```
/*PROGRAMA QUE DA SOLUCION AL MODELO DE SCHEDULING Qm || SUMA Cj. EN EL CUAL
SE TIENEN m MAQUINAS NO RELACIONADAS CON n TAREAS A REALIZAR EN ALGUNA DE
ESTAS, Y EL OBJETIVO ES EL DE MINIMIZAR EL TIEMPO TOTAL DE TERMINACION DE
LAS TAREAS.

ELABORADO POR : Marco Antonio Cruz Chavez.
PROYECTO FINAL: OPTIMIZACION COMBINATORIA.
FECHA         : DICIEMBRE 07 DE 1999.
*/

#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>

#define M_MAX 6    //Número de máquinas
#define N_MAX 50  //Número de tareas

#define REN_MAX M_MAX*N_MAX+N_MAX+1
#define COL_MAX M_MAX*N_MAX*N_MAX+REN_MAX+1

char *msg[20]={"Solución al modelo de scheduling: Pm | | suma Cj\n\n",
                "Escribe el número de máquinas en tu sistema (máximo 4): ",
                "Escribe el número de tareas en tu sistema (máximo 30): ",
               "\nTiempo de procesamiento para la tarea[%d] en máquina[%d] = "};

float tabla[REN_MAX][COL_MAX];

int columna_pivote(int,int *);
int renglon_pivote(int,int,int);
void func_obj_transf_art(int,int);
void conv_num_pivote(int,int,int);
float conv_col_pivote(int,int,int,int);
void var_sistema(int,int,int,char (*)[12]);
void imprime_var(int,int,int *,char (*)[12]);
void datos_modelo(float (*)[N_MAX],int *,int *);
float solucion_simplex(int,int,int *,char (*)[12]);
void imprime_tabla(int,int,char (*)[12],int *,char *);
void crea_tabla_simplex(float (*)[N_MAX],int,int,int * ,int *,int *);

void main()
{
 int ren,col,m=5,n=50,vbas[350];
 static float
datos[M_MAX][N_MAX]={{1,13,2,17,3,18,19,25,26,30,7,15,8,22,9,24,33,34,28,38,4,14,5,21,6,20,31,32,27,
37,7,15,8,22,9,24,33,34,28,38,21,6,20,31,32,27,37,7,15,8,},

{4,14,5,21,6,20,31,32,27,37,30,7,15,8,22,9,24,33,34,28,38,4,14,5,21,6,20,31,32,27,3,18,19,25,26,30,7,15,8,2
2,9,24,33,34,28,38,4,14,5,21,},

{7,15,8,22,9,24,33,34,28,38,21,6,20,31,32,27,37,7,15,8,22,9,24,33,34,28,38,21,6,20,19,25,26,30,7,15,8,22,9,
24,33,34,28,38,4,14,5,21,6,20,},

{1,13,2,17,3,18,19,25,27,37,30,7,15,8,22,9,6,20,31,32,27,37,7,15,8,22,37,7,15,8,22,9,24,33,34,28,38,4,14,5,2
1,6,20,31,38,4,14,5,21,6,},};
 char Tvar[COL_MAX][12];
```

```
 //datos_modelo(datos,&m,&n);
 crea_tabla_simplex(datos,m,n,&ren,&col,vbas);
 var_sistema(m,n,col,Tvar);
 //imprime_tabla(ren,col,Tvar,vbas,"\n\nTABLA INICIAL.\n");
 printf("\n\nTiempo total de terminaci¢n: %.2f",solucion_simplex(m,n,vbas,Tvar));
 imprime_var(ren,col,vbas,Tvar);
 //imprime_tabla(ren,col,Tvar,vbas,"\n\nTABLA FINAL.\n");
}

void imprime_var(int ren,int col,int *vbas,char (*var)[12])
{
int y;
 printf("\n\nTareas y posici¢n asignada en las maquinas.\n\n");
  for(y=1;y<=ren;y++)
   if(tabla[y][col]==1 && var[vbas[y]][1]=='X')printf(var[vbas[y]]);
}

void datos_modelo(float (*dat)[N_MAX],int *m, int *n)
{
 int i,j; float dato;  //Numero de tareas, m quinas y tiempos de procesamiento
   system("cls");
 puts(msg[0]);
 printf(msg[1]); scanf("%d",m);
 printf(msg[2]); scanf("%d",n);
 for(i=0;i<*m;i++)
  for(j=0;j<*n;j++){
   printf(msg[3],j+1,i+1); scanf("%lf",&dato);
   dat[i][j]=dato;
  }
}

void crea_tabla_simplex(float (*dat)[N_MAX],int m ,int n,int *ren,int *col,
                                  int *vbas)
{
 int cols,i,j,k,ri,ci,M,l=1;
 *ren=m*n+n;
 cols=m*n*n;
 *col=cols+(*ren)+1;
  for(i=cols+1,j=1,vbas[0]=0;i<*col;i++,j++)vbas[j]=i; //var basicas iniciales
