момент прохождения текущего восходящего узла:

$$L_1 = \omega + E_0 - \varepsilon_A \sin E_0,$$

$$E_0 = -2 \cdot a \tan \left(\sqrt{\frac{1 - \varepsilon_A}{1 + \varepsilon_A}} \cdot \tan \frac{\omega}{2} \right).$$

Где

1.3.2.8. Определяется текущее значение средней долготы НКА:

$$L = L_1 + n(\Delta t_{\text{np}} - (T_{\text{cp}} + \Delta T_A)W - \Delta \dot{T}_A W^2).$$

1.3.2.9. Определяется эксцентрическая аномалия путем решения уравнения Кеплера

$$L - \omega = E - \varepsilon \cdot \sin E.$$

Как правило, используется схема последовательных приближений $m = 0, 1, 2$, и т.д.:

$$E^{(m+1)} = L - \omega + \varepsilon \cdot \sin E^{(m)},$$

1.3.2.10. Вычисляются истинная аномалия и аргумент широты НКА u :

$$v = 2 \arctan \left(\sqrt{\frac{1 + \varepsilon_A}{1 - \varepsilon_A}} \tan \frac{E}{2} \right);$$

$$u = v + \omega.$$

1.3.2.11. Рассчитываются координаты центра масс НКА в геоцентрической прямоугольной пространственной системе координат:

$$p = a(1 - (\varepsilon_A)^2);$$

$$r = \frac{p}{1 + \varepsilon_A \cos u};$$

$$x(t_i) = r(\cos \lambda \cos u - \sin \lambda \sin u \cos i);$$

$$y(t_i) = r(\sin \lambda \cos u + \cos \lambda \sin u \cos i);$$

$$z(t_i) = r \sin u \sin i.$$

1.4. Алгоритм расчета ионосферной погрешности

Воспользуемся алгоритмом расчета из ИКД GPS

Модель коррекции модели ионосферы

$$T_{iono} = \begin{cases} F \cdot \left[5 \cdot 10^{-9} + AMP \cdot \left(1 - \frac{x^2}{2} + \frac{x^4}{24} \right) \right], |x| < 1.57 \\ F \cdot [5 \cdot 10^{-9}], |x| \geq 1.57 \end{cases}, [сек]$$

Определим AMP

$$AMP = \begin{cases} \sum_{n=0}^3 a_n \phi_m^n, AMP \geq 0 \\ \text{Если } AMP < 0, AMP = 0 \end{cases}, [сек]$$

Где a_n - коэффициенты кубического уравнения, представляющие амплитуду вертикальной задержки

Определим фазу

$$x = \frac{2\pi(t - 50400)}{PER}, [рад]$$

Определим PER

$$PER = \begin{cases} \sum_{n=0}^3 \beta_n \phi_m^n, PER \geq 72.000 \\ \text{Если } PER < 72.000, PER = 72.000 \end{cases}, [сек]$$

Где β_n - коэффициенты кубического уравнения, представляющие период модели

Определим коэффициент наклона

$$F = 1 + 16[0.53 - E]^3$$

Где E - угол возвышения между пользователем и спутником

Определим геомагнитную широту земной проекции точки пересечения ионосферы (средняя высота ионосферы, предполагаемая 350 км)

$$\phi_m = \phi_i + 0.064 \cos(\lambda_i - 1.617), [\text{полуцикл}]$$

Определим геодезическая долгота земной проекции точки пересечения ионосферы

$$\lambda_i = \lambda_u + \frac{\psi \sin A}{\cos \phi_i}, [\text{полуцикл}]$$

Определим геодезическая широта земной проекции точки пересечения ионосферы

$$\phi_i = \left\{ \begin{array}{l} \phi_u + \cos A, |\phi_i| \leq 0.416 \\ \text{Если } \phi_i > 0.416, \text{ тогда } \phi_i = \phi_i + 0.416 \\ \text{Если } \phi_i < -0.416, \text{ тогда } \phi_i = \phi_i - 0.416 \end{array} \right\}, [\text{полуциклы}]$$

$$\phi_i = \left\{ \begin{array}{l} \phi_u + \psi \cos A, \quad |\phi_i| \leq 0.416 \\ \text{if } \phi_i > +0.416, \text{ then } \phi_i = +0.416 \\ \text{if } \phi_i < -0.416, \text{ then } \phi_i = -0.416 \end{array} \right\} \quad (\text{semi-circles})$$

Определим центральный угол Земли между положением пользователя и проекцией на землю точки пересечения ионосферы

$$\psi = \frac{0.0137}{E + 0.11} - 0.022, [\text{полуциклы}]$$

Определим локальное время:

$$t = 4.32 \cdot 10^4 \cdot \lambda_i + time, [сек]$$

Где $t \in 0 \leq t < 86400$,

time - вычисленное системное время приемника

Значения коэффициентов a_n, β_n берется из файла, который будем скачивать с сервера «информационно-аналитического центра координатно-временного и навигационного обеспечения, по адресу: «ftp://ftp.glonass-ias.ru/MCC/BRDC/», расширение файла 21n

```

BRDC0010 (1) - Блокнот
Файл  Правка  Формат  Вид  Справка
2.10      N: GPS NAV DATA      RINEX VERSION / TYPE
EPHEM CHECK 2.10      MCC D.ZALETAEV      03-Jan-21 20:00      PGM / RUN BY / DATE
A new version of accumulating navigation files by IANC      COMMENT
7.4510D-09 -1.4900D-08 -5.9600D-08 1.1920D-07      ION ALPHA
9.0110D+04 -6.5540D+04 -1.3110D+05 4.5880D+05      ION BETA
-3.725290298460D-09-1.065814103640D-14 61440 2139 DELTA-UTC: A0,A1,T,W
18      LEAP SECONDS
END OF HEADER
1 21 1 1 0 0 0.0 7.875198498368D-04-5.911715561524D-12 0.00000000000D+00
5.10000000000D+01-7.57500000000D+01 4.207318108830D-09-1.021277967070D+00
-3.935769200325D-06 1.022381347138D-02 1.428648829460D-06 5.153692249298D+03
4.32000000000D+05 2.216547727585D-07-8.086751269421D-01 2.309679985046D-07
9.827440567889D-01 3.61937500000D+02 8.220205737136D-01-8.23357246422D-09
-3.185846989029D-10 1.00000000000D+00 2.13800000000D+03 0.00000000000D+00
2.00000000000D+00 0.00000000000D+00 5.122274160385D-09 5.10000000000D+01
4.24818000000D+05 4.00000000000D+00
2 21 1 1 0 0 0.0-5.608978681266D-04-4.320099833421D-12 0.00000000000D+00
8.30000000000D+01-6.36875000000D+01 4.862702551080D-09-7.825898141899D-01
-3.134831786156D-06 2.031416934915D-02 1.531094312668D-06 5.153582004547D+03
4.32000000000D+05-2.36559339523D-07-8.894724859850D-01 1.154839992523D-07
9.615078420210D-01 3.55593750000D+02-1.567745133186D+00-8.812509933554D-09
-1.464346710204D-10 1.00000000000D+00 2.13800000000D+03 0.00000000000D+00
2.00000000000D+00 0.00000000000D+00-1.769512891769D-08 8.30000000000D+01
4.24818000000D+05 4.00000000000D+00
3 21 1 1 0 0 0.0-4.453537985682D-05-9.890754881781D-12 0.00000000000D+00
5.10000000000D+01-1.09875000000D+02 3.918020344131D-09-2.198381241254D+00
-5.602836608887D-06 3.261363017373D-03 1.010671257973D-05 5.153687782288D+03
4.32000000000D+05-2.980232238770D-08 2.298830274365D-01-7.264316082001D-08
9.68224906020D-01 1.89781250000D+02 8.808689872262D-01-7.586744589793D-09
1.292910997790D-10 1.00000000000D+00 2.13800000000D+03 0.00000000000D+00
2.00000000000D+00 0.00000000000D+00 1.862645149231D-09 5.10000000000D+01
4.24818000000D+05 4.00000000000D+00
4 21 01 01 00 0.0-1.690830104053D-04-3.069544618484D-12 0.00000000000D+00
8.00000000000D+01 8.21875000000D+00 4.643050388609D-09 7.181579851113D-01
4.731118679047D-07 1.009354135022D-03 7.683411240578D-06 5.153595529556D+03
4.32000000000D+05 2.793967723846D-08 1.309591313346D+00 1.676380634308D-08
9.598456408522D-01 2.30781250000D+02-3.072245845474D+00-7.931044265774D-09
5.975248629220D-10 1.00000000000D+00 2.13800000000D+03 0.00000000000D+00
2.00000000000D+00 0.00000000000D+00-4.190951585770D-09 8.00000000000D+01
4.24740000000D+05 0.00000000000D+00
5 21 01 01 00 0.0-2.980604767799D-05-1.023181539495D-12 0.00000000000D+00
9.80000000000D+01-1.07406250000D+02 4.315893864515D-09 1.814955427561D+00
-5.638226866722D-06 6.027893046848D-03 1.027062535286D-05 5.153797132492D+03
4.32000000000D+05-7.636845111847D-08 1.950148724094D-01-6.332993507385D-08
9.551620486418D-01 1.81500000000D+02 8.897933829174D-01-7.873184770801D-09
6.857428513918D-11 1.00000000000D+00 2.13800000000D+03 0.00000000000D+00
2.00000000000D+00 0.00000000000D+00-1.117587089539D-08 9.80000000000D+01
4.24740000000D+05 0.00000000000D+00

```

Рисунок 3 – Пример скаченного файла с расширением 21n
Необходимые нам коэффициенты расположены на 4 и 5 строчках.

ГЛАВА 2 РЕАЛИЗАЦИЯ АЛГОРИТМОВ В ПРОГРАММЕ

2.1. Скачивание файла

Для скачивания файлов модернизируем ранее созданный алгоритм «download» и для удобства последующих вызовов перенесем его в отдельный файл функции, который назовем: «FTPdownload», на вход которой подается разные пути и названия файла.

Функция содержит заголовочный файл – «FTPdownload.h», в котором хранятся применяемые классы и методы, а также файл с кодом реализации скачивания файла – «FTPdownload.CPP»

2.2. Обработка файла

Для обработки файлов также создадим отдельные функции, для ГЛОНАСС – «parserGLNS», а для GPS – «parserGPS»

2.2.1. Обработка файла GPS

Алгоритм обработки файла строится на методе «fscanf», которая обрабатывает последовательно каждое заданное значение, далее переносим полученные значения в массив значений «almanax_GPS».

Функция содержит заголовочный файл – «parserGPS.h», в котором хранятся применяемые классы и методы, а также файл с кодом реализации обработки файла – «parserGPS.C»

2.2.1. Обработка файла ГЛОНАСС

Алгоритм обработки файла строится на методе «fscanf», которая обрабатывает последовательно каждое заданное значение, далее переносим полученные значения в массив значений «almanax_GLNS».

Функция содержит заголовочный файл – «parserGLNS.h», в котором хранятся применяемые классы и методы, а также файл с кодом реализации обработки файла – «parserGLNS.C»

2.3. Расчет координат

Расчет координат для ГЛОНАСС и GPS выведем также в отдельные функции.

Для ГЛОНАСС функция принимает название – «ephemeridsGLNS», расчет соответствует формулам из п.1.3.2;

Функция содержит заголовочный файл – «ephemeridsGLNS.h», в котором хранятся применяемые классы и методы, а также файл с кодом реализации обработки файла – «ephemeridsGLNS.cpp»

Для GPS функция имеет название – «ephemerids», расчет соответствует формулам из п.1.1.2;

Функция содержит заголовочный файл – «ephemerids.h», в котором хранятся применяемые классы и методы, а также файл с кодом реализации обработки файла – «ephemerids.cpp»

2.4. Расчет времени

В процессе расчета координат возникнет проблема – получения времени расчета на которое нужно спрогнозировать координаты.

Для этого запишем класс – «timeCalc», в котором будет производиться перерасчет времени в нужный формат для трех ГНСС – ГЛОНАСС, GPS и GALILEO.

Для создания класса необходимо подать начальные значения: число, месяц, год, часы, минуты, секунды и миллисекунды.

Далее начальные значения преобразуются в секунды, с помощью встроенной библиотеки «ctime», а также подсчитывается количество поправок ко времени, для расчета в системе GPS и GALILEO.

В классе имеется три функции расчета времени:

- «timeGLNS» - для расчета времени в системе ГЛОНАСС,

- «timeGPS» - для расчета времени в системе GPS,
- «timeGLL» - для расчета времени в системе GALILEO.

2.3. Изменение интерфейса программы

The screenshot shows a window titled "Data app" with a close button (X) in the top right corner. Inside the window, there is a dropdown menu at the top left showing "Glonass". Below it is a table with 24 rows and 3 columns. The first column contains numbers 1 through 24. The second column is labeled "SISRE, m" and the third column is labeled "SISVE, mm/s". To the right of the table is a panel with several input fields and buttons. At the top right of the panel is a button labeled "Загрузить". Below it is a label "Время прогнозирования по UTC(+3)" followed by a date picker showing "07.06.2021" and a time picker showing "0:05:09". Below these are three labels: "Введите значение высоты в метрах:", "Введите значение В в градусах:", and "Введите значение L в градусах:", each followed by a text input field containing the number "0". At the bottom of the panel is a button labeled "Обработка" and a label "Значения СКО:".

