

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
"Национальный исследовательский университет
Московский энергетический институт"

НИУ МЭИ

Радиотехнический факультет
каф. Радиотехнических систем

КУРСОВАЯ РАБОТА

Этап II

по дисциплине

«Аппаратура потребителей СРНС»

Тема курсовой работы:

Разработка модуля расчёта координат спутника GPS

(наименование темы)

Студент группы ЭР-15-14

Каримов Х.Р.

Руководитель курсовой работы

к.т.н., доцент Корогодин И.В.

Работа представлена к защите

«Допущен к защите»

Москва 2019

```
cmake_minimum_required(VERSION 2.8)
project(sputnick2)
add_executable(sputnick2 "")
set(SOURCES
    main.cpp
    kepler.h
    kepler.cpp
    gpsvspos.h
    gpsvspos.cpp
    tests.h
    tests.cpp
)
target_sources(sputnick2
    PRIVATE ${SOURCES}
)
set (CMAKE_CXX_FLAGS "-std=c++11")
```

```
#include <iostream>
#include "tests.h"
using namespace std;

int main()
{
    Tests tests;
    if(tests.run_tests())
    {
        printf("SUCCESS! TESTS PASSED\n");
    }
    return 0;
}
```

```

#include "gpsvspos.h"
#include <math.h>

#define M_PI 3.14159265359
Gpsvspos::Gpsvspos()
{
    params_init();
    coords_init();
}

typedef struct Params
{
    double A;
    double M0;
    double Toe;
    double omegaZero;
    double omega;
    double omegaDot;
    double omegaDotE;
    double eccentricity;
    double inclination;
    double delta_n;
    double mu;
    double IDOT;
    double Cus;
    double Cuc;
    double Crs;
    double Crc;
    double Cis;
    double Cic;
} Params;

static Params params;
static CoordinatesOfSatellite coordinatesOfSatellite;

// params for calculate satellite coordinates
void Gpsvspos::params_init(void)
{
    params.A = 26559371.973;
    params.M0 = 170.75306;
    params.Toe = 288000;
    params.omegaZero = 58.37197;
    params.omega = -86.81172;
    params.omegaDot = -4.5867E-07;
    params.omegaDotE = 7.2921151467E-5;
    params.eccentricity = 0.313590513542E-03;
    params.inclination = 54.99258;
    params.delta_n = 2.6429E-07;
    params.mu = 3986005000000000;
    params.IDOT = -1.7149E-08;
    params.Cus = 8.8383E-06;
    params.Cuc = -6.9290E-07;
    params.Crs = -1.2594E+01;
    params.Crc = 2.0738E+02;
    params.Cis = 5.4017E-08;
    params.Cic = 1.1176E-08;
}

```

```

void Gpsvpos::coords_init(void)
{
    coordinatesOfSatellite.ecefX = 0.0;
    coordinatesOfSatellite.ecefY = 0.0;
    coordinatesOfSatellite.ecefZ = 0.0;
}

double degToRad(double degree) {
    return degree * (M_PI/180);
}

CoordinatesOfSatellite Gpsvpos::findPositionOfSatellite(int momentOfTime) {

    params.Toe      = params.Toe + 18;
    params.M0       = degToRad(params.M0);
    params.omegaZero = degToRad(params.omegaZero);
    params.omega     = degToRad(params.omega);
    params.omegaDot  = degToRad(params.omegaDot);
    params.inclination = degToRad(params.inclination);
    params.delta_n   = degToRad(params.delta_n);
    params.IDOT      = degToRad(params.IDOT);

    int Tk = momentOfTime - (int)params.Toe;
    // Set accuracy of calculations
    double accuracy = pow(10, -8);

    if (Tk > 302400) {
        Tk = Tk - 604800;
    } else if (Tk < -302400) {
        Tk = Tk + 604800;
    }

    // Compute mean motion
    double n0 = pow(params.mu/(pow(params.A, 3)), 0.5);
    // Correct mean motion
    double n = n0 + params.delta_n;
    // Mean anomaly
    double meanAnomaly = params.M0 + n * Tk;

    // Solve Keplers equation
    double Ek = kepler.solve_kepler_equations(meanAnomaly, params.eccentricity, accuracy);
    // Callulate a true anomaly
    double Vk = atan2((pow((1 - params.eccentricity * params.eccentricity), 0.5) * sin(Ek) / (1 - params.eccentricity *
cos(Ek))), ((cos(Ek) - params.eccentricity) / (1 - params.eccentricity * cos(Ek))));

    // Argument of Latitude
    double Fk = Vk + params.omega;
    // Second harmonic perturbations
    double deltaUk = params.Cus * sin(2*Fk) + params.Cus * cos(2*Fk);
    double deltaRk = params.Crs * sin(2*Fk) + params.Crc * cos(2*Fk);
    double deltaIk = params.Cis * sin(2*Fk) + params.Cic * cos(2*Fk);
    // Correct argument of Latitude
    double Uk = Fk + deltaUk;
    // Correct radius
    double Rk = params.A * (1 - params.eccentricity * cos(Ek)) + deltaRk;
    // Correct inclination
    double Ik = params.inclination + deltaIk + params.IDOT * Tk;
    // Correct longitude of ascending node
    double Wk = params.omegaZero + (params.omegaDot - params.omegaDotE) * Tk - params.omegaDotE * params.Toe;

