

Proyecto Fin de Carrera
Ingeniero Industrial

**Programación de la secuencia de fabricación en una
máquina, con tiempos de preparación variables,
mediante la aplicación de Algoritmos Genéticos.**

**Anexo C: Manual de empleo del programa “Secuencia de
Fabricación”**

Anexo D: Código del programa

Autor: Ignacio Fernández-Baños Marín

Director: Manuel Mateo Doll

Convocatoria: Diciembre 2003 (Plan 94)



Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona



ANEXO C. MANUAL DE EMPLEO DEL PROGRAMA “SECUENCIA DE FABRICACIÓN” _____ **221**

C.1. Arranque del problema	221
C.2. Opción <i>Nuevo</i>	222
C.3. Opción <i>AbrirEjemplar</i>	223
C.4. Opción <i>AbrirColección</i>	224
C.5. Introducción de los datos	225
C.5.1. <i>Leer Nuevo</i>	225
C.5.2. <i>Leer Ejemplar</i>	229
C.5.3. <i>Leer Colección</i>	232
C.6. Opciones	233
C.7. Ejecutar	235

ANEXO D. CÓDIGO DEL PROGRAMA _____ **239**

D.1. Formulario Entrada	239
D.2. Formulario AbrirNuevo	242
D.3. Formulario AbrirEjemplar	245
D.4. Formulario AbrirColección	247
D.5. Formulario Opciones	251
D.6. Formulario ResultadosEjemplar	252
D.7. Formulario ResultadosColección	256
D.8. Formulario RetrasosPedidos	259
D.9. Formulario ÓrdenesProducción	262
D.10. Module1	263
D.11. Module2	265
D.12. Module3	284

Anexo C: Manual de empleo del programa

“Secuencia de Fabricación”

Para la programación informática del algoritmo se ha utilizado el lenguaje *Basic* y se ha compilado mediante el compilador *Visual Basic 6.0*. El programa recibe el nombre de “Secuencia de Fabricación”.

C.1. Arranque del programa

Al arrancar el programa, aparece una ventana, como la que se puede observar en la Figura C.1, que contiene dos menús: *Archivo* y *Salir*. Si se hace clic en el primero de ellos, se despliega una lista que visualiza tres sub-menús: *Nuevo*, *AbrirEjemplar* y *AbrirColeccion*. Las dos primeras órdenes sirven para examinar un solo ejemplar mientras que la última se utiliza para analizar un conjunto finito de ejemplares. Si se hace clic en el menú *Salir* se abandona la aplicación.

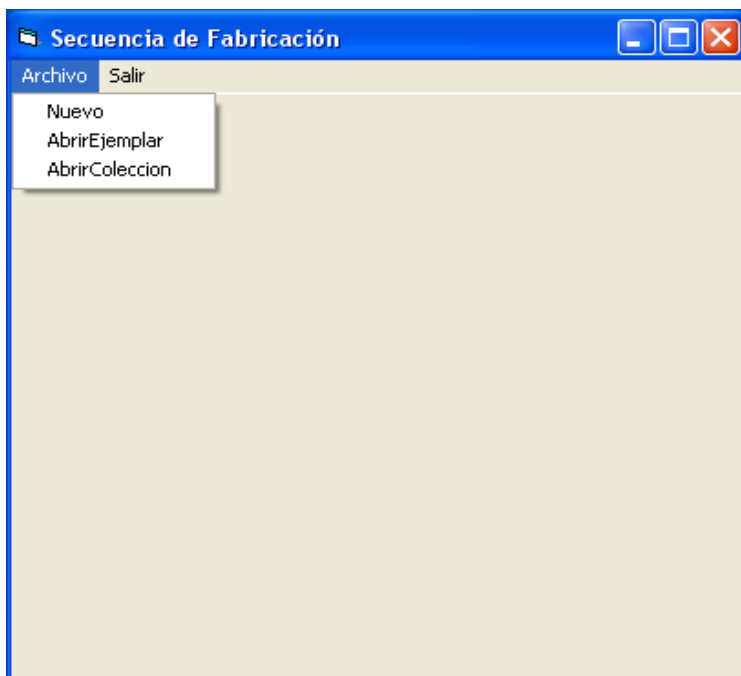


Figura C.1. Inicio de *Secuencia de Fabricación*.

C.2. Opción *Nuevo*

Es especialmente útil para un usuario que no tiene almacenados los datos requeridos, ya que el programa va guiando al usuario en la introducción de los datos.

Si se opta por esta opción, aparece una ventana con 4 botones que tienen asociados una orden: *Leer Nuevo*, *Opciones*, *Ejecutar* y *Salir*, tal como se puede observar en la Figura C.2:

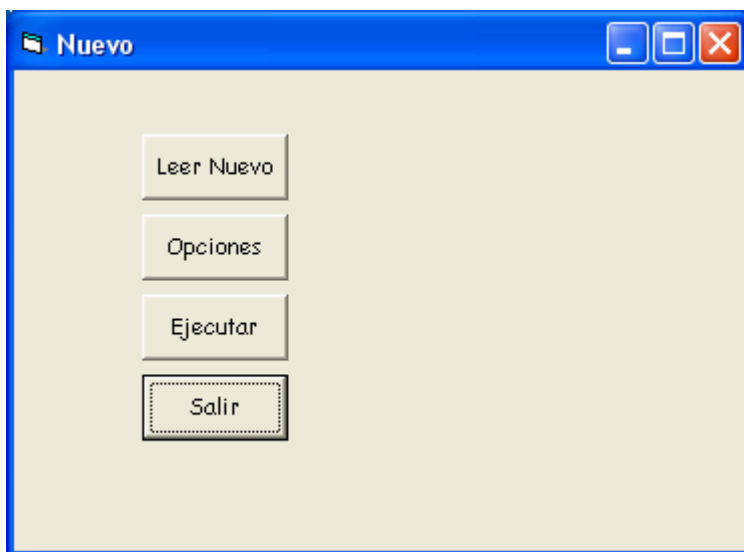


Figura C.2. Ventana de la Opción *Nuevo*.

- *Leer Nuevo* : sirve para introducir los datos del problema.
- *Opciones* : se emplea para fijar los operadores variables del algoritmo.
- *Ejecutar* : lleva a cabo el algoritmo llegando a una solución.
- *Salir* : devuelve el programa a la ventana anterior.

C.3. Opción *AbrirEjemplar*

Es la opción idónea si se dispone de los datos de un ejemplar guardados en un archivo, cuyo formato y estructura se explicará posteriormente.

Si se opta por este camino, aparece una nueva ventana (Figura C.3), con cuatro posibilidades: *Leer Ejemplar*, *Opciones*, *Ejecutar* y *Salir*.



Figura B.3. Ventana de la Opción *AbrirEjemplar*.

- *Leer Ejemplar* : sirve para introducir los datos. Éstos deben estar ya guardados en un archivo, de extensión *.txt* o *.doc*.
- *Opciones* : fija los parámetros variables del algoritmo.
- *Ejecutar* : aplica el algoritmo.
- *Salir* : devuelve el programa a la ventana inicial.

C.4. Opción *AbrirColección*

Es la opción a aplicar si el usuario dispone de un conjunto de ejemplares correctamente almacenados en un archivo, cuyo formato también será explicado con posterioridad.

Al hacer clic en esta opción, aparece una ventana (Figura C.4) con las siguientes opciones: *Leer Colección*, *Opciones*, *Ejecutar* y *Salir*.



Figura C.4. Ventana de la Opción *AbrirColección*.

- *Leer Colección* : se utiliza para introducir la colección de ejemplares, que deben estar ya guardados en un archivo, de extensión *.txt* o *.doc*.
- *Opciones* : establece los operadores variables del algoritmo.
- *Ejecutar* : aplica el algoritmo.
- *Salir* : devuelve el programa a la ventana inicial.

Lo primero que se debe hacer cuando se está ante cualquiera de estas tres opciones es introducir los datos, para después seleccionar las operadores variables del algoritmo que están contenidos en *Opciones*. Estas dos órdenes se deben realizar antes de ejecutar el algoritmo y, aunque es recomendable hacerlo en el orden designado, se puede obrar en orden inverso. No es estrictamente necesario pasar por *Opciones* antes de aplicar la orden *Ejecutar*; si no se entra en dicha opción, las variables de esta opción serán las designadas por defecto. Sin embargo, sí es necesario insertar los datos antes de *Ejecutar*, ya que resulta imposible ejecutar el algoritmo sin los datos del ejemplar.

C.5. Introducción de los datos

La introducción de los datos varía en función de la opción que se ha escogido en el inicio del programa.

C.5.1. Leer nuevo

Es la opción a través de la cual se introducen los datos si se entra mediante la opción *Nuevo*. Los datos del ejemplar se introducen mediante cajas de diálogo explicativas que guían al usuario acerca de los datos que deben ser introducidos. Para pasar de una caja de diálogo a otra se debe hacer clic en el botón *Aceptar* o apretar la tecla *Intro* del teclado. En el caso de que se haga clic en el botón *Cancelar* de una de las cajas de diálogo, se apriete la tecla *Escape* del teclado o se introduzca un dato absurdo, como puede ser una letra o un signo alfabético en vez de un número, aparece una caja de diálogo (Figura C.5) que devuelve el programa a la ventana *Nuevo*.

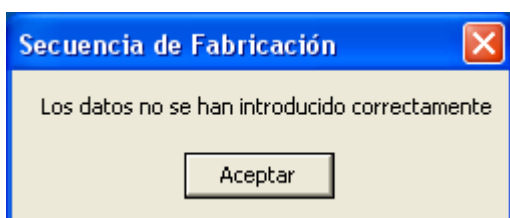


Figura C.5. Aviso de error en la introducción de los datos.

La primera caja de diálogo (Figura C.6) preguntará al usuario sobre el número de piezas n , del ejemplar:

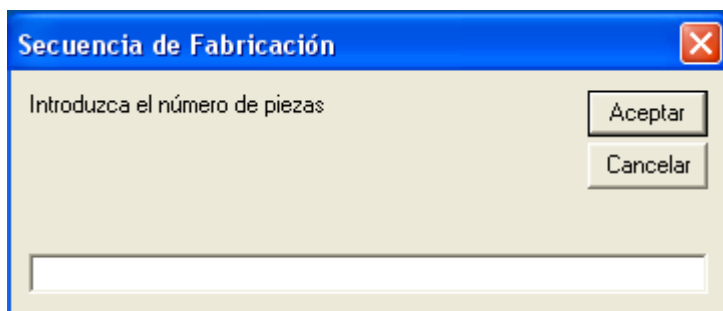


Figura C.6. Caja de diálogo para la introducción del número de piezas n .

Posteriormente, el programa requerirá la introducción de los tiempos de preparación entre las piezas, con cajas de diálogo como en la Figura C.7. Se supone que el tiempo de preparación entre piezas iguales es nulo. De esta manera, el número de cajas de diálogo que aparecen para completar esta información es de $(n-1) \times n$.

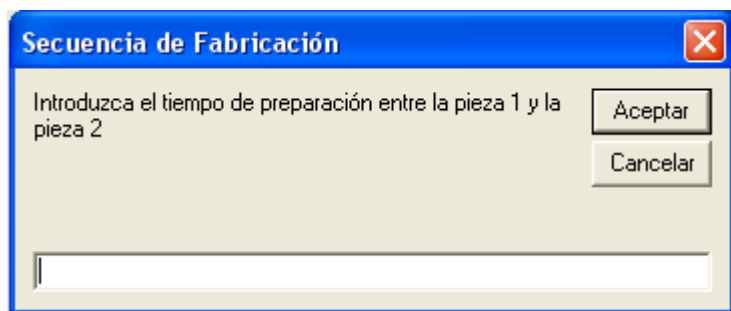


Figura C.7. Caja de diálogo para la introducción de los tiempos de preparación.

Los tiempos de preparación obviamente no pueden ser negativos. Si, por error, se introdujera un número negativo, aparecería una caja de diálogo como la de la Figura C.8 informando del error.

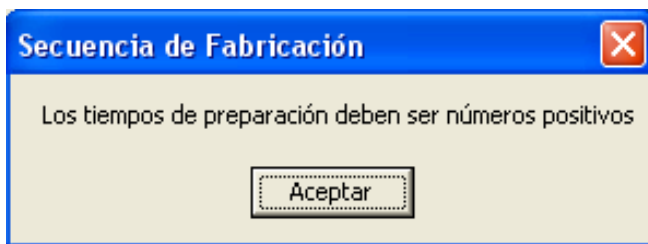


Figura C.8. Aviso de error en la introducción de los tiempos de preparación.

Una vez que se acepta el error haciendo clic en *Aceptar*, se vuelve a la ventana anterior donde se produjo el error para que el usuario pueda volver introducir el dato en cuestión correctamente.

Seguidamente, se deben introducir los tiempos unitarios de ejecución de las n piezas mediante n cajas de diálogo, como la de la Figura C.9. Si dichos tiempos contienen fracciones decimales es importante separar los enteros de los decimales con comas y no con puntos, ya que el programa no sabe interpretar estos signos alfabéticos.

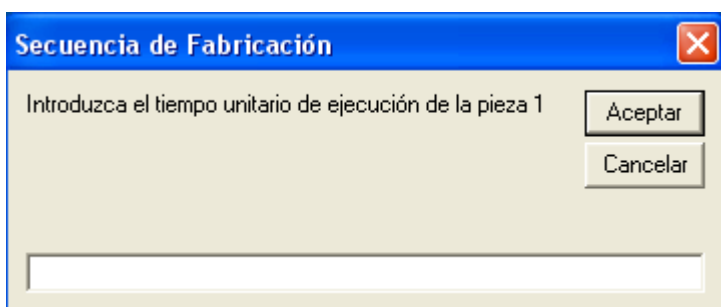


Figura C.9. Caja de diálogo para la introducción de los tiempos unitarios de fabricación.

Obviamente, los tiempos unitarios de ejecución de las piezas no pueden ser ni nulos ni negativos. Si, por error, se introduce un número negativo se enseña una caja de diálogo explicativa semejante a la Figura C.8, informando de este error.

A continuación, se pide el número de pedidos de que consta el ejemplar (Figura C.10):

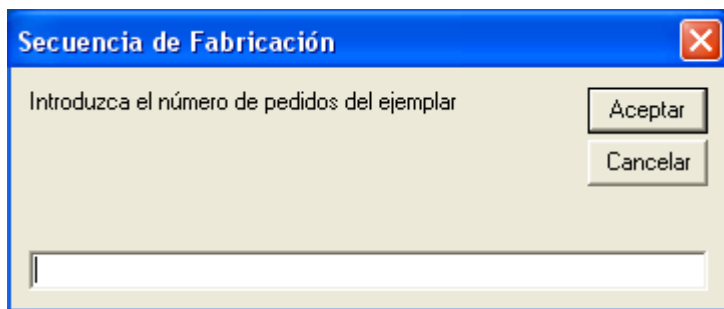


Figura C.10. Caja de diálogo para la introducción del número de pedidos del ejemplar.

