Проекция положения спутника Земли на ее поверхность называется подспутниковой точкой. Если предположить, что Земля – шар, то подспутниковая точка лежит на пересечении вектора, проведенного от центра масс Земли к центру масс КА, с поверхностью Земли.

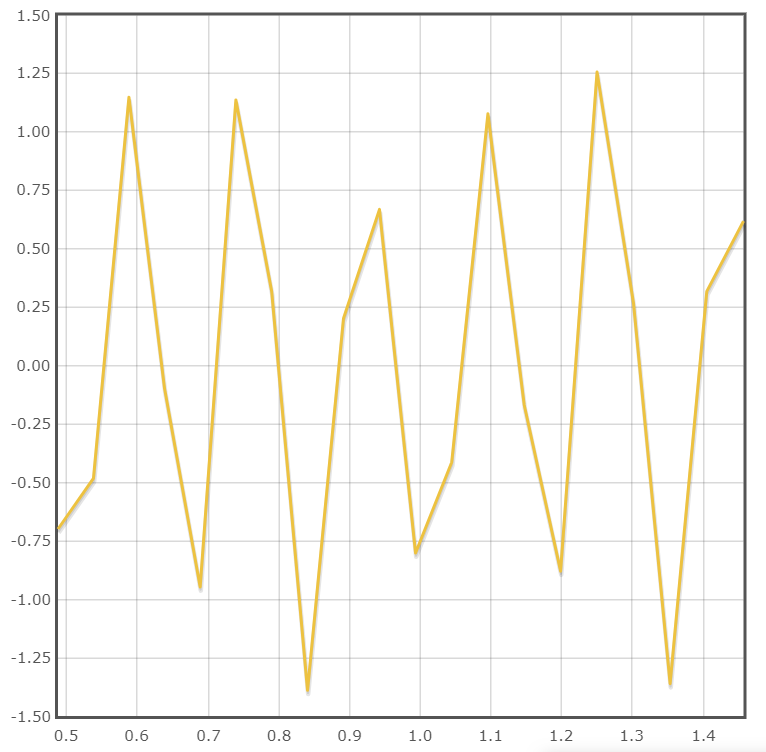
Трасса – это совокупность подспутниковых точек. При расчете трассы определяют графические координаты: долготу λ и широту ϕ геоцентрических подспутниковых точек. По их значениям строят зависимость.

Подробнее теорию по данной задаче читай в RSSS\_lekciya\_13. Из не будет все понятно.

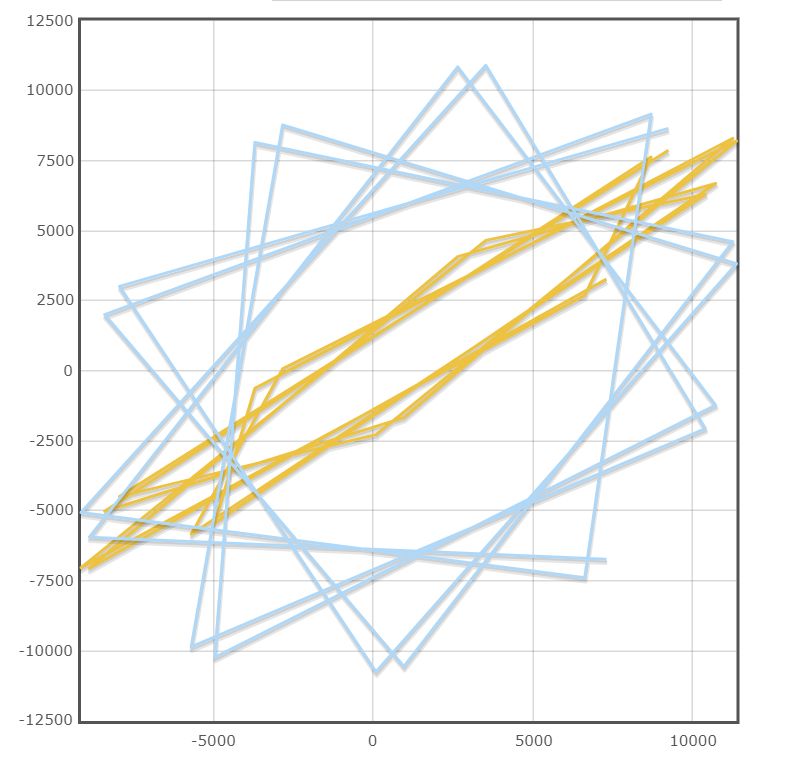
Пример.txt - Файл хранит данные которые я использовал для построения трассы. Данные получены с помощью программы.

