Warshall

```
#include<iostream.h>
#include<conio.h>

class WarAlgo
{
	int adjMatrix[20][20];
	int duplicateMatrix[20][20];
	int n;
	public:
	void graph();
	void displayMatrix();
	WarAlgo()
	{
		int i,j;
		for(i=0;i<n;i++)
		{
			for(j=0;j<n;j++)
			{
				adjMatrix[i][j]=0;
				duplicateMatrix[i][j]=0;
			}
		}
	}
};

void WarAlgo::graph()
{
	int temp1, temp2,i,j,k;
	cout<<"Enter Number of vertices:";
	cin>>n;
	cout<<"\nEnter the cost of edges in adjagency matrix:";
	for(i=0;i<n;i++)
	{
		for(j=0;j<n;j++)
		{
			cout<<"\nadjMatrix["<<i<<"]["<<j<<"]: ";
			cin>>adjMatrix[i][j];
		}
	}
	cout<<"\nDisplaying Matrix:"<<"\n";
	for(i=0;i<n;i++)
	{
		for(j=0;j<n;j++)
		{
			cout<<adjMatrix[i][j]<<" ";
		}
		cout<<"\n";
	}

	for(k=0;k<n;k++)
```

```
	{
		for(i=0;i<n;i++)
		{
			for(j=0;j<n;j++)
			{
				temp1 = adjMatrix[i][j];
				temp2 = adjMatrix[i][k]+adjMatrix[k][j];
				if(temp1<temp2)
				{
					duplicateMatrix[i][j]  = temp1;
				}
				else
				{
					duplicateMatrix[i][j] = temp2;
				}
			}
		}
		for(i=0;i<n;i++)
		{
			for(j=0;j<n;j++)
			{
				adjMatrix[i][j] = duplicateMatrix[i][j];
			}
		}
	}
}

void WarAlgo::displayMatrix()
{
	cout<<"\nFinal Adjagency Matrix:"<<"\n";
	for(int i=0;i<n;i++)
	{
		for(int j=0;j<n;j++)
		{
			cout<<adjMatrix[i][j]<<"\t";
		}
		cout<<"\n";
	}

}

void main(){
	clrscr();
	WarAlgo w;
	w.graph();
	w.displayMatrix();
	getch();
}
```

Cost Adjusted matrix

```
#include<iostream.h>
#include<conio.h>

class WarAlgo
{
	int adjMatrix[20][20];
	int n;
	public:
	void adjgraph();
	void displayMatrix();
};

void WarAlgo::adjgraph()
{
	int i,j,k;
	cout<<"Enter Number of vertices:";
	cin>>n;
	cout<<"\nEnter the value of edges in adjagency matrix:"<<"\n";
	cout<<"if direct edge then enter 1 otherwise 0";
	for(i=0;i<n;i++)
	{
		for(j=0;j<n;j++)
		{
			cout<<"\nadjMatrix["<<i<<"]["<<j<<"]: ";
			cin>>k;
			if(k==0||k==1)
			{
				adjMatrix[i][j]=k;
			}
			else
			{
				cout<<"Enter correct value";
			}
		}
	}

}

void WarAlgo::displayMatrix()
{
	cout<<"\nFinal Adjagency Matrix:"<<"\n";
	for(int i=0;i<n;i++)
	{
		for(int j=0;j<n;j++)
		{
			cout<<adjMatrix[i][j]<<"\t";
		}
		cout<<"\n";
	}
```

```
}

int main(){
	clrscr();
	WarAlgo w;
	w.adjgraph();
	w.displayMatrix();
	getch();
	return 0;
}
```

Directed graph

```
#include<iostream.h>
#include<conio.h>

class AdjagencyMatrix
{
	int adjMatrix[20][20];
	int n;
	public:
	void createGraph();
	void insertVertex();
	void deleteVertex();
	void insertEdge(int, int);
	void deleteEdge(int, int);
	void displayMatrix();
	AdjagencyMatrix()
	{
		for(int i=0;i<20;i++)
		{
			for(int j=0;j<20;j++)
			{
				adjMatrix[i][j]=0;
			}
		}
	}
};

void AdjagencyMatrix::createGraph()
{
	int i, maxEdge, origin, destination;
	cout<<"Enter Number of vertices:";
	cin>>n;
	maxEdge = (n*(n-1));
	cout<<"\nEnter the value of edges in adjagency matrix:"<<"\n";
	for(i=1;i<=maxEdge;i++)
	{
		cout<<"Enter 0 0 to exit or enter origin and destination for: "<<i<<"\n";
		cin>>origin>>destination;
		if((origin==0) ||(destination == 0))
		{
			break;
		}
		if((origin>n) || (origin<0) || (destination>n) || (destination<0))
		{
			cout<<"Invalid inputs"<<"\n";
			i--;
			return;
		}
		else
		{
			adjMatrix[origin][destination] = 1;
```

```
                    //    adjMatrix[destination][origin] = 1;
                }
        }

}

void AdjagencyMatrix::insertVertex()
{
        int i;
        n++;
        cout<<"Number of vertices are:"<<n;
        for(i=0;i<=n;i++)
        {
                adjMatrix[i][n] = 0;
                adjMatrix[n][i] = 0;
        }
}

void AdjagencyMatrix::insertEdge(int origin, int destination)
{
        if((origin > n) || (origin>n))
        {
                cout<<"Source or Destination does not exist"<<"\n";
                return;
        }
        adjMatrix[origin][destination] = 1;
    //  adjMatrix[destination][origin] = 1;
}

void AdjagencyMatrix::deleteVertex()
{
        int i;
        cout<<"Number of vertices are:"<<n;
