РЕШАВАНЕ НА СИСТЕМИ ОБИКНОВЕНИ ДИФЕРЕНЦИАЛНИ УРАВНЕНИЯ ОТ 1 РЕД С НАЧАЛНИ УСЛОВИЯ НА КОШИ

Методът на Рунге-Кута от 4-ти ред (RK4) е един от най-често използваните за решаване на системи от обикновени диференциални уравнения (ОДУ) от първи ред, тъй като предлага отличен баланс между сложност на изчисленията и точност (от 5 ред)

За да решим система, трябва да дефинираме вектор от функции.
В долния пример ще решим класическата система на **хармоничен осцилатор** (или просто две взаимосвързани уравнения):

$$dy_1/dt = y_2$$
$$dy_2/dt = -y_1 \text{ (което съответства на } y'' + y = 0)$$

```
#include <iostream>
#include <vector>
#include <iomanip>

using namespace std;

// Дефинираме системата от уравнения
// t: текущо време
// y: вектор със стойностите на променливите (y[0], y[1]...)
// dy: вектор, в който записваме производните
void system_equations(double t, const vector<double>& y, vector<double>& dy) {
        dy[0] = y[1];         // dy1/dt = y2
        dy[1] = -y[0];        // dy2/dt = -y1
}

int main() {
        int n = 2;            // Брой уравнения в системата
        double t = 0.0;       // Начално време
        double t_end = 10.0;  // Крайно време
        double h = 0.1;       // Стъпка на интеграция

        vector<double> y(n);
        // Начални условия: y1(0) = 0, y2(0) = 1
        y[0] = 0.0;
        y[1] = 1.0;

        // Помощни вектори за RK4
        vector<double> k1(n), k2(n), k3(n), k4(n), y_temp(n), dy(n);

        cout << fixed << setprecision(4);
        cout << "Time\t y1\t\t y2" << endl;
        cout << "-----------------------------------" << endl;

        while (t <= t_end) {
```

```
            cout << t << "\t " << y[0] << "\t " << y[1] << endl;

            // Стъпка 1: k1
            system_equations(t, y, k1);

            // Стъпка 2: k2
            for (int i = 0; i < n; i++) y_temp[i] = y[i] + 0.5 * h * k1[i];
            system_equations(t + 0.5 * h, y_temp, k2);

            // Стъпка 3: k3
            for (int i = 0; i < n; i++) y_temp[i] = y[i] + 0.5 * h * k2[i];
            system_equations(t + 0.5 * h, y_temp, k3);

            // Стъпка 4: k4
            for (int i = 0; i < n; i++) y_temp[i] = y[i] + h * k3[i];
            system_equations(t + h, y_temp, k4);

            // Пресмятане на крайната стойност за тази стъпка
            for (int i = 0; i < n; i++) {
                y[i] = y[i] + (h / 6.0) * (k1[i] + 2.0 * k2[i] + 2.0 * k3[i] + k4[i]);
            }

            t += h;
        }

        system("pause");
        return 0;
}
```

Методът RK4 изчислява четири различни „наклона“ (k1, k2, k3, k4) за всяка стъпка и ги комбинира чрез средно претеглена стойност. Това позволява на алгоритъма да „предвиди“ кривината на функцията, вместо просто да чертае права линия (както е при метода на Ойлер).

Как да се използва :

1. **Повече уравнения:** Ако имате система от 3 или повече уравнения, просто променете променливата n и добавете съответните формули в system_equations.
2. **Стъпка h:** Ако резултатите ви изглеждат неточни или системата е „нестабилна“, намалете стойността на h (например на 0.001).
3. **Dev-C++:** Кодът използва стандартни библиотеки (vector, iostream), така че ще работи на всяка версия на Dev-C++ без допълнителни настройки.
