Министерство образования и науки Российской Федерации
КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ФИЗИКИ
КАФЕДРА РАДИОЭЛЕКТРОНИКИ

Специальность: 011800.62 — радиофизика

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(Бакалаврская работа)

ДОРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ ФОТОМЕТРОМ ДЛЯ НАВЕДЕНИЯ В ЭКВАТОРИАЛЬНЫХ КООРДИНАТАХ

Работа завершена:

"___"_____ 2016 г. _____ (А. С. Смолин)

Работа допущена к защите:

Научный руководитель
К. ф – м. н., доцент
"___"_____ 2016 г. _____ (Р. И. Гумеров)

Заведующий кафедрой
Д. ф.-м. н., профессор
"___"_____2016 г. _____ (М. Н. Овчинников)

Казань – 2016

Оглавление

Введение

Одна из самых приоритетных и развивающихся задач современной астрономии сегодня – это автоматизация обсерваторий. Роботизированные астрономические комплексы способны резко поднять производительность телескопов (другой способ добиться этого – увеличить их поле зрения). Поскольку имеющаяся в наличии монтировкоа Synta Sky-Watcher EQ6 Pro в штатном режиме не имет системы управления в режиме удалённого доступа, в первом цикле доработки было разработано программное обеспечение для дистанционного управления. Использовалась же данная система для наблюдения свечения неба, и наведение производилось в горизонтальной системе координат. Сейчас планируется использовать её для наблюдения за объектами ближнего космоса, поэтому имеет смысл доработать имеющееся ПО и реализовать систему наведения в экваториальных координатах, учитывающую распространённые ошибки наведения. Такие ошибки возникают из-за влияния рефракции, гнутия осей монтировок, неточности установки оборудования и т.д. Поэтому, в рамках квалификационной работы, научным руководителем была поставлена задача реализации такой системы наведения и добавления её в уже существующее программное обеспечение для управления монтировкой. Таким образом, для достижения цели было необходимо решить следующий ряд задач:

1. Освоить язык программирования общего назначения C++ в объёме, необходимом для реализации данного проекта, а также интегрированную среду разработки (IDE) Microsoft Visual Studio 2010;
2. Освоить методику написания графических dialog-based программ на основе встроенной в IDE библиотеки классов MSC;
3. Изучить монтировку Synta Sky-Watcher EQ6 Pro и освоить принцип управления ею;
4. Изучить системы координат, использующиеся для наведения телескопов (в частности – экваториальную с.к.);
5. Изучить ПО, разработанное в первом цикле доработки;

6. Разработать программную реализацию системы наведения по экваториальным координатам, учитывающую ошибки наведения и интегрировать в уже имеющееся программное обеспечение.

ГЛАВА 1. МОНТИРОВКИ ТЕЛЕСКОПОВ

1.1 Типы монтировок телескопов

Монтировка телескопа (или опорно-поворотное устройство телескопа) – поворотная опора приборов для наблюдения за небесными объектами (например, телескопов или астрографов). Для того, что навести трубу телескопа на нужную звезду, необходимы две степени свободы (количество степеней свободы определяется по количеству координат, которые определяют видимое положение звезды). Их обеспечивает так называемая монтировка телескопа. Упрощённо любую монтировку можно представить, как систему из трёх элементов:

1. Основание, которое несёт на себе первую ось, несущую вторую;
2. Вторая ось, расположенная перпендикулярно первой оси;
3. Труба телескопа, прикреплённая ко второй оси.

Принципиально монтировки делят по ориентированности их осей. Наиболее распространёнными яввляются *экваториальные* (или параллактические монтировки). Впервые предложены Х. Шейнером в 1630 г. В них оси ориентированы следующим образом: первая (*полярная* или *часовая tt*) ось направлена в видимый полюс мира Р’, а вторая (*ось склонений δδ),* перпендикулярная первой, находится в плоскости небесного экватора. Это позволяет скомпенсировать среднесуточное вращение Земли и, поворачивая телескоп вокруг одной лишь полярной оси, отслеживать суточное движение звезды. Эти монтировки в свою очередь также делятся на несколько типов. Например, в английской монтировке полярная ось устанавливается на две колонны, а сам телескоп поддерживается подшипником оси склонения. Немецкая же монтировка устроена таким образом, что её полярная ось несёт корпус с расположенной в нём осью склонения. Такие монтировки хорошо подходят для установки на них лёгких труб рефракторов. В американской монтировке на конце оси размещена вилка, несущая ось склонений. В

рамках данной работы используется экваториальная монтировка немецкого типа, однако стоит упомянуть и другие их типы.

До изобретения экваториальной монтировки использовались *альт-азимунтальные* монтировки, в которых первая вертикальная ось (ось азимутов АА) направлена в зенит Z, а вторая (ось зенитных расстояний zz) располагается в плоскости горизонта. Эти монтировки проще, легче и компактнее экваториальных, а также значительно дешевле, однако компенсировать среднесуточное вращение Земли при использовании их значительно сложнее, нежели при использовании экваториальных монтировок. Этот тип монтировок до сих пор используется для установки на них любительских телескопов. Также альт-азимутальные монтировки используются в случаях, когда используемый телеском имеет большую массу и установить его на экваториальную не представляется возможным.

Следующий тип – это *горизонтальная монтировка* (или альт-альт монтировка). Используется исключительно для узко специализированных телескопов (пример: только спектральные работы). В ней одна ось расположена в плоскости горизонта в направлении с востока на запад (реже – с севера на юг), вторая же – ей перпендикулярна.

Последний тип – *трёх- и четырёхосные* монтировки. Используются для телескопов, предназначенных для наблюдения за объектами с большой скоростью движения (движение по небу искусственных спутников Земли). С их помощью можно направить третью (орбитальную) ось ОО в видимый полюс P_0 орбиты спутника и, поворачивая телескоп вокруг этой оси, отслеживать движение наблюдаемого объекта.

На рисунке 1.1.1 представлены монтировки перечисленных выше четырёх типов.

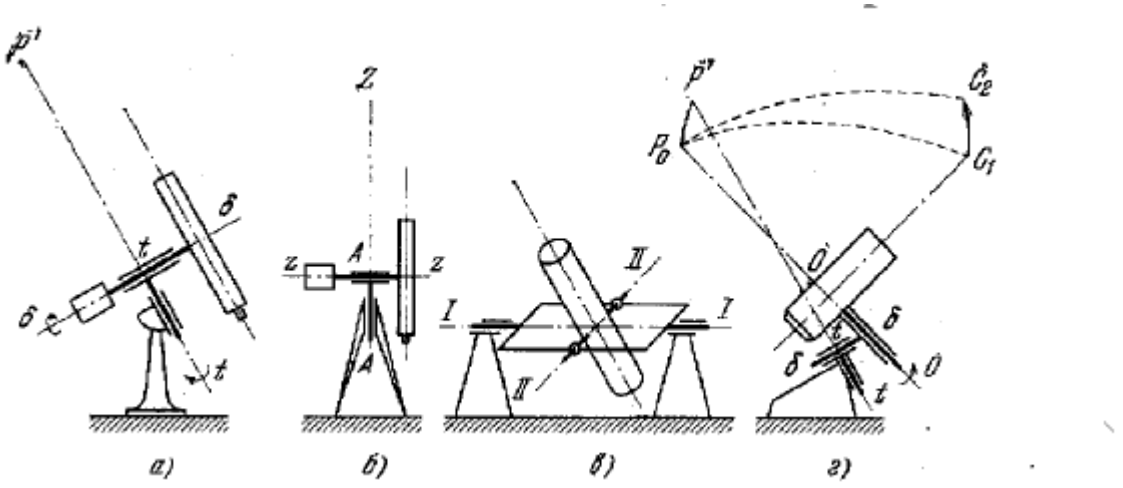


Рисунок 1.1.1 - Основные принципиальные типы монтировок

На рисунке 1.1.1 монтировки:

а – экваториальная немецкого типа (Р – видимый полюс мира);

б – альт-азимутальная (Z - зенит);

в – горизонтальная (альт-альт);

г – трёхосная (P_0 – полюс орбиты искусственного спутника Земли, C_1C_2 – его видимая траектория, ОО – орбитальная ось).

Также монтировки делятся на два типа исходя из их конструкции – это *симметричные* и *несимметричные* монтировки. Симметричные монтировки сконтструированны таким образом, что полярная ось, ось склонений и оптическая ось пересечены в одной точке. В несимметричных же оптическая ось и ось склонений пересекаются в точке, вынесенной в сторону от полярной оси. Из-за этого, чтобы уравновесить трубу телескопа, необходимо использовать противовес, что является недостатком монтировок такого рода. Симметричные и несимметричные монтировки же делятся на множество подтипов. Они представлены на рисунке 1.1.2.