En la siguiente caja de diálogo (Figura C.11) se requiere al usuario el estado inicial de la máquina, es decir, cuál ha sido la operación de la última pieza que ha pasado por la máquina antes de empezar la ejecución de las operaciones del ejemplar en marcha.

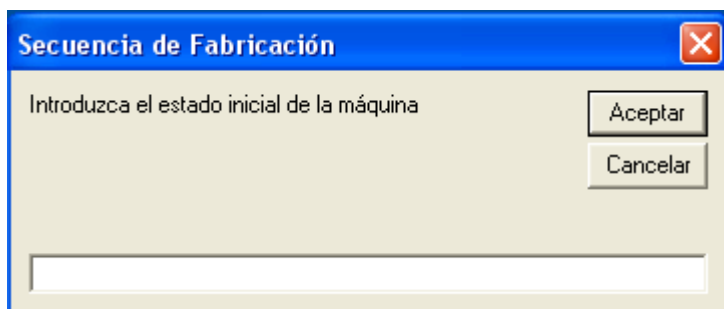


Figura C.11. Caja de diálogo para la introducción del estado inicial de la máquina.

A continuación, para cada una de los pedidos que forman el ejemplar se pide la fecha de vencimiento y el número de piezas que la componen, para justamente después pasar a pedir el código de las piezas a tratar y su cantidad. Un par de ejemplos de estas cajas de diálogo pueden ser las Figuras C.12 y C.13.

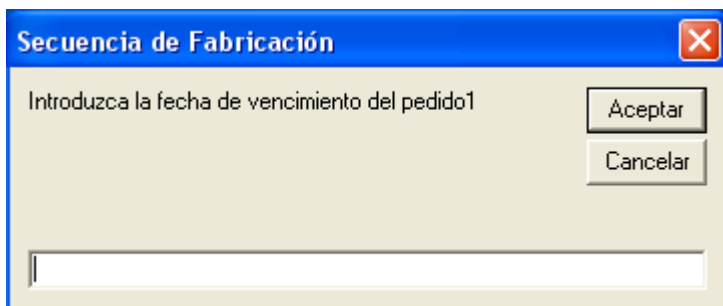


Figura C.12. Ejemplo de caja de diálogo para la introducción de las fechas de vencimiento.

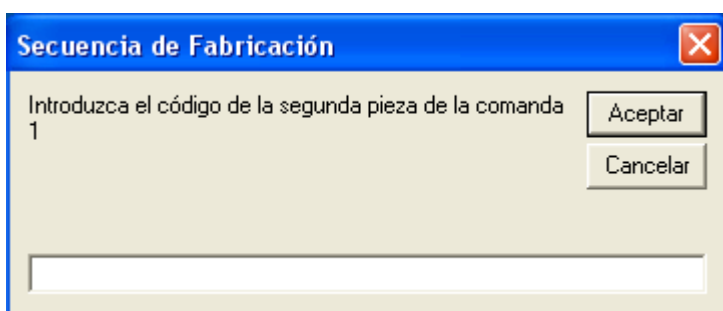


Figura C.13. Ejemplo de caja de diálogo para la introducción de los códigos de las piezas.

Una vez que se ha insertado la última cantidad de piezas del último pedido, el programa vuelve automáticamente a la ventana *Nuevo*.

C.5.2. Leer Ejemplar

Si se opta por el sub-menú *AbrirEjemplar*, los datos se introducen mediante esta opción. A diferencia de la anterior, los datos deben estar ya guardados en un archivo, de extensión .txt o .doc. Cuando se hace clic en *Leer Ejemplar* aparece una ventana (Figura C.14), a través de la cual se puede acceder al archivo requerido.

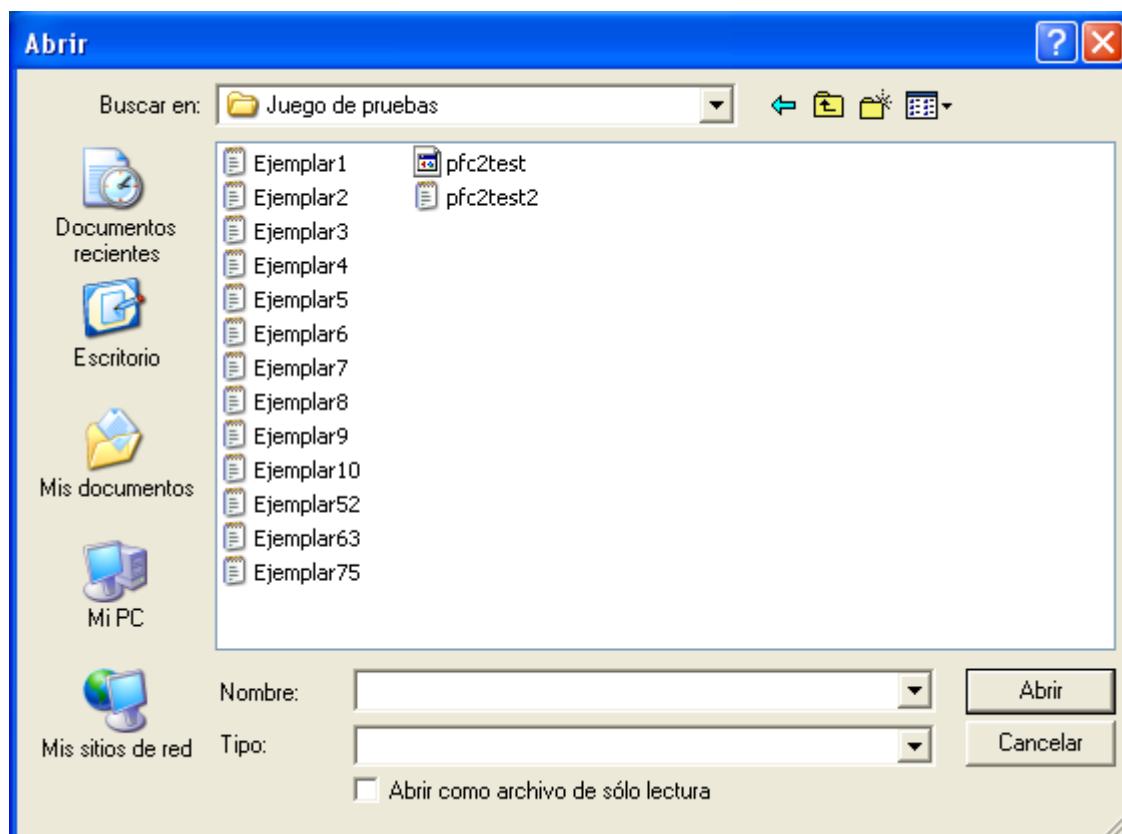


Figura C.14. Ventana de acceso a un archivo.

El formato de dicho archivo está ya establecido y debe ser el siguiente, en orden descendente:

- Primero, debe aparecer el número de piezas n .
- A continuación, se deben mostrar los tiempos de preparación entre piezas en forma de matriz cuadrada $n \times n$ donde, en principio, los tiempos que pertenecen a la diagonal, correspondientes a tiempos de preparación entre piezas iguales, son nulos.
- Seguidamente, se deben mostrar en una misma línea los n tiempos unitarios de ejecución de cada una de las piezas.
- En la siguiente línea se indica el número de pedidos de que consta el ejemplar y el estado inicial de la máquina.

- Por último, se debe señalar, separadamente para cada pedido, el número de pedidos, su fecha de vencimiento y el número de tipos de piezas que la integran. A continuación, se indica sucesivamente los tipos de piezas y la cantidad de éstas a fabricar.

Obviamente, es necesario que coincidan tanto el número de pedidos como el número de tipos de piezas a ejecutar para cada pedido que se indican en un principio, con los que posteriormente se introducen.

En la Figura C.15 se puede observar un ejemplo de cómo debe ser un archivo. En él, se pueden percibir todas las partes anteriormente mencionadas: el ejemplar dispone de 10 piezas, 10 x 10 tiempos de preparación y 10 tiempos unitarios de ejecución; existen 7 pedidos y el estado inicial de la máquina corresponde a la pieza número 4. En cada uno de los siete pedidos, se indica primero su fecha de vencimiento y el número de piezas que la integran, y posteriormente los códigos de dichos tipos de piezas seguido de las cantidades a ejecutar.

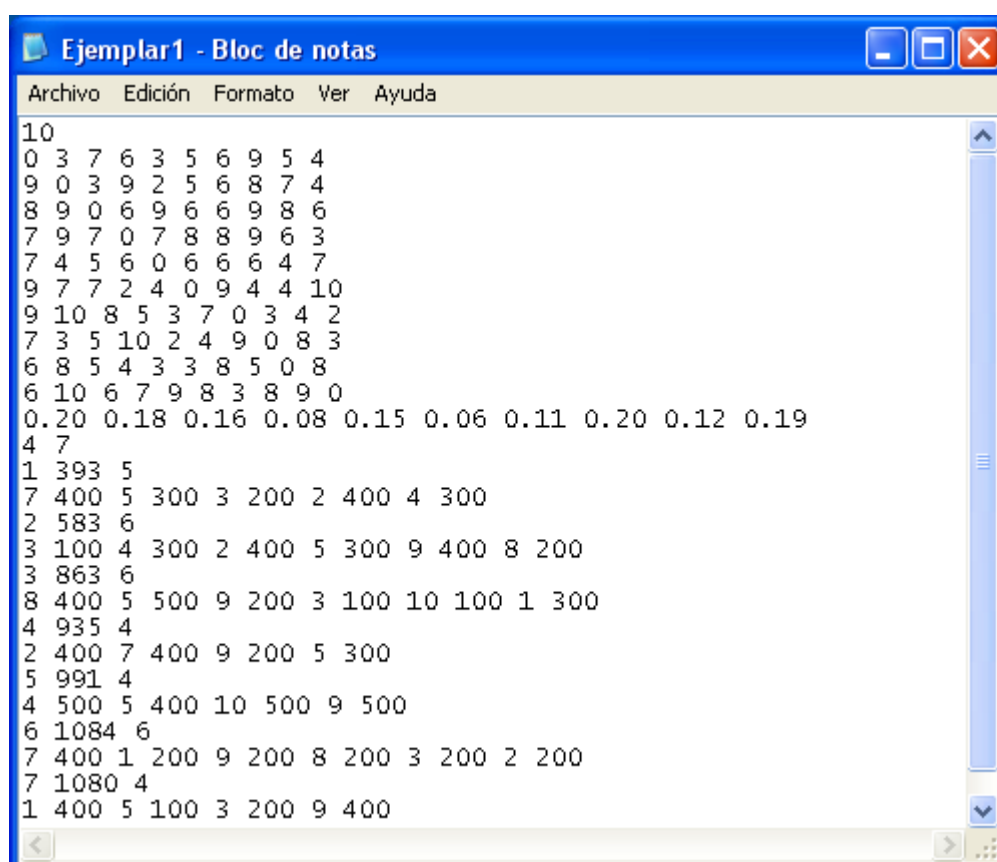
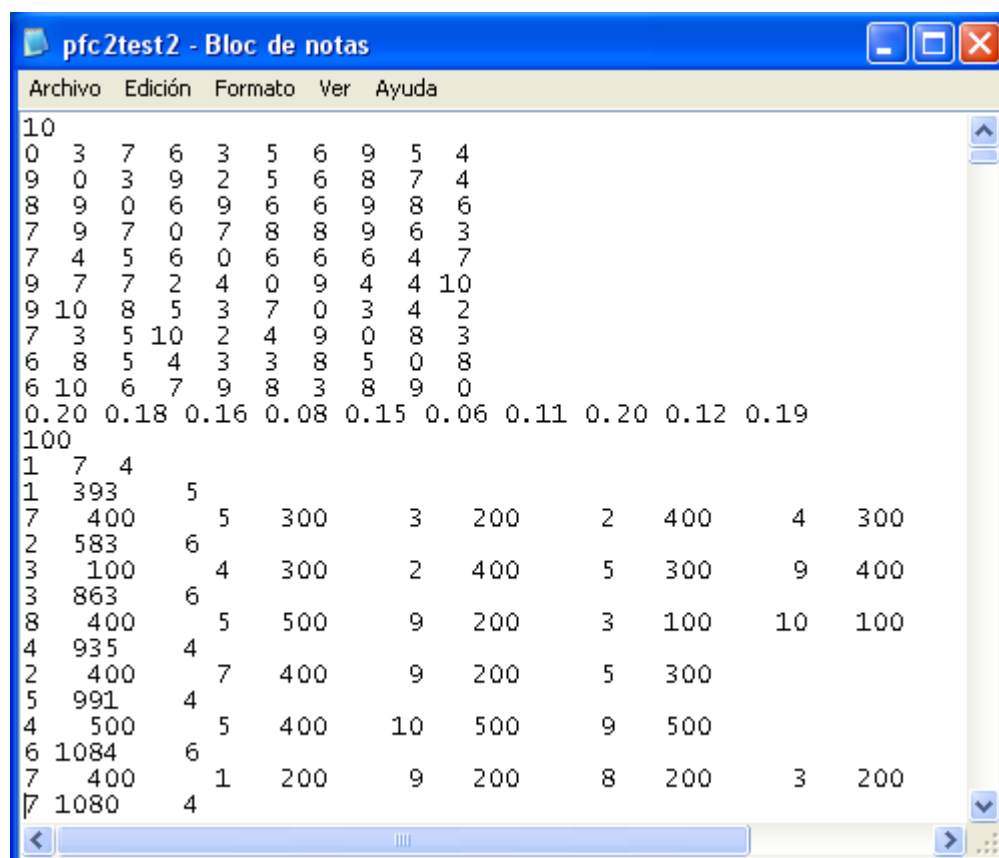


Figura C.15. Ejemplo de la estructura de un archivo que contiene un ejemplar.

C.5.3. Leer Colección

Si se pretende analizar un conjunto de ejemplares al mismo tiempo, se debe aplicar la orden *AbrirColeccion*. La introducción de los datos se realiza a través de *Leer Colección*. Cuando se hace clic en *Leer Colección* aparece una ventana (Figura C.14), a través de la cual se puede acceder al archivo requerido.

Como sucedía en *Leer Ejemplar*, los datos deben estar guardados en un archivo de extensión .doc o .txt. La estructura de este archivo es similar al anteriormente explicado, pero este debe contener la siguiente información adicional: inmediatamente después de los tiempos unitarios de ejecución de las piezas, debe señalizarse el número de ejemplares que contiene la colección y se deben numerar todos los ejemplares en orden ascendente partiendo de 1. En la Figura C.16 se puede observar la configuración que debe tener un archivo de este tipo. En este caso la colección consta de 100 ejemplares.



