MITSUBISHI ELECTRIC
Changes for the Better

PLC
Conceptos básicos de la programación (Texto estructurado)

Este curso incluye los pasos para crear los programas básicos utilizados para el manejo de los controladores programables MELSEC.
El texto estructurado (ST, por sus siglas en inglés) se utiliza para las descripciones de programas de este curso.

L(FOD)00552SPA

Introducción

Objetivo del curso

Este curso explica los pasos para crear programas de control en texto estructurado (ST) para los controladores programables MELSEC.

Es necesario realizar la totalidad del siguiente curso o poseer el conocimiento equivalente necesario antes de tomar este curso:

Conceptos básicos de la programación

El conocimiento o la experiencia en el lenguaje de programación C o BASIC lo ayudarán a entender el contenido de este curso.

Introducción Estructura del curso

El contenido de este curso es el siguiente.

Capítulo 1 - Descripción general del texto estructurado

Este capítulo describe las características y aplicaciones adecuadas del texto estructurado (ST).

Capítulo 2 - Reglas básicas de los programas de ST

Este capítulo describe las reglas básicas utilizadas para la creación de programas en ST.

Capítulo 3 - Creación de programas de control de E/S

Este capítulo describe el proceso para la creación de programas de control de E/S.

Capítulo 4 - Operaciones aritméticas

Este capítulo describe el proceso de creación programas de operación aritmética.

Capítulo 5 - Ramificación condicional

Este capítulo describe la ramificación condicional.

Capítulo 6 - Almacenamiento y manipulación de datos

Este capítulo describe cómo escribir programas concisos para el almacenamiento y la manipulación de datos.

Capítulo 7 - Manipulación de datos de cadenas

Este capítulo describe los métodos para la manipulación de datos de cadenas.

Prueba final

Nota aprobatoria: 60% o mayor

Introducción Cómo usar esta herramienta de aprendizaje en línea

Ir a la página siguiente		Ir a la página siguiente.
Regresar a la página anterior		Regresar a la página anterior.
Ir a la página deseada		Se visualizará el "Índice", lo que le permitirá navegar a la página deseada.
Salir del aprendizaje		Salir del aprendizaje.

Introducción

Precauciones para el uso

Precauciones de seguridad

Cuando aprenda con productos reales, lea con cuidado las precauciones de seguridad ubicadas en los manuales correspondientes.

Precauciones en este curso

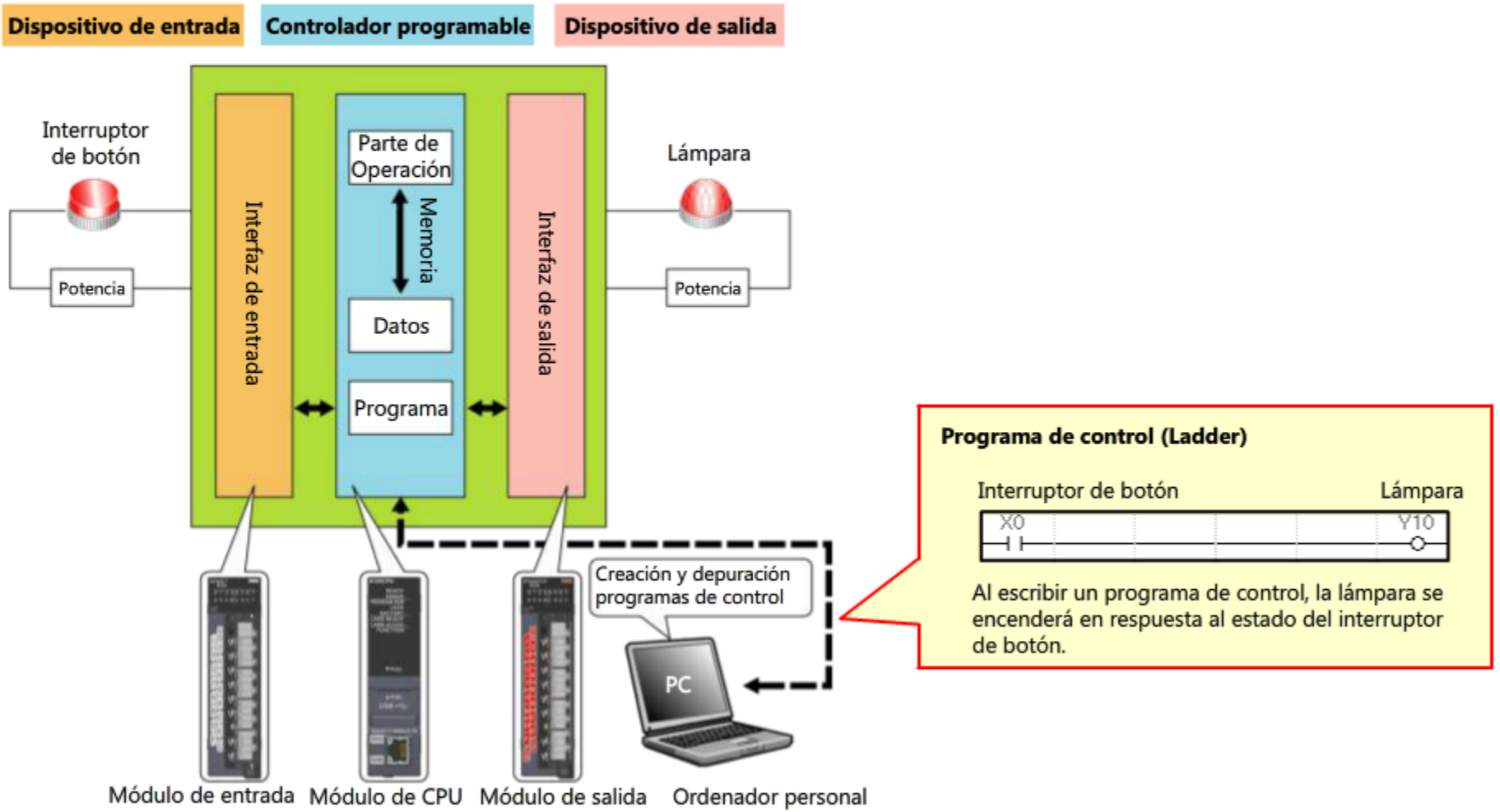
Es posible que las pantallas visualizadas del software de ingeniería MELSOFT que use sean diferentes a las de este curso. Este curso utiliza símbolos de escalera de MELSOFT GX Works3 para crear programas.

Capítulo 1 Descripción general del texto estructurado

Este capítulo describe las características y aplicaciones adecuadas del texto estructurado (ST).

1.1 Programas de control

La siguiente figura ilustra la configuración de un sistema de controlador programable.
Los controladores programables operan según los programas de control.
La operación de los controladores programables puede ser configurada como lo desee mediante la creación de programas de control.



Los lenguajes de programación para controladores programables son definidos por la norma internacional desarrollada por la International Electrotechnical Commission (IEC) (Comisión Electrotécnica Internacional).

1.2 Características del ST y comparación con otros lenguajes de IEC

La IEC 61131 es una norma internacional para los sistemas de controladores programables.
Los lenguajes de programación para los controladores programables están estandarizados por la norma IEC 61131-3.
El ST es uno de los lenguajes de programación estándar.
Cada lenguaje tiene características diferentes para adaptarse a su aplicación y a las habilidades de los programadores.

La siguiente tabla muestra las características de los lenguajes de programación IEC 61131-3.

Lenguaje de programación	Características
Ladder **D**iagram (LD) (Diagrama de escalera)	• Los símbolos para los contactos y las bobinas son usados para crear un programa que aparente ser un circuito eléctrico. • El flujo de programa es fácil de seguir y entender, aun para los principiantes.
Structured **T**ext (ST) (Texto estructurado)	• Los programas se escriben en texto (caracteres). • El ST le será fácil de aprender a aquellos que tengan experiencia en la escritura de lenguajes de programación en C o BASIC. • Las fórmulas de cálculo son similares a las expresiones matemáticas, las cuales son fáciles de entender. • ST es ideal para la manipulación de datos.
Function **B**lock **D**iagram (FBD) (Diagrama de bloques de función)	• Los programas son escritos mediante la organización de bloques con diferentes funciones e indicación de relaciones entre los bloques. • El FBD mejora la lectura ya que se puede ver toda la operación.
Sequential **F**unction **C**hart (SFC) (Gráfico de función secuencial)	• Las condiciones y los procesos están escritos como diagramas de flujo. • El flujo de programa es fácil de entender.
Instruction **L**ist (IL) (Lista de instrucciones)	• La IL es similar al lenguaje de maquinaria. • La IL casi no se usa en la actualidad.

Este curso describe los pasos para escribir programas de control básicos con ST.

1.3 Resumen

Los temas de este capítulo son:

- La relación entre los sistemas de controladores programables y los programas de control
- Norma internacional para programas de control
- Características del ST

Puntos importantes a tener en cuenta:

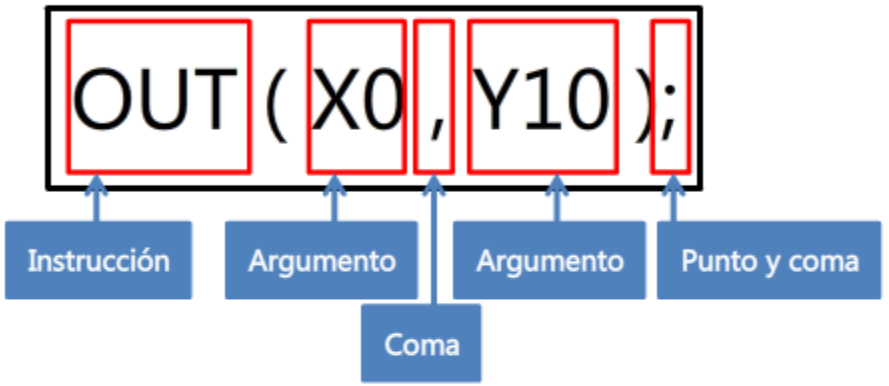
La relación entre los sistemas de controladores programables y los programas de control	• Los controladores programables operan según los programas de control. • La operación de los controladores programables puede ser configurada como lo desee mediante la creación de programas de control.
Norma internacional para programas de control	• El ST es uno de los lenguajes de programación IEC. • Otros lenguajes de programación de IEC incluyen LD, FBD, SFC e IL, que tienen sus características particulares para adaptarse a la aplicación y a las habilidades de los programadores.
Características del ST	• ST es fácil de aprender para aquellos con experiencia en la escritura de programas de lenguaje C o BASIC. • Los cálculos como la suma y la resta pueden escribirse como expresiones matemáticas utilizadas típicamente, las cuales son fáciles de entender. • ST es ideal para la manipulación de datos.

Capítulo 2 Reglas básicas de los programas en ST

Este capítulo describe las reglas básicas utilizadas para la creación de programas en ST.

2.1 Ejemplo de programa básico (Declaración de control de E/S)

Esta sección ilustra un ejemplo de un programa básico de ST.
Con el siguiente programa de ejemplo, la salida Y10 se enciende cuando la entrada X0 se enciende y la Y10 se apaga cuando la X0 se apaga.



