MITSUBISHI ELECTRIC
Changes for the Better

PLC
Noções básicas de programação (Texto estruturado)

Este curso trata de como criar programas básicos usados para controlar controladores programáveis MELSEC.
O texto estruturado (ST) é usado para descrições de programa neste curso.

Introdução Objetivo do curso

Este curso explica como criar programas de controle em texto estruturado (ST) para controladores programáveis MELSEC.

A conclusão do curso a seguir ou conhecimento equivalente é um pré-requisito antes de realizar este curso:

Noções básicas de programação

Ter conhecimento ou experiência com linguagem de programação BASIC ou C pode ajudar a compreender o conteúdo deste curso.

Introdução Estrutura do curso

O conteúdo do curso é explicado a seguir.

Capítulo 1 -Visão geral do texto estruturado

Este capítulo descreve os recursos e aplicações adequadas do texto estruturado (ST).

Capítulo 2 - Regras básicas de programas ST

Este capítulo descreve as regras básicas usadas para criar programas em ST.

Capítulo 3 - Criar programas de controle E/S

Este capítulo descreve como criar programas de controle E/S.

Capítulo 4 - Operações aritméticas

Este capítulo descreve como criar programas de operação aritmética.

Capítulo 5 - Derivação condicional

Este capítulo descreve a derivação condicional.

Capítulo 6 - Armazenamento e processamento de dados

Este capítulo descreve como elaborar programas concisos para armazenar e processar dados.

Capítulo 7 - Processamento de dados em cadeias

Este capítulo descreve os métodos para processar dados em cadeias.

Teste Final

Pontuação mínima para aprovação: 60% ou mais

Introdução Como utilizar esta ferramenta de e-Learning

Ir para a próxima página		Ir para a próxima página.
Voltar para a página anterior		Voltar para a página anterior.
Mover-se para a página desejada		O "Índice" será exibido, permitindo que você navegue até a página desejada.
Sair do curso		Sair do curso.

Introdução Cuidados para uso

Precauções de segurança

Quando você estiver aprendendo a operar os produtos reais, leia cuidadosamente as precauções de segurança dos respectivos manuais.

Precauções neste curso

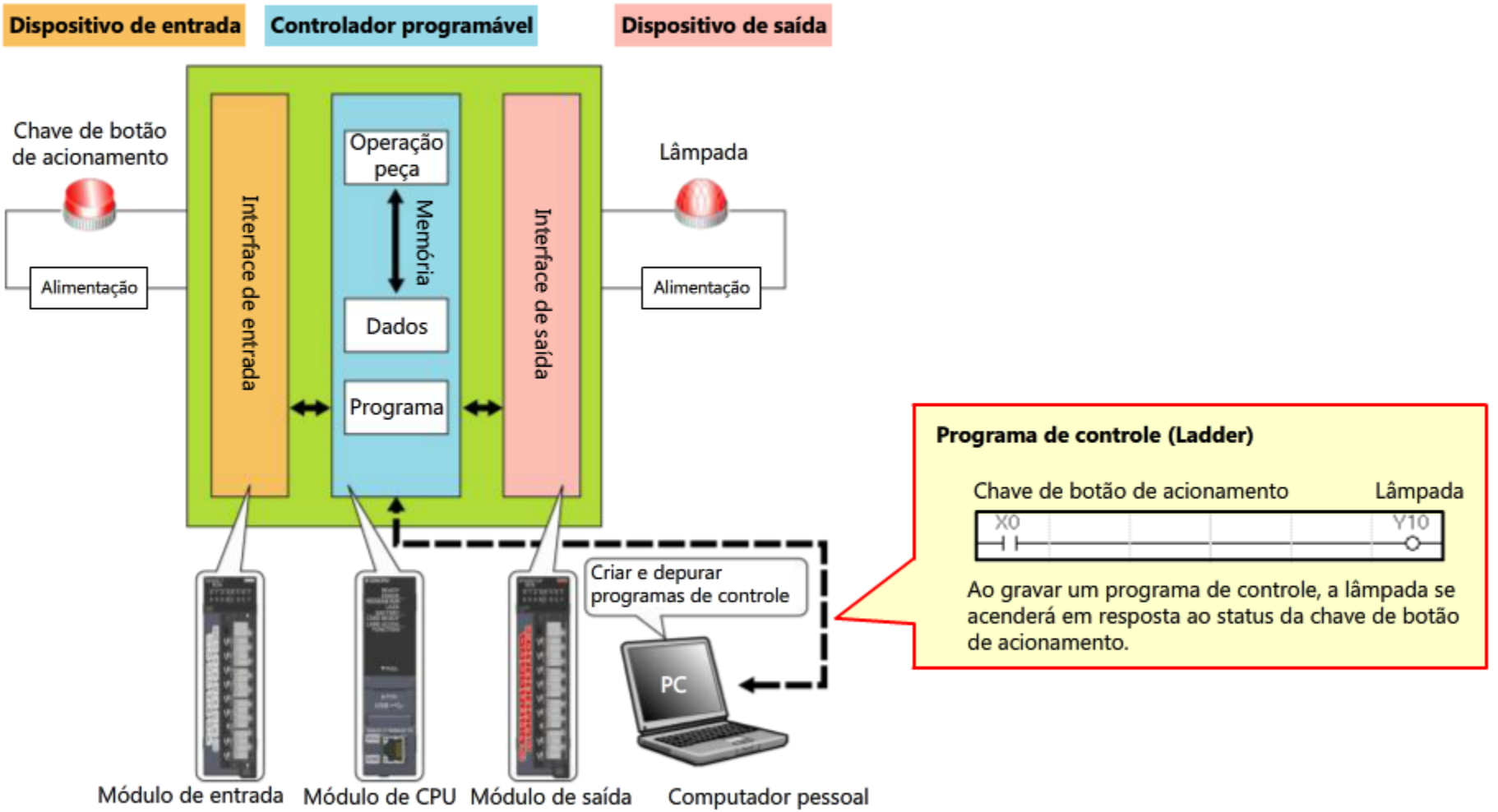
As telas exibidas do software de engenharia MELSOFT que você utiliza podem ser diferentes das apresentadas neste curso. Este curso usa símbolos ladder do MELSOFT GX Works3 para criar programas.

Capítulo 1 Visão geral do texto estruturado

Este capítulo descreve os recursos e aplicações adequadas do texto estruturado (ST).

1.1 Programas de controle

A figura a seguir ilustra a configuração de um sistema do controlador programável.
Controladores programáveis operam de acordo com programas de controle.
A operação de controladores programáveis pode ser configurada conforme desejado ao criar programas de controle.



As linguagens de programação para controladores programáveis são definidas pelo padrão internacional desenvolvido pela International Electrotechnical Commission (IEC) (Comissão eletrotécnica internacional).

1.2 Recursos do ST e comparação com outras linguagens de programação IEC

O IEC 61131 é um padrão internacional para sistemas de controlador programável.
Linguagens de programação para controladores programáveis são padronizadas pelo IEC 61131-3. O ST é uma das linguagens de programação padrão.
Cada linguagem possui diferentes recursos para acomodar sua aplicação e habilidade de programador.

A tabela a seguir lista os recursos das linguagens de programação IEC 61131-3.

Linguagem de programação	Recursos
Ladder **D**iagram (LD) (Diagrama ladder)	• Símbolos para contatos e bobinas são usadas para criar um programa que se parece com um circuito elétrico. • O fluxo do programa pode ser seguido e compreendido facilmente, até mesmo para iniciantes.
Structured **T**ext (ST) (Texto estruturado)	• Programas são escritos em texto (caracteres). • A ST é fácil de aprender para aqueles com experiência na elaboração de programas na linguagem de programação C ou BASIC. • As fórmulas de cálculo são similares a expressões matemáticas, que podem ser facilmente compreendidas. • O ST é adequado para processamento de dados.
Function **B**lock **D**iagram (FBD) (Diagrama de bloco de função)	• Programas são programados ao organizar blocos com diferentes funções e indicar as relações entre os blocos. • O FBD melhora a legibilidade uma vez que toda a operação pode ser facilmente vista.
Sequential **F**unction **C**hart (SFC) (Gráfico de função sequencial)	• Condições e processos são programados como fluxogramas. • O fluxo do programa pode ser compreendido facilmente.
Instruction **L**ist (IL) (Lista de instrução)	• A IL é similar à linguagem de máquina. • A IL é raramente usada nos dias de hoje.

Este curso descreve como fazer programas de controle básicos usando ST.

1.3 Sumário

Os conteúdos deste capítulo são:

- Relação entre sistemas de controlador programável e programas de controle
- Padrão internacional para programas de controle
- Recursos do ST

Pontos importantes a serem levados em conta:

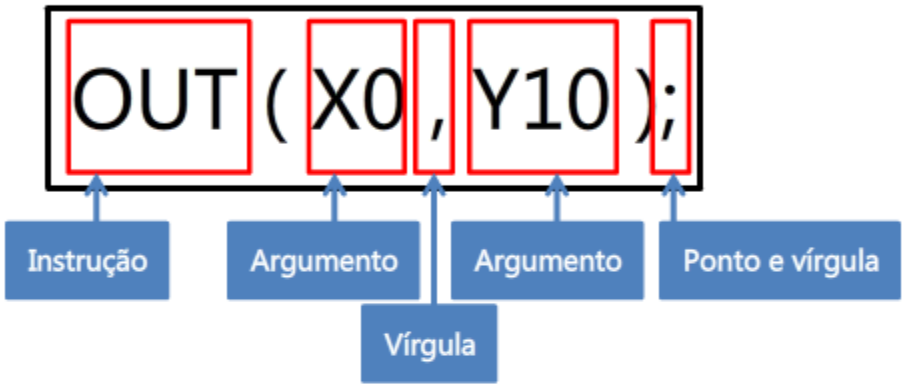
Relação entre sistemas de controlador programável e programas de controle	• Controladores programáveis operam de acordo com programas de controle. • A operação de controladores programáveis pode ser configurada conforme desejado ao criar programas de controle.
Padrão internacional para programas de controle	• O ST é uma das linguagens de programação IEC. • Outras linguagens de programação IEC incluem LD, FBD, SFC, e IL, sendo que cada uma possui diferentes recursos para acomodar sua aplicação e habilidade de programador.
Recursos do ST	• A ST é fácil de aprender para aqueles com experiência na elaboração de programas na linguagem C ou BASIC. • Cálculos, como adição e subtração, podem ser programados como expressões matemáticas típicas, de fácil compreensão. • O ST é adequado para processamento de dados.

Capítulo 2 Regras básicas de programas em ST

Este capítulo descreve as regras básicas usadas para criar programas em ST.

