**Лабораторная работа № 2**

Глущенко Сергей Юрьевич

Единые вычислительные алгоритмы (ЕВА) разработаны на языке программирования C++ и являются кроссплатформенным специальным программным обеспечением, предназначенным для внутреннего пользования в отделе.

ЕВА поставляются в виде исходных текстов классов C++. Структурно и функционально ЕВА разделены на 8 отдельных движков. Каждый движок представляет собой иерархию классов, связанных отношениями наследования и композиции. Имеют место случаи применения множественного наследования и полиморфизма. Встречаются абстрактные и статические классы.

Все классы, входящие в состав ЕВА, логически можно разделить на две группы: нетерминальные (в пользовательской программе нет смысла создавать от него объект) и терминальные (пользователь может создавать от него объект и наслаждаться всем богатством функционала).

Термин «движок» применительно к ЕВА не следует рассматривать как синоним фреймворка. Движки ЕВА по большому счету являются библиотеками. Тем не менее, в силу полноты объема решаемых задач и по причине «так исторически сложилось» библиотеки именуются движками.

Названные вычислительные алгоритмы получили наименование «единые» в силу универсальной применимости. Помимо того, что можно пользоваться исходными кодами ЕВА, включая соответствующие файлы в свой проект, над ЕВА построено несколько оболочек (DLL) для применения в C++-приложениях и в Delphi-приложениях.

Исходные коды ЕВА транслировались, линковались и тестировались в операционных системах MS Windows, MCBC, OC2000, Astra Linux следующими компиляторами: C++ Builder, MS Visual C++, MinGV, G++, GCC.

Некоторые алгоритмы ЕВА используют кроссплатформенные средства распараллеливания вычислений OpenMP. Для работы средств требуется установить флаг компилятора –openmp или –fopenmp, а также установить флаг линковщика –fopenmp (зависит от самого компилятора).

Ниже приведена часть описания (фрагмент) одного из движков ЕВА.



Рисунок 1 – Диаграмма классов

Состоит из 11 классов: TGLB, TParDv, TShellPDCM, TTasks, TMatPlan, TMPlan, TTakeTarget, TZRVNIP, TTHBLs, TAntennaTakeTarget, TShellAT.

Реализует следующие прикладные баллистические задачи:

* расчёт ПДЦМ КА;
* расчёт трассы полета, полосы обзора, освещённости;
* расчёт ЗРВ между КА и НИП;
* целеуказание наземным антенным системам;
* расчёт матрицы плнирования;
* целеуказание наземным антенным системам с учетом характеристик антенны.

## ***1. TGLB базовый класс движка***

Нетерминальный класс. Предка не имеет. Прямым потомком является TParDv. Содержит декларации типов и перечень моделей движения центра масс (ЦМ) КА.

Публичные типы:

//Кол-во точек интегрирования

**typedef** **TParamInt** **TVitokBaz**[100000];

//Структура описывает точку орбиты t, Vx, Vy, Vz, x, y, z

**struct** **TParRecStruct**

{

TParRecStruct() : t(0.0), vx(0.0), vy(0.0), vz(0.0), x(0.0), y(0.0), z(0.0) {}

**double** t, vx, vy, vz, x, y, z;

};

**struct** **TNUStruct**

{

TNUStruct() : NVit(0), T(0.0), Rx(0.0), Ry(0.0), Rz(0.0), Vx(0.0), Vy(0.0), Vz(0.0), S(0.0) {}

**long** **int** NVit; //Номер витка, к которому отнесены НУ

**double** T; //Интегральное время, к которому отнесены НУ

**double** Rx; //Проекция радиус-вектора ЦМ КА на ось OX

**double** Ry; //Проекция радиус-вектора ЦМ КА на ось OY

**double** Rz; //Проекция радиус-вектора ЦМ КА на ось OZ

**double** Vx; //Проекция вектора скорости ЦМ КА на ось OX

**double** Vy; //Проекция вектора скорости ЦМ КА на ось OY

**double** Vz; //Проекция вектора скорости ЦМ КА на ось OZ

**double** S; //Баллистический коэффициент [м3/(кг·с2)]

};

