```cpp
 1  #include <fstream>
 2  #include <array>
 3
 4  //Constant expressions appearing in the problem
 5  constexpr double PI = 3.14159265359;     //value of PI
 6  constexpr size_t STEPS = 1000;   //number of steps
 7
 8  //Definition of data types in the problem
 9  typedef double state_type;   //data type definition for dependant variables ⤸
       - array of x_0, x_1, ... x_n
10  typedef std::array<state_type, STEPS> solution;
11
12  //This is the differential Equation, with the higher order derivatives
13  state_type Pendulum(const state_type& x){
14      return - PI * PI * sin(x);
15  }
16
17  //The relaxation step
18  void relaxation_step(state_type (*Diff_Equation)(const state_type& x),      ⤸
      solution& x, const double& dt){
19      solution temporary = x;
20      for (size_t i = 1; i < STEPS - 1; i++) {
21          x[i] = 0.5 * (temporary[i-1] + temporary[i+1] - dt * dt *           ⤸
             Diff_Equation(temporary[i])); //Euler forward difference formula
22      }
23  }
24
25  int main(){
26      solution x_t{};    //variable to store the calculations
27
28      double t_0 = 0.0;    //initial time
29      double t_1 = 1.0;    //final time
30      double dt = (t_1 - t_0) / (STEPS - 1); //step size
31      state_type x_0 = 0.0, x_1 = PI / 4;    //boundary values for dependant    ⤸
          variables
32
33      //repeat the relaxation step
34      for (size_t i = 0; i < STEPS; i++) {
35          x_t[i] = x_0 + i * (x_1 - x_0) / (STEPS - 1); //initial linear        ⤸
             guess
36      }
37      for (size_t i = 0; i < 10000; i++) {
38          relaxation_step(Pendulum, x_t, dt);     //step forward
39      }
40
41      std::ofstream outfile;   //file handle to save the results in a file
42      outfile.open("./output/relaxation.txt", std::ios::out |                 ⤸
          std::ios::trunc );
43      for (size_t i = 0; i < STEPS; i++) {
```

```cpp
44              outfile << t_0 + i * dt << "\t" << x_t[i] << std::endl;
45         }
46     outfile.close();
47 }
```