2.1 Exemplo de programa básico (instrução de controle E/S)

Essa seção ilustra um exemplo de um programa ST básico.
Com o programa de exemplo a seguir, o resultado Y10 se acende quando a entrada X0 é acesa, e Y10 se apaga quando X0 se apaga.



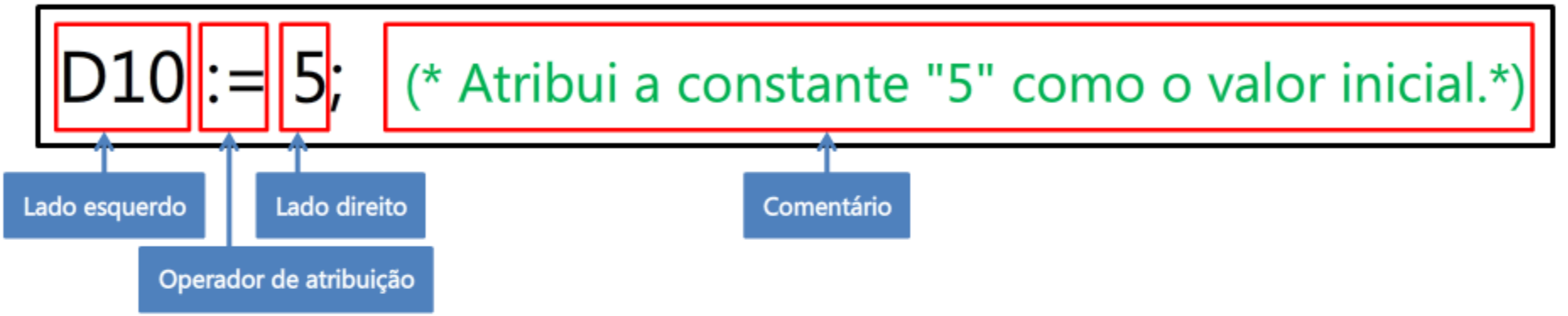
Uma instrução define a operação a ser executada.
Argumentos são escritos em parêntesis após uma instrução.
Argumentos são usados para descrever variáveis, expressões aritméticas e valores constantes.
Com controladores programáveis MELSEC, dispositivos do módulo CPU podem ser usados como variáveis.

O número de argumentos depende da instrução.
Múltiplos argumentos são separados por vírgulas (,).

A linha mostrada acima representa uma instrução. Cada instrução é encerrada por um ponto e vírgula (;).
Um programa é feito através da combinação de instruções.

2.2 Exemplo de programa básico (instrução de atribuição)

O exemplo a seguir ilustra um programa que usa uma instrução de atribuição.
A instrução a seguir atribui uma constante decimal de "5" à variável "D10".



Um operador de atribuição (:=) é usado para essa instrução de atribuição. Observe que dois pontos (:) são colocados à esquerda do sinal de igual (=).
Um operador de atribuição atribui o valor do lado direito ao lado esquerdo.

Adicionar um comentário ao programa torna a operação mais compreensível. Coloque comentários dentro de dois asteriscos (* *).

2.3 Anotação numérica

Com o programa de exemplo na página anterior, um valor decimal foi atribuído a uma variável.

Às vezes, valores diferentes do decimal, como binário e hexadecimal, são usados para controle sequencial.
A tabela a seguir lista os tipos de anotação numérica usados no ST usado para controladores programáveis MELSEC.

Tipo de anotação numérica	Método de anotação	Exemplo
Binário	Adicione um prefixo de "2#".	**2#**11010
Octal	Adicione um prefixo de "8#".	**8#**32
Decimal	Entrada direta	26
	Adicione um prefixo de "K".	**K**26
Hexadecimal	Adicione um prefixo de "16#".	**16#**1A
	Adicione um prefixo de "H".	**H**1A

Exemplos de programas para atribuir valores a variáveis são mostrados abaixo.

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

2.3.1 Anotação de bits

O bit representa condições de true/false (verdadeiro/falso) como estados on/off (ligado/desligado) de sinais. Bits também representam estabelecimento/não estabelecimento de condições.
No ST, bits não podem ser programados como "ON" e "OFF". Eles são expressos como "1" (ON) e "0" (OFF). Os bits também podem ser expressos como "TRUE" e "FALSE".

A tabela a seguir lista os diferentes tipos de anotação.

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

Aqui estão alguns exemplos de atribuir valores para variáveis de bits.

Anotação numérica

```
X0 := 1;
```

=

Anotação true/false

```
X0 := TRUE;
```

Anotação numérica

```
X0 := 0;
```

=

Anotação true/false

```
X0 := FALSE;
```

2.4 Sequência de execução de programa

Instruções de ST são executadas na ordem de cima para baixo.

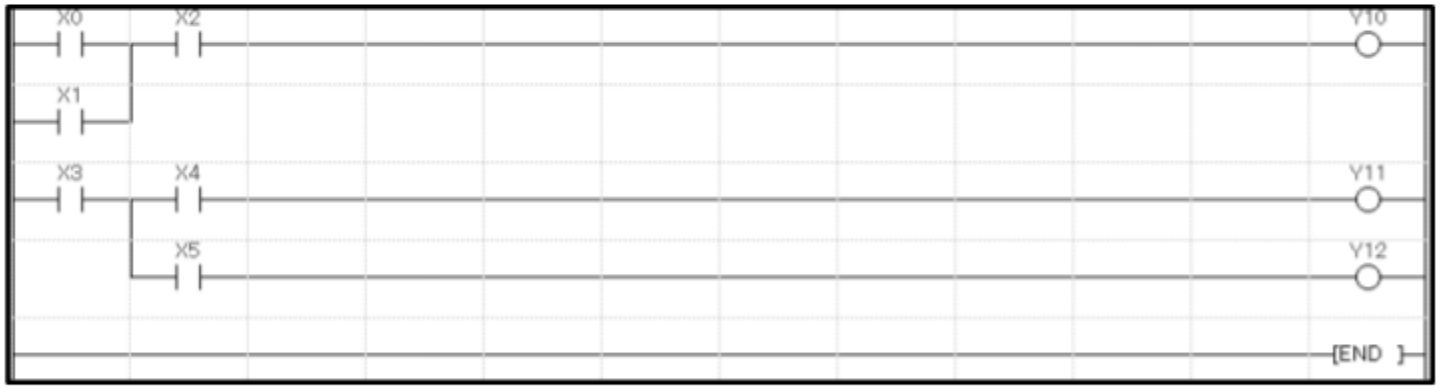
Exemplo de programa ST

```
Y10 := (X0 OR X1) AND X2;        (* Executado primeiro *)
Y11 := X3 AND X4;                (* Executado em segundo *)
Y12 := X3 AND X5;  (* Executado em terceiro. NÃO exige uma instrução END no final.*)
```

Execução repetitiva

*Embora a instrução END seja necessária ao final do programa no LD, ela não é necessária no ST.

O programa ladder a seguir representa a mesma operação como o exemplo de programa ST acima.



Execução repetitiva

Como no caso do LD, instruções no ST são executadas repetidamente ao retornar para a primeira instrução após chegar à última instrução.

2.5 Sumário

Os conteúdos deste capítulo são:

- Programa ST básico
- Formato de instrução de atribuição
- Anotação numérica
- Sequência de execução de programa
- Comentário

Pontos importantes a serem levados em conta:

Programa ST básico	• Uma instrução é o elemento mínimo de programas ST. • Cada instrução é encerrada por um ponto e vírgula (;). • Um programa é feito através da combinação de instruções.
Formato de instrução de atribuição	• Um operador de atribuição (:=) é usado para atribuições.
Anotação numérica	• Tipos de anotação numérica no ST • "1" e "0" são usados para valores de bits no ST em vez de anotações "ON" e "OFF". • Valores de bits também são declarados como "TRUE" e "FALSE" no ST.
Sequência de execução de programa	• Programas criados no ST são executados na ordem de cima para baixo. • Assim como com programas LD, os programas ST processam repetidamente, retornando para o começo do programa depois que o fim do processo é atingido.
Comentário	• Adicionar um comentário ao programa torna a operação mais compreensível. • Comentários ficam dentro de dois asteriscos (* *).

Capítulo 3 Criar programas de controle E/S

Este capítulo descreve como criar programas de controle E/S no ST.

3.1 Programas de controle E/S

O programa a seguir é um exemplo para controle de E/S de um controlador programável.

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

- Instrução de saída
- Condição de execução (argumento)
- Dispositivo de saída (argumento)

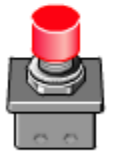
"OUT" é a instrução de saída. Um argumento especifica uma condição de execução e o dispositivo ao qual a saída é direcionada. Quando a condição de execução X0 é satisfeita, o dispositivo Y10 é ligado.

Clique na chave de entrada mostrada abaixo. A chave de entrada X0 se ativa.

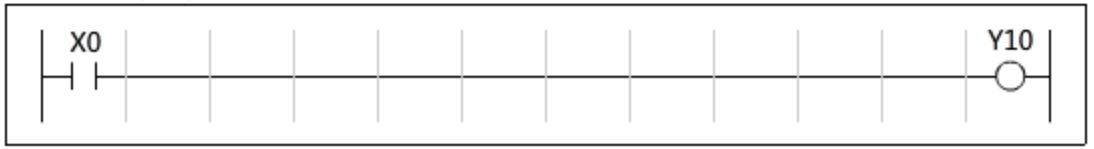
- Quando a chave de entrada X0 se ativa, a lâmpada de saída Y10 se acende.
- Quando a chave de entrada X0 é desligada, a lâmpada de saída Y10 se apaga.

Exemplo de programa de controle de E/S programado no ST

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



O mesmo programa feito no LD



3.1 Programas de controle E/S

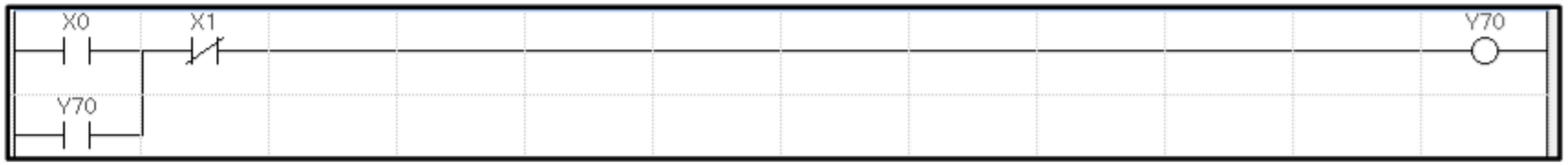
Assim como no LD, há diversas instruções disponíveis além da OUT, como instruções de controle E/S e instruções de processamento de dados.
Consulte o manual de programação de seu controlador programável para obter mais informações sobre as instruções disponíveis no ST.