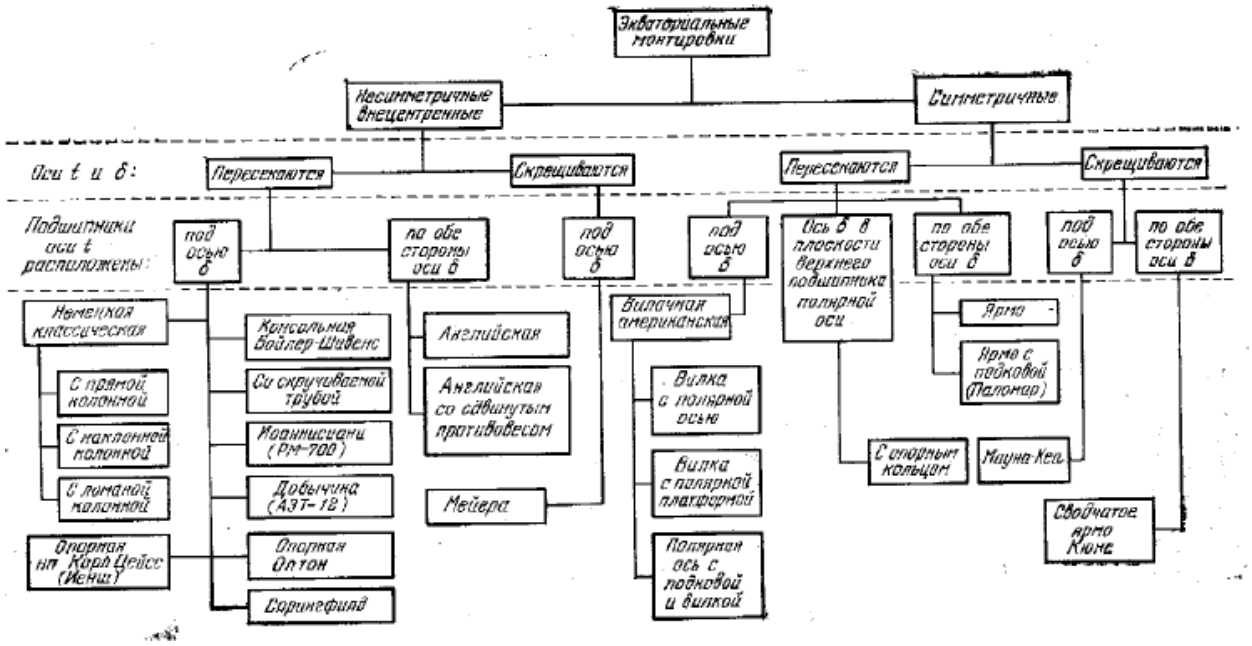


Рисунок 1.1.2 - Классификация экваториальных монтировок

Выбор типа используемой монтировки зависит от оптической схемы телескопа и его назначения. В симметричной монтировке трудна установка длинной трубы рефрактора. Для этого гораздо лучше подходят несимметричные монтировки.

1.2 Монтировка немецкого типа

Монтировка, обычно используемая для установки на неё рефракторов – это несимметричная монтировка *немецкого типа* – разновидность экваториальной монтировки, в которой на одном из концов полярной оси расположен корпус оси склонений. Один из недостатков этой монтировки заключается в том, что в тот момент, когда светило проходит меридиан, приходится прерывать наблюдение, из-за того, что труба телескопа упирается в колонну (основание). Также, из-за большого угла наклона использовать этот тип монтировок в областях, близких к полюсам, крайне неудобно. На рисунках 1.2.1 и 1.2.2 представлены монтировки немецкого типа.

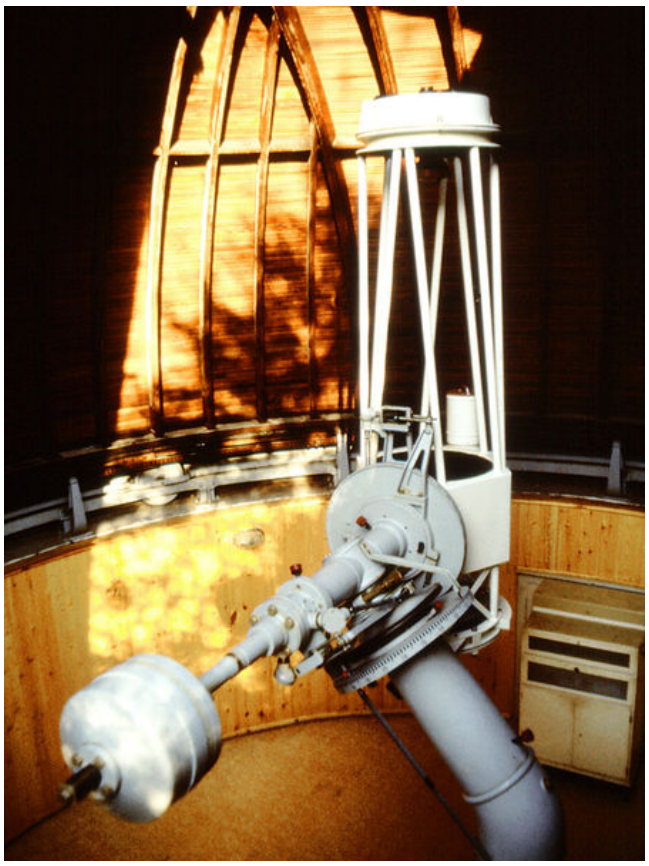
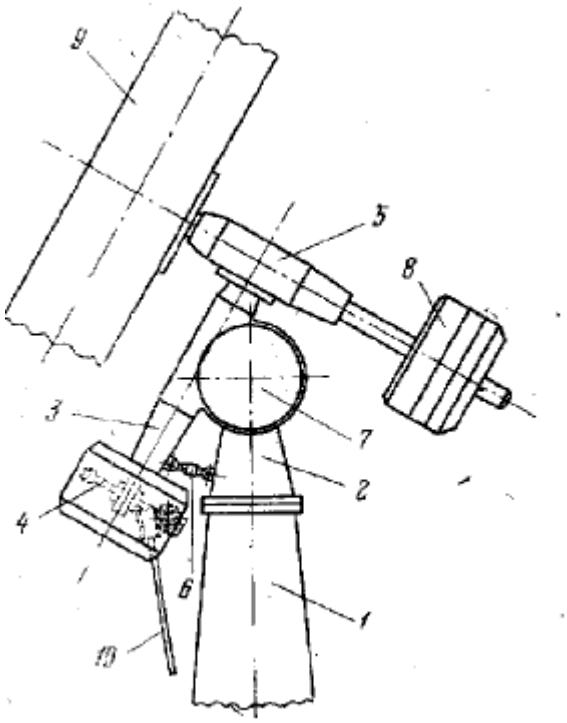


Рисунок 1.2.1 - 50 см телескоп Астрономической обсерватории в Йене, Германия, на немецкой монтировке

Рисунок 1.2.2 - Экваториальная монтировка немецкого типа для любительского телескопа.

На рисунке 1.2.3 изображена схема классической немецкой монтировки с параллактической головкой. Имеет смысл подробнее её рассмотреть.



1.2.3 - Схема классической немецкой монтировки с параллактической головкой

На вертикальной колонне 1 расположена параллактическая головка, которая состоит из корпуса 3 полярной оси и основания 2, которое его (корпус) поддерживает. В корпусе на подшипниках реализовано вращение полярной оси, к верхнему концу которой прикреплён корпус 5 оси склонений. Труба телескопа 9 прикреплена к фланцу на одном конце оси склонений, а на другом конце повешен противовес 8. Для регулирования головки по широте при помощи стяжки 6 корпус 3 полярной оси поворачивается вокруг горизонтальной вспомогательной оси 7, которая закреплена на основании 2. Через вал 10 и установленной на порярной оси систему передач к червяку, входящему в зацепление с червячной шестернёй 4, осуществляется передача вращения.

Наблюдению звёзд близ нижней кульминации мешает вертикальная колонна. Приходится прерывать наблюдения и проворачивать инструмент на

180^{o} вокруг полярной оси и на 2*(90-δ) вокруг оси склонений. Эта процедура названа *перекладкой*. Таким образом инструмент переводится через полюс.

1.3 Экваториальная монтировка EQ6 Pro

Рисунок 1.3.1 - Внешний вид монтировки EQ-6

Экваториальная монтировка Synta Sky-Watcher EQ6 Pro SynScan Goto (NEQ6-T PRO), используемая в данной работе – это монтировка немецкого типа производства компании Synta Sky-Watcher. Точность наведения монтировки лучше 1 угловой минуты, что вполне удовлетворяет требованиям к аппаратуре. Монтировка телескопа рассчитана на работу с трубами весом до 20 кг, оснащена встроенными приводами для наведения телескопа и стальной треногой регулируемой высоты. Управляется с ручного пульта.

Характеристики Synta Sky-Watcher EQ6 Pro:

- Экваториальная монтировка немецкого типа.
- Крепление телескопа осуществляется с помоющью крепёжных колец.

- Управление осуществляется ручным пультом управления с системой SynScan.
- Тренога из нержавеющей стали с диаметром ноги равным 51 мм.
- Высота треноги изменяется от 850 мм. до 1470 мм.
- Масса треноги составляет 7,5 кг.
- Диаметр штанги противовесов – 18 мм.
- Длина штанги противовесов – 285 мм.
- Штанга противовесов изготовлена из нержавеющей стали.
- Два противовеса по 5,4 кг.
- Вес монтировки – 16 кг.
- Высота монтировки – 406 мм.
- Шаговые двигатели 1.8 градуса/шаг в микрошаговом режиме.
- Дискретность электропривода – 0,144 угл. сек.(9.024,000 шагов/оборот.).
- Питание – постоянный ток, 11-15 В, 2 А.

Подробная информация доступна в источниках [1, 2].

ГЛАВА 2. ТЕОРИЯ ЭКВАТОРИАЛЬНОЙ МОНТИРОВКИ

2.1 Экваториальная система координат

Для нахождения координат звезд на небе используют так называемые *склонение* δ и *прямое восхождение* α, определяющие в пространстве сферическую систему координат, которая вращается вокруг оси мира вместе с небосводом. Ось мира проходит через точку наблюдения и полюс мира *P*. Угол α отсчитывается от точки весеннего равноденствия ϒ.

Звездное время S определяется как время, прошедшее с момента верхней кульминации точки ϒ, а *часовой угол t* - как угловое расстояние объекта Σ от плоскости меридиана.