The screenshot shows a Notepad window titled "pfc2test2 - Bloc de notas". The text content is as follows:

```

Archivo  Edición  Formato  Ver  Ayuda
10
0 3 7 6 3 5 6 9 5 4
9 0 3 9 2 5 6 8 7 4
8 9 0 6 9 6 6 9 8 6
7 9 7 0 7 8 8 9 6 3
7 4 5 6 0 6 6 6 4 7
9 7 7 2 4 0 9 4 4 10
9 10 8 5 3 7 0 3 4 2
7 3 5 10 2 4 9 0 8 3
6 8 5 4 3 3 8 5 0 8
6 10 6 7 9 8 3 8 9 0
0.20 0.18 0.16 0.08 0.15 0.06 0.11 0.20 0.12 0.19
100
1 7 4
1 393 5
7 400 5 300 3 200 2 400 4 300
2 583 6
3 100 4 300 2 400 5 300 9 400
3 863 6
8 400 5 500 9 200 3 100 10 100
4 935 4
2 400 7 400 9 200 5 300
5 991 4
4 500 5 400 10 500 9 500
6 1084 6
7 400 1 200 9 200 8 200 3 200
7 1080 4

```

Figura C.16. Ejemplo de la estructura de un archivo que contiene 100 ejemplares.

C.6. Opciones

Cuando se han introducido los datos, el siguiente paso consiste en fijar los operadores variables del algoritmo entrando en la opción *Opciones*.

La opción *Opciones* lleva al usuario a una ventana (Figura C.17), donde deberá seleccionar los operadores variables que el algoritmo deja a su elección:

Opciones

Elementos de la población inicial

- ☐ Triple del número de piezas
- ☒ Suma del número de lotes de todos los pedidos
- ☐ Otra cantidad:

Número de regeneraciones

- ☐ Cuatro veces el número de piezas
- ☒ Triple de la suma del número de lotes de todos los pedidos
- ☐ Otro número:

Probabilidad cruce:

Probabilidad mutación:

Tipos de cruce

- ☒ 1 punto de cruce
- ☒ PMAX
- ☐ OX

Tipos de mutación

- ☐ Blanda
- ☒ Pareja
- ☐ Intercambio

Salir **Aplicar**

Figura C.17. Ventana de la Opción *Opciones*.

Como se puede observar existe un total de seis operadores que son variables:

- Los elementos de la población inicial.
- El número de regeneraciones
- La probabilidad de cruce.
- La probabilidad de mutación.
- Los tipos de cruce.
- Los tipos de mutación.

Para los dos primeros operadores se debe elegir una de las tres opciones que están disponibles. Si se opta por la última opción, se debe introducir el número deseado en la caja de texto correspondiente. Para la probabilidad de cruce, se debe elegir entre un valor comprendido entre 80 y el 100% y para la probabilidad de mutación, entre el 0 y el 40%. Si no se respetan estos límites, el programa avisa al usuario mediante una caja de diálogo (Figura C.18).

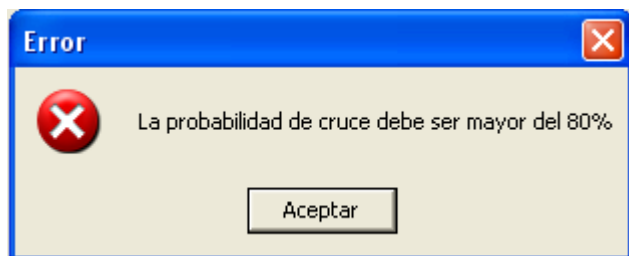


Figura C.18. Aviso de error en la introducción del valor de la probabilidad de cruce.

Para los tipos de cruce y los de mutación, se pueden escoger una, dos o las tres opciones. Eso sí, se debe escoger al menos una; en caso contrario, aparecerá una caja de diálogo semejante a la Figura C.18 avisando al usuario de este hecho.

Una vez que todos los valores de los operadores han sido seleccionados correctamente a gusto del usuario, éstos se guardan si se pulsa el botón *Aplicar*. Si, por el contrario, se selecciona el botón *Salir*, los datos no se guardan y permanecen los valores previos

seleccionados antes de entrar en la ventana *Opciones*. Tanto al seleccionar una opción como la otra, el programa vuelve a la ventana anterior, listo para la ejecución del algoritmo.

Los valores de estos operadores que están designados por defecto en el programa y, que si no son modificados por el usuario, el algoritmo los utiliza durante su ejecución están reflejados en la Figura C.17 y son los siguientes:

- Tamaño de la población inicial: suma del número de lotes de todos los pedidos del ejemplar.
- Número de regeneraciones: triple de la suma del número de lotes de todos los pedidos del ejemplar.
- Probabilidad de cruce: 100%.
- Probabilidad de mutación: 30%.
- Tipos de cruce a aplicar: 1 punto de cruce, PMX.
- Tipos de mutación a aplicar: pareja.

C.7. Ejecutar

Una vez introducidos los datos y seleccionados los parámetros variables, ya se puede aplicar el algoritmo pulsando la opción *Ejecutar*.

Dependiendo del tamaño de la población inicial y del número de regeneraciones escogidos, el programa tardará más o menos tiempo en obtener una solución. Cuanto mayores sean estos valores, mayor será el tiempo de espera.

- ✓ En el caso de que se examine un solo ejemplar (*Nuevo* y *Abrir Ejemplar*), cuando el algoritmo llega a su fin, se muestra una ventana con el nombre de *Resultados* (Figura C.19) que contiene información de la solución obtenida.

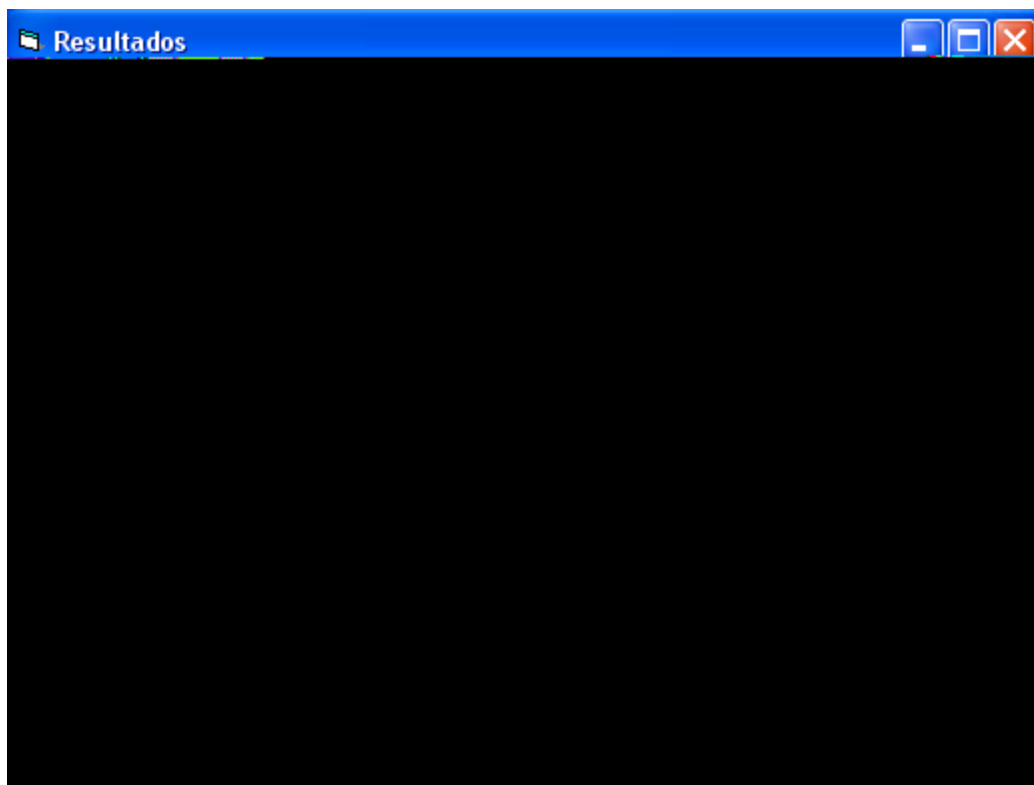


Figura C.19. Ejemplo de la ventana *Resultados*.

En esta ventana se pueden observar lo siguiente:

- Lo primero que aparece es un compendio de los operadores escogidos.
- Más abajo, aparece una tabla que muestra la solución obtenida codificada según “Número de pedido.Tipo de pieza”.
- A la izquierda de la tabla, se indica la medida de eficacia de la solución mediante los siguientes dos datos: Retraso Total y Retraso Medio de los pedidos.
- A continuación, existen dos enlaces con el nombre *Retrasos Pedidos* y *Órdenes de Producción* que contienen información más detallada de la solución obtenida.

Si se selecciona el primer enlace, aparece una ventana (Figura C.20) en la que se muestra las fechas en las que se termina la ejecución de los pedidos y el retraso producido en cada una de ellas. Si se hace clic en *Aceptar*, se retorna a la ventana anterior de *Resultados*.

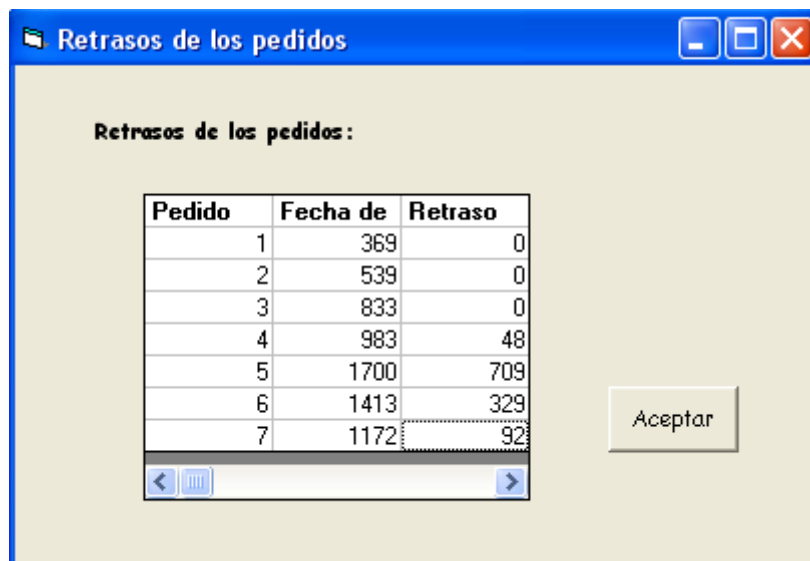


Figura C.20. Ejemplo de la ventana *Retrasos Pedidos*.

Si se selecciona el segundo enlace, *Órdenes de Producción*, se muestra un programa de producción (Figura C.21) en forma de tabla. se informa del orden de las piezas a ejecutar, así como de las cantidades de cada pieza y de los períodos de preparación entre la producción de diferentes tipos de piezas. Como información complementaria se indican los tiempos teóricos de ejecución y de preparación de las piezas y los tiempos acumulados en la ejecución de todo el ejemplar. Como en la ventana anterior, al hacer clic en *Aceptar*, el programa regresa a la ventana de *Resultados*.

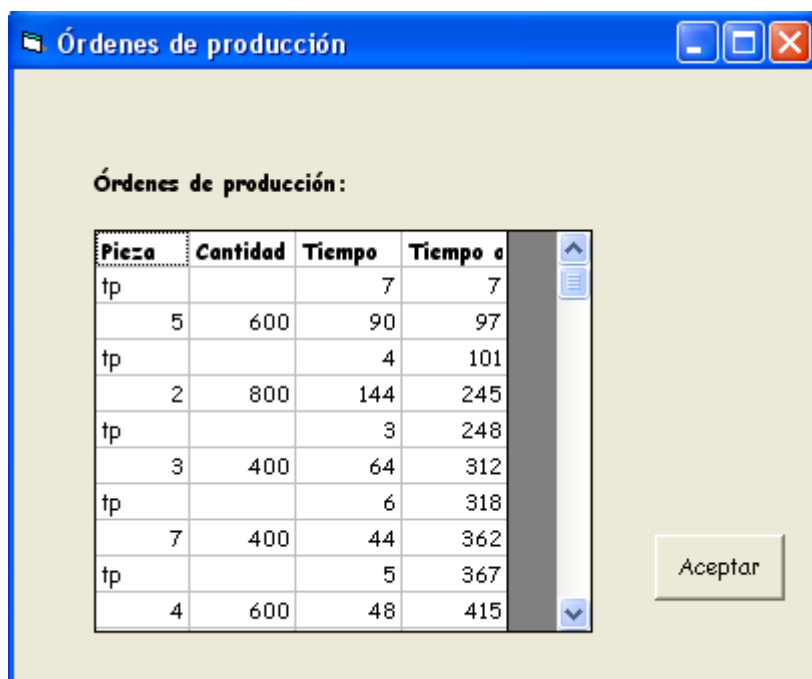


Figura C.21. Ejemplo de la ventana *Órdenes de Producción*.

- ✓ Si se examina una colección de ejemplares (*AbrirColección*), al finalizar la ejecución del algoritmo para todos ellos, se muestra una ventana con el nombre de *Resultados Colección* (Figura C.22) que contiene información de las soluciones obtenidas.

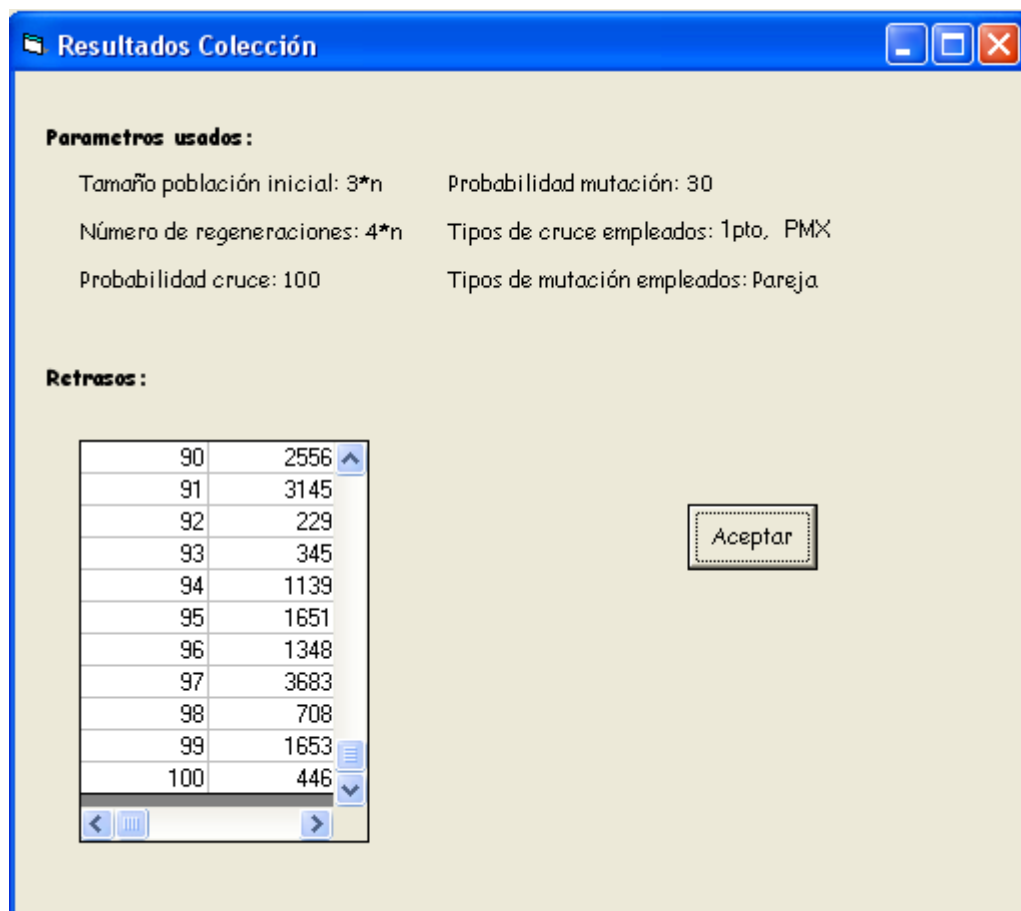


Figura C.22. Ejemplo de la ventana *Resultados Colección*.