Если захочешь построить график по этим точкам, то вот:

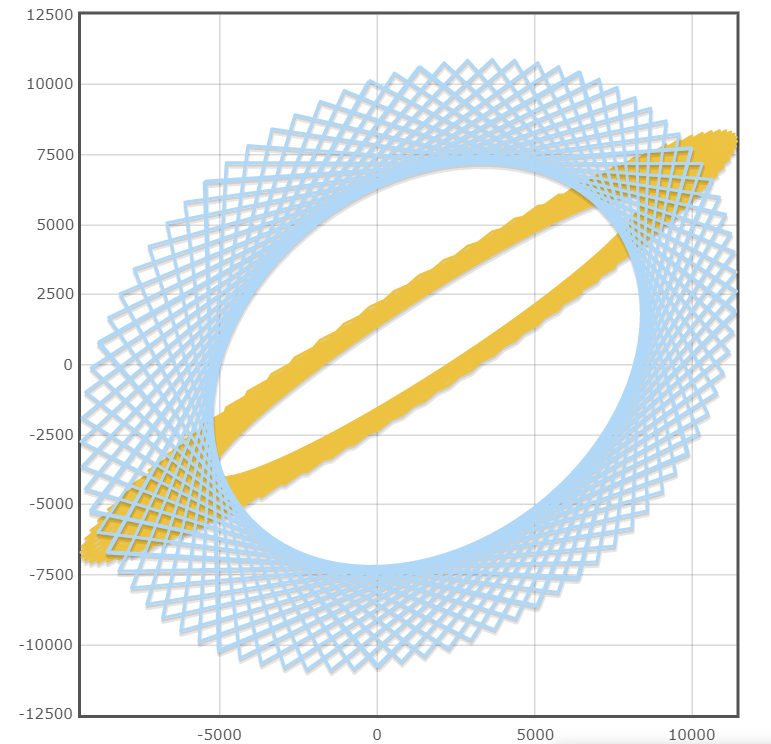
(Это сама трасса)



Вот такой график выдает, когда строю в координатном пространстве:

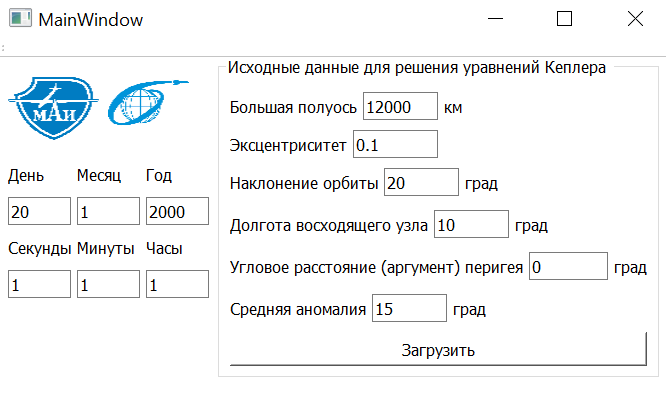


Если взять больше точек:



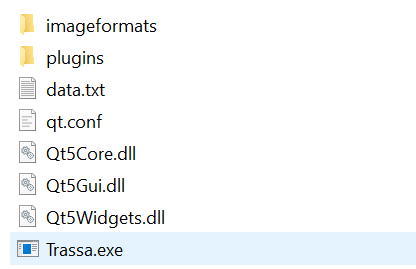
Вот ссылка где строил <https://www.kontrolnaya-rabota.ru/s/grafik/tochka/>

Примерно так будет выглядеть заполненная программа для твоего варианта, дату ты выбираешь и ставишь сам. Для примера я брал даты с периодом в день. Каждый раз, когда нажимаешь на кнопку загрузки, в файл data.txt записываются рассчитанные для этой даты параметры.



Если ты закроешь приложение, а потом опять его запустишь то вся старая информация сотрётся из файла.

Чтобы запустить приложение, нажми Trassa.exe.



Теперь по коду. Я расписал максимально подробно, но если будут вопросы ты все ровно пиши.

