MITSUBISHI ELECTRIC
Changes for the Better

PLC

Kiến thức cơ bản về lập trình (Văn bản có cấu trúc)

Khóa học này hướng dẫn cách tạo các chương trình cơ bản dùng để điều khiển bộ điều khiển khả trình MELSEC.
Khóa học này sử dụng văn bản có cấu trúc (ST) cho mô tả chương trình.

Giới thiệu Mục đích khóa học

Khóa học này giải thích cách tạo các chương trình điều khiển bằng văn bản có cấu trúc (ST) cho bộ điều khiển khả trình MELSEC.

Điều kiện tiên quyết cho khóa học này là bạn phải hoàn thành khóa học sau đây hoặc có kiến thức tương đương:

Kiến thức cơ bản về lập trình

Có kiến thức hoặc kinh nghiệm về ngôn ngữ lập trình C hoặc BASIC có thể giúp bạn hiểu được nội dung của khóa học này.

Giới thiệu Cấu trúc khóa học

Nội dung của khóa học này như sau.

Chương 1 - Tổng quan về văn bản có cấu trúc

Chương này mô tả các đặc điểm và cách ứng dụng thích hợp của văn bản có cấu trúc (ST).

Chương 2 - Những quy tắc cơ bản của chương trình viết bằng ST

Chương này mô tả những quy tắc cơ bản được sử dụng để tạo chương trình bằng ST.

Chương 3 - Tạo chương trình điều khiển I/O

Chương này mô tả cách tạo chương trình điều khiển I/O.

Chương 4 – Phép toán số học

Chương này mô tả cách tạo chương trình phép toán số học.

Chương 5 - Phân nhánh có điều kiện

Chương này mô tả về phân nhánh có điều kiện.

Chương 6 - Lưu trữ và xử lý dữ liệu

Chương này mô tả cách viết những chương trình ngắn gọn để lưu trữ và xử lý dữ liệu.

Chương 7 – Xử lý dữ liệu chuỗi

Chương này mô tả các phương pháp xử lý dữ liệu chuỗi.

Bài kiểm tra cuối khóa

Điểm đạt: 60% trở lên

Giới thiệu Cách sử dụng công cụ e-Learning này

Đến trang tiếp theo		Đến trang tiếp theo.
Trở lại trang trước		Trở lại trang trước.
Di chuyển đến trang mong muốn		"Mục lục" sẽ được hiển thị, cho phép bạn điều hướng đến trang mong muốn.
Thoát khỏi bài học		Thoát khỏi bài học.

Giới thiệu Cảnh báo khi sử dụng

Biện pháp phòng ngừa an toàn

Khi bạn học tập dựa trên các sản phẩm thực tế, hãy đọc kỹ các biện pháp phòng ngừa an toàn trong hướng dẫn sử dụng tương ứng.

Biện pháp phòng ngừa trong khóa học này

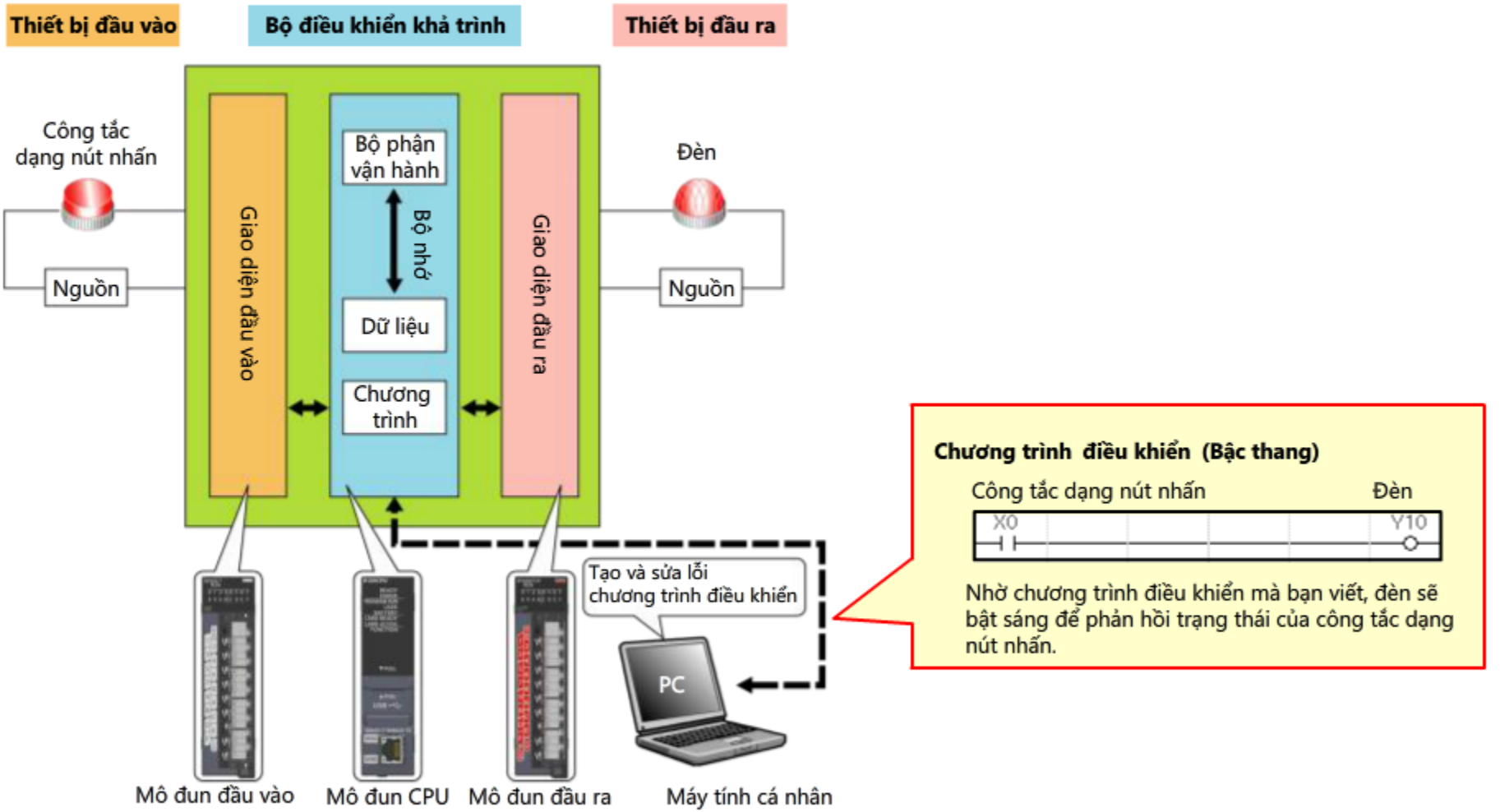
Các màn hình hiển thị của phần mềm kỹ thuật MELSOFT mà bạn sử dụng có thể khác với các màn hình trong khóa học này.
Khóa học này sử dụng các ký hiệu trong sơ đồ bậc thang của MELSOFT GX Works3 để tạo chương trình.

Chương 1 Tổng quan về văn bản có cấu trúc

Chương này mô tả các đặc điểm và cách ứng dụng thích hợp của văn bản có cấu trúc (ST).

1.1 Chương trình điều khiển

Hình dưới đây minh họa cấu hình của một hệ thống PLC.
Bộ điều khiển khả trình hoạt động dựa trên các chương trình điều khiển.
Có thể cấu hình hoạt động của các bộ điều khiển khả trình theo ý muốn bằng cách tạo chương trình điều khiển.



Ngôn ngữ lập trình cho các bộ điều khiển khả trình được định nghĩa theo tiêu chuẩn quốc tế do International Electrotechnical Commission (IEC) (Ủy ban Kỹ thuật điện quốc tế) phát triển.

1.2 Đặc điểm của ST và so sánh với các ngôn ngữ lập trình theo chuẩn IEC khác

IEC 61131 là một tiêu chuẩn quốc tế về hệ thống PLC.
Các ngôn ngữ lập trình cho bộ điều khiển khả trình được tiêu chuẩn hóa theo IEC 61131-3. ST là một trong những ngôn ngữ lập trình tiêu chuẩn.
Mỗi ngôn ngữ có những đặc điểm khác nhau để phù hợp với ứng dụng của bạn và kỹ năng của lập trình viên.