	SISRE, m	SISVE, mm/s
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		

Рисунок 4 – Изменённый интерфейс программы

Добавил две новых формы:

- Ввод даты

Ввод даты производится с помощью встроенной формы в библиотеку `wxWidgets` – `wxDatePickerCtrl`, который реализован в виде небольшого окна, показывающего

текущую дату, элемент управления можно редактировать с помощью клавиатуры, а также с помощью мышки

- Ввод времени

Ввод даты производится с помощью встроенной формы в библиотеку `wxWidgets` – `wxTimePicerCtrl`, который реализован в виде небольшого окна, показывающего текущее время, элемент управления можно редактировать с помощью клавиатуры, а также с помощью мышки

2.4. Необходимые файлы для сборки проекта

К отчету прикреплены 22 приложения, в которых содержатся основные файлы кода программы, необходимые для сборки проекта. Так файл: “dataMain.cpp”, код описан в приложение 1, содержит в себе основной алгоритм настройки окна приложения, с помощью библиотек `wxWidgets`, также содержит алгоритм скачивания файла с сервера и заполнения таблицы данными SISRE и SISVE, путь к файлу: “...\data\dataMain.cpp”.

Файл: “parser.c”, код описан в приложение 2, содержит в себе обработку скаченного файла с сервера, с помощью алгоритма из приложения 1 [2]. Под обработкой подразумевается фильтрация нужной нам информации – значения SISRE и SISVE для определенного спутника, путь к файлу: “...\data\parser.c”.

Файл: “parser.h”, код описан в приложение 3, содержит в себе обработчик массива SISerr, для использования этого массива в приложение 1 [2], данный обработчик необходим, так как приложение 2 написано на языке «C», а приложение 1 на языке «C++», путь к файлу: “...\data\parser.h”.

Файл: “data.cbp”, код описан в приложение 4 – это необходимый файл для сборки проекта, в котором прописан используемый компилятор, библиотеки, а также все необходимые заголовочные файлы, путь к файлу: “...\data\data.cbp”.

Файл: “dataMain.h”, код описан в приложение 5 – это заголовочный файл, в котором хранятся применяемые классы и методы, путь к файлу: “...\data\dataMain.h”.

Файл “xyz2enu.cpp”, код описан в приложение 6, в файле реализована функция перевода из геодезической системы координат в топоцентрические координаты (ENU), путь к файлу: “...\data \ xyz2enu.cpp”

Файл “xyz2enu.h”, код описан в приложение 7 - это заголовочный файл, в котором объявляются применяемые классы и методы для функции перевода из геодезической системы координат в топоцентрические координаты (ENU) : “...\data \ xyz2enu.h”

Файл “ephemeridsGLNS.cpp”, код описан в приложение 8, в файле реализована функция описание движения спутников ГЛОНАСС по орбитам и нахождения координат спутников ГЛОНАСС в определенный момент, путь к файлу: “...\data \ ephemeridsGLNS.cpp ”

Файл “ ephemeridsGLNS.h”, код описан в приложение 9 - это заголовочный файл, в котором объявляются применяемые классы и методы для функции “ephemerids”, путь к файлу: “...\data \ ephemeridsGLNS.h ”

Файл “angle.cpp”, код описан в приложение 10, в файле реализована функция расчета видимости спутников, путь к файлу: “...\data \ angle.cpp ”

Файл “ angle.h ”, код описан в приложение 11 - это заголовочный файл, в котором объявляются применяемые классы и методы расчета видимости спутников, путь к файлу: “...\data \ angle.h ”

Файл: “datadiaslog.wxh”, код описан в приложение 12 – это файл описания графического пользовательского интерфейса для плагина wxSmith, путь к файлу: “...\data\ wxsmith\datadiaslog.wxh”.

Файл “timeCalc.cpp”, код описан в приложение 13, в файле реализован класс перевода времени для трех систем СРНС, путь к файлу: “...\data \ timeCalc.cpp ”

Файл “timeCalc.h”, код описан в приложение 14 - это заголовочный файл, в котором объявляются применяемые классы и методы для класса перевода времени для трех систем СРНС : “...\data \ timeCalc.h”

Файл “FTPdownload.cpp”, код описан в приложение 15, в файле реализована функция скачивания файлов с сервера, путь к файлу: “...\data \ FTPdownload.cpp ”

Файл “FTPdownload.h ”, код описан в приложение 16 - это заголовочный файл, в котором объявляются применяемые классы и методы скачивания файлов с сервера, путь к файлу: “...\data \ FTPdownload.h ”

Файл: “parserGLNS.c”, код описан в приложение 17, содержит в себе обработку скаченного файла с сервера. Под обработкой подразумевается фильтрование нужной нам информации – значения SISRE и SISVE для определенного спутника, путь к файлу: “...\data\parserGLNS.c”.

Файл: “ parserGLNS.h”, код описан в приложение 18, содержит в себе обработчик массива SISerr, данный обработчик необходим, так как приложение 2 написано на языке «C», а приложение 1 на языке «C++», путь к файлу: “...\data\parserGLNS.h”.

Файл: “parserGPS.c”, код описан в приложение 19, содержит в себе обработку скаченного файла с сервера. Под обработкой подразумевается фильтрование нужной нам информации – значения SISRE и SISVE для определенного спутника, путь к файлу: “...\data\parserGPS.c”.

Файл: “ parserGPS.h”, код описан в приложение 20, содержит в себе обработчик массива SISerr, данный обработчик необходим, так как

приложение 2 написано на языке «С», а приложение 1 на языке «С++», путь к файлу: "...\\data\\parserGPS.h".

Файл "ephemerids.cpp", код описан в приложение 21, в файле реализована функция описание движения спутников GPS по орбитам и нахождения координат спутников GPS в определенный момент, путь к файлу: "...\\data \\ ephemerids.cpp "

Файл "ephemerids.h", код описан в приложение 22 - это заголовочный файл, в котором объявляются применяемые классы и методы для функции "ephemerids", путь к файлу: "...\\data \\ ephemerids.h "

ГЛАВА 3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Минимальные требования:

Для запуска программы необходимо иметь windows 7/10 и подключенное устройство к интернету.

Инструкция:

1. Запустите программу “Data.exe” от имени администратора

Если программа не запустится отключите антивирус.

У вас появится диалоговое окно:

Data app

Glonass

Загрузить

Время прогнозирования по UTC(+3)

07.06.2021

1:02:50

Введите значение высоты в метрах:

0

Введите значение B в градусах:

0

Введите значение L в градусах:

0

Обработка

Значения СКО:

	SISRE, m	SISVE, mm/s
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		

Рисунок 3.1. Интерфейс программы

2. Выберите необходимую вам НС:

Data app

Глобальная навигационная спутниковая система	Время прогноза по UTC(+3)	Высота в метрах	В широте	В долготу
Глобальная навигационная спутниковая система	07.06.2021	0		
GPS	1:02:50			
Galileo				
Beidou				
QZSS				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				

Загрузить

Время прогнозирования по UTC(+3)
07.06.2021

1:02:50

Введите значение высоты в метрах:
0

Введите значение В в градусах:
0

Введите значение L в градусах:
0

Обработка

Значения СКО:

Рисунок 3.2. Интерфейс выбора НС

3. Нажмите кнопку «Загрузить»

Data app

Гlonass

	SISRE, m	SISVE, mm/s
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		

Загрузить

Время прогнозирования по UTC(+3)
07.06.2021
1:02:50

Введите значение высоты в метрах:
0

Введите значение В в градусах:
0

Введите значение L в градусах:
0

Обработка

Значения СКО:

Рисунок 3.3. Интерфейс выбранной НС

Получили значения SISRE и SISVE для каждого спутника, если значения равны 0.00, то данный спутник отсутствует.

Если при загрузке возникла ошибка существует два варианта решения ее:

- Отключите антивирус,
- Включите брандмауэр.

4. Для того, чтобы скачать данные для других НС, перейдите к п. 2 инструкции.

Data app

Glomass

Загрузить

	SISRE, m	SISVE, mm/s
1	0.427	0.508
2	1.205	0.650
3	0.863	0.511
4	0.851	0.569
5	0.682	0.597
6	0.000	0.000
7	0.686	0.469
8	2.104	0.506
9	2.751	1.216
10	0.000	0.000
11	0.000	0.000
12	1.390	0.753
13	1.451	0.657
14	0.914	0.540
15	0.519	0.562
16	1.392	0.580
17	0.440	0.701
18	1.481	0.551
19	1.023	0.591
20	1.190	0.768
21	0.849	0.636
22	1.324	0.796
23	0.000	0.000
24	0.690	0.534

Введите значение высоты:

Введите значение В:

Введите значение L:

Обработка

СКО для x:

СКО для y:

СКО для z:

СКО для D:

Рисунок 3.4. Полученные значения

5. Далее вводим значения:

Data app

Glomass

	SISRE, m	SISVE, mm/s
1	1.647	0.604
2	1.488	0.764
3	0.696	0.642
4	0.618	0.647
5	0.438	0.571
6	0.000	0.000
7	1.364	0.600
8	1.760	0.638
9	1.522	0.853
10	0.000	0.000
11	0.000	0.000
12	0.837	0.733
13	1.695	0.714
14	0.802	0.633
15	0.933	0.709
16	1.664	0.680
17	0.896	0.737
18	0.974	0.635
19	1.029	0.782
20	0.868	0.819
21	1.534	0.641
22	1.183	0.618
23	0.000	0.000
24	1.499	0.639

Загрузить

Время прогнозирования по UTC(+3)
07.06.2021

1:07:05

Введите значение высоты в метрах:
200

Введите значение В в градусах:
37.611680

Введите значение L в градусах:
55.819715

Обработка

Значения СКО:

Рисунок 3.5. Ввод значений в формы

6. Нажимаем кнопку обработка:

The screenshot shows a software window titled "Data app" with a close button (X) in the top right corner. Inside the window, there is a dropdown menu set to "Glonass". Below it is a table with 24 rows and 3 columns. The first column contains numbers 1 through 24. The second column is labeled "SISRE, m" and the third is labeled "SISVE, mm/s". To the right of the table, there is a "Загрузить" (Load) button. Below this button, there are three input fields: "Время прогнозирования по UTC(+3)" with the value "07.06.2021", a time field with "1:07:39", a height field labeled "Введите значение высоты в метрах:" with the value "200", a latitude field labeled "Введите значение В в градусах:" with the value "37.611680", and a longitude field labeled "Введите значение L в градусах:" with the value "55.819715". Below these fields is an "Обработка" (Processing) button. At the bottom right, there is a section titled "Значение СКО:" (SD value) with the following text: "СКО для x: 1.478 м", "СКО для y: 1.681 м", "СКО для z: 1.665 м", and "СКО для D: 1.480, м СКО: 2.790 м".

	SISRE, m	SISVE, mm/s
1	1.647	0.604
2	1.488	0.764
3	0.696	0.642
4	0.618	0.647
5	0.438	0.571
6	0.000	0.000
7	1.364	0.600
8	1.760	0.638
9	1.522	0.853
10	0.000	0.000
11	0.000	0.000
12	0.837	0.733
13	1.695	0.714
14	0.802	0.633
15	0.933	0.709
16	1.664	0.680
17	0.896	0.737
18	0.974	0.635
19	1.029	0.782
20	0.868	0.819
21	1.534	0.641
22	1.183	0.618
23	0.000	0.000
24	1.499	0.639

Рисунок 3.6. Полученные значения

Получаем значения СКО.

7. Чтобы закрыть программу нажмите крестик в диалоговом окне

ЗАКЛЮЧЕНИЕ

Получил значения СКО для x, y, z - координаты и погрешности временной шкалы, которые рассчитываются по данным альманахам соответствующих группировок для созвездий ГЛОНАСС и GPS.

Не все задачи удалось выполнить: добавить алгоритм расчета ионосферных и тропосферных погрешностей, из-за затянувшегося изменения функции расчета координат НКА по данным альманахов, так как для изменения ее пришлось написать и переписать ряд функций: функции обработки файлов с альманахами, перерасчет времени, алгоритм скачивания файлов с сервера.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1]. Сервер «информационно-аналитического центра координатно-временного и навигационного обеспечения «<ftp://glonass-iac.ru>» »

[2]. Отчет НИР за 9 семестр

ПРИЛОЖЕНИЕ

Приложение 1

```
#include "dataMain.h"
#include "FTPdownl.h"
#include "parser.h"
#include "parserGPS.H"
#include "angle.h"
#include "ephemerids.h"
#include "xyz2enu.h"
#include "parserGLNS.H"
#include "ephemeridsGLNS.h"
#include "timeCalc.h"
#include <wx/msgdlg.h>
#include <windows.h>
#include <wininet.h>
#include <iostream>
#include <string>
#include <stdio.h>
#include <armadillo>
#include <fstream>

#include <wx/string.h>
#include <wx/textfile.h>
#include <wx/dialog.h>
#include <wx/msgdlg.h>
#include <wx/spinctrl.h>
#include <wx/intl.h>
#include <wx/settings.h>

#define SQUARE(val) val * val

using namespace std;
using namespace arma;

//(*InternalHeaders(dataDialog)
#include <wx/intl.h>
#include <wx/settings.h>
#include <wx/string.h>
//*)

//helper functions
enum wxbuildinfoformat
{
```



```

    short_f, long_f
};

wxString wxbuildinfo(wxbuildinfoformat format)
{
    wxString wxbuild(wxVERSION_STRING);

    if (format == long_f )
    {
#ifdef __WXMSW__
        wxbuild << _T("-Windows");
#elif defined(__UNIX__)
        wxbuild << _T("-Linux");
#endif

#ifdef wxUSE_UNICODE
        wxbuild << _T("-Unicode build");
#else
        wxbuild << _T("-ANSI build");
#endif // wxUSE_UNICODE
    }

    return wxbuild;
}

//(*IdInit(dataDialog)
const long dataDialog::ID_DATEPICKERCTRL1 = wxNewId();
const long dataDialog::ID_CHOICE1 = wxNewId();
const long dataDialog::ID_BUTTON2 = wxNewId();
const long dataDialog::ID_NOTEBOOK1 = wxNewId();
const long dataDialog::ID_BUTTON1 = wxNewId();
const long dataDialog::ID_TEXTCTRL1 = wxNewId();
const long dataDialog::ID_TEXTCTRL2 = wxNewId();
const long dataDialog::ID_TEXTCTRL3 = wxNewId();
const long dataDialog::ID_STATICTEXT1 = wxNewId();
const long dataDialog::ID_STATICTEXT2 = wxNewId();
const long dataDialog::ID_STATICTEXT3 = wxNewId();
const long dataDialog::ID_STATICTEXT4 = wxNewId();
const long dataDialog::ID_TIMEPICKERCTRL1 = wxNewId();
const long dataDialog::ID_BUTTON4 = wxNewId();
const long dataDialog::ID_STATICTEXT5 = wxNewId();
const long dataDialog::ID_SASHWINDOW1 = wxNewId();
//*)

const long dataDialog::ID_GRID = wxNewId();
BEGIN_EVENT_TABLE(dataDialog,wxDialog)