S, *t* и α связаны простым соотношением:

$$t = S - \alpha.$$

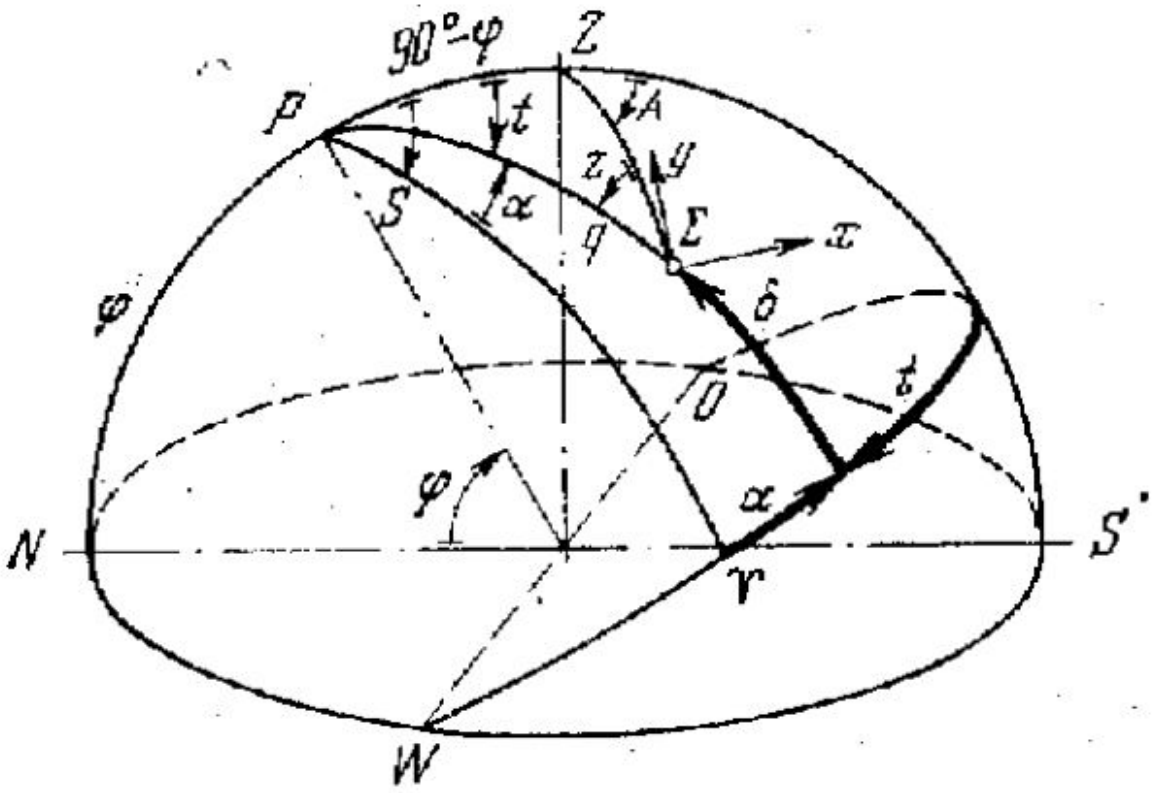


Рисунок 2.1.1 – Различные системы небесных координат.

NOSW – плоскость горизонта, P – полюс мира, NPZS’ – плоскость небесного меридиана, Z – зенит, φ – угол, равный широте места наблюдения. Связанный с полярной осью телескопа лимб позволяет определить часовой угол t, а в астрономическом ежегоднике или звездном каталоге можно найти прямое восхождение α. Тогда, зная звездное время S в момент наблюдения, можно навести телескоп на звезду, воспользовавшись приведенной выше формулой. Для слежения за звездой необходимо поворачивать телескоп вокруг полярной оси со скоростью 15 секунд дуги в 1 секунду звездного времени.

2.2 Ошибки наведения

Атмосферная рефракция вносит некоторые поправки, а именно уменьшает зенитное расстояние z светила на

$$\Delta z = r\, tg\, z, \qquad (2.2.1)$$

где

$$r = \frac{r_0 B}{760(1-0{,}00366\, T)}, \qquad (2.2.2)$$

r_0 – постоянная рефракции,

B – давление воздуха (мм.р.ст.),

T – его температура(в ℃) [7].

Вектор рефракции раскладывается на составляющие:

$$\Delta t_r = -r \sin q \sec\delta\, tg\, z = -r\, \frac{\cos\varphi \sin t \sec\delta}{\sin\varphi \sin\delta + \cos\varphi \cos\delta \cos t}, \qquad (2.2.3)$$

$$\Delta\delta_r = r \cos q\ \ tg\, z = r\, \frac{\sin\varphi \cos\delta - \cos\varphi \sin\delta \cos t}{\sin\varphi \sin\delta + \cos\varphi \cos\delta \cos t}, \qquad (2.2.4)$$

где q - угол при звезде между направлением на полюс и на зенит; это *параллактический угол* [7]. Со временем углы t, z, q меняются, что приводит к

изменению компонентов рефракции Δt_r и $\Delta\delta_r$.

Рефракция влияет не только на видимое расположение звезд, но также сдвигает точку P в точку P', называемую *видимым полюсом*, в связи с чем в процессе суточного движения склонение звезды смещается, а движение по часовому углу является неравномерным, из-за чего необходимо использовать гидирование. Кроме того, видимый полюс P' меняет свое положение, так как рефракция зависит от температуры и давления. Также нужно учитывать частые ошибки установки телескопа по широте и азимуту $\Delta\varphi$ и ΔA соответственно.

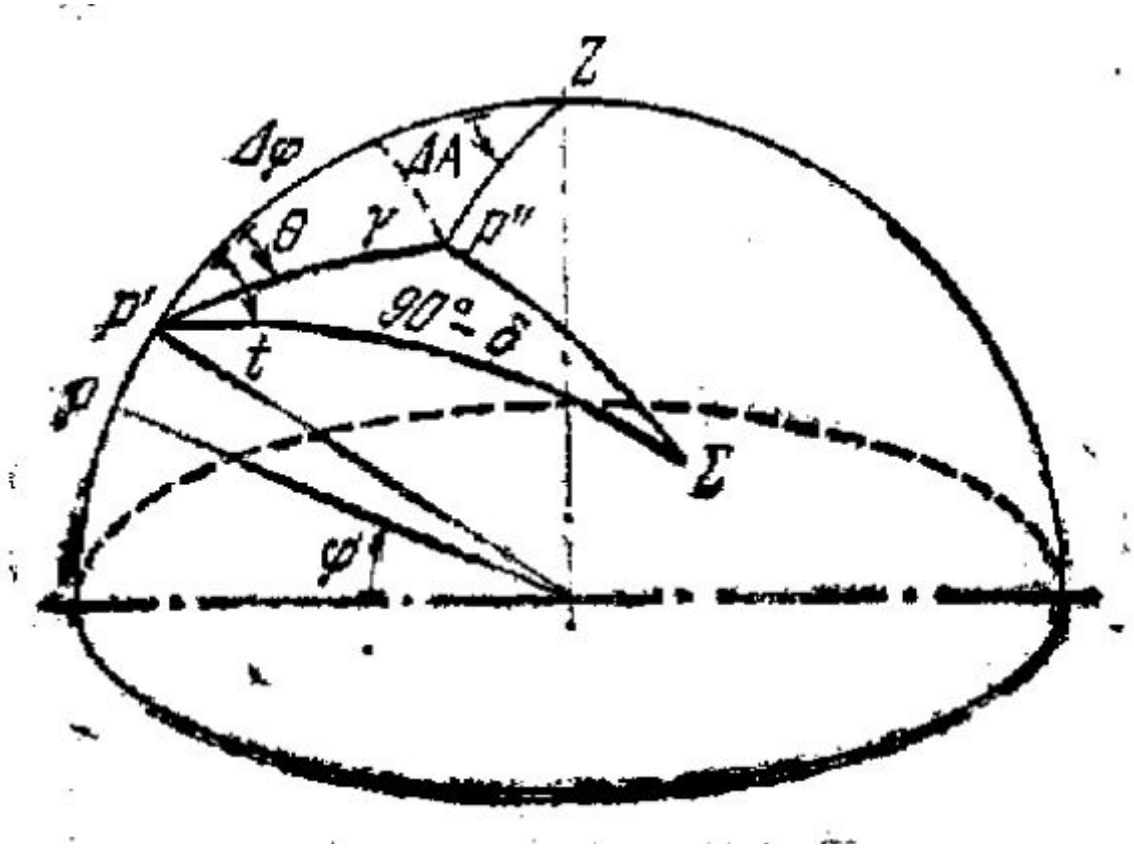


Рисунок 2.2.1 – Истинный (P), видимый (P') и инструментальный (P'') полюса мира.

В итоге полярная ось телескопа направлена на точку P''. Эту точку называют *инструментальный полюс*. Ошибки $\Delta\varphi$ и ΔA выражаются через угловое расстояние γ между точками P" и P' и часовой угол θ точки P" как [7]

$$\Delta\varphi = \gamma\cos\theta, \qquad \Delta A = \gamma\sin\theta\sec\varphi. \qquad (2.2.5)$$

Тогда формулы для вычисления компонент ошибки наведения телескопа на звезду Σ с координатами t и δ имеют вид [7]

$$\cos\delta\,\Delta t = \gamma\sin(t-\theta)\sin\delta, \qquad \Delta\delta = \gamma\cos(t-\theta)\,. \qquad (2.2.6)$$

Полная ошибка вычисляется по формуле [7]

$$\sqrt{(\cos\delta\,\Delta t)^2 + (\Delta\delta)^2}. \qquad (2.2.7)$$

Если телескоп ведет идеальный часовой механизм и нет вмешательства со стороны наблюдателя, то в процессе наблюдения звезда опишет дугу эллипса с полуосями $a = \gamma, b = \gamma\,sin\,\delta$.

Отклонение угла между осью склонения и полярной осью от перпендикулярного на величину i (ошибка наклонности) приводит к ошибке наведения [7]:

$$\cos\delta\,\Delta t = i\sin\delta, \qquad \Delta\delta = 0, \qquad (2.2.8)$$

а отклонение угла между оптической осью и осью склонения от перпендикулярного на величину c (ошибка коллимации) в свою очередь к ошибке наведения [7]:

$$\cos\delta\,\Delta t = c\,, \qquad \Delta\delta = 0. \qquad (2.2.9)$$

Также ошибку дает гнутие оси склонений [7]:

$$\cos\delta\,\Delta t = b(\sin\varphi\sin\delta + \cos\varphi\cos\delta\cos t) = b\cos z\,, \;\; \Delta\delta = 0, \quad (2.2.10)$$

Где
z - зенитное расстояние,
b – величина изгиба при $z = 0$.

