C. PROGRAMA EN C

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define MAX_ELEMENT 200



int inicializar (int**);
int inicializar_calles_posic (int[],int[]);
int buscar_calles (int,int,int,int[],int[][]);
int buscar_tramos (int,int,int,int[],int**);
int numero_filas (int,int,int[],int[],int**);
int actualiza_final (int,int,int,int,int,int[],int[],float[],int**,int[]);
int borrar_registros (int,int**);
int tramos_recorridos (int,int,int**);
int actualizar (int,int,int[],int[]);
int recorre_todas (int,int,int**,int[]);
int imprimir_result (int,int,float[],int**);



int main (int argc,char *argv[])
{
  int tramos,seleccionados,i,j,num1;
  float km[MAX_ELEMENT];
  int precedencias[MAX_ELEMENT][MAX_ELEMENT];
  char salir;
  int primera_vez,cont,tramo_actual,num_calles,num_posic;
  int ultimo;
  int calles[8],posiciones[3500],siguiente[150],seleccion[150];
  int **final;
  int final_bucle;
  final_bucle =0;
  final=(int **) calloc(6000,sizeof(int *));
  for(i=0;i<=6000;i++)
    final[i]=(int *) calloc(6000,sizeof(int));
```

```
//L E E R    D A T O S
  num1=1;
  FILE * entrada1;FILE * entrada2; FILE * entrada3;
  entrada1=fopen("bellavista-1(km).txt","rt");
  entrada2=fopen("bellavista-1(conexiones).txt","rt");
  entrada3=fopen("bellavista-1(seleccion).txt","rt");
  fscanf(entrada1,"%d",&tramos);
  fscanf(entrada3,"%d",&seleccionados);
  printf("\n\n\n>>>DATOS DE ENTRADA\n\n");
  printf("TRAMOS Y KILOMETROS\n\n");
  for (i=1;i<=tramos;i++)
  {
    fscanf(entrada1,"%f",&km[i]);
    printf("%3d\t",num1);
    printf("%0.3f\n",km[i]);num1++;
  }
  system("PAUSE");

  printf("\n\nMATRIZ DE CONEXIONES\n\n");num1=1;
  for (i=1;i<=tramos;i++)
  {
    printf("%3d ",num1);
    for (j=1;j<=tramos;j++)
    {
      fscanf(entrada2,"%d",&precedencias[i][j]);
      printf("%d",precedencias[i][j]);
    }
    printf("\n");num1++;
  }
  system("PAUSE");

  printf("\n\nTRAMOS PARA RECORRER\n\n");num1=1;
  for (i=1;i<=seleccionados;i++)
  {

    fscanf(entrada3,"%d",&seleccion[i]);
    printf(" %d",seleccion[i]);printf("\n");
  }
  system("PAUSE");
```

```
// ENCONTRAR TODAS LAS POSIBLES SOLUCIONES
printf("\n\n>>>SOLUCIONES\n\n");
inicializar(final);
primera_vez=0;
// Inicializa datos
 do{
// Comprobar si es la primera vez que se mete en el bucle
  if (primera_vez==0)
  {
     primera_vez=1;
     tramo_actual=1;
     ultimo=1;
  }
// Guarda en un vector todos los tramos de las calles a las
// que pueden ir a partir del tramo actual
   inicializar_calles_posic(calles,posiciones);
   buscar_calles (cont,tramo_actual,tramos,calles,precedencias);
// Recorre la matriz final para buscar en que posiciones se
// encuentra el tramo actual para después actualizarlo
   buscar_tramos (tramo_actual,tramos,ultimo,posiciones,final);
// busca el número de filas que tienen calles y final
   int numero;
   numero=1;
   num_calles=numero_filas(numero,tramo_actual,calles,posiciones,final);
   numero=2;
   num_posic=numero_filas(numero,tramo_actual,calles,posiciones,final);
   numero=3;
   ultimo=numero_filas(numero,tramo_actual,calles,posiciones,final);
// Actualiza la matriz final
   ultimo=actualiza_final(tramos,ultimo,num_calles,num_posic,tramo_actual,
              calles,posiciones,km,final,seleccion);
// Comprueba si ya se han recorrido todos los tramos
   final_bucle=tramos_recorridos(ultimo,final_bucle,final);
// Actualiza el tramo actual
   tramo_actual=actualizar(tramo_actual,num_calles,calles,siguiente);
 }while (final_bucle==0);


