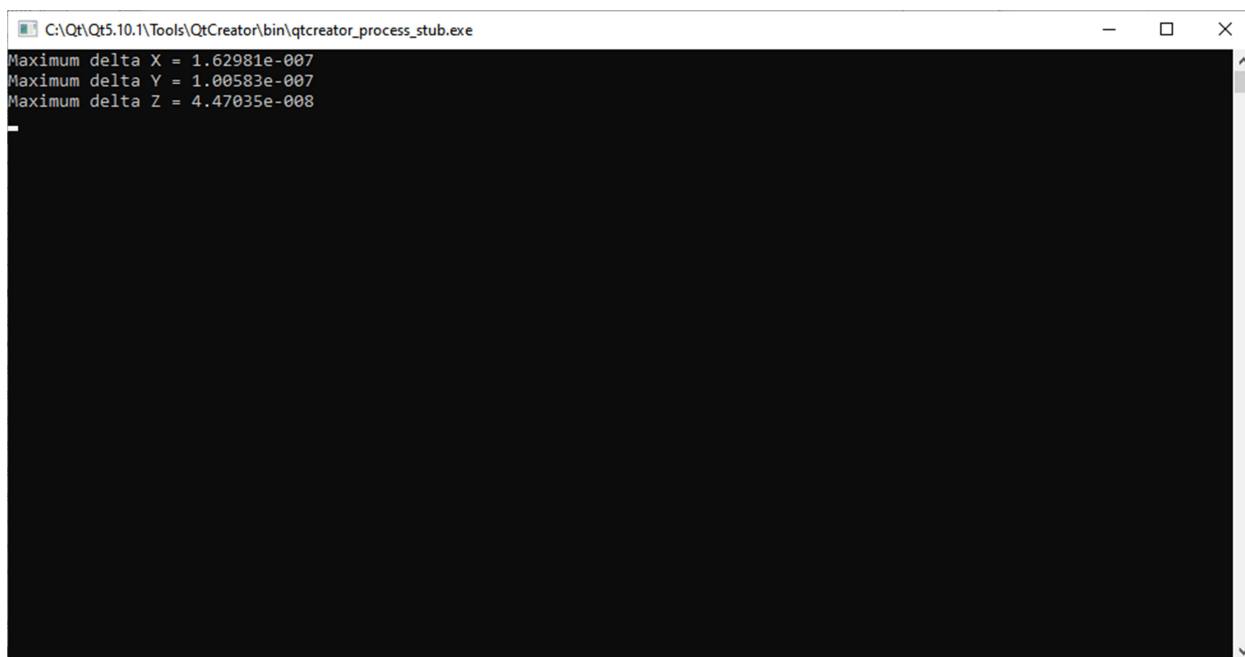


Этап 3. Реализация.

На этом этапе работы надо разработать на языке C++ функцию расчёта положения спутника Beidou на заданное время по шкале UTC. Написание программы производилось в Visual studio 2019. Алгоритм расчёта местоположения спутника включает в себя уравнение Кеплера. Необходимо протестировать программу.

A screenshot of a Qt Creator console window. The title bar shows the file path 'C:\Qt\Qt5.10.1\Tools\QtCreator\bin\qtcreator_process_stub.exe'. The console output displays three lines of text: 'Maximum delta X = 1.62981e-007', 'Maximum delta Y = 1.00583e-007', and 'Maximum delta Z = 4.47035e-008'. The rest of the console area is black.

```
C:\Qt\Qt5.10.1\Tools\QtCreator\bin\qtcreator_process_stub.exe
Maximum delta X = 1.62981e-007
Maximum delta Y = 1.00583e-007
Maximum delta Z = 4.47035e-008
```

Рисунок 14 – Результаты теста программы.

Заключение.

При выполнении данного курсового проекта была построена орбита заданного спутника Beidou, получена траектория движения этого спутника на заданный промежуток времени.

Так же были изучены все этапы нахождения местоположения спутника и реализованы в Matlab.

Последним этапом курсового проекта была разработка функции расчёта местоположения на языке программирования C++ и сравнение результатов с программой из Matlab.