**typedef** **struct** **TNUStruct** **TNU**;

//Слои атмосферы

**typedef** **double** **TAtmSloi**[9];

**struct** **TAtmConsStruct**

{

**TAtmSloi** ht, a, k1, k2;

};

//Параметры слоя атмосферы

**typedef** **struct** **TAtmConsStruct** **TAtmCons**;

**struct** **TOskulElemStruct**

{

TOskulElemStruct() : ael(0.0), e(0.0), nakl(0.0), omegab(0.0), omegam(0.0) {}

**double** ael; //Большая полуось

**double** e; //Эксцентриситет

**double** nakl; //Наклонение плоскости орбиты к плоскости экватора

**double** omegab; //Долгота восходящего узла (омега большое)

**double** omegam; //Аргумент перигея

};

**typedef** **struct** **TOskulElemStruct** **TOskulElem**; //Оскулирующие элементы

//Структура записи результатов счета PARDV

**struct** **TRezPARDVStruct**

{

TRezPARDVStruct () : NVit(0), S(0.0), L(0.0), H(0.0), TOB(0.0), NvSut(0), ArgSh(0.0), BettaSoln(0.0) {}

**long** **int** NVit; //Номер витка НУ

**TParRec** Otn; //НУ в Относительной системе координат

**double** S; //Баллистический коэффициент [м3/(кг·с2)]

**TParRec** Absk; //НУ в Абсолютной системе координат

**double** L; //L

**double** H; //Высота

**double** TOB; //Период обращения

**long** **int** NvSut; //Суточный виток

**TOskulElem** OsclEl; //Оскулирующие элементы

**TExtrElemRec** HMIN; //Элементы минимальной высоты

**TExtrElemRec** HMAX; //Элементы максимальной высоты

**TExtrElemRec** NISX; //Элементы нисходящего узла

**TExtrElemRec** TP; //Элементы перигея

**double** ArgSh; //Аргумент широты - угол от направления на ВУЗ до радиус-вектора

**double** BettaSoln; //Угол "Солнце - плоскость орбиты"

};

//Структура записи результатов счета PARDV

**typedef** **struct** **TRezPARDVStruct** **TRezPARDV**;

//Структура результатов расчета параметров трассы полета и полосы обзора

**struct** **TTHBLStruct**

{

TTHBLStruct() : t(0.0), h(0.0), b(0.0), l(0.0), hs(0.0) {}

**double** t; //Время

**double** h; //Высота или наклонная дальность

**double** b; //Широта

**double** l; //Долгота

**double** hs; //Высота Солнца

};

//Структура результатов счета THBL

**typedef** **struct** **TTHBLStruct** **TTHBL**;

**struct** **TParRecPDCMStruct**

{

TParRecPDCMStruct() : NVit(0), t(0.0), Vx(0.0), Vy(0.0), Vz(0.0), x(0.0), y(0.0), z(0.0), BolPO(0.0), E(0.0), Nak(0.0), DVUz(0.0), ArgPer(0.0), ArgSh(0.0), S(0.0) {}

**long** **int** NVit;

**double** t, Vx, Vy, Vz, x, y, z;

**double** BolPO; //Большая полуось орбиты

**double** E; //Эксцентриситет орбиты

**double** Nak; //Наклонение плоскости орбиты к плоскости экватора

**double** DVUz; //Долгота

**double** ArgPer; //Аргумент перигея

**double** ArgSh; //Аргумент широты (угол от направления на ВУЗ до радиус-вектора)

**double** S; //Баллистический коэффициент [м3/(кг·с2)]

**TTHBL** ParVix; //Параметры подспутниковой точки

**TIntMas** a7Vix; //Значения ускорений в 7 разгонных точках

};

**typedef** **struct** **TParRecPDCMStruct** **TParRecPDCM**;

//Структура записи результатов расчета параметров трассы полета и полосы обзора

**struct** **TRezTHBL**

{

TRezTHBL() : NVit(0) {}

**long** **int** NVit; //Номер витка

**TTHBL** THBL[3]; //Параметры трассы (индекс 0), левой границы (индекс 1) и правой границы (индекс 2) полосы обзора

};