Bảng dưới đây liệt kê đặc điểm của các ngôn ngữ lập trình theo chuẩn IEC 61131-3.

Ngôn ngữ lập trình	Đặc điểm
Ladder **D**iagram (LD) (Sơ đồ bậc thang)	• Sử dụng các biểu tượng tiếp điểm và cuộn cảm để tạo ra chương trình giống một mạch điện. • Dòng chương trình dễ theo dõi và dễ hiểu, kể cả với người mới học.
Structured **T**ext (ST) (Văn bản có cấu trúc)	• Chương trình được viết bằng văn bản (ký tự). • ST dễ học đối với người có kinh nghiệm viết chương trình bằng ngôn ngữ lập trình C hoặc BASIC. • Công thức tính toán tương tự như các biểu thức toán học nên rất dễ hiểu. • ST thích hợp để xử lý dữ liệu.
Function **B**lock **D**iagram (FBD) (Sơ đồ khối chức năng)	• Viết chương trình bằng cách sắp xếp các khối chứa các hàm số khác nhau và chỉ định mối liên hệ giữa các khối. • FBD dễ đọc hơn vì có thể thấy được toàn bộ quá trình vận hành.
Sequential **F**unction **C**hart (SFC) (Sơ đồ chức năng trình tự)	• Biểu diễn các điều kiện và quá trình dưới dạng sơ đồ tiến trình. • Dòng chương trình dễ hiểu.
Instruction **L**ist (IL) (Danh sách lệnh)	• IL tương tự như ngôn ngữ máy. • Ngày nay, IL hiếm khi được sử dụng.

Khóa học này mô tả cách viết chương trình điều khiển cơ bản bằng ST.

1.3 Tổng kết

Nội dung của chương này như sau:

- Mối liên hệ giữa hệ thống PLC và các chương trình điều khiển
- Tiêu chuẩn quốc tế về chương trình điều khiển
- Đặc điểm của ST

Những điểm quan trọng cần cân nhắc:

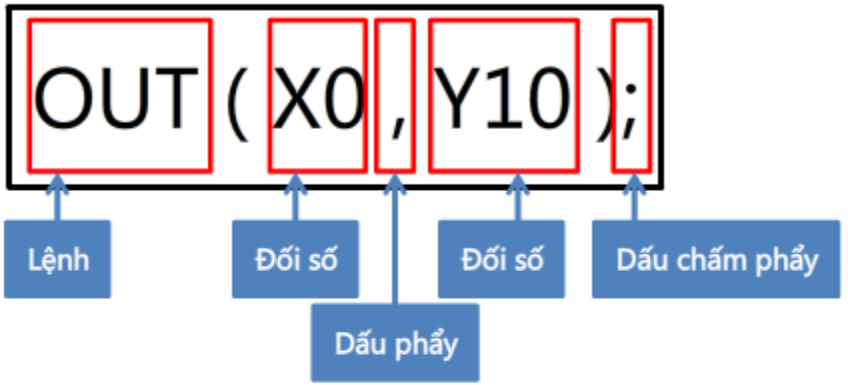
Mối liên hệ giữa hệ thống PLC và các chương trình điều khiển	• Bộ điều khiển khả trình hoạt động dựa trên các chương trình điều khiển. • Có thể cấu hình hoạt động của các bộ điều khiển khả trình theo ý muốn bằng cách tạo chương trình điều khiển.
Tiêu chuẩn quốc tế về chương trình điều khiển	• ST là một trong những ngôn ngữ lập trình theo chuẩn IEC. • Các ngôn ngữ lập trình theo chuẩn IEC khác bao gồm LD, FBD, SFC và IL. Mỗi ngôn ngữ lập trình này có những đặc điểm khác nhau để phù hợp với ứng dụng của bạn và kỹ năng của lập trình viên.
Đặc điểm của ST	• ST dễ học đối với người có kinh nghiệm viết chương trình bằng ngôn ngữ C hoặc BASIC. • Các phép tính như phép cộng và phép trừ có thể được viết dưới dạng biểu thức toán học thông thường nên rất dễ hiểu. • ST thích hợp để xử lý dữ liệu.

Chương 2 Những quy tắc cơ bản của chương trình viết bằng ST

Chương này mô tả những quy tắc cơ bản được sử dụng để tạo chương trình bằng ST.

2.1 Ví dụ chương trình cơ bản (Câu lệnh điều khiển I/O)

Phần này minh họa ví dụ về một chương trình viết bằng ST cơ bản.
Với chương trình ví dụ sau đây, đầu ra Y10 bật khi đầu vào X0 bật và Y10 tắt khi X0 tắt.



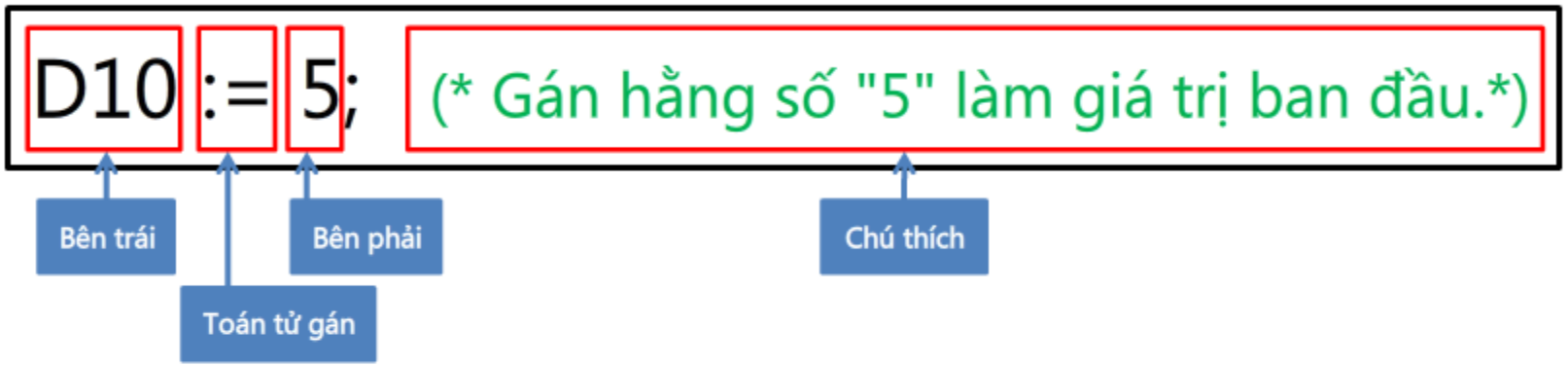
Lệnh xác định quy trình vận hành được thực hiện.
Các đối số được viết trong ngoặc đơn, sau lệnh.
Đối số được sử dụng để mô tả các biến số, biểu thức số học và giá trị hằng số.
Với các bộ điều khiển khả trình MELSEC, có thể sử dụng các thiết bị thuộc mô đun CPU làm biến số.

Số lượng đối số phụ thuộc vào lệnh.
Khi có nhiều đối số thì sử dụng dấu phẩy (,) để phân tách.

Dòng ở trên biểu diễn một câu lệnh. Mỗi câu lệnh kết thúc bằng dấu chấm phẩy (;).
Kết hợp các câu lệnh sẽ tạo nên một chương trình.

2.2 Ví dụ chương trình cơ bản (Câu lệnh gán)

Ví dụ sau đây minh họa một chương trình sử dụng câu lệnh gán.
Câu lệnh sau gán hằng số thập phân "5" cho biến số "D10".



Câu lệnh gán này sử dụng một toán tử gán (:=). Lưu ý rằng dấu hai chấm (:) nằm ở bên trái dấu bằng (=).
Toán tử gán sẽ gán giá trị ở bên phải cho bên trái.

Thêm chú thích vào chương trình sẽ giúp quy trình vận hành dễ hiểu hơn. Đặt chú thích ở giữa hai dấu hoa thị (* *).

2.3 Ký hiệu số

Với chương trình ví dụ ở trang trước, một giá trị thập phân đã được gán cho một biến số.

Đôi khi, các giá trị không phải thập phân, như giá trị nhị phân hay thập lục phân, được sử dụng trong điều khiển theo trình tự. Bảng sau liệt kê các kiểu ký hiệu số trong ST được sử dụng cho các bộ điều khiển khả trình MELSEC.