        for(i=0;i<=n;i++)
        {
                adjMatrix[i][n] = 0;
                adjMatrix[n][i] = 0;
        }
        n--;
}


void AdjagencyMatrix::deleteEdge(int origin, int destination)
{
        if((origin > n) || (origin>n))
        {
                cout<<"Source or Destination does not exist"<<"\n";
                return;
        }
        adjMatrix[origin][destination] = 0;
    //  adjMatrix[destination][origin] = 0;
```

```
}


void AdjagencyMatrix::displayMatrix(void)
{
	cout<<"\nFinal Adjagency Matrix:"<<"\n";
	for(int i=1;i<=n;i++)
	{
		for(int j=1;j<=n;j++)
		{
			cout<<adjMatrix[i][j]<<"\t";
		}
		cout<<"\n";
	}

}

void main()
{
	clrscr();

	AdjagencyMatrix a;
	char ch='y';
	int origin, destination, option;
	while(ch=='y')
	{
		cout<<"Selection operation:"<<"\n";
		cout<<"1. Create Graph"<<"\n";
		cout<<"2. Insert Vertex"<<"\n";
		cout<<"3. Insert Edge"<<"\n";
		cout<<"4. Delete Vertex"<<"\n";
		cout<<"5. Delete Edge"<<"\n";
		cout<<"6. Display Final Matrix"<<"\n";
		cout<<"Enter an option: ";
		cin>>option;
		switch(option)
		{
			case 1:
			{
				a.createGraph();
				cout<<"\nDo you want to continue?";
				break;
			}
			case 2:
			{
				a.insertVertex();
				a.displayMatrix();
				cout<<"\nDo you want to continue?";
				break;
			}
			case 3:
```

```
                        {
                                cout<<"Enter source & Destination:";
                                cin>>origin>>destination;
                                a.insertEdge(origin, destination);
                                a.displayMatrix();
                                cout<<"\nDo you want to continue?";
                                break;
                        }
                        case 4:
                        {
                                a.deleteVertex();
                                a.displayMatrix();
                                cout<<"\nDo you want to continue?";
                                break;
                        }
                        case 5:
                        {
                                cout<<"Enter source & Destination:";
                                cin>>origin>>destination;
                                a.deleteEdge(origin, destination);
                                a.displayMatrix();
                                cout<<"\nDo you want to continue?";
                                break;
                        }
                        case 6:
                        {
                                a.displayMatrix();
                                cout<<"\nDo you want to continue?";
                                break;
                        }
                }
                cin>>ch;
        }getch();
}
```

Undirected graph

```
#include<iostream.h>
#include<conio.h>

class AdjagencyMatrix
{
	int adjMatrix[20][20];
	int n;
	public:
	void createGraph();
	void insertVertex();
	void deleteVertex();
	void insertEdge(int, int);
	void deleteEdge(int, int);
	void displayMatrix();
	AdjagencyMatrix()
	{
		for(int i=0;i<20;i++)
		{
			for(int j=0;j<20;j++)
			{
				adjMatrix[i][j]=0;
			}
		}
	}
};

void AdjagencyMatrix::createGraph()
{
	int i, maxEdge, origin, destination;
	cout<<"Enter Number of vertices:";
	cin>>n;
	maxEdge = (n*(n-1))/2;
	cout<<"\nEnter the value of edges in adjagency matrix:"<<"\n";
	for(i=1;i<=maxEdge;i++)
	{
		cout<<"Enter 0 0 to exit or enter origin and destination for: "<<i<<"\n";
		cin>>origin>>destination;
		if((origin==0) ||(destination == 0))
		{
			break;
		}
		if((origin>n) || (origin<0) || (destination>n) || (destination<0))
		{
			cout<<"Invalid inputs"<<"\n";
			i--;
			return;
		}
		else
		{
			adjMatrix[origin][destination] = 1;
```

```
                        adjMatrix[destination][origin] = 1;
                }
        }

}

void AdjagencyMatrix::insertVertex()
{
        int i;
        n++;
        cout<<"Number of vertices are:"<<n;
