```c
/*
* Este programa implementa o algoritmo de Dijkstra para o problema do
* caminho de custo minimo em grafos dirigidos com custos positivos nas
* arestas.
*
* @autor : vanderson lucio
* @e-mail: vanderson.gold@gmail.com
*
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define FLSH gets(l)

int destino, origem, vertices = 0;
int custo, *custos = NULL;

void dijkstra(int vertices,int origem,int destino,int *custos)
{
    int i,v, cont = 0;
    int *ant, *tmp;
    int *z;     /* vertices para os quais se conhece o caminho minimo */
    double min;
    double dist[vertices]; /* vetor com os custos dos caminhos */


    /* aloca as linhas da matriz */
    ant = calloc (vertices, sizeof(int *));
    tmp = calloc (vertices, sizeof(int *));
    if (ant == NULL) {
        printf ("** Erro: Memoria Insuficiente **");
        exit(-1);
    }

    z = calloc (vertices, sizeof(int *));
    if (z == NULL) {
        printf ("** Erro: Memoria Insuficiente **");
        exit(-1);
    }

    for (i = 0; i < vertices; i++) {
        if (custos[(origem - 1) * vertices + i] !=- 1) {
            ant[i] = origem - 1;
            dist[i] = custos[(origem-1)*vertices+i];
        }
        else {
            ant[i]= -1;
            dist[i] = HUGE VAL;
        }
        z[i]=0;
    }
    z[origem-1] = 1;
    dist[origem-1] = 0;

    /* Laco principal */
    do {

        /* Encontrando o vertice que deve entrar em z */
        min = HUGE VAL;
        for (i=0;i<vertices;i++)
            if (!z[i])
                if (dist[i]>=0 && dist[i]<min) {
                    min=dist[i];v=i;
                }

        /* Calculando as distancias dos novos vizinhos de z */
        if (min != HUGE VAL && v != destino - 1) {
            z[v] = 1;
```

```
71              for (i = 0; i < vertices; i++)
72                  if (!z[i]) {
73                      if (custos[v*vertices+i] != -1 && dist[v] + custos[v*vertices+i
                        ] < dist[i]) {
74                          dist[i] = dist[v] + custos[v*vertices+i];
75                          ant[i] =v;
76                      }
77                  }
78          }
79      } while (v != destino - 1 && min != HUGE VAL);
80
81      /* Mostra o Resultado da busca */
82      printf("\tDe %d para %d: \t", origem, destino);
83      if (min == HUGE VAL) {
84          printf("Nao Existe\n");
85          printf("\tCusto: \t- \n");
86      }
87      else {
88          i = destino;
89          i = ant[i-1];
90          while (i != -1) {
91          //  printf("<-%d",i+1);
92              tmp[cont] = i+1;
93              cont++;
94              i = ant[i];
95          }
96
97          for (i = cont; i > 0 ; i--) {
98              printf("%d -> ", tmp[i-1]);
99          }
100         printf("%d", destino);
101
102         printf("\n\tCusto:  %d\n",(int) dist[destino-1]);
103     }
104 }
105
106 void limpar(void)
107 {
108      printf("\033[2J"); /* limpa a tela */
109      printf("\033[1H"); /* poe o curso no topo */
110 }
111
112 void cabecalho(void)
113 {
114     limpar();
115     printf("Implementacao do Algoritmo de Dijasktra\n");
116     printf("Comandos:\n");
117     printf("\t d - Adicionar um Grafo\n"
118            "\t r - Procura Os Menores Caminhos no Grafo\n"
119            "\t CTRL+c - Sair do programa\n");
120     printf(">>> ");
121 }
122
123 void add(void)
124 {
125     int i, j;
126
127     do {
128         printf("\nInforme o numero de vertices (no minimo 2 ): ");
129         scanf("%d",&vertices);
130     } while (vertices < 2 );
131
132     if (!custos)
133         free(custos);
134     custos = (int *) malloc(sizeof(int)*vertices*vertices);
135     for (i = 0; i <= vertices * vertices; i++)
136         custos[i] = -1;
137
138     printf("Entre com as Arestas:\n");
139     do {
```

```
140             do {
141                 printf("Origem da aresta (entre 1 e %d ou '0' para sair): ", vertices);
142                 scanf("%d",&origem);
143             } while (origem < 0 || origem > vertices);
144 
145             if (origem) {
146                 do {
147                     printf("Destino da aresta (entre 1 e %d, menos %d): ", vertices,
                        origem);
148                     scanf("%d", &destino);
149                 } while (destino < 1 || destino > vertices || destino == origem);
150 
151                 do {
152                     printf("Custo (positivo) da aresta do vertice %d para o vertice
                        %d: ",
153                             origem, destino);
154                     scanf("%d",&custo);
155                 } while (custo < 0);
156                 //alteracao
157                 custos[(origem-1) * vertices + destino - 1] = custo;
158                 if(custos[(origem-1) * vertices + destino - 1]!=-1&&custos[(destino-1)
                    * vertices + origem - 1]!=-1){
159                   custos[(origem-1) * vertices + destino - 1]=0;
160                   custos[(destino-1) * vertices + origem - 1]=0;
161                 }
162 
163                 //fim alteracao
164             }
165 
166         } while (origem);
167     }
168 
169     void procurar(void)
170     {
171         int i, j;
172 
173         /* Azul */
174         printf("\033[36;1m");
175         printf("Lista dos Menores Caminhos no Grafo Dado: \n");
176 
177         for (i = 1; i <= vertices; i++) {
178             for (j = 1; j <= vertices; j++)
179                 dijkstra(vertices, i,j, custos);
180             printf("\n");
181         }
182 
183         printf("<Pressione ENTER para retornar ao menu principal>\n");
184         /* Volta cor nornal */
185         printf("\033[m");
186     }
187 
188 
189 
190     int main(int argc, char **argv) {
191         int i, j;
192         char opcao[3], l[50];
193         int e;
194         e=5;
195         do {
196 
197             cabecalho();
198             scanf("%s", &opcao);
199 
200             if ((strcmp(opcao, "d")) == 0) {
201                 add();
202             }
203             FLSH;
204 
205 
206             if ((strcmp(opcao, "r") == 0) && (vertices > 0) ) {
```

```
207
208             procurar();
209               FLSH;
210           }
211
212       } while (opcao != "x");
213
214       printf("\nAte a proxima...\n\n");
215
216       return 0;
217   }
218
```