Kiểu ký hiệu số	Phương pháp ký hiệu	Ví dụ
Nhị phân	Thêm tiền tố "2#".	**2#**11010
Bát phân	Thêm tiền tố "8#".	**8#**32
Thập phân	Nhập trực tiếp	26
	Thêm tiền tố "K".	**K**26
Thập lục phân	Thêm tiền tố "16#".	**16#**1A
	Thêm tiền tố "H".	**H**1A

Ví dụ chương trình gán giá trị cho các biến số được trình bày dưới đây.

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

2.3.1 Ký hiệu bit

Bit biểu diễn các điều kiện true/false (đúng/sai), chẳng hạn như trạng thái on/off (bật/tắt) của tín hiệu. Bit còn biểu diễn trạng thái xác lập/không xác lập của điều kiện.
Trong ST, không thể viết bit dưới dạng "ON" và "OFF". Bit được biểu diễn dưới dạng "1" (ON) và "0" (OFF). Cũng có thể biểu diễn bit dưới dạng "TRUE" và "FALSE".

Bảng dưới đây liệt kê các kiểu ký hiệu khác nhau.

Trạng thái	ON	OFF
	True	False
Ký hiệu số	1	0
Ký hiệu true/false	TRUE	FALSE

Dưới đây là một số ví dụ về gán giá trị cho các biến số kiểu bit.

Ký hiệu số

```
X0 := 1;
```

=

Ký hiệu true/false

```
X0 := TRUE;
```

Ký hiệu số

```
X0 := 0;
```

=

Ký hiệu true/false

```
X0 := FALSE;
```

2.4 Trình tự thực hiện chương trình

Các câu lệnh ST được thực hiện theo thứ tự từ trên xuống dưới.

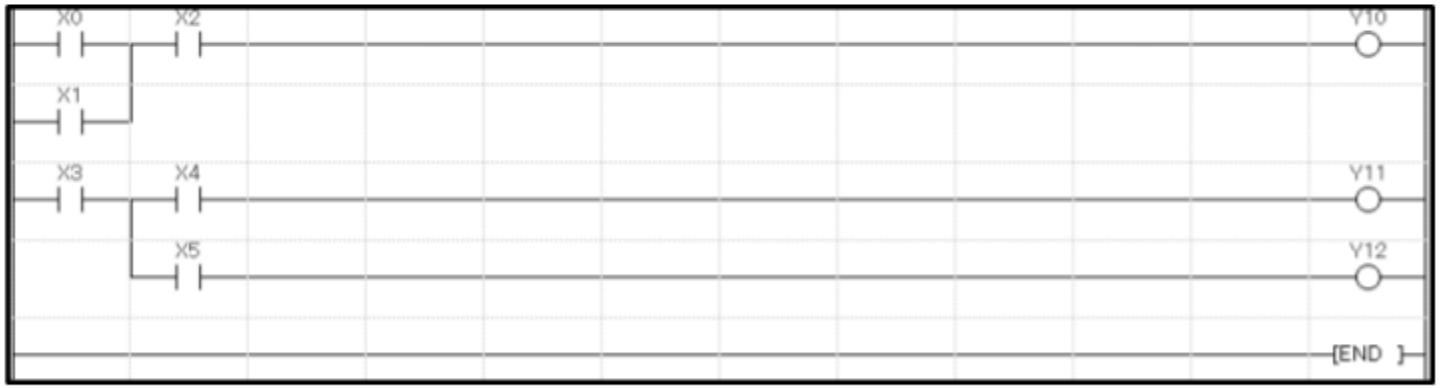
Ví dụ chương trình viết bằng ST

```
Y10 := (X0 OR X1) AND X2;        (* Thực hiện đầu tiên *)
Y11 := X3 AND X4;                (* Thực hiện thứ hai *)
Y12 := X3 AND X5;    (* Thực hiện thứ ba. Không yêu cầu câu lệnh END ở cuối.*)
```

Thực hiện lặp lại

*Cần có câu lệnh END ở cuối chương trình viết bằng LD nhưng không cần câu lệnh này khi viết chương trình bằng ST.

Chương trình bậc thang sau đây biểu diễn cùng một quá trình vận hành như ví dụ chương trình viết bằng ST ở trên.



Thực hiện lặp lại

Cũng như trường hợp viết bằng LD, các lệnh viết bằng ST được thực hiện lặp lại bằng cách trở lại lệnh đầu tiên sau khi đến lệnh cuối cùng.

2.5 Tổng kết

Nội dung của chương này như sau:

- Chương trình viết bằng ST cơ bản
- Định dạng của câu lệnh gán
- Ký hiệu số
- Trình tự thực hiện chương trình
- Chú thích

Những điểm quan trọng cần cân nhắc:

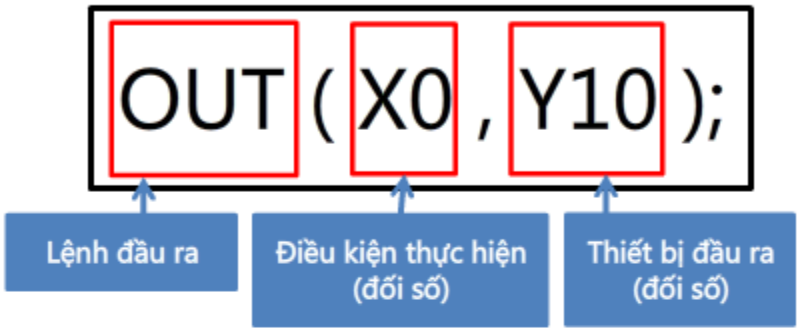
Chương trình viết bằng ST cơ bản	• Câu lệnh là thành tố tối thiểu của chương trình viết bằng ST. • Mỗi câu lệnh kết thúc bằng dấu chấm phẩy (;). • Kết hợp các câu lệnh sẽ tạo nên một chương trình.
Định dạng của câu lệnh gán	• Sử dụng toán tử gán (:=) để gán.
Ký hiệu số	• Các kiểu ký tự số trong ST • Sử dụng "1" và "0" cho các giá trị bit trong ST thay cho ký hiệu "ON" và "OFF". • Trong ST, cũng có thể biểu diễn các giá trị bit dưới dạng "TRUE" và "FALSE".
Trình tự thực hiện chương trình	• Chương trình tạo bằng ST được thực hiện theo thứ tự từ trên xuống dưới. • Cũng như chương trình viết bằng LD, chương trình viết bằng ST được thực hiện lặp lại, trở lại đầu chương trình sau khi đến cuối chương trình.
Chú thích	• Thêm chú thích vào chương trình sẽ giúp quy trình vận hành dễ hiểu hơn. • Chú thích được đặt giữa hai dấu hoa thị (* *).

Chương 3 Tạo chương trình điều khiển I/O

Chương này mô tả cách tạo chương trình điều khiển I/O bằng ST.

3.1 Chương trình điều khiển I/O

Dưới đây là một ví dụ chương trình điều khiển I/O của một bộ điều khiển khả trình.



"OUT" là lệnh đầu ra. Đối số chỉ định điều kiện thực hiện và thiết bị xuất đầu ra.
Khi thỏa mãn điều kiện thực hiện X0, thiết bị Y10 sẽ bật.

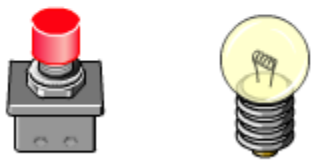
Nhấp vào công tắc đầu vào hiển thị bên dưới. Công tắc đầu vào X0 sẽ bật lên.

- Khi công tắc đầu vào X0 bật, đèn đầu ra Y10 sẽ bật.
- Khi công tắc đầu vào X0 tắt, đèn đầu ra Y10 sẽ tắt.