//Imprime por pantalla los casos que por algún motivo fuera de lo
//normal (p.ej. necesitar más filas o columnas, etc) no han
//podido procesarse
```

```
  for (i=1;i<=ultimo;i++)
  {
    printf("\n");
    for (j=1;j<=400;j++)
    {
      if (final[i][j]!=0)
      {
        printf("%d ",final[i][j]);
      }
    }
  }

//Libera espacio de la matriz de punteros
free(final);



//SALIR
  printf("\n\n\n>>>FIN DE LA APLICACION\n\n");
  system("PAUSE");return 0;

}




//>>>>>>>>>>>>>>>>>>>>>>>>F U N C I O N E S<<<<<<<<<<<<<<<<<<<<<<<<
int inicializar(int **final)
{
  final[1][1]=1;
}


int inicializar_calles_posic (int calles [8],int posiciones[2500])
{
  int i;
  for (i=1;i<=8;i++)
  {
    calles[i]=0;
  }

   for (i=1;i<=2500;i++)
  {
    posiciones[i]=0;
  }
}
```

```
int buscar_calles (int cont,int tramo_actual,int tramos,
                int calles[8],int precedencias[MAX_ELEMENT][MAX_ELEMENT])
{
    int i, tramo;
    cont=tramo=0;
    for (i=1;i<=tramos;i++)
    {
        cont=cont+1;
        if ((precedencias[tramo_actual][i]==1)&&
            (cont!=tramo_actual))   //si tramo actual = calle tratada,la ignora
        {
          tramo=tramo+1;
          calles[tramo]=cont;
        }
    }
}


int buscar_tramos (int tramo_actual,int tramos,int ultimo,
                int posiciones[2500],int **final)
{
   int i,j,k;
   k=0;
   for (i=1;i<=ultimo;i++)
   {
      for (j=1;j<=400;j++)
      {
        if ((final[i][j]==tramo_actual)&&(final[i][j+1]==0))
        {
          k=k+1;
          posiciones[k]=i;
        }
      }
   }
}


int numero_filas (int numero,int tramo_actual,int calles[8],
                int posiciones[],int **final)
{
   int i, cont, flag, flag2,devolver;
   cont=flag=flag2=0;
   if (numero==1)   //NUMERO DE CALLES
   {
     for (i=1;i<=8;i++)
     {
       if (calles[i]==0)
       {
         devolver=cont;
         flag=1;
```

```
        }
        if (flag==0)
        {
          cont=cont+1;
        }
      }
    }
    else if (numero==2)   // NUMERO POSICIONES
    {
      cont=flag=0;
      for (i=1;i<=2500;i++)
      {
        if (posiciones[i]==0)
        {
          devolver=cont;
          flag=1;
        }
        if (flag==0)
        {
          cont=cont+1;
        }
      }
    }
    else if (numero==3)   //NUMERO FILAS MATRIZ FINAL
    {
      cont=flag=0;
      for (i=1;i<=6000;i++)
      {
        if (final[i][1]==0)
        {
          devolver=cont;
          flag=1;
        }
        if (flag==0)
        {
          cont=cont+1;
        }
      }
    }
    return devolver;
  }
```