**typedef** **struct** **TRezTHBL** **TRezTHBL**;

//Результаты расчета зон освещенности полос обзора по точкам (для одной точки)

**struct** **TRezOsvPol\_Point**

{

TRezOsvPol\_Point() : NVitNacZonOsv(0), NVitVremy(0), T(0.0), BLeft(0.0), LLeft(0.0), BRight(0.0), LRight(0.0) {}

**long** **int** NVitNacZonOsv; //Номер витка начала зоны

**long** **int** NVitVremy; //Номер витка, соответствующий данному моменту времени

**double** T; //Интегральное время

**double** BLeft; //Широта левой границы

**double** LLeft; //Долгота левой границы

**double** BRight; //Широта правой границы

**double** LRight; //Долгота правой границы

};

**typedef** **struct** **TRezOsvPol\_Point** **TRezOsvPol\_Point**;

//Результаты расчета зон освещенности полос обзора по зонам (для одной зоны)

**struct** **TMassRezOsvPol\_Zone**

{

**std**::**vector** <**TRezOsvPol\_Point**> MassRezOsvPol\_Zone; //Массив точек зоны

};

**typedef** **struct** **TMassRezOsvPol\_Zone** **TMassRezOsvPol\_Zone**;

//Результаты расчета характеристик зоны освещенности КА (для одной зоны)

**struct** **TRezOsvKA\_Point**

{

TRezOsvKA\_Point() : NVitNac(0), NVitKon(0), TNac(0.0), TKon(0.0) {}

**long** **int** NVitNac; //Номер витка начала зоны

**long** **int** NVitKon; //Номер витка конца зоны

**double** TNac; //Время начала зоны

**double** TKon; //Время конца зоны

};

**typedef** **struct** **TRezOsvKA\_Point** **TRezOsvKA\_Point**;

//Структура для записи значений освещенности КА в каждой точке интегр-я на витке

**struct** **TPointVit\_OsvKA**

{

TPointVit\_OsvKA() : T(0.0), NVit(0), OsvKA(0.0) {}

**double** T; //Время точки

**long** **int** NVit; //Виток точки

**double** OsvKA; //Значение освещенности

};

**typedef** **struct** **TPointVit\_OsvKA** **TPointVit\_OsvKA**;

//Значения по всем точкам интегрирования

**struct** **TMassPointVit\_OsvKA**

{

**std**::**vector** <**TPointVit\_OsvKA**> PointVit\_OsvKA;

};

**typedef** **struct** **TMassPointVit\_OsvKA** **TMassPointVit\_OsvKA**;

//Параметры видимости

**struct** **TParVidStruct**

{

TParVidStruct() : T(0.0), DN(0.0), Azim(0.0), Betta(0.0) {}

**double** T; //Время

**double** DN; //Дальность

**double** Azim; //Азимут

**double** Betta; //Бета

**TParRec** NUDZona; //НУ на время расчета параметров видимости

};

**typedef** **struct** **TParVidStruct** **TParVid**;

//Исходные параметры НИПа

**struct** **TParNIPStruct**

{

TParNIPStruct() : NNIP(0), bn(0.0), xn(0.0), yn(0.0), zn(0.0), sL(0.0), cL(0.0), sB(0.0), cB(0.0) {}

**long** **int** NNIP; //Номер НИП

**double** bn, xn, yn, zn; //Геоцентрические координаты

**double** sL, cL, sB, cB; //Синус и косинус B, L

};

**typedef** **struct** **TParNIPStruct** **TParNIP**;

//Постоянные параметры для НИПа

**struct** **TConstNIP**

{

TConstNIP() : NNIP(0), B(0.0), L(0.0), H(0.0) {}

**long** **int** NNIP; //Номер НИП

**double** B; //Широта

**double** L; //Долгота

**double** H; //Превышение

};

