**Национальный исследовательский университет**

**«МЭИ»**

**Институт радиотехники и электроники**

**Кафедра радиотехнических систем**

**Методы оптимального приема сигналов в аппаратуре потребителей СРНС**

Курсовой проект по курсу

«**Аппаратура потребителей СРНС**»

Проверил: Выполнил студент:

Корогодин И.В. группы Эр-15-14

Воронов Е.Г.

Москва 2019

**Этап 3. Реализация**

**1. Код реализации**

Требуется разработать на языке С/С++ функцию расчета положения спутника GPS на заданное время по шкале GPST, минимизируя время её исполнения и количество затрачиваемой оперативной памяти. Вызов функции не должен приводить к выбросу исключений или утечкам памяти при любом наборе входных данных.

Весь проект состоит из 5-ти файлов: два заголовочных и три исходных.

Реализация была продела на языке С++ с использованием компилятора Visual Studio.

main.cpp

#include <iostream>

#include <fstream>

#include <string>

#include "Сoordinate\_calculation.h"

#include "Kepler\_equation.h"

#include <ctime>

using namespace std;

void main()

{

Efemeridi Efemer;

const double PI = 3.141592653589793;

Efemer.mu = 3.986004418E14;

Efemer.we = 7.2921151467E-5;

Efemer.toe = 288000;

Efemer.A = 26559353.945;

Efemer.e = 0.00189325;

Efemer.M0 = 158.68863 \* PI / 180;

Efemer.omega = 36.26292 \* PI / 180;

Efemer.i0 = 55.15975 \* PI / 180;

Efemer.omega0 = -3.32161 \* PI / 180;

Efemer.del\_n = PI / 180 \* 2.6989E-7;

Efemer.i\_dot = PI / 180 \* 7.0804E-9;

Efemer.omega\_dot = PI / 180 \* (-4.6172E-7);

Efemer.c\_uc = -5.9418E-7;

Efemer.c\_us = 7.1991E-6;

Efemer.c\_rc = 2.3891E2;

Efemer.c\_rs = -1.0062E1;

Efemer.c\_ic = -5.7742E-8;

Efemer.c\_is = -2.9802E-8;

time\_t start, end;

double t = 3.5\*24\*3600-3\*3600;

double delta\_t = 0.1;

double\* coord = new double[3];

double\* coord\_matlab = new double[3];

double max\_del = 0;

ofstream out;

out.open("D:\\C\\KP\_CHAST\_3\\Cpp.txt");

ifstream in("D:\\C\\KP\_CHAST\_3\\matlab.txt");

time(&start);

for (int i = 0; i < (12\*3600/delta\_t); i++)

{

gps\_coord(t, coord, Efemer);

t += delta\_t;

string coord\_str1 = to\_string(coord[0]);

string coord\_str2 = to\_string(coord[1]);

string coord\_str3 = to\_string(coord[2]);

out << coord\_str1 << " " << coord\_str2 << " " << coord\_str3 << endl;

in >> coord\_matlab[0] >> coord\_matlab[1] >> coord\_matlab[2];

for (int j = 0; j < 3; j++)

{

if (abs(coord[j] - coord\_matlab[j]) > max\_del)

{

max\_del = abs(coord[j] - coord\_matlab[j]);

}

}

}

time(&end);

in.close();

delete[] coord;

delete[] coord\_matlab;

coord = nullptr;

coord\_matlab = nullptr;

double seconds = difftime(end, start);

string seconds1 = to\_string(seconds \* 1000000 / (12 \* 3600 / delta\_t));

setlocale(LC\_ALL, "rus");

cout << "\n\t\tСреднее время расчёта, мкс: " << seconds1 << endl;

string max\_del1 = to\_string(max\_del);

cout << "\t\tМаксимальная разность координат: " << max\_del1 << endl;

out.close();

in.close();

В данном коде реализуется запись полученных координат и их сравнение с координатами полученными на втором этапе в среде MATLAB (код программы, записи приведён в приложении 1).