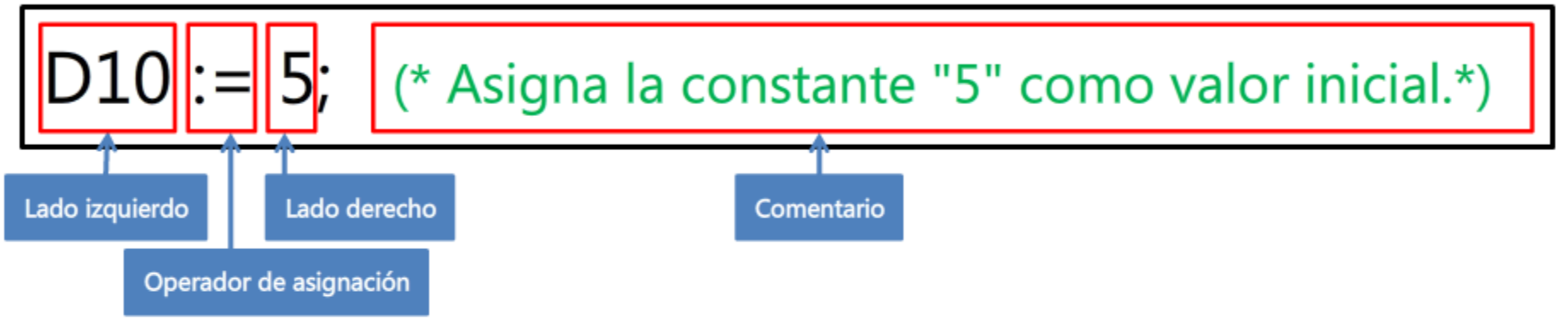
Una instrucción define la operación a ejecutarse.
Los argumentos se escriben entre paréntesis luego de una instrucción.
Los argumentos se utilizan para describir variables, expresiones aritméticas y valores constantes.
Con los controladores programables MELSEC, los dispositivos del módulo CPU pueden usarse como variables.

El número de argumentos depende de la instrucción.
Los distintos argumentos se separan por comas (,).

La única línea que se muestra arriba representa una declaración. Cada declaración termina con un punto y coma (;).
Un programa se escribe mediante la combinación de declaraciones.

2.2 Ejemplo de programa básico (Declaración de asignación)

El siguiente ejemplo ilustra un programa que utiliza una declaración de asignación.
La siguiente declaración asigna la constante decimal "5" a la variable "D10".



Se utiliza un operador de asignación (:=) para esta declaración de asignación. Note que los dos puntos (:) se ubican a la izquierda del signo igual (=).
Un operador de asignación asigna el valor del lado derecho al lado izquierdo.

La adición de un comentario a un programa hace que la operación sea más entendible. Encierre los comentarios entre dos asteriscos (* *).

2.3 Notación numérica

Con el programa de ejemplo de la página anterior, se asignó un valor decimal a la variable.

A veces se utilizan valores diferentes a decimales como los binarios y hexadecimales para el control secuencial.
La siguiente tabla muestra los tipos de notación numérica que se usan en ST para los controladores programables MELSEC.

Tipo de notación numérica	Método de notación	Ejemplo
Binario	Agregue un prefijo de "2#".	**2#**11010
Octal	Agregue un prefijo de "8#".	**8#**32
Decimal	Entrada directa	26
	Agregue un prefijo de "K".	**K**26
Hexadecimal	Agregue un prefijo de "16#".	**16#**1A
	Agregue un prefijo de "H"	**H**1A

Los ejemplos de programas para asignar valores a variables se muestran a continuación.

```
D10 := 8#32;
D10 := K26;
D10 := H1A;
```

2.3.1 Notación de bits

Los bits representan condiciones true/false (verdaderas/falsas) tales como los estados de on/off (encendido/apagado) de las señales. Los bits también representan la creación/no-creación de condiciones.
EN ST, los bits no pueden ser escritos como "ON" y "OFF". Estos son expresados como "1" (ON) y "0" (OFF). Los bits también pueden ser expresados como "TRUE" y "FALSE".

La siguiente lista muestra los diferentes tipos de notación.

Estado	ON	OFF
	True	False
Notación numérica	1	0
Notación true/false	TRUE	FALSE

Estos son algunos ejemplos de la asignación de valores para variables de tipo bit.

Notación numérica

```
X0 := 1;
```

=

Notación true/false

```
X0 := TRUE;
```

Notación numérica

```
X0 := 0;
```

=

Notación true/false

```
X0 := FALSE;
```

2.4 Secuencia de ejecución de programa

Las declaraciones de ST se ejecutan de arriba hacia abajo.

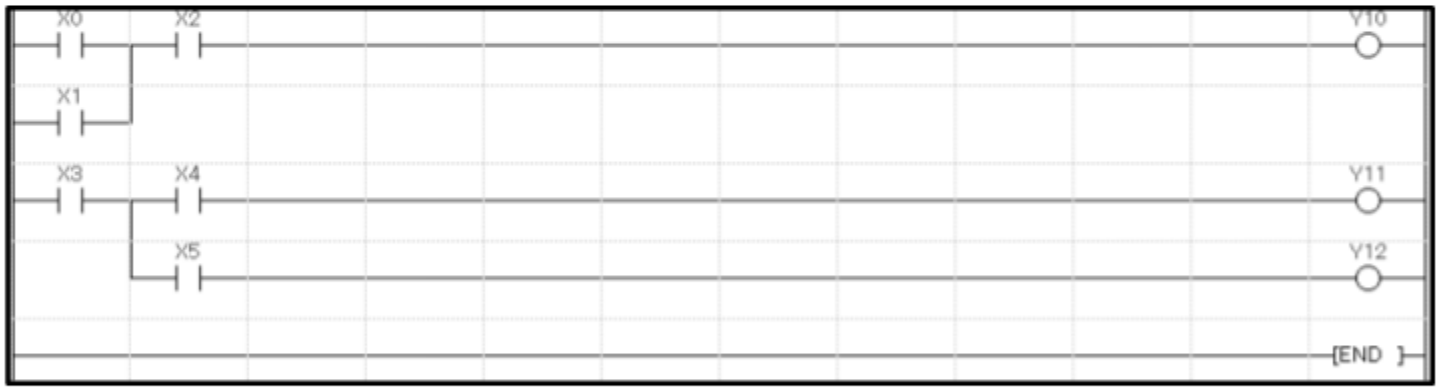
Ejemplo de programa de ST

```
Y10 := (X0 OR X1) AND X2;          (* Ejecutado primero *)
Y11 := X3 AND X4;                  (* Ejecutado segundo *)
Y12 := X3 AND X5;   (* Ejecutado tercero. No requiere una declaración END al final.*)
```

Ejecución repetitiva

*Si bien es necesaria una declaración END al final del programa en LD, no es necesario en ST.

El siguiente programa de escalera representa la misma operación que el ejemplo de programa de ST anterior.



Ejecución repetitiva

Al igual que con LD, las instrucciones en ST se ejecutan de manera repetitiva mediante el regreso a la primera instrucción luego de alcanzar la última.

2.5 Resumen

Los temas de este capítulo son:

- Programa ST básico
- Formato de declaración de asignación
- Notación numérica
- Secuencia de ejecución de programa
- Comentario

Puntos importantes a tener en cuenta:

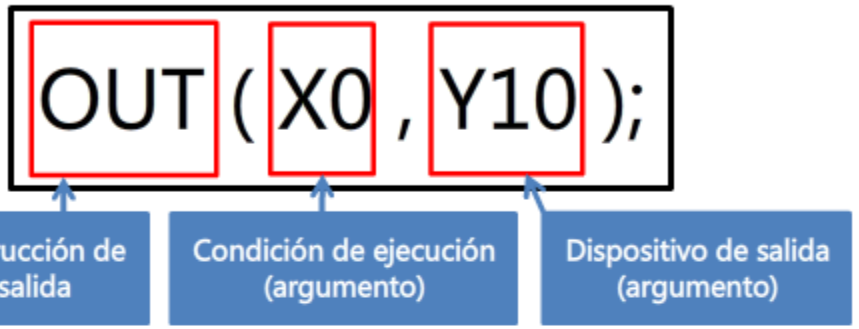
Programa ST básico	• Una declaración es el elemento mínimo de los programas de ST. • Cada declaración termina con un punto y coma (;). • Un programa se escribe mediante la combinación de declaraciones.
Formato de declaración de asignación	• Para las asignaciones se utiliza un operador de asignación (:=).
Notación numérica	• Tipos de notación numérica en ST • "1" y "0" se usan para valores bit en ST en vez de la notación "ON" y "OFF". • Los valores bit pueden expresarse como "TRUE" y "FALSE" en ST.
Secuencia de ejecución de programa	• Los programas creados en ST se ejecutan de arriba hacia abajo. • Al igual que con los programas LD, los ST procesan de manera repetitiva volviendo al comienzo del programa una vez que se ha alcanzado el fin del proceso.
Comentario	• La adición de un comentario a un programa hace que la operación sea más entendible. • Los comentarios están encerrados entre dos asteriscos (* *).

Capítulo 3 Creación de programas de control de E/S

Este capítulo describe los pasos para la creación de programas de control de E/S en ST.

3.1 Programas de control de E/S

El siguiente es un ejemplo de programa para el control de E/S de un controlador programable.



"OUT" es la salida de instrucción. Un argumento especifica una condición de ejecución y el dispositivo al cual está dirigida. Cuando la condición de ejecución X0 se satisface, el dispositivo Y10 se enciende.

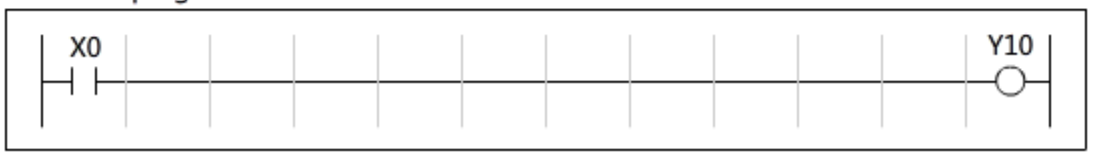
Haga clic en el interruptor de entrada que se muestra a continuación. El interruptor de entrada X0 se enciende.

- Cuando el interruptor de entrada X0 se enciende, la lámpara de salida Y10 se enciende.
- Cuando el interruptor de entrada X0 se apaga, la lámpara de salida Y10 se apaga.

Ejemplo de control de programa de E/S escrito en ST

```
OUT(X0,Y10);
```

El mismo programa escrito en LD



3.1 Programas de control de E/S

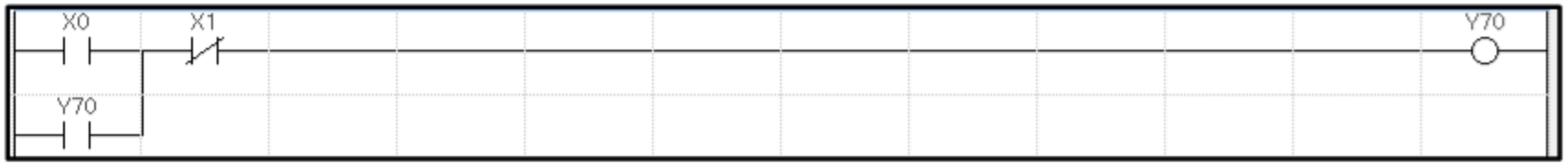
De manera similar al LD, existen muchas instrucciones disponibles además de la instrucción OUT, tales como las instrucciones de control de E/S y las instrucciones de procesamiento de datos.
Consulte el manual de programación de su controlador programable para obtener más información sobre las instrucciones disponibles en ST.