**typedef** **struct** **TConstNIP** **TConstNIPs**;

//Параметры зоны видимости

**struct** **TParZonyStruct**

{

**TParVid** Vxod, Max, Vix, VxodMZ, VixMZ; //Вход, Максимум, Выход, Вход в МЗ, Выход из МЗ

};

**typedef** **struct** **TParZonyStruct** **TParZony**;

//Смежные точки интегрирования для НИПа

**struct** **TPamNIPStruct**

{

TPamNIPStruct() : t1(0.0), gam1(0.0), Vy1(0.0), t2(0.0), gam2(0.0), Vy2(0.0), GminNIP(0.0), GmaxNIP(0.0) {}

**TParNIP** ParNIP;

**TParRec** ParSat;

**TParZony** ParZony;

**double** t1, gam1, Vy1, t2, gam2, Vy2;

**double** GminNIP; //Минимальный угол места АУ ППИ

**double** GmaxNIP; //Максимальный угол места АУ ППИ

};

**typedef** **struct** **TPamNIPStruct** **TPamNIP**;

//Структура результатов расчета зон радиовидимости

**struct** **TRezZRVNStruct**

{

TRezZRVNStruct() : NVitNac(0), NVitKon(0), NNIP(0), Dlit(0.0) {}

**long** **int** NVitNac; //Номер витка начала зоны

**long** **int** NVitKon; //Номер витка конца зоны

**long** **int** NNIP; //Номер НИП

**TParZony** ParZ; //Параметры всей зоны видимости

**double** Dlit; //Длительность зоны

};

**typedef** **struct** **TRezZRVNStruct** **TRezZRVNS**;

//Структура параметров НИП

**struct** **TPnktStruct**

{

TPnktStruct() : NomPnkt(0), MinUglPnkt(0.0), MaxUglPnkt(0.0), MinDltZRV(0), L(0.0), B(0.0), H(0.0) {}

**long** **int** NomPnkt; //Номер i-го ППИ(КИП)

**double** MinUglPnkt; //Пред. мин. угол места при кот. сохраняется радиовид. с i‑ым ППИ(КИП), [рад]

**double** MaxUglPnkt; //Пред. макс. угол места при кот. сохраняется радиовид. с i-ым ППИ(КИП), [рад]

**long** **int** MinDltZRV; //Минимально допустимая длительность ЗРВ КА с i-тым ППИ(КИП), [с]

**double** L; //Долгота i-го ППИ(КИП), [рад]

**double** B; //Геодезическая широта i-го ППИ(КИП), [рад]

**double** H; //Превышение i-го ППИ(КИП) над ОЗЭ, [м]

};

**typedef** **struct** **TPnktStruct** **TPnkt**;

//Результат расчёта Целеуказания для текущей точки

**struct** **TargetResultStruct**

{

TargetResultStruct() : Azimuth(0.0), CornerOfPlace(0.0), InclinedRange(0.0), DerivativeOfInclinedRange(0.0), NVit(0), RadioVisibility(0) {}

**double** Azimuth; //Азимут, [град]

**double** CornerOfPlace; //Угол места, [град]

**double** InclinedRange; //Наклонная дальность от локатора до КА, [м]

**double** DerivativeOfInclinedRange; //Производная от наклонной дальности, [м/с]

**long** **int** NVit; //Номер витка

**TParRec** ParMove; //Параметры движения КА

**long** **int** RadioVisibility; //Признак нахождения в зоне радио видимости: 0-есть, 1-в 7-ой зоне, 2-в "мертвой" зоне

};

**typedef** **struct** **TargetResultStruct** **TargetResult**;

**struct** **BLHLocatorStruct**

{

BLHLocatorStruct() : L(0.0), B(0.0), H(0.0) {}

**double** L; //Долгота локатора целеуказания, [рад]

**double** B; //Геодезическая широта локатора целеуказания, [рад]

**double** H; //Превышение локатора целеуказания над ОЗЭ, [м]

};

**typedef** **struct** **BLHLocatorStruct** **BLHLocator**;

**struct** **TPamLocatorStruct**

{

**TParNIP** ParNIP;

**TParRec** ParSat;

};

