Aim: write a program in c to implement all pairs shortest path using floyed's algorithm

Algorithm:

1. Initialize the distances between all pairs of nodes as infinity or zero (if the nodes are the same).
2. Get the distances between all pairs of nodes from user input.
3. Find all pairs shortest path by repeatedly checking if there is a shorter path through a intermediate node.
4. Repeat step 3 for all possible intermediate nodes.
5. The final distances between all pairs of nodes will be the solution to the problem.

Source Code:

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

#define MAX\_NODES 100

#define MAX\_DISTANCE INT\_MAX

*int* n;

*int* distances[MAX\_NODES][MAX\_NODES];

*void* floyd() {

  for (*int* i = 0; i < n; i++) {

    for (*int* j = 0; j < n; j++) {

      if (i == j) {

        distances[i][j] = 0;

      } else {

        distances[i][j] = MAX\_DISTANCE;

      }

    }

  }

  for (*int* i = 0; i < n; i++) {

    for (*int* j = 0; j < n; j++) {

*int* distance;

      scanf("%d", &distance);

      distances[i][j] = distance;

    }

  }

  for (*int* k = 0; k < n; k++) {

    for (*int* i = 0; i < n; i++) {

      for (*int* j = 0; j < n; j++) {

        if (distances[i][j] > distances[i][k] + distances[k][j]) {

          distances[i][j] = distances[i][k] + distances[k][j];

        }

      }

    }

  }

}

*int* main() {

  printf("Enter the number of nodes: ");

  scanf("%d", &n);

  floyd();

  printf("All pairs shortest path:\n");

  for (*int* i = 0; i < n; i++) {

    for (*int* j = 0; j < n; j++) {

      printf("%d ", distances[i][j]);

    }

    printf("\n");

  }

  return 0;

}

Output:

Enter the number of nodes: 4

0 3 9999 5

2 0 9999 4

9999 1 0 9999

9999 9999 2 0

All pairs shortest path:

0 3 7 5

2 0 6 4

3 1 0 5

5 3 2 0