```c
#include <stdio.h>
#define N 6

// Floyd Algorithmus iterativ abarbeiten
void floyd(int n, int am[][N], int zm[][N]) {
   int i=0, j=0, k=0;
   for (i=0; i<n; i++)                // alle Knoten in beliebiger Reihenfolge
      for (j=0; j<n; j++)             // von Knoten
         for (k=0; k<n; k++)          // nach Knoten
            // wenn Verbindungen (Kanten) j->i und i->k existierten
            if (am[j][i] >= 0 && am[i][k] >= 0)
               if ( am[j][k] < 0 || (am[j][k] > am[j][i] + am[i][k]) ){ // Verb. kürzer
                  am[j][k] = am[j][i] + am[i][k];  // Neue Kante j->k
                  zm[j][k] = i;               // Verbindung j->k ueber Zwischenknoten i
                  printf("neue Verbind. von Knoten %d nach Knoten %d ueber Knoten %d\n",
                          j,k,i);
               }
}

// Verbindung über Zwischenknoten
void zknoten(int i, int j, int zm[][N]) {
   if (zm[i][j] < 0)  printf("%3d", j);
   else {
           zknoten(i,zm[i][j], zm); // rekursiver Aufruf
           zknoten(zm[i][j],j, zm); // rekursiver Aufruf
   }
}

// Ausgabe der Ergebnisse
void output(int n, int am[][N], int zm[][N]) {
   int i=0, j=0;
   for (i=0; i<n; i++)
      for (j=0; j<n; j++)
         if (i != j && am[i][j] >= 0)
            if (zm[i][j] >= 0) {
               printf("von Knoten %d nach Knoten %d Distance %d\t(%3d",i,j,am[i][j],i);
               zknoten(i,j,zm);
               printf(" )\n");
            }  else printf("von Knoten %d nach Knoten %d Distance %d\n", i,j,am[i][j]);
}

// Initialisierung Matrizen und Vektoren
void init(int n, int am[][N], int zm[][N]){
   int i=0, j=0;
   for (i=0; i<n; i++)
      for (j=0; j<n; j++) {
         if (i==j)
            am[i][j] = 0;  // Entfernung zu sich selbst ist 0
         else
            am[i][j] = -1;   // sonst keine Verbindung
         zm[i][j] = -1;      // keine Verbindung ueber Zwischenknoten
      }
}
```

```c
void out_zm(int zm[][N], int n)
{
   int i=0, j=0;
   printf("\nMatrix zum Aufloesen der Zwischenknoten:\n");
   printf("     nach | ");

   for (i = 0; i < n; i++)
   {
      printf("%5d", i);
   }

   printf("\n------------");
   for (i = 0; i < n; i++){
      printf("-----");
   }
   printf("\n");

   for (i = 0; i < n; i++)
   {
      if(i==0)
         printf("von %5d | ", i);
       else
         printf("    %5d | ", i);
      for (j = 0; j < n; j++)
         printf("%5d", (zm[i][j] >= 0) ? zm[i][j]:-1);
      printf("\n");
   }
}

int main(){
   int am[N][N];    // Instantiierung Matrizen und Vektoren
   int zm[N][N];
   init(N, am, zm); // Initialisierung Matrizen und Vektoren

   am[0][1] = 30;   // Kanten eintragen (Beispieldaten)
   am[0][4] = 100;
   am[0][5] = 90;
   am[1][2] = 10;
   am[1][3] = 40;
   am[2][0] = 40;
   am[2][5] = 10;
   am[3][4] = 30;
   am[5][4] = 20;
   printf("Adjazenzmatrix fuer n = %d Knoten\n", N);
   output(N, am, zm);
   printf("\nStart Floyd-Algorithmus\n");
   floyd(N, am, zm);

   printf("\nEntfernungen:\n");
   output(N, am, zm);
   out_zm(zm, N);
   getchar();
}
```

```
/*
Adjazenzmatrix fuer n = 6 Knoten
von Knoten 0 nach Knoten 1 Distance 30
von Knoten 0 nach Knoten 4 Distance 100
von Knoten 0 nach Knoten 5 Distance 90
von Knoten 1 nach Knoten 2 Distance 10
von Knoten 1 nach Knoten 3 Distance 40
von Knoten 2 nach Knoten 0 Distance 40
von Knoten 2 nach Knoten 5 Distance 10
von Knoten 3 nach Knoten 4 Distance 30
von Knoten 5 nach Knoten 4 Distance 20

Start Floyd-Algorithmus
neue Verbindung von Knoten 2 nach Knoten 1 ueber Knoten 0
neue Verbindung von Knoten 2 nach Knoten 4 ueber Knoten 0
neue Verbindung von Knoten 0 nach Knoten 2 ueber Knoten 1
neue Verbindung von Knoten 0 nach Knoten 3 ueber Knoten 1
neue Verbindung von Knoten 2 nach Knoten 3 ueber Knoten 1
neue Verbindung von Knoten 0 nach Knoten 5 ueber Knoten 2
neue Verbindung von Knoten 1 nach Knoten 0 ueber Knoten 2
neue Verbindung von Knoten 1 nach Knoten 4 ueber Knoten 2
neue Verbindung von Knoten 1 nach Knoten 5 ueber Knoten 2
neue Verbindung von Knoten 1 nach Knoten 4 ueber Knoten 3
neue Verbindung von Knoten 0 nach Knoten 4 ueber Knoten 5
neue Verbindung von Knoten 1 nach Knoten 4 ueber Knoten 5
neue Verbindung von Knoten 2 nach Knoten 4 ueber Knoten 5

Entfernungen:
von Knoten 0 nach Knoten 1 Distance 30
von Knoten 0 nach Knoten 2 Distance 40  (  0  1  2 )
von Knoten 0 nach Knoten 3 Distance 70  (  0  1  3 )
von Knoten 0 nach Knoten 4 Distance 70  (  0  1  2  5  4 )
von Knoten 0 nach Knoten 5 Distance 50  (  0  1  2  5 )
von Knoten 1 nach Knoten 0 Distance 50  (  1  2  0 )
von Knoten 1 nach Knoten 2 Distance 10
von Knoten 1 nach Knoten 3 Distance 40
von Knoten 1 nach Knoten 4 Distance 40  (  1  2  5  4 )
von Knoten 1 nach Knoten 5 Distance 20  (  1  2  5 )
von Knoten 2 nach Knoten 0 Distance 40
von Knoten 2 nach Knoten 1 Distance 70  (  2  0  1 )
von Knoten 2 nach Knoten 3 Distance 110 (  2  0  1  3 )
von Knoten 2 nach Knoten 4 Distance 30  (  2  5  4 )
von Knoten 2 nach Knoten 5 Distance 10
von Knoten 3 nach Knoten 4 Distance 30
von Knoten 5 nach Knoten 4 Distance 20

Matrix zum Aufloesen der Zwischenknoten:
     nach |     0    1    2    3    4    5
-----------------------------------------
von     0 |    -1   -1    1    1    5    2
        1 |     2   -1   -1   -1    5    2
        2 |    -1    0   -1    1    5   -1
        3 |    -1   -1   -1   -1   -1   -1
        4 |    -1   -1   -1   -1   -1   -1
        5 |    -1   -1   -1   -1   -1   -1
*/
```