**typedef** **struct** **TPamLocatorStruct** **TPamLocator**;

//Тип описывает указатель на процедуру CALL BACk

**typedef** **void** (\***THowCalculation**)(**int** index, **int** quantity);

**struct** **TOPKAT**

{

TOPKAT() : Obj\_oh(0), Obj\_n(0), Point\_n(0), BC(0.0), LC(0.0), HC(0.0) {}

**long** **int** Obj\_oh; //Оперативное направление (должно быть больше нуля)

**long** **int** Obj\_n; //Номер объекта внутри оперативного направления (должно быть больше нуля)

**long** **int** Point\_n; //Часть территории объекта (один объект может быть разделен на несколько частей) (должно быть больше нуля)

**double** BC; //Геодезическая широта объекта, [рад]

**double** LC; //Геодезическая долгота объекта, [рад]

**double** HC; //Превышение объекта над общеземным эллипсоидом, [м]

};

**typedef** **struct** **TOPKAT** **TOPKAT**;

**struct** **TRezMPln**

{

TRezMPln() : NVit(0), TTr(0.0), Obj\_oh(0), Obj\_n(0), Point\_n(0), BC(0.0), LC(0.0), HC(0.0), BT(0.0), LT(0.0), Kren(0.0), HSolnca(0.0), DNakl(0.0), PV(0) {}

**long** **int** NVit; //Текущий номер витка

**double** TTr; //Время траверза

**long** **int** Obj\_oh; //Оперативное направление

**long** **int** Obj\_n; //Номер объекта внутри оперативного направления

**long** **int** Point\_n; //Часть территории объекта (один объект может быть разделен на несколько частей)

**double** BC; //Широта цели

**double** LC; //Долгота цели

**double** HC; //Превышение цели над общеземным эллипсоидом

**double** BT; //Широта траверза

**double** LT; //Долгота траверза

**double** Kren; //Угол крена

**double** HSolnca; //Высота Солнца

**double** DNakl; //Наклонная дальность

**long** **int** PV; //Признак ветви 0\1 восходящая\нисходящая

**TParRec** NUTr; //НУ на траверзе

};

**typedef** **struct** **TRezMPln** **TRezMPln**;

**struct** **Corners**

{

Corners() : MinCorner(0.0), MaxCorner(0.0) {}

**double** MinCorner;

**double** MaxCorner;

};

**typedef** **struct** **Corners** **Corners**;

**struct** **AntennaAttributs**

{

**BLHLocator** BLH; //Геодезические координаты антенны, на которую рассчитываются ЦУ

**Corners** MinMax;

};

**typedef** **struct** **AntennaAttributs** **AntennaAttributs**;

Публичный метод:

### *1.1.1. TargetGeoOrbit расчёт целеуказания для заданной точки на Земле при наведении на неподвижный геостационарный спутник*

На вход:

- гринвичские координаты геостационара;

- геодезические координаты антенны наведения.

На выходе:

- азимут антенны;

- угол места антенны.

Синтаксис:

**void** TargetGeoOrbit(**double** X, **double** Y, **double** Z, **double** B, **double** L, **double** H, **double** \*Azimuth, **double** \*Corner)

## ***2. TParDv класс расчёта прогноза движения, выхода на заданное время***

Нетерминальный класс. Предком является TGLB. Прямым потомком является TTasks. Содержит основные алгоритмы решения начальной задачи Коши для системы обыкновенных дифференциальных уравнений, описывающих траекторию полета спутника вокруг Земли.