Tenga en cuenta que si escribe "OUT(X0, Y10);" como "Y10 := X0;" se produce la misma operación.

```
Y10 := X0;  (* Misma operación que "OUT(X0, Y10);" *)
```

3.2 Combinación de condiciones múltiples

El siguiente programa de escalera representa un circuito de autorretención.



El mismo programa puede escribirse en ST de la siguiente manera.

```
Y70 := (X0 OR Y70) AND NOT X1;
```

Operador lógico

Como se muestra anteriormente, los operadores lógicos se utilizan para combinar condiciones múltiples en ST.

La siguiente tabla muestra los operadores lógicos.

Operador	Significado
OR	Lógico O
AND	Lógico Y
NOT	Negación lógica
XOR	Exclusivo O

3.3 Definición del significado de las variables

Usando el ST con los controladores programables MELSEC, se pueden asignar a variables ambos dispositivos y etiquetas como alias. Los usuarios pueden usar etiquetas según las aplicaciones.

Cuando se asigna una etiqueta relacionada a la aplicación, la operación es más fácil de entender.

```
Y10 := (X0 OR X1) AND X2;     (* Escrito con los nombres de los dispositivos *)
```

```
Lamp := (Switch0 OR Switch1) AND Switch2;     (* Escrito con las etiquetas *)
```

Las etiquetas pueden renombrarse con el software de ingeniería de MELSOFT.

Los ejemplos posteriores de programas en este curso se describen con etiquetas.

3.4 Resumen

Los temas de este capítulo son:

Ejemplos de programa de control de E/S

- Los operadores lógicos se utilizan para combinar condiciones múltiples en ST.
- Los nombres de los dispositivos y las etiquetas pueden ser usados como nombres de variables.

Puntos importantes a tener en cuenta:

Combinación de condiciones múltiples	• Los operadores lógicos se utilizan para combinar condiciones en ST.
Definición del significado de las variables	• Cuando se asigna una etiqueta relacionada a la aplicación, la operación es más fácil de entender.

Capítulo 4 Operaciones aritméticas

Este capítulo describe el proceso de creación programas de operación aritmética.

- Descripción de operaciones aritméticas
- Especificación de tipos de datos correspondientes a los rangos numéricos
- Asignación de nombres a las variables para evitar inconsistencias en tipos de datos

4.1 Operaciones aritméticas básicas

Este programa de ejemplo saca el total del volumen de producción de dos líneas de producción separadas.
El lado derecho de una ecuación es una operación aritmética que contiene variables y operadores aritméticos.

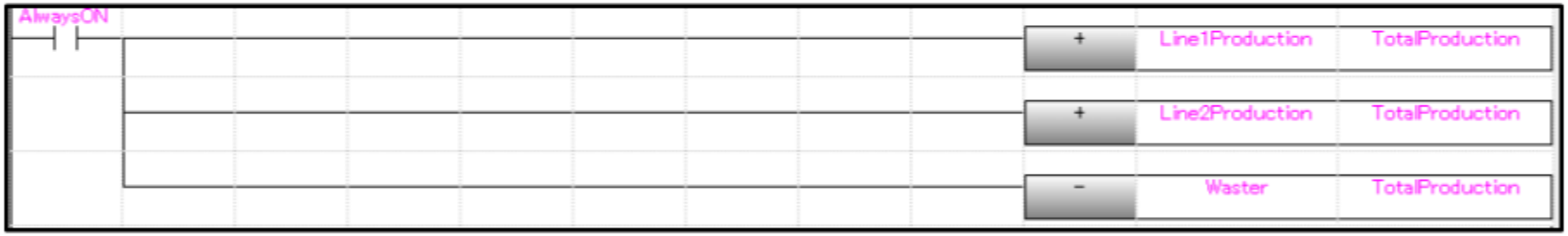
Programa aritmético de ejemplos escrito con ST

Operador de suma

Operador de resta

```
TotalProduction := Line1Production + Line2Production - Waster;
(* Saque el total del volumen de producción de las dos líneas, reste el número de los productos defectuosos del total y asigne el valor obtenido. *)
```

El mismo programa escrito en LD se muestra a continuación.



Como se muestra anteriormente, el programa debe ser escrito con 3 líneas en Ladder, pero con ST, puede ser escrito en una línea.

La siguiente tabla muestra los operadores aritméticos básicos.

Operador	Significado
+	Suma
-	Resta
*	Multiplicación
/	División

4.2 Tipos de datos de las variables

Se debe especificar un tipo de datos a cada variable para definir el rango de valores a ser manipulado.
Los tipos de datos para valores numéricos usados en ST son tipos de números bit, enteros y reales.

Entre los tipos de datos usados en ST, la tabla de abajo muestra los tipos de datos usados en este curso.

Tipo de datos		Rango de datos
Bit		Estado ON/OFF de los dispositivos bit y los resultados de ejecución de estado true/false
Entero	Palabra (sin firmar)	0 - 65.535
	Palabra (firmado)	-32.768 - 32.767
	Palabra doble (sin firmar)	0 - 4.294.967.295
	Palabra doble (firmado)	-2.147.483.648 - 2.147.483.647

Al usar el tipo entero, seleccione el tipo de palabra o palabra doble de acuerdo con el rango de datos y seleccione la opción firmado o sin firmar de acuerdo con la necesidad de manipular valores negativos.
Especifique el tipo de datos de una variable cuando el nombre de etiqueta esté definido con el software de ingeniería de MELSOFT.

4.3 Nombres de variables que representan tipos de datos

El uso de tipos de datos diferentes en los lados izquierdo y derecho de una ecuación de asignación puede causar un error de compilación o un resultado inesperado.
A continuación aparece un ejemplo de dicho caso.

```
ValueA := ValueB;      (* ValueA: Palabra entero  ValueB: Palabra doble entero *)
```

Un entero de palabra doble no puede ser asignado a un entero de una palabra. Sin embargo, en este caso, el tipo de datos no es reconocible.

Los prefijos que representan el tipo de datos pueden ser agregados a nombres de variables para que los tipos de datos sean identificables visualmente.
Este tipo de nombramiento de variables es conocido como notación húngara.

Tipo de datos		Rango de datos	Prefijo	Expansión del prefijo
Bit		Estado ON/OFF de los dispositivos bit y los resultados de ejecución de estado true/false	b	**B**it
Entero	Palabra (sin firmar)	0 - 65.535	u	**u**nsigned word (palabra sin firmar)
	Palabra (firmado)	-32.768 - 32.767	w	signed **w**ord (palabra firmada)
	Palabra doble (sin firmar)	0 - 4.294.967.295	ud	**u**nsigned **d**ouble-word (palabra doble sin firmar)
	Palabra doble (firmado)	-2.147.483.648 - 2.147.483.647	d	signed **d**ouble-word (palabra doble firmada)

El programa de ejemplo en la parte superior de esta página puede ser escrito de la siguiente manera con la notación húngara:

```
wValueA := dValueB;  (* La variable de la palabra doble no puede ser asignada a una variable de palabra. *)
```

Al usar la notación húngara, las inconsistencias en tipos de datos pueden identificarse en el proceso de escritura de un programa.

En el resto del curso, los nombres de las variables en los ejemplos están escritos en notación húngara.

4.4 Resumen

Los temas de este capítulo son:

- Descripción de operaciones aritméticas
- Especificación de tipos de datos correspondientes a los rangos numéricos
- Agregado de nombres de variables que representan tipos de datos

Puntos importantes a tener en cuenta:

Operaciones aritméticas básicas	• Los operadores comunes a los lenguajes de programación pueden usarse en ST para expresar cálculos rápidos.
Tipos de datos de las variables	• Se debe especificar un tipo de datos a cada variable para definir el rango de valores a ser manipulado.
Agregado de nombres de variables que representan tipos de datos	• La descripción de nombres de variables con notación húngara permite que las inconsistencias en los tipos de datos de las variables puedan ser identificadas al escribir programas.

Capítulo 5 Ramificación condicional

Los programas de control también contienen secciones de código en las cuales el procesamiento real cambia según las condiciones especificadas.
Este capítulo describe la ramificación condicional.

5.1 Ramificación condicional (IF)

Se utilizan las declaraciones IF para la ramificación condicional. Las declaraciones se describen a continuación.

```
IF conditional expression THEN
    Execution statement;        (* La declaración es ejecutada si la expresión condicional se satisface. *)
END_IF;                         (* END_IF; debe ser ubicado al final de las declaraciones IF. *)
```

En este programa de ejemplo, la declaración es ejecutada cuando la expresión condicional se satisface. La declaración no es ejecutada cuando la expresión condicional no se satisface.

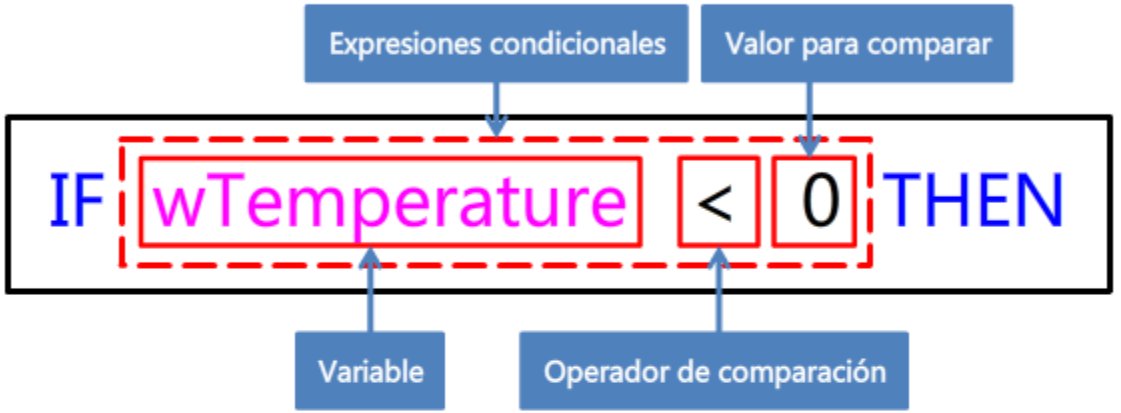
La siguiente figura ilustra el flujo de operaciones en este programa de ejemplo.

El siguiente ejemplo ilustra la ramificación del programa mediante la comparación de valores de variables.
En el programa de ejemplo, el calefactor se enciende cuando la temperatura en el panel de control es menor a 0 grados.

```
IF wTemperature < 0 THEN
  bHeater := 1;  (* El calefactor se enciende cuando la temperatura en el panel de control es menor a 0 grados. *)
END_IF;
```

5.1.1 Escritura de expresiones condicionales

La página anterior describe una expresión condicional de "wTemperature < 0", que significa "cuando el valor de la variable wTemperature es menor a 0".
Al igual que esta expresión, las expresiones condicionales usan operadores de comparación para representar la relación entre las variables y los valores para comparar.



En los lados izquierdo y derecho de un operador de comparación, los valores están escritos como variables o constantes para la comparación.

Además de comparar las variables y constantes, se pueden escribir expresiones condicionales para comparar variables y realizar operaciones lógicas de resultados de comparación o variables de tipo bit.