Observe que escrever "OUT(X0, Y10);" como "Y10 := X0;" produz a mesma operação.

```
Y10 := X0;  (* Mesma operação que "OUT(X0, Y10);" *)
```

3.2 Combinar múltiplas condições

O programa ladder a seguir representa um circuito de auto-retenção.



O mesmo programa pode ser escrito em ST da maneira a seguir.

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

Operador lógico

Como mostrado acima, operadores lógicos são usados para combinar múltiplas condições no ST.

A tabela a seguir lista os operadores lógicos.

Operador	Significado
OR	Lógica OR
AND	Lógica AND
NOT	Lógica de negação
XOR	Exclusivo OR

3.3 Definir significados a variáveis

Usando o ST com controladores programáveis MELSEC, ambos dispositivos e etiquetas podem ser atribuídos como aliases a variáveis.
Os usuários podem usar etiquetas de acordo com os aplicativos.

Quando uma etiqueta relacionada a um aplicativo é atribuída, é mais fácil compreender a operação.

```
Y10 := (X0 OR X1) AND X2;     (* Escrito usando nomes de dispositivos *)
```

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

Etiquetas podem ser nomeadas usando o software de engenharia MELSOFT.

Exemplos de programas subsequentes neste curso são descritos usando etiquetas.

3.4 Sumário

Os conteúdos deste capítulo são:

Exemplos de programas de controle E/S

- Operadores lógicos são usados para combinar múltiplas condições no ST.
- Nomes de dispositivos e etiquetas podem ser usados como nomes de variáveis.

Pontos importantes a serem levados em conta:

Combinar múltiplas condições	• Operadores lógicos são usados para combinar condições no ST.
Definir significados a variáveis	• Quando uma etiqueta relacionada a um aplicativo é atribuída, é mais fácil compreender a operação.

Capítulo 4 Operações aritméticas

Este capítulo descreve como criar programas de operação aritmética.

- Descrever operações aritméticas
- Especificar tipos de dados que correspondem a intervalos numéricos
- Nomear variáveis para evitar inconsistência de tipos de dados

4.1 Operações aritméticas básicas

Esse exemplo de programa soma o volume de produção de duas linhas de produção separadas.
O lado direito de uma equação é uma operação aritmética que contém variáveis e operadores aritméticos.

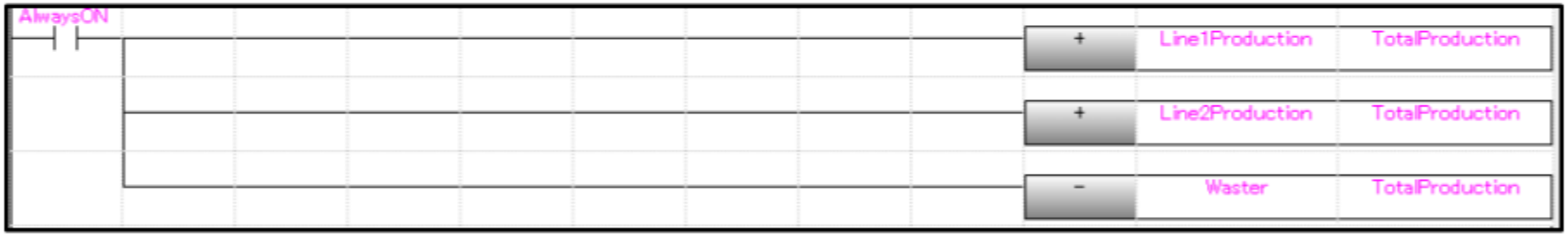
Exemplo de programa aritmético programado em ST

Operador de adição

Operador de subtração

```
TotalProduction := Line1Production + Line2Production - Waster;
(* Total de volume de produção de duas linhas de produção, subtraindo o
número de produtos com defeitos e chegando o valor obtido. *)
```

O mesmo programa programado em LD é mostrado abaixo.



Como mostrado acima, o programa deve ser feito usando 3 linhas no Ladder, mas, no ST, ele pode ser programado em 1 linha.

A tabela a seguir lista os operadores aritméticos básicos.

Operador	Significado
+	Adição
-	Subtração
*	Multiplicação
/	Divisão

4.2 Tipos de dados de variáveis

É necessário especificar um tipo de dado para cada variável a fim de definir o intervalo dos valores a serem processados. Os tipos de dados para valores numéricos usados no ST são bits, números inteiros e números reais.

Entre os tipos de dados usados no ST, a tabela abaixo mostra aqueles usados neste curso.

Tipo de dados		Intervalo de dados
Bit		Estado LIGA/DESLIGA dos dispositivo de bits e estado verdadeiro/falso de resultados de execução
Número inteiro	Palavra (não assinada)	0 - 65.535
	Palavra (assinada)	-32.768 - 32.767
	Palavra dupla (não assinada)	0 - 4.294.967.295
	Palavra dupla (assinada)	-2.147.483.648 - 2.147.483.647

Ao usar o tipo de número inteiro, selecione o tipo de palavra dupla ou palavra segundo o intervalo de dados, e selecione o tipo assinada ou não assinada de acordo com a necessidade de processar valores negativos. Especifique o tipo de dados de uma variável quando o nome da etiqueta é definido usando o software de engenharia MELSOFT.

4.3 Nomes de variáveis que representam tipos de dados

O uso de diferentes tipos de dados nos lados esquerdo e direito de uma equação de atribuição pode causar um erro de compilação ou resultado inesperado.
Abaixo há um exemplo de um desses casos.

```
ValueA := ValueB;      (* ValueA: Número inteiro de palavra ValueB: Número inteiro de palavra dupla *)
```

Um número inteiro de palavra dupla não pode ser atribuído a um número inteiro de palavra. Qualquer que seja o caso, o tipo de dado não é reconhecível.

Prefixos que representam o tipo de dados podem ser adicionados a nomes de variáveis para facilitar a identificação visual dos tipos de dados.
Esse tipo de nomenclatura variável é conhecido como anotação húngara.

Tipo de dados		Intervalo de dados	Prefixo	Expansão de prefixo
Bit		Estado LIGA/DESLIGA dos dispositivo de bits e estado verdadeiro/falso de resultados de execução	b	**B**it
Número inteiro	Palavra (não assinada)	0 - 65.535	u	**u**nsigned word (palavra não assinada)
	Palavra (assinada)	-32.768 - 32.767	w	signed **w**ord (palavra assinada)
	Palavra dupla (não assinada)	0 - 4.294.967.295	ud	**u**nsigned **d**ouble-word (palavra dupla não assinada)
	Palavra dupla (assinada)	-2.147.483.648 - 2.147.483.647	d	signed **d**ouble-word (palavra dupla assinada)

O programa de exemplo no topo dessa página pode ser escrito da seguinte forma usando a anotação húngara:

```
wValueA := dValueB;  (* Variável de palavra dupla não pode ser atribuída a uma variável de palavra. *)
```

Ao usar a anotação húngara, as inconsistências de tipos de dados podem ser identificadas no processo de programação de um programa.

No restantes do curso, os nomes de variáveis no exemplo são escritos em anotação húngara.

4.4 Sumário

Os conteúdos deste capítulo são:

- Descrever operações aritméticas
- Especificar tipos de dados que correspondem a intervalos numéricos
- Adicionar nomes de variáveis que representam tipos de dados

Pontos importantes a serem levados em conta:

Operações aritméticas básicas	• Operadores comuns para linguagens de programação gerais podem ser usados em ST para expressar cálculo.
Tipos de dados de variáveis	• É necessário especificar um tipo de dado para cada variável a fim de definir o intervalo dos valores a serem processados.
Adicionar nomes de variáveis que representam tipos de dados	• Descrever nomes de variáveis usando a anotação húngara permite que inconsistências em tipos de dados variáveis sejam identificados ao programar programas.

Capítulo 5 Derivação condicional

Programas de controle também contêm seções de códigos nas quais o processamento real muda de acordo com as condições especificadas.
Este capítulo descreve a derivação condicional.

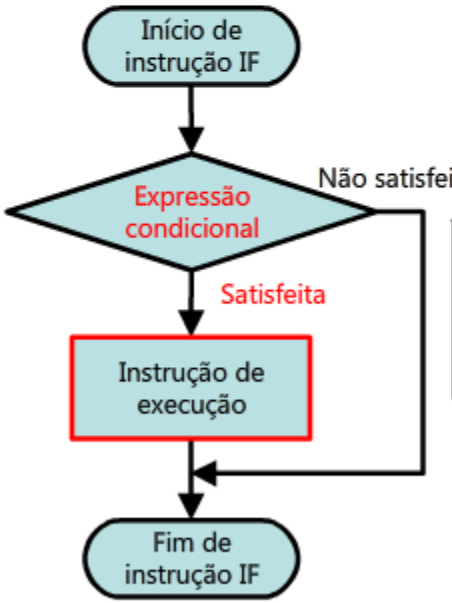
5.1 Derivação condicional (IF)

Instruções IF são usadas para derivação condicional. Instruções IF são usadas da seguinte maneira.

```
IF expressão condicional THEN
  Execution statement;          (* Instrução é executada se a expressão condicional for satisfeita. *)
END_IF;                         (* END_IF; deve ser colocado no final de instruções IF. *)
```

Neste programa de exemplo, uma instrução é executada quando a expressão condicional é satisfeita. A instrução não é executada quando a expressão condicional não é satisfeita.

A figura a seguir ilustra o fluxo de operação nesse programa de exemplo.

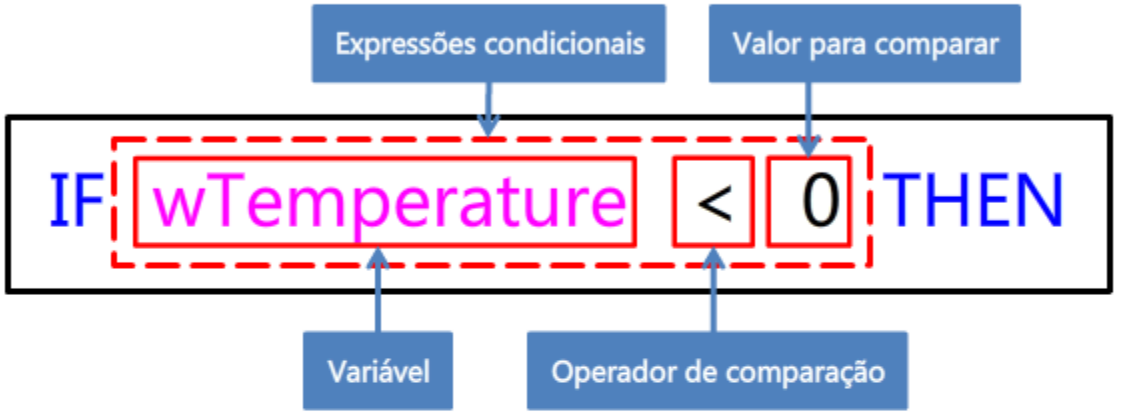


O exemplo a seguir ilustra a derivação do programa comparando valores de variáveis. No exemplo de programa, o aquecedor se ativa quando a temperatura no painel de controle cai para menos de 0 graus.

```
IF wTemperature < 0 THEN
  bHeater := 1;  (* O aquecedor é acionado quando a temperatura no painel de controle cai para menos de 0 graus. *)
END_IF;
```

5.1.1 Programação de expressões condicionais

A página anterior descreveu uma expressão condicional de "wTemperature < 0", que significa "quando o valor da variável wTemperature for inferior a 0".
Assim como essa expressão, as expressões condicionais usam operadores de comparação para representar a relação entre as variáveis e valores para comparar.



