INF111 – TP5

Lucas Pastor

1 Exercice 5.5

```c
#include <stdio.h>

/* Nombre maximum de villes. */
#define NMAX 16

int main(int argc, char *argv[])
{
    /* Nombre de villes. */
    int nbVilles;

    /* Numéro de la ville de départ. */
    int villeDepart;

    /* Distances inter-villes. */
    int distances[NMAX + 1][NMAX + 1];

    /* Villes déjà visitées. */
    int dejaVisitee[NMAX + 1];

    /* Etapes de l'itinéraire. */
    int etapes[NMAX + 2];

    /* Distances parcourues depuis la ville de départ. */
    int distanceCumulee[NMAX + 2];

    /* Numéro de la ville visitée lors de chaque étape. */
    int villeCourante;

    /* Indices de boucle. */
    int i, j, k;

    /* Distance saisie par l'utilisateur. */
    int d;

    /* Ville ayant la plus petite distance par rapport à la ligne
     * en cours.
     */
    int vmin;

    /* On demande à l'utilisateur le nombre de villes à visiter. */
    printf("Entrez le nombre de villes à visiter : ");
    scanf("%d", &nbVilles);
    printf("%d.\n", nbVilles);

    /* Ici on va récupérer les distances inter-villes. */
    for (i = 1; i <= nbVilles; i++)
    {
        /* La diagonale de la matrice est à 0 (la distance une ville et elle
         * même est nulle).
         */
        distances[i][i] = 0;

        for (j = i + 1; j <= nbVilles; j++)
        {
            /* On demande à l'utilisateur la distance entre la ville i et la ville j. */
            printf("Distance ville %d ville %d : ", i, j);
            scanf("%d", &d);
```

```
58              printf("%d.\n", d);
59
60              /* On met à jour les distances dans le tableau correspondant.
61               * La distance entre la ville i et la ville j est la même
62               * distance qu'entre la ville j et i.
63               */
64              distances[i][j] = d;
65              distances[j][i] = d;
66          }
67      }
68
69      /* On demande à l'utilisateur le numéro de la ville de départ. */
70      printf("Entrez le numéro de la ville de départ : ");
71      scanf("%d", &villeDepart);
72      printf("%d.\n", villeDepart);
73
74      /* Affichages. */
75      printf("Nombre de villes à visiter : %d.\n", nbVilles);
76      printf("Numéro de la ville de départ : %d.\n", villeDepart);
77      printf("Tableau des distances : \n");
78
79      /* Affichage des trois premières lignes du tableau. */
80      for (i = 0; i <= nbVilles; i++)
81      {
82          printf("------");
83      }
84      printf("\n");
85      printf("ville|");
86      for (i = 1; i <= nbVilles; i++)
87      {
88          printf("%5d|", i);
89      }
90      printf("\n");
91
92
93      /* Affichage du tableau des distances. */
94      for (i = 1; i <= nbVilles; i++)
95      {
96          for (k = 0; k <= nbVilles; k++)
97          {
98              printf("-----|");
99          }
100
101         printf("\n");
102         printf("%5d|", i);
103         for (j = 1; j <= nbVilles; j++)
104         {
105             printf("%5d|", distances[i][j]);
106         }
107         printf("\n");
108
109     }
110
111     /* Dernière ligne du tableau. */
112     for (i = 0; i <= nbVilles; i++)
113     {
114         printf("------");
115     }
116     printf("\n");
117
118
119     /* Initialement, on a visité la ville de départ. */
120     /* Les autres ne sont pas visitées. */
121     for (i = 1; i <= nbVilles; i++)
122     {
123         dejaVisitee[i] = 0;
124     }
125
126     dejaVisitee[villeDepart] = 1;
127
128     /* La première étape est la ville de départ. */
```

```
129     etapes[1] = villeDepart;
130
131     /* La distance parcourue à la première étape des nulles. */
132     distanceCumulee[1] = 0;
133
134     /* La ville en cours de visite est la ville de départ. */
135     villeCourante = villeDepart;
136
137     for (i = 2; i <= nbVilles; i++)
138     {
139         /* On recherche la ville vmin non visitée la plus proche de la ville
140          * courante. Ceci revient à chercher le minimum dans la ligne
141          * de la ville en cours dans la tableau distance.
142          * On fera attention de vérifier que la ville qu'on teste
143          * n'est pas déjà visitée.
144          */
145
146         /* Arbitrairement, la ville la plus petite est la ville suivante
147          * qui n'est pas déjà visitée (modulo nbVilles).
148          */
149
150         vmin = (villeCourante % nbVilles) + 1;
151         while (dejaVisitee[vmin] == 1)
152         {
153             vmin = (vmin % nbVilles) + 1;
154         }
155
156         for (j = 1; j <= nbVilles; j++)
157         {
158             /* Si la ville n'est pas déjà visitée, on regarde
159              * si la distance est plus petite.
160              */
161             if (dejaVisitee[j] == 0 &&
162                 distances[villeCourante][j] < distances[villeCourante][vmin])
163             {
164                 vmin = j;
165             }
166
167         }
168         printf("vmin = %d.\n", vmin);
169
170         /* On mémorise la ville trouvée. */
171         dejaVisitee[vmin] = 1;
172
173         /* Lors de la ième étape, on est passé
174          * par la ville vmin.
175          */
176         etapes[i] = vmin;
177
178         /* On calcul la distance cumulée. */
179         distanceCumulee[i] = distanceCumulee[i - 1] + distances[villeCourante][vmin];
180
181         /* Mise à jour de la nouvelle ville en cours. */
182         villeCourante = vmin;
183     }
184
185     /* La dernière étape consiste à revenir à la ville
186      * de départ.
187      */
188     etapes[nbVilles + 1] = villeDepart;
189
190     /* Calcul final de la distance cumulée. */
191     distanceCumulee[nbVilles + 1] =
192         distanceCumulee[nbVilles] + distances[etapes[nbVilles]][villeDepart];
193
194     /* Affichage des résultats. */
195     printf("No étape | Ville visitée | Distance cumulée\n");
196
197     for (i = 1; i <= nbVilles + 1; i++)
198     {
199         printf("---------|---------------|-----------------\n");
```

```
200             printf("%9d|%15d|%17d\n", i, etapes[i], distanceCumulee[i]);
201         }
202
203
204         return 0;
205 }
```