```cpp
#include <vector>
#include <list>
#include <map>
#include <set>
#include <deque>
#include <stack>
#include <bitset>
#include <algorithm>
#include <functional>
#include <numeric>
#include <utility>
#include <sstream>
#include <iostream>
#include <iomanip>
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <string.h>
using namespace std;
typedef struct temparature_data {
    int boundary_condition;
    double x,y,thermal_conductivity,temparature;
}temp_data;
typedef struct tt {
    double temparature;
}temp_data_mid;
int n,conv_stat_x,conv_stat_y,conv_stat;
temp_data **temp = new temp_data*[101];
temp_data_mid **temp_mid = new temp_data_mid*[101];
//temp_data** temp = new temp_data[101][101];
//temp_data_mid** temp_mid = new temp_data_mid[101][101];
//temp_data *temp = new temp_data[101];
double alpha,delta_t,delta_x,delta_y,A,B,C,D,a1[101],b1[101],d1[101],source1[101],var1[101],error[101][101],a2[101],b2[101],d2[101],source2[101],var2[101];
void data_initialization_1st_half();
void data_initialization_2nd_half();
void creat_grid();
void check_convergence();
void tdma1();
void tdma2();
void boundary_value_assignment();
int main() {
    for(int i=0;i<101;i++) {
        temp[i] = new temp_data[101];
        temp_mid[i] = new temp_data_mid[101];
    }


    int i,j,k,done = 0,direction = 1;
    data_initialization_1st_half();
    data_initialization_2nd_half();
    boundary_value_assignment();
    ofstream myfile3;
    myfile3.open("timet0.txt");
    for(int xx=0;xx<n;xx++) {
```

```
    for(int yy=0;yy<n;yy++) {
        myfile3 << temp[xx][yy].temparature << "        ";
    }
    myfile3 << endl;
}
myfile3.close();
creat_grid();
            // cartesian grid of total(xx) = n and total(yy) = n

ofstream myfile2;
myfile2.open ("result.txt");
for(int ii=0;(ii<100000)&&(done != 1);ii++) {
    //1st half.....................
    for(int yy = 1;yy<(n-1);yy++) {
        for(int xx=1;xx<(n-1);xx++) {
            source1[xx] = -temp[xx][yy].temparature - (A* (temp[xx][yy+1].temparature - (2*temp[xx][yy].temparature) + temp[xx][yy-1].temparature) );
        }
        tdma1();
        for(int xx=1;xx<(n-1);xx++) {
            temp_mid[xx][yy].temparature = var1[xx];
        }
    }
    //2nd half.....................
    for(int xx = 1;xx<(n-1);xx++) {
        for(int yy = 1;yy<(n-1);yy++) {
            source2[yy] = -temp_mid[xx][yy].temparature - (C* (temp_mid[xx+1][yy].temparature - (2*temp_mid[xx][yy].temparature) + temp_mid[xx-1][yy].temparature) );
        }
        // source2[0] and source[n-1] = 0 by boundary conditions.............
        tdma2();
        //for(int xx=1;xx<(n-1);xx++) {
            for(int yy=1;yy<(n-1);yy++) {
                error[xx][yy] = abs(var2[yy] - temp[xx][yy].temparature);
                temp[xx][yy].temparature = var2[yy];
            }
        //}
    }
    // intermediate file writing !!!.........................
    if(ii%500 == 0) {
        for(int xx=0;xx<n;xx++) {
            for(int yy = 0;yy<n;yy++) {
                myfile2 << temp[xx][yy].temparature<<" ";
            }
            myfile2 << endl;
        }
    }
    // convergence check !!.................................