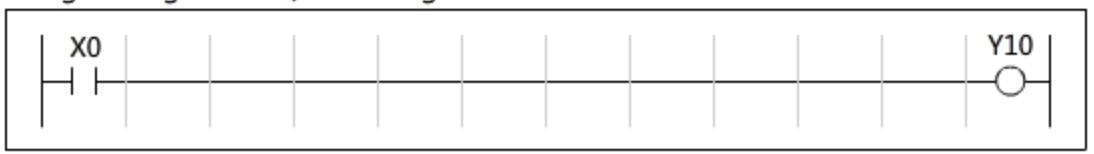
Ví dụ về chương trình điều khiển I/O viết bằng ST

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

Công tắc đầu vào X0 Đèn đầu ra Y10



Cùng chương trình được viết bằng LD



3.1 Chương trình điều khiển I/O

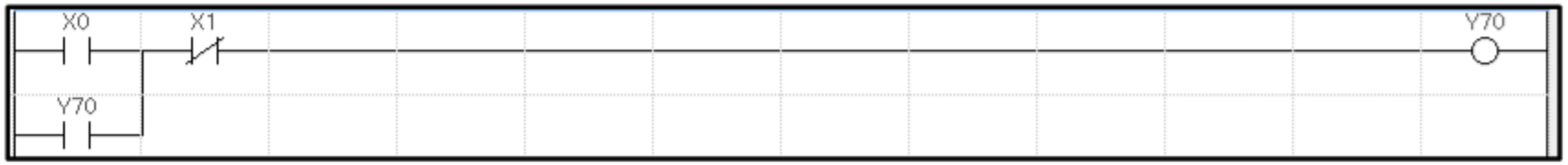
Tương tự như LD, ngoài lệnh OUT còn có nhiều lệnh khác như lệnh điều khiển I/O và lệnh xử lý dữ liệu.
Tham khảo hướng dẫn lập trình của bộ điều khiển khả trình để biết thêm thông tin về các lệnh có trong ST.

Lưu ý rằng viết "OUT(X0, Y10);" thành "Y10 := X0;" cũng cho cùng kết quả vận hành.

```
Y10 := X0;  (* Vận hành như "OUT(X0, Y10);" *)
```

3.2 Kết hợp nhiều điều kiện

Chương trình bậc thang dưới đây biểu diễn một mạch tự duy trì.



Có thể viết cùng chương trình này bằng ST như sau.

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

Toán tử logic

Như minh họa ở trên, các toán tử logic được sử dụng để kết hợp nhiều điều kiện trong ST.

Bảng dưới đây liệt kê các toán tử logic.

Toán tử	Ý nghĩa
OR	Phép OR logic
AND	Phép AND logic
NOT	Phủ định logic
XOR	Phép OR có loại trừ

3.3 Xác định ý nghĩa cho biến số

Khi sử dụng ST với các bộ điều khiển khả trình MELSEC, có thể gán cả thiết bị và nhãn làm tên hiệu cho các biến số.
Người dùng có thể sử dụng nhãn tùy theo ứng dụng.

Khi gán nhãn liên quan đến ứng dụng, quá trình vận hành sẽ dễ hiểu hơn.

```
Y10 := (X0 OR X1) AND X2;     (* Viết sử dụng tên thiết bị *)
```

```
Lamp := (Switch0 OR Switch1) AND Switch2;     (* Viết sử dụng nhãn *)
```

Có thể đặt tên nhãn bằng phần mềm kỹ thuật MELSOFT.

Các ví dụ chương trình tiếp theo trong khóa học này được mô tả bằng nhãn.

3.4 Tổng kết

Nội dung của chương này như sau:

Ví dụ chương trình điều khiển I/O

- Các toán tử logic được sử dụng để kết hợp nhiều điều kiện trong ST.
- Có thể sử dụng cả tên thiết bị và nhãn làm tên biến số.

Những điểm quan trọng cần cân nhắc:

Kết hợp nhiều điều kiện	• Các toán tử logic được sử dụng để kết hợp các điều kiện trong ST.
Xác định ý nghĩa cho biến số	• Khi gán nhãn liên quan đến ứng dụng, quá trình vận hành sẽ dễ hiểu hơn.

Chương 4 Phép toán số học

Chương này mô tả cách tạo chương trình phép toán số học.

- Mô tả các phép toán số học
- Chỉ định kiểu dữ liệu tương ứng với các khoảng số
- Đặt tên biến số để tránh kiểu dữ liệu không thống nhất

4.1 Phép toán số học cơ bản

Chương trình ví dụ này tính tổng khối lượng sản xuất của hai dây chuyền sản xuất riêng biệt.
Bên phải đẳng thức là một phép toán số học chứa các biến số và toán tử số học.

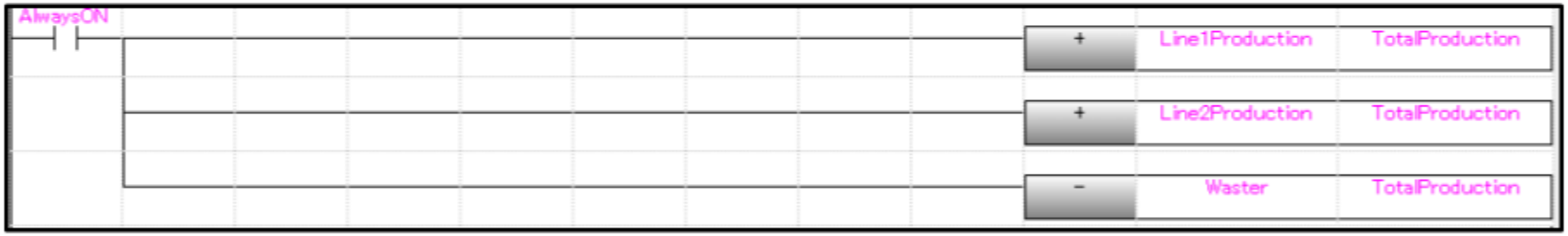
Chương trình số học ví dụ viết bằng ST

Toán tử cộng

Toán tử trừ

```
TotalProduction := Line1Production + Line2Production - Waster;
(* Tính tổng khối lượng sản xuất của hai dây chuyền sản xuất, trừ số lượng sản phẩm bị lỗi khỏi tổng, và gán giá trị thu được. *)
```

Bên dưới là cùng chương trình này khi viết bằng LD.



Như minh họa ở trên, khi viết chương trình trong Sơ đồ bậc thang phải dùng 3 dòng, nhưng với ST có thể viết trong 1 dòng.

Bảng dưới đây liệt kê các toán tử số học cơ bản.

Toán tử	Ý nghĩa
+	Phép cộng
-	Phép trừ
*	Phép nhân
/	Phép chia

4.2 Kiểu dữ liệu của biến số

Phải chỉ định kiểu dữ liệu cho mỗi biến số để xác định phạm vi dữ liệu cần xử lý.
Kiểu dữ liệu cho các giá trị số dùng trong ST là kiểu bit, số nguyên và số thực.

Trong các kiểu dữ liệu dùng trong ST, bảng dưới đây chỉ trình bày các kiểu dữ liệu được sử dụng trong khóa học này.

Kiểu dữ liệu		Phạm vi dữ liệu
Bit		Trạng thái BẬT/TẮT của thiết bị bit và trạng thái đúng/sai của kết quả thực hiện
Số nguyên	Từ (không dấu)	0 - 65.535
	Từ (có dấu)	-32.768 - 32.767
	Từ kép (không dấu)	0 - 4.294.967.295
	Từ kép (có dấu)	-2.147.483.648 - 2.147.483.647

Khi sử dụng kiểu số nguyên, chọn kiểu từ hoặc từ kép tùy theo phạm vi dữ liệu và chọn kiểu có dấu hoặc không dấu tùy vào sự cần thiết của việc xử lý các giá trị âm.
Chỉ định kiểu dữ liệu cho một biến số khi đặt tên nhãn bằng phần mềm kỹ thuật MELSOFT.

4.3 Tên biến số đại diện cho kiểu dữ liệu

Sử dụng các kiểu dữ liệu khác nhau ở bên trái và bên phải của một đẳng thức gán có thể gây lỗi biên dịch hoặc cho kết quả ngoài dự kiến.
Bên dưới là ví dụ về một trường hợp như vậy.

```
ValueA := ValueB;      (* ValueA: Kiểu số nguyên từ  ValueB: Kiểu số nguyên từ kép *)
```

Không thể gán kiểu số nguyên từ kép cho kiểu số nguyên từ. Tuy nhiên, trong trường hợp này, không nhận ra được kiểu dữ liệu.