En la ventana se puede observar los valores de los operadores escogidos y una tabla en la que se reflejan los retrasos totales de las soluciones encontradas.

Anexo D. Codificación del Programa

D.1. Formulario entrada

```
Private Sub MenuNuevo_Click()
```

```
AbrirNuevo.Show vbModal
```

```
End Sub
```

```
Private Sub MenuAbrirEjemplar_Click()
```

```
AbrirEjemplar.Show vbModal
```

```
End Sub
```

```
Private Sub MenuAbrirColeccion_Click()
```

```
AbrirColeccion.Show vbModal
```

```
End Sub
```

```
Private Sub MenuSalir_Click()
```

```
End
```

```
End Sub
```

D.2. Formulario AbrirNuevo

```
Private Sub BotonLeerNuevo_Click()
```

```
'en caso de error en la introducción de datos se enseña un mensaje de error  
On Error GoTo fin:
```

```
'se introduce el número de piezas
```

```
n = InputBox("Introduzca el número de piezas")
```

```
'se introducen los tiempos de preparación, que no pueden ser menores que cero
```

```
ReDim tp(n, n) As Integer
```

```
For i = 1 To n
```

```
    For j = 1 To n
```

```
tp:
```

```
    If i = j Then
```

```
        tp(i, j) = 0
```

```
    Elseif i <> j Then
```

```
        tp(i, j) = InputBox("Introduzca el tiempo de preparación entre la pieza " & i & " y la  
pieza " & j)
```

```
    End If
```

```
    If tp(i, j) < 0 Then
```

```
        Call MsgBox("Los tiempos de preparación deben ser números positivos")
```

```
        GoTo tp:
```

```
    End If
```

```
    Next j
```

```
Next i
```

```
'Se introducen los tiempos unitarios de ejecución, que no pueden ser menores que cero
```

```
ReDim tu(n) As Variant
```

```
For i = 1 To n
```

```
tu:
```

```
tu(i) = InputBox("Introduzca el tiempo unitario de ejecución de la pieza " & i)
```

```
    If tu(i) < 0 Then
```

```
        Call MsgBox("Los tiempos unitarios de producción deben ser mayores que 0")
```

```
        GoTo tu:
```

```
    End If
```

```
Next i
```

```
'Se introduce el número de pedidos del ejemplar
```

```
ncom = InputBox("Introduzca el número de pedidos del ejemplar")
```

```
'Se introduce el estado inicial de la máquina que debe pertenecer a un tipo de pieza del  
ejemplar
```

```
clin:
```

```
clin = InputBox("Introduzca el estado inicial de la máquina")
```

```
If clin < 1 Or clin > n Then
```

```
Call MsgBox("El estado inicial de la máquina debe corresponder a un tipo de pieza  
existente (entre 1 y " & n & ")")
```

GoTo clin:

End If

'Se introducen las fechas de vencimiento, los tipos de piezas existentes) en el ejemplar y
'las cantidades (mayores que cero)

ReDim comandas(ncom) As comanda

conta2 = 0

conta = 0

For i = 1 To ncom

comandas(i).dv = InputBox("Introduzca la fecha de vencimiento del pedido" & i)

comandas(i).npec = InputBox("Introduzca el número de tipos de piezas a ejecutar del
pedido " & i)

comandas(i).numerocomanda = i

ReDim comandas(i).piezas(comandas(i).npec) As pieza

For j = 1 To comandas(i).npec

If j = 1 Then

x = "primera"

Elseif j = 2 Then

x = "segunda"

Elseif j = 3 Then

x = "tercera"

Elseif j = 4 Then

x = "cuarta"

Elseif j = 5 Then

x = "quinta"

Elseif j = 6 Then

x = "sexta"

End If

codpieza:

comandas(i).piezas(j).codpieza = InputBox("Introduzca el código de la " & x & " pieza
de la comanda " & i)

If comandas(i).piezas(j).codpieza < 1 Or comandas(i).piezas(j).codpieza > n Then

Call MsgBox("Los tipos de piezas están comprendidos entre 1 y " & n)

GoTo codpieza:

End If

quantitat:

comandas(i).piezas(j).quantitat = InputBox("Introduzca la cantidad de piezas de tipo " &
comandas(i).piezas(j).codpieza & " a ejecutar de la comanda " & i)

If comandas(i).piezas(j).quantitat < 0 Then

Call MsgBox("La cantidad de piezas a ejecutar debe ser mayor que 0")

GoTo quantitat:

End If

conta2 = conta2 + 1

conta = conta + 1

comandas(i).piezas(j).numeropieza = conta

comandas(i).piezas(j).numpieza = conta2

Next j

conta2 = 0

Next i

Exit Sub

fin:

Call MsgBox("Los datos no se han introducido correctamente")

End Sub

Private Sub BotonOpciones_Click()

Opciones.Show vbModal

End Sub

Private Sub BotonEjecutar_Click()

Call Ejecutar
Resultados.Show vbModal

End Sub

Private Sub BotonSalir_Click()

Unload Me

End Sub

D.3. Formulario AbrirEjemplar

```
Private Sub BotonLeer_Click()
```

```
Dim archivo As Variant
```

```
Dim auxtp, auxu As Variant
```

```
Dim auxn, auxnejem, ejem, auxejem, auxncom, auxclin As Integer
```

```
Dim ausi, auxdv, auxnpec, auxcodpieza, auxquantitat As Integer
```

```
Dim conta, conta2 As Integer
```

```
CommonDialog1.ShowOpen
```

```
archivo = FreeFile
```

```
Open CommonDialog1.FileName For Input As archivo
```

```
'guarda el número de piezas
```

```
Input #archivo, auxn
```

```
n = Int(auxn)
```

```
'guarda los tiempos de preparación de la máquina dependientes de las piezas
```

```
'precedente y siguiente
```

```
ReDim tp(1 To n, 1 To n) As Integer
```

```
For i = 1 To n
```

```
    For j = 1 To n
```

```
        Input #archivo, auxtp
```

```
        tp(i, j) = auxtp
```

```
    Next j
```

```
Next i
```

```
'guarda los tiempos de ejecución unitarios de las piezas
```

```
ReDim tu(1 To n) As Variant
```

```
For i = 1 To n
```

```
    Input #archivo, auxu
```

```
    tu(i) = auxu
```

```
Next i
```

```
'se guardan el resto de variables del ejemplar
```

```
Input #archivo, auxclin
```

```
clin = Int(auxclin)
```

```
Input #archivo, auxncom
```

```
ncom = Int(auxncom)
```

```
ReDim comandas(ncom) As comanda
```

```
conta2 = 0
```

```
conta = 0
```

```
For i = 1 To ncom
```

```
    Input #archivo, aux
```

```
    ausi = Int(aux)
```

```
    Input #archivo, auxdv
```

```
    comandas(i).dv = auxdv
```

```
    Input #archivo, auxnpec
```

```
comandas(i).npec = auxnpec
comandas(i).numerocomanda = i
ReDim comandas(i).piezas(auxnpec) As pieza
For j = 1 To auxnpec
    Input #archivo, auxcodpieza
    comandas(i).piezas(j).codpieza = auxcodpieza
    Input #archivo, auxquantitat
    comandas(i).piezas(j).quantitat = auxquantitat
    conta2 = conta2 + 1
    conta = conta + 1
    comandas(i).piezas(j).numeropieza = conta
    comandas(i).piezas(j).numpieza = conta2
Next j
conta2 = 0
Next i

Close archivo

End Sub

Public Sub BotonEjecutar_Click()

Call Ejecutar
Resultados.Show vbModal

End Sub

Private Sub BotonOpciones_Click()

Opciones.Show vbModal

End Sub

Private Sub BotonSalir_Click()

Unload Me

End Sub
```

D.4. Formulario AbrirColección

```
Private Sub BotonLeerColeccion_Click()
```

```
Dim archivo As Variant
```

```
'Abre el archivo que contiene la información necesaria para ejecutar el programa
```

```
CommonDialog1.ShowOpen
```

```
archivo = FreeFile
```

```
Open CommonDialog1.FileName For Input As archivo
```

```
End Sub
```

```
Private Sub BotonEjecutar_Click()
```

```
Dim archivo As Variant
```

```
Dim auxtp, auxtu As Variant
```

```
Dim auxn, auxnejem, ejem, auxejem, auxncom, auxclin As Integer
```

```
Dim auxdv, auxnpec, ausi, aux, auxcodpieza, auxquantitat As Integer
```

```
Dim conta, conta2 As Integer
```

```
Dim tiempo(100), tiempo2(100) As Variant
```

```
tiempo(0) = Timer
```

```
men = tiempo(0)
```

```
may = 0
```

```
archivo = FreeFile
```

```
Open CommonDialog1.FileName For Input As archivo
```

```
'guarda el número de piezas
```

```
Input #archivo, auxn
```

```
n = Int(auxn)
```

```
'guarda los tiempos de preparación de la máquina dependientes de las piezas
```

```
'precedente y siguiente
```

```
ReDim tp(1 To n, 1 To n) As Integer
```

```
For i = 1 To n
```

```
    For j = 1 To n
```

```
        Input #archivo, auxtp
```

```
        tp(i, j) = auxtp
```

```
    Next j
```

```
Next i
```

```
'guarda los tiempos de ejecución unitarios de las piezas
```

```
ReDim tu(1 To n) As Variant
```

```
For i = 1 To n
```

```
    Input #archivo, auxtu
```

```
    tu(i) = auxtu
```

Next i

'guarda el número de ejemplares de la colección

Input #archivo, auxnejem

nejem = Int(auxnejem)

ejem = 0

'se empieza a completar la tabla de retrasos totales

ResultadosColeccion.Grid2.Appearance = flexFlat

ResultadosColeccion.Grid2.FixedCols = 0

ResultadosColeccion.Grid2.FixedRows = 0

ResultadosColeccion.Grid2.Rows = nejem + 1

ResultadosColeccion.Grid2.Cols = 2

ResultadosColeccion.Grid2.Row = 0

ResultadosColeccion.Grid2.Col = 0

ResultadosColeccion.Grid2.Text = "Ejemplar"

ResultadosColeccion.Grid2.CellFontBold = True

ResultadosColeccion.Grid2.Col = 1

ResultadosColeccion.Grid2.Row = 0

ResultadosColeccion.Grid2.Text = "Retraso"

ResultadosColeccion.Grid2.CellFontBold = True

While ejem < nejem

'se guardan las variables de cada ejemplar

Input #archivo, auxejem

ejem = Int(auxejem)

Input #archivo, auxncom

ncom = Int(auxncom)

Input #archivo, auxclin

clin = Int(auxclin)

ReDim comandas(ncom) As comanda

conta2 = 0

conta = 0

For i = 1 To ncom

Input #archivo, aux

ausi = Int(aux)

Input #archivo, auxdv

comandas(i).dv = auxdv

Input #archivo, auxnpec

comandas(i).npec = auxnpec

comandas(i).numerocomanda = i

ReDim comandas(i).piezas(auxnpec) As pieza

For j = 1 To auxnpec

Input #archivo, auxcodpieza

comandas(i).piezas(j).codpieza = auxcodpieza

Input #archivo, auxquantitat

comandas(i).piezas(j).quantitat = auxquantitat

conta2 = conta2 + 1

conta = conta + 1


```
comandas(i).piezas(j).numeropieza = conta
comandas(i).piezas(j).numpieza = conta2
Next j
conta2 = 0
Next i

'se ejecuta el algoritmo para el ejemplar en curso
Call Ejecutar

'se completa la tabla de retrasos totales
ResultadosColeccion.Grid2.Col = 0
ResultadosColeccion.Grid2.Row = ejem
ResultadosColeccion.Grid2.Text = ejem
ResultadosColeccion.Grid2.Col = 1
ResultadosColeccion.Grid2.Text = Int(retraso(1))

'para calcular los tiempos mínimos, máximos y promedio por ejemplar
tiempo(ejem) = Timer
tiempo2(ejem) = tiempo(ejem) - tiempo(ejem - 1)
su = su + tiempo2(ejem)
If tiempo2(ejem) < men Then
men = tiempo2(ejem)
End If
If tiempo2(ejem) > may Then
may = tiempo2(ejem)
End If
Wend

med = su / ejem

ResultadosColeccion.Grid2.Row = 0
ResultadosColeccion.Grid2.Col = 0
ResultadosColeccion.Grid2.Text = may
ResultadosColeccion.Grid2.Row = 1
ResultadosColeccion.Grid2.Text = men

ResultadosColeccion.Grid2.Col = 1
ResultadosColeccion.Grid2.Row = 0
ResultadosColeccion.Grid2.Text = med

'una vez analizados todos los ejemplares se muestran los resultados
If ejem = nejem Then
ResultadosColeccion.Show vbModal
End If

End Sub

Private Sub BotonOpciones_Click()

Opciones.Show vbModal
```

End Sub

Private Sub BotonSalir_Click()

Unload Me

End Sub

D.5. Formulario Opciones

```
Private Sub BotonAplicar_Click()
```

```
'Se debe aplicar al menos un tipo de cruce y un tipo de mutación
```

```
If Opciones.Cruce1 = Unchecked And Opciones.CruceOX = Unchecked And  
Opciones.CrucePMX = Unchecked Then
```

```
MsgBox "Debe seleccionar algun tipo de cruce", vbCritical, "Error"
```

```
End If
```

```
If Opciones.MutBlanda = Unchecked And Opciones.MutIntercambio = Unchecked And  
Opciones.MutPareja = Unchecked Then
```

```
MsgBox "Debe seleccionar algun tipo de mutación", vbCritical, "Error"
```

```
End If
```

```
'La probabilidad de cruce debe ser mayor del 80% y la de mutación menor del 40%
```

```
If Opciones.Textoprobcruce < 80 Then
```

```
MsgBox "La probabilidad de cruce debe ser mayor del 80%", vbCritical, "Error"
```

```
End If
```

```
If Opciones.Textoprobmutacion > 40 Then
```

```
MsgBox "La probabilidad de mutación debe ser menor del 40%", vbCritical, "Error"
```

```
End If
```

```
If Opciones.Textoprobcruce >= 80 And Opciones.Textoprobmutacion <= 40 And  
(Opciones.Cruce1 = Checked Or Opciones.CruceOX = Checked Or Opciones.CrucePMX  
= Checked) And (Opciones.MutBlanda = Checked Or Opciones.MutIntercambio =  
Checked Or Opciones.MutPareja = Checked) Then
```

```
Hide
```

```
End If
```

```
End Sub
```

```
Private Sub BotonSalir_Click()
```

```
Unload Me
```

```
End Sub
```