Этот фактор особо существенен для несимметричных монтировок. Когда для ведения телескопа используется один лишь часовой механизм, звезда будет колебаться относительно оптической оси с максимальными отклонениями

$$x' = b\cos(\varphi - \delta), \qquad x'' = b\cos(\varphi + \delta). \quad (2.2.11)$$

Дифференциальное гнутие объективного и окулярного концов дает следующую ошибку

$$\cos\delta\Delta t = b'_p cos\,\varphi \sin t,\ \ \Delta\delta = -b'_k(\sin\varphi\, cos\,\delta - \cos\varphi \sin\delta \cos t), \quad (2.2.12)$$

где b'_p и b'_k − экстремальные гнутия трубы в направлении вдоль часового круга (от полюса к экватору) и вдоль круга параллелей, направленного к нему под углом в 90°. Тогда звезда опишет эллипс с центром

$$x = 0,\ \ y = -b'_k \sin\varphi \cos\delta, \quad (2.2.13)$$

а полуоси будут равны

$$a = b'_p \cos\varphi,\ \ b = b'_k \cos\varphi \sin\delta. \quad (2.2.14)$$

Суммарная ошибка [7]:

$$\left.\begin{aligned}
&\Delta t + r\,\mathrm{tg}\,z \sin q \sec\delta = \\
&\quad = \Delta T + (\Delta A \cos\varphi \sin t - \Delta\varphi \cos t \pm i \pm b \sin\varphi)\,\mathrm{tg}\,\delta + \\
&\qquad + (b'_p \cos\varphi \sin t \pm c)\sec\delta \pm b\cos\varphi\cos t, \\
&\Delta\delta - r\,\mathrm{tg}\,z\cos q = \pm\Delta D + \Delta A\cos\varphi\cos t + \\
&\qquad + \Delta\varphi \sin t - b'_h(\sin\varphi\cos\delta - \cos\varphi\sin\delta\cos t),
\end{aligned}\right\} \quad (2.2.15)$$

2.3 Гидирование и вращение поля

Гидирование − процедура контроля положения телескопа и его исправление (в автоматическом или ручном режимах). Необходимость в нём возникает из-за наличия при установке телескопа вышеобозначенных ошибок. Можно определить скорость ухода звезды с креста нитей микрометра [7]

$$\begin{aligned}
57{,}3\,\frac{d\Delta t}{dt} &= \gamma\,\mathrm{tg}\,\delta\cos(t-\theta) + b'_p\cos\varphi\sec\delta\cos t - \\
&\quad - r\,\frac{\cos\varphi}{\cos\delta}\cdot\frac{\cos\varphi\cos\delta + \sin\varphi\sin\delta\cos t}{(\sin\varphi\sin\delta + \cos\varphi\cos\delta\cos t)^2} \mp b\cos\varphi\sin t, \\
57{,}3\,\frac{d\Delta\delta}{dt} &= -\gamma\sin(t-\theta) - b'_h\cos\varphi\sin\delta\sin t + \\
&\quad + r\,\frac{\sin\varphi\cos\varphi\sin t}{(\sin\varphi\sin\delta + \cos\varphi\cos\delta\cos t)^2}.
\end{aligned} \quad (2.3.1)$$

Наличие ошибок атмосферной рефракции и инструментальной обибки экваториала приводят к появлению вращения поля. За время Δt поле повернётся на угловую величину Δw [7].

$$\Delta\omega = \frac{d\omega}{dt}\Delta t = \left[\gamma \cos(t-\theta)\sec\delta + b_p' \cos\varphi \operatorname{tg}\delta \cos t + \mu \cos\varphi \sin\delta \sin t - r \operatorname{tg}\delta \cos\varphi \frac{\cos\varphi\cos\delta + \sin\varphi\sin\delta\cos t}{(\sin\varphi\sin\delta + \cos\varphi\cos\delta\cos t)^2}\right]\Delta t, \quad (2.3.2)$$

90 – γ – склонение,

θ – часовой угол инструментального склонения полюся,

μ – значение скручивания трубы телескома.

ГЛАВА 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

3.1 IDE Microsoft Visual C++ 2010 Express

Языком программирования был выбран компилируемый язык общего назначения C++, а используемой средой разработки – Microsoft Visual Studio 2010.

MS Visual C++ – интегрированная среда разработки приложений на языке C++, содержащая в себе набор инструментов, позволяющих разрабатывать приложения для операционных систем семейства Windows. В число этих инструментов входит библиотека Microsoft Foundation Classes (MFC). Пакет Microsoft Foundation Classes (MFC) — библиотека на языке C++, разработанная Microsoft и призванная облегчить разработку GUI-приложений для Microsoft Windows путём использования богатого набора библиотечных классов [4]. На Рисунок 2.1.1 представлена стартовая страница проекта Dialog Based MFC-приложения.

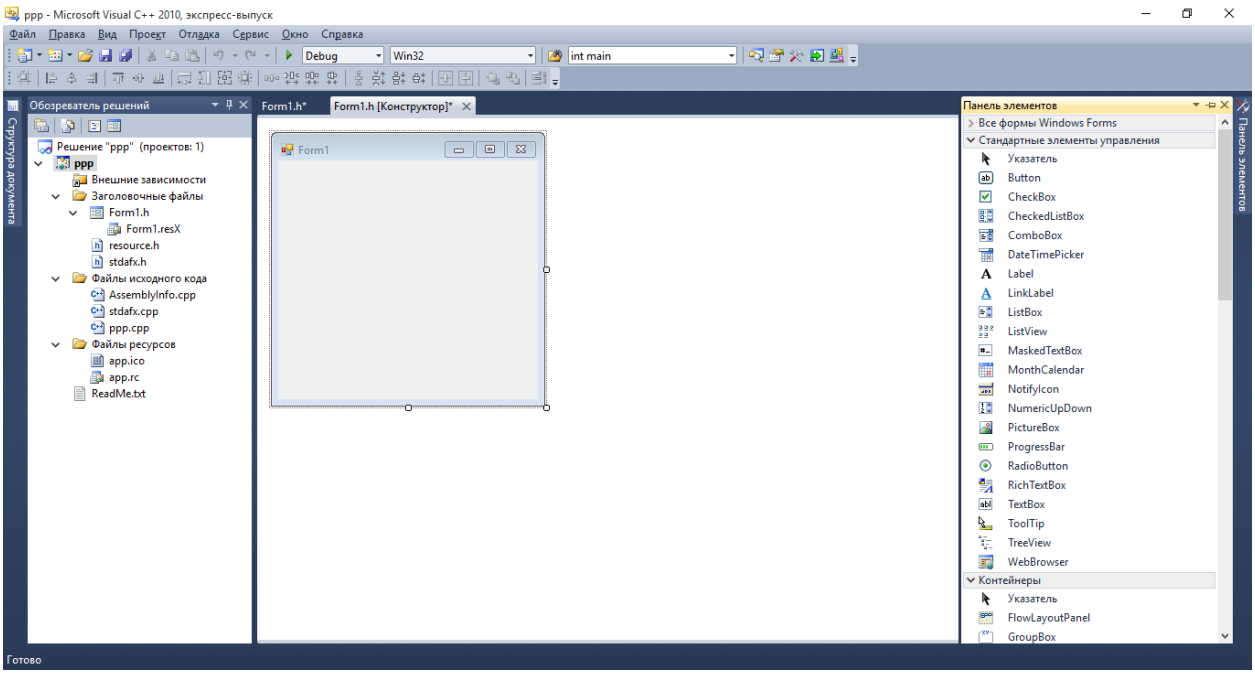


Рисунок 3.1.1 - Стартовый экран проекта MFC-приложения в IDE MS Visual C++.

Для изучениия среды Visual C++ и синтаксиса языка C++ были использованы следующие учебные пособия [5, 6].

3.2 Структура программы «ClientControl»

В рамках этой работы необходимо было доработать систему наведения, реализованную в программе ClientControl, и интегрировать в неё все наработки, которые были произведены при выполнении работы.

На рисунке 3.2.1 представлена диаграмма классов программы ClientControl.

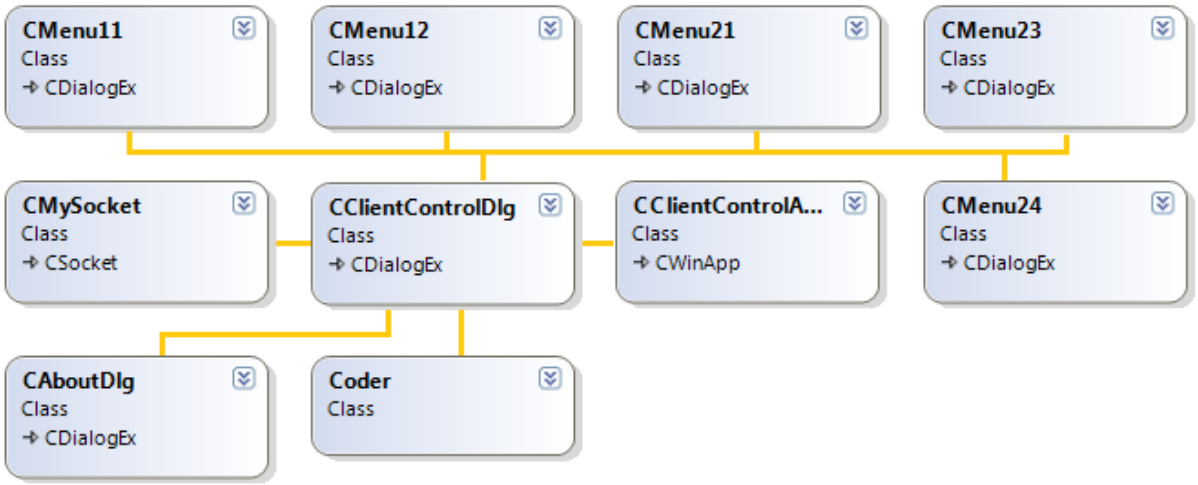


Рисунок 3.2.1 - Диаграмма классов программы ClientControl

Как видно из диаграммы, главным классом является CClientControlDlg. Через него происходит взаимодействие остальных классов программы. Данный класс производит сбор данных (координаты объекта, на который необходимо произвести наведение), выбирает необходимый «метод» из класса Coder для обработки данных (первод экваториальных координат в координаты Ax1/Ax2, используемые монтировкой), отправляет данные на обработку соответствующему методу, принимает обработанные значения и отправляет их в виде команды на сервер.