Có thể thêm các tiền tố đại diện cho kiểu dữ liệu vào tên biến số để dễ dàng nhìn và nhận ra kiểu dữ liệu.
Cách đặt tên biến số này được gọi là ký hiệu Hungary.

Kiểu dữ liệu		Phạm vi dữ liệu	Tiền tố	Mở rộng tiền tố
Bit		Trạng thái BẬT/TẮT của thiết bị bit và trạng thái đúng/sai của kết quả thực hiện	b	**B**it (bit)
Số nguyên	Từ (không dấu)	0 - 65.535	u	**u**nsigned word (từ không dấu)
	Từ (có dấu)	-32.768 - 32.767	w	signed **w**ord (từ có dấu)
	Từ kép (không dấu)	0 - 4.294.967.295	ud	**u**nsigned **d**ouble-word (từ kép không dấu)
	Từ kép (có dấu)	-2.147.483.648 - 2.147.483.647	d	signed **d**ouble-word (từ kép có dấu)

Có thể sử dụng cách ký hiệu Hungary để viết chương trình ví dụ ở đầu trang này như sau:

```
wValueA := dValueB;  (* Không thể gán biến số kiểu từ kép cho biến số kiểu từ. *)
```

Bằng cách sử dụng ký hiệu Hungary, có thể phát hiện kiểu dữ liệu không thống nhất trong khi viết chương trình.

Trong phần còn lại của khóa học, tên biến số trong ví dụ sẽ được viết bằng ký hiệu Hungary.

4.4 Tổng kết

Nội dung của chương này như sau:

- Mô tả các phép toán số học
- Chỉ định kiểu dữ liệu tương ứng với các khoảng số
- Thêm tên biến số đại diện cho kiểu dữ liệu

Những điểm quan trọng cần cân nhắc:

Phép toán số học cơ bản	• Có thể sử dụng các toán tử thường gặp ở những ngôn ngữ lập trình nói chung trong ST để biểu diễn các phép tính.
Kiểu dữ liệu của biến số	• Phải chỉ định kiểu dữ liệu cho mỗi biến số để xác định phạm vi dữ liệu cần xử lý.
Thêm tên biến số đại diện cho kiểu dữ liệu	• Mô tả tên biến số bằng ký hiệu Hungary giúp xác định sự không thống nhất trong kiểu dữ liệu của biến số khi viết chương trình.

Chương 5 Phân nhánh có điều kiện

Chương trình điều khiển cũng chứa các phần mã mà theo đó quá trình xử lý thực tế sẽ thay đổi tùy theo điều kiện cụ thể. Chương này mô tả về phân nhánh có điều kiện.

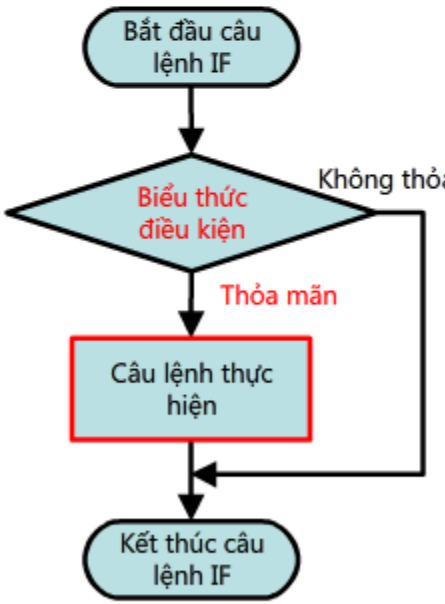
5.1 Phân nhánh có điều kiện (IF)

Câu lệnh IF được sử dụng trong phân nhánh có điều kiện. Câu lệnh IF được mô tả như sau.

```
IF conditional expression THEN
    Execution statement;        (* Thực hiện câu lệnh nếu biểu thức điều kiện được thỏa mãn. *)
END_IF;                         (* Phải đặt END_IF; ở cuối câu lệnh IF. *)
```

Trong chương trình ví dụ này, câu lệnh được thực hiện khi biểu thức điều kiện được thỏa mãn. Câu lệnh sẽ không được thực hiện khi biểu thức điều kiện không được thỏa mãn.

Hình sau đây minh họa dòng vận hành trong chương trình ví dụ này.

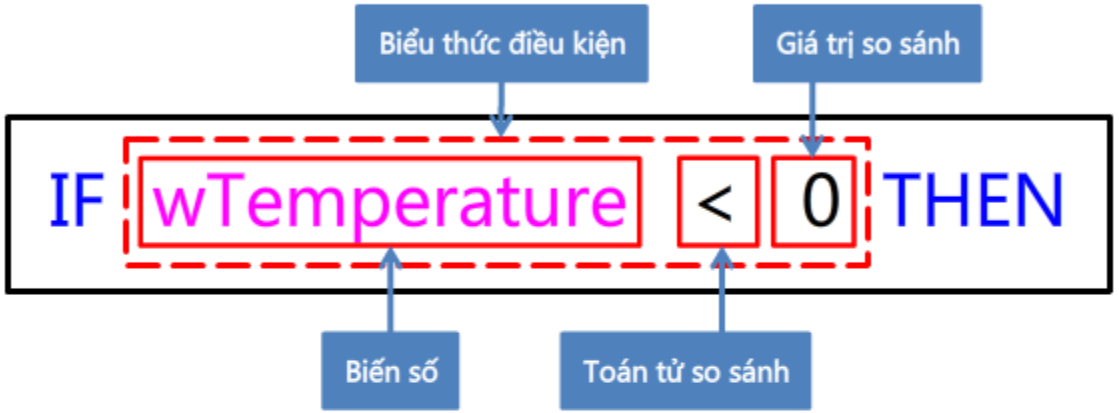


Ví dụ sau đây minh họa việc phân nhánh chương trình bằng cách so sánh giá trị của các biến số.
Trong chương trình ví dụ, bộ phát nhiệt sẽ bật khi nhiệt độ trong bảng điều khiển xuống dưới 0 độ.

```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Bộ phát nhiệt sẽ bật khi nhiệt độ trong bảng điều khiển xuống dưới 0 độ. *)
END_IF;
```

5.1.1 Viết biểu thức điều kiện

Trang trước đã mô tả biểu thức điều kiện "wTemperature < 0", có nghĩa là "khi giá trị của biến số wTemperature thấp hơn 0". Giống như biểu thức này, biểu thức điều kiện sử dụng các toán tử so sánh để biểu diễn mối quan hệ giữa các biến số và giá trị so sánh.



Ở bên trái và bên phải toán tử so sánh, các giá trị được viết dưới dạng biến số hoặc hằng số để so sánh.

Không chỉ so sánh biến số và hằng số, có thể viết biểu thức điều kiện để so sánh các biến số và thực hiện các phép toán logic cho kết quả so sánh hoặc biến số kiểu bit.

Biến số so sánh

- uValue1 <= uValue2

Phép toán logic đối với hai kết quả so sánh

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

Phép toán logic đối với hai biến số kiểu bit

- bSwitch0 OR bSwitch1

Bảng dưới đây liệt kê các loại toán tử so sánh.

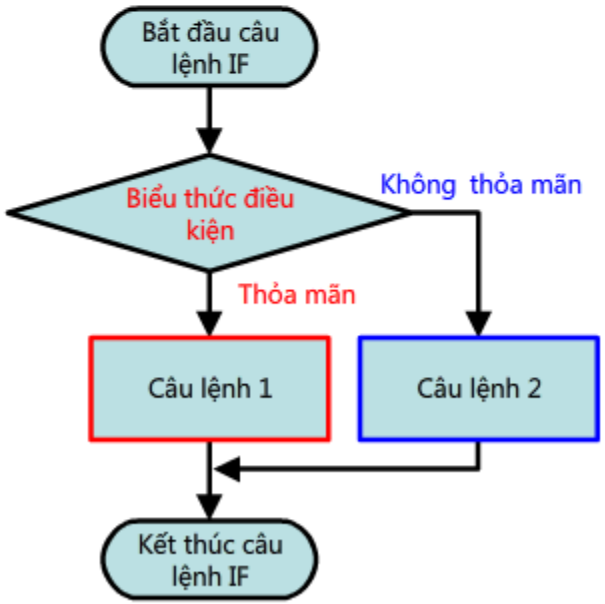
Toán tử	Ý nghĩa
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
=	Bằng
<>	Không bằng

5.1.2 Phân nhánh ngoại lệ cho câu lệnh IF (ELSE)

Câu lệnh IF đơn giản (xem mục 5.1) được sử dụng để thực hiện câu lệnh khi một biểu thức điều kiện được thỏa mãn. Để thực hiện một câu lệnh khác khi biểu thức điều kiện không được thỏa mãn, ta sử dụng câu lệnh ELSE.

```
IF conditional expression THEN
  Execution statement 1;          (* Thực hiện câu lệnh 1 nếu biểu thức điều kiện được thỏa mãn. *)
  ELSE
  Execution statement 2;          (* Thực hiện câu lệnh 2 nếu biểu thức điều kiện không được thỏa mãn *)
END_IF;
```

Hình sau đây minh họa dòng vận hành khi sử dụng câu lệnh ELSE.



