НИУ «МЭИ»

Кафедра Радиотехнических систем

Расчетно-пояснительная записка к курсовому проекту по

дисциплине «Аппаратура потребителей СРНС»

Выполнил: Чепелев И.И.

Группа: ЭР-15-17

Проверил: Корогодин И.В.

Москва 2022

**Этап 3. Моделирование в C++**

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

Функции расчета положения спутника в Matlab относительно проста, т.к. доступны библиотеки линейной алгебры и решения уравнений. Но при разработке встраиваемого ПО приходится сохранять лицензионную частоту, минимизировать вычислительную нагрузку и затраты памяти. Требуется выполнить свою реализацию решения трансцендентного уравнения.

Программный модуль должен сопровождаться unit-тестами под check:

* Тесты функций решения уравнения Кеплера.
* Тест расчетного положения спутника в сравнении с Matlab с шагом 0.1 секунды.

Во время второго теста должно вычисляться и выводиться средняя длительность исполнения функции. Допускается использовать одни и те же эфемериды на весь рассматриваемый интервал.

Требуется провести проверку на утечки памяти.

**Разработка алгоритма расчета положения спутника на языке С++**

Функция была реализована на языке программирования С/С++. Так как функция расчета относительно проста, весь расчет выполняется внутри главной функции программы main (). Функция, помимо вычислений координат, так же считывает координаты из файла, сравнивает их с рассчитанными, находит максимальную разницу, а также вычисляет общее время выполнения.

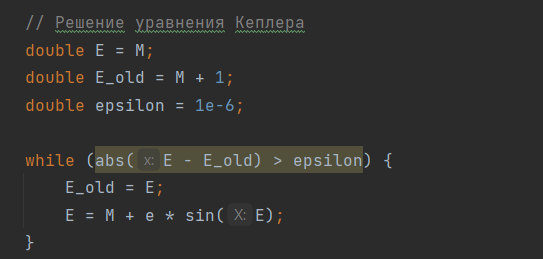


Рисунок 1 –– Алгоритм решения уравнения Кеплера

Для того, чтобы достичь высокой точности, необходимо реализовать передачу и вывод чисел с высокой точностью (количеством знаков после запятой).



Рисунок 2 –– Результат работы программы

После реализации и отладки кода, была проведена работа по анализу объема потребляемой памяти и нахождения утечки памяти. Данные действия были проведены в программе Microsoft Visual Studio 2022.

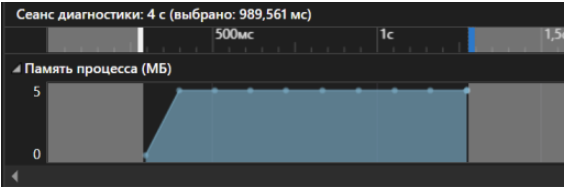
****

Рисунок 3 –– Анализ объема потребляемой памяти

**Вывод:**

На третьем этапе курсового проекта была написана программа на языке C++, что целесообразно при разработке встраиваемого ПО. Было показано, что реализация алгоритмов в C++и Matlab не приводит к серьезному расхождению в результатах. Объем потребляемой памяти составил 4111,85 КБ, утечка памяти не была обнаружена.

**Код программы:**

#include <iostream>  
#include <fstream>  
#include <cmath>  
  
  
using namespace std;  
  
int main()  
{  
  
 setlocale(LC\_ALL, "rus");  
 std::cout << "Начало расчета!\n";  
 time\_t t\_start, t\_stop;  
 time(&t\_start);  
  
 double SatNum = 1;  
 double toe = 93600;  
 double Crs = -4.031250e+00;  
 double Dn = 1.606622e-9;  
 double M0 = -1.300067e+01;  
 double Cuc = -1.061708e-07;  
 double e = 6.875264e-03;  
 double Cus = 8.167699e-06;  
 double sqrtA = 5.153699e+03;  
 double Cic = 8.009374e-08;  
 double Omega0 = 1.268903e+02;  
 double Cis = 6.705523e-08;  
 double i0 = 5.377037e+01;  
 double Crc = 2.095312+02;  
 double omega = 2.118201e+01;  
 double OmegaDot = -4.743470e-07;  
 double iDot = -2.506795e-09;  
 double Tgd = 6.984919e-09;  
 double toc = 93600;  
 double af2 = 0.000000e+00;  
 double af1 = 3.183231e-12;  
 double af0 = 1.830263e-04;  
 double URA = 0;  
 double IODE = 9;  
 double IODC = 8;  
 double codeL2 = 1;  
 double L2P = 0;  
 double WN = 149;  
  
 // Значения констант  
 double mu = 3.986004418e14; // гравитационная постоянная  
 double omega\_e = 7.2921151467e-5;// скорость вращения  
  
 // Временной промежуток  
 int begin\_time = 24 \* 2 \* 60 \* 60;// время начала 0:00 по МСК 14 февраля  
 int end\_time = 24 \* 3 \* 60 \* 60;// время окончания 0:00 по МСК 15 февраля  
  
 //Длина временного промежутка  
 int t\_arr = end\_time - begin\_time;  
  
 //Большая полуось  
 double A = pow(sqrtA,2);  
  
 // Среднее движение  
 double n0 = sqrt(mu /pow(A,3));  
 double n = n0 + Dn;  
  
 // Координаты  
 double\*\* coord = new double\*[3];  
 for (int i = 0; i < 3; i++) {  
 coord[i] = new double[t\_arr];  
 }  
  
 // Координаты из матлаба  
 double\*\* coordMat = new double\*[3];  
 for (int i = 0; i < 3; i++) {  
 coordMat[i] = new double[t\_arr];  
 }  
  
 for (int k = 0; k < t\_arr; k++) {  
 // Время  
 double t = begin\_time + k - (int)toe;  
  
 if (t > 302400)  
 t = t - 604800;  
 if (t < -302400)  
 t = 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 (abs(E - E\_old) > epsilon) {  
 E\_old = E;  
 E = M + e \* sin(E);  
 }  
 // Истинная аномалия  
 double nu = atan2(sqrt(1 - e\*e) \* 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  
 double Xk = x \* cos(lambda) - y \* cos(i) \* sin(lambda);  
 double Yk = x \* sin(lambda) + y \* cos(i) \* cos(lambda);  
 double Zk = y \* sin(i);  
  
 coord[0][k] = Xk;  
 coord[1][k] = Yk;  
 coord[2][k] = Zk;  
 }  
  
 // Чтение координат из матлаба  
 ifstream file("data\_matlab.txt");  
  
 if (!file.is\_open())  
 cout << "Файл не может быть открыт!" << endl;  
 else {  
 for (int k = 0; k < t\_arr; k++) {  
 file >> coordMat[0][k] >> coordMat[1][k] >> coordMat[2][k];  
 }  
 file.close();  
 }  
  
  
 // Сравнение с матлабом  
 double max\_del = 0;  
 for (int i = 0; i < 3; i++) {  
 for (int k = 0; k < t\_arr; k++) {  
 if (abs(coord[0][k] - coordMat[0][k]) > max\_del) {  
 max\_del = abs(coord[0][k] - coordMat[0][k]);  
 }  
 }  
 }  
  
// Очищение памяти  
 delete[] \*coord;  
 delete[] coord;  
 delete[] \* coordMat;  
 delete[] coordMat;  
  
 time(&t\_stop);  
  
 double del\_time = difftime(t\_stop, t\_start);  
  
 cout << "Максимальная разница координат (м) = " << max\_del << endl;  
 cout << "Время выполнения расчета (с) = " << del\_time << endl;  
}