D.6. Formulario Resultados

```
Private Sub Form_Load()
```

```
Grid1.Appearance = flexFlat  
Grid1.FixedCols = 0  
Grid1.FixedRows = 0  
Grid1.Rows = suma + 1  
Grid1.Cols = 2
```

```
Grid1.ColWidth(1) = 2000  
Grid1.ColWidth(0) = 550  
Grid1.RowHeight(0) = 450
```

'la primera columna de la tabla contiene el orden cronológico de piezas a ejecutar

```
Grid1.Row = 0  
Grid1.Col = 0  
Grid1.Text = "Orden"  
Grid1.CellFontBold = True
```

```
For i = 1 To suma  
Grid1.Row = i  
Grid1.Text = i & "."  
Grid1.CellFontBold = True  
Next i
```

'la segunda columna señala los conjuntos de piezas a ejecutar según la codificación del algoritmo

```
Grid1.Col = 1  
Grid1.Row = 0  
Grid1.Text = "Comanda.Tipo de Pieza"  
Grid1.CellFontBold = True
```

```
For i = 1 To suma  
Grid1.Row = i  
Grid1.Text = Mid(poblacioninicial(1, i), 1, 2) & "." & Mid(poblacioninicial(1, i), 3, 2)  
Next i
```

'se indica el retraso medio de los pedidos

```
Dim retrasomedio As Single  
retrasomedio = Int(retraso(1)) / ncom  
LabelDato.Caption = retrasomedio
```

'se indica la suma de los retrasos de las piezas

```
LabelDato1.Caption = Int(retraso(1))
```

'se señalan las opciones elegidas para desarrollar el algoritmo

```
If Opciones.Opcion1poblacion = True Then  
can = "3*n"
```

```
Elseif Opciones.Opcion2poblacion = True Then  
can = "suma"  
Elseif Opciones.Opcion3poblacion = True Then  
can = Opciones.Textoelementos  
End If
```

```
If Opciones.Opcion1regeneraciones = True Then  
rege = "4*n"  
Elseif Opciones.Opcion2regeneraciones = True Then  
rege = "3*suma"  
Elseif Opciones.Opcion3regeneraciones = True Then  
rege = Opciones.Textoregeneraciones  
End If
```

```
Label1.Caption = "Tamaño población inicial: " & can  
Label2.Caption = "Número de regeneraciones: " & rege  
Label3.Caption = "Probabilidad cruce: " & Opciones.Textoprobcruce  
Label4.Caption = "Probabilidad mutación: " & Opciones.Textoprobmutacion
```

```
su = 0  
If Opciones.Cruce1 = Checked Then  
su = su + 1  
End If  
If Opciones.CruceOX = Checked Then  
su = su + 1  
End If  
If Opciones.CrucePMX = Checked Then  
su = su + 1  
End If
```

```
If su = 3 Then  
Label9.Caption = "1pto, "  
Label10.Caption = "OX, "  
Label11.Caption = "PMX"  
Elseif su = 2 Then  
If Opciones.Cruce1 = Unchecked Then  
Label9.Caption = "PMX, "  
Label10.Caption = "OX"  
Elseif Opciones.CruceOX = Unchecked Then  
Label9.Caption = "1pto, "  
Label10.Caption = "PMX"  
Elseif Opciones.CrucePMX = Unchecked Then  
Label9.Caption = "1pto, "  
Label10.Caption = "OX"  
End If  
Elseif su = 1 Then  
If Opciones.Cruce1 = Checked Then  
Label9.Caption = "1pto"  
Elseif Opciones.CruceOX = Checked Then  
Label9.Caption = "OX"  
Elseif Opciones.CrucePMX = Checked Then
```

```
Label9.Caption = "PMX"
End If
End If

su = 0
If Opciones.MutBlanda = Checked Then
su = su + 1
End If
If Opciones.MutPareja = Checked Then
su = su + 1
End If
If Opciones.MutIntercambio = Checked Then
su = su + 1
End If

If su = 3 Then
Label12.Caption = "Pareja, "
Label13.Caption = "Blanda, "
Label14.Caption = "Intercambio"
Elseif su = 2 Then
If Opciones.MutBlanda = Unchecked Then
Label12.Caption = "Pareja, "
Label13.Caption = "Intercambio"
Elseif Opciones.MutPareja = Unchecked Then
Label12.Caption = "Intercambio, "
Label13.Caption = "Blanda"
Elseif Opciones.MutIntercambio = Unchecked Then
Label12.Caption = "Pareja, "
Label13.Caption = "Blanda"
End If
Elseif su = 1 Then
If Opciones.MutBlanda = Checked Then
Label12.Caption = "Blanda"
Elseif Opciones.MutPareja = Checked Then
Label12.Caption = "Pareja"
Elseif Opciones.CrucePMX = Checked Then
Label12.Caption = "Intercambio"
End If
End If

End Sub

Private Sub OrdenesResultados_Click()

OrdenesProduccion.Show vbModal

End Sub

Private Sub RetrasosResultados_Click()
```

RetrasosPedidos.Show vbModal

End Sub

Private Sub AceptarResultados_Click()

Unload Me

End Sub

D.7. Formulario Resultados Colección

```
Private Sub Form_Load()
```

```
'se señalan las opciones elegidas para desarrollar el algoritmo
```

```
If Opciones.Opcion1poblacion = True Then
```

```
can = "3*n"
```

```
Elseif Opciones.Opcion2poblacion = True Then
```

```
can = "suma"
```

```
Elseif Opciones.Opcion3poblacion = True Then
```

```
can = Opciones.Textoelementos
```

```
End If
```

```
If Opciones.Opcion1regeneraciones = True Then
```

```
rege = "4*n"
```

```
Elseif Opciones.Opcion2regeneraciones = True Then
```

```
rege = "3*suma"
```

```
Elseif Opciones.Opcion3regeneraciones = True Then
```

```
rege = Opciones.Textoregeneraciones
```

```
End If
```

```
Label1.Caption = "Tamaño población inicial: " & can
```

```
Label2.Caption = "Número de regeneraciones: " & rege
```

```
Label3.Caption = "Probabilidad cruce: " & Opciones.Textoprobcruce
```

```
Label4.Caption = "Probabilidad mutación: " & Opciones.Textoprobmutacion
```

```
su = 0
```

```
If Opciones.Cruce1 = Checked Then
```

```
su = su + 1
```

```
End If
```

```
If Opciones.CruceOX = Checked Then
```

```
su = su + 1
```

```
End If
```

```
If Opciones.CrucePMX = Checked Then
```

```
su = su + 1
```

```
End If
```

```
If su = 3 Then
```

```
Label9.Caption = "1pto, "
```

```
Label10.Caption = "OX, "
```

```
Label11.Caption = "PMX"
```

```
Elseif su = 2 Then
```

```
    If Opciones.Cruce1 = Unchecked Then
```

```
        Label9.Caption = "PMX, "
```

```
        Label10.Caption = "OX"
```

```
    Elseif Opciones.CruceOX = Unchecked Then
```

```
        Label9.Caption = "1pto, "
```

```
        Label10.Caption = "PMX"
```



```
Elseif Opciones.CrucePMX = Unchecked Then
    Label9.Caption = "1pto, "
    Label10.Caption = "OX"
End If
Elseif su = 1 Then
    If Opciones.Cruce1 = Checked Then
        Label9.Caption = "1pto"
        Elseif Opciones.CruceOX = Checked Then
            Label9.Caption = "OX"
        Elseif Opciones.CrucePMX = Checked Then
            Label9.Caption = "PMX"
        End If
    End If
End If

su = 0
If Opciones.MutBlanda = Checked Then
    su = su + 1
End If
If Opciones.MutPareja = Checked Then
    su = su + 1
End If
If Opciones.MutIntercambio = Checked Then
    su = su + 1
End If

If su = 3 Then
    Label12.Caption = "Pareja, "
    Label13.Caption = "Blanda, "
    Label14.Caption = "Intercambio"
Elseif su = 2 Then
    If Opciones.MutBlanda = Unchecked Then
        Label12.Caption = "Pareja, "
        Label13.Caption = "Intercambio"
        Elseif Opciones.MutPareja = Unchecked Then
            Label12.Caption = "Intercambio, "
            Label13.Caption = "Blanda"
        Elseif Opciones.MutIntercambio = Unchecked Then
            Label12.Caption = "Pareja, "
            Label13.Caption = "Blanda"
        End If
    Elseif su = 1 Then
        If Opciones.MutBlanda = Checked Then
            Label12.Caption = "Blanda"
        Elseif Opciones.MutPareja = Checked Then
            Label12.Caption = "Pareja"
        Elseif Opciones.CrucePMX = Checked Then
            Label12.Caption = "Intercambio"
        End If
    End If
End If

End Sub
```

Private Sub AceptarResultados_Click()

Unload Me

End Sub

D.8. Formulario OrdenesProduccion

```
Private Sub Form_Load()
```

```
Dim aux, aux2, aux3, aux4, aux5 As Integer
```

```
Dim contador, contador2 As Integer
```

```
GridOrdenes.Appearance = flexFlat
```

```
GridOrdenes.FixedCols = 0
```

```
GridOrdenes.FixedRows = 0
```

```
GridOrdenes.Rows = suma * 2
```

```
GridOrdenes.Cols = 4
```

```
'la tercera columna contiene los tiempos de ejecución de cada operación
```

```
GridOrdenes.Row = 0
```

```
GridOrdenes.Col = 2
```

```
GridOrdenes.Text = "Tiempo"
```

```
GridOrdenes.CellFontBold = True
```

```
GridOrdenes.ColWidth(2) = 800
```

```
'la cuarta columna contiene los tiempos acumulados
```

```
GridOrdenes.Col = 3
```

```
GridOrdenes.Text = "Tiempo acumulado"
```

```
GridOrdenes.CellFontBold = True
```

```
GridOrdenes.ColWidth(3) = 800
```

```
'la segunda columna contiene la cantidad a producir de cada pieza
```

```
GridOrdenes.Col = 1
```

```
GridOrdenes.Text = "Cantidad"
```

```
GridOrdenes.CellFontBold = True
```

```
GridOrdenes.ColWidth(1) = 800
```

```
'la primera columna señala el tipo de pieza a ejecutar o si se debe preparar la máquina  
'para otra operación
```

```
GridOrdenes.Col = 0
```

```
GridOrdenes.Text = "Pieza"
```

```
GridOrdenes.CellFontBold = True
```

```
GridOrdenes.ColWidth(0) = 700
```

```
contador = 0
```

```
aux = Int(Mid(poblacioninicial(1, 1), 3, 2))
```

```
'se comprueba el estado inicial de la máquina para ver si hay que prepararla para la  
'primera operación
```

```
If clin <> aux Then
```

```
contador = contador + 1
```

```
GridOrdenes.Row = contador
```

```
GridOrdenes.Text = "tp"
```

```

GridOrdenes.Col = 2
GridOrdenes.Text = tp(clin, aux)
End If

```

'se registra en la tabla el tiempo de ejecución del primer conjunto de piezas

```

contador = contador + 1
GridOrdenes.Col = 0
GridOrdenes.Row = contador
GridOrdenes.Text = aux
aux3 = Int(Mid(poblacioninicial(1, 1), 1, 2))
aux4 = Int(Mid(poblacioninicial(1, 1), 6, 1))
aux5 = tu(aux) * comandas(aux3).piezas(aux4).quantitat
GridOrdenes.Col = 2
GridOrdenes.Text = aux5

```

'se registra en la tabla la cantidad a producir del primer conjunto de piezas

```

GridOrdenes.Col = 1
GridOrdenes.Text = comandas(aux3).piezas(aux4).quantitat
GridOrdenes.Col = 0

```

'se va registrando en la tabla el resto de valores

For i = 2 To suma

```

aux = Int(Mid(poblacioninicial(1, i - 1), 3, 2))

```

```

aux2 = Int(Mid(poblacioninicial(1, i), 3, 2))

```

If aux <> aux2 Then

```

GridOrdenes.Col = 0
contador = contador + 1
GridOrdenes.Row = contador
GridOrdenes.Text = "tp"
GridOrdenes.Col = 2
GridOrdenes.Text = tp(aux, aux2)
contador = contador + 1
GridOrdenes.Col = 0
GridOrdenes.Row = contador
GridOrdenes.Text = aux2
GridOrdenes.Col = 2
aux3 = Int(Mid(poblacioninicial(1, i), 1, 2))
aux4 = Int(Mid(poblacioninicial(1, i), 6, 1))
aux5 = tu(aux2) * comandas(aux3).piezas(aux4).quantitat
GridOrdenes.Text = aux5
GridOrdenes.Col = 1
GridOrdenes.Text = comandas(aux3).piezas(aux4).quantitat

```

'si existen conjuntos consecutivos del mismo tipo de piezas, se aglutinan en la misma
'orden

Elseif aux = aux2 Then

```

GridOrdenes.Col = 2
aux3 = Int(Mid(poblacioninicial(1, i), 1, 2))
aux4 = Int(Mid(poblacioninicial(1, i), 6, 1))
aux5 = tu(aux2) * comandas(aux3).piezas(aux4).quantitat
GridOrdenes.Text = GridOrdenes.Text + aux5
GridOrdenes.Col = 1

```

```
GridOrdenes.Text = GridOrdenes.Text + comandas(aux3).piezas(aux4).quantitat
End If
Next i
```

```
GridOrdenes.Rows = contador + 1
```

```
'se registran en la tabla los tiempos acumulados
```

```
contador2 = 0
```

```
For i = 1 To contador
```

```
GridOrdenes.Row = i
```

```
GridOrdenes.Col = 2
```

```
contador2 = contador2 + GridOrdenes.Text
```

```
GridOrdenes.Col = 3
```

```
GridOrdenes.Text = contador2
```

```
Next i
```

```
End Sub
```

```
Private Sub AceptarOrdenes_Click()
```

```
Unload Me
```

```
End Sub
```

D.9. Formulario RetrasosPedidos

```
Private Sub Form_Load()
```

```
GridRetrasos.Appearance = flexFlat  
GridRetrasos.FixedCols = 0  
GridRetrasos.FixedRows = 0
```

```
GridRetrasos.Rows = ncom + 1  
GridRetrasos.Cols = 3
```

```
'la primera columna contiene el número de cada pedido
```

```
GridRetrasos.Row = 0  
GridRetrasos.Col = 0  
GridRetrasos.Text = "Pedido"  
GridRetrasos.CellFontBold = True
```

```
'la segunda columna señala las fechas de terminio de los pedidos
```

```
GridRetrasos.Col = 1  
GridRetrasos.Text = "Fecha de terminio"  
GridRetrasos.CellFontBold = True
```

```
'la tercera columna señala los retrasos de los pedidos, es decir, las diferencias entre  
'las fechas de vencimiento y las de terminio para cada pedido
```

```
GridRetrasos.Col = 2  
GridRetrasos.Text = "Retraso"  
GridRetrasos.CellFontBold = True
```

```
'se completa la tabla con toda la información
```

```
For i = 1 To ncom  
GridRetrasos.Row = i  
GridRetrasos.Col = 0  
GridRetrasos.Text = i  
GridRetrasos.Col = 1  
GridRetrasos.Text = fecha(1, i)  
GridRetrasos.Col = 2  
GridRetrasos.Text = fitness(1, i)  
If fitness(1, i) < 0 Then  
GridRetrasos.Text = 0  
End If  
Next i
```

```
End Sub
```

```
Private Sub AceptarRetrasos_Click()
```

```
Unload Me
```

```
End Sub
```

