```cpp
// NV_Stokes.cpp : This file contains the 'main' function. Program execution begins and ends there.

#include <iostream>
using namespace std;
#include "mainMesh.h"
#include "xStaggered.h"
#include "yStaggered.h"
#include "PHI.h"
#include "Pressure.h"

/*Functions Declaration | _R_U_ */
void _R_U_Initializer(vector<vector<double>>& _R_U_, vector<vector<double>>& U_Pred, float MeshSize_X, float MeshSize_Y);
vector<vector<double>> _R_U_OPS(CellX xStaggeredMesh, massFlowRate_StaggeredX massFlowRate_StaggeredX_, u_Staggered_X u_Staggered_X_, PHI_u u, double mu, float MeshSize_X, float MeshSize_Y, vector<vector<double>> _R_U_);
/*Functions Declaration | _R_V_ */
void _R_V_Initializer(vector<vector<double>>& _R_V_, vector<vector<double>>& V_Pred, float MeshSize_X, float MeshSize_Y);
vector<vector<double>> _R_V_OPS(CellY yStaggeredMesh, massFlowRate_StaggeredY massFlowRate_StaggeredY_, v_Staggered_Y v_Staggered_Y_, PHI_v v, double mu, float MeshSize_X, float MeshSize_Y, vector<vector<double>> _R_V_);
/*Functions Declaration | Predictor uP*/
vector<vector<double>> U_Predictor(vector<vector<double>>& _R_U_, vector<vector<double>>& U_Pred, u_Staggered_X u_Staggered_X_, double Delta_t, double roh, float MeshSize_X, float MeshSize_Y);
/*Functions Declaration | Predictor vP*/
vector<vector<double>> V_Predictor(vector<vector<double>>& _R_V_, vector<vector<double>>& V_Pred, v_Staggered_Y v_Staggered_Y_, double Delta_t, double roh, float MeshSize_X, float MeshSize_Y);
/*CFL Condition | Δt */
double CFL(u_Staggered_X u_Staggered_X_, v_Staggered_Y v_Staggered_Y_, double MeshSize_X, double MeshSize_Y, double roh, double Viscosity);

int main()
{
    float L = 1.0, H = 1.00, Nx = 5.0, My = 5.0, mu = 1.8E-5;
    /*Define Re*/
    double Re = 200;
    double roh = Re * mu, Delta_t = 0.1, Conv_Crit = 1E-7, t = 0.0;
    int itrMax = 10000, LoopNo = 1;
    vector<vector<double>> _R_U_, _R_V_, U_Pred, V_Pred;
    double Simulation_time = 10; /* a Rule of thumb: The fluid should pass through the domain 10 times to reach S.S. */

    /* -+-+-+-+-+--+-+-+-+-+-Class: ONE-+-+-+-+-+--+-+-+-+-+- */
    /*<<mainMesh>> a Class that Generates the Mesh, with a Constructor taking [L, H, Nx, My] as I/P */
    mainMesh MeshTrial(L, H, Nx, My);
    /*-+-+-+-+-+-+-+-+-+-+-+-+ Position/Area Structs -+-+-+-+-+-+-+-+-+-+-+-+-+*/
    Cell Cell_Struct = MeshTrial.Cell_Setter();
    Position Position_Struct = MeshTrial.Position_Getter();
    Area Area_Struct = MeshTrial.Area_Getter(Position_Struct);
    /*-+-+-+-+-+-+-+-+-+-+-+-+checkMesh-+-+-+-+-+-+-+-+-+-+-+-+-+*/
    MeshTrial.checkPosition(Position_Struct);
    MeshTrial.checkArea(Area_Struct);
    MeshTrial.checkMesh(Cell_Struct);
    /*-+-+-+-+-+-+-+-+-+-+-+-+Staggering-+-+-+-+-+-+-+-+-+-+-+-+-+*/
    /* Class: TWO */
    /*Definition of Return Struct / Class Object*/

    StaggeredMesh_X mainMesh_X_arg;
    StaggeredMesh_Y mainMesh_Y_arg;

    xStaggered xStaggered(L, H, Nx, My);
    yStaggered yStaggered(L, H, Nx, My);

    mainMesh_X_arg = xStaggered.structGetter_X(Cell_Struct, Position_Struct, Area_Struct
```

```
    );
56      mainMesh_Y_arg = yStaggered.structGetter_Y(Cell_Struct, Position_Struct, Area_Struct
    );
57      CellX xStaggeredMesh = xStaggered.xStaggeringOPs(mainMesh_X_arg);
58      CellY yStaggeredMesh = yStaggered.yStaggeringOPs(mainMesh_Y_arg);
59
60      /*-+-+-+-+-+-+-+-+-+-+-+staggeringCheck-+-+-+-+-+-+-+-+-+-+-+-+*/
61      xStaggered.checkMesh(xStaggeredMesh);
62      yStaggered.checkMesh(yStaggeredMesh);
63      /*/*-+-+-+-+-+-+-+-+-+-+-+Initialization, Resizing and &Reference