Chương trình ví dụ sau đây thực hiện các câu lệnh khác nhau tùy theo điều kiện có được thỏa mãn hay không.

Chương trình ví dụ ở mục 5.1 có nhược điểm là bộ phát nhiệt sẽ tiếp tục tăng nhiệt độ kể cả khi đã đạt 0 độ. Tuy nhiên, chương trình sau đây sẽ tắt bộ phát nhiệt khi "wTemperature" vượt quá 0 độ.

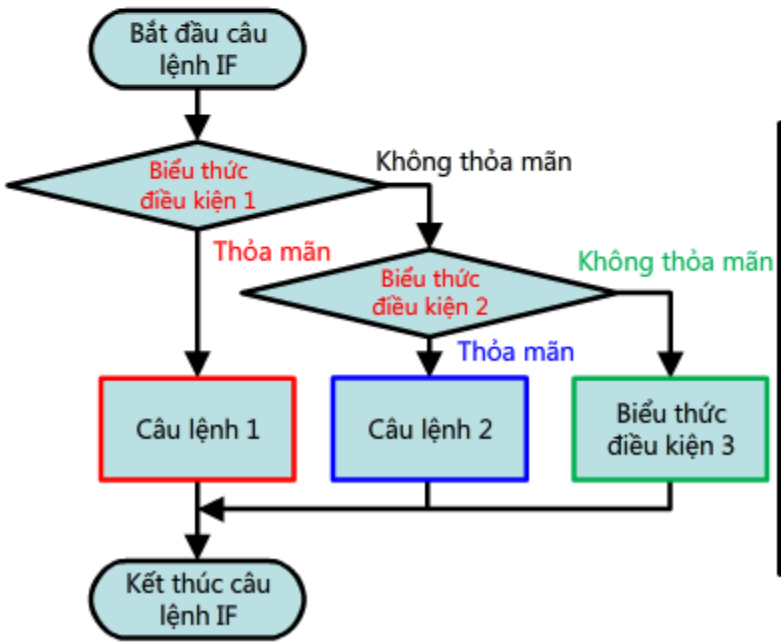
```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Bật bộ phát nhiệt khi nhiệt độ xuống dưới 0 độ. *)
  ELSE
  bHeater := 0;  (* Tắt bộ phát nhiệt khi nhiệt độ đạt hoặc vượt quá 0 độ. *)
END_IF;
```

5.1.3 Phân nhánh bổ sung cho câu lệnh IF (ELSIF)

Câu lệnh ELSE được sử dụng để thực hiện một câu lệnh khác khi biểu thức điều kiện không được thỏa mãn.
Có thể thêm một nhánh điều kiện khác bằng cách sử dụng câu lệnh ELSIF, có nghĩa là nếu biểu thức điều kiện trước không được thỏa mãn thì một biểu thức điều kiện khác sẽ được kiểm tra.

```
IF Conditional expression 1 THEN
    Execution statement 1;      (* Thực hiện câu lệnh 1 nếu biểu thức điều kiện 1 được thỏa mãn. *)
  ELSIF Conditional expression 2 THEN
    Execution statement 2;      (* Thực hiện câu lệnh 2 nếu biểu thức điều kiện 1 không được thỏa mãn và biểu thức điều kiện 2 được thỏa mãn. *)
  ELSE
    Execution statement 3;      (* Thực hiện câu lệnh 3 nếu các biểu thức điều kiện 1 và 2 không được thỏa mãn. *)
END_IF;
```

Hình sau đây minh họa dòng vận hành khi sử dụng câu lệnh ELSEIF.



Câu lệnh ELSIF được thêm vào ví dụ chương trình minh họa ở mục 5.1.2 để xử lý trường hợp nhiệt độ vượt quá 40 độ.

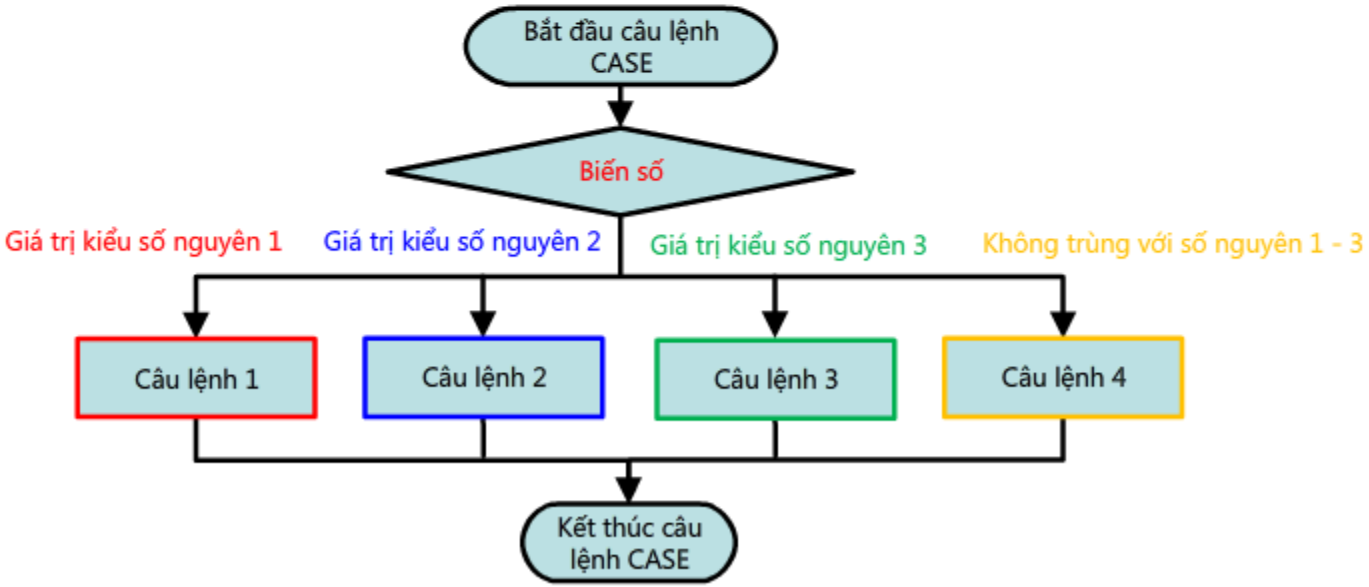
```
IF wTemperature < 0 THEN
   bHeater := 1; (* Bật bộ phát nhiệt khi nhiệt độ xuống dưới 0 độ. *)
   bCooler := 0; (*  Tắt bộ làm mát khi nhiệt độ xuống dưới 0 độ. *)
   ELSIF 40 < wTemperature THEN
   bHeater := 0; (* Tắt bộ phát nhiệt nếu nhiệt độ vượt quá 40 độ. *)
   bCooler := 1; (* Bật bộ làm mát nếu nhiệt độ vượt quá 40 độ. *)
   ELSE
   bHeater := 0; (*  Tắt bộ phát nhiệt nếu các điều kiện trước không được thỏa mãn. *)
   bCooler := 0; (*  Tắt bộ làm mát nếu các điều kiện trước không được thỏa mãn. *)
END_IF;
```

5.2 Phân nhánh có điều kiện dựa vào giá trị kiểu số nguyên (CASE)

Câu lệnh IF được sử dụng để phân nhánh dựa vào việc các biểu thức điều kiện có được thỏa mãn hay không.
Câu lệnh CASE được sử dụng để phân nhánh dựa vào các giá trị kiểu số nguyên.
Hình sau đây minh họa cách viết một câu lệnh CASE.

```
CASE Variable OF
    Integer value 1: Execution statement 1;      (* Thực hiện câu lệnh 1 khi biến số trùng với giá trị kiểu số nguyên 1. *)
    Integer value 2: Execution statement 2;      (* Thực hiện câu lệnh 2 khi biến số trùng với giá trị kiểu số nguyên 2. *)
    Integer value 3: Execution statement 3;      (* Thực hiện câu lệnh 3 khi biến số trùng với giá trị kiểu số nguyên 3. *)
    ELSE Execution statement 4;          (* Thực hiện câu lệnh 4 nếu biến số không trùng với bất kỳ giá trị kiểu số nguyên nào. *)
END_CASE;                                (* Phải đặt "END_CASE;" ở cuối câu lệnh CASE. *)
```

Hình sau đây minh họa dòng vận hành khi sử dụng câu lệnh CASE.