```

```

        //(*EventTable(dataDialog)
        //*)
    END_EVENT_TABLE()
/*
    bool download(LPCSTR server, LPCSTR login, LPCSTR pass, LPCSTR local_file,
LPCSTR remote_file)
    {
        bool status;
        HINTERNET hOpen, hConnection;

        hOpen = InternetOpen(NULL, INTERNET_OPEN_TYPE_DIRECT, NULL, NULL, 0);
        if (hOpen == NULL)
            return false;

        hConnection = InternetConnectA(hOpen, server, 21, login, pass,
INTERNET_SERVICE_FTP, INTERNET_FLAG_PASSIVE, 0 );
        if (hConnection == NULL)
        {
            InternetCloseHandle(hOpen);
            return false;
        }

        status=FtpGetFileA(hConnection, local_file, remote_file, true, 0,
FTP_TRANSFER_TYPE_UNKNOWN, 0);
        InternetCloseHandle(hConnection);
        InternetCloseHandle(hOpen);
        return status;
    }
*/

dataDialog::dataDialog(wxWindow* parent,wxWindowID id)
{
    //(*Initialize(dataDialog)
    Create(parent, wxID_ANY, _("Data app"), wxDefaultPosition, wxDefaultSize,
wxDEFAULT_DIALOG_STYLE, _T("wxID_ANY"));
    SetClientSize(wxSize(533,556));
    SetMinSize(wxSize(-1,-1));
    SetMaxSize(wxSize(-1,-1));

    SetBackgroundColour(wxSystemSettings::GetColour(wxSYS_COLOUR_INACTIVEBORDER));

    SashWindow1 = new wxSashWindow(this, ID_SASHWINDOW1, wxPoint(56,40),
wxSize(480,504), wxSW_3D|wxCLIP_CHILDREN, _T("ID_SASHWINDOW1"));
    SashWindow1-
>SetForegroundColour(wxSystemSettings::GetColour(wxSYS_COLOUR_WINDOWTEXT));

```

```

SashWindow1-
>SetBackgroundColour(wxSystemSettings::GetColour(wxSYS_COLOUR_MENUBAR));

    DatePickerCtrl1 = new wxDatePickerCtrl(SashWindow1, ID_DATEPICKERCTRL1,
wxDefaultDateTime, wxPoint(305,65), wxSize(85,21), wxDP_DEFAULT|wxDP_SHOWCENTURY,
wxDefaultValidator, _T("ID_DATEPICKERCTRL1"));

    DatePickerCtrl1-
>SetForegroundColour(wxSystemSettings::GetColour(wxSYS_COLOUR_ACTIVEBORDER));

    DatePickerCtrl1-
>SetBackgroundColour(wxSystemSettings::GetColour(wxSYS_COLOUR_HIGHLIGHT));

    Choice1 = new wxChoice(SashWindow1, ID_CHOICE1, wxPoint(33,16),
wxSize(244,21), 0, 0, 0, wxDefaultValidator, _T("ID_CHOICE1"));
    Choice1->SetSelection( Choice1->Append(_("Glonass")) );
    Choice1->Append(_("GPS"));
    Choice1->Append(_("Galileo"));
    Choice1->Append(_("Beidou"));
    Choice1->Append(_("QZSS"));

    Button2 = new wxButton(SashWindow1, ID_BUTTON2, _("Загрузить"),
wxPoint(305,16), wxSize(127,23), 0, wxDefaultValidator, _T("ID_BUTTON2"));

    Notebook1 = new wxNotebook(SashWindow1, ID_NOTEBOOK1, wxPoint(124,214),
wxDefaultSize, 0, _T("ID_NOTEBOOK1"));

    Down = new wxButton(SashWindow1, ID_BUTTON1, _("Обработка"),
wxPoint(305,266), wxSize(127,23), 0, wxDefaultValidator, _T("ID_BUTTON1"));

    TextCtrlH = new wxTextCtrl(SashWindow1, ID_TEXTCTRL1, _("0"),
wxPoint(305,145), wxSize(127,-1), 0, wxDefaultValidator, _T("ID_TEXTCTRL1"));

    TextCtrlB = new wxTextCtrl(SashWindow1, ID_TEXTCTRL2, _("0"),
wxPoint(305,190), wxSize(127,-1), 0, wxDefaultValidator, _T("ID_TEXTCTRL2"));

    TextCtrlL = new wxTextCtrl(SashWindow1, ID_TEXTCTRL3, _("0"),
wxPoint(305,235), wxSize(127,-1), 0, wxDefaultValidator, _T("ID_TEXTCTRL3"));

    StaticText1 = new wxStaticText(SashWindow1, ID_STATICTEXT1, _("Введите
значение высоты в метрах:"), wxPoint(305,127), wxDefaultSize, 0,
_T("ID_STATICTEXT1"));

    StaticText2 = new wxStaticText(SashWindow1, ID_STATICTEXT2, _("Введите
значение В в градусах:"), wxPoint(305,172), wxDefaultSize, 0, _T("ID_STATICTEXT2"));

    StaticText3 = new wxStaticText(SashWindow1, ID_STATICTEXT3, _("Введите
значение L в градусах:"), wxPoint(305,217), wxDefaultSize, 0, _T("ID_STATICTEXT3"));

    StaticText4 = new wxStaticText(SashWindow1, ID_STATICTEXT4, _("Значения
СКО:"), wxPoint(306,294), wxDefaultSize, 0, _T("ID_STATICTEXT4"));

    TimePickerCtrl1 = new wxTimePickerCtrl(SashWindow1, ID_TIMEPICKERCTRL1,
wxDateTime::Now(), wxPoint(305,93), wxSize(85,21), 0, wxDefaultValidator,
_T("ID_TIMEPICKERCTRL1"));

    Button3 = new wxButton(SashWindow1, ID_BUTTON4, _("для отладки/test"),
wxPoint(303,463), wxDefaultSize, 0, wxDefaultValidator, _T("ID_BUTTON4"));

    StaticText5 = new wxStaticText(SashWindow1, ID_STATICTEXT5, _("Время
прогнозирования по UTC(+3)"), wxPoint(305,49), wxDefaultSize, 0,
_T("ID_STATICTEXT5"));

    SashWindow1->SetSashVisible(wxSASH_TOP, true);

```

```

        SashWindow1->SetSashVisible(wxSASH_BOTTOM, true);
        SashWindow1->SetSashVisible(wxSASH_LEFT, true);
        SashWindow1->SetSashVisible(wxSASH_RIGHT, true);

Connect( ID_DATEPICKERCTRL1,wxEVT_DATE_CHANGED,(wxObjectEventFunction)&dataDialog::On
DatePickerCtrl1Changed);

Connect( ID_BUTTON2,wxEVT_COMMAND_BUTTON_CLICKED,(wxObjectEventFunction)&dataDialog::
OnButton2Click);

Connect( ID_BUTTON1,wxEVT_COMMAND_BUTTON_CLICKED,(wxObjectEventFunction)&dataDialog::
OnButton1Click1);

Connect( ID_TEXTCTRL1,wxEVT_COMMAND_TEXT_UPDATED,(wxObjectEventFunction)&dataDialog::
OnTextCtrl1Text1);

Connect( ID_TIMEPICKERCTRL1,wxEVT_DATE_CHANGED,(wxObjectEventFunction)&dataDialog::On
TimePickerCtrl1Changed);

Connect( ID_BUTTON4,wxEVT_COMMAND_BUTTON_CLICKED,(wxObjectEventFunction)&dataDialog::
OnButton3Click2);

Connect( ID_SASHWINDOW1,wxEVT_SASH_DRAGGED,(wxObjectEventFunction)&dataDialog::OnSash
Window1SashDragged);

Connect( wxID_ANY,wxEVT_INIT_DIALOG,(wxObjectEventFunction)&dataDialog::OnInit);
        /*)

        Grid = new wxGrid(SashWindow1, ID_GRID, wxPoint(33,60), wxSize(244,490), 0,
_T("ID_GRID"));

        Grid->CreateGrid(24,2);
        Grid->SetColLabelValue(0, _("SISRE, m"));
        Grid->SetColLabelValue(1, _("SISVE, mm/s"));
        Grid->SetDefaultCellFont( Grid->GetFont() );
        Grid->SetDefaultCellTextColour( Grid->GetForegroundColour() );
    }

dataDialog::~dataDialog()
{
    /*(*Destroy(dataDialog)
    /**)
}

```

```

void dataDialog::OnQuit(wxCommandEvent& event)
{
    Close();
}

void dataDialog::OnInit(wxInitDialogEvent& event) {};

void dataDialog::OnAbout(wxCommandEvent& event)
{
    wxString msg = wxbuildinfo(long_f);
    wxMessageBox(msg, _("Welcome to..."));
}

void dataDialog::OnChoice1Select(wxCommandEvent& event)
{
}

void dataDialog::OnSashWindow1SashDragged(wxSashEvent& event)
{
}

void dataDialog::OnButton2Click(wxCommandEvent& event)
{
    wxString s;
    const char* File1 ;
    const char* file ;
    //wxMessageBox(Choice1->GetString(Choice1->GetSelection()), _(""));
    if ((Choice1->GetString(Choice1->GetSelection()))== "GPS"s)
    {
        File1 = "/MCC/PRODUCTS/LATEST/MERMS-GSC_C.ete";
        file = "MERMS-GSC_C.ete";
        wxTextFile file11(wxT("MERMS-GSC_C.ete"));
        if (file11.Exists())
        {
            wxRemoveFile(file);
        }
    }
    if ((Choice1->GetString(Choice1->GetSelection()))== "Glonass"s)
    {
        File1 = "/MCC/PRODUCTS/LATEST/MERMS-RSC_C.ete";
        file = "MERMS-RSC_C.ete";
        wxTextFile file11(wxT("MERMS-RSC_C.ete"));
    }
}

```

```

        if (file11.Exists())
        {
            wxRemoveFile(file);
        }
    }
    if ((Choice1->GetString(Choice1->GetSelection()))== "Galileo"s)
    {
        File1 = "/MCC/PRODUCTS/LATEST/MERMS-ESC_C.ete";
        file = "MERMS-ESC_C.ete";
        wxTextFile file11(wxT("MERMS-ESC_C.ete"));
        if (file11.Exists())
        {
            wxRemoveFile(file);
        }
    }
    if ((Choice1->GetString(Choice1->GetSelection()))== "Beidou"s)
    {
        File1 = "/MCC/PRODUCTS/LATEST/MERMS-CSC_C.ete";
        file = "MERMS-CSC_C.ete";
        wxTextFile file11(wxT("MERMS-CSC_C.ete"));
        if (file11.Exists())
        {
            wxRemoveFile(file);
        }
    }
    if ((Choice1->GetString(Choice1->GetSelection()))== "QZSS"s)
    {
        File1 = "/MCC/PRODUCTS/LATEST/MERMS-JSC_C.ete";
        file = "MERMS-JSC_C.ete";
        wxTextFile file11(wxT("MERMS-JSC_C.ete"));
        if (file11.Exists())
        {
            wxRemoveFile(file);
        }
    }
    bool down = download( "glonass-iac.ru", NULL, NULL, File1, file);
    if (!down)
    {
        wxMessageBox(_("Error"), _("Error"));
        return;
    }
    Gridd(file);
}

```

```

void dataDialog::Gridd(const char* file)
{
    int k=0;
    int sizeY;
    if (Grid != NULL)
    {
        delete Grid;
    }
    memset(&SISerr,0, sizeof(SISerr));
    int max_sats = parse(file);
    sizeY=490;
    Grid = new wxGrid(SashWindow1, ID_GRID, wxPoint(33,60), wxSize(244,sizeY), 0,
_T("ID_GRID"));
    wxString s;
    Grid->CreateGrid(max_sats,2);
    Grid->SetColLabelValue(0, _("SISRE, m"));
    Grid->SetColLabelValue(1, _("SISVE, mm/s"));
    Grid->SetDefaultCellFont( Grid->GetFont() );
    Grid->SetDefaultCellTextColour( Grid->GetForegroundColour() );
    for (k=0; k<max_sats; k++ )
    {
        Grid->SetCellValue((k), 0, wxString::Format("%.3f", SISerr[k].SISRE));
        Grid->SetCellValue((k), 1, wxString::Format("%.3f", SISerr[k].SISVE));
    }
}

void dataDialog::OnChoice1Select3(wxCommandEvent& event)
{
}

void dataDialog::OnTextCtrl1Text1(wxCommandEvent& event)
{
}

void dataDialog::OnSpinCtrl1Change(wxSpinEvent& event)
{
}

void dataDialog::OnChoice1Select4(wxCommandEvent& event)
{
}

void dataDialog::OnButton1Click1(wxCommandEvent& event)
{
    StaticText4 ->ClearBackground();
}