```
int actualiza_final(int tramos,int ultimo,int num_calles,int num_posic,
                    int tramo_actual,int calles[8],int posiciones[2500],
                    float km[MAX_ELEMENT],int **final,int seleccion[150])
{
  int i,j,k,l,flag,flag2,cont,borr,seguir;
  int borrar[1500];
  flag=flag2=cont=borr=seguir=0;


  for (i=1;i<=1500;i++)
  {
    borrar[i]=0;
  }
  for (i=1;i<=num_calles;i++)
  {
    flag2=0;
    if ((i>=2)&&(num_posic>=1))
    {
      ultimo=ultimo+1;
    }
    for (j=1;j<=num_posic;j++)
    {
       flag=0;
       cont=0;
       if ((i>=2)&&(j>=2))
       {
         ultimo=ultimo+1;
         flag2=0;
       }
       for (k=1;k<=400;k++)
       {
          if((final[posiciones[j]][k]==calles[i])&&
            (calles[i]!=42))
          {
            cont=cont+1;
          }
          if ((i==1)&&(flag==0))
          {
             if (final[posiciones[j]][k]==0)
             {
                final[posiciones[j]][k]=calles[i];
                if (calles[i]==1)
                {
                  seguir=recorre_todas(tramos,posiciones[j],
                         final,seleccion);
                  if (seguir==1)
                  {
                    borr=borr+1;
                    borrar[borr]=posiciones[j];
                  }
```

```
                    else
                    {
                      imprimir_result(tramos,posiciones[j],km,final);
                      borr=borr+1;
                      borrar[borr]=posiciones[j];
                    }
                  }
                  flag=1;


/*Nº de pasos*/       if (cont>=2)
                  {
                    final[posiciones[j]][k+1]=999;
                    borr=borr+1;
                    borrar[borr]=posiciones[j];
                  }
                }
              }
              else if ((i>1)&&(flag2==0))
              {
                final[ultimo][k]=final[posiciones[j]][k];
                if ((final[ultimo][k]==0)&&(flag2==0))
                {
                  if (final[ultimo][k-1]==999)
                  {
                    final[ultimo][k-2]=calles[i];
/*Nº de pasos*/       if (cont>=2)
                    {
                      final[ultimo][k-1]=0;
                      if (calles[i]==1)
                      {
                        seguir = recorre_todas(tramos,ultimo,
                                 final,seleccion);
                        if (seguir==1)
                        {
                          borr=borr+1;
                          borrar[borr]=ultimo;
                        }
                        else{
                          imprimir_result(tramos,posiciones[j],km,final);
                          borr=borr+1;
                          borrar[borr]=posiciones[j];
                        }
                      }
                    }
                    else{
                      borr=borr+1;
                      borrar[borr]=ultimo;
                    }
                  }
```

```
                else
                {
                  final[ultimo][k-1]=calles[i];
/*Nº de pasos*/         if (cont>=2)
                  {
                    final[ultimo][k]=999;
                    borr=borr+1;
                    borrar[borr]=ultimo;
                  }
                  else
                  {
                    if (calles[i]==1)
                    {  // comprueba si ya ha pasado por todas los tramos
                      seguir = recorre_todas(tramos,ultimo,
                            final,seleccion);
                      if (seguir==1)
                      {
                        borr=borr+1;
                        borrar[borr]=ultimo;
                      }
                      else
                      {
                        imprimir_result(tramos,posiciones[j],km,final);
                        borr=borr+1;
                        borrar[borr]=posiciones[j];
                      }
                    }
                  }
                }
                flag2=1;
              }
            }
          }
        }
      }
// Borra de la matriz final aquellas filas "sin solución"
    for (i=borr;i>=1;i--)
    {
      for (j=1;j<=400;j++)
      {
        if (borrar[i]!=ultimo)
        {
          final[borrar[i]][j]=final[ultimo][j];
        }
        final[ultimo][j]=0;
      }
      ultimo=ultimo-1;
    }
```