D.10. Module1

```
Public n As Integer
Public tp() As Integer
Public tu() As Variant
Public clin As Integer
Public ncom As Integer
```

```
Public suma As Integer
Public cant As Integer
Public can As Integer
Public proporc As Variant
```

```
Public Type pieza
    codpieza As Integer
    quantitat As Integer
    numeropieza As Integer 'en general
    numpieza As Integer 'para cada pedido
End Type
```

```
Public Type comanda
    numerocomanda As Integer
    dv As Integer
    npec As Integer
    piezas() As pieza
End Type
```

```
Public comandas() As comanda
```

```
Public regeneraciones As Integer
Public rege As Integer
```

```
Public vector() As Variant
```

```
Public auxi() As Integer
```

```
Public poblacioninicial()
```

```
Public fmax() As Long
Public fecha()
Public fitness()
Public retraso()
Public probabilidad()
```

```
Public fmax2() As Long
Public fecha2()
Public fitness2()
Public retraso2()
```

Public individuo1 As Integer
Public individuo2 As Integer

Public auxpadre1() As Integer
Public auxpadre2() As Integer
Public hijo1() As Variant
Public auxhijo1() As Integer
Public hijo2() As Variant
Public auxhijo2() As Integer

Public auxiliar As Integer

D.11. Module2

```
Public Sub generacion_poblacioninicial_alternativa(cant, n, ncom, suma, vector())
```

```
Dim proporcion As Integer
```

```
Dim limitesup, limiteinf As Integer
```

```
Dim division As Integer
```

```
Dim lugar As Integer
```

```
Dim opta, opta2, tope, marca As Integer
```

```
'hallado(i,j) indica si la posición (i,j) de la matriz auxi contiene o no algún valor
```

```
ReDim hallado(cant, suma) As Boolean
```

```
For i = 1 To cant
```

```
    For j = 1 To suma
```

```
        hallado(i, j) = False
```

```
    Next j
```

```
Next i
```

```
'auxi contiene, en números enteros, "cant" combinaciones que servirán para configurar  
'la población inicial
```

```
ReDim auxi(cant, suma) As Integer
```

```
proporcion = Int(cant / 3)
```

```
'se genera la primera proporción según los pedidos
```

```
limitesup = ncom
```

```
limiteinf = 1
```

```
'sos contiene "cant" combinaciones de "ncom" números enteros auxsos(i,j) indica si la  
'posición (i,j) de la matriz sos contiene o no algún valor
```

```
ReDim sos(cant, ncom) As Integer
```

```
ReDim auxsos(cant, ncom) As Boolean
```

```
For j = 1 To proporcion
```

```
    For i = 1 To ncom
```

```
        auxsos(j, i) = False
```

```
    Next i
```

```
Next j
```

```
For i = 1 To proporcion
```

```
    sos(i, 1) = Int((limitesup - limiteinf + 1) * Rnd + limiteinf)
```

```
    auxsos(i, sos(i, 1)) = True
```

```
    For k = 2 To ncom
```

```
Comienzo2:
```

```
    sos(i, k) = Int((limitesup - limiteinf + 1) * Rnd + limiteinf)
```

```
    If auxsos(i, sos(i, k)) = False Then
```

```
        auxsos(i, sos(i, k)) = True
```

```
    ElseIf auxsos(i, sos(i, k)) = True Then
```

```
        GoTo Comienzo2:
```

```

End If
Next k
Next i

```

'de la matriz sos (pedidos) se pasa a la matriz auxi (piezas)

```

For i = 1 To proporcion
  lugar = 1
  For k = 1 To ncom
    For j = 1 To suma
      If Int(Mid(vector(j), 1, 2)) = sos(i, k) Then
        auxi(i, lugar) = Int(Mid(vector(j), 8, 2))
        lugar = lugar + 1
      End If
    Next j
  Next k
Next i

```

'se genera el segundo tercio de la población inicial

```
division = Int(suma / 4)
```

```
limitesup = division
```

```
limiteinf = 1
```

Comienzo4:

```

For i = proporcion + 1 To proporcion * 2
  auxi(i, limiteinf) = Int((limitesup - limiteinf + 1) * Rnd + limiteinf)
  hallado(i, auxi(i, limiteinf)) = True
  For k = limiteinf + 1 To limitesup

```

Comienzo3:

```

  auxi(i, k) = Int((limitesup - limiteinf + 1) * Rnd + limiteinf)
  If hallado(i, auxi(i, k)) = False Then
    hallado(i, auxi(i, k)) = True
  ElseIf hallado(i, auxi(i, k)) = True Then
    GoTo Comienzo3:
  End If
Next k
Next i

```

```
limitesup = limitesup + division
```

```
limiteinf = limiteinf + division
```

```
If (suma - limitesup) < division And (suma - limitesup) > 0 Then
```

```
  limitesup = suma
```

```
GoTo Comienzo4:
```

```
Elseif (suma - limitesup) = 0 Or (suma - limitesup) >= division Then
```

```
GoTo Comienzo4:
```

```
End If
```

'se genera la tercera parte de la población

```
opta = 0
```

```
opta2 = 0
```

```

For i = 1 To suma - 1
    opta = opta + tu(Int(Mid(vector(i), 3, 2))) * comandas(Int(Mid(vector(i), 1,
2))), piezas(Int(Mid(vector(i), 6, 1))).quantitat
    If Int(Mid(vector(i + 1), 6, 1)) = 1 Then
        opta2 = comandas(Int(Mid(vector(i), 1, 2))).dv
        'marca indica el pedido en que la suma de los tiempos unitarios (opta) superan la 'fecha
        de vencimiento (opta2)
        If (opta < opta2) Then
            marca = Int(Mid(vector(i), 1, 2))
        End If
    End If
Next i

opta = opta + tu(Int(Mid(vector(suma), 3, 2))) * comandas(Int(Mid(vector(suma), 1,
2))), piezas(Int(Mid(vector(suma), 6, 1))).quantitat
marca = marca - 1
tope = 0

'tope indica la pieza que marca el límite entre una vía de generación y otra
For i = 1 To marca
    tope = tope + comandas(i).npec
Next i

'hasta "tope" las piezas se agrupan acorde al tipo de piezas
ReDim sos(cant, n) As Integer
ReDim auxsos(cant, n) As Boolean

limitesup = n
limiteinf = 1

For j = 1 To cant - (proporcion * 2)
    For i = 1 To n
        auxsos(j, i) = False
    Next i
Next j

For i = proporcion * 2 + 1 To cant
    sos(i, 1) = Int((limitesup - limiteinf + 1) * Rnd + limiteinf)
    auxsos(i, sos(i, 1)) = True
    For k = 2 To n
        Comienzo5:
        sos(i, k) = Int((limitesup - limiteinf + 1) * Rnd + limiteinf)
        If auxsos(i, sos(i, k)) = False Then
            auxsos(i, sos(i, k)) = True
        ElseIf auxsos(i, sos(i, k)) = True Then
            GoTo Comienzo5:
        End If
    Next k
Next i

```

```

For i = proporcion * 2 + 1 To cant
  lugar = 1
  For k = 1 To n
    For j = 1 To tope
      If Int(Mid(vector(j), 1, 2)) = sos(i, k) Then
        auxi(i, lugar) = Int(Mid(vector(j), 8, 2))
        lugar = lugar + 1
      End If
    Next j
  Next k
Next i

```

'a partir de "tope" las piezas se agrupan según los pedidos, como sucede con el
 'primer tercio generado
 ReDim sos(cant, ncom) As Integer
 ReDim auxsos(cant, ncom) As Boolean

```

limitesup = ncom
limiteinf = 1

```

```

For j = 1 To cant - (proporcion * 2)
  For i = 1 To ncom
    auxsos(j, i) = False
  Next i
Next j

```

```

For i = proporcion * 2 + 1 To cant
  sos(i, 1) = Int((limitesup - limiteinf + 1) * Rnd + limiteinf)
  auxsos(i, sos(i, 1)) = True
  For k = 2 To ncom
    Comienzo6:
    sos(i, k) = Int((limitesup - limiteinf + 1) * Rnd + limiteinf)
    If auxsos(i, sos(i, k)) = False Then
      auxsos(i, sos(i, k)) = True
    ElseIf auxsos(i, sos(i, k)) = True Then
      GoTo Comienzo6:
    End If
  Next k
Next i

```

```

For i = proporcion * 2 + 1 To cant
  lugar = tope + 1
  For k = 1 To ncom
    For j = tope + 1 To suma
      If Int(Mid(vector(j), 1, 2)) = sos(i, k) Then
        auxi(i, lugar) = Int(Mid(vector(j), 8, 2))
        lugar = lugar + 1
      End If
    Next j
  Next k
Next i

```

End Sub

Public Sub calculo_fitness_padres()

Dim contpiezas() As Integer
 Dim aux1, aux2, aux3 As Long
 Dim ert, ert2 As Integer
 Dim sumatorio_retrasos As Long
 Dim propor As Integer

ReDim contpiezas(cant, ncom) As Integer

For i = 1 To cant
 For j = 1 To ncom
 contpiezas(i, j) = 0
 Next j
Next i

For j = 1 To cant

'aux1 calcula el tiempo de preparación de la primera pieza

If Mid(poblacioninicial(j, 1), 3, 1) = 0 Then
 aux1 = tp(clin, Mid(poblacioninicial(j, 1), 4, 1))
 ElseIf Mid(poblacioninicial(j, 1), 3, 1) = 1 Then
 aux1 = tp(clin, Mid(poblacioninicial(j, 1), 3, 2))
 End If

aux2 = 0

aux3 = 0

ert = Int(Mid(poblacioninicial(j, 1), 3, 2))

ert2 = Int(Mid(poblacioninicial(j, 1), 1, 2))

'aux3 calcula los tiempos unitarios de las piezas

aux3 = tu(ert) * comandas(ert2).piezas(Int(Mid(poblacioninicial(j, 1), 6, 1))).quantitat
 contpiezas(j, Int(Mid(poblacioninicial(j, 1), 1, 2))) = contpiezas(j, Int(Mid(poblacioninicial(j, 1), 1, 2))) + 1

For i = 2 To suma

'aux2 calcula el resto de tiempos de preparación

If Mid(poblacioninicial(j, i), 3, 2) <> Mid(poblacioninicial(j, i - 1), 3, 2) Then

aux2 = tp(Int(Mid(poblacioninicial(j, i - 1), 3, 2)), Int(Mid(poblacioninicial(j, i), 3, 2))) +

aux2

End If

ert = Int(Mid(poblacioninicial(j, i), 3, 2))

ert2 = Int(Mid(poblacioninicial(j, i), 1, 2))

aux3 = tu(ert) * comandas(ert2).piezas(Int(Mid(poblacioninicial(j, i), 6, 1))).quantitat +

aux3

contpiezas(j, Int(Mid(poblacioninicial(j, i), 1, 2))) = contpiezas(j,

Int(Mid(poblacioninicial(j, i), 1, 2))) + 1

'fecha(.) calcula la fecha en la cual termina cada pedido de cada padre

'fitness(.) calcula la diferencia entre la fecha en la cual termina cada pedido y la de

'vencimiento de cada una

If contpiezas(j, Int(Mid(poblacioninicial(j, i), 1, 2))) =

comandas(Int(Mid(poblacioninicial(j, i), 1, 2))).npec Then

```

    fecha(j, Int(Mid(poblacioninicial(j, i), 1, 2))) = aux1 + aux2 + aux3
    fitness(j, Int(Mid(poblacioninicial(j, i), 1, 2))) = aux1 + aux2 + aux3 -
comandas(Int(Mid(poblacioninicial(j, i), 1, 2))).dv
    End If
Next i
'fmax() calcula Fmax para cada individuo
fmax(j) = aux1 + aux2 + aux3
Next j

'retraso() calcula la suma de los retrasos de los pedidos
For i = 1 To cant
    For j = 1 To ncom
        If fitness(i, j) >= 0 Then
            retraso(i) = fitness(i, j) + retraso(i)
        End If
    Next j
Next i

For i = 1 To cant
    retraso(i) = retraso(i) + fmax(i) / 10000
Next i
'se otorgan probabilidades a cada inividuo, asignando mayor probabilidad a la cuarta
'parte de la población con mejor fitness
sumatorio_retrasos = 0

For i = 1 To cant
    sumatorio_retrasos = sumatorio_retrasos + retraso(i)
Next i

For i = 1 To cant
    probabilidad(i) = 0.8 * retraso(i) / sumatorio_retrasos
Next i

For i = 1 To Int(cant / 4)
    probabilidad(i) = probabilidad(i) + 0.2 / Int(cant / 4)
Next i

End Sub

Public Sub cruce_OX()

Dim limitesup, limiteinf As Integer
Dim mataux1() As Integer
Dim mataux2() As Integer
Dim azar(2), azar2(2) As Integer
Dim cont As Integer

limitesup = suma
limiteinf = 1
'se eligen, al azar, dos puntos de cruce

```

reazar:

For i = 1 To 2

azar(i) = Int((limitesup - limiteinf + 1) * Rnd + limiteinf)

Next i

'los puntos de cruce no pueden ser iguales

If azar(1) = azar(2) Then GoTo reazar:

'azar(1) debe ser menor que azar(2)

If azar(1) > azar(2) Then

aux = azar(1)

azar(1) = azar(2)

azar(2) = aux

End If

'se procede a ejecutar el cruce

ReDim mataux1(suma) As Integer

For i = azar(1) + 1 To azar(2)

auxhijo1(i) = auxpadre1(i)

mataux1(i) = 0

Next i

For i = 1 To azar(1)

mataux1(i) = auxpadre1(i)

Next i

For i = azar(2) + 1 To suma

mataux1(i) = auxpadre1(i)

Next i

cont = 1

For i = 1 To suma

For j = 1 To suma

If auxpadre2(i) = mataux1(j) Then

auxhijo1(cont) = auxpadre2(i)

cont = cont + 1

If cont = azar(1) + 1 Then

cont = azar(2) + 1

End If

End If

Next j

Next i

'análogamente, se obtiene el segundo hijo

ReDim mataux2(suma) As Integer

For i = azar(1) + 1 To azar(2)

auxhijo2(i) = auxpadre2(i)

mataux2(i) = 0

Next i

```
For i = 1 To azar(1)
mataux2(i) = auxpadre2(i)
Next i
```

```
For i = azar(2) + 1 To suma
mataux2(i) = auxpadre2(i)
Next i
```

```
cont = 1
For i = 1 To suma
  For j = 1 To suma
    If auxpadre1(i) = mataux2(j) Then
      auxhijo2(cont) = auxpadre1(i)
      cont = cont + 1
      If cont = azar(1) + 1 Then
        cont = azar(2) + 1
      End If
    End If
  Next j
Next i

End Sub
```

```
Public Sub cruce_1()
```

```
Dim limitesup, limiteinf As Integer
Dim mataux1() As Integer
Dim mataux2() As Integer
Dim cont As Integer
Dim azar, azar2 As Integer
```

```
limitesup = suma
limiteinf = 1
'al azar se escoge un punto de cruce
azar = Int((limitesup - limiteinf + 1) * Rnd + limiteinf)
```

```
'se obtiene el primer hijo
ReDim mataux1(suma) As Integer
```

```
For i = 1 To azar
auxhijo1(i) = auxpadre1(i)
mataux1(i) = 0
Next i
```

```
For i = azar + 1 To suma
mataux1(i) = auxpadre1(i)
Next i
```

```
cont = azar + 1
```



```

For i = 1 To suma
  For j = 1 To suma
    If auxpadre2(i) = mataux1(j) Then
      auxhijo1(cont) = auxpadre2(i)
      cont = cont + 1
    End If
  Next j
Next i