        for(i=0;i<=n;i++)
        {
                adjMatrix[i][n] = 0;
                adjMatrix[n][i] = 0;
        }
}

void AdjagencyMatrix::insertEdge(int origin, int destination)
{
        if((origin > n) || (origin>n))
        {
                cout<<"Source or Destination does not exist"<<"\n";
                return;
        }
        adjMatrix[origin][destination] = 1;
        adjMatrix[destination][origin] = 1;
}

void AdjagencyMatrix::deleteVertex(void)
{
        int i;
        cout<<"Number of vertices are:"<<n;
        for(i=0;i<=n;i++)
        {
                adjMatrix[i][n] = 0;
                adjMatrix[n][i] = 0;
        }
        n--;
}


void AdjagencyMatrix::deleteEdge(int origin, int destination)
{
        if((origin > n) || (origin>n))
        {
                cout<<"Source or Destination does not exist"<<"\n";
                return;
        }
        adjMatrix[origin][destination] = 0;
        adjMatrix[destination][origin] = 0;
```

```
}


void AdjagencyMatrix::displayMatrix(void)
{
	cout<<"\nFinal Adjagency Matrix:"<<"\n";
	for(int i=1;i<=n;i++)
	{
		for(int j=1;j<=n;j++)
		{
			cout<<adjMatrix[i][j]<<" ";
		}
		cout<<"\n";
	}

}

void main()
{
	clrscr();

	AdjagencyMatrix a;
	char ch='y';
	int origin, destination, option;
	while(ch=='y')
	{
		cout<<"Selection operation:"<<"\n";
		cout<<"1. Create Graph"<<"\n";
		cout<<"2. Insert Vertex"<<"\n";
		cout<<"3. Insert Edge"<<"\n";
		cout<<"4. Delete Vertex"<<"\n";
		cout<<"5. Delete Edge"<<"\n";
		cout<<"6. Display Final Matrix"<<"\n";
		cout<<"Enter an option: ";
		cin>>option;
		switch(option)
		{
			case 1:
			{
				a.createGraph();
				cout<<"\nDo you want to continue?";
				break;
			}
			case 2:
			{
				a.insertVertex();
				a.displayMatrix();
				cout<<"\nDo you want to continue?";
				break;
			}
			case 3:
```

```
                        {
                                cout<<"Enter source & Destination:";
                                cin>>origin>>destination;
                                a.insertEdge(origin, destination);
                                a.displayMatrix();
                                cout<<"\nDo you want to continue?";
                                break;
                        }
                        case 4:
                        {
                                a.deleteVertex();
                                a.displayMatrix();
                                cout<<"\nDo you want to continue?";
                                break;
                        }
                        case 5:
                        {
                                cout<<"Enter source & Destination:";
                                cin>>origin>>destination;
                                a.deleteEdge(origin, destination);
                                a.displayMatrix();
                                cout<<"\nDo you want to continue?";
                                break;
                        }
                        case 6:
                        {
                                a.displayMatrix();
                                cout<<"\nDo you want to continue?";
                                break;
                        }
                }
                cin>>ch;
        }getch();
}
```

Graph Traversal DFS

```
#include<iostream.h>
#include<conio.h>
class gtdfs
{
	int adj[10][10],varr[10],n;
	public:
	gtdfs()
	{
		int i,j;
		for(i=0;i<n;i++)
		{
			for(j=0;j<n;j++)
				adj[i][j]=0;
			varr[i]=0;
		}
	}
	void buildadj();
	void dfs(int);
};
void gtdfs::buildadj()
{
	int i,j;
	cout<<"\nEnter number of vertex: ";
	cin>>n;
	cout<<"\nEnter the edges: ";
	for(i=0;i<n;i++)
	{
		for(j=0;j<n;j++)
			cin>>adj[i][j];
	}
}
void gtdfs::dfs(int x)
{
	int j;
	varr[x]=1;
	cout<<"\n"<<x<<" is visited";
	for(j=0;j<n;j++)
	{
		if(adj[x][j]==1 && varr[j]==0)
			dfs(j);
	}
}
void main() {
	clrscr();
	gtdfs g;
	int i;
	g.buildadj();
	g.dfs(0);
	getch();
}
```