Nos lados esquerdo e direito de um operador de comparação, valores são escritos como variáveis ou limitações para comparação.

Além de comparar variáveis e limitações, expressões condicionais podem ser escritas para comparar variáveis e realizar operações lógicas de resultados de comparação de variáveis de tipo de bits.

Comparar variáveis

- uValue1 <= uValue2

Operação lógica para dois resultados de comparação

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

Operação lógica para duas variáveis de tipos de bits

- bSwitch0 OR bSwitch1

A tabela a seguir lista os tipos de operadores de comparação.

Operador	Significado
>	Superior a
<	Inferior a
>=	Superior a ou igual a
<=	Inferior a ou igual a
=	Igual a
<>	Diferente de

5.1.2 Derivação excepcional para instrução IF (ELSE)

Instruções IF simples (consulte a seção 5.1) são usadas para executar uma instrução quando a expressão condicional é satisfeita. Para executar uma instrução diferente quando a expressão condicional não é satisfeita e a instrução ELSE é usada.

```
IF conditional expression THEN
  Execution statement 1;          (* Instrução 1 é executada se expressão condicional for satisfeita. *)
  ELSE
  Execution statement 2;          (* Instrução 2 é executada se expressão condicional não for satisfeita *)
END_IF;
```

A figura a seguir ilustra o fluxo de operação ao usar uma instrução ELSE.

O programa de exemplo a seguir executa diferentes instruções dependendo da condição estar ou não satisfeita.

O programa de exemplo na seção 5.1 possuía uma desvantagem que o aquecedor continuava elevando a temperatura mesmo depois de atingir 0 graus. No entanto, o programa a seguir desliga o aquecedor quando a variável "wTemperature" ultrapassa 0 graus.

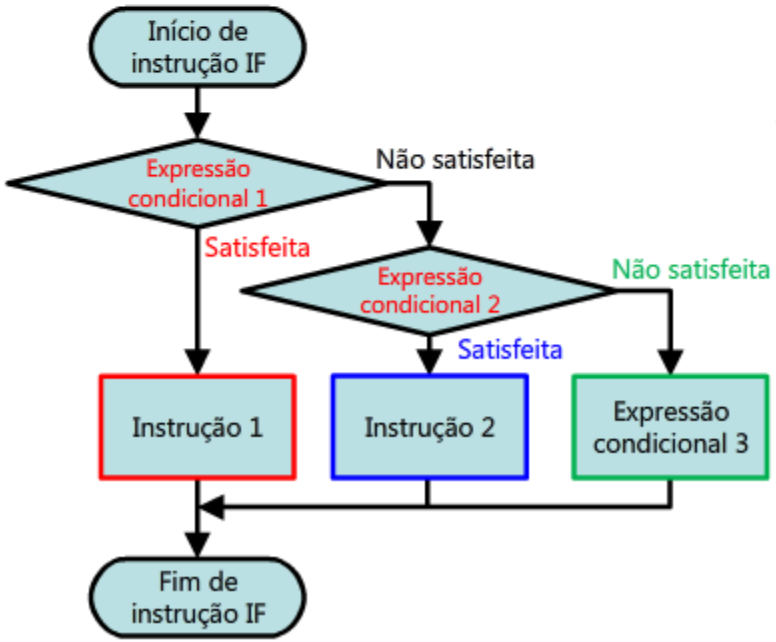
```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Liga o aquecedor quando a temperatura cai para menos de 0 graus. *)
  ELSE
  bHeater := 0;  (* Desliga o aquecedor quando a temperatura atinge ou ultrapassa 0 graus *)
END_IF;
```

5.1.3 Derivação adicional para instrução IF (ELSIF)

Instruções ELSE são usadas para executar uma instrução diferente quando a expressão condicional não é satisfeita. Outra derivação condicional pode ser adicionada usando as instruções ELSIF, o que significa que se a expressão condicional anterior não foi satisfeita, outra expressão condicional é verificada.

```
IF Conditional expression 1 THEN
  Execution statement 1;      (* Instrução 1 é executada se expressão condicional 1 for satisfeita. *)
  ELSIF Conditional expression 2 THEN
  Execution statement 2;      (* A instrução 2 é executada se a expressão condicional 1 não for satisfeita e a expressão condicional 2 for satisfeita. *)
  ELSE
  Execution statement 3;      (* A instrução 3 é executada se as expressões condicionais 1 e 2 não forem satisfeitas. *)
END_IF;
```

A figura a seguir ilustra o fluxo de operação ao usar uma instrução ELSEIF.



Instrução ELSIF é adicionada ao programa de exemplo ilustrado em 5.1.2 para lidar com o caso quando a temperatura ultrapassa 40 graus.

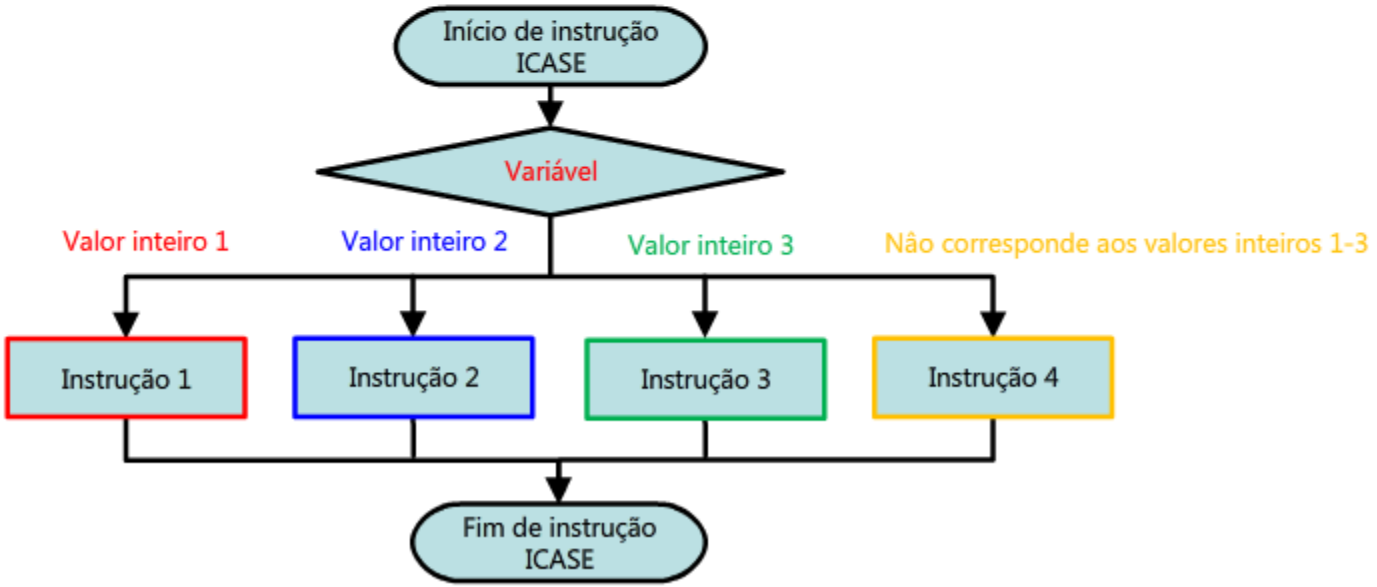
```
IF wTemperature < 0 THEN
   bHeater := 1; (* Liga o aquecedor quando a temperatura cai para menos de 0 graus. *)
   bCooler := 0; (*  Desliga o resfriador quando a temperatura fica abaixo de 0 graus. *)
   ELSIF 40 < wTemperature THEN
   bHeater := 0; (* Desliga o aquecedor se a temperatura atingir ou ultrapassar 40 graus. *)
   bCooler := 1; (* Liga o resfriador se a temperatura atingir ou ultrapassar 40 graus. *)
   ELSE
   bHeater := 0; (*  Desliga o aquecedor se nenhuma das condições anteriores for satisfeita. *)
   bCooler := 0; (*  Desliga o resfriador se nenhuma das condições anteriores for satisfeita. *)
END_IF;
```

5.2 Derivação condicional de acordo com valores inteiros (CASE)

Instruções IF são usadas para derivação dependendo de expressões condicionais serem ou não satisfeitas.
Instruções CASE são usadas para derivação de acordo com os valores inteiros.
A figura a seguir ilustra como uma instrução CASE é programada.

```
CASE Variable OF
    Integer value 1: Execution statement 1;     (* Instrução 1 é executada quando a variável corresponde ao valor inteiro 1. *)
    Integer value 2: Execution statement 2;     (* Instrução 2 é executada quando a variável corresponde ao valor inteiro 2. *)
    Integer value 3: Execution statement 3;     (* Instrução 3 é executada quando a variável corresponde ao valor inteiro 3. *)
    ELSE Execution statement 4;                 (* Instrução 4 é executada se a variável não corresponde a nenhum dos valores inteiros. *)
END_CASE;                                       (* "END_CASE;" deve ser colocado no final da instrução CASE. *)
```

A figura a seguir ilustra o fluxo de operação ao usar uma instrução CASE.



5.2.1 Programa de exemplo para instrução CASE

A execução de instrução CASE é descrita usando a operação de programa de exemplo.

Clique em para prosseguir para a próxima página.
Para assistir à animação novamente, clique no botão "Reproduzir".

