DVR.cpp

```cpp
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    int nn;
    cout << "\nEnter Number of Nodes: ";
    cin >> nn;
    // Define the graph with dynamic size
    vector<vector<int>> graph(nn, vector<int>(nn, -1));
    vector<vector<int>> via(nn, vector<int>(nn, -1));
    // Vertex names
    vector<char> ch{'A', 'B', 'C', 'D', 'E', 'F', 'G'};
    // Get input for the graph distances
    for (int i = 0; i < nn; ++i)
    {
        for (int j = 0; j < nn; ++j)
        {
            if (i == j)
            {
                graph[i][j] = 0;
            }
            else if (graph[i][j] == -1)
            {
                cout << "\nEnter Distance between " << ch[i] << " - " << ch[j] << " : ";
                int t;
                cin >> t;
                graph[i][j] = graph[j][i] = t;
            }
        }
    }



    cout << "\nAfter Initialization:";
    // Display table initialization
    for (int i = 0; i < nn; ++i)
    {
        cout << "\n"
             << ch[i] << " Table";
        cout << "\nNode\tDist\tVia";
        for (int j = 0; j < nn; ++j)
        {
            cout << "\n"
                 << ch[j] << "\t" << graph[i][j] << "\t" << via[i][j];
        }
    }
```

```
// Sharing table
vector<vector<vector<int>>> sh(nn, vector<vector<int>>(nn, vector<int>(nn, -1)));
for (int i = 0; i < nn; ++i)
{
    for (int j = 0; j < nn; ++j)
    {
        for (int k = 0; k < nn; ++k)
        {
            if ((graph[i][j] > -1) && (graph[j][k] > -1))
            {
                sh[i][j][k] = graph[j][k] + graph[i][j];
            }
            else
            {
                sh[i][j][k] = -1;
            }
        }
    }
}


// Displaying shared table
for (int i = 0; i < nn; ++i)
{
    cout << "\n\nFor " << ch[i];
    for (int j = 0; j < nn; ++j)
    {
        cout << "\nFrom " << ch[j];
        for (int k = 0; k < nn; ++k)
        {
            cout << "\n"
                 << ch[k] << " " << sh[i][j][k];
        }
    }
}


// Updating final distances
vector<vector<int>> final(nn, vector<int>(nn, -1));
for (int i = 0; i < nn; ++i)
{
    for (int j = 0; j < nn; ++j)
    {
        final[i][j] = graph[i][j];
        via[i][j] = i;
        for (int k = 0; k < nn; ++k)
        {
            if ((final[i][j] > sh[i][k][j] || final[i][j] == -1) && sh[i][k][j] >-1)
            {
                final[i][j] = sh[i][k][j];
```

```
                    via[i][j] = k;
                }
            }
            if (final[i][j] == -1)
            {
                for (int k = 0; k < nn; ++k)
                {
                    if ((final[i][k] != -1) && (final[k][j] != -1))
                    {
                        if (final[i][j] == -1 || (final[i][j] != -1 && final[i][j] >
                                                  final[i][k] + final[k][j]))
                        {
                            if (final[i][k] + final[k][j] > -1)
                            {
                                final[i][j] = final[i][k] + final[k][j];
                                via[i][j] = k;
                            }
                        }
                    }
                }
            }
        }
    }


    cout << "\nAfter Update:";
    // Display table update
    for (int i = 0; i < nn; ++i)
    {
        cout << "\n"
             << ch[i] << " Table";
        cout << "\nNode\tDist\tVia";
        for (int j = 0; j < nn; ++j)
        {
            cout << "\n"
                 << ch[j] << "\t" << final[i][j] << "\t";
            if (i == via[i][j])
            {
                cout << "-";
            }
            else
            {
                cout << ch[via[i][j]];
            }
        }
    }
    cin.get(); // To pause the console
    cin.get(); // To wait for user input before closing
    return 0;
}
```

Output:

```
PS D:\Academics> g++ DVR.cpp -o dvr
PS D:\Academics> ./dvr

Enter Number of Nodes: 5

Enter Distance between A - B : 2

Enter Distance between A - C : 3

Enter Distance between A - D : 5

Enter Distance between A - E : 6

Enter Distance between B - C : 4

Enter Distance between B - D : 1

Enter Distance between B - E : 9

Enter Distance between C - D : 8

Enter Distance between C - E : 10

Enter Distance between D - E : 11

After Initialization:
A Table
Node    Dist    Via
A       0       -1
B       2       -1
C       3       -1
D       5       -1
E       6       -1
B Table
Node    Dist    Via
A       2       -1
B       0       -1
C       4       -1
D       1       -1
E       9       -1
C Table
Node    Dist    Via
A       3       -1
B       4       -1
C       0       -1
D       8       -1
E       10      -1
D Table
Node    Dist    Via
```

A 5 -1
B 1 -1
C 8 -1
D 0 -1
E 11 -1
E Table
Node Dist Via
A 6 -1
B 9 -1
C 10 -1
D 11 -1
E 0 -1

For A
From A
A 0
B 2
C 3
D 5
E 6
From B
A 4
B 2
C 6
D 3
E 11
From C
A 6
B 7
C 3
D 11
E 13
From D
A 10
B 6
C 13
D 5
E 16
From E
A 12
B 15
C 16
D 17
E 6

For B
From A
A 2
B 4
C 5
D 7
E 8
From B

A 2
B 0
C 4
D 1
E 9
From C
A 7
B 8
C 4
D 12
E 14
From D
A 6
B 2
C 9
D 1
E 12
From E
A 15
B 18
C 19
D 20
E 9

For C
From A
A 3
B 5
C 6
D 8
E 9
From B
A 6
B 4
C 8
D 5
E 13
From C
A 3
B 4
C 0
D 8
E 10
From D
A 13
B 9
C 16
D 8
E 19
From E
A 16
B 19
C 20

D 21
E 10

For D
From A
A 5
B 7
C 8
D 10
E 11
From B
A 3
B 1
C 5
D 2
E 10
From C
A 11
B 12
C 8
D 16
E 18
From D
A 5
B 1
C 8
D 0
E 11
From E
A 17
B 20
C 21
D 22
E 11

For E
From A
A 6
B 8
C 9
D 11
E 12
From B
A 11
B 9
C 13
D 10
E 18
From C
A 13
B 14
C 10
D 18

E 20
From D
A 16
B 12
C 19
D 11
E 22
From E
A 6
B 9
C 10
D 11
E 0
After Update:
A Table

Node	Dist	Via
A	0	-
B	2	-
C	3	-
D	3	B
E	6	-

B Table

Node	Dist	Via
A	2	-
B	0	-
C	4	-
D	1	-
E	8	A

C Table

Node	Dist	Via
A	3	-
B	4	-
C	0	-
D	5	B
E	9	A

D Table

Node	Dist	Via
A	3	B
B	1	-
C	5	B
D	0	-
E	10	B

E Table

Node	Dist	Via
A	6	-
B	8	A
C	9	A
D	10	B
E	0	-