 for(i=0;i<=*ren;i++)for(j=0;j<=*col;j++)tabla[i][j]=0.0;     //inicializacion
 tabla[0][0]=1.0;
 for(j=0;j<n;j++)for(i=0;i<m;i++)for(k=1;k<=n;k++)          //funcion objetivo
  tabla[0][l++]=k*dat[i][j];
 for(ri=1,ci=1,M=m*n;ri<=n;ci=M+1,M+=m*n,ri++)        //primeras restricciones
  for(j=ci;j<=M;tabla[ri][j]=1.0,j++);
 for(ci=1;ri<=*ren;ci+=1,ri++)                        //segundas restricciones
  for(j=ci;j<=cols;tabla[ri][j]=1.0,j+=m*n);
 for(i=1,j=cols+1;i<=*ren;tabla[i][j]=1.0,i++,j++);   //var artific y holguras
 for(j=cols+1;j<=cols+n;j++)tabla[0][j]=1000.0;       //var artific en func obj
 for(i=1;i<=*ren;tabla[i][*col]=1.0,i++);             //terminos independientes
 system("cls");
}

void var_sistema(int m,int n,int col,char (*var)[12])
{
 int i,j,k,x;
```

```
 char str[2];
 strcpy(var[0],"     ");
 strcat(var[0],"Z");
 for(j=0,x=1;j<n;j++)for(i=0;i<m;i++)for(k=1;k<=n;k++,x++){
   strcpy(var[x]," ");
   strcat(var[x],"X");
   itoa(i+1,str,10); strcat(var[x],str);
   itoa(k,str,10);   strcat(var[x],str);
   itoa(j+1,str,10); strcat(var[x],str);
  }
  for(i=x,j=1,k=1;i<col;i++,j++,x++){
   strcpy(var[x],"    ");
   if(j<=n){
    strcat(var[x],"A");
    itoa(j,str,10); strcat(var[x],str);
   }
   else{
    strcat(var[x],"S");
    itoa(k++,str,10); strcat(var[x],str);
   }
  }
}

void imprime_tabla(int ren,int col,char (*var)[12],int *vbas,char *msg)
{
 int i,j;
 for(j=0,puts(msg),printf("Basic");j<col;printf(var[j++]));
 for(i=0,putchar('\n');i<=ren;putchar('\n'),i++)
  for(j=0,printf(var[vbas[i]]);j<=col;printf("%5.0f",tabla[i][j++]));
}

float solucion_simplex(int m,int n,int *vbas,char (*var)[12])
{
 int ren,col,cols,cp,rp,optimo; float op;
 ren=m*n+n;
 cols=m*n*n;
 col=cols+ren+1;
 func_obj_transf_art(col,n);       //cambia a formato de eliminacion gaussiana
// imprime_tabla(ren,col,var,vbas,"\n\nARREGLO A ELIMINACION GAUSSIANA.\n");
 for(;;){
  cp=columna_pivote(col,&optimo);
  if(optimo)return(op);
  rp=renglon_pivote(ren,col,cp);
  vbas[rp]=cp;                          //var bas entrante en ren pivote
  conv_num_pivote(col,rp,cp);
  op=conv_col_pivote(ren,col,rp,cp);
 // imprime_tabla(ren,col,var,vbas,"\n\nITERACION TABLA.\n");
 }
}

float conv_col_pivote(int ren,int col,int rp,int cp)
{
 int i,j; float piv;
 for(i=0;i<=ren;i++)
  for(j=1,piv=tabla[i][cp];j<=col;j++)      //convierte columna pivote a ceros
   if(rp!=i)tabla[i][j]=-piv*tabla[rp][j]+tabla[i][j];
```

```
 return(tabla[0][col]);
}

int renglon_pivote(int ren,int col,int cp)
{
 int i,rp;
 float razonmin,v;
 for(i=1,rp=1,v=1000.0;i<=ren;i++)                    //renglon pivote
  if(tabla[i][cp]>0.0){
   razonmin=tabla[i][col]/tabla[i][cp];
   if(v > razonmin){
    v=razonmin;
    rp=i;
   }
  }
  return(rp);
}

int columna_pivote(int col,int *optimo)
{
 int j,cp;
 float v;
 for(j=2,*optimo=0,cp=1,v=tabla[0][1];j<col;j++)          //columna pivote
  if(v > tabla[0][j]){
   v=tabla[0][j];
   cp=j;
  }
  if(v>=0.0)*optimo=1;
 return(cp);
}

void func_obj_transf_art(int col,int n)
{
 int i,j;
 for(i=1;i<=n;i++)                  //transforma para eliminacion gaussiana
  for(j=1;j<=col;j++)
   tabla[0][j]+=-1000.0*tabla[i][j];
}

void conv_num_pivote(int col,int rp,int cp)
{
int j;                              //se transforma a 1 el numero pivote
for(j=1;j<=col;tabla[rp][j]/=tabla[rp][cp],j++);
```