Comparación de variables

- uValue1 <= uValue2

La operación lógica para dos resultados de comparación

- (10 < uValue) AND (uValue <= 50)

La operación lógica para dos variables de tipo bit

- bSwitch0 OR bSwitch1

La siguiente tabla muestra los tipos de operadores de comparación.

Operador	Significado
>	Mayor a
<	Menor a
>=	Mayor o igual a
<=	Menor o igual a
=	Es igual a
<>	No es igual a

5.1.2 Ramificación excepcional para la declaración IF (ELSE)

Las declaraciones simples IF (ver 5.1) se utilizan para ejecutar una declaración cuando la expresión condicional se satisface. Para ejecutar una declaración diferente cuando la expresión condicional no se satisface, se usa una declaración ELSE.

```
IF conditional expression THEN
  Execution statement 1;          (* La declaración 1 se ejecuta si la expresión condicional se satisface. *)
  ELSE
  Execution statement 2;          (* La declaración 2 se ejecuta si la expresión condicional no se satisface. *)
END_IF;
```

La siguiente figura ilustra el flujo de operaciones cuando se usa una declaración ELSE.

El siguiente programa de ejemplo ejecuta diferentes declaraciones si la condición se satisface o no.

El programa de ejemplo en 5.1 tiene la desventaja de que el calefactor sigue subiendo la temperatura incluso después de alcanzar los 0 grados. Sin embargo, el siguiente programa apaga el calefactor cuando "wTemperature" supera los 0 grados.

```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Se enciende el calefactor cuando la temperatura cae por debajo de los 0 grados centígrados. *)
  ELSE
  bHeater := 0;  (* Se apaga el calefactor cuando la temperatura alcanza o excede los 0 grados centígrados *)
END_IF;
```

5.1.3 Ramificación adicional para la declaración IF (ELSIF)

Las declaraciones IF se utilizan para ejecutar una declaración diferente cuando la expresión condicional no se satisface.
Se puede agregar otra ramificación condicional con el uso de declaraciones ELSIF, que significa que si la expresión condicional previa no se satisface, entonces otra expresión condicional será verificada.

```
IF Conditional expression 1 THEN
  Execution statement 1;      (* La declaración 1 se ejecuta si la expresión condicional 1 se satisface. *)
  ELSIF Conditional expression 2 THEN
  Execution statement 2;      (* La declaración 2 se ejecuta si la expresión condicional 1 no se satisface y si la expresión condicional 2 se satisface. *)
  ELSE
  Execution statement 3;      (* La declaración 3 se ejecuta si las expresiones condicionales 1 y 2 no se satisfacen. *)
END_IF;
```

La siguiente figura ilustra el flujo de operaciones cuando se usa una declaración ELSEIF.

La declaración ELSIF se agrega al ejemplo de programa ilustrado en 5.1.2 para lidiar con el caso cuando la temperatura supera los 40 grados.

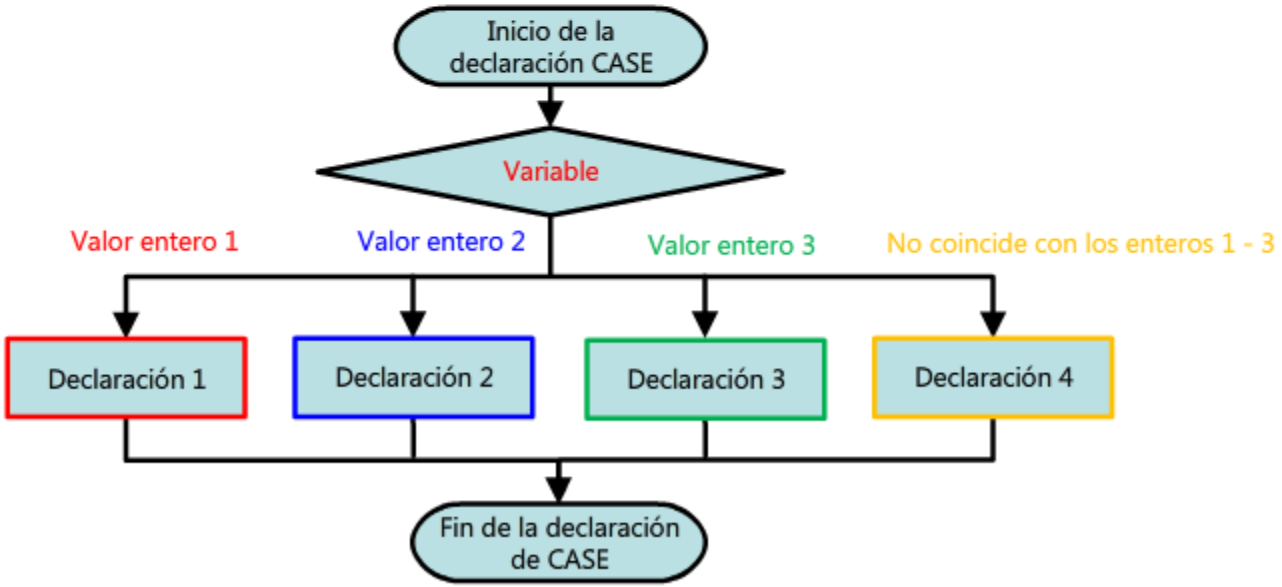
```
IF wTemperature < 0 THEN
  bHeater := 1; (* Se enciende el calefactor cuando la temperatura cae por debajo de los 0
                   grados centígrados. *)
  bCooler := 0; (* Se apaga cuando la temperatura es menor a los 0 grados centígrados. *)
  ELSIF 40 < wTemperature THEN
  bHeater := 0; (* Apaga el calefactor cuando la temperatura excede los 40 grados centígrados. *)
  bCooler := 1; (* Enciende el enfriador cuando la temperatura excede los 40 grados centígrados. *)
  ELSE
  bHeater := 0; (* Apaga el calefactor si ninguna de las condiciones previas se satisfacen. *)
  bCooler := 0; (* Apaga el enfriador si ninguna de las condiciones previas se satisfacen. *)
END_IF;
```

5.2 Ramificación condicional según los valores enteros (CASE)

Las declaraciones IF se utilizan para la ramificación según si las expresiones condicionales se satisfacen.
Las declaraciones CASE se utilizan para la ramificación según los valores enteros.
La siguiente figura ilustra cómo se escribe una declaración CASE.

```
CASE Variable OF
    Integer value 1: Execution statement 1;   (* La declaración 1 se ejecuta cuando la variable coincide con el valor entero 1. *)
    Integer value 2: Execution statement 2;   (* La declaración 2 se ejecuta cuando la variable coincide con el valor entero 2. *)
    Integer value 3: Execution statement 3;   (* La declaración 3 se ejecuta cuando la variable coincide con el valor entero 3. *)
    ELSE Execution statement 4;               (* La declaración 4 se ejecuta si la variable no coincide con ninguno de los valores enteros. *)
END_CASE;                                     (* "END_CASE;" debe ser ubicado al final de la declaración CASE. *)
```

La siguiente figura ilustra el flujo de operaciones cuando se usa una declaración CASE.



5.2.1 Programa de ejemplo para la declaración CASE

La ejecución de la declaración CASE se escribe con el uso del ejemplo de la operación del programa.

Haga clic en para pasar a la siguiente página.
Para ver nuevamente la animación, haga clic en el botón "Reproducir".

Reproducir


```
CASE wWeight OF
  0..20:    uSize := 1;
  21..30:   uSize := 2;
  31..40:   uSize := 3;
  ELSE      uSize := 4;
END_CASE;
```

Peso	uSize	Grado
0 a 20 kg	1	M
21 a 30 kg	2	L
31 a 40 kg	3	XL
41 kg y más	4	Oversize

5.3 Resumen

Los temas de este capítulo son:

- Ramificación condicional con declaraciones IF
- Escritura de expresiones condicionales
- Ramificación condicional según los valores enteros (declaración CASE)

Puntos importantes a tener en cuenta:

Declaración IF	• Con una declaración IF, el programa se ramifica cuando una expresión condicional se satisface. • Se utiliza una declaración ELSE para la ramificación cuando no se satisface la expresión condicional. • Se usa una declaración ELSIF para agregar otra ramificación cuando no se satisface la expresión condicional en la declaración IF.
Expresión condicional	• Las expresiones condicionales representan la relación entre las variables y los valores para comparar mediante los operadores de comparación.
Declaraciones CASE	• Las declaraciones CASE se utilizan para la ramificación según los valores enteros.

Capítulo 6 Almacenamiento y manipulación de datos

Además de las aplicaciones de control de E/S, hoy en día los controladores programables se utilizan para procesar grandes cantidades de datos como el núcleo de los sistemas de producción.
Para procesar grandes cantidades de datos, la información debe ser almacenada y luego leída según sea necesario.
Este capítulo describe los pasos para escribir programas concisos para el almacenamiento y procesamiento de datos.

- Los arreglos se utilizan para secuenciar y organizar las variables.
- Las estructuras de datos se utilizan para organizar las variables relacionadas.
- Los programas de procesamiento de bucles procesan de manera efectiva los arreglos usando las declaraciones FOR.

Los programas concisos para almacenar y manipular datos pueden ser creados mediante el uso de arreglos, estructuras de datos y declaraciones FOR.

6.1 Secuenciación y almacenamiento de datos (Arreglo)

Con el uso de los arreglos, se pueden manipular valores múltiples con una variable.
En el siguiente ejemplo, los datos de volumen de producción en una planta fabricadora de automóviles son almacenados por el país de destino.

Destino	País A	País B	País C
Volumen de producción	35	75	65

Los datos de volumen de producción por el país de destino están asignados a una variable. Sin el uso de arreglos, se debe crear una variable para cada destino.
Con el uso de arreglos, sin embargo, el volumen de producción para destinos múltiples puede grabarse y asignarse en una variable.

Sin uso de arreglo

uProductionA
uProductionB
uProductionC

Con uso de arreglo

uProduction

Las variables individuales en el arreglo están especificadas con el uso de números de elementos. Los números de elementos comienzan en cero [0].

uProduction [0]

Nombre de la variable

Número del elemento

Destino (fila)

País A	[0]	35
País B	[1]	75
País C	[2]	65

En el siguiente ejemplo de programa, la variable del volumen de producción planeado para el País A está asignado.

```
uShowProductionPlan := uProduction[0];
(* Especifica el número de elemento para el país A. *)
```

6.1.1 Arreglo de matrices

A continuación los datos del color de pintura son utilizados además de los datos de destino.

Destino	País A			País B			País C		
Color de la pintura									
Volumen de producción	10	5	20	15	40	20	25	30	10
	35 en total			75 en total			65 en total		

Como se ilustra en la siguiente tabla, los datos pueden ser separados y almacenados por color de pintura (columna) para cada país de destino (fila).