    Return/*-+-+-+-+-+-+-+-+-+-+-+*/
64      xStaggered.massFlowRate_StaggeredX_Initializer();
65      /*-+-+-+-+-+-+-+-+-+-+-+Initialization, Resizing and Reference
    Return-+-+-+-+-+-+-+-+-+-+-+*/
66      yStaggered.massFlowRate_StaggeredY_Initializer();
67      /*-+-+-+-+-+-+-+-+-+-+-+Initialization, Resizing and Reference
    Return-+-+-+-+-+-+-+-+-+-+-+*/
68      xStaggered.u_Staggered_X_Initializer(); /*uP, uE, uW, uN, uS*/
69      u_Staggered_X u_Staggered_X = xStaggered.get_u_Staggered_X();
70      /*-+-+-+-+-+-+-+-+-+-+-+Initialization, Resizing and Reference
    Return-+-+-+-+-+-+-+-+-+-+-+*/
71      yStaggered.v_Staggered_Y_Initializer(); /*vP, vE, vW, vN, vS*/
72      v_Staggered_Y v_Staggered_Y = yStaggered.get_v_Staggered_Y();
73      /*-+-+-+-+-+-+-+-+-+-+-+Advected FIELD: PHI | Velocity-+-+-+-+-+-+-+-+-+-+-+*/
74      PHI PHI(L, H, Nx, My);
75
76      PHI.PHI_Initializer(); /* RESIZING*/
77      PHI.PHI_Setter_BC(1.0, u_Staggered_X, v_Staggered_Y); /*B. Convection*/
78      PHI.PHI_Convection_Scheme(u_Staggered_X, v_Staggered_Y);
79
80      PHI_u u = PHI.getPHI_u(); /*[u_e, u_w, u_n, u_s]*/
81      PHI_v v = PHI.getPHI_v(); /*[v_e, v_w, v_n, v_s]*/
82
83      xStaggered.massFlowRate_StaggeredX_Setter(u_Staggered_X, xStaggeredMesh,
    v_Staggered_Y);
84      massFlowRate_StaggeredX massFlowRate_StaggeredX  = xStaggered.
    getMassFlowRate_StaggeredX();
85      yStaggered.massFlowRate_StaggeredY_Setter(v_Staggered_Y, yStaggeredMesh,
    u_Staggered_X);
86      massFlowRate_StaggeredY massFlowRate_StaggeredY = yStaggered.
    getMassFlowRate_StaggeredY();
87
88
89
90
91      /*R Operations | Predictor Velocity*/
92      _R_U_Initializer(_R_U_, U_Pred, Nx, My);
93      _R_U_ = _R_U_OPS(xStaggeredMesh, massFlowRate_StaggeredX, u_Staggered_X, u, mu, Nx,
    My,_R_U_);
94      U_Pred = U_Predictor(_R_U_, U_Pred, u_Staggered_X, Delta_t, roh, Nx, My);
95
96      _R_V_Initializer(_R_V_, V_Pred, Nx, My);
97      _R_V_ = _R_V_OPS(yStaggeredMesh, massFlowRate_StaggeredY, v_Staggered_Y, v, mu, Nx,
    My, _R_V_);
98      V_Pred = V_Predictor(_R_V_, V_Pred, v_Staggered_Y, Delta_t, roh, Nx, My);
99
100     /*PressureField Operations*/
101     Pressure PressureField(L, H, Nx, My);
102     //PressureField.PressureField_Initialization();
103     PressureField.PressureField_Initialization();
104     PressureField.Coeff_Vectors_Initialization();
105     PressureField.Coeff_Vectors_InternalNodes(Cell_Struct, Area_Struct, U_Pred, V_Pred,
    Delta_t, roh);
106     PressureField.Coeff_Vectors_BCs();
107     Coeff_Vectors PressureLINSYS = PressureField.Coffs_Vector_Getter();
108     vector<vector<double>> PressureField_ = PressureField.PressureField_Getter();
109
110     PressureField.LIN_EQ_Solver(PressureLINSYS, PressureField_, Conv_Crit, itrMax);
111     vector<vector<double>> PressureField_New = PressureField.PressureFieldNew_Getter();
```

```
112         xStaggered.u_Staggered_X_OPs(Cell_Struct, U_Pred, Delta_t, roh, PressureField_New);
113         yStaggered.v_Staggered_Y_OPs(Cell_Struct, V_Pred, Delta_t, roh, PressureField_New);
114         /*Return u_P and v_P | Staggered X and Y*/
115         u_Staggered_X = xStaggered.get_u_Staggered_X();
116         v_Staggered_Y = yStaggered.get_v_Staggered_Y();
117         /*CFL Condition */
118         double timeStep = CFL(u_Staggered_X, v_Staggered_Y, Nx, My, roh, mu);
119
120
121
122
123         /*-+-+-+-+-+-+-+-+-+-+-+-+-+-+Check Staggered massFlowRates-+-+-+-+-+-+-+-+-+-+-+-+-+*/
124         cout << "\033[1;32m massFlowRate Staggered X \t " << massFlowRate_StaggeredX.