```

```

void Gpsvpos::coords_init(void)
{
    coordinatesOfSatellite.ecefX = 0.0;
    coordinatesOfSatellite.ecefY = 0.0;
    coordinatesOfSatellite.ecefZ = 0.0;
}

double degToRad(double degree) {
    return degree * (M_PI/180);
}

CoordinatesOfSatellite Gpsvpos::findPositionOfSatellite(int momentOfTime) {

    params.Toe      = params.Toe + 18;
    params.M0       = degToRad(params.M0);
    params.omegaZero = degToRad(params.omegaZero);
    params.omega     = degToRad(params.omega);
    params.omegaDot  = degToRad(params.omegaDot);
    params.inclination = degToRad(params.inclination);
    params.delta_n   = degToRad(params.delta_n);
    params.IDOT      = degToRad(params.IDOT);

    int Tk = momentOfTime - (int)params.Toe;
    // Set accuracy of calculations
    double accuracy = pow(10, -8);

    if (Tk > 302400) {
        Tk = Tk - 604800;
    } else if (Tk < -302400) {
        Tk = Tk + 604800;
    }

    // Compute mean motion
    double n0 = pow(params.mu/(pow(params.A, 3)), 0.5);
    // Correct mean motion
    double n = n0 + params.delta_n;
    // Mean anomaly
    double meanAnomaly = params.M0 + n * Tk;

    // Solve Keplers equation
    double Ek = kepler.solve_kepler_equations(meanAnomaly, params.eccentricity, accuracy);
    // Callulate a true anomaly
    double Vk = atan2((pow((1 - params.eccentricity * params.eccentricity), 0.5) * sin(Ek) / (1 - params.eccentricity *
cos(Ek))), ((cos(Ek) - params.eccentricity) / (1 - params.eccentricity * cos(Ek))));

    // Argument of Latitude
    double Fk = Vk + params.omega;
    // Second harmonic perturbations
    double deltaUk = params.Cus * sin(2*Fk) + params.Cus * cos(2*Fk);
    double deltaRk = params.Crs * sin(2*Fk) + params.Crc * cos(2*Fk);
    double deltaIk = params.Cis * sin(2*Fk) + params.Cic * cos(2*Fk);
    // Correct argument of Latitude
    double Uk = Fk + deltaUk;
    // Correct radius
    double Rk = params.A * (1 - params.eccentricity * cos(Ek)) + deltaRk;
    // Correct inclination
    double Ik = params.inclination + deltaIk + params.IDOT * Tk;
    // Correct longitude of ascending node
    double Wk = params.omegaZero + (params.omegaDot - params.omegaDotE) * Tk - params.omegaDotE * params.Toe;

```

```

// Positions in orbitalplane
double x = Rk * cos(Uk);
double y = Rk * sin(Uk);
// Earth-fixed coordinates
// Coordinates in ECEF system
double ecefX = x * cos(Wk) - y * cos(Ik) * sin(Wk);
double ecefY = x * sin(Wk) + y * cos(Uk) * cos(Wk);
double ecefZ = y * sin(Ik);

coordinatesOfSatellite.ecefX = ecefX;
coordinatesOfSatellite.ecefY = ecefY;
coordinatesOfSatellite.ecefZ = ecefZ;

return coordinatesOfSatellite;
}

```

kepler.cpp

```

#include "kepler.h"
#include <math.h>

Kepler::Kepler()
{
}

double Kepler::solve_kepler_equations(double meanAnomaly, double orbitalEccentricity, double accuracy) {

    double prevE_k = 0;
    double keplersSolution = 0;

    while (true)
    {
        keplersSolution = meanAnomaly + orbitalEccentricity * sin(prevE_k);

        if(abs(prevE_k - keplersSolution) <= accuracy) {
            break;
        }

        prevE_k = keplersSolution;
    }

    return keplersSolution;
}

```



```
Application Output
sputnick2
SUCCESS! TESTS PASSED21:30:26: C:/Users/Lincoln/Desktop/build-sp2-Desktop_Qt_5_9_4_MinGW_32bit-Debug/sputnick2.exe exited with code 0

21:30:35: Starting C:/Users/Lincoln/Desktop/build-sp2-Desktop_Qt_5_9_4_MinGW_32bit-Debug/sputnick2.exe...
0 0.000000 0.000000 0.000000
46799 0.000000 0.000000 0.000000
SUCCESS! TESTS PASSED
21:30:37: C:/Users/Lincoln/Desktop/build-sp2-Desktop_Qt_5_9_4_MinGW_32bit-Debug/sputnick2.exe exited with code 0

21:31:12: Starting C:/Users/Lincoln/Desktop/build-sp2-Desktop_Qt_5_9_4_MinGW_32bit-Debug/sputnick2.exe...
SUCCESS! TESTS PASSED
21:31:13: C:/Users/Lincoln/Desktop/build-sp2-Desktop_Qt_5_9_4_MinGW_32bit-Debug/sputnick2.exe exited with code 0

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 Debugger Console 6 General Messages 8 Test Results
```

