3 этап:

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.

Полученная программа:

main.cpp

#include <iostream>

#include <fstream>

#include <gpssvpos.h>

#include <kepler.h>

#include <ctime>

using namespace std;

int **main**()

{

time\_t start, end;

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

double delt = 0.1;

double \*koord = new double[3];

std::ofstream out;

out.open("D:\\rez\_cpp.txt");

time(&start);

for (int i; i < (12\*3600/delt); i++)

{

gps\_coord(t,koord);

t += delt;

std::string koord\_str1 = std::to\_string(koord[0]);

std::string koord\_str2 = std::to\_string(koord[1]);

std::string koord\_str3 = std::to\_string(koord[2]);

out << koord\_str1 << " " << koord\_str2 << " " << koord\_str3 << std::endl;

}

time(&end);

double seconds = difftime(end, start);

std::string seconds1 = std::to\_string(seconds\*1000000/(12\*3600/delt));

cout << seconds1 << std::endl;

out.close();

}

Gpssvpos.cpp

#include <gpssvpos.h>

#include <kepler.h>

#include <iostream>

#include <cmath>

#include <ostream>

using namespace std;

void **gps\_coord**(double t, double \*koord)

{

double mu = 3.986005E+14;

double we = 7.292115E-05;

double toe = .309584000000E+06;

double a\_sqr = .515366030693E+04;

double e = .432204524986E-02;

double M0 = -.138501624765E+01;

double omega = -.275322944457E+01;

double i0 = .962764890954E+00;

double omega0 = -.614370397393E-01;

double del\_n = .472269705981E-08;

double i\_dot = .306084178206E-09;

double omega\_dot = -.811176645906E-08;

double cuc = -.802800059319E-06;

double cus = .711344182491E-05;

double crc = .241281250000E+03;

double crs = -.144062500000E+02;

double cic = .447034835815E-07;

double cis = .167638063431E-07;

double tk = t - toe;

double Mk = M0 + (sqrt(mu)/pow(a\_sqr,3) + del\_n)\*tk;

double Ek = kepler(Mk,e);

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

double Uk = omega + Vk + cuc\*cos(2\*(omega + Vk)) + cus\*sin(2\*(omega + Vk));

double rk = a\_sqr\*a\_sqr\*(1-e\*cos(Ek)) + crc\*cos(2\*(omega+Vk))+crs\*sin(2\*(omega+Vk));

double ik = i0 + i\_dot\*tk + cic\*cos(2\*(omega + Vk)) + cis\*sin(2\*(omega + Vk));

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

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

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

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

}

Kepler.cpp

#include <kepler.h>

#include <cmath>

double **kepler**(double Mk, double e){

double Ek = Mk;

double Ek1 = Ek;

do{

Ek1 = Ek;

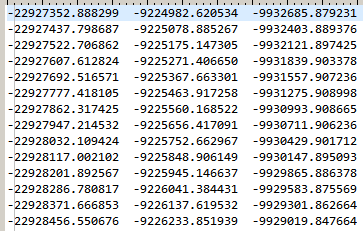
Ek = Mk + e\*sin(Ek);

}while(fabs(Ek1-Ek)/fabs(Ek) > 0.0001);

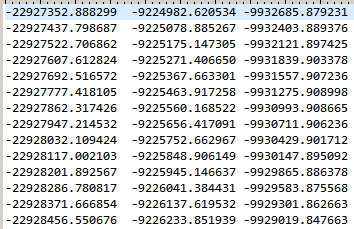
return Ek;

}

Результаты работы программы на С++



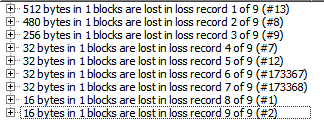
Результаты работы программы в Matlab



Как видно из результатов, данные отличаются друг от друга на величины до 0.000001 м по всем координатам. Это может быть вызвано различными способами округления результатов.

Тест на утечки памяти. Использовалось встроенное решение heob в компиляторе Qt Creator.

Результаты теста:



Результаты теста на время выполнения:

Замерять время исполнения функции будем измерением времени выполнения цикла и делением на количество итераций (оценка приближенная).





Время приведено в микросекундах.