Reproduzir


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

Peso	uSize	Classificação
0 a 20 kg	1	M
21 a 30 kg	2	L
31 a 40 kg	3	XL
41 kg e superior	4	Oversize

5.3 Sumário

Os conteúdos deste capítulo são:

- Derivação condicional com instruções IF
- Programação de expressões condicionais
- Derivação condicional de acordo com valores inteiros (instrução CASE)

Pontos importantes a serem levados em conta:

Instrução IF	• Com uma instrução IF, o programa é derivado quando uma expressão condicional é satisfeita. • A instrução ELSE é usada para derivação quando a expressão condicional não é satisfeita. • Uma instrução ELSIF é usada para adicionar outra derivação quando a expressão condicional na instrução IF não é satisfeita.
Expressão condicional	• Expressões condicionais representam a relação entre as variáveis e os valores para comparar usando operadores de comparação.
Instrução CASE	• Instruções CASE são usadas para derivação de acordo com os valores inteiros.

Capítulo 6 Armazenamento e processamento de dados

Além de serem usados para aplicações de controle de E/S, atualmente os controladores programáveis são empregados para processar grandes quantidades de dados como o núcleo de sistemas de produção.
Para processar grandes quantidades de dados, os dados devem ser armazenados e depois lidos conforme necessários.
Este capítulo descreve como elaborar programas concisos para armazenar e processar dados.

- Matrizes são usadas para sequenciar e organizar variáveis.
- Estruturas de dados são usadas para organizar variáveis relacionadas.
- Programas de processamento de loop processam matrizes de forma eficiente usando instruções FOR.

Programas concisos para armazenar e processar dados podem ser criados usando matrizes, estruturas de dados e instruções FOR.

6.1 Sequenciamento e armazenamento de dados (matriz)

Ao usar matrizes, múltiplos valores podem ser processados por uma variável.
No exemplo a seguir, os dados de volume de produção em uma fábrica automotiva são armazenados por país de destino.

Destino	País A	País B	País C
Volume de produção	35	75	65

Os dados de volume de produção por país de destino são atribuídos a uma variável. Sem usar matrizes, uma variável deve ser criada para cada destino.
No entanto, ao usar matrizes, o volume de produção para múltiplos destinos pode ser atribuído e armazenado em uma variável.

Sem usar matrizes

uProductionA
uProductionB
uProductionC

Usando matrizes

uProduction

Variáveis individuais na matriz são especificadas usando números de elementos. Números de elementos começam do zero [0].

No programa de exemplo a seguir, a variável do volume de produção planejado para o País A é atribuída.

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

6.1.1 Matriz de estrutura

Em seguida, dados de cor de pintura são usados além dos dados de destino.

Destino	País A			País B			País C		
Cor de pintura									
Volume de produção	10	5	20	15	40	20	25	30	10
	35 no total			75 no total			65 no total		

Como ilustrado na tabela a seguir, dados podem ser separados e armazenados por cor de pintura (coluna) para cada país de destino (linha).

Cor de pintura (coluna)

Destino (linha)	Vermelho	Amarelo	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 o destino

Número de elemento que representa a cor de pintura

Matrizes que organizam dados em linhas e colunas dessa maneira são conhecidas como matrizes de estrutura. Números de elementos que representam as linhas e as colunas são separados por vírgulas.

Variável de matriz (matriz de estrutura)

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

6.1.2 Atribuição de matriz de estrutura

Usando matrizes de estrutura, o programa de exemplo a seguir atribui o número de carros a serem fabricados com urgência, além do volume de produção planejado de automóveis amarelos para o País B.

```
uAdditionalProduction := 5;
uProduction[1,1] := uProduction[1,1] + uAdditionalProduction;
(* Acrescenta a quantidade adicional de produção (5 unidades) ao volume de produção inicialmente planejado. *)
```

Destino	País A			País B			País C		
Cor de pintura									
Volume de produção	10	5	20	15	40	20	25	30	10
	35 no total			75 no total			65 no total		

Cor de pintura (coluna)

Destino (linha)	Vermelho	Amarelo	Azul
País A	[0,0] 10	[0,1] 5	[0,2] 20
País B	[1,0] 15	[1,1] 40 -> 45	[1,2] 20
País C	[2,0] 25	[2,1] 30	[2,2] 10

6.1.3 Processamento de informações armazenadas em matrizes de estrutura

Usando matrizes de estrutura, o programa de exemplo a seguir calcula o volume de produção total planejado para todas as cores de tintas para o País C e atribui o valor a uma variável

```
uProductionToday := uProduction[2,0] + uProduction[2,1] +uProduction[2,2];
(* Calcula o volume de produção total para o dia para todas as cores de tinta para o País C e atribui o valor à variável "uProductionToday". *)
```

Destino	País A			País B			País C		
Cor de pintura									
Volume de produção	10	5	20	15	45	20	25	30	10
	35 no total			80 no total			65 no total		

Cor de pintura (coluna)

Destino (linha)	Vermelho	Amarelo	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 Looping (FOR)

O programa de exemplo na página anterior (o volume de produção planejado para hoje é atribuído) é mostrado abaixo novamente.

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

Com esse exemplo de programa, quando o número de cores pintadas aumenta, mais variáveis serão adicionadas. Em seguida, a expressão fica maior, dificultando a leitura.

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

Nesse caso, as instruções de loop podem ser usadas para criar códigos mais legíveis.

Instruções de loop para instruções FOR, WHILE e REPEAT. Esse curso trata das instruções FOR.

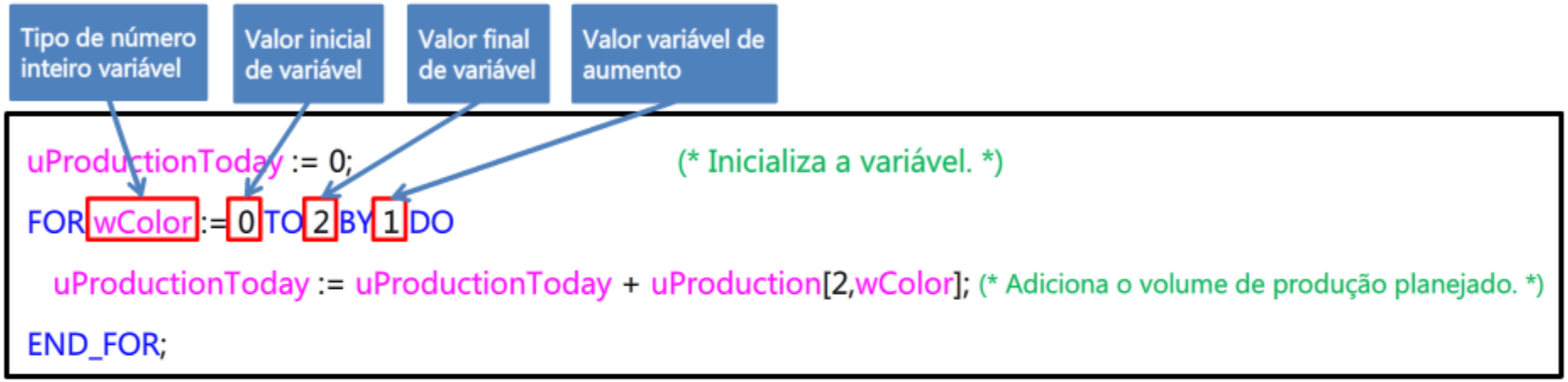
Instruções FOR são usadas da seguinte maneira.

```
FOR variable := initial value TO final value BY increments DO
   Execution statement; (* instrução é executada em loop até a variável atingir o valor final. *)
END_FOR;                (* END_FOR; deve ser colocado no final de instruções FOR.  *)
```

A instrução é repetida até o valor final da variável ser atingido e o código "END_FOR;" ser executado.

6.2 Looping (FOR)

Usando uma instrução FOR, o programa de exemplo a seguir obtém o volume de produção planejado para todas as cores de pintura do País C.



Usando a instrução, a variável "wColor" aumenta em um começando do valor inicial de zero e a instrução é repetida até a variável atingir o valor final de dois.
A variável "wColor" é especificada como o segundo número de elemento na matriz "uProduction" descrita na instrução de execução.
O valor da variável "wColor" aumenta toda vez que a instrução é repetida. O volume de produção planejado para cada cor de pinta é adicionado a cada tempo para obter o total.

Esse programa de exemplo é executado em loop três vezes. (Primeira vez: vermelho [0] => Segunda vez: amarelo [1] => Terceira vez: azul [2])

A operação desse programa é ilustrada na próxima página.

6.2 Looping (FOR)

A execução da instrução FOR é descrita usando a operação do programa de exemplo.

Matriz de volume de produção estimada

	Vermelho	Amarelo	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

Clique em para prosseguir para a próxima página.
Para assistir à animação novamente, clique no botão "Reproduzir".