Публичные методы:

### *2.1. ParDVKA расчет параметров движения центра масс КА на заданном интервале витков*

Синтаксис:

**long** **int** ParDVKA(**void**)

### *2.2. VixTz выход на заданное время методом Рунге-Кутта*

Синтаксис:

**TParRecPDCM** VixTz(**int** \*ReturnCode, **const** **double** tz, **const** **double** step\_int, **const** **double** dh, **const** **double** gkr)

### *2.3. SetNomVitNach установить виток начала диапазона записать*

Синтаксис:

**void** SetNomVitNach(**const** **long** **int** FNVN1)

### *2.4. GetNomVitNach получить виток начала диапазона прочитать*

Синтаксис:

**long** **int** GetNomVitNach(**void**)

### *2.5. SetNomVitKon установить виток конца диапазона записать*

Синтаксис:

**void** SetNomVitKon(**const** **long** **int** FNVK1)

### *2.6. GetNomVitKon получить виток конца диапазона прочитать*

Синтаксис:

**long** **int** GetNomVitKon(**void**)

### *2.7. SetModAndStep установить модель движения и шаг интегрирования*

Синтаксис:

**int** SetModAndStep(**const** **TTypeModDv** FMD, **const** **double** FST)

### *2.8. SetTimesNachKon установить время начала и окончания диапазона расчета*

Синтаксис:

**void** SetTimesNachKon(**const** **double** TStart, **const** **double** TStop)

### *2.9. SetAtm81 установить среднее значение солнечной активности за последние 81 сутки*

Синтаксис:

**void** SetAtm81(**const** **double** Value)

### *2.10. SetNumberDayNU установить номер суток в году. В этих сутках расположены первые НУ из массива НУ*

Синтаксис:

**void** SetNumberDayNU(**const** **double** Value)

### *2.11. VixTzAdams выход на заданное время методом Адамса с правыми частями модели движения*

Синтаксис:

**void** VixTzAdams(**const** **double** Time, **int**\* NVit, **double**\* Rx, **double**\* Ry, **double**\* Rz, **double**\* Vx, **double**\* Vy, **double**\* Vz)

### *2.12. GetModDv возвращет текущую модель движения*

Синтаксис:

**TTypeModDv** GetModDv(**void**)

Публичные свойства:

### *2.13. OWNER свойство указывает на класс-держатель экземпляра объекта TParDv*

Синтаксис:

**void** \*OWNER

### *2.14. MassNU свойство Массив НУ*

Синтаксис:

**std**::**vector** <**TNU**> MassNU

### *2.15. RezInDiapasonVUZ свойство Результаты счета в диапазоне по ВУЗам*

Синтаксис:

**TMassRezParDv** RezInDiapasonVUZ

### *2.16. RezInDiapasonStep свойство Результаты счета в диапазоне по шагам*

Синтаксис:

**TArrayParRec** RezInDiapasonStep

### *2.17. HowCalculation свойство Указатель на функцию обратного вызова.*

Присваивать значение этому полю НЕОБЯЗАТЕЛЬНО

Синтаксис:

**THowCalculation** HowCalculation

## ***3. TShellPDCM класс-оболочка движка***

Терминальный класс, представляющий пользователю основной функционал прогнозирования положения центра масс спутника. Предка не имеет. Включает в себя экземпляр TParDv (создает композицию). Прямым потомком является TShellAT.

Увязывает экземпляр класса TParDv и экземпляр класса, описывающего правую часть соответствующей модели движения.

Не имеет публичных методов, кроме конструктора и деструктора.

Публичные свойства:

### *3.1. ParDv свойство Движок расчета ПДЦМ (экземпляр класса TParDv)*

Синтаксис:

**TParDv** \*ParDv

### *3.2. ResParDv свойство Код ошибки расчета ПДЦМ*

Синтаксис:

**int** ResParDv

## ***4. TTasks абстрактный класс-предок всех прикладных баллистических задач***

Нетерминальный класс. Предком является TParDv. Прямыми потомками являются TMatPlan, TMPlan, TTakeTarget, TZRVNIP, TTHBLs.

Кроме конструктора и чисто виртуального деструктора, публичных методов и свойств не имеет.

## ***5. TMatPlan класс расчёта параметров точки траверза***

Нетерминальный класс. Предком является TTasks. Прямых потомков не имеет.