5.2.1 Chương trình ví dụ về câu lệnh CASE

Việc thực hiện câu lệnh CASE được mô tả bằng quá trình vận hành chương trình ví dụ.

Nhấp vào để đến trang tiếp theo.
Để xem lại ảnh động, nhấp vào nút "Phát".

Phát


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

Trọng lượng	uSize	Cỡ
0 đến 20 kg	1	M
21 đến 30 kg	2	L
31 đến 40 kg	3	XL
41 kg trở lên	4	Oversize

5.3 Tổng kết

Nội dung của chương này như sau:

- Phân nhánh có điều kiện với câu lệnh IF
- Viết biểu thức điều kiện
- Phân nhánh có điều kiện dựa vào giá trị kiểu số nguyên (câu lệnh CASE)

Những điểm quan trọng cần cân nhắc:

Câu lệnh IF	• Với câu lệnh IF, chương trình được phân nhánh khi một biểu thức điều kiện được thỏa mãn. • Câu lệnh ELSE được sử dụng để phân nhánh khi biểu thức điều kiện không được thỏa mãn. • Câu lệnh ELSIF được sử dụng để thêm một phân nhánh khác khi biểu thức điều kiện trong câu lệnh IF không được thỏa mãn.
Biểu thức điều kiện	• Biểu thức điều kiện sử dụng toán tử so sánh để biểu diễn mối quan hệ giữa các biến số và giá trị so sánh.
Câu lệnh CASE	• Câu lệnh CASE được sử dụng để phân nhánh dựa vào các giá trị kiểu số nguyên.

Chương 6 Lưu trữ và xử lý dữ liệu

Cũng như đối với các ứng dụng điều khiển I/O, ngày nay, bộ điều khiển khả trình được sử dụng để xử lý khối lượng dữ liệu lớn và đóng vai trò cốt lõi trong hệ thống sản xuất.
Để có thể xử lý khối lượng dữ liệu lớn, dữ liệu phải được lưu trữ và được đọc khi cần.
Chương này mô tả cách viết những chương trình ngắn gọn để lưu trữ và xử lý dữ liệu.

- Sử dụng mảng để lập trình tự và tổ chức các biến số.
- Sử dụng cấu trúc dữ liệu để tổ chức các biến số liên quan.
- Chương trình xử lý theo vòng lặp sử dụng câu lệnh FOR để xử lý mảng một cách hiệu quả.

Có thể tạo các chương trình ngắn gọn để lưu trữ và xử lý dữ liệu bằng cách sử dụng mảng, cấu trúc dữ liệu và câu lệnh FOR.

6.1 Lập trình tự và lưu trữ dữ liệu (Mảng)

Bằng cách sử dụng mảng, ta có thể xử lý nhiều giá trị bằng một biến số.
Trong ví dụ sau, dữ liệu về khối lượng sản xuất tại một nhà máy sản xuất ô tô được lưu trữ theo quốc gia điểm đến.

Điểm đến	Quốc gia A	Quốc gia B	Quốc gia C
Khối lượng sản xuất	35	75	65

Dữ liệu về khối lượng sản xuất theo quốc gia điểm đến được gán cho một biến số. Nếu không sử dụng mảng, ta sẽ phải tạo một biến số cho mỗi điểm đến.
Tuy nhiên, nhờ sử dụng mảng, ta có thể gán và lưu trữ khối lượng sản xuất cho nhiều điểm đến trong một biến số.

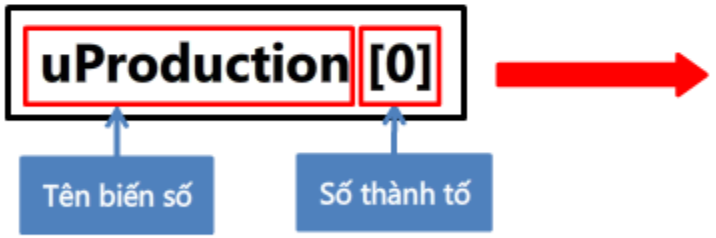
Không sử dụng mảng

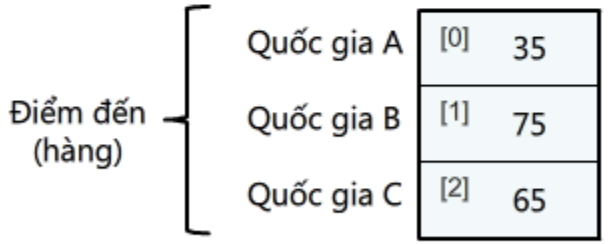
uProductionA
uProductionB
uProductionC

Sử dụng mảng

uProduction

Từng biến số trong mảng được chỉ định bằng các số thành tố. Số thành tố bắt đầu từ số không [0].



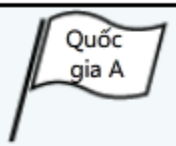
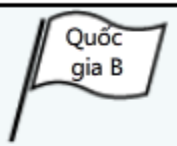
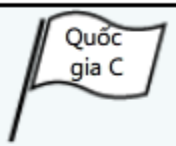


Trong ví dụ chương trình sau, biến số của khối lượng sản xuất theo kế hoạch cho Quốc gia A đã được gán.

```
uShowProductionPlan := uProduction[0];
(* Chỉ định số thành tố cho quốc gia A. *)
```

6.1.1 Mảng ma trận

Tiếp theo, dữ liệu về màu sơn được sử dụng cùng với dữ liệu về điểm đến.

Điểm đến									
Màu sơn									
Khối lượng sản xuất	10	5	20	15	40	20	25	30	10
	Tổng cộng 35			Tổng cộng 75			Tổng cộng 65		

Như minh họa trong bảng sau, có thể tách biệt và lưu trữ dữ liệu theo màu sơn (cột) cho từng quốc gia điểm đến (hàng).

Màu sơn (cột)

Điểm đến (hàng)	Đỏ	Vàng	Lam
Quốc gia A	[0,0] 10	[0,1] 5	[0,2] 20
Quốc gia B	[1,0] 15	[1,1] 40	[1,2] 20
Quốc gia C	[2,0] 25	[2,1] 30	[2,2] 10

Số thành tố đại diện cho điểm đến

Số thành tố đại diện cho màu sơn

Mảng tổ chức dữ liệu thành các hàng và cột như thế này được gọi là mảng ma trận.
Các số thành tố đại diện cho hàng và cột được phân tách bằng dấu phẩy.