Reproduzir

Number of repetition of the loop: 3

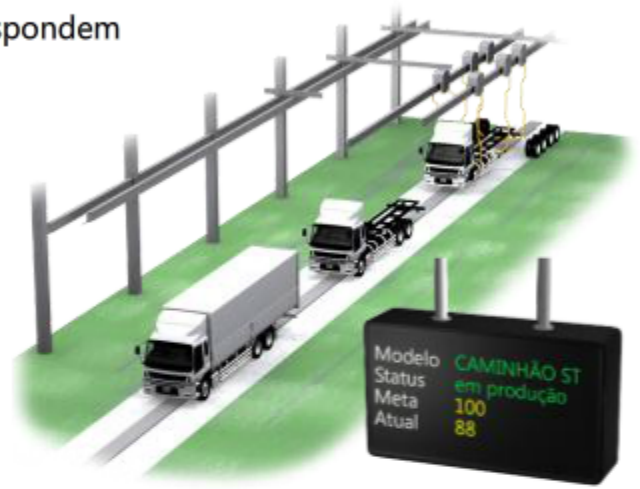
```
uProductionToday := 0;

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

6.3 Armazenamento de dados relacionados (estrutura)

Uma estrutura possibilita que um nome de variável represente múltiplas variáveis relacionadas.
No exemplo a seguir, o status de uma linha de produção automóvel é mostrado no Andon (quadro de exibição).

A tabela a seguir lista os nomes de variáveis, valores e os tipos de dados que correspondem aos itens mostrados.

Item	Nome da variável	Valor	Tipo de dados variáveis
Modelo	sModel	'CAMINHÃO ST'	Cadeia de texto
Status	bStatus	'em produção'	Tipo de bits
Volume de produção de meta para hoje	uPlanQty	'100'	Palavra do tipo inteiro (não assinada)
Volume de produção atual	uActualQty	'88'	Palavra do tipo inteiro (não assinada)



Se uma estrutura não for usada, os nomes variáveis devem ser alterados para cada linha quando houver múltiplas linhas de produção.
Abaixo há alguns exemplos de nomes variáveis por linha de produção.

Primeira linha de produção

s1stLineModel
b1stLineStatus
u1stLinePlanQty
u1stLineActualQty

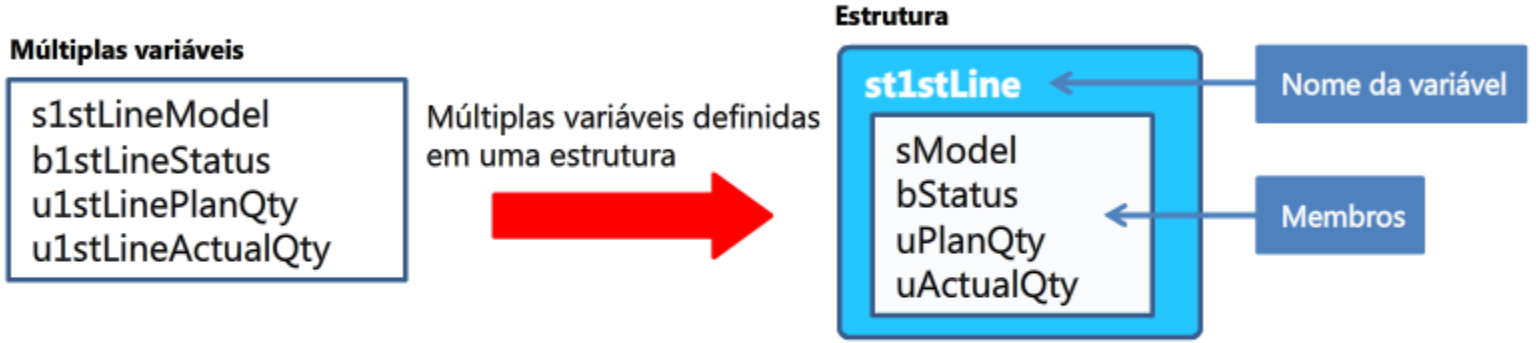
Segunda linha de produção

s2ndLineModel
b2ndLineStatus
u2ndLinePlanQty
u2ndLineActualQty

Quando o número de linha de produção aumenta, o número de variáveis a ser processado também aumenta. Em seguida, o programa fica maior, dificultando a leitura.

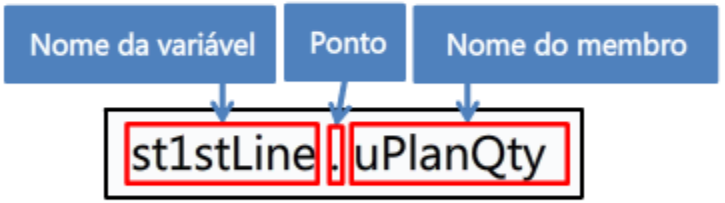
6.3 Armazenamento de dados relacionados (estrutura)

Usar estruturas possibilita que um nome de variável represente múltiplas variáveis relacionadas a uma linha de produção. Dessa forma, estruturas são usadas para organizar, armazenar e processar dados em lote para condições e especificações de objetos físicos como dispositivos, equipamentos e peças.



A **st**ructure variable (variável de estrutura) contém um prefixo de "st" para representar que essa é uma estrutura.
As variáveis individuais definidas pela estrutura são conhecidas como membros. Tipos de dados de cada membro podem ser diferentes.

Cada membro de matrizes de estrutura pode ser especificado após o número de elemento da matriz usando um ponto antes do nome do membro.



No programa de exemplo a seguir, uma constante é atribuída a um membro da variável de estrutura para a primeira linha de produção.

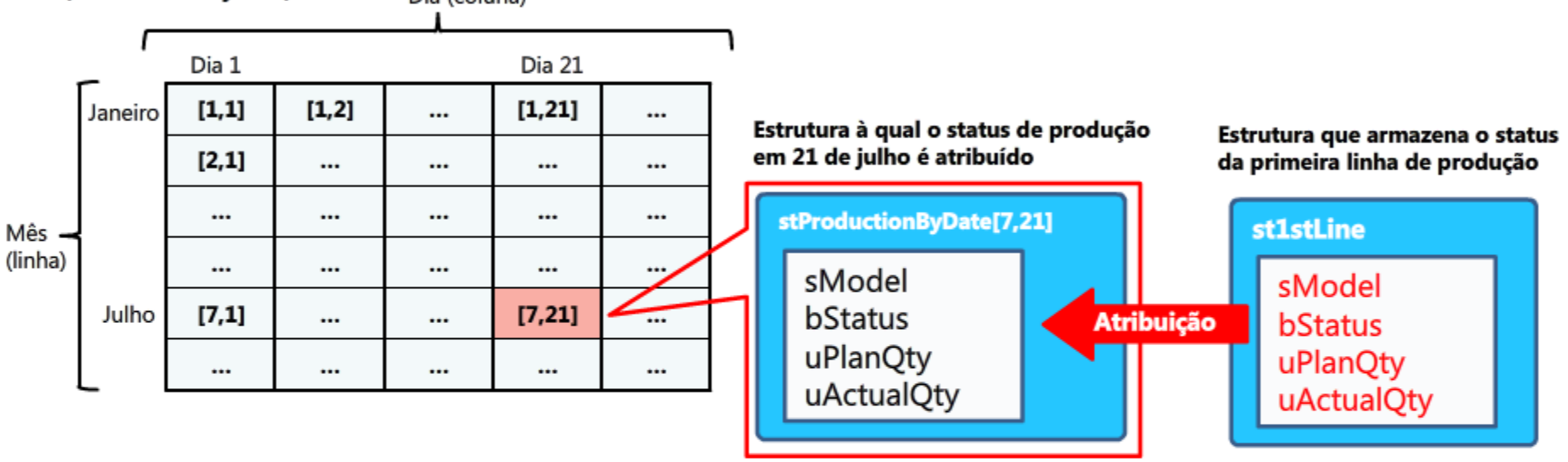
```
st1stLine.uPlanQty := 150;
 (* Define a meta de produção de hoje da primeira linha de produção para 150. *)
```

6.3.1 Armazenamento de matrizes de estrutura

Estruturas podem ser criadas como matrizes.
No exemplo a seguir, o status de produção é armazenado por data.

Estrutura organizada* por data (stProductionByDate)

* Nessa matriz, o número de elemento começa a partir de "1".

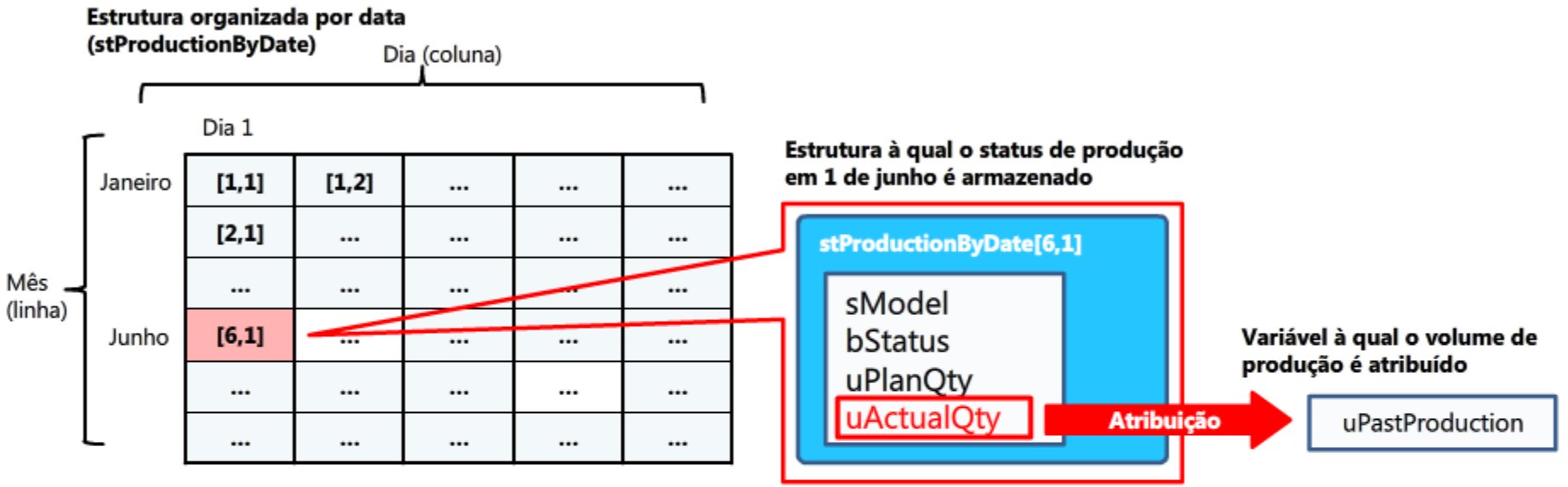



```
stProductionByDate[7,21] := st1stLine;
 (* O status de produção em 21 de julho é armazenado na estrutura
organizada por data (stProductionByDate). *)
```

Dessa forma, membros não precisam ser especificados individualmente para atribuição de estrutura.

6.3.2 Leitura de matrizes de estrutura

No exemplo a seguir, o volume de produção é lido de uma estrutura organizada por data e depois atribuída a uma variável.