    conv_stat = 0;
    check_convergence();
    if(conv_stat == 0) {
        cout << "Solution converged at timestep"<<" "<<ii+1<<endl;
        for(int xx=0;xx<n;xx++) {
            for(int yy=0;yy<n;yy++) {
                myfile2 << temp[xx][yy].temparature<<" ";
            }
            myfile2 << endl;
        }
        done = 1;
```

```
        }
        if(ii%100 == 0)
        cout << "computation at time step  "<<ii<<" done!!"<<endl;
        //---------------------------------------------------------------------------------------------------------------------------------------


    }
    myfile2.close();

    if(done == 1) {
        cout << "Solution converged !!!\n";
    }else
        cout << "Solution Not Converged More iterations required !!\n";

    //delete[] temp;

}
void creat_grid() {
    ofstream myfile1;
    myfile1.open("grid.txt");
    for(int xx=0;xx<n;xx++) {
        for(int yy=0;yy<n;yy++) {
            temp[xx][yy].x = delta_x*(xx);
            temp[xx][yy].y = delta_y*(yy);
        }
    }
    for(int xx=0;xx<n;xx++) {
        for(int yy=0;yy<n;yy++) {
            myfile1 << temp[xx][yy].x<<"  "<<temp[xx][yy].y<<"  ";
        }
        myfile1 << endl;
    }
    myfile1.close();
}
void boundary_value_assignment() {
    for(int yy = 0;yy<n;yy++) {
        temp[0][yy].temparature = 100; temp[n-1][yy].temparature = 0;
        temp_mid[0][yy].temparature = 100; temp_mid[n-1][yy].temparature = 0;
    }
    for(int xx=1;xx<(n-1);xx++) {
        temp[xx][0].temparature = 50; temp[xx][n-1].temparature = 50;
        temp_mid[xx][0].temparature = 50; temp_mid[xx][n-1].temparature = 50;
    }
}
void data_initialization_1st_half() {
    int i,j;
    n = 101;
    delta_x = 0.01; alpha = 0.000111; delta_t = 0.20,delta_y = 0.01;
    /* set A and B for sweep in X direction !!! */
    // data initialization for TDMA >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
    A = (alpha*delta_t)/(2*delta_x*delta_x);
    B = 1 + ((alpha*delta_t)/(delta_x*delta_x));

    for(i=1;i<(n-1);i++) {
        d1[i] = -B;
    }
    d1[0] = 1; d1[n-1] = 1;
```

```
    for(i=1;i<(n-1);i++) {
        a1[i] = -A;
    }
    a1[0] = 0.0; a1[n-1] = 0.0;
    for(i=1;i<(n-1);i++) {
        b1[i] = -A;
    }
    b1[0] = 0.0; b1[n-1] = 0.0;
    // initial and final source terms are defined by boundary conditions.....................
    source1[0] = 100; source1[n-1] = 0;
    //.....................................................................................
    // initial value assignment.....................................................................
    for(int xx=0;xx<n;xx++) {
        for(int yy=0;yy<n;yy++) {
            temp[xx][yy].temparature = 0;
            temp_mid[xx][yy].temparature = 0;
        }
    }
}
void tdma1() {
    int i,j;
    double aa1[n],cc1[n];
    aa1[0] = 0; aa1[n-1] = 0;
    for(i=1;i<(n-1);i++) {
        aa1[i] = a1[i]/(d1[i] - (b1[i]*aa1[i-1]));
    }
    cc1[0] = source1[0]; cc1[n-1] = source1[n-1];
    for(i=1;i<(n-1);i++) {
        cc1[i] = (b1[i]*cc1[i-1] + source1[i])/(d1[i] - (b1[i]*aa1[i-1]));
    }
    var1[n-1] = source1[n-1];
    for(i=n-1;i>0;i--) {
        var1[i] = (aa1[i]*var1[i+1]) + cc1[i];
    }
}
void data_initialization_2nd_half() {
    int i,j;
    n = 101;
    delta_x = 0.01; alpha = 0.000111; delta_t = 0.20,delta_y = 0.01;
    /* set A and B for sweep in X direction !!! */
    C = (alpha*delta_t)/(2*delta_y*delta_y);
    D = 1 + ((alpha*delta_t)/(delta_y*delta_y));
    for(i=1;i<(n-1);i++) {
        d2[i] = -D;
    }
    d2[0] = 1; d2[n-1] = 1;
    for(i=1;i<(n-1);i++) {
        a2[i] = -C;
    }
    a2[0] = 0.0; a2[n-1] = 0.0;
    for(i=1;i<(n-1);i++) {
        b2[i] = -C;
    }
    b2[0] = 0.0; b2[n-1] = 0.0;
    // initial and final source terms are defined by boundary conditions.....................
    source2[0] = 50; source2[n-1] = 50;
    //.....................................................................................
```

```
}
void tdma2() {
    int i,j;
    double aa2[n],cc2[n];
    aa2[0] = 0; aa2[n-1] = 0;
    for(i=1;i<(n-1);i++) {
        aa2[i] = a2[i]/(d2[i] - (b2[i]*aa2[i-1]));
    }
    cc2[0] = source2[0]; cc2[n-1] = source2[n-1];
    for(i=1;i<(n-1);i++) {
        cc2[i] = (b2[i]*cc2[i-1] + source2[i])/(d2[i] - (b2[i]*aa2[i-1]));
    }
    var2[n-1] = source2[n-1];
    for(i=n-1;i>0;i--) {
        var2[i] = (aa2[i]*var2[i+1]) + cc2[i];
    }
}
void check_convergence() {
    for(int xx=1;xx<(n-1);xx++) {
        for(int yy=1;yy<(n-1);yy++) {
            if(error[xx][yy] >= 0.0000001) {
                conv_stat += 1;
            }
        }
    }
}
//void write_data() {
    //myfile2 << "Writing this to a file.\n";
//    for(int i=0;i<n;i++) {
//        myfile2 << temp[i].temparature<<endl;
//    }
//    myfile2.close();
//}
```