            massFlowRate_East.size() << "*" << massFlowRate_StaggeredX.massFlowRate_North[0].size
            () << "\t" << "\033[1;31m PHI u: \t " << u.u_e.size() << "*" << u.u_e[0].size() <<
            "\n";
125         cout << "\033[1;32m massFlowRate Staggered Y \t " << massFlowRate_StaggeredY.
            massFlowRate_East.size() << "*" << massFlowRate_StaggeredY.massFlowRate_North[0].size
            () << "\t" << "\033[1;31m PHI v: \t " << v.v_e.size() << "*" << v.v_e[0].size() <<
            "\n";
126
127
128         //xStaggered.printMassFlowRates();
129         //yStaggered.printMassFlowRates();
130         PHI.printAndCheck();
131
132     }
133
134
135
136
137
138
139
140
141     /*Functions Implementation: Use Utility Class Later*/
142     void _R_U_Initializer(vector<vector<double>>& _R_U_, vector<vector<double>>& U_Pred,
        float MeshSize_X, float MeshSize_Y)
143     {
144         _R_U_.resize(MeshSize_X + 1, vector<double>(MeshSize_Y + 2 ));
145         U_Pred.resize(MeshSize_X + 1, vector<double>(MeshSize_Y + 2));
146     }
147
148     vector<vector<double>> _R_U_OPS(CellX xStaggeredMesh, massFlowRate_StaggeredX
        massFlowRate_StaggeredX_, u_Staggered_X u_Staggered_X_, PHI_u u, double mu, float
        MeshSize_X, float MeshSize_Y, vector<vector<double>> _R_U_)
149     {
150         for (int i = 1; i < MeshSize_X; ++i)
151             for (int j = 1; j <= MeshSize_Y; ++j)
152             {
153                 {
154                     _R_U_[i][j] = -(massFlowRate_StaggeredX_.massFlowRate_East[i][j -1] * u.
                        u_e[i][j - 1] - massFlowRate_StaggeredX_.massFlowRate_West[i][j - 1] * u.
                        u_w[i][j - 1] + massFlowRate_StaggeredX_.massFlowRate_North[i ][j - 1] *
                        u.u_n[i ][j - 1] - massFlowRate_StaggeredX_.massFlowRate_South[i ][j - 1]
                         * u.u_s[i ][j - 1])
155                        + mu * (u_Staggered_X_.u_P[i + 1][j] - u_Staggered_X_.u_P[i][j]) *
                           xStaggeredMesh.D_East[i][j - 1]
156                        - mu * (u_Staggered_X_.u_P[i][j] - u_Staggered_X_.u_P[i - 1][j]) *
                           xStaggeredMesh.D_West[i][j - 1]
157                        + mu * (u_Staggered_X_.u_P[i][j + 1] - u_Staggered_X_.u_P[i][j]) *
                           xStaggeredMesh.D_North[i][j - 1]
158                        - mu * (u_Staggered_X_.u_P[i][j] - u_Staggered_X_.u_P[i][j - 1]) *
                           xStaggeredMesh.D_South[i][j - 1];
159                     _R_U_[i][j] /= xStaggeredMesh.Volume[i][j - 1];
160                 }
161             }
162         return _R_U_;
163     }
```

```
164
165 void _R_V_Initializer(vector<vector<double>>& _R_V_, vector<vector<double>>& V_Pred,
    float MeshSize_X, float MeshSize_Y)
166 {
167       _R_V_.resize(MeshSize_X + 2, vector<double>(MeshSize_Y + 1));
168     V_Pred.resize(MeshSize_X + 2, vector<double>(MeshSize_Y + 1));
169 }
170
171 vector<vector<double>> _R_V_OPS(CellY yStaggeredMesh, massFlowRate_StaggeredY
    massFlowRate_StaggeredY_, v_Staggered_Y v_Staggered_Y_, PHI_v v, double mu, float
    MeshSize_X, float MeshSize_Y, vector<vector<double>> _R_V_)
172 {
173     for (int i = 1; i <= MeshSize_X; ++i)
174         for (int j = 1; j < MeshSize_Y; ++j)
175         {
176             {
177               _R_V_[i][j] = -(massFlowRate_StaggeredY_.massFlowRate_East[i-1][j] * v.v_e[
                  i - 1][j] - massFlowRate_StaggeredY_.massFlowRate_West[i - 1][j] * v.v_w[i
