```cpp
/*
    Descrição: Solução para a saturação em um escoamento bifásico
    Autor: Naim J. S. Carvalho (njscarvalho@iprj.uerj.br)
    Data: 01 de Dezembro de 2021
*/

#include <iostream>
#include <iomanip>
#include <cmath>
#include <vector>
#include <string>
#include <fstream>
#include <algorithm>     //std::fill

// Protótipos das funções usadas:
void show_parameters();
void save_saturation_data(const std::vector<double>& X, const std::vector<double>& Y, const std::string& filename);
template <typename T> std::vector<T> linspace(const double xi, const double xf, int Num);
double function_f(const double sw);
void solve_by_upwind(std::vector<double>& Sat, const std::vector<double>& Pos);

// Variáveis do problema:
constexpr double k {10e-15};             // permeabilidade
constexpr double phi {0.25};             // porosidade
constexpr double Sat_ini {0.2};          // saturação inicial
constexpr double Lx {5000.0};            // dimensão em x
constexpr double Ly {40.0};              // dimensão em y
constexpr double Lz {10.0};              // dimensão em y
constexpr double mu_o {1.0e-3};          // viscosidade do óleo
constexpr double mu_w {0.8e-3};          // viscosidade da água
constexpr double A {Ly*Lz};              // área da seção transversal
constexpr double q_t {50.0};             // vazão no lado esquerdo
constexpr auto C1 = q_t/(A*phi);
constexpr auto C2 = mu_o/mu_w;

// Variáveis da simulação:
constexpr int N {128};                              // número de células
constexpr double dx = Lx/N;                         // refinamento da malha
constexpr double t_max {365.0*3};                   // tempo de simulação, em dias
constexpr double max_u_value{1.00415};              // valor máximo da derivada de f_w
constexpr auto dt_max = dx/max_u_value;             // valor máximo permitido para o passo de tempo
constexpr auto dt = 0.95*dt_max;                    // valor do passo de tempo efetivamente usado
constexpr auto nsteps = static_cast<int>(t_max/dt);

int main(int argc, char* argv[]){
    show_parameters();
    std::vector<double> Pos = linspace<double>(0.0, Lx, N); // vetor para plotar Sw por x
    std::vector<double> Sat (N, Sat_ini);                   // vetor com as Saturações

    solve_by_upwind(Sat, Pos);
}

void solve_by_upwind(std::vector<double>& Sat, const std::vector<double>& Pos){

    std:fill(Sat.begin(), Sat.end(), Sat_ini);

    double sw_i {0.0}, sw_iprev {0.0};
    for (int n = 1; n <= nsteps; n++){
        // Primeira célula:
        Sat[0] = 1.0;
        // Iteramos da segunda até a penúltima célula:
        for (int i = 1; i < N; i++){
            sw_i = Sat[i];
            sw_iprev = Sat[i-1];
            Sat[i] = Sat[i] - (dt/dx)*(function_f(sw_i) - function_f(sw_iprev));
        }
```

```cpp
    }
    std::string name_out {"saturation_via_upwind.txt"};
    save_saturation_data(Pos, Sat, name_out);
}

double function_f(const double sw){
    return C1/(1.0 + C2*(1.0*std::pow(1-sw, 2))/(1.0*std::pow(sw, 2)));
}

void save_saturation_data(const std::vector<double>& X, const std::vector<double>& Y, const std::string& filename){
    std::fstream saver{filename, std::ios::out|std::ios::trunc};

    saver << std::setw(10) << "x (m) " << std::setw(10) << "Saturacao" << std::endl;
    const auto N = X.size();
    const auto M = Y.size();
    if ( N != M)
        return;
    for (int i = 0; i < N; i++){
        saver << std::setw(10) << X[i] << " " << std::setw(10) << std::setprecision(10) << Y[i] << std::endl;
    }
}

template <typename T>
std::vector<T> linspace(const double xi, const double xf, int Num){

    if (Num == 0 || Num == 1)
        Num++;

    std::vector<T> V (Num, 0.0);

    auto h = (xf - xi) / (Num-1);
    auto n = static_cast<int>(V.size());
    for (int i = 0; i < n; i++){
        V[i] = xi + i*h;
    }

    return V;
}

void show_parameters(){
    std::cout << "Numero de celulas: " << N << std::endl;
    std::cout << "Refinamento (dx): " << dx << std::endl;
    std::cout << "Tempo total: "  << t_max << " dias" << std::endl;
    std::cout << "Valor maximo da derivada de fw (u_max): " << max_u_value << std::endl;
    std::cout << "Passo de tempo maximo (dx/u_max): " << dt_max << " dia(s)" << std::endl;
    std::cout << "Passo de tempo usado: " << dt << " dia(s)" << std::endl;
    std::cout << "Numero de passos de tempo a calcular: " << nsteps  << std::endl;
}
```