```

```

// Вводим значения h,B,L
double h;
double Bgrad;
double Lgrad;
TextCtrlH->GetValue().ToDouble(&h);
TextCtrlB->GetValue().ToDouble(&Bgrad);
TextCtrlL->GetValue().ToDouble(&Lgrad);

double PI = M_PI;
double B; //Latitude
double L; //Longitude
B=Bgrad*PI/180;
L=Lgrad*PI/180;
double N;
double e=0;
double a=6378136; // радиус З

// Получение координат потребителя
N=a/sqrt(1-(e*e)*(sin(B))*(sin(B)));

double Coord_x;
double Coord_y;
double Coord_z;
Coord_x = (N+h)*cos(B)*cos(L);
Coord_y = (N+h)*cos(B)*sin(L);
Coord_z = ((1-e*e)*N+h)*sin(B);
double Coord_user[3];
Coord_user[0]=(N+h)*cos(B)*cos(L);
Coord_user[1]= (N+h)*cos(B)*sin(L);
Coord_user[2]= ((1-e*e)*N+h)*sin(B);

double Coord_sput[3];
double alpha;
//передаем в класс определения времени
//Т.е получили время на которое необходимо предсказать
//Далее "найдем" файл от которого будет высчитывать само предсказание
//если от сегодн. дня, то день -1; тк файл загружается ~ в 18 00;

// Считаем сегодняшн. дату
int year_predsk;
int month_predsk;
int day_predsk;
int hour_predsk;
int min_predsk;
int sec_predsk;
//время от которого скачиваем

```



```

    int year_down;
    int month_down;
    int day_down;
    //int hour_down;
    //int min_down;
    //int sec_down ;
    wxDateTime T;
    T = DatePickerCtrl1->GetValue();
    day_predsk = T.GetDay(); //для скачивания файла
    month_predsk = T.GetMonth()+1; // тк 1 месяц равен 0;
    year_predsk = T.GetYear();
    //получаем дату и время
    // Дата и время от которой предсказывать:
    TimePickerCtrl1->GetTime(&hour_predsk, &min_predsk, &sec_predsk);

    time_t nowsec = time(0);
    tm *ltm = localtime(&nowsec);
    int yeartoday = 1900+ltm->tm_year;
    int monthtoday = 1 + ltm->tm_mon;
    int daytoday = ltm->tm_mday;
    int hourrtoday = ltm->tm_hour;
    int mintoday = 1 + ltm->tm_min;
    int sectoday = 1 + ltm->tm_sec;

    // Если предсказание в прошлом, то год ии месяц предсказания остется тот же,
но день - прошлый
    day_down = day_predsk-1;
    year_down = year_predsk;
    month_down = month_predsk;
    ofstream f;
    f.open("test/test2.txt");
    f<< "year_predsk="<< year_predsk<<endl;
    f<< "month_predsk="<< month_predsk<<endl;
    f<< "day_predsk="<< day_predsk<<endl;
    f<< "hour_predsk="<< hour_predsk<<endl;
    f<< "min_predsk="<< min_predsk<<endl;
    f<< "sec_predsk="<< sec_predsk<<endl;

    //если предсказание уже на будущее, то год,месяц остается сегодняшний, а день
минус 1
    //если год = году сейчас, но день больше или равен, то день минус 1
    // по сути можно облегчить и сделать 1 цикл через "или"
    if (year_predsk>yeartoday)
    {

```

```

        day_down = daytoday -1;
        year_down = yeartoday;
        month_down = monthtoday;
    }
    if (year_predsk == yeartoday)
    {
        if ( day_predsk >= daytoday)
        {
            day_down = daytoday -1;
            year_down = yeartoday;
            month_down = monthtoday;
        }
    }
    f<< "day_download="<< day_down<<endl;
    f<< "month_download="<< month_down<<endl;
    f<< "year_download="<< year_down<<endl;
    timeCalc
    calc(day_predsk,month_predsk,year_predsk,hour_predsk,min_predsk,sec_predsk,00);

    //
    //преобразование в слово для скачивания
    string textYear = to_string(year_down);
    string text5 = "MCCJ_";
    string text2 = to_string(year_down -2000);
    string text3 ;
    if (month_down<10)
    {
        text3 = "0"s + to_string(month_down);
    }
    else
    {
        text3 = to_string(month_down);
    }
    string text4;

    if (day_down<10)
    {
        text4 = "0"s +to_string(day_down);
    }
    else
    {
        text4 = to_string(day_down);
    }
    string text1;
    string text0;
    mat sko;

```

```

if ((Choice1->GetString(Choice1->GetSelection()))== "GPS")
{

    text1 = text5+text2+text3+text4+".agp"s;
    text0 =  "/MCC/ALMANAC/" + textYear + "/" + text1;
    f<< " day_down="<< day_down<<endl;
    f<< " text4="<< text4<<endl;
    const char* File1 ;
    const char* file ;
    File1 = text0.c_str();//"/MCC/ALMANAC/2015/MCCJ_150307.agp"//перевод строки
с строку Си
    file = text1.c_str();
    //! добавить если файла нет, искать ближайший!
    f<< "const char* File1"<< File1<<endl;
    f<< " file"<< file<<endl;
    bool down = download( "glonass-iac.ru", NULL, NULL, File1, file);
    int max_sats = parseGPS(file);

// Расчет матрицы Dn, Hn, SKO
    int numberSput = 32;
    int vsb[numberSput] ;
    int sumvsb = 0;
    vector<int> Visibles; //вектор из кол-во элементов - visibles
    calc.timeGPS();
    double toe=calc.sec_since_week;

    f<< "GPS:"<<endl;
    f<<"toe()calc.sec_since_week="<<toe<<endl;
    f<<"week="<<calc.week<<endl;
    Coordinates Coord_sp; // можно потом заменить в 482 строке и ниже.

    for (int i=1; i<=numberSput; i++)
    {
        // Получение коорд спутников
        //ephemerids(double toe,int t_almanax, double M0, double sqrtA, double E, double
I, double Om0, double time_week ))
        Coord_sp = ephemerids(toe,
                                almanax_GPS[i-1].t_almanax,
                                almanax_GPS[i-1].M0,
                                almanax_GPS[i-1].sqrtA,
                                almanax_GPS[i-1].E,
                                almanax_GPS[i-1].I,
                                almanax_GPS[i-1].Om0,
                                almanax_GPS[i-1].time_week);

        f <<"i-1 (номер спут)"<<i-1<<endl;
    }
}

```

```

f<<"Coord_sp.X="<<Coord_sp.X<<endl;
f <<"Coord_sp.Y =" <<Coord_sp.Y<<endl;
f <<"Coord_sp.Z =" <<Coord_sp.Z<<endl;

Coord_sput[0] = Coord_sp.X;
Coord_sput[1] = Coord_sp.Y;
Coord_sput[2] = Coord_sp.Z;
// Определение угла
alpha = 90 - (angle(Coord_sput, Coord_user, B, L)*180/PI);
// определение видимости спутника
vsb[i]=0;
if ((alpha) >5)
{
vsb[i]=1;
sumvsb++;
Visibles.push_back(i); // добавление элемента в конец вектора
}
}

// получение матрицы Dn
int i = 0;
mat Dn;
Dn.zeros(sumvsb, sumvsb);
for (int k=1; k<=numberSput; k++)
{
if ((vsb[k]) == 1)
{
Dn(i,i) = SISerr[i].SISRE;
i++;
}
}
double max_val_Dn = Dn.max();
for (int i= 0; i<sumvsb; i++)
{
if ( Dn(i,i) == 0)
{
Dn(i,i) = max_val_Dn;
}
}

// получение матрицы H
double dx;
double dy;
double dz;
double Ri;

```

```

mat H(sumvsb, 4);
H.zeros();

int numspout = 0;
for (int k=1; k<=numberSput; k++)
{
    if ((vsb[k]) == 1)
    {

        Coord_sp = ephemerids(toe,
                                almanax_GPS[k-1].t_almanax,
                                almanax_GPS[k-1].M0,
                                almanax_GPS[k-1].sqrtA,
                                almanax_GPS[k-1].E,
                                almanax_GPS[k-1].I,
                                almanax_GPS[k-1].Om0,
                                almanax_GPS[k-1].time_week);

        dx=(Coord_sp.X-Coord_x);
        dy=(Coord_sp.Y-Coord_y);
        dz=(Coord_sp.Z- Coord_z);
// Ri = sqrt (SQUARE(dx)+SQUARE(dy)+SQUARE(dz));
        Ri = sqrt (pow(dx,2)+pow(dy,2)+pow(dz,2));

        H(numspout, 0 ) = dx/Ri;
        H(numspout, 1) = dy/Ri;
        H(numspout, 2) = dz/Ri;
        H(numspout, 3) = 1;
        numspout++ ;
    }
}
mat Htr = H.t();
sko = sqrt((inv(Htr*inv(Dn)*H)).t());

}
else if ((Choice1->GetString(Choice1->GetSelection()))== "Glonass")
{
    text1 = text5+text2+text3+text4+".agl"s;
    text0 =  "/MCC/ALMANAC/"+ textYear +"/"+text1;
    f<< " day_down="<< day_down<<endl;
    f<< " text4="<< text4<<endl;
    const char* File1 ;
    const char* file ;
    File1 = text0.c_str();//"/MCC/ALMANAC/2015/MCCJ_150307.agl"//перевод строки
с строку Си
    file = text1.c_str();

```

```

    //! добавить если файла нет, искать ближайший!
    f<< "const char* File1"<< File1<<endl;
    f<< " file"<< file<<endl;

    bool down = download( "glonass-iac.ru", NULL, NULL, File1, file);
    int max_sats = parseGLNS(file);

    int numberSput = 24;
    int vsb[numberSput] ;
    int sumvsb = 0;
    vector<int> Visibles; //вектор из кол-во элементов - visibles
    //double toe=44271.777;//время
    f<< "Glns:"<<endl;
    f<<"calc.GLNS_numb_fouryear_period                (N4)="                <<
calc.GLNS_numb_fouryear_period<<endl;;
    f <<"calc.GLNS_sec_since_week=" <<calc.GLNS_sec_since_week<<endl;;
    GlonassCoordinates Coord_sp;
    for (int i=1; i<=numberSput; i++)
    {
        // Получение коорд спутников
        //ephemerids(double toe,int t_almanax, double M0, double sqrtA, double E, double
I, double Om0, double time_week ))

        calc.timeGLNS();
        timeCalc GLNSephemTime( almanax_GLNS[i-1].date,almanax_GLNS[i-1].month,
almanax_GLNS[i-1].year,0,0,0,0);
        Coord_sp = ephemeridsGLNS(calc.GLNS_numb_fouryear_period, //N4
                                calc.GLNS_day_after_vis_year,
                                calc.GLNS_sec_since_week,
                                GLNSephemTime.GLNS_numb_fouryear_period, //Na
берем из расчета даты альманаха
                                almanax_GLNS[i-1].tLA,
                                almanax_GLNS[i-1].dT,
                                almanax_GLNS[i-1].dT,
                                almanax_GLNS[i-1].dTT,
                                almanax_GLNS[i-1].E,
                                almanax_GLNS[i-1].w,
                                almanax_GLNS[i-1].Lam);

        f <<"i-1 (номер спутн) ="<<i-1<<endl;
        f                <<                "GLNS_numb_fouryear_period                (Na)"<<
GLNSephemTime.GLNS_numb_fouryear_period<<endl;
        f<<"Coord_sp.X="<<Coord_sp.X<<endl;
        f <<"Coord_sp.Y =" <<Coord_sp.Y<<endl;
        f <<"Coord_sp.Z =" <<Coord_sp.Z<<endl;

```

```

        Coord_sput[0] = Coord_sp.X;
        Coord_sput[1] = Coord_sp.Y;
        Coord_sput[2] = Coord_sp.Z;
// Определение угла

        alpha = 90 - (angle(Coord_sput, Coord_user, B, L)*180/PI);
// определение видимости спутника
        vsb[i]=0;
        if ((alpha) >5)
        {
            vsb[i]=1;
            sumvsb++;
            Visibles.push_back(i); // добавление элемента в конец вектора
        }
    }

// получение матрицы Dn
    int i = 0;
    mat Dn;
    Dn.zeros(sumvsb, sumvsb);
    for (int k=1; k<=numberSput; k++)
    {
        if ((vsb[k]) == 1)
        {
            Dn(i,i) = SISerr[i].SISRE;
            i++;
        }
    }
    double max_val_Dn = Dn.max();
    for (int i= 0; i<sumvsb; i++)
    {
        if ( Dn(i,i) == 0)
        {
            Dn(i,i) = max_val_Dn;
        }
    }

// получение матрицы H
    double dx;
    double dy;
    double dz;
    double Ri;

    mat H(sumvsb, 4);
    H.zeros();

```

```

int numspu = 0;
for (int k=1; k<=numberSpu; k++)
{
    if ((vsb[k]) == 1)
    {
        timeCalc      GLNSephemTime(      almanax_GLNS[k-1].date,almanax_GLNS[k-
1].month, almanax_GLNS[k-1].year,0,0,0,0);
        Coord_sp = ephemeridsGLNS(calc.GLNS_numb_fouryear_period, //N4
                                calc.GLNS_day_after_vis_year,
                                calc.GLNS_sec_since_week,
                                GLNSephemTime.GLNS_numb_fouryear_period,
//Na берем из расчета даты альманаха
                                almanax_GLNS[k-1].tLA,
                                almanax_GLNS[k-1].dT,
                                almanax_GLNS[k-1].dT,
                                almanax_GLNS[k-1].dTT,
                                almanax_GLNS[k-1].E,
                                almanax_GLNS[k-1].w,
                                almanax_GLNS[k-1].Lam);

        dx=(Coord_sp.X-Coord_x);
        dy=(Coord_sp.Y-Coord_y);
        dz=(Coord_sp.Z- Coord_z);

// Ri = sqrt (SQUARE(dx)+SQUARE(dy)+SQUARE(dz));
        Ri = sqrt (pow(dx,2)+pow(dy,2)+pow(dz,2));
        H(numspu, 0 ) = dx/Ri;
        H(numspu, 1) = dy/Ri;
        H(numspu, 2) = dz/Ri;
        H(numspu, 3) = 1;
        numspu++;
    }
}
//для ион
/*   text1 = "BRDC1510.21n"s;

text_0 = "/MCC/BRDC/" +textYear + "/" + text_1";

const char* File11 ;
const char* file1 ;
File11 = text0.c_str();//""//перевод строки с строку Си
file1 = text1.c_str();
bool down = download( "glonass-iac.ru", NULL, NULL, File11, file1);
*/
mat Htr = H.t();