Поскольку данная программа учитывала при наведении рефракцию только при фиксированном значении давления и температуры (760 мм. р. с. и

10^{o} С), а также не учитывала погрешностей, вносимых отклонением видимого полюса мира от инструментального; неперпендикулярностью оси склонений и полярной оси, оптической оси и оси склонений; гнутием оси склонений, дифференциальным гнутием объективного и окулярного концов, ошибками нуль-пунктов лимбовых углов и склонений, мной была разработана библиотека классов Guidance, в которой и был реализован учёт всех этих погрешностей наведения. Рассчёт введётся в методах классов исходя из вышепредставленных формул. В результате работы эта библиотека была интегрирована в программу ClientControl.

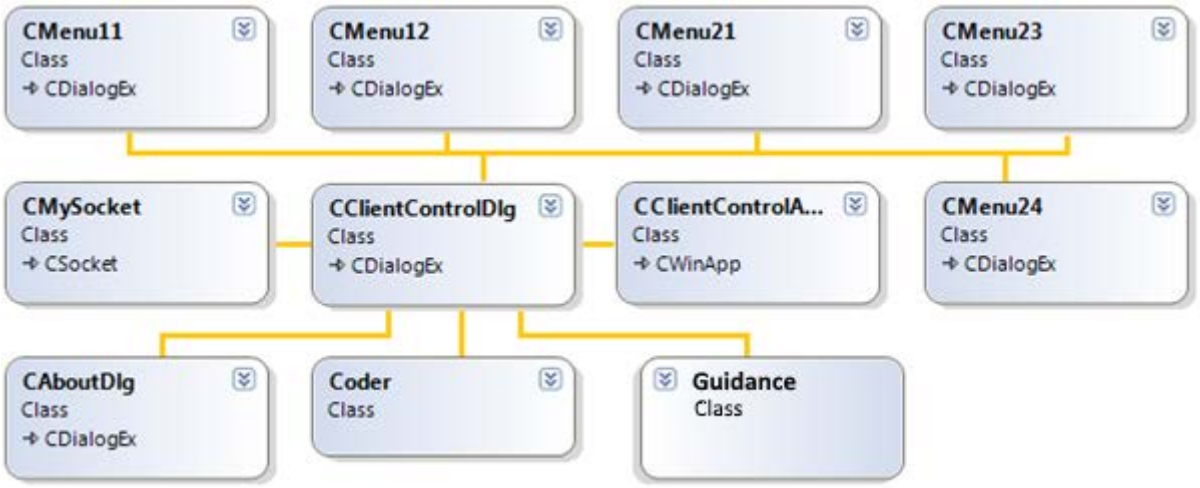


Рисунок 3.2.2 - Диаграмма классов программы ClientControl после интеграции библиотеки классов учёта погрешностей

Краткое содержание библиотеки классов Guidance:

1. Class R{}; – рассчёт рефракции с учётом температуры и давления;
2. Class Time{}; – хранение, перерасчёт временного значения;
3. Class AtmRefr{}; – рассчёт поправок на рефракцию по компонентам часового угла и склонения;
4. Class OffsetPole{}; – рассчёт поправок на смещение видимого и инструментально полюсов;
5. Class CollimationError{}; – рассчёт поправок на коллимацию;
6. Class BendingError{}; – рассчёт поправок на гнутие оси склонений;

7. Class DiffBendingError{}; – рассчёт поправок на гнутие трубы в напрвлении вдоль часового круга (от полюса к экватору) и вдоль круга параллелей, перпендикулярному часовому кругу;
8. Class SumErr{}; – рассчёт итогой поправки (ведётся с учётом всех остальных поправок);
9. Class VelocityStarGid{}; – рассчёт поправок на скорость ухода объекта гидирования;
10. Class RotatingField{}; – рассчёт поправок на вращение поля.

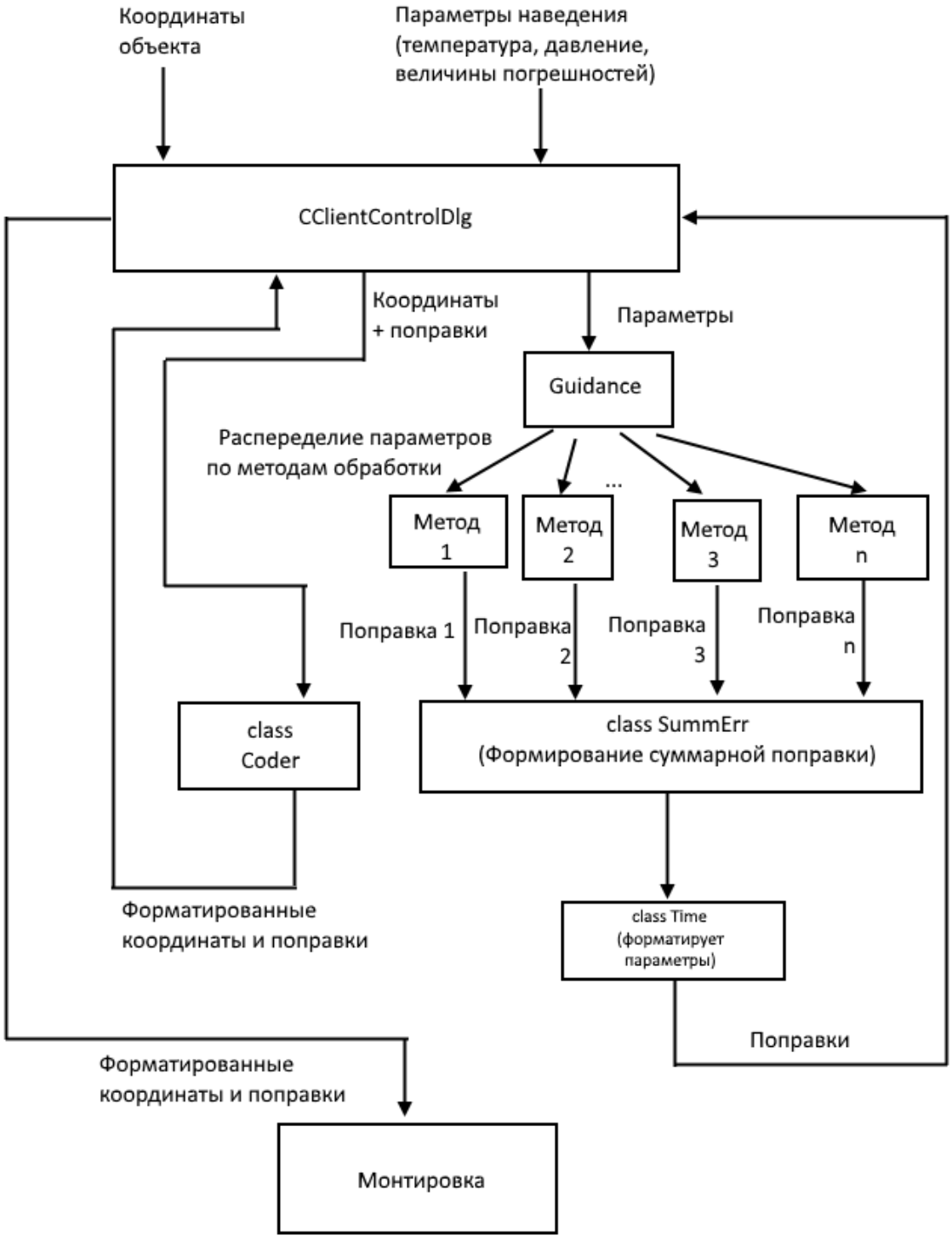
В качестве примера представлен класс рассчёта поправки на рефракцию:

```
class AtmRefr {
public:
   // Вводим углы фи, т, дельта в градусах
   void set_angles(double _phi, double _t, double _dlt){
          // Переводим их в радианы
          phi = _phi; // полюс мира = широте наблюдения
          t = _t; // часовой угол
          dlt = _dlt; // склонение звезды
          phi*=PI/180; // Перевод в радианы
          t*=PI/180;
          dlt*=PI/180;
   }

   // Получаем поправку по часовому углу (в секундах)
   double get_r_t(){
          r_t = -r0*( (cos(phi)*sin(t)/cos(dlt)) / \
                 ((sin(phi)*sin(dlt))+(cos(phi)*cos(dlt)*cos(t))) );
          return r_t;
   }

   // Получаем поправку по склонениию (в секундах)
   double get_r_dlt(){
          r_dlt = r0*(\
                 ( (sin(phi)*cos(dlt)) - (cos(phi)*sin(dlt)*cos(t)) ) / \
                 ( (sin(phi)*cos(dlt)) + (cos(phi)*sin(dlt)*cos(t)) ));
          return r_dlt;
   }

private:
   double r_t;
   double r_dlt;
   double phi;
   double t;
   double dlt;
};
```

Конечный размер библиотеки Guidance составил 476 строк кода + 167 строк для связки файлов Guidance.cpp и CClientControlDlg.cpp.

Размер доработанного программного обеспечения – 1.82 МБ.

На рисунке 3.2.3 представлен упрощённый алгоритм работы программы с учётом внесённых изменений.



3.2.3 - Упрощённый алгоритм работы программы

3.3 Графический интерфейс

Изменениям также подвергся и графический интерфейс программы. Поскольку для рассчёта поправок и последующего наведения необходимо откуда-то принимать значения коэффициентов ошибок, в диалоговое окно ПО ClientControl была добавлена панель ввода ошибок, учитывающихся при рассчёте поправок. На рисунке 3.3.1 показана эта панель. Ниже, под полями ввода, распололодена панель, отображающая основные поправки, итоговую поправку (FULL ERROR) и итоговое значение углов поворота для наведения на объект с учётом всех поправок.