Color de pintura (columna)

Destino (fila)	Rojo	Amarillo	Azul
País A	[0,0] 10	[0,1] 5	[0,2] 20
País B	[1,0] 15	[1,1] 40	[1,2] 20
País C	[2,0] 25	[2,1] 30	[2,2] 10

Número de elemento que representa el destino

Número de elemento que representa el color de la pintura

Variable de arreglos (arreglo de matrices)

```
uProduction[1,1]
```

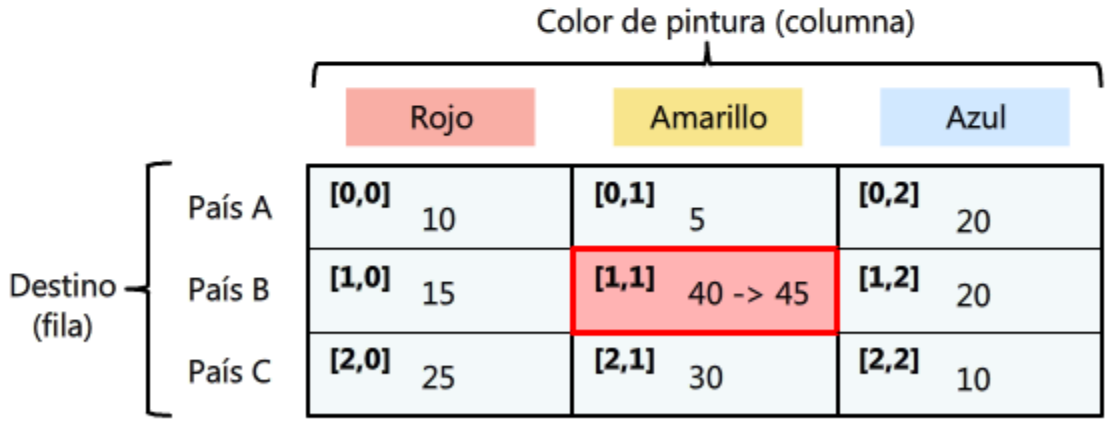
Los arreglos que organizan la información en filas y columnas de esta manera son conocidos como arreglos de matrices. Los números de elementos que representan las filas y las columnas están separados por comas.

6.1.2 Asignación de arreglo de matrices

Con el uso de arreglos de matriz, el siguiente programa de ejemplo asigna el número de autos a fabricarse urgentemente además del volumen de producción planeado de automóviles amarillos para el País B.

```
uAdditionalProduction := 5;
uProduction[1,1] := uProduction[1,1] + uAdditionalProduction;
(* Agrega el monto adicional de producción (5 unidades) al volumen de producción planeado inicialmente. *)
```

Destino	País A			País B			País C		
Color de la pintura									
Volumen de producción	10	5	20	15	40	20	25	30	10
	35 en total			75 en total			65 en total		



6.1.3 Procesamiento de información almacenada en los arreglos de matrices

Con los arreglos de matriz, el siguiente programa de ejemplo calcula el volumen de producción total planeado para todos los colores de pintura para el País C y asigna el valor a la variable.

```
uProductionToday := uProduction[2,0] + uProduction[2,1] +uProduction[2,2];
(* Calcula el volumen de producción total planeado para hoy para todos los colores de pintura para el País C y asigna el valor a "uProductionToday". *)
```

Destino	País A			País B			País C		
Color de la pintura									
Volumen de producción	10	5	20	15	45	20	25	30	10
	35 en total			80 en total			65 en total		

Color de pintura (columna)

Destino (fila)	Rojo	Amarillo	Azul
País A	[0,0] 10	[0,1] 5	[0,2] 20
País B	[1,0] 15	[1,1] 45	[1,2] 20
País C	[2,0] 25	[2,1] 30	[2,2] 10

6.2 Bucles (FOR)

El programa de ejemplo de la página anterior (el volumen de producción planeada para hoy es asignado) se muestra a continuación nuevamente.

```
uProductionToday := uProduction[2,0] + uProduction[2,1] + uProduction[2,2];
```

Con este ejemplo de programa, cuando el número de colores de pintura aumenta, se agregarán más variables. Luego, la expresión se hace más larga, lo que dificulta la lectura.

```
uProductionToday := uProduction[2,0] + uProduction[2,1] + uProduction[2,2]
                  + uProduction[2,3] + uProduction[2,4] + uProduction[2,5] ...
```

En este caso, las declaraciones de bucles pueden usarse para crear el código de limpiador.

Las declaraciones de bucle incluyen FOR, WHILE y REPEAT. Este curso cubre las declaraciones FOR.

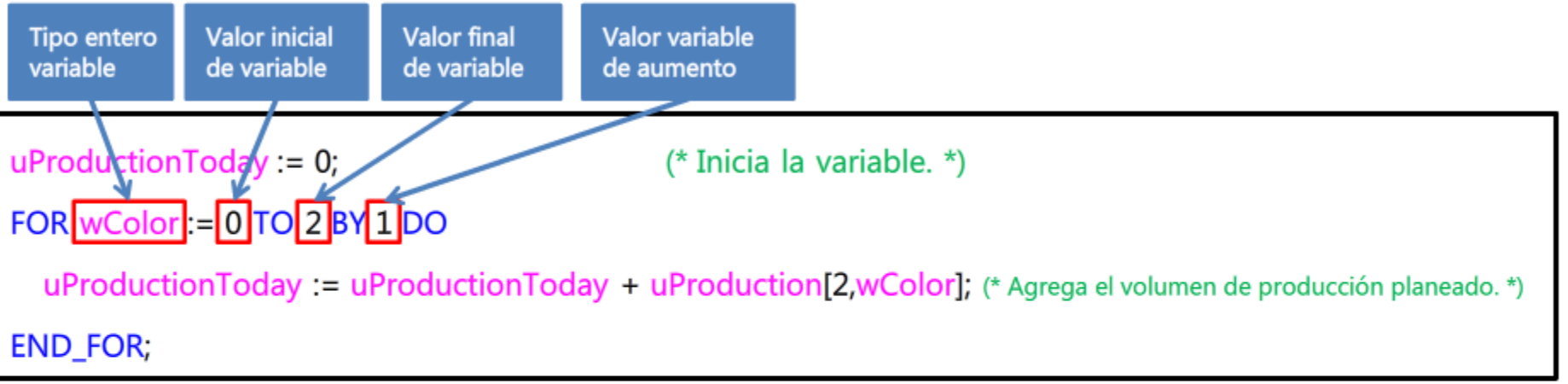
Las declaraciones FOR se describen a continuación.

```
FOR variable := initial value TO final value BY increments DO
   Execution statement; (* La declaración se ejecuta en un bucle hasta que la variable alcanza el valor final. *)
END_FOR;                (* END_FOR; debe ser ubicado al final de las declaraciones FOR.  *)
```

La declaración es repetida hasta que se alcanza el valor final de la variable y se ejecuta el código "END_FOR;".

6.2 Bucles (FOR)

Con una declaración FOR, el siguiente ejemplo de programa obtiene el volumen de producción planeado para todos los colores de pintura para el País C.




```
uProductionToday := 0;                                    (* Inicia la variable. *)
FOR wColor:=0 TO 2 BY 1 DO
   uProductionToday := uProductionToday + uProduction[2,wColor]; (* Agrega el volumen de producción planeado. *)
END_FOR;
```

Con la declaración FOR, la variable "wColor" aumenta de a uno comenzando desde el valor inicial de cero y la declaración se repite hasta que la variable alcanza el valor final de dos.
La variable "wColor" está especificada como el segundo número de elemento en el arreglo "uProduction" descrito en la declaración de ejecución.
El valor de la variable "wColor" aumenta cada vez que se repite la declaración. El volumen de producción planeado para cada color de pintura es agregado cada vez para obtener el total.

El programa de ejemplo se ejecuta tres veces en un bucle. (Primera vez: rojo [0] => Segunda vez: amarillo [1] => Tercera vez: azul [2])

La operación de este programa se ilustra en la siguiente página.

6.2 Bucles (FOR)

La ejecución de la declaración FOR se describe con la operación del ejemplo de programa.

Arreglo del volumen de producción estimado

	Rojo	Amarillo	Azul
País A	[0,0] 10	[0,1] 5	[0,2] 20
País B	[1,0] 15	[1,1] 45	[1,2] 20
País C	[2,0] 25	[2,1] 30	[2,2] 10

Haga clic en  para pasar a la siguiente página.
Para ver nuevamente la animación, haga clic en el botón "Reproducir".

Reproducir

Number of repetition of the loop: 3

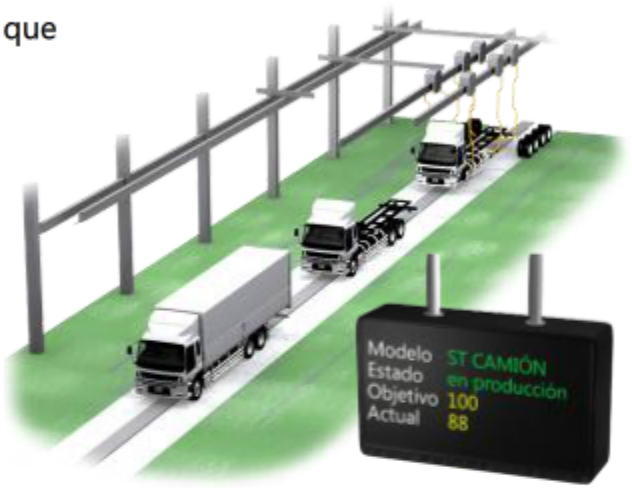
```
uProductionToday := 0;

FOR wColor := 0 TO 2 BY 1 DO
        2
   uProductionToday := uProductionToday + uProduction[2,wColor];
                                          65
END_FOR;
```

6.3 Almacenamiento de datos relacionados (Estructura)

Una estructura permite que un nombre de variable represente variables múltiples relacionadas.
En el siguiente ejemplo, el estado de una línea de producción de automóviles se muestra en Andon (tablero de visualización).

La siguiente tabla muestra los nombres de variables, los valores y los tipos de datos que corresponden a los puntos mostrados.

Elemento	Nombre de la variable	Valor	Tipos de datos variable
Modelo	sModel	'ST CAMIÓN'	Cadena de texto
Estado	bStatus	'en producción'	Tipo de bit
Objetivo de volumen de producción para hoy	uPlanQty	'100'	Palabra de tipo entero (sin firmar)
Volumen de producción actual	uActualQty	'88'	Palabra de tipo entero (sin firmar)



Si no se utiliza una estructura, se deberán cambiar los nombres de las variables para cada línea cuando existan varias líneas de producción.
El siguiente muestra ejemplos de nombres de variables por la línea de producción.

Primera línea de producción

s1stLineModel
b1stLineStatus
u1stLinePlanQty
u1stLineActualQty

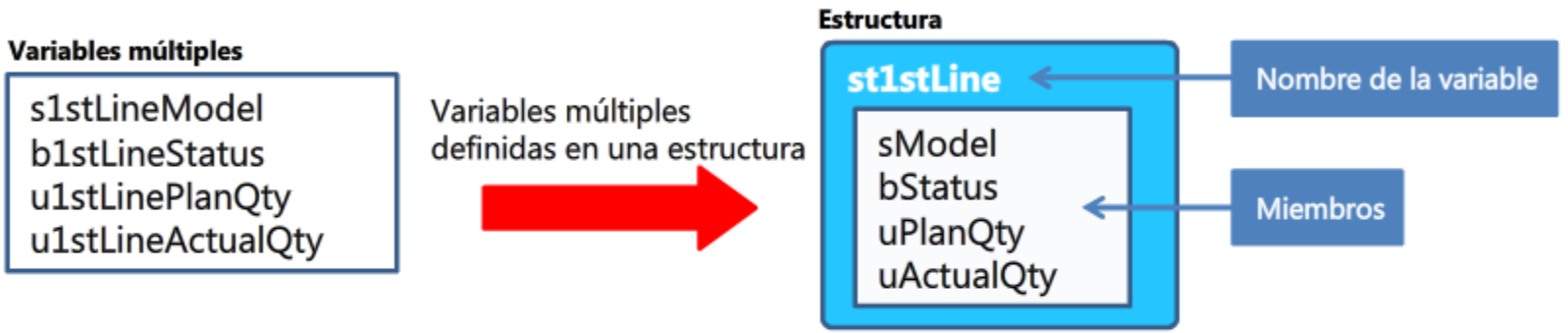
Segunda línea de producción

s2ndLineModel
b2ndLineStatus
u2ndLinePlanQty
u2ndLineActualQty

Cuando aumenta el número de líneas de producción, el número de variables a manipularse también aumenta. Entonces, el programa se hace más largo, lo que dificulta la lectura.