```

'análogamente, se obtiene el segundo hijo
 ReDim mataux2(suma) As Integer

```

For i = 1 To azar
  auxhijo2(i) = auxpadre2(i)
  mataux2(i) = 0
Next i

```

```

For i = azar + 1 To suma
  mataux2(i) = auxpadre2(i)
Next i

```

```

cont = azar + 1
For i = 1 To suma
  For j = 1 To suma
    If auxpadre1(i) = mataux2(j) Then
      auxhijo2(cont) = auxpadre1(i)
      cont = cont + 1
    End If
  Next j
Next i

```

End Sub

Public Sub cruce_PMX()

```

Dim limitesup, limiteinf As Integer
Dim mataux1() As Integer
Dim mataux2() As Integer
Dim azar(2), azar2(2) As Integer

```

```

For i = 1 To suma
  auxhijo1(i) = 0
  auxhijo2(i) = 0
Next i

```

```

limitesup = suma
limiteinf = 1
'se obtienen, al azar, los dos puntos de cruce
reazar:
For i = 1 To 2

```

```
azar(i) = Int((limitesup - limiteinf + 1) * Rnd + limiteinf)
Next i
```

'los dos puntos de cruce no deben ser iguales

```
If azar(1) = azar(2) Then GoTo reazar:
```

'azar(1) debe ser menor que azar(2)

```
If azar(1) > azar(2) Then
```

```
  aux = azar(1)
```

```
  azar(1) = azar(2)
```

```
  azar(2) = aux
```

```
End If
```

'se obtiene el primer hijo

```
ReDim mataux1(suma) As Integer
```

```
For i = azar(1) + 1 To azar(2)
```

```
  auxhijo1(i) = auxpadre1(i)
```

```
  mataux1(i) = 0
```

```
Next i
```

```
For i = 1 To azar(1)
```

```
  mataux1(i) = auxpadre1(i)
```

```
Next i
```

```
For i = azar(2) + 1 To suma
```

```
  mataux1(i) = auxpadre1(i)
```

```
Next i
```

```
For i = 1 To azar(1)
```

```
  For j = 1 To suma
```

```
    If auxpadre2(i) = mataux1(j) Then
```

```
      auxhijo1(i) = auxpadre2(i)
```

```
      mataux1(j) = 0
```

```
    End If
```

```
  Next j
```

```
Next i
```

```
For i = azar(2) + 1 To suma
```

```
  For j = 1 To suma
```

```
    If auxpadre2(i) = mataux1(j) Then
```

```
      auxhijo1(i) = auxpadre2(i)
```

```
      mataux1(j) = 0
```

```
    End If
```

```
  Next j
```

```
Next i
```

```
For i = 1 To suma
```

```
  For j = 1 To suma
```

```
    If auxpadre2(i) = mataux1(j) Then
```

```
      h = 1
```

```
        While auxhijo1(h) <> 0
            h = h + 1
        Wend
        auxhijo1(h) = auxpadre2(i)
    End If
Next j
Next i

'análogamente, se obtiene el segundo hijo
ReDim mataux2(suma) As Integer

For i = azar(1) + 1 To azar(2)
    auxhijo2(i) = auxpadre2(i)
    mataux2(i) = 0
Next i

For i = 1 To azar(1)
    mataux2(i) = auxpadre2(i)
Next i

For i = azar(2) + 1 To suma
    mataux2(i) = auxpadre2(i)
Next i

For i = 1 To azar(1)
    For j = 1 To suma
        If auxpadre1(i) = mataux2(j) Then
            auxhijo2(i) = auxpadre1(i)
            mataux2(j) = 0
        End If
    Next j
Next i

For i = azar(2) + 1 To suma
    For j = 1 To suma
        If auxpadre1(i) = mataux2(j) Then
            auxhijo2(i) = auxpadre1(i)
            mataux2(j) = 0
        End If
    Next j
Next i

For i = 1 To suma
    For j = 1 To suma
        If auxpadre1(i) = mataux2(j) Then
            h = 1
            While auxhijo2(h) <> 0
                h = h + 1
            Wend
            auxhijo2(h) = auxpadre1(i)
        End If
    Next j
Next i
```

```
Next j
Next i
```

```
End Sub
```

```
Public Sub mutacion_blanda()
```

```
Dim limitesup, limiteinf As Integer
Dim azar, azar2 As Integer
Dim aux As Integer
```

```
limitesup = suma
```

```
limiteinf = 1
```

```
'se obtiene al azar el gen que será mutado (por su inmediato)
```

```
azar = Int((limitesup - limiteinf + 1) * Rnd + limiteinf)
```

```
'se muta el primer hijo
```

```
If azar <> suma Then
```

```
aux = auxhijo1(azar)
```

```
auxhijo1(azar) = auxhijo1(azar + 1)
```

```
auxhijo1(azar + 1) = aux
```

```
Elseif azar = suma Then
```

```
aux = auxhijo1(azar)
```

```
auxhijo1(azar) = auxhijo1(1)
```

```
auxhijo1(1) = aux
```

```
End If
```

```
'se muta el segundo hijo
```

```
If azar <> suma Then
```

```
aux = auxhijo2(azar)
```

```
auxhijo2(azar) = auxhijo2(azar + 1)
```

```
auxhijo2(azar + 1) = aux
```

```
Elseif azar = suma Then
```

```
aux = auxhijo2(azar)
```

```
auxhijo2(azar) = auxhijo2(1)
```

```
auxhijo2(1) = aux
```

```
End If
```

```
End Sub
```

```
Public Sub mutacion_intercambio()
```

```
Dim limitesup, limiteinf As Integer
```

```
Dim azar(2), azar2(2) As Integer
```

```
Dim aux As Integer
```

```
limitesup = suma
```

```
limiteinf = 1
```

'se obtienen, al azar, los dos genes que se van a mutar

reazar:

For i = 1 To 2

azar(i) = Int((limitesup - limiteinf + 1) * Rnd + limiteinf)

Next i

'los dos genes deben ser diferentes

If azar(1) = azar(2) Then GoTo reazar:

'se muta el primer hijo

aux = auxhijo1(azar(1))

auxhijo1(azar(1)) = auxhijo1(azar(2))

auxhijo1(azar(2)) = aux

'análogamente se muta el segundo hijo

aux = auxhijo2(azar(1))

auxhijo2(azar(1)) = auxhijo2(azar(2))

auxhijo2(azar(2)) = aux

End Sub

Public Sub mutacion_pareja()

Dim limitesup, limiteinf As Integer

Dim azar, azar2 As Integer

Dim guardado As Variant

Dim mutapieza, encontrado As Integer

limitesup = suma

limiteinf = 1

'azar es el gen que se va a mutar

aza:

azar = Int((limitesup - limiteinf + 1) * Rnd + limiteinf)

'dicho gen puede ser cualquiera excepto el primero

If azar = 1 Then

GoTo aza:

End If

'en este caso, interesa tener la información codificada de los genes y no números 'enteros asociados

For i = 1 To suma

j = 1

While auxhijo1(i) <> Int(Mid(poblacioninicial(individuo1, j), 8, 2))

j = j + 1

Wend

hijo1(i) = poblacioninicial(individuo1, j)

Next i

For i = 1 To suma

j = 1

```

While auxhijo2(i) <> Int(Mid(poblacioninicial(individuo2, j), 8, 2))
j = j + 1
Wend
hijo2(i) = poblacioninicial(individuo2, j)
Next i

```

'se coloca el gen sito en el lugar 'azar' en un gen anterior a este tal que dos piezas del mismo tipo sean ejecutados consecutivamente

```

i = azar
mutapieza = Int(Mid(hijo1(i), 3, 2))
encontrado = 0
While i <> 1 And encontrado <> mutapieza
i = i - 1
encontrado = Int(Mid(hijo1(i), 3, 2))
Wend

```

```

If encontrado = mutapieza Or (i = 1 And mutapieza = clin) Then
guardado = hijo1(azar)
j = azar
While j <> i + 1
hijo1(j) = hijo1(j - 1)
j = j - 1
Wend
hijo1(i + 1) = guardado
End If

```

'análogamente, se muta el segundo hijo

```

i = azar
mutapieza = Int(Mid(hijo2(i), 3, 2))
encontrado = 0
While i <> 1 And encontrado <> mutapieza
i = i - 1
encontrado = Int(Mid(hijo2(i), 3, 2))
Wend

```

```

If encontrado = mutapieza Or (i = 1 And mutapieza = clin) Then
guardado = hijo2(azar)
j = azar
While j <> i + 1
hijo2(j) = hijo2(j - 1)
j = j - 1
Wend
hijo2(i + 1) = guardado
End If

```

'se pasa a enteros asociados como el resto de cruces/mutaciones

```

For i = 1 To suma
auxhijo1(i) = Int(Mid(hijo1(i), 8, 2))
auxhijo2(i) = Int(Mid(hijo2(i), 8, 2))
Next i

```

End Sub

Public Sub aplicarcruce()

Dim sumtot As Integer

Dim cru1, cruOX, cruPMX As Boolean

Dim probcru1, probcruOX, probcruPMX

Dim elegircruce

sumtot = 0

cru1 = False

cruOX = False

cruPMX = False

probcru1 = 0

probcruOX = 0

probcruPMX = 0

'sumtot indica el número de cruces elegidos

If Opciones.Cruce1 = Checked Then

cru1 = True

sumtot = sumtot + 1

End If

If Opciones.CruceOX = Checked Then

cruOX = True

sumtot = sumtot + 1

End If

If Opciones.CrucePMX = Checked Then

cruPMX = True

sumtot = sumtot + 1

End If

'si se elige mas de un tipo de cruce, la elección será aleatoria y, con la misma
'probabilidad para todas ellos

elegircruce = Rnd

If sumtot = 1 Then

 If cru1 = True Then

 Call cruce_1

 Elseif cruOX = True Then

 Call cruce_OX

 Elseif cruPMX = True Then

 Call cruce_PMX

 End If

End If

If sumtot = 2 Then

 If cru1 = True Then

 probcru1 = 0.5

 If elegircruce <= 0.5 Then

 Call cruce_1

```

    End If
End If
If cruOX = True Then
    probcruOX = 0.5 + probcru1
    If probcruOX = 0.5 And elegircruce <= 0.5 Then
        Call cruce_OX
    ElseIf probcruOX = 1 And elegircruce > 0.5 Then
        Call cruce_OX
    End If
End If
If cruPMX = True Then
    probcruPMX = 0.5 + probcru1 + probcruOX
    If probcruPMX = 0.5 And elegircruce <= 0.5 Then
        Call cruce_PMX
    ElseIf probcruPMX = 1 And elegircruce > 0.5 Then
        Call cruce_PMX
    End If
End If
End If

If sumtot = 3 Then
    If elegircruce <= 1 / 3 Then
        Call cruce_1
    ElseIf elegircruce > 1 / 3 And elegircruce <= 2 / 3 Then
        Call cruce_OX
    ElseIf elegircruce > 2 / 3 And elegircruce <= 1 Then
        Call cruce_PMX
    End If
End If

End Sub

```

```
Public Sub aplicarmutacion()
```

```

Dim sumtot As Integer
Dim mutablanda, mutaintercambio, mutapareja As Boolean
Dim probmutablanda, probmutaintercambio, probmutapareja
Dim elegirmutación

```

```

sumtot = 0
mutablanda = False
mutaintercambio = False
mutapareja = False
probmutablanda = 0
probmutaintercambio = 0
probmutapareja = 0

```

```

'sumtot indica el número de mutaciones elegidas
If Opciones.MutBlanda = Checked Then
    mutablanda = True

```



```
sumtot = sumtot + 1
```

```
End If
```

```
If Opciones.MutIntercambio = Checked Then
```

```
mutaintercambio = True
```

```
sumtot = sumtot + 1
```

```
End If
```

```
If Opciones.MutPareja = Checked Then
```

```
mutapareja = True
```

```
sumtot = sumtot + 1
```

```
End If
```

'si se elige mas de un tipo de mutación, la elección será aleatoria y, con la misma
'probabilidad para todas ellos

```
elegirmutacion = Rnd
```

```
If sumtot = 1 Then
```

```
  If mutablanda = True Then
```

```
    Call mutacion_blanda
```

```
  ElseIf mutaintercambio = True Then
```

```
    Call mutacion_intercambio
```

```
  ElseIf mutapareja = True Then
```

```
    Call mutacion_pareja
```

```
  End If
```

```
End If
```

```
If sumtot = 2 Then
```

```
  If mutablanda = True Then
```

```
    probmutablanda = 0.5
```

```
    If elegirmutacion <= 0.5 Then
```

```
      Call mutacion_blanda
```

```
    End If
```

```
  End If
```

```
  If mutaintercambio = True Then
```

```
    probmutaintercambio = 0.5 + probmutablanda
```

```
    If probmutaintercambio = 0.5 And elegirmutacion <= 0.5 Then
```

```
      Call mutacion_intercambio
```

```
    ElseIf probmutaintercambio = 1 And elegirmutacion > 0.5 Then
```

```
      Call mutacion_intercambio
```

```
    End If
```

```
  End If
```

```
  If mutapareja = True Then
```

```
    probmutapareja = 0.5 + probmutablanda + probmutaintercambio
```

```
    If probmutapareja = 0.5 And elegirmutacion <= 0.5 Then
```

```
      Call mutacion_pareja
```

```
    ElseIf probmutapareja = 1 And elegirmutacion > 0.5 Then
```

```
      Call mutacion_pareja
```

```
    End If
```

```
  End If
```

```
End If
```

```
If sumtot = 3 Then
```

```

If elegirmutacion <= 1 / 3 Then
    Call mutacion_blanda
Elseif elegirmutacion > 1 / 3 And elegirmutacion <= 2 / 3 Then
    Call mutacion_pareja
Elseif elegirmutacion > 2 / 3 And elegirmutacion <= 1 Then
    Call mutacion_intercambio
End If
End If

End Sub

```

```

Public Sub calculo_fitness_hijos(clin, auxiliar, ncom, poblaciongenerada(), tp() As Integer,
tu() As Variant, comandas() As comanda)

```

```

Dim contpiezas2() As Integer
Dim aux1, aux2, aux3 As Long
Dim ert, ert2 As Long

