

```

program lid_driven_cavity
implicit none
integer, parameter :: N = 50    ! grid size (NxN grid)
real :: dx, dy, dt, Re          ! grid spacing, time step, Reynolds number
real :: u(N, N), v(N, N), p(N, N) ! velocity and pressure fields
integer :: i, j, step
real :: start_time, end_time, elapsed_time

! Parameters
dx = 1.0 / (N-1)    ! Grid spacing in x direction
dy = 1.0 / (N-1)    ! Grid spacing in y direction
dt = 0.001          ! Time step size
Re = 100            ! Reynolds number

! Initialize arrays
u = 0.0
v = 0.0
p = 0.0

! Initialize the top boundary (lid) velocity
u(N, :) = 1.0

! Start timing
call cpu_time(start_time)

! Main loop for time stepping
do step = 1, 1000
    call compute_velocity(u, v, p, dx, dy, dt, Re)
    call update_pressure(p, dx, dy)

    ! Output or check convergence
    if (mod(step, 100) == 0) then
        print *, 'Step: ', step
    end if
end do

! Stop timing
call cpu_time(end_time)
elapsed_time = end_time - start_time
print *, 'Elapsed time for CFD simulation: ', elapsed_time, ' seconds'

contains

! Function to update the velocity and pressure fields (simplified)
subroutine compute_velocity(u, v, p, dx, dy, dt, Re)
    real, dimension(:, :), intent(inout) :: u, v, p
    real, intent(in) :: dx, dy, dt, Re
    integer :: i, j

    ! Simple explicit method for velocity (simplified)
    do i = 2, N-1
        do j = 2, N-1
            u(i, j) = u(i, j) - dt * ( (u(i, j) * (u(i+1, j) - u(i-1, j))) / (2*dx)
                                         (v(i, j) * (u(i, j+1) - u(i, j-1))) / (2*dy) )
        end do
    end do

    ! Simple velocity update for v (similar)
    do i = 2, N-1
        do j = 2, N-1
            v(i, j) = v(i, j) - dt * ( (u(i, j) * (v(i+1, j) - v(i-1, j))) / (2*dx)
                                         (v(i, j) * (v(i, j+1) - v(i, j-1))) / (2*dy) )
        end do
    end do
end subroutine compute_velocity

```

! Function to solve for pressure (simplified Poisson equation solver)  
subroutine update\_pressure(p, dx, dy)  
 real, dimension(:, :), intent(inout) :: p  
 real, intent(in) :: dx, dy  
 integer :: i, j

```
1  ! Simple pressure Poisson equation (Jacobi iteration)
2  do i = 2, N-1
3      do j = 2, N-1
4          p(i, j) = 0.25 * ( p(i+1, j) + p(i-1, j) + p(i, j+1) + p(i, j-1) )
5      end do
6  end do
7  end subroutine update_pressure
8
9  end program lid_driven_cavity
```