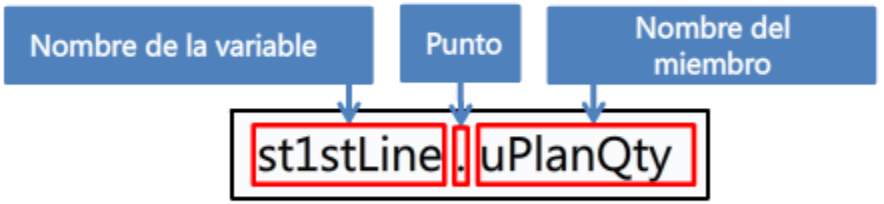
6.3 Almacenamiento de datos relacionados (Estructura)

El uso de estructuras permite que un nombre de variable represente múltiples variables relacionadas a una línea de producción. De igual manera, las estructuras se utilizan para organizar, almacenar y manipular los datos en un lote para condiciones y especificaciones de objetos físicos tales como dispositivos y piezas de trabajo.



Structure variable (variable estructural) contiene el prefijo "st" para representar que esta es una estructura.
Las variables individuales definidas por la estructura son conocidas como miembros. Los tipos de datos de cada miembro pueden ser diferentes.

Cada miembro de los arreglos de estructura puede ser especificado luego del número de elemento con el uso de un punto antes del nombre.

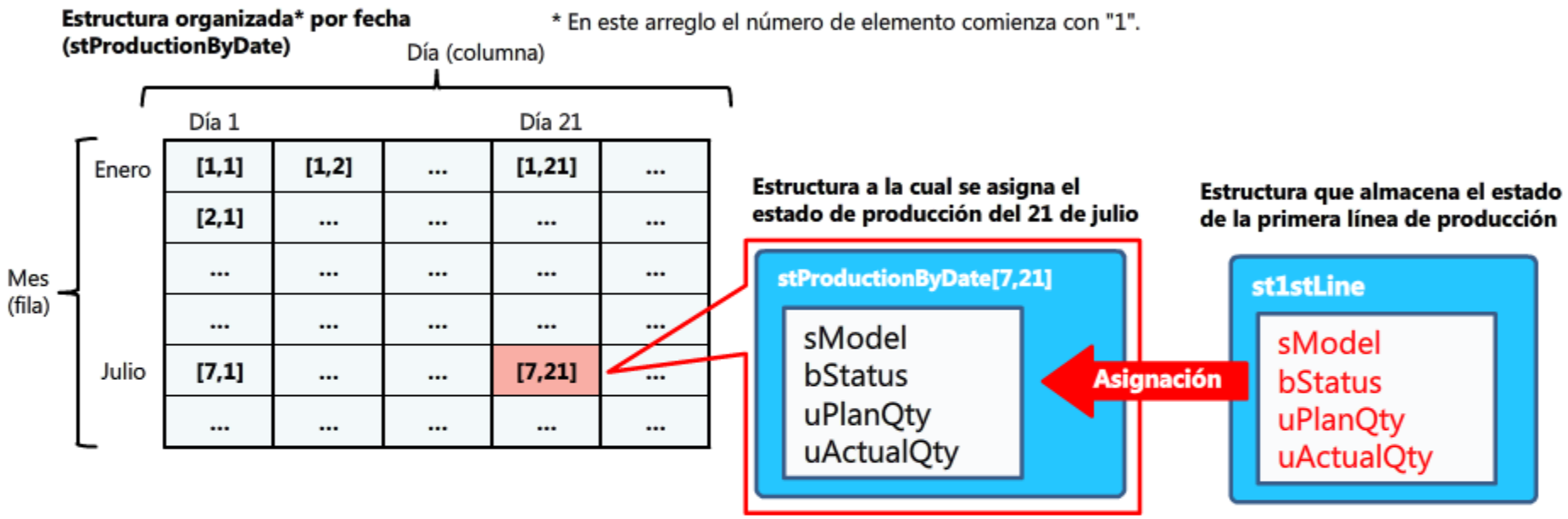


En el siguiente programa de ejemplo, se asigna una constante a un miembro de la variable estructural para la primera línea de producción.

```
st1stLine.uPlanQty := 150;
(* Define el objetivo de producción para hoy de la primera línea de producción a 150. *)
```

6.3.1 Almacenamiento de arreglos de estructuras

Las estructuras pueden crearse como arreglos.
En el siguiente ejemplo, el estado de producción se almacena por fecha.

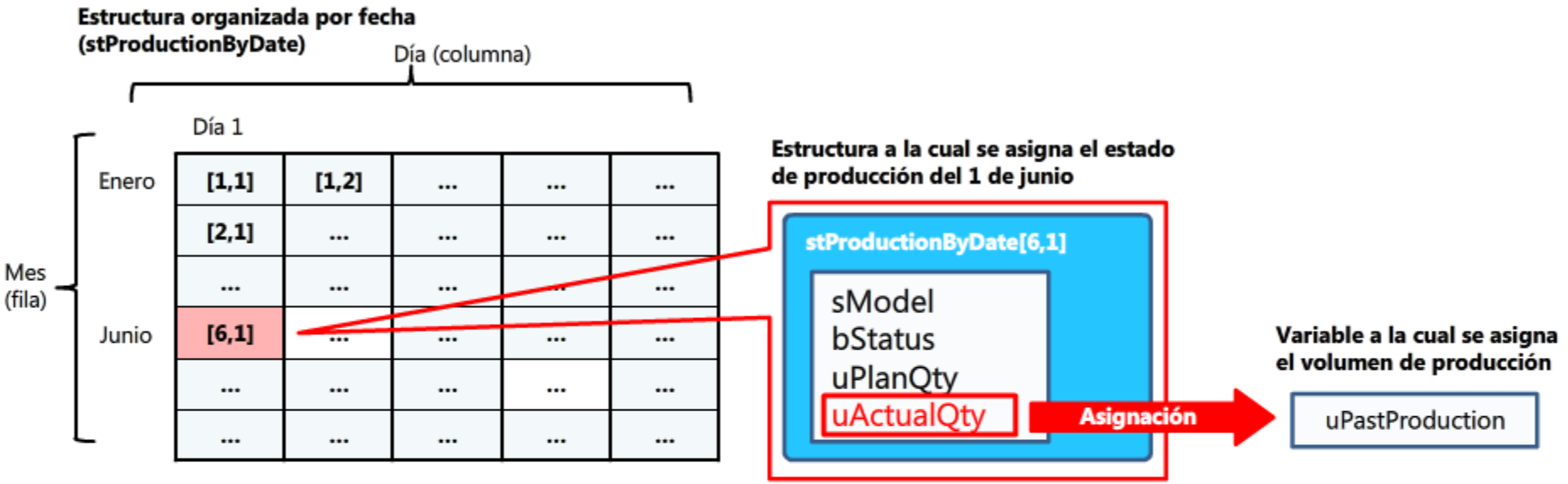



```
stProductionByDate[7,21] := st1stLine;
(* El estado de producción del 21 de julio es almacenado en la estructura ordenada por fecha (stProductionByDate). *)
```

De igual manera, los miembros no necesitan especificarse individualmente para la asignación de estructura.

6.3.2 Lectura de arreglos de estructuras

En el siguiente ejemplo, se lee el volumen de producción de una estructura ordenada por fecha y luego se asigna a una variable.




```
uPastProduction := stProductionByDate[6,1].uActualQty;
(* Asigna el volumen de producción del 1º de junio a la variable uPastProduction. *)
```

Se puede especificar cada miembro de los arreglos de estructura con un punto (.) y un nombre de miembro al número de elemento del arreglo.

6.4 Resumen

Los temas de este capítulo son:

- Descripción general y uso de los arreglos
- Procesamiento de bucles con declaraciones FOR
- Descripción general y uso de las estructuras

Puntos importantes a tener en cuenta:

Arreglo	• Una variable puede manipular valores múltiples con el uso de arreglos. • Las variables individuales en arreglos son especificadas por números de elementos agregados al final de los nombres de las variables.
Declaraciones FOR	• Las declaraciones de bucles se usan en programas cuando se desea una operación repetitiva. • Las declaraciones FOR se utilizan para repetir la operación hasta que las condiciones para el final de la operación del bucle se satisfacen. Las declaraciones antes de la declaración "END_FOR;" se ejecutan de manera repetitiva.
Estructura	• Las estructuras permiten que un nombre de variable represente múltiples variables relacionadas. Las estructuras pueden incluir variables de diferentes tipos de datos. • Las variables o miembros individuales definidos en estructuras se especifican con un punto y el nombre del miembro luego de un nombre de variable de estructura.

Capítulo 7 Manipulación de datos de cadenas

En algunos casos, los controladores programables usan datos de cadena para enviar comandos o recibir retroalimentación de dispositivos conectados como lectores de códigos de barras, controladores de temperatura o balanzas electrónicas. Para estos fines, es necesario añadir o extraer los datos de cadena según sea necesario.

