```
#include <stdio.h>
#include <math.h>

// Definimos las funciones que describen el sistema de EDOs
double f(double t, double x, double v_x) {
    return v_x;
}

double g(double t, double x, double v_x, double q) {
    return -x-q*v_x;
}

// Implementamos el método de Runge-Kutta de segundo orden con arreglos
void rungeKutta(double t0, double x0, double v_x0, double h, double t_final, double x[], double
v_x[]) {
    double t, k1_x, k1_v_x,k2_x,k2_v_x, q;

    int i = 0;
    q = 0.5;
    t = t0;
    x[i] = x0;
    v_x[i] = v_x0;

    while (t < t_final) {
        k1_x = h * f(t, x[i], v_x[i]);
        k1_v_x = h * g(t, x[i], v_x[i], q);
        k2_x = h * f(t + h, x[i] + k1_x, v_x[i] + k1_v_x);
        k2_v_x = h * g(t + h, x[i] + k1_x, v_x[i] + k1_v_x, q);

        x[i+1] = x[i]+(k1_x+k2_x)/2;
        v_x[i+1] = v_x[i]+(k1_v_x+k2_v_x)/2;
        t += h;
        i++;

    }
}

int main() {
    double t0 = 0.0;       // Valor inicial de t
    double x0 = 1.0;       // Valor inicial de x
    double v_x0 = 0.0;     // Valor inicial de v_x
    double h = 0.2;  // Tamaño del paso
    double t_final = 4*M_PI;  // Valor final de t
```

```
    int n = (int)((t_final - t0) / h) +1;  // Calculamos el número de puntos
    double t, x[n], v_x[n];  // Arreglos para almacenar los valores de t, x, y v_x

    rungeKutta(t0, x0, v_x0, h, t_final, x, v_x);

    printf("Valores de t, x, y v_x:\n");
    for (int i = 0; i < n; i++) {
        printf("t = %lf, x = %lf, v_x = %lf\n", i*h, x[i], v_x[i]);
    }
     FILE*arch=fopen("Sub.txt","w");{
        for (int i = 0; i < n; i++) {
            fprintf(arch, "%lf, %lf, %lf\n", i*h, x[i], v_x[i]);
            }
        }
    fclose(arch);
}
```