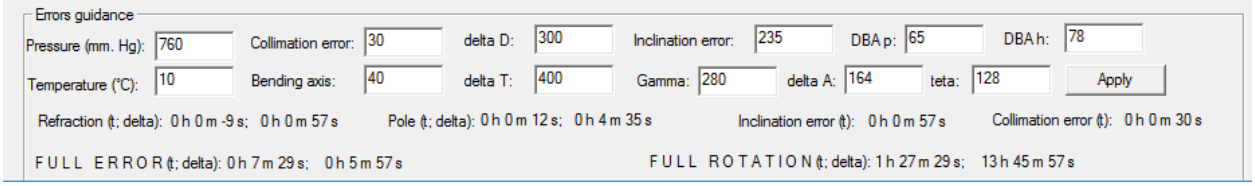


Рисунок 3.3.1 - Панель ввода данных

Здесь:

Pressure – атмосферное давление;

Temperature – температура воздуха;

Collimation Error – ошибка коллимации *i*;

Bending axis – величина изгиба оси склрнений;

delta D – ошибка нулль-пунктов лимбов склонений;

delta T – ошибка нульь-пунктов лимбов часовых углов;

Inclination error – ошибка наклонности *c*;

Gamma – угловое расстояние между инструментальным и видимым полюсами;

DBA p – дифференциальное гнутие объективного конца трубы;

DBA h - дифференциальное гнутие оккулярного конца трубы;

Delta A – ошибка установки телескопа по азимуту;

Teta – часовой угол инструментального полюса мира;

Все угловые параметры задаются в секундах!

Если необходимо передать повернуть телескоп, без учёта поправок, достаточно заполнить все поля ввода в панели Errors guidance нулями.

Рассчёт поправок и поворот телескопа производится нажанием кнопки «GO TO».

На рисунке 3.3.2 представлен итоговый графический интерфейс доработанной программы.

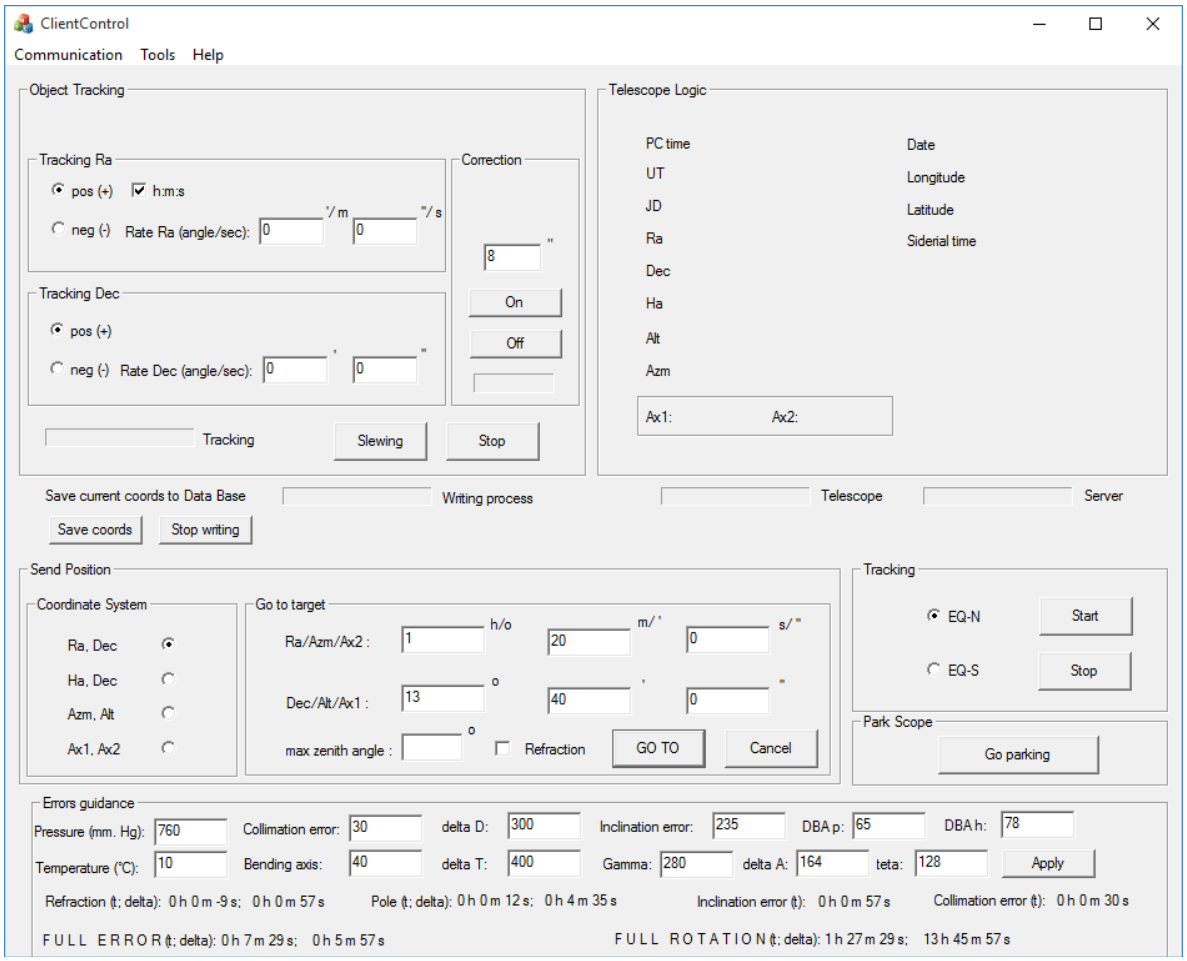


Рис 3.3.2 - Графический интерфейс итоговой программы.