*//* *Здесь* *идет* *описание* *интерфейса,* *это* *для* *тебя* *не* *особо* *важно*

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QPixmap>

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

#include <windows.h>

#include <locale.h>

#define \_USE\_MATH\_DEFINES

#include <math.h>

# define M\_PI 3.14159265358979323846 */\** *pi* *\*/*

*using* *namespace* std;

MainWindow::**MainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

*this*->setStyleSheet("background-color: white;");

QPixmap pix(":/resource/img/MAI.jpg");

QPixmap pix2(":/resource/img/6.jpg");

int w = ui->image->width();

int h = ui->image->height();

int w\_2 = ui->image\_2->width();

int h\_2 = ui->image\_2->height();

ui->image->setPixmap(pix.scaled(w,h,Qt::*KeepAspectRatio*));

ui->image\_2->setPixmap(pix2.scaled(w\_2,h\_2,Qt::*KeepAspectRatio*));

ofstream output("data.txt");

output << "Долгота Широта Дата Координата\n"

"подспутниковой подспутниковой с/мин/час/день/месяц/год x,y,z\n"

"точки точки" << endl;

output.close();

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

*//* *Тут* *вся* *математика*

void MainWindow::**on\_pushButton\_clicked**()

{

setlocale(LC\_ALL, "Rus");

SetConsoleCP(1251);

ofstream output("data.txt", ios\_base::app); *//* *Файл* *хранит* *необходимые* *по* *заданию* *данные*

*//* *Координаты* *КА* *в* *оределенный* *момент* *времени*

double Xa = 0, Ya = 0, Za = 0;

*//* *Твое* *задание*

double a = 0,*//12000,*

e = 0,*//0.1,*

i = 0,*//20,*

OM = 0,*//10,*

om = 0,*//0*

M0 = 0;*//15;*

*//* *OM* *и* *om* *-* *омега* *большое* *и* *омега* *малое*

double

ra = 0, *//* *Модуль* *радиус* *-* *вектора* *КА* *в* *ИСК*

u = 0, *//* *Аргумент* *широты* *орбиты*

fita = 0, *//* *Истинная* *аномалия* *КА* *на* *орбите,* *буква* *фита*

Ei = 0, Ei1 = 0, *//* *эксцентрическая* *аномалия*

M = 0, *//* *Изменение* *средней* *аномалии* *во* *времени*

t = 0, *//* *Время* *t* *конечное* *max* *43334* *-* *период* *обращения* *донного* *КА* *относительно* *земли*

n = 0, *//* *Cреднее* *движение*

y = 0, *//* *Просто* *переменная*

*//* *Константы*

A = -19089.451590, B = 8640184.812866, C = 0.093104, D = -0.0000062,

JD0 = 2451545, JDD = 36525, DS2R = 727220.5216643,

f = 0, alpha = 0, *//* *Переменные* *для* *определения* *гринвеческого* *угла*

JD = 0, *//* *Юлианская* *дата*

fi = 0, *//* *Широта* *подспутниковой* *точки*

lambda = 0; *//* *Долгота* *подспутниковой* *точки*

int

year = 0, month = 0, day = 0, *//* *Текущий* *год*

hour = 0, min = 0, sec = 0; *//* *Текущее* *время*

*//* *Эта* *часть* *нужна* *для* *того* *чтобы* *считывать* *информацию*

QString lineEdit\_3 = ui->lineEdit\_3->text();

QString lineEdit\_5 = ui->lineEdit\_5->text();

QString lineEdit\_6 = ui->lineEdit\_6->text();

QString lineEdit\_7 = ui->lineEdit\_7->text();

QString lineEdit\_8 = ui->lineEdit\_8->text();

QString lineEdit\_9 = ui->lineEdit\_9->text();

QString lineEdit\_4 = ui->lineEdit\_4->text();

QString lineEdit = ui->lineEdit->text();

QString lineEdit\_2 = ui->lineEdit\_2->text();

QString lineEdit\_10 = ui->lineEdit\_10->text();

QString lineEdit\_11 = ui->lineEdit\_11->text();

QString lineEdit\_12 = ui->lineEdit\_12->text();

sec=lineEdit\_5.toDouble();

min=lineEdit\_3.toDouble();

hour = lineEdit\_6.toDouble();

day =lineEdit\_7.toDouble();

month =lineEdit\_8.toDouble();

year = lineEdit\_9.toDouble();

a = lineEdit\_4.toDouble();

e = lineEdit.toDouble();

i = lineEdit\_2.toDouble();

OM = lineEdit\_10.toDouble();

om = lineEdit\_11.toDouble();

M0 = lineEdit\_12.toDouble();

*//* *Параметры* *для* *того* *чтобы* *не* *допустить* *ошибку* *в* *дате*

*if* (sec <=0 || min <=0 || hour <=0 || day <=0||month<=0||year<=0)

{

output << "Ошибка в знаке или 0; ";

output << sec << "." << min << "." << hour << "/" << day <<"."<<month <<"."<< year<< endl;

}

*else* *if* (sec > 60 || min >60 || hour > 24 || month > 12)