```



```

        sko = sqrt((inv(Htr*inv(Dn)*H)).t());
    }
    else
    {
        wxMessageBox(_("Выберите другую ГНСС"), _("Error"));
    }

    wxString s;
    s.Printf("Значение СКО:\nСКО для x: %.3f м\nСКО для y: %.3f м\nСКО для z:
%.3f м\nСКО для D: %.3f, м СКО: %.3f м",
        sko(0,0),        sko(1,1),        sko(2,2),        sko(3,3),        sqrt
(pow(sko(0,0),2)+pow(sko(1,1),2)+pow(sko(2,2),2) ));
    StaticText4->SetLabel(s);
    f.close();

}

void dataDialog::OnDatePickerCtrl1Changed(wxDateEvent& event)
{
}

void dataDialog::OnTimePickerCtrl1Changed(wxDateEvent& event)
{
}

void dataDialog::OnButton3Click1(wxCommandEvent& event)
{
    /*int *hour;
    int *minn;
    int *sec;

    TimePickerCtrl1->GetTime(hour, minn, sec);
    wxString s;
    s.Printf("Значение max_sats: %f\nЗначение PRN: %f\nЗначение t_almanax:
%f\nЗначение v0m0:", hour, minn, sec);
    StaticText5->SetLabel(s);
    */
}

```

Приложение 2

```

/*****
* Includes
*****/

// Подключение необходимого минимума заголовочных файлов
// Первым должен подключаться интерфейсный файл модуля

```

```

#include <stdio.h>
#include "parser.h"

/*****

* Macro Definitions

*****/

// Локальные макроопределения

/*****

* Extern Data

*****/

// Объявления экземпляров экспортируемых данных
data_t SISerr[75];

/*****

* Local Data

*****/

// Локальные объявления типов и данных

/*****

* Local Function Prototypes

*****/

// Прототипы локальных функций (без комментариев)

/*****

* Function Pointers

*****/

// Объявление указателей на функции (без комментариев)

/*****

* Function Definitions

*****/

// Реализация функций
// Сначала реализация интерфейсных функций, далее реализация локальных функций
// Все функции должны иметь описание

int parse(const char* file)
{
    int i;
    FILE* fd;
    char systype;
    char dummy[30];
    fd = fopen(file, "r");
    while ( !feof(fd) )
    {
        fscanf(fd, "%c", &systype);
        fscanf(fd, "%c", &systype);

```

```

/* if (systype !=('G' || 'R'))
{
    printf("Sync error\n");
    break;
}*/
fscanf(fd, "%d", &i); //01 123 - орбитальный слот или PRN (R01-R24 для ГЛОНАСС, G01-G32 для GPS),
либо

SISerr[i-1].systype = systype;
fscanf(fd, "%d", (int *)dummy); //02 1 - учет признака пригодности (0-только пригодные КА по данным
навигационных сообщений,
// 1 - возможно использование непригодных по данным эфемерид,
// 2 - возможно использование непригодных по данным альманахов,
// 3 - возможно использование любых непригодных эфемерид)
fscanf(fd, "%u", (unsigned int *)dummy); //03 12 - число
fscanf(fd, "%c", dummy);
fscanf(fd, "%u", (unsigned int *)dummy); //04 12 - месяц
fscanf(fd, "%c", dummy);
fscanf(fd, "%u", (unsigned int *)dummy); //05 12 - год
fscanf(fd, "%u", (unsigned int *)dummy); //06 12 - часы UTC
fscanf(fd, "%c", dummy);
fscanf(fd, "%u", (unsigned int *)dummy); //07 12 - минуты UTC
fscanf(fd, "%c", dummy);
fscanf(fd, "%u", (unsigned int *)dummy); //08 12 - секунды UTC
fscanf(fd, "%lf", (double *)dummy); //09 123.123 - длительность интервала оценки, сутки
fscanf(fd, "%u", (unsigned int *)dummy); //10 123456 - общее число обработанных точек
fscanf(fd, "%u", (unsigned int *)dummy); //11 123 - % вошедших в оценку точек R, N, B
fscanf(fd, "%u", (unsigned int *)dummy); //12 123 - % вошедших в оценку точек часы
fscanf(fd, "%u", (unsigned int *)dummy); //13 123 - % вошедших в оценку точек SISRE
fscanf(fd, "%u", (unsigned int *)dummy); //14 123 - % вошедших в оценку точек Vr, Vn, Vb
fscanf(fd, "%u", (unsigned int *)dummy); //15 123 - % вошедших в оценку точек частота
fscanf(fd, "%u", (unsigned int *)dummy); //16 123 - % вошедших в оценку точек SISVE
fscanf(fd, "%lf", (double *)dummy); //17 123.123 - МО ошибки R, м MIN ошибки R, м Med модуля
ошибки R, м
fscanf(fd, "%lf", (double *)dummy); //18 123.123 - МО ошибки N, м MIN ошибки N, м Med модуля
ошибки N, м
fscanf(fd, "%lf", (double *)dummy); //19 123.123 - МО ошибки B, м MIN ошибки B, м Med модуля
ошибки B, м
fscanf(fd, "%lf", (double *)dummy); //20 123.123 - МО ошибки часов, нс MIN ошибки часов, нс Med модуля
ошибки часов, нс
fscanf(fd, "%lf", (double *)dummy); //21 123.123 - МО ошибки SISRE, м MIN ошибки SISRE, м Med
модуля ошибки SISRE, м
fscanf(fd, "%lf", (double *)dummy); //22 123.123 - СКП R, м MAX ошибки R, м 95% модуля
ошибки R, м
fscanf(fd, "%lf", (double *)dummy); //23 123.123 - СКП N, м MAX ошибки N, м 95% модуля
ошибки N, м

```

```

fscanf(fd, "%lf", (double *)dummy);//24 123.123 - СКП В, м MAX ошибки В, м 95% модуля
ошибки В, м
fscanf(fd, "%lf", (double *)dummy);//25 123.123 - СКП часов, нс MAX ошибки часов, нс 95% модуля
ошибки часов, нс
fscanf(fd, "%lf", &(SISerr[i-1].SISRE));//26 123.123 - СКП SISRE, м MAX ошибки SISRE, м 95%
модуля ошибки SISRE, м
fscanf(fd, "%lf", (double *)dummy);//27 123.123 - МО ошибки VR, мм/с MIN ошибки VR, мм/с Med
ошибки VR, мм/с
fscanf(fd, "%lf", (double *)dummy);//28 123.123 - МО ошибки VN, мм/с MIN ошибки VN, мм/с Med
ошибки VN, мм/с
fscanf(fd, "%lf", (double *)dummy);//29 123.123 - МО ошибки VB, мм/с MIN ошибки VB, мм/с Med
ошибки VB, мм/с
fscanf(fd, "%lf", (double *)dummy);//30 123.123 - МО ошибки частоты, нс/с MIN ошибки частоты, нс Med
ошибки частоты, нс
fscanf(fd, "%lf", (double *)dummy);//31 123.123 - МО ошибки SISVE, м MIN ошибки SISVE, м Med
ошибки SISVE, мм/с
fscanf(fd, "%lf", (double *)dummy);//32 123.123 - СКП VR, мм/с MAX ошибки VR, мм/с 95% ошибки
VR, мм/с
fscanf(fd, "%lf", (double *)dummy);//33 123.123 - СКП VN, мм/с MAX ошибки VN, мм/с 95% ошибки
VN, мм/с
fscanf(fd, "%lf", (double *)dummy);//34 123.123 - СКП VB, мм/с MAX ошибки VB, мм/с 95% ошибки
VB, мм/с
fscanf(fd, "%lf", (double *)dummy);//35 123.123 - СКП частоты, нс/с MAX ошибки часов, нс/с 95% ошибки
часов, нс/с
fscanf(fd, "%lf", &(SISerr[i-1].SISVE));//36 123.123 - СКП SISVE, мм/с MAX ошибки SISVE, мм/с 95%
ошибки SISVE, мм/с
fscanf(fd, "%c", &systype);
}
int imax=i;
fclose(fd);
return imax;
}

```

Приложение 3

```

#ifndef PARSER_H
#define PARSER_H

#ifdef __cplusplus
extern "C" {
#endif

typedef struct
{

char systype;

```

```

double SISRE;
double SISVE;
} data_t;

extern data_t SISerr[75];
int parse(const char* file);

```

```

#ifdef __cplusplus
}
#endif
#endif

```

Приложение 4

```

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<CodeBlocks_project_file>
  <FileVersion major="1" minor="6" />
  <Project>
    <Option title="Data" />
    <Option pch_mode="2" />
    <Option compiler="gcc" />
    <Build>
      <Target title="Debug">
        <Option platforms="Windows;" />
        <Option output="bin/Debug/Data" prefix_auto="1" extension_auto="1" />
        <Option object_output="obj/Debug/" />
        <Option type="0" />
        <Option compiler="gcc" />
        <Option projectLinkerOptionsRelation="2" />
        <Compiler>
          <Add option="-g" />
          <Add option="-D__WXDEBUG__" />
        </Compiler>
        <Linker>
          <Add option="-static-libstdc++" />
          <Add option="-static-libgcc" />
          <Add option="-static" />
          <Add library="libwxmsw30u.a" />
          <Add library="libwxpng.a" />
          <Add library="libwxjpeg.a" />
          <Add library="libwxtiff.a" />
          <Add library="libwxzlib.a" />
          <Add library="libwininet.a" />
        </Linker>
      </Target>
    </Build>
  </Project>

```

```

<Compiler>
    <Add option="-Wall" />
    <Add option="-pipe" />
    <Add option="-mthreads" />
    <Add option="-D__GNUWIN32__" />
    <Add option="-D__WXMSW__" />
    <Add option="-DwxUSE_UNICODE" />
    <Add directory="$(#wx)/include" />
    <Add directory="$(#wx)/lib/gcc_lib/mswud" />
    <Add directory="." />
</Compiler>
<ResourceCompiler>
    <Add directory="$(#wx)/include" />
    <Add directory="$(#wx)/lib/gcc_lib/mswud" />
</ResourceCompiler>
<Linker>
    <Add option="-static-libstdc++" />
    <Add option="-static-libgcc" />
    <Add option="-static" />
    <Add option="-mthreads" />
    <Add library="libwxmsw30u.a" />
    <Add library="libwxpng.a" />
    <Add library="libwxjpeg.a" />
    <Add library="libwxtiff.a" />
    <Add library="libwxzlib.a" />
    <Add library="libkernel32.a" />
    <Add library="libuser32.a" />
    <Add library="libgdi32.a" />
    <Add library="libwinspool.a" />
    <Add library="libcomdlg32.a" />
    <Add library="libadvapi32.a" />
    <Add library="libshell32.a" />
    <Add library="libole32.a" />
    <Add library="liboleaut32.a" />
    <Add library="libuuid.a" />
    <Add library="libcomctl32.a" />
    <Add library="libwsck32.a" />
    <Add library="libodbc32.a" />
    <Add library="libuxtheme.a" />
    <Add library="libopenblas.a" />
    <Add library="libshlwapi.a" />
    <Add library="libversion.a" />
    <Add library="liboleacc.a" />
    <Add directory="$(#wx)/lib/gcc_lib" />
    <Add directory="." />
</Linker>

```

```

<Unit filename="FTPdownl.cpp" />
<Unit filename="FTPdownl.h" />
<Unit filename="angle.cpp" />
<Unit filename="angle.h" />
<Unit filename="data.cbp" />
<Unit filename="dataApp.cpp" />
<Unit filename="dataApp.h" />
<Unit filename="dataMain.cpp" />
<Unit filename="dataMain.h" />
<Unit filename="ephemerids.cpp" />
<Unit filename="ephemerids.h" />
<Unit filename="ephemeridsGLNS.cpp" />
<Unit filename="ephemeridsGLNS.h" />
<Unit filename="parser.c">
    <Option compilerVar="CC" />
</Unit>
<Unit filename="parser.h" />
<Unit filename="parserGLNS.H" />
<Unit filename="parserGLNS.c">
    <Option compilerVar="CC" />
</Unit>
<Unit filename="parserGPS.H" />
<Unit filename="parserGPS.c">
    <Option compilerVar="CC" />
</Unit>
<Unit filename="resource.rc">
    <Option compilerVar="WINDRES" />
</Unit>
<Unit filename="timeCalc.cpp" />
<Unit filename="timeCalc.h" />
<Unit filename="wxsmith/datadialog.wxs" />
<Unit filename="xyz2enu.cpp" />
<Unit filename="xyz2enu.h" />
<Extensions>
    <code_completion />
    <envvars />
    <debugger />
    <lib_finder disable_auto="1" />
    <wxsmith version="1">
        <gui name="wxWidgets" src="dataApp.cpp" main="dataDialog"
init_handlers="necessary" language="CPP" />
        <resources>
            <wxDialog wxs="wxsmith/datadialog.wxs" src="dataMain.cpp"
hdr="dataMain.h" fwddecl="0" i18n="1" name="dataDialog" language="CPP" />
        </resources>
    </wxsmith>