Публичные методы:

### *5.1. SetB задать широту точки траверза*

Синтаксис:

**void** SetB(**const** **double** Value)

### *5.2. GetB прочитать широту точки траверза*

Синтаксис:

**double** GetB(**void**)

### *5.3. SetL задать долготу точки траверза*

Синтаксис:

**void** SetL(**const** **double** Value)

### *5.4. GetL прочитать долготу точки траверза*

Синтаксис:

**double** GetL(**void**)

### *5.5. SetH задать превышение точки траверза над Общеземным эллипсоидом*

Синтаксис:

**void** SetH(**const** **double** Value)

### *5.6. GetH прочитать превышение точки траверза над ОЗЭ*

Синтаксис:

**double** GetH(**void**)

### *5.7. Traverz вычислить параметры в точке траверза*

Синтаксис:

**long** **int** Traverz(**void**)

Публичные свойства:

### *5.8. PravTochn свойство Признак расчета с использованием правых частей СДУ*

Синтаксис:

**bool** PravTochn

### *5.9. OutNU свойство Начальные условия в точке траверза*

Синтаксис:

**TGLB**::**TNU** OutNU

## ***6. TMPlan класс расчёта матрицы планирования***

Терминальный класс. Предком является TTasks. Прямых потомков не имеет. Композицией включает в себя экземпляр TMatPlan, но он не доступен пользователю напрямую.

Имеет единственный публичный метод:

### *6.1. RunMatPlan расчет параметров матрицы планирования*

Синтаксис:

**long** **int** RunMatPlan(**void**)

Публичные свойства:

### *6.2. PravTochn свойство Признак расчета выхода на время траверза по правым частям КА или по общим правым частям*

Синтаксис:

**bool** PravTochn

### *6.3. GamMin свойство Минимальный угол крена КА*

Синтаксис:

**double** GamMin

### *6.4. GamMax свойство Максимальный угол крена КА*

Синтаксис:

**double** GamMax

### *6.5. Dmin свойство Минимальная дальность до объекта наблюдения*

Синтаксис:

**double** Dmin

### *6.6. Dmax свойство Максимальная дальность до объекта наблюдения*

Синтаксис:

**double** Dmax

### *6.7. HSolMin свойство Минимальная высота Солнца*

Синтаксис:

**double** HSolMin

### *6.8. Katalog свойство Каталог объектов наблюдения*

Синтаксис:

**std**::**vector** <**TGLB**::**TOPKAT**> Katalog

### *6.9. ResultMP свойство Результаты расчета матрицы планирования в диапазоне витков*

Синтаксис:

**std**::**vector** <**TGLB**::**TRezMPln**> ResultMP

## ***7. TTakeTarget класс расчёта целеуказания наземным антенным системам***

Терминальный класс. Предком является TTasks. Прямым потомком является TAntennaTakeTarget.

Имеет единственный публичный метод:

### *7.1.* Start *расчет целеуказания*

Синтаксис:

**long** **int** Start(**TGLB**::**BLHLocator** &Pnkt)

Имеет единственное публичное свойство:

### *7.2. TargetInDiapason свойство Вектор результатов расчета целеуказания в диапазоне*

Синтаксис:

**std**::**vector**<**TGLB**::**TargetResult**> TargetInDiapason

## ***8. TZRVNIP класс расчёта ЗРВ между КА и НИП***

Терминальный класс. Предком является TTasks. Прямым потомком является TAntennaTakeTarget.

Имеет единственный публичный метод:

### *8.1.* ZoneZRV *расчет параметров ЗРВ между КА и НИП*

Синтаксис:

**long** **int** ZoneZRV(**std**::**vector**<**TGLB**::**TPnkt**> &Pnkt)

Имеет единственное публичное свойство:

### *8.2. ZRVInDiapason свойство Вектор результатов расчета ЗРВ в диапазоне*

Синтаксис:

**std**::**vector**<**TGLB**::**TRezZRVNS**> ZRVInDiapason

## ***9. TTHBLs класс расчёта трассы полета, полосы обзора, освещённости***

Терминальный класс. Предком является TTasks. Прямых потомков не имеет.