```
uPastProduction := stProductionByDate[6,1].uActualQty;
(* Atribui o volume de produção em 1 de junho para a variável uPastProduction. *)
```

Cada membro das matrizes de estrutura pode ser especificado ao inserir um ponto (.) e um nome de membro ao número de elemento da matriz.

6.4 Sumário

Os conteúdos deste capítulo são:

- Visão geral e uso de matrizes
- Processamento de loop para instruções FOR
- Visão geral e uso de estruturas

Pontos importantes a serem levados em conta:

Matriz	• Múltiplos valores podem ser processados por uma variável usando matrizes. • Variáveis individuais em matrizes são especificados por números de elemento adicionados ao fim de nomes de variáveis.
Instrução FOR	• Instruções de loop são usadas nos programas quando se busca uma operação repetitiva. • Instruções FOR são usadas para repetir a operação até as condições para o fim da operação de loop serem satisfeitas. A instrução antes da instrução "END_FOR;" é executada repetidamente.
Estrutura	• Estruturas permitem que um nome de variável represente múltiplas variáveis relacionadas. Estruturas podem incluir variáveis de diferentes tipos de dados. • Variáveis individuais, ou membros, definidas em estruturas são especificadas ao adicionar um ponto e o nome do membro após o nome da variável de estrutura.

Capítulo 7 Processamento de dados em cadeias

Em alguns casos, controladores programáveis usam dados de cadeia para enviar comandos para ou receber retorno de dispositivos conectados, como leitores de código de barras, controladores de temperatura ou balanças eletrônicas. Para tais finalidades, é necessário unir ou extrair dados de cadeia conforme necessário.

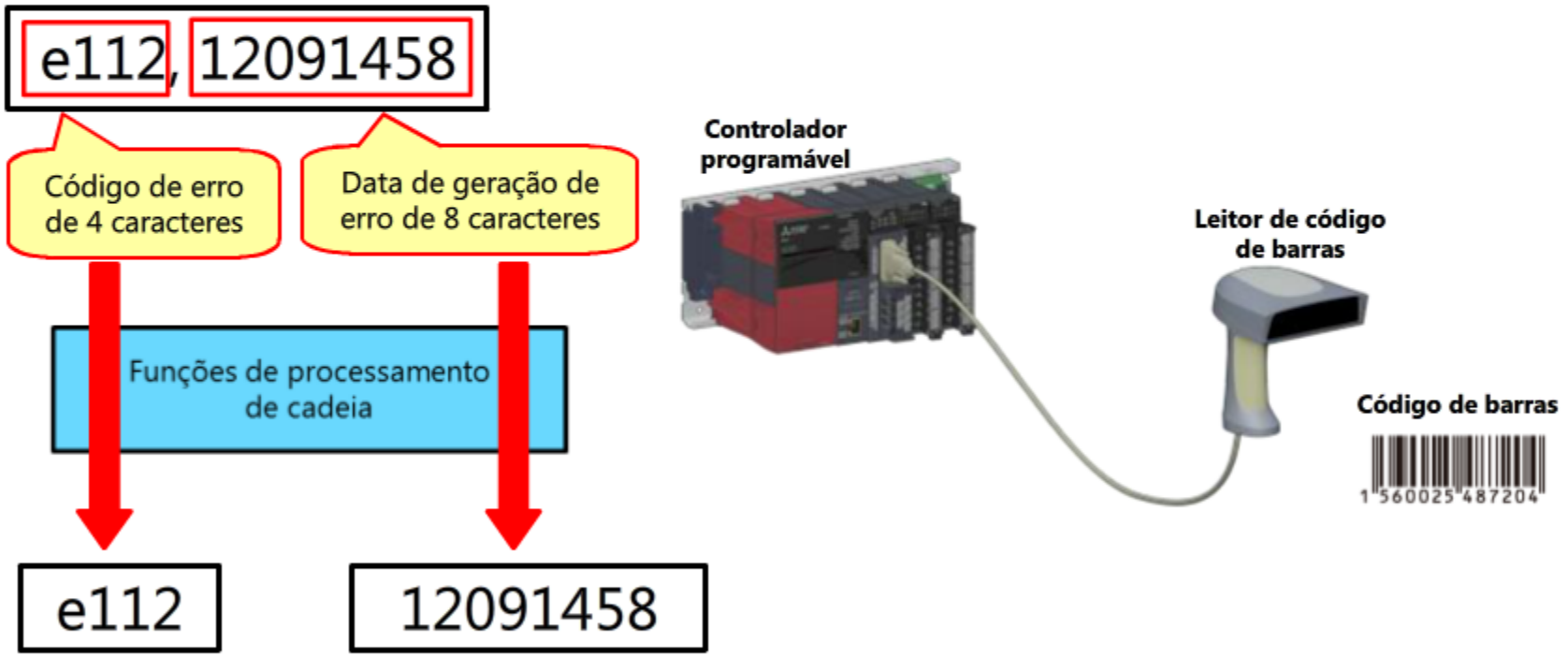
Este capítulo descreve como processar dados em cadeias.

7.1 Exemplo de processamento de dados em cadeias

Como um exemplo de processamento de cadeias, o exemplo ilustra uma situação na qual dados são lidos de um leitor de código de barras.
Funções (um tipo de instruções) são usadas para processar cadeias.

Como ilustrado abaixo, cadeias lidas pelo leitor de código de barras contêm um código de erro de comprimento fixo de 4 caracteres e dados de mês, minuto, hora e data de comprimento fixo e 8 caracteres.
O exemplo de programa de processamento de cadeias será descrito neste sistema.

Exemplo de dados de cadeia lidos de um leitor de código de barras



Um código de erro é extraído.
7.3 Extração de cadeias (LEFT)

A data e hora de ocorrência de erro (14:58, 09 de dezembro) são extraídas.
7.4 Extração de cadeias (MID)

7.2 Atribuição de cadeias

Antes de explicar como extrair cadeias, essa seção descreve os tipos de dados para cadeias.

Os tipos de dados para cadeias que podem ser usados com controladores programáveis são listados na tabela a seguir.

Tipo de dados	Tipo de caractere que pode ser processado	Prefixos de anotação húngara	Expansão de prefixo
Cadeia	Cadeias de caracteres alfanuméricos e números (ASCII) ou japoneses (Shift-JIS)	s	**s**tring (cadeia)
Cadeia [Unicode]	Cadeias de muitas linguagens e símbolos diferentes	ws	**w**ide **s**tring (cadeia ampla)

O tipo de cadeia a ser usado depende do dispositivo estar conectado ao controlador programável ou da linguagem correspondente.
Esse capítulo descreve diferentes prefixos de cadeias de texto.

Quando um tipo cadeia é atribuído a uma variável de cadeia, inclua a cadeia dentro de aspas simples (').

```
sDefault := 'e112,12091458';  (* Atribuição de cadeias *)
```

7.3 Extração de cadeias (LEFT)

O código de erro "e112" é extraído da variável de cadeia "sBarcodeData" que contém a cadeia "e112,12091458".

Nome da variável	Cadeia armazenada
sBarcodeData	e112, 12091458

A função LEFT extrai apenas o número de caracteres especificados começando pelo lado esquerdo da cadeia de entrada.
Os dados a seguir ilustram um programa de exemplo.

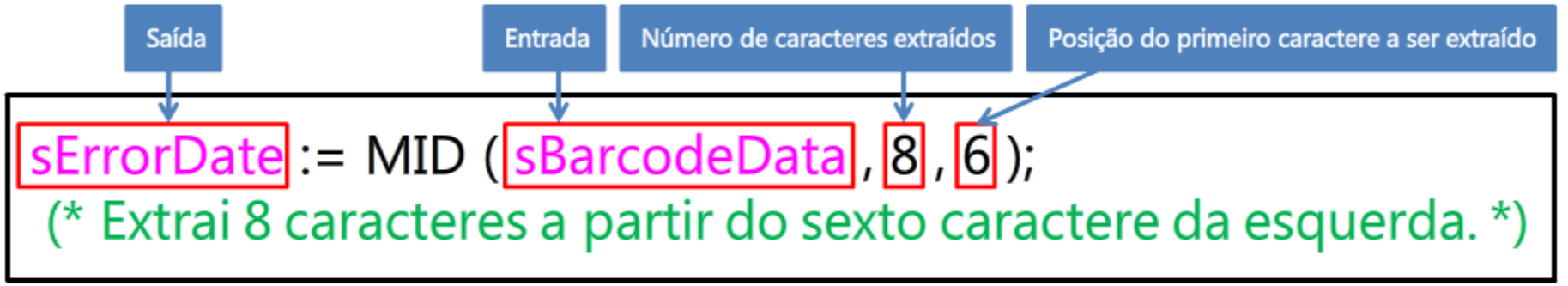
Quatro caracteres são extraídos a partir da esquerda. Um valor de "e112", que é a cadeia que representa o código de erro, é atribuído ao lado esquerdo.

7.4 Extração de cadeias (MID)

O tempo de geração de erro "12091458" é extraído da variável de cadeia "sBarcodeData" que contém a cadeia "e112,12091458".

Nome da variável	Cadeia armazenada
sBarcodeData	e112,12091458

A função MID extrai o número especificado de caracteres a partir da posição inicial especificada na cadeia de entrada.
Os dados a seguir ilustram um programa de exemplo.



Nesse exemplo, uma cadeia de 8 caracteres é extraída a partir do sexto caractere. Um valor de "12091458", que é a cadeia que representa a hora de ocorrência do erro, é atribuído ao lado esquerdo.

7.5 Sumário

Os conteúdos deste capítulo são:

- Métodos para atribuir cadeias a variáveis de cadeias
- Funções que extraem cadeias (LEFT e MID)

Pontos importantes a serem levados em conta:

Atribuição de cadeias	• Para atribuir uma cadeia a uma variável de cadeia, inclua a cadeia dentro de aspas simples ('). • Use o tipo de cadeia ou o tipo [Unicode] de cadeia de acordo com o dispositivo conectado ao controlador programável ou a linguagem correspondente.
Funções para processamento de cadeias	• Funções são usadas para processar cadeias.

7.6 Resumo do curso

Este curso tratou das noções básicas sobre como criar programas em ST.
Isso nos leva ao final deste curso de aprendizado virtual.

Os programas ST são criados usando o software de engenharia MELSOFT.
Para obter detalhes sobre as etapas específicas, como cadastrar dados, editar, salvar e compilar programas com o software de engenharia MELSOFT, consulte os recursos a seguir.

- Curso de aprendizado virtual da Mitsubishi FA "MELSOFT GX Works3 (Structured Text)" (MELSOFT GX Works3 (Texto estruturado)) (lançamento em breve)
- Manual de operação de seu software de engenharia MELSOFT

Para obter mais informações sobre o ST, consulte os recursos a seguir.

- Guia de programação de seu controlador programável

Para obter informações sobre instruções e funções de seu aplicativo, consulte os recursos a seguir.

- Manual de programação de seu controlador programável

Teste Final

Agora que concluiu todas as lições do curso Noções básicas de programação (texto estruturado), você está pronto para fazer o teste final. Se tiver qualquer dúvida sobre os tópicos abrangidos, aproveite esta oportunidade para revê-los.
O Teste Final é composto por 12 perguntas (20 itens).
Você pode fazer o teste final quantas vezes desejar.

Como é feita a pontuação do teste
Depois de selecionar a resposta, não se esqueça de clicar no botão **Resposta**. Sua resposta será perdida se você continuar sem clicar nesse botão. (O sistema assumirá que essa pergunta não foi respondida).

Resultados da pontuação
O número de respostas corretas, o número de perguntas, a porcentagem de respostas corretas e o resultado (aprovado/reprovado) aparecem na página de pontuação.

Respostas corretas: **5**

Total de perguntas: **5**

Porcentagem: **100%**

Para passar no teste, você precisa responder corretamente a **60%** das perguntas.

Continuar | Rever

- Clique no botão **Continuar** para sair do teste.
- Clique no botão **Rever** para rever o teste. (Verificar a resposta correta)
- Clique no botão **Repetir** para refazer o teste.

Teste — Teste final 1

Características do texto estruturado (ST)
Selecione a descrição correta do ST.

- A ST é fácil de aprender para aqueles com experiência na elaboração de programas na linguagem C ou BASIC.
- Cálculos, como adição e subtração, podem ser programados como expressões matemáticas típicas.
- Símbolos para contatos e bobinas são usadas para criar um programa que se parece com um circuito elétrico.
- O ST é adequado para processamento de dados.

Resposta | Voltar

Teste — Teste final 2

Princípios básicos do ST
Selecione a instrução correta programada em ST.

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

Resposta Voltar

Teste

Teste final 3

Descrever comentários
Selecione o comentário correta programado em ST.

- ' Atribui um valor de 1 à variável.
- (* Atribui um valor de 1 à variável. *)
- { Atribui um valor de 1 à variável. }
- <!-- Atribui um valor de 1 à variável. -->

Resposta Voltar

Teste — Teste final 4

Sequência de execução de programa ST
*O valor inicial de "uTotalProduction" é "100". O valor da variável "uTotalProduction" será "101" após o programa de exemplo a seguir ser processado. Selecione o status "uTotalProduction" correto depois de alguns segundos.

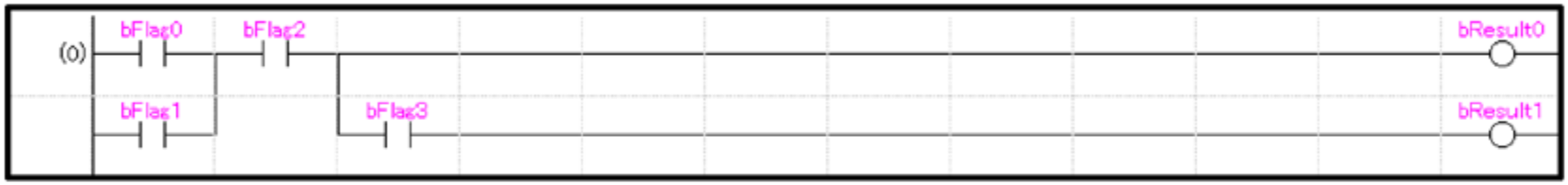
```
uTotalProduction := uTotalProduction + 1;
```

- O valor permanece em 101.
- O valor continua mudando.

Resposta Voltar

Teste — Teste final 5

Combinar múltiplas condições
Selecione o exemplo de programa ST correto que represente a mesma operação que o exemplo de programa a seguir em LD.



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

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

Resposta | Voltar

Teste — Teste final 6

Descrição de elementos IF no ST
A operação a seguir é executada pelo programa abaixo.
•Se a temperatura cair para 5 graus ou menos, o aquecedor se ativa e o resfriador é desligado.
•Se a temperatura ultrapassar 50 graus, o aquecedor se desliga e o resfriador é ligado.
•Se a temperatura não se aplica às instruções acima, ambos aquecedor e resfriador são desligados.

*Nomes de variáveis: Temperatura (wTemperature), aquecedor (bHeater) e resfriador (bCooler)
Selecione a escolha correta para cada seção em branco do programa de exemplo.