```

```

        </Extensions>
    </Project>
</CodeBlocks_project_file>

```

Приложение 5

```

#ifndef dataMAIN_H
#define dataMAIN_H

//(*Headers(dataDialog)
#include <wx/button.h>
#include <wx/choice.h>
#include <wx/datectrl.h>
#include <wx/dateevt.h>
#include <wx/dialog.h>
#include <wx/notebook.h>
#include <wx/sashwin.h>
#include <wx/stattext.h>
#include <wx/textctrl.h>
#include <wx/timectrl.h>
//*)
#include <wx/grid.h>
#include <wx/spinctrl.h>
//#include <wx/gdicmn.h>

class dataDialog: public wxDialog
{
public:

    dataDialog(wxWindow* parent,wxWindowID id = -1);
    virtual ~dataDialog();

private:

    //(*Handlers(dataDialog)
    void OnQuit(wxCommandEvent& event);
    void OnAbout(wxCommandEvent& event);
    void OnChoice1Select(wxCommandEvent& event);
    void OnSashWindow1SashDragged(wxSashEvent& event);
    void OnChoice1Select12(wxCommandEvent& event);
    void OnChoice1Select2(wxCommandEvent& event);
    void OnButton1Click(wxCommandEvent& event);
    void OnButton2Click(wxCommandEvent& event);
    void OnComboBox1Selected(wxCommandEvent& event);

```



```

void OnButton3Click(wxCommandEvent& event);
void OnChoice1Select1(wxCommandEvent& event);
void OnTextCtrl1Text(wxCommandEvent& event);
void OnInit(wxInitDialogEvent& event);
void OnGrid1CellLeftClick(wxGridEvent& event);
void OnChoice1Select3(wxCommandEvent& event);
void OnTextCtrl1Text1(wxCommandEvent& event);
void OnButton1Click1(wxCommandEvent& event);
void OnRichTextCtrl1Text(wxCommandEvent& event);
void OnSpinCtrl1Change(wxSpinEvent& event);
void OnCustom1Paint(wxPaintEvent& event);
void OnTextCtrlLText(wxCommandEvent& event);
void OnTextCtrlTEST1Text(wxCommandEvent& event);
void OnChoice1Select4(wxCommandEvent& event);
void OnTextCtrlZText(wxCommandEvent& event);
void OnButton1Click2(wxCommandEvent& event);
void OnDatePickerCtrl1Changed(wxDateEvent& event);
void OnButton3Click1(wxCommandEvent& event);
void OnButton3Click2(wxCommandEvent& event);
void OnTimePickerCtrl1Changed(wxDateEvent& event);
//*)

```

```

void Gridd(const char* file);
//(*Identifiers(dataDialog)
static const long ID_DATEPICKERCTRL1;
static const long ID_CHOICE1;
static const long ID_BUTTON2;
static const long ID_NOTEBOOK1;
static const long ID_BUTTON1;
static const long ID_TEXTCTRL1;
static const long ID_TEXTCTRL2;
static const long ID_TEXTCTRL3;
static const long ID_STATICTEXT1;
static const long ID_STATICTEXT2;
static const long ID_STATICTEXT3;
static const long ID_STATICTEXT4;
static const long ID_TIMEPICKERCTRL1;
static const long ID_BUTTON4;
static const long ID_STATICTEXT5;
static const long ID_SASHWINDOW1;
//*)
// static const long ID_STATICTEXT4;
static const long ID_GRID;
//(*Declarations(dataDialog)
wxButton* Button2;

```

```

wxButton* Button3;
wxButton* Down;
wxChoice* Choice1;
wxDatePickerCtrl* DatePickerCtrl1;
wxNotebook* Notebook1;
wxSashWindow* SashWindow1;
wxStaticText* StaticText1;
wxStaticText* StaticText2;
wxStaticText* StaticText3;
wxStaticText* StaticText4;
wxStaticText* StaticText5;
wxTextCtrl* TextCtrlB;
wxTextCtrl* TextCtrlH;
wxTextCtrl* TextCtrlL;
wxTimePickerCtrl* TimePickerCtrl1;
//*)

// wxStaticText* StaticText4;
wxGrid* Grid;

DECLARE_EVENT_TABLE()
};

#endif // dataMAIN_H

```

Приложение 6

```

#include <math.h>

void xyz2enu(const double lat, const double lon, double * xyz2enu)
{
    /*% *****
    %*   Copyright c 2001 The board of trustees of the Leland Stanford   *
    %*           Junior University. All rights reserved.                 *
    %*   This script file may be distributed and used freely, provided   *
    %*   this copyright notice is always kept with it.                   *
    %*           *
    %*   Questions and comments should be directed to Todd Walter at:   *
    %*   twalter@stanford.edu                                             *
    % *****33*****
    %
    %FINDXYZ2ENU find the rotation matrix to go from XYZ ECEF coordinates
    %   to a local East North Up frame
    %   [xyz2enu] = FINDXYZ2ENU(LAT, LON)
    %   LAT, LON specify the coordinates of the center of the local frame in radians

```

```

% XYZ2ENU is the rotation matrix such that DELTA_ENU = XYZ2ENU*DELTA_XYZ */
xyz2enu[0] = sin(lon);
xyz2enu[1] = cos(lon);
xyz2enu[2] = 0.0;

xyz2enu[5] = cos(lat);
xyz2enu[8] = sin(lat);

xyz2enu[3] = -xyz2enu[1] * xyz2enu[8];
xyz2enu[4] = -xyz2enu[0] * xyz2enu[8];

xyz2enu[6] = xyz2enu[1] * xyz2enu[5];
xyz2enu[7] = xyz2enu[0] * xyz2enu[5];

xyz2enu[0] = -xyz2enu[0];

}

```

Приложение 7

```

#ifndef xyz2enu_H
#define xyz2enu_H

void xyz2enu(const double lat, const double lon, double * xyz2enu);

#endif // dataAPP_H

```

Приложение 8

```

#include <windows.h>
#include <wininet.h>
#include <iostream>
#include <string>
#include <stdio.h>

#include <wx/msgdlg.h>
#include <wx/string.h>
#include <wx/textfile.h>
#include <wx/dialog.h>
#include <wx/msgdlg.h>
#include <wx/spinctrl.h>
#include <wx/intl.h>

```

```

#include <wx/settings.h>

GlonassCoordinates ephemeridsGLNS(double N4 ,
    double N,
    double ti,
    double N_A,
    double tlymbda_A,
    double dT,
    double dI,
    double dTT,
    double Ee,
    double omegaA,
    double Lam)
{
    //t0e - время на которое нужно рассчитать;
    GlonassCoordinates Coordinates;
    //1. Определяется интервал прогноза пр Δtпр в секундах:
    double dN_A;
    double Tsr = 43200; //номинальное значение периода обращения НКА, в секундах (Tsr определено в
    пинтерфейсе соответствующего сигнала)
    double isr = 63; // Из указаний
    if (N4 == 27)
        dN_A = N - N_A - (floor((N - N_A) / 1461) * 1460); //() вычисление целого, ближайшего к x
    else
        dN_A = N - N_A - (floor((N - N_A) / 1461) * 1461);
    double dtpr = dN_A * 86400 + (ti - tlymbda_A);
    //2. Рассчитывается количество целых витков W на интервале прогноза:
    double W = floor(dtpr / (Tsr + dT)); // dTa = dT ? поправка к среднему значению драконического периода обращения.
    //3 Определяется текущее наклонение:
    double i = ((isr / 180.0) + dI) * M_PI; //ΔiA = dI ? – поправка к среднему значению наклонения орбиты.
    printf("i = %0.3f c\n", i);
    //4 Определяются средний драконический период на витке W+1 и среднее движение:
    double Tdr = Tsr + dT + (2 * W + 1) * dTT; //dT'a = dTT; половинная скорость изменения драконического периода.
    printf("Tdr = %0.3f c\n", Tdr);
    double n = 2 * M_PI / Tdr;
    printf("n = %0.9f c\n", n);
    //5. Методом последовательных приближений m = 0, 1, 2... рассчитывается большая полуось орбиты a:

    double GM = 398600441.8e6; //геоцентрическая константа гравитационного поля Земли с учетом атмосферы,
    double ae = 6378136; // большая (экваториальная) полуось общеземного эллипсоида ПЗ-90
    double J02 = 1082.62575e-6; //зональный гармонический коэффициент второй степени
    double Tosk;
    double p;
    double a;
    double epsA = Ee;
    //стартовые значения
    double a_old = 0;

```

```

a = 1;
Tosk = Tdr;
p = 0;
//int ksh = 1;
/*

while (abs(a-a_old)>1e-2)
{
a_old = a;
a = pow((pow((Tosk/(2*M_PI)),2)*GM),1.0/3); //Tock = Tdr;
p = a*(1-pow(epsA,2));
Tosk = Tdr/((1-(3.0/2)*J02*pow((ae/p),2))*((2-(5.0/2)*pow(sin(i),2))*((pow(1-
pow(epsA,2),3.0/2))/(1+epsA*pow(cos(omegaA*M_PI),2)))+(
(1+epsA*pow(cos(omegaA*M_PI),3))/(1-pow(epsA,2)))));
cout << "----"<<ksh<<endl;
printf("a = %0.3f c\n", a);
printf("p = %0.3f c\n", p);
printf("Tosk = %0.3f c\n", Tosk );
ksh++;
}
*/

double eq1 = 2 - (5.0/2)*pow(sin(i),2);
double eq2 = 1 - pow(epsA,2);
double eq3 = 1 + epsA*cos(omegaA*M_PI);
double eq4 = pow(eq2,3.0/2);
double eq5 = pow(eq3,2);
double eq6 = eq4 / eq5;
double eq7 = pow(eq3,3);
double eq8 = eq7 / eq2;
double Big_div = eq1 * eq6 + eq8;
double eq9;

while (fabs(a - a_old) > 1e-5)
{
a_old = a;
a = pow((pow((Tosk/(2*M_PI)),2)*GM),1.0/3); //err
p = a*eq2;
eq9 = 1 - (3.0/2)*J02*(pow((ae/p),2));
Tosk = Tdr/(eq9*Big_div);
// ksh++;
}

//6Определяются текущие значения долготы восходящего узла орбиты и аргумента
//перигея с учетом их векового движения под влиянием сжатия Земли:
double omegaZ = 7.2921150e-5; //угловая скорость вращения Земли

double lymbda = Lam*M_PI-(omegaZ+(3.0/2)*J02*n*pow((ae/p),2)*cos(i))*dtptr;

```

```

double omega = omegaA*M_PI-(3.0/4)*J02*n*pow((ae/p),2)*(1-5.0*pow(cos(i),2))*dtp;

//7Рассчитывается значение средней долготы на момент прохождения текущего
//восходящего узла:
double E0 = -2*atan((sqrt((1-epsA)/(1+epsA)))*tan(omega/2.0));
double L1 = omega + E0 - epsA*sin(E0);

//8Определяется текущее значение средней долготы НКА:
double L = L1+n*(dtp-(Tsr+dT)*W-dTT*pow(W,2));

//10Определяется эксцентрическая аномалия путем решения уравнения Кеплера
double E = L - omega;
double Eold = 0;
while (abs(E-Eold)>1e-9)
{
    Eold = E;
    //////////////////////////////////////
    E = L-omega+ epsA*sin(E); // eps ?! !! стра 81, п10 икд
    //////////////////////////////////////
}

//11Вычисляются истинная аномалия  $\varphi$  и аргумент широты НКА u:
double v = 2*atan((sqrt((1-epsA)/(1+epsA)))*tan(E/2));
double u = v + omega;

//12Рассчитываются координаты центра масс НКА в геоцентрической
//прямоугольной пространственной системе координат:
double r = p/(1+epsA*cos(v));

Coordinates.X = r*(cos(lambda)*cos(u)-sin(lambda)*sin(u)*cos(i));
Coordinates.Y = r*(sin(lambda)*cos(u)+cos(lambda)*sin(u)*cos(i));
Coordinates.Z = r*sin(u)*sin(i);
return Coordinates;
}

```

Приложение 9

```

#ifndef EPHEMERIDSGLNS_H
#define EPHEMERIDSGLNS_H

typedef struct
{
    double X;
    double Y;

```

```

double Z;

} GlonassCoordinates;
GlonassCoordinates ephemeridsGLNS(double N4 ,//номер текущего 4х летия
double N,//N – календарный номер суток внутри четырехлетнего периода, начиная с
високосного года

//на которых находится заданный момент времени ti в секундах по шкале МДВ
double ti, //количество секунд от начала текущих суток. (берется из шкалы времени)
double N_A, //календарный номер суток по шкале МДВ внутри четырехлетнего интервала,
передаваемый НКА в составе неоперативной информации;
//будем рассчитывать в ручную исходя из строки 2 4-5-6 пункт.
double tlymbda_A, //2 строка-7 - время прохождения первого узла, на которое все дано, с
double dT, //поправка к драконическому периоду, с
double dI,
double dTT,
double Ee, // эксцентриситет
double omegaA, //аргумент перигей )
double Lam); // долгота узла, полуциклы, она же лямбда А

#endif

```

Приложение 10

```

#include "angle.h"
#include "xyz2enu.h"

#define _USE_MATH_DEFINES
#include <math.h>

#define SQUARE(val) val * val

double angle(double Coord_sput[3], double Coord[3], double B, double L)
{
double ES[3];
double Renu[3];
double R[3];
double lengthES;
double C[9];
double a;

ES[0]=(Coord_sput[0]-Coord[0]);
ES[1]=(Coord_sput[1]-Coord[1]);
ES[2]=(Coord_sput[2]- Coord[2]);