{

output << "Ошибка во времени; ";

output << sec << "." << min << "." << hour << "/" << day <<"."<<month <<"."<< year<< endl;

}

*else* *if* (((month == 1||month == 3||month ==5||month ==7||month ==8||month ==10||month ==12)&(day>31))||

((month == 4||month == 6||month ==9||month ==11)&(day>30)))

{

output << "Ошибка в дате; ";

output << sec << "." << min << "." << hour << "/" << day <<"."<<month <<"."<< year<< endl;

}

*else* *if* (((year%4==0&year%100!=0)||year%400==0)&month==2&day>28)

{

output << "Ошибка в дате, год весокосный не больше 28; ";

output << sec << "." << min << "." << hour << "/" << day <<"."<<month <<"."<< year<< endl;

}

*else* *if* (year%4!=0&month==2&day>29)

{

output << "Ошибка в дате, год не весокосный не больше 29; ";

output << sec << "." << min << "." << hour << "/" << day <<"."<<month <<"."<< year<< endl;

}

*else*

{

*//* *Тут* *вся* *математика*

*//* *Вычисление* *Юлианской* *даты*

y = (year + 4800 - ((14 - month) / 12));

JD = (day + ((153 \* (month + 12 \* ((14 - month) / 12) - 3) + 2) / 5) + 365 \* y + y / 4 - y / 100 + y / 400 - 32045) + (hour - 12) / 24 + min / 1440 + sec / 86400;

*//* *Определим* *гринвичский* *угол* *по* *соотношениям*

t = (JD - JD0) / JDD;

*//* *Определим* *гринвечиский* *угол*

f = 86400 + fmod(JD, 1.0);

alpha = DS2R \* ((A + (B + (C + D \* t)\*t)\*t) + f);

alpha = fmod(alpha, M\_PI);

*if* (alpha < 0) alpha = alpha + 2 \* M\_PI;

*//* *Изменение* *средней* *аномалии* *во* *времени* *характеризуется* *следующим* *соотношением*

n = sqrt(398600.4415 / a \* a \* a);

M = M0 + n \* t; *//* *В* *определенный* *момент* *времени* *изменение* *аномалии*

Ei1 = M; *//* *Начальное* *значение* *эксцентрической* *аномалии*

*//* *Находим* *методом* *приближений* *из* *уравнения* *Кеплера*

*while* ((abs(Ei1 - Ei) >= 0.001))

{

Ei = Ei1;

Ei1 = M + e \* sin(Ei);

}

*//* *Находим* *истинную* *аномалию*

fita = 2 \* atan(sqrt((1 + e) / (1 - e))\*tan(Ei1 / 2));

*//* *Находим* *аргумент* *широты* *орбиты*

u = fita + om;

*//* *Находим* *модуль* *радиу-вектора*

ra = (a\*(1 - e \* e)) / (1 + e \* cos(fita)); *//* *=* *a\*(1* *-* *e* *\** *cos(E))*

*//* *Положение* *КА* *в* *инерциальной* *системе* *координат*

Xa = ra \* (cos(u)\*cos(OM) - sin(u)\*sin(OM)\*cos(i));

Ya = ra \* (cos(u)\*sin(OM) - sin(u)\*cos(OM)\*cos(i));

Za = ra \* sin(u)\*sin(i);

*//* *Построение* *трассы*

*//* *Находим* *широту*

fi = asin(Za / sqrt(pow((Xa\*sin(alpha) - Ya \* cos(alpha)), 2) + pow((Xa\*cos(alpha) + Ya \* sin(alpha)), 2) + pow(Za, 2)));

*//* *Находим* *долготу*

*if* (Xa > 0)

{

lambda = atan(Ya / Xa);

}

*else* *if* (Xa < 0 & Ya >= 0)

{

lambda = atan(Ya / Xa) + M\_PI;

}

*else* *if* (Xa < 0 & Ya < 0)

{

lambda = atan(Ya / Xa) - M\_PI;

}

*else* *if* (Xa == 0 & Ya > 0)

{

lambda = M\_PI / 2;

}

*else* *if* (Xa == 0 & Ya < 0)

{

lambda = -M\_PI / 2;

}

*else* *if* (Xa == 0 & Ya == 0)

{

output << " Долгота не определена";

}

*//* *Вывод* *в* *файл*

output.setf(ios::fixed);

output << alpha;

output.width(15);

output << fi;

output.width(10);

output << sec << "." << min << "." << hour << "/" << day <<"."<<month <<"."<< year;

output.width(25);

output << Xa << " " <<Ya <<" "<<Za << endl;

}

output.close();

}

Тут я матрицы преобразовывал в 3 пункте.