Рисунок 1 Результат прохождения теста




```
Application Output
sputnick2
0ns
20:38:00: C:/Users/Lincoln/Desktop/build-sp2-Desktop_Qt_5_9_4_MinGW_32bit-Debug/sputnick2.exe exited with code 0

20:38:48: Starting C:/Users/Lincoln/Desktop/build-sp2-Desktop_Qt_5_9_4_MinGW_32bit-Debug/sputnick2.exe...
0ns
0ns
20:38:48: C:/Users/Lincoln/Desktop/build-sp2-Desktop_Qt_5_9_4_MinGW_32bit-Debug/sputnick2.exe exited with code 0

20:41:44: Starting C:/Users/Lincoln/Desktop/build-sp2-Desktop_Qt_5_9_4_MinGW_32bit-Debug/sputnick2.exe...
22488600ns
20:41:45: C:/Users/Lincoln/Desktop/build-sp2-Desktop_Qt_5_9_4_MinGW_32bit-Debug/sputnick2.exe exited with code 0

1 Issues (4) 2 Search Results 3 Application Output 4 Compile Output 5 Debugger Console 6 General Messages 8 Test Results
```

Рисунок 2 Время выполнения программы (22,4886мс)



```
Command Window
Elapsed time is 0.372123 seconds.
fx >>
```

Рисунок 3 Время выполнения программы на этапе симуляции (37,2123 мс)

```
ioann@ioann-HP-Pavilion-Laptop-15-ck0xx ~/Desktop/build-sp3-Desktop_Qt_5_12_2_GCC... - + x
File Edit View Search Terminal Help
ioann@ioann-HP-Pavilion-Laptop-15-ck0xx ~/Desktop/build-sp3-Desktop_Qt_5_12_2_GCC...
C_64bit-Debug $ sputnick2
sputnick2: command not found
ioann@ioann-HP-Pavilion-Laptop-15-ck0xx ~/Desktop/build-sp3-Desktop_Qt_5_12_2_GCC...
C_64bit-Debug $ valgrind --leak-check=yes sputnick2
valgrind: sputnick2: command not found
ioann@ioann-HP-Pavilion-Laptop-15-ck0xx ~/Desktop/build-sp3-Desktop_Qt_5_12_2_GCC...
C_64bit-Debug $ valgrind -v
valgrind: no program specified
valgrind: Use --help for more information.
ioann@ioann-HP-Pavilion-Laptop-15-ck0xx ~/Desktop/build-sp3-Desktop_Qt_5_12_2_GCC...
C_64bit-Debug $ ./sputnick2
SUCCESS! TESTS PASSED
ioann@ioann-HP-Pavilion-Laptop-15-ck0xx ~/Desktop/build-sp3-Desktop_Qt_5_12_2_GCC...
C_64bit-Debug $ valgrind ./sputnick2
==10746== Memcheck, a memory error detector
==10746== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==10746== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==10746== Command: ./sputnick2
==10746==
SUCCESS! TESTS PASSED
==10746==
==10746== HEAP SUMMARY:
==10746==    in use at exit: 72,704 bytes in 1 blocks
==10746==    total heap usage: 2 allocs, 1 frees, 73,728 bytes allocated
==10746==
==10746== LEAK SUMMARY:
==10746==    definitely lost: 0 bytes in 0 blocks
==10746==    indirectly lost: 0 bytes in 0 blocks
==10746==    possibly lost: 0 bytes in 0 blocks
==10746==    still reachable: 72,704 bytes in 1 blocks
==10746==    suppressed: 0 bytes in 0 blocks
==10746== Rerun with --leak-check=full to see details of leaked memory
==10746==
==10746== For counts of detected and suppressed errors, rerun with: -v
==10746== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
ioann@ioann-HP-Pavilion-Laptop-15-ck0xx ~/Desktop/build-sp3-Desktop_Qt_5_12_2_GCC...
C_64bit-Debug $
```

Рисунок 4 Результат работы valgrind

Как видно из рисунка 4 утечек памяти нет. Плюс работа тестов подтверждают корректность работы сборки

Вывод по курсовому проекту:

В ходе 1го этапа данного проекта были извлечены эфемериды спутника из двоичных файлов. Эти эфемериды в дальнейшем использовались для построения модели движения спутника и нахождения его координат с прогнозом на 1 сутки вперед. После получения эталонных координат была предпринята попытка написания библиотеки для расчета координаты спутника на языке Си. Эталонные координаты, полученные при моделировании, использовались как тесты. Т.е. при схожести результатов СИ-кода и кода из Матлаб делается вывод о корректной работе написанных библиотек.

Программа на СИ коде оказалась на 65% быстрее, чем ее аналог в Матлабе. Хотя в некоторых источниках утверждается о потенциально больших показателях производительности (до 50-100 раз). Попытки оптимизировать код не производились, поэтому быстродействие практически соизмеримо.