3.4 Листинг кода

// Guidance.cpp

```
#include <iostream>
#include <cmath>
using namespace std;

// постоянная рефракции в секундах
static const double r0 = 60;
static const double PI = 3.14159265;

class R{
public:
    void set_param(double _B, double _T){
           B=_B;
           T=_T;
    }
    double get_r(){
           double r;
           r0=60;
           r=(r0*B)/(760*(1-(0.00366*T)));
           return r;
    }
private:
    double B;
    double T;
    double r0;
};

class Time{
public:
    // Принимаем число секунд
    void set_time(double _num){
           num = _num;
    }
    // Вытаскиваем из него число часов
    int get_hour(){
           int hour;
           hour = num / 3600;
           return hour;
    }
    // Число минут
    int get_min(){
           int min;
           min = ((int)num % 3600) / 60;
           return min;
    }
    // Число секунд
    int get_sec(){
           int sec;
           sec = ((int)num % 3600) % 60;
           return sec;
    }
private:
    double num;
};

// Атмосферная рефракция
class AtmRefr {
public:
    // Вводим углы фи, т, дельта в градусах
    void set_angles(double _phi, double _t, double _dlt){
           // Переводим их в радианы
           phi = _phi; // полюс мира = широте наблюдения
           t = _t; // часовой угол
           dlt = _dlt; // склонение звезды
           phi*=PI/180;
```

```
            t*=PI/180;
            dlt*=PI/180;
    }

    // Получаем поправку по часовому углу (в секундах)
    double get_r_t(){
            r_t = -r0*( (cos(phi)*sin(t)/cos(dlt)) / \
                    ((sin(phi)*sin(dlt))+(cos(phi)*cos(dlt)*cos(t))) );
            return r_t;
    }

    // Получаем поправку по склонениию (в секундах)
    double get_r_dlt(){
            r_dlt = r0*(\
                    ( (sin(phi)*cos(dlt)) - (cos(phi)*sin(dlt)*cos(t)) ) / \
                    ( (sin(phi)*cos(dlt)) + (cos(phi)*sin(dlt)*cos(t)) ));
            return r_dlt;
    }

private:
    double r_t;
    double r_dlt;
    double phi;
    double t;
    double dlt;
};


// Смещение полюса P
class OffsetPole{
public:
    double set_param(double _gamma, double _tetta, double _phi){
            phi = _phi;
            tetta = _tetta;
            gamma = _gamma;
            phi *=  PI / 180;
            tetta *=  PI / 180;
    }
    double get_delta_phi(){
            double delta_phi;
            delta_phi = gamma * cos(tetta);
            return delta_phi;
    }
    double get_delta_A(){
            double delta_A;
            delta_A = gamma * sin(tetta) / cos(phi);
            return delta_A;
    }
private:
    double shir_er;
    double az_er;
    double gamma;
    double tetta;
    double phi;
};

class InclinationsError{
public:
    void set_param(double _i, double _delta){
            delta = _delta;
            i = _i;
            delta *=  PI / 180;
    }
    double get_inclination_errror(){
            dlt_t = i * sin(delta) / cos(delta);
```

```
            return dlt_t;
    }
private:
    double i;
    double delta;
    double dlt_t;
};

class CollimationError{
public:
    void set_param(double _c, double _delta){
            delta = _delta;
            c = _c;
            delta *=  PI / 180;
    }
    double get_collimation_errror(){
            double dlt_t;
            dlt_t = c / cos(delta);
            return dlt_t;
    }
private:
    double c;
    double delta;
};

class BendingError{
public:
    void set_param(double _b, double _delta, double _phi,\
            double _t, double _z){
            delta = _delta;
            b = _b;
            delta = _delta;
            phi = _phi;
            t = _t;
            z = _z;
            delta *=  PI / 180;
            phi *=  PI / 180;
            t *=  PI / 180;
    }
    double get_bend_err(){
            double dlt_t;
            double mult1;
            double mult2;
            mult1=sin(phi)*sin(delta);
            mult2=cos(delta)*cos(phi)*cos(t);
            dlt_t = (mult1+mult2)/cos(delta);
            return dlt_t;
    }
private:
    double delta;
    double b;
    double phi;
    double t;
    double z;
};

class DiffBendingError{
public:
    void set_param(double _bs_p, double _bs_h, double _delta, \
            double _phi, double _t){
            delta = _delta;
            bs_p = _bs_p;
            bs_h = _bs_h;
            delta = _delta;
            phi = _phi;
```

```
            t = _t;
            delta *=  PI / 180;
            phi *=  PI / 180;
            t *=  PI / 180;
    }
    double get_diff_bend_err_dlt_t(){
            double dlt_t;
            double mult1;
            mult1=bs_p*sin(phi)*sin(t);
            dlt_t=mult1/cos(delta);
            return dlt_t;
    }
    double get_diff_bend_err_dlt_delta(){
            double dlt_delta;
            double mult1;
            double mult2;
            mult1=sin(phi)*cos(delta);
            mult2=cos(phi)*sin(delta)*cos(t);
            dlt_delta= -bs_h*(mult1-mult2);
            return dlt_delta;
    }
private:
    double delta;
    double b;
    double phi;
    double t;
    double bs_p;
    double bs_h;
};

class SumErr{
public:
    // Задать рабочие параметры
    void set_param(double _dlt_A, double _phi, double _t, double _dlt_phi,\
            double _i, double _b, double _dlt, double _bs_p, double _c,\
            double _dlt_D, double _bs_h, double _dlt_T){
            dlt_A = _dlt_A;
            phi = _phi;
            t = _t;
            dlt_phi = _dlt_phi;
            i = _i;
            b = _b;
            dlt = _dlt;
            bs_p = _bs_p;
            c = _c;
            dlt_D = _dlt_D;
            bs_h = _bs_h;
    }

    // Далее знаки "+" и "-" (т.е. функции с постфиксами
    // _plus и _minus в именах) зависят от положения лимба
    // склонений относительно трубы телескопа (слева или
    // справа). Это положение меняется при перекладке.

    // Получить значение суммарной ошибки по компоненте dlt_t
    double get_sum_err_dlt_t_plus(){
            double slag1;
            double slag2;
            double slag3;
            double mult1;
            double mult3;
            double mult4;
            double dlt_t;
            slag1=dlt_A*cos(phi)*sin(t);
            mult1=slag1-dlt_phi+cos(t)+i+(b*sin(phi));
```

```
        slag2=dlt_T+(mult1*tan(dlt));
        mult3=((bs_p*cos(phi)*sin(t))+c)/cos(dlt);
        mult4=b*cos(phi)*cos(t);
        slag3=mult3+mult4;
        dlt_t=slag2+slag3;
        return dlt_t;
    }
    // Получить значение суммарной ошибки по компоненте dlt_t
    double get_sum_err_dlt_t_minus(){
        double slag1;
        double slag2;
        double slag3;
        double mult1;
        double mult3;
        double mult4;
        double dlt_t;
        slag1=dlt_A*cos(phi)*sin(t);
        mult1=slag1-dlt_phi+cos(t)-i-(b*sin(phi));
        slag2=dlt_T+(mult1*tan(dlt));
        mult3=((bs_p*cos(phi)*sin(t))-c)/cos(dlt);
        mult4=b*cos(phi)*cos(t);
        slag3=mult3-mult4;
        dlt_t=slag2+slag3;
        return dlt_t;
    }
    // Получить значение суммарной ошибки по компоненте dlt_dlt
    double get_sum_err_dlt_dlt_plus(){
        double temp1;
        double temp2;
        double temp3;
        double temp4;
        double temp5;
        double temp6;
        double dlt_dlt;
        temp1=dlt_A*cos(phi)*cos(t);
        temp2=dlt_phi*sin(t);
        temp3=temp1+temp2;
        temp4=-bs_h*(sin(phi)*cos(dlt));
        temp5=cos(phi)*sin(dlt)*cos(t);
        temp6=temp4-temp5;
        dlt_dlt=dlt_D+temp3+temp6;
    }
    // Получить значение суммарной ошибки по компоненте dlt_dlt
    double get_sum_err_dlt_dlt_minus(){
        double temp1;
        double temp2;
        double temp3;
        double temp4;
        double temp5;
        double temp6;
        double dlt_dlt;
        temp1=dlt_A*cos(phi)*cos(t);
        temp2=dlt_phi*sin(t);
        temp3=temp1+temp2;
        temp4=-bs_h*(sin(phi)*cos(dlt));
        temp5=cos(phi)*sin(dlt)*cos(t);
        temp6=temp4-temp5;
        dlt_dlt=-dlt_D+temp3+temp6;
    }

private:
    double dlt_A;
    double phi;
    double t;
    double dlt_phi;
```

```
    double i; // ошибка наклонности
    double b; // изгиб оси склонений при z = 0
                        // z - зенитное расстояние звезды
    double dlt;
    double bs_p;
    double c; // ошибка коллимации
    double bs_h;
    double dlt_D; // ошибки нуль-пунктов лимбов часовых углов
    double dlt_T; // и склонений
};

class VelocityStarGid{
public:
    void set_param(double _gamma,double _teta, double _t, double _delta,\
           double _bs_p, double _r, double _phi, double _vel_dlt_t,\
           double _vel_dlt_dlt, double _b, double _bs_h){
           gamma=_gamma;
           teta=_teta;
           t=_t;
           delta=_delta;
           bs_p=_bs_p;
           bs_h=_bs_h;
           r=_r;
           b=_b;
           phi=_phi;
           vel_dlt_t=_vel_dlt_t;
           vel_dlt_dlt=_vel_dlt_dlt;
           delta*=PI/180;
           t*=PI/180;
           teta*=PI/180;
           phi*=PI/180;
    }
    double get_vel_star_gid_dlt_t_minus(){
           double slag1;
           double slag2;
           double slag3;
           double slag4;
           double mult1;
           double chisl;
           double sqrt_znam;
           double vel_star_gid_dlt_t;
           slag1=gamma*tan(delta)*cos(t-teta);
           slag2=bs_p*cos(phi)+cos(t)/cos(delta);
           mult1=r*cos(phi)/cos(delta);
           chisl=(cos(phi)*cos(delta))-(sin(phi)*sin(delta)*cos(t));
           sqrt_znam=(sin(phi)*sin(delta))-(cos(phi)*cos(delta)*cos(t));
           slag3=(mult1*chisl)/(sqrt_znam*sqrt_znam);
           slag4=b*cos(phi)*sin(t);
           vel_star_gid_dlt_t=slag1+slag2-slag3-slag4;
           return vel_star_gid_dlt_t;
    }
    double get_vel_star_gid_dlt_t_plus(){
           double slag1;
           double slag2;
           double slag3;
           double slag4;
           double mult1;
           double chisl;
           double sqrt_znam;
           double vel_star_gid_dlt_t;
           slag1=gamma*tan(delta)*cos(t-teta);
           slag2=bs_p*cos(phi)+cos(t)/cos(delta);
           mult1=r*cos(phi)/cos(delta);
           chisl=(cos(phi)*cos(delta))-(sin(phi)*sin(delta)*cos(t));
           sqrt_znam=(sin(phi)*sin(delta))-(cos(phi)*cos(delta)*cos(t));
```

```
        slag3=(mult1*chisl)/(sqrt_znam*sqrt_znam);
        slag4=b*cos(phi)*sin(t);
        vel_star_gid_dlt_t=slag1+slag2-slag3+slag4;
        return vel_star_gid_dlt_t;
    }
    double get_vel_star_gid_dlt_dlt(){
        double slag1;
        double slag2;
        double slag3;
        double mult1;
        double chisl;
        double sqrt_znam;
        double vel_star_gid_dlt_dlt;
        slag1=gamma*sin(t-teta);
        slag2=bs_h*cos(phi)*sin(delta)*sin(t);
        chisl=sin(phi)*cos(phi)*sin(t);
        sqrt_znam=(sin(phi)*sin(delta))+(cos(phi)*cos(delta)*cos(t));
        slag3=(r*chisl)/(sqrt_znam*sqrt_znam);
        vel_star_gid_dlt_dlt=-slag1-slag2+slag3;
        return vel_star_gid_dlt_dlt;
    }

private:
    double gamma;
    double teta;
    double t;
    double delta;
    double bs_p;
    double bs_h;
    double b;
    double r;
    double phi;
    double vel_dlt_t;
    double vel_dlt_dlt;
};

class RotatingField{
public:
    void set_param(double _gamma,double _teta, double _t, double _delta,\
        double _bs_p, double _r, double _phi, double _mu,\
        double _dlt_t, double _b){
        gamma=_gamma;
        teta=_teta;
        t=_t;
        delta=_delta;
        bs_p=_bs_p;
        r=_r;
        b=_b;
        phi=_phi;
        mu=_mu;
        dlt_t=_dlt_t;
        delta*=PI/180;
        t*=PI/180;
        teta*=PI/180;
        phi*=PI/180;
    }
    double get_rot_field_err(){
        double slag1;
        double slag2;
        double slag3;
        double slag4;
        double mult1;
        double chisl;
        double sqrt_znam;
        double rot_field_err;
```

```
            slag1=gamma*cos(t-teta)/cos(delta);
            slag2=bs_p*cos(phi)*tan(delta)*cos(t);
            slag3=mu*cos(phi)*sin(delta)*sin(t);
            chisl=(cos(phi)*cos(delta))+(sin(phi)*sin(delta)*cos(t));
            sqrt_znam=(sin(phi)*sin(delta))+(cos(phi)*cos(delta)*cos(t));
            slag4=r*tan(delta)*cos(phi)*chisl/(sqrt_znam*sqrt_znam);
            mult1=slag1+slag2+slag3-slag4;
            rot_field_err=mult1*dlt_t;
        }
    private:
        double gamma;
        double teta;
        double t;
        double delta;
        double bs_p;
        double b;
        double r;
        double phi;
        double mu;
        double dlt_t;
    };
```