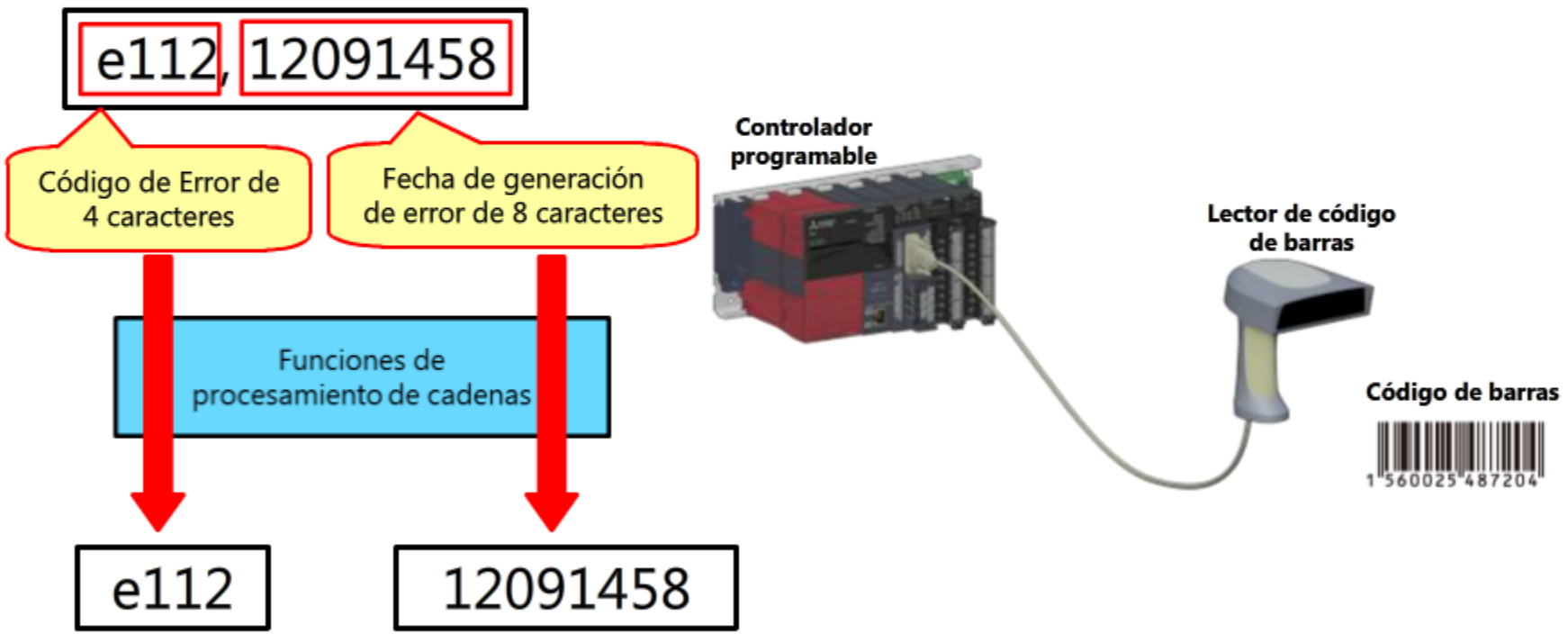
Este capítulo describe los pasos para la manipulación de datos de cadenas.

7.1 Ejemplo de manipulación de datos de cadenas

Como ejemplo de procesamiento de cadenas, el ejemplo ilustra una situación en la que se leen los datos de un lector de códigos de barras.
Las funciones (un tipo de instrucción) se utilizan para procesar cadenas.

Como se ilustra a continuación, las cadenas leídas por el lector de códigos de barras contienen un código de error de longitud fija de 4 caracteres e información del minuto, la hora, el día y el mes con una longitud fija de 8 caracteres.
El ejemplo de programa de procesamiento de cadenas será descrito con este sistema.

Datos de cadena de ejemplo leídos de un lector de códigos de barras



Se extrajo un código de error.
7.3 Extracción de cadenas (LEFT)

Se extrajo la fecha y la hora en que ocurrió el error (14:58, 9 de diciembre).
7.4 Extracción de cadenas (MID)

7.2 Asignación de cadenas

Antes de explicar los pasos para extraer cadenas, esta sección describe los tipos de datos para cadenas.

Los tipos de datos para cadenas que pueden usarse con controladores programables se muestran en la siguiente tabla.

Tipo de datos	Tipo de carácter puede ser procesado	Prefijos de notación húngara	Expansión del prefijo
Cadena	Las cadenas de caracteres alfanuméricos y números (ASCII) o japonés (Shift-JIS)	s	**s**tring (cadena)
Cadena [Unicode]	Cadenas de idiomas y símbolos diferentes	ws	**w**ide **s**tring (cadena ancha)

El tipo de cadena a usarse depende del dispositivo que se conecta al controlador programable o el idioma correspondiente.
Este capítulo describe los tipos diferentes de cadenas de texto.

Cuando se asigna un tipo de cadena a una variable de cadena, encierre la cadena con marcas de cita simples (').

```
sDefault := 'e112,12091458';  (* Asignación de cadenas *)
```

7.3 Extracción de cadenas (LEFT)

El código de error "e112" se extrae de la variable de cadena "sBarcodeData" que contiene la cadena "e112,12091458".

Nombre de la variable	Cadena almacenada
sBarcodeData	e112, 12091458

La función LEFT (izquierda) solo extrae el número especificado de caracteres que comienza del lado izquierdo de la cadena de entrada.
El siguiente ilustra un programa de ejemplo.

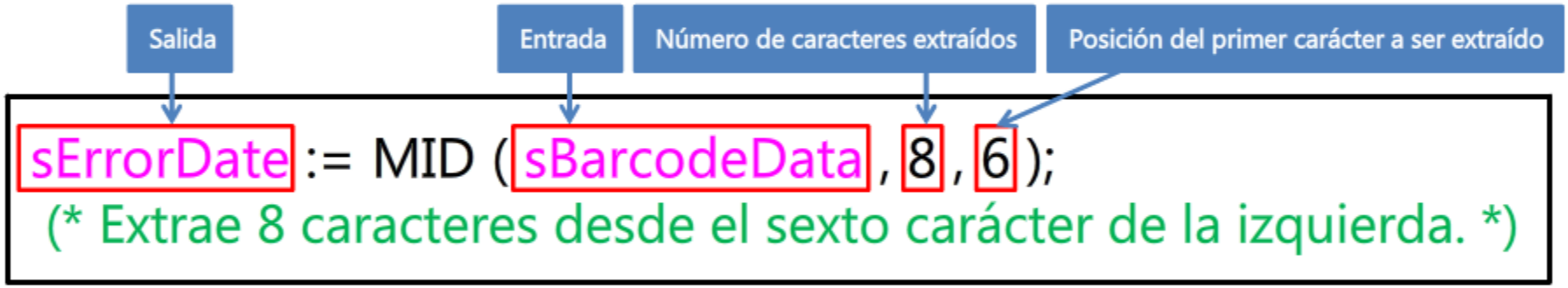
Se extraen cuatro caracteres de la izquierda. El valor "e112", que es la cadena que representa el código de error, se asigna al lado izquierdo.

7.4 Extracción de cadenas (MID)

La hora de generación del error "12091458" se extrae de la variable de cadena "sBarcodeData" que contiene la cadena "e112,12091458".

Nombre de la variable	Cadena almacenada
sBarcodeData	e112,12091458

La función MID (medio) extrae el número especificado de caracteres de la posición de comienzo especificada de la cadena de entrada.
El siguiente ilustra un programa de ejemplo.



En este ejemplo, se extrae una cadena de 8 caracteres del sexto carácter. El valor "12091458", que es la cadena que representa la hora en que ocurrió el error, se asigna al lado izquierdo.

7.5 Resumen

Los temas de este capítulo son:

- Métodos de asignación de cadenas a variables de cadenas
- Funciones para extraer cadenas (LEFT y MID)

Puntos importantes a tener en cuenta:

Asignación de cadenas	• Para asignar una cadena a una variable de cadena, encierre la cadena con marcas de cita simples ('). • Puede usar el tipo de cadena común o el tipo de cadena [Unicode] de acuerdo con el dispositivo conectado al controlador programable o el idioma correspondiente.
Funciones para la manipulación de cadenas	• Las funciones se utilizan para la manipulación de cadenas.

7.6 Resumen del curso

Este capítulo cubrió los conceptos básicos sobre la creación de programas en ST.
Con esto concluye este curso de e-learning.

Los programas de ST son creados con el software de ingeniería de MELSOFT.
Para obtener detalles sobre pasos especificados como el ingreso de datos, la edición, el guardado y la compilación de programas con el software de ingeniería de MELSOFT, consulte aquí.

- Mitsubishi FA e-Learning Course "MELSOFT GX Works3 (Structured Text)" (MELSOFT GX Works3 (Texto estructurado)) (disponible muy pronto)
- Manual de operación de su software de ingeniería MELSOFT

Para obtener más información sobre ST, consulte aquí.

- Guía de programación de su controlador programable

Para obtener más información sobre instrucciones y funciones de su aplicación, consulte aquí.

- Manual de programación de su controlador programable

Prueba Prueba final

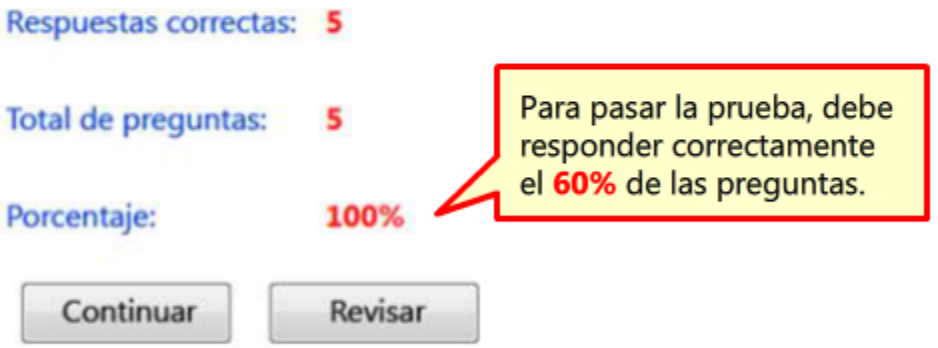
Ahora que ha completado todas las lecciones del curso Conceptos básicos de la programación (Texto estructurado) está listo para tomar la prueba final. Si no tiene claro alguno de los temas cubiertos, tome esta oportunidad para revisar esos temas.
Hay un total de 12 preguntas (20 puntos) en esta Prueba Final.
Puede tomar la prueba final las veces que desee.

Cómo calificar la prueba

Luego de seleccionar la respuesta, asegúrese de hacer clic en el botón **Responder**. Su responder se perderá si no hace clic en el botón Responder. (Se considerará como pregunta sin responder.)

Resultados de la calificación

El número de respuestas correctas, el número de preguntas, el porcentaje de respuestas correctas, y el resultado sobre si aprobó o no aparecerá en la página de calificación.



- Haga clic en el botón **Continuar** para salir de la prueba.
- Haga clic en el botón **Revisar** para revisar la prueba. (Verificar la respuesta correcta)
- Haga clic en el botón **Volver a intentar** para tomar la prueba nuevamente.

Prueba final 1

Características del texto estructurado (ST)
Seleccione la descripción incorrecta de ST.

- ST es fácil de aprender para aquellos con experiencia en la escritura de programas de lenguaje C o BASIC.
- Los cálculos como la suma y la resta pueden ser escritos como expresiones matemáticas generalmente utilizadas.
- Los símbolos para los contactos y las bobinas son usados para crear un programa que aparente ser un circuito eléctrico.
- ST es ideal para la manipulación de datos.

Responder Volver

Prueba final 2

Principios básicos de ST

Seleccione la declaración correcta escrita en ST.

- uProduction = 15
- uProduction := 15:
- uProduction := 15;
- uProduction = 15;

Responder Volver

Prueba final 3

Descripción de comentarios
Seleccione el comentario correcto escrito en ST.

- ' Asigna un valor de 1 a la variable.
- (* Asigna un valor de 1 a la variable. *)
- { Asigna un valor de 1 a la variable. }
- <!-- Asigna un valor de 1 a la variable. -->

Responder Volver

Prueba Prueba final 4

Secuencia de ejecución de programa ST
*El valor inicial de “uTotalProduction” es “100”. El valor de la variable “uTotalProduction” será “101” después de que se procese el siguiente programa de ejemplo. Seleccione el estado correcto de “uTotalProduction” después de que transcurran algunos segundos.