```
  // Al tener problemas de rendimiento se ha hecho una selección
  // al azar, es decir si el número de filas de la matriz final
  // supera 4500, borra el 30%  se sus registros
     if (ultimo>4500)
     {
        ultimo=borrar_registros(ultimo,final);
     }
     return ultimo;
  }


  int recorre_todas(int tramos,int i,int **final,int seleccion[150])
  {
    int seguir,j,cont,k;
    int vec[tramos];
    seguir=0;
    cont=1;
/*
    for (j=1;j<=tramos;j++)
    {
       vec[j]=cont;
       cont=cont+1;
    }*/ int aux;
    for (j=1;j<=tramos;j++)
    {
        aux=seleccion[j];
        vec[j]=aux;
    }

    for (j=1;j<=400;j++)
    {
        if (final[i][j]!=0)
        {
           for (k=1;k<=tramos;k++)
           {
              if (final[i][j]==vec[k])
              {
                 vec[k]=0;
              }
           }
        }
    }


    for (j=1;j<=tramos;j++)
    {
       if (vec[j]!=0)
       {
          seguir=1;
       }
```

```
  }
  return seguir;
}


int imprimir_result(int tramos,int fila,float km[MAX_ELEMENT],int **final)
{
   int l,k;
   float total;
   total=0;

// Imprime las soluciones por pantalla
   for (l=1;l<=400;l++)
   {
      if (final[fila][l]!=0)
      {
         printf("%d ",final[fila][l]);
      }

      // Calcula los km's recorridos
      for (k=1;k<=tramos;k++)
      {
         if (final[fila][l]==k)
         {
            total=total+km[k];
         }
      }
   }
   printf("\nKM= ");
   printf("%0.3f ",total);
   printf("\n\n");
}


int borrar_registros(int ultimo,int **final)
{
  float numero;
  int r,n,j,entero,var;
  n=9;
  r=rand()%n + 1;
  if (r==0)
  {
     r=rand()%n + 3;
  }
  numero=(ultimo*3)/10;
  entero=fabs(numero);
  var=ultimo;

  while (entero>=0)
  {
     var=var-r;
```

```
    if (var<0)
    {
      var = ultimo - r;
    }
    for (j=1;j<=400;j++)
    {
      final[var][j]=final[ultimo][j];
      final[ultimo][j]=0;
    }
    ultimo=ultimo-1;
    entero=entero-1;
  }
  return ultimo;
}


int tramos_recorridos(int ultimo,int final_bucle,int **final)
{
  int i,j,flag,aux;
  flag=aux=0;

  for (i=1;i<=ultimo;i++)
  {
    aux=0;
    for(j=1;j<=400;j++)
    {
      if ((final[i][j]==0)&&(aux==0))
      {
        if ((final[i][j-1]!=1)&&
            (final[i][j-1]!=999))
        {
          flag=1;
          aux=1;
        }
        else
        {
          aux=1;
        }
      }
    }
  }
  if (flag==0)
  {
    final_bucle=1;
  }

   return final_bucle;
}
```

```
int actualizar(int tramo_actual,int num_calles,
            int calles[8],int siguiente[150])
{
  int i,j,flag;
  flag=0;

  if (siguiente[1]!=0)
  {
    tramo_actual=siguiente[1];
    for (i=1;i<=150;i++)
    {
      siguiente[i]=siguiente[i+1];
      if (siguiente[i]!=0)
      {
        for (j=1;j<=8;j++)
        {
          if (siguiente[i]==calles[j])
          {
            calles[j]=0;
          }
        }
      }
    }
    for (i=1;i<=150;i++)
    {
      if ((siguiente[i]==0)&&(flag==0))
      {
        for (j=1;j<=8;j++)
        {
          if ((tramo_actual!=calles[j])&&
              (calles[j]!=1)&&(calles[j]!=0))
          {
            siguiente[i]=calles[j];
            i=i+1;
          }
        }
        flag=1;
      }
    }
  }
  else
  {
    if (calles[1]!=1)
    {
      tramo_actual=calles[1];
      if (num_calles>1)
      {
        for (i=1;i<=7;i++)
        {
          if (calles[i+1]!=1)
```

```
            {
              siguiente[i]=calles[i+1];
            }
          }
        }
      }
      else
      {
        tramo_actual=calles[2];
        if (num_calles>2)
         {
          for (i=1;i<=6;i++)
          {
            if (calles[i+2]!=1)
            {
              siguiente[i]=calles[i+2];
            }
          }
        }
      }
    }
    return tramo_actual;
}
```