Сoordinate\_calculation.cpp

#include "Сoordinate\_calculation.h"

#include <iostream>

#include <math.h>

#include <ostream>

#include "Kepler\_equation.h"

using namespace std;

void gps\_coord(double t, double\* coord, Efemeridi Ef)

{

double tk = t - Ef.toe;

double Mk = Ef.M0 + (sqrt(Ef.mu) / pow(sqrt(Ef.A), 3) + Ef.del\_n) \* tk;

double Ek = kepler(Mk, Ef.e);

double Vk = atan2(sqrt(1 - pow(Ef.e, 2)) \* sin(Ek), cos(Ek) - Ef.e);

double Uk = Ef.omega + Vk + Ef.c\_uc \* cos(2 \* (Ef.omega + Vk)) + Ef.c\_us \* sin(2 \* (Ef.omega + Vk));

double rk = Ef.A \* (1 - Ef.e \* cos(Ek)) + Ef.c\_rc \* cos(2 \* (Ef.omega + Vk)) + Ef.c\_rs \* sin(2 \* (Ef.omega + Vk));

double ik = Ef.i0 + Ef.i\_dot \* tk + Ef.c\_ic \* cos(2 \* (Ef.omega + Vk)) + Ef.c\_is \* sin(2 \* (Ef.omega + Vk));

double lambk = Ef.omega0 + (Ef.omega\_dot - Ef.we) \* tk - Ef.we \* Ef.toe;

coord[0] = (cos(-lambk) \* cos(-Uk) - sin(-lambk) \* cos(-ik) \* sin(-Uk)) \* rk;

coord[1] = (-sin(-lambk) \* cos(-Uk) - cos(-lambk) \* cos(-ik) \* sin(-Uk)) \* rk;

coord[2] = (sin(-ik) \* sin(-Uk)) \* rk;

}

В данном программном коде происходит расчёт координат по эфемеридным данным, находящимся в main.cpp.

Kepler\_equation.p

#include "Kepler\_equation.h"

#include <math.h>

double kepler(double Mk, double e) {

double Ek = Mk;

double Ek1;

do {

Ek1 = Ek;

Ek = Mk + e \* sin(Ek);

} while (fabs(Ek1 - Ek) > 0.0000001);

return Ek;

}

Реализуется решение уравнения Кеплера.

Далее представлены два заголовочных файла.

Kepler\_equation.h

#ifndef KEPLER\_EQUATION\_H

#define KEPLER\_EQUATION\_H

double kepler(double Mk, double e);

#endif /\* #ifndef KEPLER\_H \*/#pragma once

Сoordinate\_calculation.h

#ifndef COORDINATE\_CALCULATION\_H

#define COORDINATE\_CALCULATION\_H

typedef struct {

double mu;

double we;

double toe;

double A;

double e;

double M0;

double omega;

double i0;

double omega0;

double del\_n;

double i\_dot;

double omega\_dot;

double c\_uc;

double c\_us;

double c\_rc;

double c\_rs;

double c\_ic;

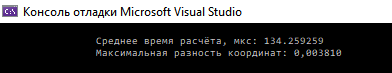
double c\_is;

} Efemeridi;

void gps\_coord(double t, double\* coord, Efemeridi Ef);

#endif /\* #ifndef GPSSVPOS\_H \*/#pragma once

**2.** **Результаты**

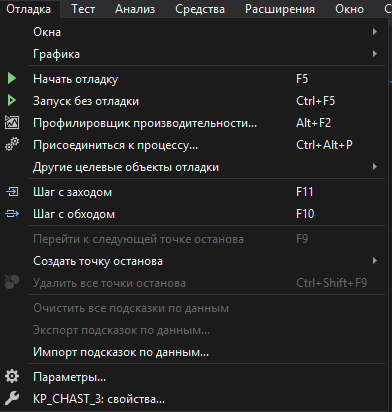


*Рисунок 12 – Результаты, выведенные в консоль.*

Из рисунка 12, максимальная разность координат 0,00381 метров; среднее время одиночного расчёта - 134.259259 мкс.

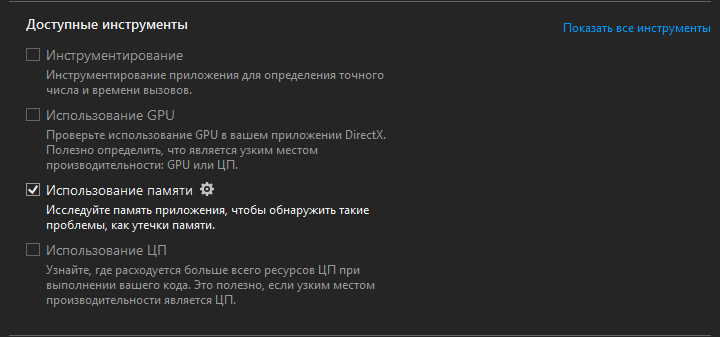
**3. Утечки памяти**

В Visual Studio во вкладке «отладка» есть пункт профилировщик производительности (рисунок 13).