Приложение.

```
#include <iostream>
#include <math.h>

using namespace std;

int main()
{
    // Эфемериды
    double SatNum = 6;
    double toe = 241200;
    double Crs = -9.2875000000000000e+01;
    double Dn = 8.71107710704449589e-13;
    double M0 = 2.32726368913121773e+00;
    double Cuc = -2.62074172496795654e-06;
    double e = 1.05765871703624725e-02;
    double Cus = 2.34702602028846741e-05;
    double sqrtA = 6.49287138557434082e+03;
    double Cic = -1.12690031528472900e-07;
    double Omega0 = 6.63759799965142852e-01;
    double Cis = 3.25962901115417480e-09;
    double i0 = 9.46015118241178121e-01;
    double Crc = -4.82140625000000000e+02;
    double omega = -2.20504767262928070e+00;
    double OmegaDot = -1.773288508356074e-12;
    double iDot = -2.00008331149807446e-14;
    double Tgd = 9.7500000000000000e+05;
    double toc = 2.1960000000000000e+08;
    double af2 = 1.48307593848345250e-22;
    double af1 = 5.06794606280891458e-12;
    double af0 = 3.53220045566558838e-01;
    double URA = 0;
    double IODE = 257;
    double IODC = 1;
    double codeL2 = 0;
    double L2P = 0;
    double WN = 789;

    // Значения констант
    double mu = 3.986004418e14; // гравитационная постоянная
    double omega_e = 7.2921151467e-5; // скорость вращения

    // Временной промежуток
    double begin_time = (24*2+18-3)*60*60; // время начала 8:00 по МСК 16
    февраля
    double end_time = (24*3+6-3)*60*60; // время окончания 6:00 по МСК 17
    февраля

    // Длина временного промежутка
    double step_time = 1;
    int t_len = 1 + (end_time - begin_time) / step_time;

    double *X0 = new double[t_len];
    double *Y0 = new double[t_len];
    double *Z0 = new double[t_len];

    // Большая полуось
    double A = pow(sqrtA, 2);

    // Среднее движение
    double n0 = sqrt(mu/pow(A, 3));
    double n = n0+Dn;
```

```

    for (int t_ = begin_time, k = 0; t_ <= end_time; t_ += step_time, k++)
    {

        double t = t_ - toe;

        // Vremya
        if (t > 302400) {
            t -= 604800;
        }
        if (t < -302400) {
            t += 604800;
        }

        // Средняя аномалия
        double M = M0+n*t;

        // Решение уравнения Кеплера
        double E = M;
        double E_old = M+1;
        double epsilon = 1e-6;

        while (fabs(E-E_old) > epsilon) {
            E_old = E;
            E = M+e*sin(E);
        }

        // Истинная аномалия
        double nu = atan2(sqrt(1-pow(e,2))*sin(E), cos(E)-e);

        // Коэффициент коррекции
        double cos_correction = cos(2*(omega+nu));
        double sin_correction = sin(2*(omega+nu));

        // Аргумент широты
        double u = omega+nu+Cuc*cos_correction+Cus*sin_correction;

        // Радиус
        double r = A*(1-
e*cos(E))+Crc*cos_correction+Crs*sin_correction;

        // Наклон
        double i = i0+iDot*t+Cic*cos_correction+Cis*sin_correction;

        // Долгота восходящего угла
        double lambda = Omega0+(OmegaDot-omega_e)*t-omega_e*toe;

        // Положение на орбите
        double x = r*cos(u);
        double y = r*sin(u);

        // Координаты
        X0[k] = x*cos(lambda)-y*cos(i)*sin(lambda);
        Y0[k] = x*sin(lambda)+y*cos(i)*cos(lambda);
        Z0[k] = y*sin(i);
    }

    // Считываем значения из матлаба
    double *X0m = new double[t_len];
    double *Y0m = new double[t_len];
    double *Z0m = new double[t_len];

    FILE *file;
    if ((file = fopen("../data_matlab.txt", "rb+")) == NULL) {

```

```

        printf("Cannot open file.\n");
    }
    else {
        for (int i = 0; i < t_len; i++) {
            fscanf(file, "%le %le %le\n", &X0m[i], &Y0m[i],
&Z0m[i]);
        }
        fclose(file);
    }

    double deltaX, deltaY, deltaZ;
    double maxDeltaX = 0, maxDeltaY = 0, maxDeltaZ = 0;

    // Сравниваем значения
    for (int i = 0; i < t_len; i++) {
        // Определение максимальной разницы по X
        deltaX = fabs(X0[i] - X0m[i]);
        if (deltaX > maxDeltaX)
            maxDeltaX = deltaX;

        // Определение максимальной разницы по Y
        deltaY = fabs(Y0[i] - Y0m[i]);
        if (deltaY > maxDeltaY)
            maxDeltaY = deltaY;

        // Определение максимальной разницы по Z
        deltaZ = fabs(Z0[i] - Z0m[i]);
        if (deltaZ > maxDeltaZ)
            maxDeltaZ = deltaZ;
    }

    cout << "Maximum delta X = " << maxDeltaX << endl;
    cout << "Maximum delta Y = " << maxDeltaY << endl;
    cout << "Maximum delta Z = " << maxDeltaZ << endl;

    return 0;
}

```