```
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--

Resposta Voltar

Teste — Teste final 7

Instruções CASE

Selecione a opção correta para cada um (Q1 a Q5) da descrição a seguir das instruções CASE.

Instruções CASE são usadas para derivação de acordo com o valor de (Q1).
No programa de exemplo a seguir, quando o valor de (Q2) é 25, a variável (Q3) recebe o valor de (Q4).
Quando o valor da variável (Q2) não é igual a 10, 25 ou o 8, a variável (Q3) recebe o 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--

Resposta Voltar

Teste — Teste final 8

Matrizes ST e instruções repetitivas

O programa de exemplo a seguir soma o volume de produção planejado de todos os modelos destinados ao país Y e depois atribui esse valor a uma variável. Selecione a seção da matriz que é lida após a instrução FOR ser executada em um loop 3 vezes.

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

Matriz usada para armazenar o número estimado de unidades produzidas por modelo e destino (uProduction)

Destino (linha)	Modelo 1	Modelo 2	Modelo 3	Model 4
			Modelo (coluna)	
País X	[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]

- A
- B
- C
- D

Resposta | Voltar

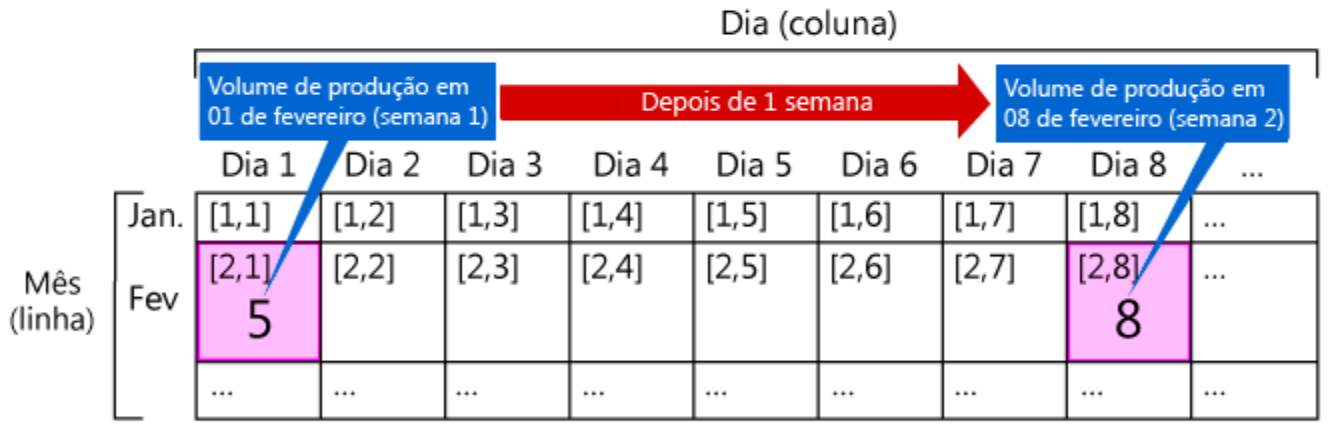
Teste — Teste final 9

Matrizes ST e instruções repetitivas

O programa de exemplo a seguir obtenha o volume total de produção nos mesmos dias da semana. O total em 4 semanas é obtido da matriz que armazena o volume de produção por dia. Selecione o número correto para o programa de exemplo.

```
uTotalProduction := 0;
FOR wOnceAWeek := 1 TO ■ BY 7 DO
  uTotalProduction := uTotalProduction + uProductionByDate[2,wOnceAWeek];
END_FOR;
(* Extrai os totais de volume de produção nos mesmos dias da semana ao longo de 4 semanas a partir de 1 de fevereiro. *)
```

Matriz que armazena o volume de produção por dia (uProductionByDate)



- 22
- 21
- 4
- 28

Resposta | Voltar

Teste — Teste final 10

Características de estruturas em ST
Selecione a descrição incorreta das estruturas.

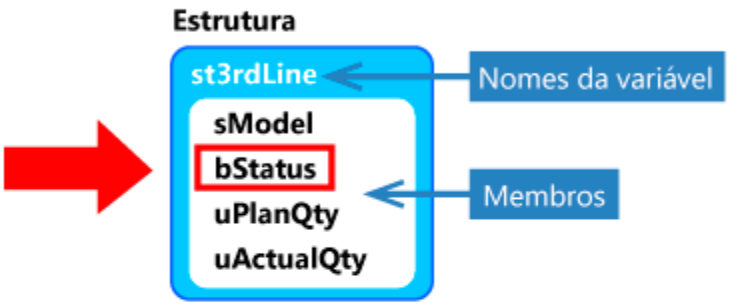
- Estruturas são usadas para organizar e armazenar dados em dispositivos por condições como status e especificações.
- Programas que processam grandes quantidades de dados podem ser programados de forma concisa usando estruturas.
- Todos os membros definidos na estrutura devem ter o mesmo tipo de dados.
- Os valores podem ser atribuídos aos membros na mesma estrutura sem serem individualmente especificados.

Resposta | Voltar

Teste — Teste final 11

Especificar membros para estruturas no ST
A estrutura a seguir organiza as varia´veis relacionadas a uma linha de produção automotiva.
Selecione a descrição correta para especificar o membro “bStatus” nessa estrutura.

Parâmetro	Nomes da variável
Modelo	sModel
Status	bStatus
Meta de produção para o dia atual	uPlanQty
Número de produção atual	uActualQty

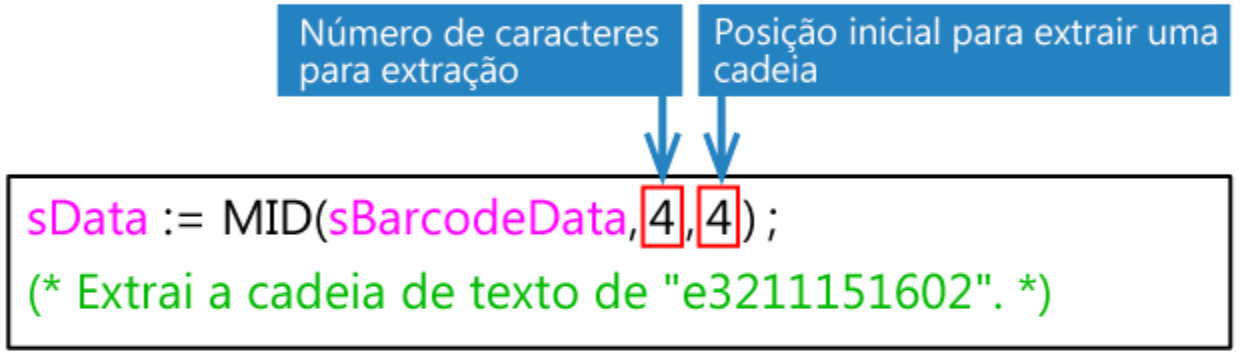


- st3rdLine.bStatus
- st3rdLine->bStatus
- st3rdLine[bStatus]
- st3rdLine[1]

Resposta Voltar

Teste — Teste final 12

Processamento de cadeias no ST
O programa de exemplo a seguir extrai uma cadeia específica da cadeia "e3211151602" armazenada na variável "sBarcodeData". A função MID extrai o número especificado de caracteres a partir da posição inicial especificada. Selecione a cadeia extraída corretamente.



- 1151
- 1602
- e321
- 1115

Resposta | Voltar

Teste Pontuação do Teste

Você concluiu o Teste Final. Seus resultados são os seguintes.
Para terminar o Teste Final, vá para a próxima página.

Respostas corretas:	12
Total de perguntas:	12
Porcentagem:	100%

Continuar Rever

Parabéns. Você passou no teste.

Você concluiu o curso Noções básicas de programação (texto estruturado).

Muito obrigado por fazer este curso.

Esperamos que tenha gostado das lições e que as informações adquiridas sejam úteis no futuro.

Você pode rever o curso quantas vezes quiser.

Rever Fechar