```

```

ReDim contpiezas2(cant, ncom) As Integer
For i = 1 To auxiliar
    For j = 1 To ncom
        contpiezas2(i, j) = 0
    Next j
Next i

```

```

For j = 1 To auxiliar

```

```

'aux1 calcula el tiempo de preparación de la primera pieza

```

```

    If Mid(poblaciongenerada(j, 1), 3, 1) = 0 Then
        aux1 = tp(clin, Mid(poblaciongenerada(j, 1), 4, 1))
    Elseif Mid(poblaciongenerada(j, 1), 3, 1) = 1 Then
        aux1 = tp(clin, Mid(poblaciongenerada(j, 1), 3, 2))
    End If

```

```

    aux2 = 0

```

```

    aux3 = 0

```

```

    ert = Int(Mid(poblaciongenerada(j, 1), 3, 2))

```

```

    ert2 = Int(Mid(poblaciongenerada(j, 1), 1, 2))

```

```

'aux3 calcula los tiempos unitarios de las piezas

```

```

    aux3 = tu(ert) * comandas(ert2).piezas(Int(Mid(poblaciongenerada(j, 1), 6, 1))).quantitat
    contpiezas2(j, Int(Mid(poblaciongenerada(j, 1), 1, 2))) = contpiezas2(j,
Int(Mid(poblaciongenerada(j, 1), 1, 2))) + 1

```

```

'aux2 calcula el resto de tiempos de preparación

```

```

    For i = 2 To suma

```

```

        If Mid(poblaciongenerada(j, i), 3, 2) <> Mid(poblaciongenerada(j, i - 1), 3, 2) Then
            aux2 = tp(Int(Mid(poblaciongenerada(j, i - 1), 3, 2)), Int(Mid(poblaciongenerada(j, i),
3, 2))) + aux2

```

```

        End If

```

```

        ert = Int(Mid(poblaciongenerada(j, i), 3, 2))

```

```

        ert2 = Int(Mid(poblaciongenerada(j, i), 1, 2))

```

```

        aux3 = tu(ert) * comandas(ert2).piezas(Int(Mid(poblaciongenerada(j, i), 6,
1))).quantitat + aux3

```

```

    contpiezas2(j, Int(Mid(poblaciongenerada(j, i), 1, 2))) = contpiezas2(j,
Int(Mid(poblaciongenerada(j, i), 1, 2))) + 1
'fecha2(.) calcula la fecha en la cual termina cada pedido de cada padre
'fitness2(.) calcula la diferencia entre la fecha en la cual termina cada pedido y la de
'vencimiento de cada una
    If contpiezas2(j, Int(Mid(poblaciongenerada(j, i), 1, 2))) =
comandas(Int(Mid(poblaciongenerada(j, i), 1, 2))).npec Then
        fecha2(j, Int(Mid(poblaciongenerada(j, i), 1, 2))) = aux1 + aux2 + aux3
        fitness2(j, Int(Mid(poblaciongenerada(j, i), 1, 2))) = aux1 + aux2 + aux3 -
comandas(Int(Mid(poblaciongenerada(j, i), 1, 2))).dv
    End If
Next i
'fmax2() calcula Fmax para cada individuo
fmax2(j) = aux1 + aux2 + aux3
Next j

'retraso2() calcula la suma de los retrasos de los pedidos
For i = 1 To auxiliar
    For j = 1 To ncom
        If fitness2(i, j) >= 0 Then
            retraso2(i) = fitness2(i, j) + retraso2(i)
        End If
    Next j
Next i

For i = 1 To cant
    retraso(i) = retraso(i) + fmax(i) / 10000
Next i

End Sub

```

D.12. Module3

Public Sub Ejecutar()

```
Dim dig1, dig2, dig3, dig4, dig5 As Variant
Dim aux As Integer
Dim poblaciongenerada()
Dim numreg, numcru As Integer
Dim sumato, elegido1, elegido2
Dim probcruce, probmutacion
Dim iguales1, iguales2, iguales3, iguales4 As Boolean
Dim retraso_global()
Dim matretraso()
Dim auxpobini()
Dim liminf, limsup As Integer
```

'la variable "suma" guarda el número total de piezas de todas los pedidos a ejecutar

```
suma = 0
For i = 1 To ncom
    suma = suma + comandas(i).npec
Next i
```

'Codificación de los lotes de piezas agrupadas en la variable vector

```
ReDim vector(suma) As Variant
For j = 1 To ncom
    If comandas(j).numerocomanda < 10 Then
        dig1 = 0
        dig2 = comandas(j).numerocomanda
    ElseIf comandas(j).numerocomanda >= 10 And comandas(j).numerocomanda < 20
Then
        dig1 = 1
        dig2 = comandas(j).numerocomanda - 10
    ElseIf comandas(j).numerocomanda >= 20 Then
        dig1 = 2
        dig2 = comandas(j).numerocomanda - 20
    End If
    For k = 1 To comandas(j).npec
        If comandas(j).piezas(k).codpieza < 10 Then
            dig3 = 0
            dig4 = comandas(j).piezas(k).codpieza
        ElseIf comandas(j).piezas(k).codpieza >= 10 Then
            dig3 = 1
            dig4 = comandas(j).piezas(k).codpieza - 10
        End If
        aux = comandas(j).piezas(k).numeropieza
        dig5 = comandas(j).piezas(k).numpieza
        vector(aux) = dig1 & dig2 & dig3 & dig4 & "." & dig5 & "." & aux & " "
    Next k
Next j
```

'guarda en la variable "cant" la opción del tamaño de población inicial elegida

```
If Opciones.Opcion1poblacion.Value = True Then  
cant = n * 3  
Elseif Opciones.Opcion2poblacion.Value = True Then  
cant = suma  
Elseif Opciones.Opcion3poblacion.Value = True Then  
cant = Opciones.Textoelementos.Text  
End If
```

'generación de la población inicial

Call generacion_poblacioninicial_alternativa(cant, n, ncom, suma, vector())

ReDim poblacioninicial(cant, suma) As Variant

'de la matriz auxi de numeros enteros se pasa a la matriz poblacioninicial de vectores

'codificados

```
For i = 1 To cant  
  For j = 1 To suma  
    For k = 1 To suma  
      If auxi(i, j) = Int(Mid(vector(k), 8, 2)) Then  
        poblacioninicial(i, j) = vector(k)  
      End If  
    Next k  
  Next j  
Next i
```

'fmax() calcula Fmax para cada individuo

ReDim fmax(cant) As Long

'fecha(.) calcula la fecha en la cual termina cada pedido de cada padre

ReDim fecha(cant, ncom)

'fitness(.) calcula la diferencia entre la fecha de terminio de cada pedido y la de

'vencimiento de cada una

ReDim fitness(cant, ncom)

'retraso() calcula la suma de los retrasos de los pedidos

ReDim retraso(cant)

'probabilidad() contiene las probabilidades asignadas a cada individuo para sus

'posteriores cruces

ReDim probabilidad(cant)

'Se procede a calcular el fitness de la población inicial

Call calculo_fitness_padres

'se guarda en regeneraciones la opción de número de regeneraciones elegida

```
If Opciones.Opcion1regeneraciones = True Then  
regeneraciones = n * 4  
Elseif Opciones.Opcion2regeneraciones = True Then  
regeneraciones = suma * 3  
Elseif Opciones.Opcion3regeneraciones = True Then  
regeneraciones = Opciones.Textoregeneraciones  
End If
```

'numreg marca la iteración en la que está el algoritmo

numreg = 1

otraregeneracion:

'numcru marca el número de cruces que se llevan realizados en la iteración

numcru = 1

ReDim poblaciongenerada(cant, suma)

While numcru < (Int(cant / 2) * 2) + 1

'se eligen dos individuos al azar para decidir cruces

volver:

eleccion_individuos:

elegido1 = Rnd

elegido2 = Rnd

If elegido1 = 0 Or elegido2 = 0 Then GoTo eleccion_individuos:

sumato = 0

individuo1 = 0

i = 1

While sumato < elegido1

sumato = sumato + probabilidad(i)

individuo1 = individuo1 + 1

i = i + 1

Wend

sumato = 0

individuo2 = 0

i = 1

While sumato < elegido2

sumato = sumato + probabilidad(i)

individuo2 = individuo2 + 1

i = i + 1

Wend

'los dos individuos deben ser der diferentes

If individuo1 = individuo2 Then GoTo eleccion_individuos:

ReDim auxpadre1(suma) As Integer

ReDim auxpadre2(suma) As Integer

ReDim hijo1(suma) As Variant

ReDim auxhijo1(suma) As Integer

ReDim hijo2(suma) As Variant

ReDim auxhijo2(suma) As Integer

'se pasa de la cadena codificada a enteros simples para que la programación de los

'cruces sea más sencilla

For i = 1 To suma

auxpadre1(i) = Int(Mid(poblacioninicial(individuo1, i), 8, 2))

auxpadre2(i) = Int(Mid(poblacioninicial(individuo2, i), 8, 2))

Next i

'se aplican cruces/mutaciones según probabilidades

probmutación = Rnd

probcruce = Rnd

If probcruce <= Opciones.Textoprobcruce / 100 Then

'esta función opta al azar por los diversos cruces elegidos

Call aplicarcruce

'si padre e hijo se repiten, vuelve a mutar y cruzar

iguales1 = True

iguales2 = True

iguales3 = True

iguales4 = True

For i = 1 To suma

If auxhijo1(i) <> auxpadre1(i) Then

iguales1 = False

End If

Next i

If iguales1 = True Then

GoTo volver:

End If

For i = 1 To suma

If auxhijo1(i) <> auxpadre2(i) Then

iguales2 = False

End If

Next i

If iguales2 = True Then

GoTo volver:

End If

For i = 1 To suma

If auxhijo2(i) <> auxpadre1(i) Then

iguales3 = False

End If

Next i

If iguales3 = True Then

GoTo volver:

End If

For i = 1 To suma

If auxhijo2(i) <> auxpadre2(i) Then

iguales4 = False

End If

Next i

```

If iguales4 = True Then
GoTo volver:
End If

```

'para que si no hay cruce, las matrices auxhijo1, auxhijo2 no queden vacías

```

Elseif probcruce > Opciones.Textoprobcruce / 100 Then
  For i = 1 To suma
    auxhijo1(i) = auxpadre1(i)
    auxhijo2(i) = auxpadre2(i)
  Next i
End If

```

If probmutacion <= (Opciones.Textoprobmutacion / 100) Then
 'esta función opta al azar por las diversas mutaciones elegidas
 Call aplicarmutacion
 End If

'Una vez realizados los cruces/mutaciones se pasa de enteros asociados a cadena
 'codificada en la matriz poblaciongenerada

```

For i = 1 To suma
  j = 1
  While auxhijo1(i) <> Int(Mid(poblacioninicial(individuo1, j), 8, 2))
    j = j + 1
  Wend
  poblaciongenerada(numcru, i) = poblacioninicial(individuo1, j)
Next i

```

```

For i = 1 To suma
  j = 1
  While auxhijo2(i) <> Int(Mid(poblacioninicial(individuo2, j), 8, 2))
    j = j + 1
  Wend
  poblaciongenerada(numcru + 1, i) = poblacioninicial(individuo2, j)
Next i

```

```

numcru = numcru + 2
Wend

```

```

auxiliar = Int(cant / 2) * 2

```

'fmax2() calcula Fmax para cada hijo

```
ReDim fmax2(auxiliar) As Long
```

'fecha2(,) calcula la fecha en la cual termina cada pedido de cada hijo

```
ReDim fecha2(auxiliar, ncom)
```

'fitness(,) calcula la diferencia entre la fecha de terminio de cada pedido y la de
 'vencimiento de cada una

```
ReDim fitness2(cant, ncom)
```

'retraso2() calcula la suma de los retrasos de los pedidos

```
ReDim retraso2(auxiliar)
```

'se calcula el fitness de la poblacion generada

Call calculo_fitness_hijos(cclin, auxiliar, ncom, poblaciongenerada(), tp(), tu(), comandas())

'se agrupan los retrasos de los individuos de padres e hijos en una misma matriz

ReDim retraso_global(cant + auxiliar)

For i = 1 To cant

retraso_global(i) = retraso(i)

Next i

For i = cant + 1 To cant + auxiliar

retraso_global(i) = retraso2(i - cant)

Next i

'selección de la población que constituirá la población inicial (contenida en auxpobini)

'de la siguiente iteración

ReDim matretraso(cant + auxiliar)

ReDim auxpobini(cant, suma)

'matretraso=0 si ese individuo no ha sido cogido. en caso contrario, vale 1

For i = 1 To (cant + auxiliar)

matretraso(i) = 0

Next i

'se selecciona directamente una proporción de los individuos con mejor fitness

For nose = 1 To Int((12 / 20) * cant)

pos = 1

While matretraso(pos) = 1

pos = pos + 1

Wend

elmejor = pos

For j = 1 To (cant + auxiliar)

If retraso_global(j) < retraso_global(elmejor) And matretraso(j) = 0 Then

elmejor = j

End If

Next j

If elmejor <= cant Then

For j = 1 To suma

auxpobini(nose, j) = poblacioninicial(elmejor, j)

Next j

Elseif elmejor <= (cant + auxiliar) And elmejor > cant Then

For j = 1 To suma

auxpobini(nose, j) = poblaciongenerada(elmejor - cant, j)

Next j

End If

matretraso(elmejor) = 1

Next nose

'se selecciona, al azar, otra proporción entre la población inicial

For nose = Int(((12 / 20) * cant)) + 1 To Int((16 / 20) * cant)

reaza:

limsup = cant

liminf = 1

aza = Int((limsup - liminf + 1) * Rnd + liminf)

```

If matretraso(aza) = 1 Then
GoTo reaza:
Elseif matretraso(aza) = 0 Then
  For j = 1 To suma
    auxpobini(nose, j) = poblacioninicial(aza, j)
  Next j
End If
Next nose

```

'se selecciona, al azar, otra proporción entre la población filial

```

For nose = Int(((16 / 20) * cant)) + 1 To cant
reaza2:
limsup = cant + auxiliar
liminf = cant + 1
aza2 = Int((limsup - liminf + 1) * Rnd + liminf)
  If matretraso(aza2) = 1 Then
    GoTo reaza2:
  Elseif matretraso(aza2) = 0 Then
    For j = 1 To suma
      auxpobini(nose, j) = poblaciongenerada(aza2 - cant, j) 'aza2-
      tamaño_matriz_poblaciongenerada
    Next j
  End If
Next nose

```

```

For i = 1 To cant
  For j = 1 To suma
    poblacioninicial(i, j) = auxpobini(i, j)
  Next j
Next i

```

'se calcula el fitness de esta población como paso previo a la siguiente iteración

```

ReDim retraso(cant)
Call calculo_fitness_padres

```

'se suma una iteración al contador de regeneraciones

```
numreg = numreg + 1
```

'si no se han cumplido el número de regeneraciones elegido, se procede a iterar de nuevo

```
If numreg < regeneraciones Then GoTo otraregeneracion:
```

```
can = cant
```

```
rege = regeneraciones
```

```
End Sub
```