```

```

lengthES = sqrt (pow((Coord_sput[0]- Coord[0]),2)+pow((Coord_sput[1]-Coord[1]),2)+pow((Coord_sput[2]-
Coord[2]),2));
//lengthES = sqrt (SQUARE(Coord_sput[0]- Coord[0])+SQUARE(Coord_sput[1]-
Coord[1])+SQUARE(Coord_sput[2]- Coord[2]));
R[0]= ES[0]/lengthES; // вектор столбец
R[1]= ES[1]/lengthES;
R[2]= ES[2]/lengthES;

xyz2enu(B, L, C); // получение матрицы ENU в -> C

Renu[0]=C[0]*R[0]+C[1]*R[1]+C[2]*R[2]; //вектор столбец
Renu[1]=C[3]*R[0]+C[4]*R[1]+C[5]*R[2];
Renu[2]=C[6]*R[0]+C[7]*R[1]+C[8]*R[2];

a = acos (Renu[2]); // rad
return a;
}

```

Приложение 11

```

// Начнем с директив препроцессора. ADD_H – это произвольное уникальное имя (обычно используется имя
заголовочного файла)
#ifndef ANGLE_H
#define ANGLE_H

double angle(double Coord_sput[3],double Coord[3], double B, double L);

#endif

```

Приложение 12

```

<?xml version="1.0" encoding="utf-8" ?>
<wxsmith>
    <object class="wxDialog" name="dataDialog">
        <title>Data app</title>
        <size>533,556</size>
        <bg>wxSYS_COLOUR_INACTIVEBORDER</bg>
        <minsize>-1,-1</minsize>
        <maxsize>-1,-1</maxsize>
        <id_arg>0</id_arg>
        <handler function="OnInit" entry="EVT_INIT_DIALOG" />
        <object class="wxSashWindow" name="ID_SASHWINDOW1" variable="SashWindow1"
member="yes">
            <pos>56,40</pos>

```



```

<size>480,504</size>
<fg>wxSYS_COLOUR_WINDOWTEXT</fg>
<bg>wxSYS_COLOUR_MENUBAR</bg>
<handler function="OnSashWindow1SashDragged" entry="EVT_SASH_DRAGGED" />
<object          class="wxDatePickerCtrl"          name="ID_DATEPICKERCTRL1"
variable="DatePickerCtrl1" member="yes">
    <pos>305,65</pos>
    <size>85,21</size>
    <fg>wxSYS_COLOUR_ACTIVEBORDER</fg>
    <bg>wxSYS_COLOUR_HIGHLIGHT</bg>
    <handler
entry="EVT_DATE_CHANGED" />
function="OnDatePickerCtrl1Changed"

</object>
<object class="wxChoice" name="ID_CHOICE1" variable="Choice1" member="yes">
    <content>
        <item>Glonass</item>
        <item>GPS</item>
        <item>Galileo</item>
        <item>Beidou</item>
        <item>QZSS</item>
    </content>
    <selection>0</selection>
    <pos>33,16</pos>
    <size>244,21</size>
</object>
<object class="wxButton" name="ID_BUTTON2" variable="Button2" member="yes">
    <label>Загрузить</label>
    <pos>305,16</pos>
    <size>127,23</size>
    <handler function="OnButton2Click" entry="EVT_BUTTON" />
</object>
<object  class="wxNotebook"  name="ID_NOTEBOOK1"  variable="Notebook1"
member="yes">
    <pos>124,214</pos>
</object>
<object class="wxButton" name="ID_BUTTON1" variable="Down" member="yes">
    <label>Обработка</label>
    <pos>305,266</pos>
    <size>127,23</size>
    <handler function="OnButton1Click1" entry="EVT_BUTTON" />
</object>
<object  class="wxTextCtrl"  name="ID_TEXTCTRL1"  variable="TextCtrlH"
member="yes">
    <value>0</value>
    <pos>305,145</pos>
    <size>127,-1</size>

```

```

        <handler function="OnTextCtrl1Text1" entry="EVT_TEXT" />
    </object>
    <object class="wxTextCtrl" name="ID_TEXTCTRL2" variable="TextCtrlB"
member="yes">
        <value>0</value>
        <pos>305,190</pos>
        <size>127,-1</size>
    </object>
    <object class="wxTextCtrl" name="ID_TEXTCTRL3" variable="TextCtrlL"
member="yes">
        <value>0</value>
        <pos>305,235</pos>
        <size>127,-1</size>
    </object>
    <object class="wxStaticText" name="ID_STATICTEXT1" variable="StaticText1"
member="yes">
        <label>Введите значение высоты в метрах:</label>
        <pos>305,127</pos>
    </object>
    <object class="wxStaticText" name="ID_STATICTEXT2" variable="StaticText2"
member="yes">
        <label>Введите значение В в градусах:</label>
        <pos>305,172</pos>
    </object>
    <object class="wxStaticText" name="ID_STATICTEXT3" variable="StaticText3"
member="yes">
        <label>Введите значение L в градусах:</label>
        <pos>305,217</pos>
    </object>
    <object class="wxStaticText" name="ID_STATICTEXT4" variable="StaticText4"
member="yes">
        <label>Значения СКО:</label>
        <pos>306,294</pos>
    </object>
    <object class="wxTimePickerCtrl" name="ID_TIMEPICKERCTRL1"
variable="TimePickerCtrl1" member="yes">
        <pos>305,93</pos>
        <size>85,21</size>
        <handler function="OnTimePickerCtrl1Changed"
entry="EVT_DATE_CHANGED" />
    </object>
    <object class="wxButton" name="ID_BUTTON4" variable="Button3" member="yes">
        <label>для отладки/test</label>
        <pos>303,463</pos>
        <handler function="OnButton3Click2" entry="EVT_BUTTON" />
    </object>

```

```

member="yes">
        <object class="wxStaticText" name="ID_STATICTEXT5" variable="StaticText5"
        <label>Время прогнозирования по UTC(+3)</label>
        <pos>305,49</pos>
        </object>
    </object>
</wxsmith>

```

Приложение 13

```

#include "timeCalc.h"
#include <iostream>
#define _USE_MATH_DEFINES
#include <math.h>
#include <ctime>

```

```

timeCalc::timeCalc(int date,
    int month,
    int year,
    int hour,
    int minutes,
    int sec,
    int m_sec)
{
    c_date = date;
    c_month= month;
    c_year = year;
    c_hour = hour;
    c_minutes = minutes;
    c_sec = sec;
    c_m_sec = m_sec;
    tm tm;
    tm.tm_year=year-1900;
    tm.tm_mon = month-1;
    tm.tm_mday = date;
    tm.tm_hour = hour;
    tm.tm_min=minutes;
    tm.tm_sec=sec;
    time_t time = mktime(&tm);
    timeSec =time;
    int dt=0;
    if ((tm.tm_year) > 81)
        dt++;
    if ((tm.tm_year) > 82)

```

```

dt++;
if ((tm.tm_year) > 83)
dt++;
if ((tm.tm_year) > 85)
dt++;
if ((tm.tm_year) > 87)
dt++;
if ((tm.tm_year) > 89)
dt++;
if ((tm.tm_year) > 90)
dt++;
if ((tm.tm_year) > 92)
dt++;
if ((tm.tm_year) > 93)
dt++;
if ((tm.tm_year) > 94)
dt++;
if ((tm.tm_year) > 95)
dt++;
if ((tm.tm_year) > 97)
dt++;
if ((tm.tm_year) > 98)
dt++;
if ((tm.tm_year) > 105)
dt++;
if ((tm.tm_year) > 108)
dt++;
if ((tm.tm_year) > 112)
dt++;
if ((tm.tm_year) > 115)
dt++;
if ((tm.tm_year) > 116)
dt++;
dT = dt;
}
timeCalc::~timeCalc()
{

}

void timeCalc::timeGLNS()
{
tm tmm;
tmm.tm_year=c_year-1900;
tmm.tm_mon = c_month-1;
tmm.tm_mday = c_date;
tmm.tm_hour = c_hour;

```

```

tmm.tm_min=c_minutes;
tmm.tm_sec=c_sec;
tm GlonassVis;
tm tmGlonass;
tmGlonass.tm_year=96;//год (1900 год = 0)
tmGlonass.tm_mon = 0; // месяц года (январь = 0) [0,11]
tmGlonass.tm_mday = 1;// день месяца [1,31]
tmGlonass.tm_hour = 0; // часы после полуночи [0,23]
tmGlonass.tm_min=0; //минуты после часов [0,59]
tmGlonass.tm_sec=0; // секунды после минут [0,59]
timee glonass;
tmm.tm_hour = tmm.tm_hour+3;
time_t time22 = mktime(&tmm);
time_t timebaseGlonass = mktime(&tmGlonass);
double timeGlonass = time22 - timebaseGlonass;
glonass.numb_fouryear_period = (static_cast<double>(tmm.tm_year) - static_cast<double>(tmGlonass.tm_year))/4;
int deltayear = tmm.tm_year%4;

GlonassVis.tm_year = tmm.tm_year - deltayear;
GlonassVis.tm_mon = 0; // месяц года (январь = 0) [0,11]
GlonassVis.tm_mday = 1;// день месяца [1,31]
GlonassVis.tm_hour = 0; // часы после полуночи [0,23]
GlonassVis.tm_min=0; //минуты после часов [0,59]
GlonassVis.tm_sec=0; // секунды после минут [0,59]
time_t time_afet_vis_year = mktime(&GlonassVis);
glonass.delta_day_after_vis_year = time22 - time_afet_vis_year;
glonass.day_after_vis_year =glonass.delta_day_after_vis_year/(60*60*24); //дней после висок года .. учесть 1 дей.
glonass.week = timeGlonass/(60*60*24*7);
glonass.sec_after_week=fmod(timeGlonass,(60*60*24*7));
glonass.sum_sec = tmm.tm_hour*60*60 + tmm.tm_min*60 + tmm.tm_sec;
GLNS_sec_since_week = glonass.sum_sec;
GLNS_numb_fouryear_period = ceil(glonass.numb_fouryear_period); //N4 -for GLONASS ;
GLNS_day_after_vis_year = ceil(glonass.day_after_vis_year) ;
}

void timeCalc::timeGPS()
{
tm tmgps;
tmgps.tm_year=80;//год (1900 год = 0)
tmgps.tm_mon = 0; // месяц года (январь = 0) [0,11]
tmgps.tm_mday = 6;// день месяца [1,31]
tmgps.tm_hour = 0; // часы после полуночи [0,23]
tmgps.tm_min=0; //минуты после часов [0,59]
tmgps.tm_sec=0; // секунды после минут [0,59]
time_t timebase = mktime(&tmgps);
timee gps;

```

```

double time2 = timeSec;
gps.sec = (time2 - timebase)+dT;
gps.week = gps.sec/(60*60*24*7);
//gps.sec_after_week = static_cast<int>(round(gps.sec))%(60*60*24*7);
gps.sec_after_week=fmod(gps.sec,(60*60*24*7));

sec_since_week = (gps.sec_after_week);
week = static_cast<int>(gps.week);
}

void timeCalc::timeGGL()
{
tm tmgalileo;
tmgalileo.tm_year=99;//год (1900 год = 0)
tmgalileo.tm_mon = 7; // месяц года (январь = 0) [0,11]
tmgalileo.tm_mday = 22;// день месяца [1,31]
tmgalileo.tm_hour = 0; // часы после полуночи [0,23]
tmgalileo.tm_min=0; //минуты после часов [0,59]
tmgalileo.tm_sec=0; // секунды после минут [0,59]
timee galileo;
time_t timebasegalileo = mktime(&tmgalileo);
double time2 = timeSec;
galileo.sec = (time2 - timebasegalileo)+dT;
galileo.week = galileo.sec/(60*60*24*7);
galileo.sec_after_week = fmod(galileo.sec,(60*60*24*7));
week = static_cast<int>(galileo.week);
sec_since_week = galileo.sec_after_week;
}

```

Приложение 14

```

#ifndef TIMECALC_H
#define TIMECALC_H

struct timee{
double sec;
double week;
double sec_after_week;
double day_after_vis_year;
double delta_day_after_vis_year ;
double sum_sec;
double numb_fouryear_period;
double sec_after_week_plus_delta;
};

class timeCalc
{

```

```

public:
    timeCalc(int date,
        int month,
        int year,
        int hour,
        int minutes,
        int sec,
        int m_sec ); // msec пока не используется.
    ~timeCalc(); // дописать
    int dT; // поправки ко времени
    int sec_since_week; //GPS,GALILEO
    int week;//GPS,GALILEO
    int GLNS_sec_since_week;
    int GLNS_numb_fouryear_period; //N4 -for GLONASS ;
    int GLNS_day_after_vis_year;//NT - for GLONASS

    void timeGLNS();
    void timeGPS();
    void timeGLL();
    double timeSec;
    int c_date;
    int c_month;
    int c_year; //в формате 2015
    int c_hour;
    int c_minutes;
    int c_sec;
    int c_m_sec;
protected:
    /*double timeSec;
    int c_date;
    int c_month;
    int c_year; //в формате 2015
    int c_hour;
    int c_minutes;
    int c_sec;
    int c_m_sec;*/
};

#endif

```

Приложение 15

```

#include "FTPdownl.h"
#include <windows.h>
#include <wininet.h>