                  - 1][j] + massFlowRate_StaggeredY_.massFlowRate_North[i - 1][j] * v.v_n[i -
                   1][j] - massFlowRate_StaggeredY_.massFlowRate_South[i - 1][j] * v.v_s[i -
                  1][j])
178                     + mu * (v_Staggered_Y_.v_P[i + 1][j] - v_Staggered_Y_.v_P[i][j]) *
                        yStaggeredMesh.D_East[i - 1][j]
179                     - mu * (v_Staggered_Y_.v_P[i][j] - v_Staggered_Y_.v_P[i - 1][j]) *
                        yStaggeredMesh.D_West[i - 1][j]
180                     + mu * (v_Staggered_Y_.v_P[i][j + 1] - v_Staggered_Y_.v_P[i][j]) *
                        yStaggeredMesh.D_North[i - 1][j]
181                     - mu * (v_Staggered_Y_.v_P[i][j] - v_Staggered_Y_.v_P[i][j - 1]) *
                        yStaggeredMesh.D_South[i - 1][j];
182               _R_V_[i][j] /= yStaggeredMesh.Volume[i-1][j];
183             }
184         }
185     return _R_V_;
186 }
187
188 vector<vector<double>> U_Predictor(vector<vector<double>>& _R_U_, vector<vector<double>>&
     U_Pred, u_Staggered_X u_Staggered_X_, double Delta_t, double roh, float MeshSize_X,
    float MeshSize_Y)
189 {
190
191     for (int i = 1; i < MeshSize_X; ++i)
192         for (int j = 1; j <=  MeshSize_Y ; ++j)
193         {
194             {
195                 /* Forward Euler | To Converted Later to Adam-Bashforth Second Order
                    Temporal Scheme */
196                 U_Pred[i][j] = u_Staggered_X_.u_P[i][j] + (Delta_t / roh) * _R_U_[i][j];
197             }
198         }
199
200     return U_Pred;
201 }
202
203 vector<vector<double>> V_Predictor(vector<vector<double>>& _R_V_, vector<vector<double>>&
     V_Pred, v_Staggered_Y v_Staggered_Y_, double Delta_t, double roh, float MeshSize_X,
    float MeshSize_Y)
204 {
205
206     for (int i = 1; i <= MeshSize_X; ++i)
207         for (int j = 1; j < MeshSize_Y; ++j)
208         {
209             {
210                 /*Forward Euler | To Converted Later to Adam-Bashforth Second Order
                    Temporal Scheme */
211                 V_Pred[i][j] = v_Staggered_Y_.v_P[i][j] + (Delta_t / roh) * _R_V_[i][j];
212             }
213         }
214     return V_Pred;
215 }
```

```
216
217 double CFL(u_Staggered_X u_Staggered_X_, v_Staggered_Y v_Staggered_Y_, double MeshSize_X,
     double MeshSize_Y, double roh, double Viscosity)
218 {
219
220     double dX = 1.0 / MeshSize_X;
221     double dY = 1.0 / MeshSize_Y;
222     vector<vector<double>> CFL_Cond, CFL_Conv;
223     CFL_Cond.resize(MeshSize_X + 2, vector<double>(MeshSize_Y + 2));
224     CFL_Conv.resize(MeshSize_X + 2, vector<double>(MeshSize_Y + 2));
225
226     for (int i = 1; i <= MeshSize_X; ++i) {
227         for (int j = 1; j <= MeshSize_Y; ++j) {
228             double abs_U = (u_Staggered_X_.u_P[i][j] + u_Staggered_X_.u_P[i - 1][j]) /
                2.0;
229             double abs_V = (v_Staggered_Y_.v_P[i][j] + v_Staggered_Y_.v_P[i][j - 1]) /
                2.0;
230             CFL_Conv[i][j] = 0.35 * dX / sqrt(abs_U * abs_U + abs_V * abs_V);
231             CFL_Cond[i][j] = 0.2 * dX * dX / ( Viscosity / roh );
232         }
233     }
234
235     // Finding Minimum Δt
236     double min_delta_t = std::numeric_limits<double>::max();
237
238     for (int i = 1; i <= MeshSize_X; ++i) {
239         for (int j = 1; j <= MeshSize_Y; ++j) {
240             double delta_t = std::min(CFL_Cond[i][j], CFL_Conv[i][j]);
241             min_delta_t = std::min(min_delta_t, delta_t);
242         }
243     }
244     return min_delta_t;
245 }
246
```