Публичные типы:

//Тип используется при расчёте параметров освещенности полосы обзора

**struct** **ZoneOsvPol**

{

ZoneOsvPol() : ActiveRun(**false**), MinUglSln(0.0), MinDlitOsv(0) {}

**bool** ActiveRun;

**double** MinUglSln;

**long** **int** MinDlitOsv;

};

**typedef** **struct** **ZoneOsvPol** **TZoneOsvPol**;

Публичные методы:

### *9.1. Start главный метод класса по запуску расчёта*

Синтаксис:

**long** **int** Start(**void**)

### *9.2.* SetGamKrenMax *установить максимальный угол крена спутника*

Синтаксис:

**long** **int** SetGamKrenMax(**const** **double** Value)

### *9.3. G*etGamKrenMax *прочитать максимальный угол крена спутника*

Синтаксис:

**double** GetGamKrenMax(**void**)

Публичные свойства:

### *9.4. ZoneOsvPol свойство Параметры освещённости полосы обзора*

Синтаксис:

**TZoneOsvPol** ZoneOsvPol

### *9.5. LineStrip свойство Результаты расчёта трассы и границ полосы обзора (на диапазон витков)*

Синтаксис:

**std**::**vector** <**TGLB**::**TRezTHBL**> LineStrip

### *9.6. IlluminationStrip свойство Результаты расчёта характеристик всех зон освещённости, входящих в полосу обзора (на диапазон витков)*

Синтаксис:

**std**::**vector** <**TGLB**::**TMassRezOsvPol\_Zone**> IlluminationStrip

### *9.7. IlluminationSpaceShip свойство Результаты расчёта характеристик всех зон освещённости КА (на диапазон витков)*

Синтаксис:

**std**::**vector** <**TGLB**::**TRezOsvKA\_Point**> IlluminationSpaceShip

## ***10. TAntennaTakeTarget класс расчёта целеуказания наземным антенным системам с учётом характеристик антенны***

Терминальный класс. Предками при множественном наследовании являются TTakeTarget, TZRVNIP. Прямых потомков не имеет.

Имеет единственный публичный метод:

### *10.1.* StartATT *расчет целеуказания с учётом характеристик антенны*

Синтаксис:

**long** **int** StartATT(**TGLB**::**AntennaAttributs** &AA)

Имеет единственное публичное свойство:

### *10.2. AntennaTargetInDiapason свойство Вектор результатов расчета целеуказания в диапазоне*

Синтаксис:

**std**::**vector**<**TGLB**::**TargetResult**> AntennaTargetInDiapason

## ***11. TShellAT класс-оболочка движка***

Терминальный класс, представляющий пользователю основной функционал прикладных баллистических задач. Предком является TShellPDCM. Прямых потомков не имеет.

Композицией включает в себя экземпляры TAntennaTakeTarget, TTHBLs, TTakeTarget, TZRVNIP, TMPlan. Эти экземпляры доступны пользователю в виде свойств. Публичных методов, кроме конструкторов и деструктора не имеет.

Каждый из вышеназванных экземпляров классов решает свою прикладную баллистическую задачу.

Решение задачи расчета ПДЦМ КА обеспечивается функционалом класса TShellPDCM, который является предком TShellAT.

Каждая из шести прикладных баллистических задач не зависит от других задач и является полностью автономной.

Публичные свойства:

### *11.1. LineStrip свойство Расчёт трассы полета, полосы обзора, освещённости (экземпляр класса TTHBLs)*

Синтаксис:

**TTHBLs** \*LineStrip

### *11.2. ZRVNIP свойство Расчёт зон радиовидимости между КА и НИП (экземпляр класса TZRVNIP)*

Синтаксис:

**TZRVNIP** \*ZRVNIP

### *11.3. CU свойство Расчёт целеуказания наземным антенным системам (экземпляр класса TTakeTarget)*

Синтаксис:

**TTakeTarget** \*CU

### *11.4. MP свойство Расчет матрицы планирования (экземпляр класса TMPlan)*

Синтаксис:

**TMPlan** \*MP

### *11.5. ACU свойство Расчёт целеуказания наземным антенным системам с учётом характеристик антенны (экземпляр класса TAntennaTakeTarget)*

Синтаксис:

**TAntennaTakeTarget** \*ACU