// Изменения, внесённые в файл CClientControlDlg.cpp

```
 . . .

//ASC
// Строки 101 - 118
        a_dlt_A = 0;
        a_gamma = 0;
        a_phi = 0;
        a_t = 0;
        a_dlt_phi = 0;
        a_i = 0;
        a_b = 0;
        a_dlt = 0;
        a_teta = 0;
        a_bs_p = 0;
        a_c = 0;
        a_bs_h = 0;
        a_dlt_D = 0;
        a_dlt_T = 0;
        a_P = 760;
        a_Temp = 10;
        //EASC
. . .
//ASC
// Строки 125 - 138
        DDX_Text(pDX, IDC_EDIT5, a_P);
        DDX_Text(pDX, IDC_EDIT7, a_Temp);
        DDX_Text(pDX, IDC_EDIT15, a_c);
        DDX_Text(pDX, IDC_EDIT16, a_b);
        DDX_Text(pDX, IDC_EDIT18, a_dlt_D);
        DDX_Text(pDX, IDC_EDIT19, a_dlt_T);
        DDX_Text(pDX, IDC_EDIT14, a_i);
        DDX_Text(pDX, IDC_EDIT20, a_teta);
        DDX_Text(pDX, IDC_EDIT17, a_bs_p);
```

```
        DDX_Text(pDX, IDC_EDIT4, a_bs_h);
        DDX_Text(pDX, IDC_EDIT22, a_dlt_A);
        DDX_Text(pDX, IDC_EDIT21, a_gamma);
        //EASC

//ASC
// Строки 492 - 614

        /*
        a_dlt_A = 0;
        a_gamma = 0;
        a_phi = 0;
        a_t = 0;
        a_dlt_phi = 0;
        a_i = 0;
        a_b = 0;
        a_dlt = 0;
        a_bs_p = 0;
        a_c = 0;
        a_bs_h = 0;
        a_dlt_D = 0;
        a_dlt_T = 0;
        a_P = 760;
        a_Temp = 10;
        */
        // m_st - звёздное время
        const double PI = 3.14159265;
        int LST = 0;
        double a_alpha = 0;
        SYSTEMTIME a_lt;
        GetLocalTime(&a_lt);

        a_phi = ((m_latg*60) + m_latm)*60;       // широта в секундах
        a_phi = a_phi*PI/(180*3600);             // широта в радианах

        a_dlt = (m_decg*3600) + (m_decm*60) + m_decs;   // склонение в секундах
        a_dlt = a_dlt*PI/(180*3600);                                  // склонение в
радианах

        a_alpha = (m_rah*3600) + (m_ram*60) + m_ras;    // восхождение в секундах
        a_alpha = a_alpha*PI/(180*3600);                        // восхождение в радианах

        LST=(a_lt.wHour*3600)+(a_lt.wMinute*60)+a_lt.wSecond; // Звёздное время в секундах
        a_t = LST - a_alpha;        // часовой угол в секундах
        a_t = a_t*PI/(180*3600);    // часовой угол в радианах

        double refr_dlt_t = 0;
        double refr_dlt_dlt = 0;
        double r=(60*a_P)/(760*(1-(0.00366*a_Temp)));
        // рефракция в секундах
        refr_dlt_t = -
r*(cos(a_phi)*sin(a_t)/cos(a_dlt))/((sin(a_phi)*sin(a_dlt))+(cos(a_phi)*cos(a_dlt)*cos(a_
t)));
        int refr_dlt_t_h = refr_dlt_t/3600;
        int refr_dlt_t_m = refr_dlt_t/60;
        int refr_dlt_t_s = refr_dlt_t - (refr_dlt_t_h*3600) - (refr_dlt_t_m*60);
        refr_dlt_dlt = r*((sin(a_phi)*cos(a_dlt))-
(cos(a_phi)*sin(a_dlt)*cos(a_t)))/((sin(a_phi)*sin(a_dlt))+(cos(a_phi)*cos(a_dlt)*cos(a_t
)));
        int refr_dlt_dlt_h = refr_dlt_dlt/3600;
        int refr_dlt_dlt_m = refr_dlt_dlt/60;
        int refr_dlt_dlt_s = refr_dlt_dlt - (refr_dlt_dlt_h*3600) - (refr_dlt_dlt_m*60);

        CString str;
```

```
	str.Format("%i h %i m %i s;   %i h %i m %i s", refr_dlt_t_h, refr_dlt_t_m,
refr_dlt_t_s, refr_dlt_dlt_h, refr_dlt_dlt_m, refr_dlt_dlt_s);
	SetDlgItemText(IDC_EDITRA2, str);


	a_teta *= PI/(180*3600);
	a_dlt_phi = a_gamma*cos(a_teta);
	a_dlt_A = a_gamma*sin(a_teta)/cos(a_phi);

	double pol_dlt_t = a_gamma*sin(a_t-a_teta)*sin(a_dlt)/cos(a_dlt);
	double pol_dlt_dlt = a_gamma*cos(a_t-a_teta);

	int pol_dlt_t_h = pol_dlt_t/3600;
	int pol_dlt_t_m = pol_dlt_t/60;
	int pol_dlt_t_s = pol_dlt_t - (pol_dlt_t_h*3600) - (pol_dlt_t_m*60);

	int pol_dlt_dlt_h = pol_dlt_dlt/3600;
	int pol_dlt_dlt_m = pol_dlt_dlt/60;
	int pol_dlt_dlt_s = pol_dlt_dlt - (pol_dlt_dlt_h*3600) - (pol_dlt_dlt_m*60);

	str.Format("%i h %i m %i s;   %i h %i m %i s", pol_dlt_t_h, pol_dlt_t_m,
pol_dlt_t_s, pol_dlt_dlt_h, pol_dlt_dlt_m, pol_dlt_dlt_s);
	SetDlgItemText(IDC_EDITRA3, str);


	double i_dlt_t = a_i*sin(a_dlt)/cos(a_dlt);

	int i_dlt_t_h = i_dlt_t/3600;
	int i_dlt_t_m = i_dlt_t/60;
	int i_dlt_t_s = i_dlt_t - (i_dlt_t_h*3600) - (i_dlt_t_m*60);

	str.Format("%i h %i m %i s", i_dlt_t_h, i_dlt_t_m, i_dlt_t_s);
	SetDlgItemText(IDC_EDITRA4, str);


	double c_dlt_t = a_c/cos(a_dlt);

	int c_dlt_t_h = c_dlt_t/3600;
	int c_dlt_t_m = c_dlt_t/60;
	int c_dlt_t_s = c_dlt_t - (c_dlt_t_h*3600) - (c_dlt_t_m*60);

	str.Format("%i h %i m %i s", c_dlt_t_h, c_dlt_t_m, c_dlt_t_s);
	SetDlgItemText(IDC_EDITRA5, str);

	double sum_dlt_t = a_dlt_T+(((a_dlt_A*cos(a_phi)*sin(a_t))-
(a_dlt_phi*cos(a_t))+a_i+(a_b*sin(a_phi)))*tan(a_dlt))
+(((a_bs_p*cos(a_phi)*sin(a_t))+a_c)/cos(a_dlt)) + (a_b*cos(a_phi)*cos(a_t)) +
refr_dlt_t;
	double sum_dlt_dlt = a_dlt_D+(a_dlt_A*cos(a_phi)*cos(a_t))+(a_dlt_phi*sin(a_t))-
(a_bs_h*((sin(a_phi)*cos(a_dlt))-(cos(a_phi)*sin(a_dlt)*cos(a_t))))+refr_dlt_dlt;

	int sum_dlt_t_h = sum_dlt_t/3600;
	int sum_dlt_t_m = sum_dlt_t/60;
	int sum_dlt_t_s = sum_dlt_t - (sum_dlt_t_h*3600) - (sum_dlt_t_m*60);

	int sum_dlt_dlt_h = sum_dlt_dlt/3600;
	int sum_dlt_dlt_m = sum_dlt_dlt/60;
	int sum_dlt_dlt_s = sum_dlt_dlt - (sum_dlt_dlt_h*3600) - (sum_dlt_dlt_m*60);

	str.Format("%i h %i m %i s;    %i h %i m %i s", sum_dlt_t_h, sum_dlt_t_m,
sum_dlt_t_s, sum_dlt_dlt_h, sum_dlt_dlt_m, sum_dlt_dlt_s);
	SetDlgItemText(IDC_EDITRA6, str);
```

```
	str.Format("%i h %i m %i s;    %i h %i m %i s", m_rah+sum_dlt_t_h,
m_ram+sum_dlt_t_m, m_ras+sum_dlt_t_s, m_decg+sum_dlt_dlt_h, m_decm+sum_dlt_dlt_m,
m_decs+sum_dlt_dlt_s);
	SetDlgItemText(IDC_EDITRA7, str);

// Строка 627
m_send = coderObj.CoderFromRaDecToString(m_rah+sum_dlt_t_h, m_ram+sum_dlt_t_m,
m_ras+sum_dlt_t_s, m_decg+sum_dlt_dlt_h, m_decm+sum_dlt_dlt_m, m_decs+sum_dlt_dlt_s, 1,
m_checRefr);

//Строка 639
m_send = coderObj.CoderFromRaDecToString(m_rah+sum_dlt_t_h, m_ram+sum_dlt_t_m,
m_ras+sum_dlt_t_s, m_decg+sum_dlt_dlt_h, m_decm+sum_dlt_dlt_m, m_decs+sum_dlt_dlt_s, 2,
m_checRefr);


	//EASC
```

Заключение

В рамках данной работы было сделано следующее:

1. Освоен язык программирования общего назначения C++ в объёме, необходимом для реализации данного проекта, а также интегрированная среда разработки (IDE) Microsoft Visual Studio 2010;
2. Изучена монтировка Synta Sky-Watcher EQ6 Pro и освоен принцип управления ею;
3. Изучены системы координат, использующиеся для наведения телескопов (в частности – экваториальную с.к.);
4. Изучено программное обеспечение из первого цикла доработки;
5. Доработано ПО наведения телескопа и реализовано наведение по экваториальным координатам с учётом всех необходимых поправок и погрешностей.

Поставленные задачи были выполнены в полном объёме, цель достигнута. Результатом работы стало усовершенствованное ПО управления монтировкой EQ6 в режиме удалённого доступа, обеспечивающее более точное наведение на объекты исследования.

Литература

1. Sky-Watcher [электронный ресурс]. – Режим доступа: http://www.skywatcher.com/_english/02_mounts/02_detail.php?sid=58, свободный – Яз. англ. (Дата обращения: 12.10.1015)
2. Instruction manual SynScan. – URL: http://www.skywatcher.com/upfiles/en_download_caty01312941979.pdf, свободный (Дата обращения: 3.10.1015)
3. NexStar Communication Protocol. – URL: http://www.nexstarsite.com/PCControl/ProgrammingNexStar.htm, свободный (Дата обращения: 8.10.1012)
4. Microsoft Foundation Classes [электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft_Foundation_Classes, свободный (Дата обращения: 15.11.2015)
5. Страуструп Б. Язык программирования C++ / Б. Страуструп. – Пер. с англ. – М: Бином - 2011г. - 1045 с.
6. Лафоре Р. Объектно-ориентированное программирование в C++, 4-е издание / Р. Лафоре. – Пер. с англ. – СПб.: Питер - 2004г. - 962 с.
7. Михельсон Н. Н. Оптические телескопы. Теория и конструкция, 1-е издание / Н. Н. Михельсон. – М.: Издательство «Наука», 1976г., С.336-384