#include <iostream>

#include <vector>

#include <fstream>

#include <string>

using namespace std;

typedef vector<double> functionType(double, vector<double>);

typedef vector<double> initialConditionsType();

// Пример 1 система

vector<double> example1system(double t, vector<double> v) {

vector<double> a;

a.resize(3);

a[0] = -3 \* v[0] + 4 \* v[1] - 2 \* v[2];

a[1] = v[0] + v[2];

a[2] = 6 \* v[0] - 6 \* v[1] + 5 \* v[2];

return a;

}

// Пример 1 начальные условия

vector<double> example1InitialConditions() {

vector<double> tmp;

tmp.resize(3);

tmp[0] = 4;//x(0) = -4

tmp[1] = -2;//y(0) = 2

tmp[2] = -5;//z(0) = 5

return tmp;

}

//Пример 2 система

vector<double> example2system(double t, vector<double> v) {

vector<double> a;

a.resize(2);

a[0] = 2\*v[0] - v[1];

a[1] = -2\*v[0] + v[1] + 18\*t;

return a;

}

// Пример 2 начальные условия

vector<double> example2InitialConditions() {

vector<double> tmp;

tmp.resize(2);

tmp[0] = 1;//x(0) = -4

tmp[1] = 1;//y(0) = 2

return tmp;

}

// Пример 3 система

vector<double> example3system(double t, vector<double> v) {

vector<double> a;

a.resize(2);

a[0] = t/v[1];

a[1] = -t/v[0];

return a;

}

// Пример 3 начальные условия

vector<double> example3InitialConditions() {

vector<double> tmp;

tmp.resize(2);

tmp[0] = 1;//x(0) = -4

tmp[1] = 2;//y(0) = 2

return tmp;

}

// сложение 2-х векторов

vector<double> addVectors(vector<double> a, vector<double> b) {

for (int i = 0; i < a.size(); ++i) {

a[i] += b[i];

}

return a;

}

// сложение 4-х векторов

vector<double> add4Vectors(vector<double> a, vector<double> b, vector<double> c, vector<double> d) {

for (int i = 0; i < a.size(); ++i) {

a[i] += b[i] + c[i] + d[i];

}

return a;

}

// умножение ветора на число

vector<double> multVectorByNumber(vector<double> a, double b) {

for (int i = 0; i < a.size(); ++i) {

a[i] \*= b;

}

return a;

}

//k\_1 = f(t\_n, y\_n)

vector<double> coefK1(functionType fn, double t, vector<double> v) {

return fn(t, v);

}

//k\_2 = f(t + h, y\_n + h\*k\_1)

vector<double> coefK2(functionType fn, double t, vector<double> v, vector<double> k, double h) {

return fn(t + h/2, addVectors(v, multVectorByNumber(k, h/2)));

}

//k\_2 = f(t + h, y\_n + h\*k\_1)

vector<double> coefK4(functionType fn, double t, vector<double> v, vector<double> k, double h) {

return fn(t + h, addVectors(v, multVectorByNumber(k, h)));

}

// метод рунге 1-го прядка

vector<double> runge(functionType fn, double t, vector<double> v, double h) {

vector<double> k1 = coefK1(fn, t, v);

vector<double> k2 = coefK2(fn, t, v, k1, h);

vector<double> k3 = coefK2(fn, t, v, k2, h);

vector<double> k4 = coefK4(fn, t, v, k3, h);

return addVectors(v, multVectorByNumber(add4Vectors(k1, multVectorByNumber(k2, 2), multVectorByNumber(k3, 2), k4), h / 6));

}

int main() {

int n;

cout << "Chose system: ";

cin >> n;

functionType\* function = example1system;

initialConditionsType\* initialConditions = example1InitialConditions;

if (n == 1) {

function = example1system;

initialConditions = example1InitialConditions;

}

else if (n == 2) {

function = example2system;

initialConditions = example2InitialConditions;

}

else if (n == 3) {

function = example3system;

initialConditions = example3InitialConditions;

}

ofstream output;

output.open("System " + to\_string(n) + ".txt");

output << "t x y z" << endl;

double a = 0;

double b = 2;

double h = 0.01;

//нулевой шаг

vector<double> v0 = initialConditions();

double t = a;

output << t;

for (int i = 0; i < v0.size(); ++i) {

output << " " << v0[i];

}

output << endl;

//первый шаг

vector<double> v1 = runge(function, t, v0, h);

t += h;

output << t;

for (int i = 0; i < v1.size(); ++i) {

output << " " << v1[i];

}

output << endl;

//второй шаг

vector<double> v2 = runge(function, t, v1, h);

t += h;

output << t;

for (int i = 0; i < v2.size(); ++i) {

output << " " << v2[i];

}

output << endl;

//третий шаг

vector<double> v3 = runge(function, t, v2, h);

t += h;

output << t;

for (int i = 0; i < v3.size(); ++i) {

output << " " << v3[i];

}

output << endl;

//Метод Адамса

while (t <= b) {

vector<double> vNext = addVectors(v3, multVectorByNumber(add4Vectors(multVectorByNumber(function(t, v3), 55),

multVectorByNumber(function(t, v2), -59), multVectorByNumber(function(t, v1), 37), multVectorByNumber(function(t, v0), -9)), h / 24));

t += h;

output << t;

for (int i = 0; i < vNext.size(); ++i) {

output << " " << vNext[i];

}

output << endl;

v0 = v1;

v1 = v2;

v2 = v3;

v3 = vNext;

}

output.close();

return 0;

}