```

```

#include <iostream>
#include <stdio.h>

bool download(LPCSTR server, LPCSTR login, LPCSTR pass, LPCSTR local_file, LPCSTR remote_file)
{
    bool status;
    HINTERNET hOpen, hConnection;

    hOpen = InternetOpen(NULL, INTERNET_OPEN_TYPE_DIRECT, NULL, NULL, 0);
    if (hOpen == NULL)
        return false;

    hConnection = InternetConnectA(hOpen, server, 21, login, pass, INTERNET_SERVICE_FTP,
INTERNET_FLAG_PASSIVE, 0);
    if (hConnection == NULL)
    {
        InternetCloseHandle(hOpen);
        return false;
    }

    status=FtpGetFileA(hConnection, local_file, remote_file, true, 0, FTP_TRANSFER_TYPE_UNKNOWN, 0);
    InternetCloseHandle(hConnection);
    InternetCloseHandle(hOpen);
    return status;
}

```

Приложение 16

```

#include <windows.h>

#ifdef FTPdonwl_H
#define FTPdonwl_H

bool download(LPCSTR server, LPCSTR login, LPCSTR pass, LPCSTR local_file, LPCSTR remote_file);

#endif

```

Приложение 17

```

/*****
* \file filename.c
* \brief Краткое описание, назначение
* \remarks Необязательная секция, для комментариев

```



```

* \author Автор
*
* \if use_svn_keywords
* ::                $: Revision of last commit
* ::                $: Date of last commit
* ::                $: Author of last commit
* \endif
*
* \b LICENSE \b INFORMATION \n
* Copyright (c) Year, Company, City, Country
*****/

/*****

* Includes
*****/

// Подключение необходимого минимума заголовочных файлов
// Первым должен подключаться интерфейсный файл модуля
#include <stdio.h>
#include "parserGLNS.H"

// #include "parserGPS.h"

/*****

* Macro Definitions
*****/

// Локальные макроопределения

/*****

* Extern Data
*****/

// Объявления экземпляров экспортируемых данных
data_almanax_GLNS almanax_GLNS[75];

/*****

* Local Data
*****/

// Локальные объявления типов и данных

/*****

* Local Function Prototypes
*****/

// Прототипы локальных функций (без комментариев)

/*****

* Function Pointers
*****/

// Объявление указателей на функции (без комментариев)

```

```

/*****

* Function Definitions

*****/

// Реализация функций
// Сначала реализация интерфейсных функций, далее реализация локальных функций
// Все функции должны иметь описание

/*****

* END OF FILE

*****/

int parseGLNS(const char* file)
{
    int i;
    int Numbb;
    char systype;
    char dummy[50];
    FILE *fd;
    // string file = "MCCJ_150306.AGP";
    fd = fopen(file, "r");
    //fd = fopen("test.txt", "r");
    if (fd == NULL)
    {
        printf("ERORR");
    }
    else{ printf("OK"); }
    while ( !feof(fd) )           // пока не конец файла
    {
        // almanax_GLNS[i-1].systype = systype;

//Строка 1
fscanf(fd, "%u", (int *)dummy); //1 - число
nhttps://vk.com/doc165113148_595394058?hash=f2e6d67d96e5f1a609&dl=ab5ddaf49d0048c0feолучения альманаха
fscanf(fd, "%u", (int *)dummy); //2 - месяц получения альманаха
fscanf(fd, "%u", (int *)dummy); //3 - год получения альманаха
fscanf(fd, "%u", &Numbb); //4 -

//Строка 2
fscanf(fd, "%u", &(i)); //1 - номер PRN
almanax_GLNS[i-1].PRN=i;
almanax_GLNS[i-1].NN = Numbb;
fscanf(fd, "%u", (int *)dummy); //2 - номер частотного слота (-7 - 24)
fscanf(fd, "%u", (int *)dummy); //3 - признак здоровья по альманаху (0 - 1)
fscanf(fd, "%u", &(almanax_GLNS[i-1].date)); //4 - число

```

```

fscanf(fd, "%u", &(almanax_GLNS[i-1].month)); //5 - месяц
fscanf(fd, "%u", &(almanax_GLNS[i-1].year)); //6 - год
fscanf(fd, "%lf", &(almanax_GLNS[i-1].tLA)); //7 - время прохождения первого узла, на которое все дано, с
fscanf(fd, "%lf", (int *)dummy); //8 - поправка ГЛОНАСС-UTC, с
fscanf(fd, "%lf", (int *)dummy); //9- поправка GPS-ГЛОНАСС, с
fscanf(fd, "%lf", (int *)dummy); //10 - поправка времени КА ГЛОНАСС относительно системного времени, с

//Строка 3
fscanf(fd, "%lf", &( almanax_GLNS[i-1].Lam )); //1 - Lam - долгота узла, полуциклы
fscanf(fd, "%lf", &( almanax_GLNS[i-1].dI)); //2 - dI - коррекция наклона, полуциклы
fscanf(fd, "%lf", &( almanax_GLNS[i-1].w)); //3 - w - аргумент перигея, полуциклы
fscanf(fd, "%lf", &( almanax_GLNS[i-1].E)); //4 - E - эксцентриситет
fscanf(fd, "%lf", &( almanax_GLNS[i-1].dT)); //5 - dT - поправка к драконическому периоду, с
fscanf(fd, "%lf", &( almanax_GLNS[i-1].dTt)); //6 - dTt - поправка к драконическому периоду, с/виток

almanax_GLNS[i-1].PRN=i;

}
int imax=i;
almanax_GLNS[0].mass = imax;
fclose(fd);

//printf("num_week = %d ", almanax_GPS[1].num_week);

return imax;
}

```

Приложение 18

```

#ifndef PARSE_RGLNS_H
#define PARSE_RGLNS_H

#ifdef __cplusplus
extern "C" {
#endif

typedef struct
{
    int NN;

```

```

char systype;
int date;
int month;
int year;
double t_almanax;
unsigned int PRN;
double tLA; // строка 2 -п.7
double Lam ;
double dI;
double w;
double E;
double dT;
double dTT;

int mass;
} data_almanax_GLNS;

extern data_almanax_GLNS almanax_GLNS[75];
int parseGLNS(const char* file);

#ifdef __cplusplus
}
#endif
#endif

```

Приложение 19

```

/*****
* \file filename.c
* \brief Краткое описание, назначение
* \remarks Необязательная секция, для комментариев
* \author Автор
*
* \if use_svn_keywords
* ::                $: Revision of last commit
* ::                $: Date of last commit
* ::                $: Author of last commit
* \endif
*
* \b LICENSE \b INFORMATION \n
* Copyright (c) Year, Company, City, Country
*****/

```

```

/*****

* Includes

*****/

// Подключение необходимого минимума заголовочных файлов
// Первым должен подключаться интерфейсный файл модуля
#include <stdio.h>
#include "parserGPS.H"

// #include "parserGPS.h"

/*****

* Macro Definitions

*****/

// Локальные макроопределения

/*****

* Extern Data

*****/

// Объявления экземпляров экспортируемых данных
data_almanax almanax_GPS[75];

/*****

* Local Data

*****/

// Локальные объявления типов и данных

/*****

* Local Function Prototypes

*****/

// Прототипы локальных функций (без комментариев)

/*****

* Function Pointers

*****/

// Объявление указателей на функции (без комментариев)

/*****

* Function Definitions

*****/

// Реализация функций
// Сначала реализация интерфейсных функций, далее реализация локальных функций
// Все функции должны иметь описание

/*****

* END OF FILE

*****/

```

```

int parseGPS(const char* file)
{
    int i;
    int Numbb;
    char systype;
    char dummy[50];
    FILE *fd;
    // string file = "MCCJ_150306.AGP";
    fd = fopen(file, "r");
    //fd = fopen("test.txt", "r");
    if (fd == NULL)
    {
        printf("ERORR");
    }
    else{ printf("OK"); }
    while ( !feof(fd) )           // пока не конец файла
    {
        // almanax_GPS[i-1].systype = systype;

//Строка 1
fscanf(fd, "%u", (int *)dummy); //1 - число получения альманаха
fscanf(fd, "%u", (int *)dummy); //2 - месяц получения альманаха
fscanf(fd, "%u", (int *)dummy); //3 - год получения альманаха
fscanf(fd, "%u", &Numbb); //4 -

//Строка 2
fscanf(fd, "%u", &(i)); //1 - номер PRN
almanax_GPS[i-1].PRN=i;
almanax_GPS[i-1].NN = Numbb;
fscanf(fd, "%u", (int *)dummy); //2 - обобщенный признак здоровья
fscanf(fd, "%u", &( almanax_GPS[i-1].num_week)); //3 - неделя GPS (альманаха)
fscanf(fd, "%u", &( almanax_GPS[i-1].time_week)); //4 - время недели GPS, с (альманаха)
fscanf(fd, "%u", (int *)dummy); //5 - число
fscanf(fd, "%u", (int *)dummy); //6 - месяц
fscanf(fd, "%u", (int *)dummy); //7 - год
fscanf(fd, "%lf", &( almanax_GPS[i-1].t_almanax)); //8 - время альманаха, с
fscanf(fd, "%lf", (int *)dummy); //9 - поправка времени КА GPS относительно системного времени, с
fscanf(fd, "%lf", (int *)dummy); //10- скорость поправки времени КА GPS относительно системного времени, с/с
fscanf(fd, "%lf", &( almanax_GPS[i-1].vOm0)); //11- Om0 - скорость долготы узла, полуциклы/с

//Строка 3
fscanf(fd, "%lf", &( almanax_GPS[i-1].Om0)); //1 - Om0 - долгота узла, полуциклы
fscanf(fd, "%lf", &( almanax_GPS[i-1].I)); //2 - I - наклонение, полуциклы
fscanf(fd, "%lf", &( almanax_GPS[i-1].w)); //3 - w - аргумент перигея, полуциклы
fscanf(fd, "%lf", &( almanax_GPS[i-1].E)); //4 - E - эксцентриситет

```

```

fscanf(fd, "%lf", &( almanax_GPS[i-1].sqrtA)); //5 - SQRT(A) - корень из большой полуоси, м**0.5
fscanf(fd, "%lf", &( almanax_GPS[i-1].M0)); //6 - M0 - средняя аномалия, полуциклы

almanax_GPS[i-1].PRN=i;

}

int imax=i;
almanax_GPS[0].mass = imax;
fclose(fd);

//printf("num_week = %d ", almanax_GPS[1].num_week);

return imax;
}

```

Приложение 20

```

#ifndef PARSEGPS_H
#define PARSEGPS_H

#ifdef __cplusplus
extern "C" {
#endif

typedef struct
{
    int NN;
    char systype;
    int num_week;
    int time_week;
    double t_almanax;

    // double dt;
    // double vt;
    unsigned int PRN;
    double vOm0;
    double Om0;
    double I;
    double w;
    double E;
    double sqrtA;
    double M0;

```

```

    int mass;
} data_almanax;

extern data_almanax almanax_GPS[75];
int parseGPS(const char* file);

#ifdef __cplusplus
}
#endif
#endif

```

Приложение 21

```

#include "ephemerids.h"

#define _USE_MATH_DEFINES
#include <math.h>

#include <windows.h>
#include <wininet.h>
#include <iostream>
#include <string>
#include <stdio.h>

#include <wx/msgdlg.h>
#include <wx/string.h>
#include <wx/textfile.h>
#include <wx/dialog.h>
#include <wx/msgdlg.h>
#include <wx/spinctrl.h>
#include <wx/intl.h>
#include <wx/settings.h>

Coordinates ephemerids(double toe,
    double t_almanax,
    double almanax_M0,
    double sqrtA,
    double E,
    double I,
    double Om0,
    double time_week )

```



```

{//toe - время на которое нужно рассчитать;
Coordinates Coordinates;

double M0 = almanax_M0*M_PI;
double A0 = pow((sqrt(A),2);
double en = E;
double omegan = 0.16578805*M_PI;
double i0n = I*M_PI;
double Omega0n = Om0*M_PI;
double nu = 3.986005*pow(10,14); //из икд
double OMEGA_REF=(-2.6*pow(10,-9))*M_PI;//из икд
double OMEGA_e=7.2921151467*pow(10,-5);//из икд
int week_almanax = time_week;
double t_sec_almanax = t_almanax;
double tk = toe- t_sec_almanax;
if (tk > 302400)
{
tk = 604800-tk;
}
else if (tk<-302400)
{
tk = 604800+tk;
}
double n0 = sqrt(nu/pow(A0,3));
double nA = n0;
double Mk = M0 + nA*tk;
double Ek;
double Ekold = 0;
Ek = en*sin(0)+Mk;
while (abs(Ek- Ekold)>0.000000001 )
{ Ekold = Ek;
Ek = en*sin(Ek)+Mk;
}

double vk = atan2(sqrt(1-(pow(en,2)))*sin(Ek), (cos(Ek)-en) );
double Ak = A0;
double rk = Ak*(1-en*cos(Ek));
double Fk = vk + omegan;
double uk = Fk;
double xkk = rk*cos(uk);
double ykk = rk*sin(uk);
double OMEGA = OMEGA_REF;
double OMEGAk = Omega0n+(OMEGA-OMEGA_e)*tk-OMEGA_e*t_sec_almanax;
double ik = i0n;
double xk = xkk*cos(OMEGAk)-ykk*cos(ik)*sin(OMEGAk);
double yk = xkk*sin(OMEGAk)+ykk*cos(ik)*cos(OMEGAk);

```

```

double zk = ykk*sin(ik);

Coordinates.X = xkk*cos(OMEGAk)-ykk*cos(ik)*sin(OMEGAk);
Coordinates.Y = xkk*sin(OMEGAk)+ykk*cos(ik)*cos(OMEGAk);
Coordinates.Z = ykk*sin(ik);
return Coordinates;
}

```

Приложение 22

```

#ifndef EPHEMERIDS_H
#define EPHEMERIDS_H

typedef struct
{
    double X;
    double Y;
    double Z;

} Coordinates;

Coordinates ephemerids(double toe,
                        double t_almanax,
                        double almanax_M0,
                        double sqrtA,
                        double E,
                        double I,
                        double Om0,
                        double time_week );

#endif

```