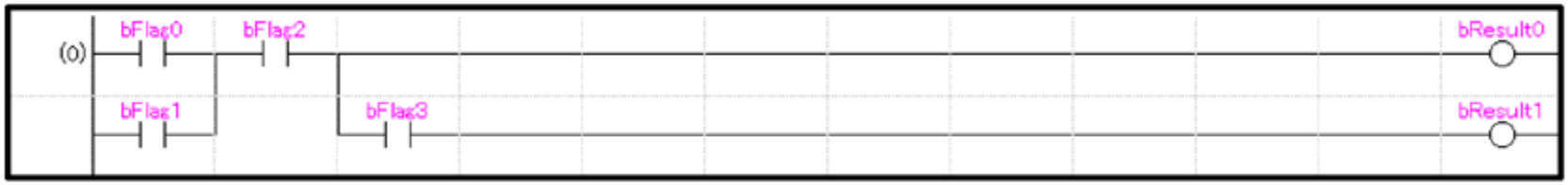
```
uTotalProduction := uTotalProduction + 1;
```

- ( ) El valor permanece en 101.
- ( ) El valor sigue cambiando.

Responder Volver

Prueba Prueba final 5

Combinación de condiciones múltiples
Seleccione el ejemplo de programa ST correcto que represente la misma operación que el siguiente ejemplo de programa en LD.



- bResult0 := (bResult0 OR bFlag1) AND bFlag2;
  bResult1 := bResult0 AND bFlag3;

- bResult0 := (bFlag0 OR bFlag2) AND bFlag1;
  bResult1 := bResult0 AND bFlag3;

Responder Volver

Prueba Prueba final 6

Descripción de declaraciones IF en ST
La siguiente operación es ejecutada por el siguiente programa de ejemplo.
•Si la temperatura cae unos 5 grados o más, se enciende el calefactor y se apaga el enfriador.
•Si la temperatura supera los 50 grados o más, se apaga el calefactor y se enciende el enfriador.
•Si la temperatura no responde a las declaraciones anteriores, tanto el calefactor como el enfriador se apagan.

*Nombres de variable: Temperatura (wTemperature), calefactor (bHeater) y enfriador (bCooler)
Seleccione la opción correcta para cada sección en blanco del programa de ejemplo.

```
IF wTemperature [Q1] 5 [Q2]
    bHeater := 1;
    bCooler := 0;
[Q3] 50 [Q4] wTemperature [Q2]
    bHeater := 0;
    bCooler := 1;
[Q5]
    bHeater := 0;
    bCooler := 0;
END_IF;
```

Q1 --Select--

Q2 --Select--

Q3 --Select--

Q4 --Select--

Q5 --Select--

Responder Volver

Prueba Prueba final 7

Declaraciones CASE

Seleccione la opción correcta para cada uno (Q1 a Q5) de la siguiente descripción de declaraciones CASE.

Las declaraciones CASE son usadas para la ramificación según el valor de (Q1).
En el siguiente programa de ejemplo, cuando el valor de (Q2) es 25, a la variable (Q3) se le asigna un valor de (Q4). Cuando el valor de la variable (Q2) no es igual a 10, 25 u 8, a la variable (Q3) se le asigna un valor de (Q5).

```
CASE wCode OF
    10:     uLane := 1;
    25:     uLane := 2;
    8:      uLane := 3;
   ELSE     uLane := 4;
END_CASE;
```

Q1 --Select--

Q2 --Select--

Q3 --Select--

Q4 --Select--

Q5 --Select--

Responder Volver

Prueba Prueba final 8

Arreglos ST y declaraciones repetitivas
El siguiente programa de ejemplo da un total del volumen de producción planeado de todos modelos destinos para el País Y y luego asigna este valor a una variable. Seleccione la sección del arreglo que es leído luego de que se ejecute 3 veces la declaración en un bucle.

```
uProductionToday := 0;
FOR wCarModel := 0 TO 3 BY 1 DO
  uProductionToday := uProductionToday + uProduction[1,wCarModel];
END_FOR;
```

Arreglo usado para almacenar el número estimado de unidades producidas por modelo y destino (uProduction)

Destino (fila)	Modelo 1	Modelo 2	Modelo 3	Modelo 4
PaísX	[0,0]	[0,1]	[0,2] C	[0,3]
País Y	[1.0]	[1.1] A	[1.2] D	[1.3] E
País Z	[2,0]	[2,1] B	[2,2]	[2,3]

Modelo (columna)

- A
- B
- C
- D

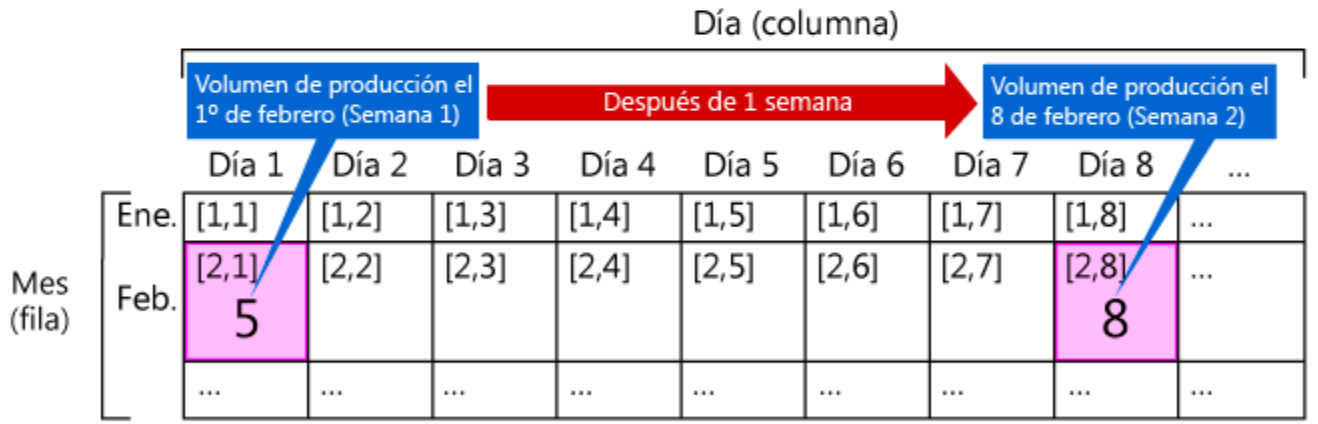
Responder Volver

Prueba — Prueba final 9

Arreglos ST y declaraciones repetitivas
El siguiente programa de ejemplo obtiene el volumen total de producción en los mismos días de la semana. El total en 4 semanas se obtiene del arreglo que almacena el volumen de producción por día. Seleccione la figura correcta para el programa de ejemplo.

```
uTotalProduction := 0;
FOR wOnceAWeek := 1 TO ■ BY 7 DO
  uTotalProduction := uTotalProduction + uProductionByDate[2,wOnceAWeek];
END_FOR;
(* Extrae y obtiene el total del volumen de producción en los mismos días de la semana durante 4 semanas desde el 1º de febrero. *)
```

Arreglo que almacena el volumen de producción por día (uProductionByDate)



- 22
- 21
- 4
- 28

Responder | Volver

Prueba Prueba final 10

Características de las estructuras en ST
Seleccione una descripción incorrecta de estructuras.

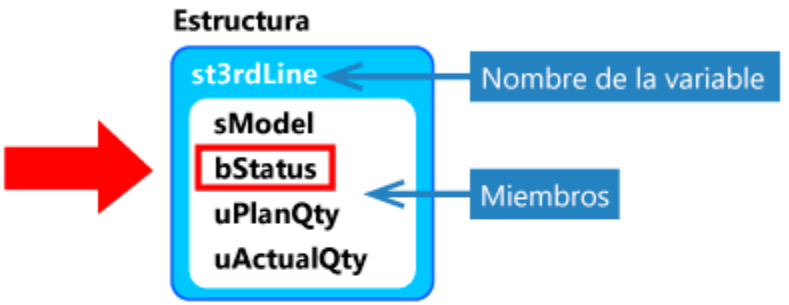
- Las estructuras son usadas para organizar y almacenar datos en dispositivos mediante condiciones como el estado y las especificaciones.
- Los programas que procesan grandes cantidades de datos pueden ser escritos de manera concisa con el uso de estructuras.
- Todos los miembros definidos en la estructura deben tener el mismo tipo de datos.
- Se pueden asignar los valores a los miembros en la misma estructura sin ser especificados individualmente.

Responder Volver

Prueba Prueba final 11

Especificación de miembros para estructuras en ST
La siguiente estructura organiza las variables relacionadas a una línea de producción de automóviles.
Seleccione la descripción correcta para la especificación del miembro "bStatus" en esta estructura.

Parámetro	Nombre de la variable
Modelo	sModel
Estado	bStatus
Objetivo de producción para el día actual	uPlanQty
Número de producción actual	uActualQty

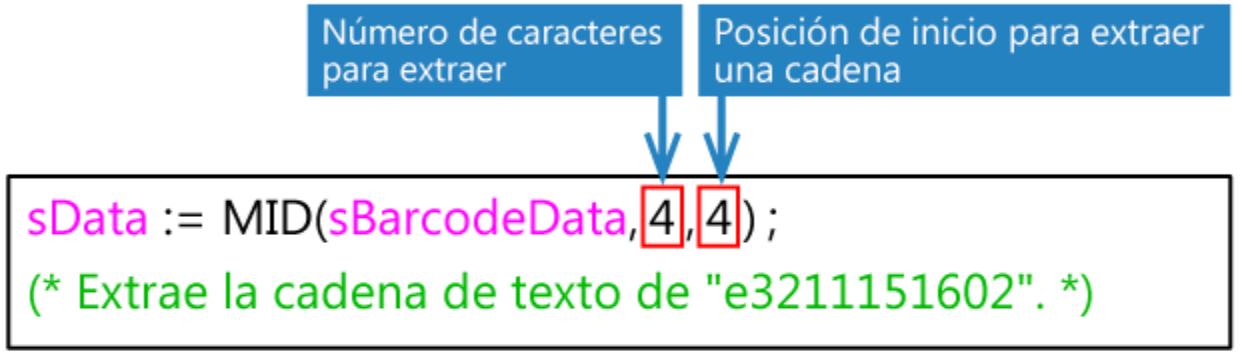


- st3rdLine.bStatus
- st3rdLine->bStatus
- st3rdLine[bStatus]
- st3rdLine[1]

Responder Volver

Prueba final 12

Manipulación de cadenas en ST
El siguiente programa de ejemplo extrae una cadena específica de la cadena “e3211151602” almacenada en la variable “sBarcodeData”. La función MID (medio) extrae el número especificado de caracteres de la posición de comienzo especificada. Seleccione la cadena extraída correctamente.



- 1151
- 1602
- e321
- 1115

Responder Volver

Prueba Calificación de la prueba

Ha completado la prueba final. Sus resultados del área son los siguientes.
Para finalizar la prueba final, continúe con la próxima página.

Respuestas correctas : 12

Total de preguntas : 12

Porcentaje : 100%

Continuar Revisar

Felicitaciones. Aprobó la prueba.

Ha completado el curso Conceptos básicos de la programación (Texto estructurado).

Gracias por tomar este curso.

Esperamos que haya disfrutado las lecciones y que la información recibida en este curso le sea útil en el futuro.

Puede revisar el curso las veces que desee.

Revisar Cerrar