*Рисунок 13 –Интерфейс Visual Studio .*

Выбор данного пункта позволяет открыть окно инструментов (рисунок 14).



*Рисунок 14 –Интерфейс Visual Studio .*

Выбрав пункт «Использование памяти» и нажав кнопку «начать» можно обнаружить утечки памяти.

**4. Выводы**

Различия определения координат при использовании MATLAB и при использовании языка программирования C++ присутствую, но являются достаточно малыми величинами. Максимальное различие составляет примерно 4 см.

**Приложение 1**

clc; clear all; close all;

tic;

%% эфемеридные данные

t\_oe = 288000;

e = 0.00189325;

omega = (pi/180)\*36.26292;

M\_0 = (pi/180)\*158.68863;

a = 26559353.945;

i\_0 = (pi/180)\*55.15975;

omega\_0 = -3.32161\*pi/180;

delta\_n = (pi/180)\*2.6989e-7;

i\_dot = (pi/180)\*7.0804e-9;

omega\_dot = (pi/180)\*(-4.6172e-7);

c\_us = 7.1991e-6;

c\_rc = 2.3891e2;

c\_rs = -1.0062e1;

c\_ic = -5.7742e-8;

c\_is = -2.9802e-8;

c\_uc = -5.9418e-7;

% a = [1;1;3];

%% Константы

mu = 3.986004418e14; %м^3/с^2 Геоцентрическая гравитационная постоянная

w\_e = 7.2921151467e-5; %рад/с Средняя угловая скорость Земли

%% Расчёт

t = 3\*24\*60\*60+12\*60\*60-3\*3600;

N = 432000; % сек

X\_Y\_Z = nan(N,3);

X\_Y\_Z\_2 = nan(N,3);

Rz = 6371000;

GPS\_time = nan(N,2);

L = nan(N,1);

E = nan(N,1);

R = nan(N,1);

az = nan(N,1);

elev = nan(N,1);

GPS\_time(:,1) = 2040;

latitude = 55.756727964;

longitude = 37.703259108;

height = 160;

for i = 1:N

GPS\_time(i,2) = t;

t\_k = t-t\_oe;

M\_k = M\_0+(sqrt(mu/a^3)+delta\_n)\*t\_k;

k = 2;

E\_k(k-1) = M\_k;

E\_k(k) = M\_k+e\*sin(E\_k(k-1));

while abs(E\_k(k)-E\_k(k-1))>=10^-8

k = k+1;

E\_k(k) = M\_k+e\*sin(E\_k(k-1));

end

E\_k = E\_k(k);

v\_k = atan2((sqrt(1-e^2)\*sin(E\_k)),cos(E\_k)-e);

u\_k = omega+v\_k+c\_uc\*cos(2\*(omega+v\_k))+c\_us\*sin(2\*(omega+v\_k));

r\_k = a\*(1-e\*cos(E\_k))+c\_rc\*cos(2\*(omega+v\_k))+c\_rs\*sin(2\*(omega+v\_k));

i\_k = i\_0+i\_dot\*t\_k+c\_ic\*cos(2\*(omega+v\_k))+c\_is\*sin(2\*(omega+v\_k));

lambda\_k = omega\_0+(omega\_dot-w\_e)\*t\_k-w\_e\*t\_oe;

l\_k = -lambda\_k;

R\_3\_lambda\_k = [cos(l\_k) sin(l\_k) 0; -sin(l\_k) cos(l\_k) 0; 0 0 1];

R\_1\_i\_k = [1 0 0; 0 cos(-i\_k) sin(-i\_k); 0 -sin(-i\_k) cos(-i\_k)];

R\_3\_u\_k = [cos(-u\_k) sin(-u\_k) 0; -sin(-u\_k) cos(-u\_k) 0;0 0 1];

X\_Y\_Z(i,:) = R\_3\_lambda\_k\*R\_1\_i\_k\*R\_3\_u\_k\*[r\_k;0;0];

t = t+0.1;

end

fid = fopen('matlab.txt', 'w+');

for i = 1:length(X\_Y\_Z(:,1))

fprintf(fid,'%.6f\t%.6f\t%.6f\n', X\_Y\_Z(i,1), X\_Y\_Z(i,2), X\_Y\_Z(i,3));

end

fclose(fid);

toc;