Biến số dạng mảng (mảng ma trận)

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

6.1.2 Gán mảng ma trận

Chương trình ví dụ sau sử dụng mảng ma trận để gán số lượng ô tô màu vàng cần sản xuất gấp ngoài khối lượng sản xuất theo kế hoạch cho Quốc gia B.

```
uAdditionalProduction := 5;
uProduction[1,1] := uProduction[1,1] + uAdditionalProduction;
(* Cộng khối lượng sản xuất thêm (5 xe) vào khối lượng sản xuất theo kế hoạch ban đầu. *)
```

Điểm đến	Quốc gia A			Quốc gia B			Quốc gia C		
Màu sơn									
Khối lượng sản xuất	10	5	20	15	40	20	25	30	10
	Tổng cộng 35			Tổng cộng 75			Tổng cộng 65		

Màu sơn (cột)

Điểm đến (hàng)	Đỏ	Vàng	Lam
Quốc gia A	[0,0] 10	[0,1] 5	[0,2] 20
Quốc gia B	[1,0] 15	[1,1] 40 -> 45	[1,2] 20
Quốc gia C	[2,0] 25	[2,1] 30	[2,2] 10

6.1.3 Xử lý thông tin lưu trữ trong mảng ma trận

Chương trình ví dụ sau sử dụng mảng ma trận để tính toán tổng khối lượng sản xuất theo kế hoạch đối với ô tô ở mọi màu sơn cho Quốc gia C và gán giá trị đó cho một biến số.

```
uProductionToday := uProduction[2,0] + uProduction[2,1] +uProduction[2,2];
(* Tính toán tổng khối lượng sản xuất theo kế hoạch trong ngày đối với ô tô ở mọi màu sơn cho Quốc gia C và gán giá trị cho "uProductionToday". *)
```

Điểm đến	Quốc gia A			Quốc gia B			Quốc gia C		
Màu sơn									
Khối lượng sản xuất	10	5	20	15	45	20	25	30	10
	Tổng cộng 35			Tổng cộng 80			Tổng cộng 65		

Màu sơn (cột)

Điểm đến (hàng)	Đỏ	Vàng	Lam
Quốc gia A	[0,0] 10	[0,1] 5	[0,2] 20
Quốc gia B	[1,0] 15	[1,1] 45	[1,2] 20
Quốc gia C	[2,0] 25	[2,1] 30	[2,2] 10

6.2 Tạo vòng lặp (FOR)

Dưới đây vẫn là chương trình ví dụ ở trang trước (khối lượng sản xuất trong ngày theo kế hoạch đã được gán).

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

Với ví dụ chương trình này, khi số màu sơn tăng, các biến số sẽ được bổ sung. Khi đó, biểu thức sẽ dài hơn và khó đọc hơn.

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

Trong trường hợp này, có thể sử dụng câu lệnh vòng lặp để viết mã gọn hơn.

Các câu lệnh vòng lặp bao gồm câu lệnh FOR, WHILE và REPEAT. Khóa học này sẽ trình bày câu lệnh FOR.

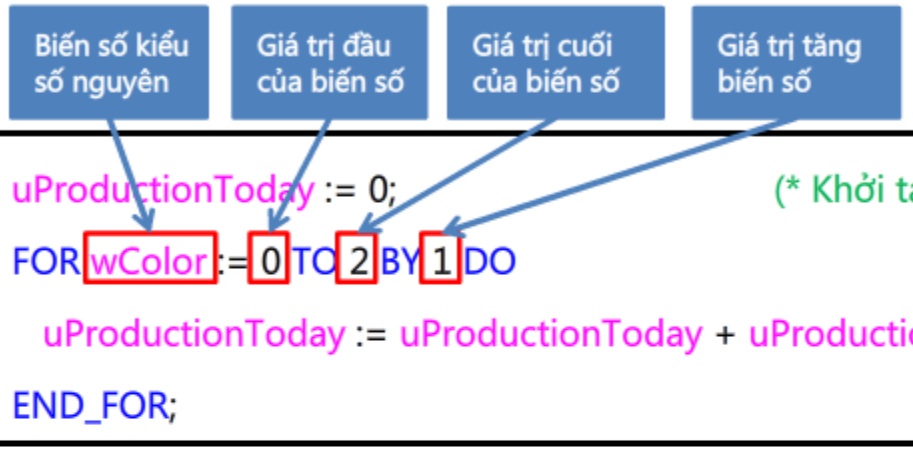
Câu lệnh FOR được mô tả như sau.

```
FOR variable := initial value TO final value BY increments DO
   Execution statement; (* Thực hiện câu lệnh trong vòng lặp đến khi biến số đạt giá trị cuối cùng. *)
END_FOR;                (* Phải đặt END_FOR; ở cuối câu lệnh FOR.  *)
```

Câu lệnh được lặp lại đến khi biến số đạt giá trị cuối cùng và mã "END_FOR;" được thực hiện.

6.2 Tạo vòng lặp (FOR)

Chương trình ví dụ sau sử dụng câu lệnh FOR để thu được khối lượng sản xuất theo kế hoạch đối với ô tô ở mọi màu sơn cho Quốc gia C.



Khi sử dụng câu lệnh FOR, biến số "wColor" tăng thêm một đơn vị từ giá trị ban đầu là 0 và câu lệnh được lặp lại đến khi biến số đạt giá trị cuối cùng là 2.
Biến số "wColor" được chỉ định là số thành tố thứ hai trong mảng "uProduction" mô tả trong câu lệnh thực hiện.
Giá trị của biến số "wColor" sẽ tăng lên mỗi khi câu lệnh lặp lại. Trong mỗi lần đó, khối lượng sản xuất theo kế hoạch đối với ô tô ở mỗi màu sơn sẽ được cộng vào để thu được giá trị tổng.

Chương trình ví dụ này được thực hiện theo vòng lặp ba lần. (Lần thứ nhất: đỏ [0] => Lần thứ hai: vàng [1] => Lần thứ ba: lam [2])

Quá trình vận hành chương trình này được minh họa ở trang tiếp theo.

6.2 Tạo vòng lặp (FOR)

Việc thực hiện câu lệnh FOR được mô tả bằng quá trình vận hành ví dụ chương trình.

Mảng khối lượng sản xuất dự tính

	Đỏ	Vàng	Lam
Quốc gia A	[0,0] 10	[0,1] 5	[0,2] 20
Quốc gia B	[1,0] 15	[1,1] 45	[1,2] 20
Quốc gia C	[2,0] 25	[2,1] 30	[2,2] 10

Nhấp vào  để đến trang tiếp theo.
Để xem lại ảnh động, nhấp vào nút "Phát".

Phát

Number of repetition of the loop: 3

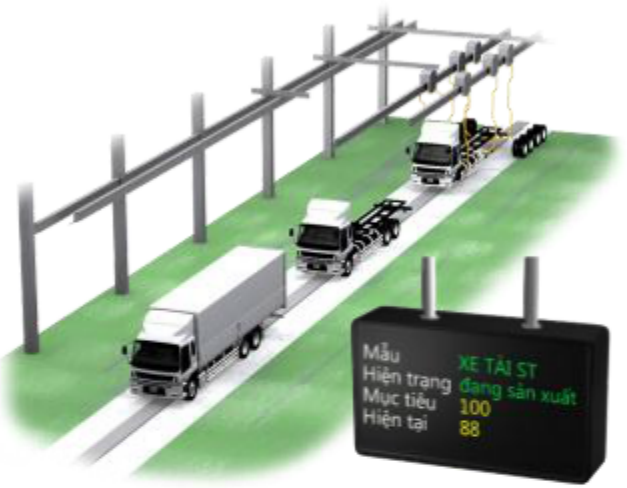
```
uProductionToday := 0;

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

6.3 Lưu trữ dữ liệu liên quan (Cấu trúc)

Cấu trúc cho phép một tên biến số đại diện cho nhiều biến số liên quan.
Trong ví dụ sau, hiện trạng của một dây chuyền sản xuất ô tô hiển thị trên Andon (bảng hiển thị).

Bảng sau liệt kê tên biến số, giá trị và kiểu dữ liệu tương ứng với từng thông tin được hiển thị.

Thông tin	Tên biến số	Giá trị	Kiểu dữ liệu của biến số
Mẫu	sModel	'XE TẢI ST'	Chuỗi văn bản
Hiện trạng	bStatus	'đang sản xuất'	Kiểu bit
Khối lượng sản xuất mục tiêu trong ngày	uPlanQty	'100'	Kiểu số nguyên Từ (không dấu)
Khối lượng sản xuất hiện tại	uActualQty	'88'	Kiểu số nguyên Từ (không dấu)



Nếu không sử dụng cấu trúc, phải đổi tên biến số cho mỗi dây chuyền khi có nhiều dây chuyền sản xuất.
Dưới đây là ví dụ về tên biến số theo dây chuyền sản xuất.

Dây chuyền sản xuất thứ nhất

s1stLineModel
b1stLineStatus
u1stLinePlanQty
u1stLineActualQty

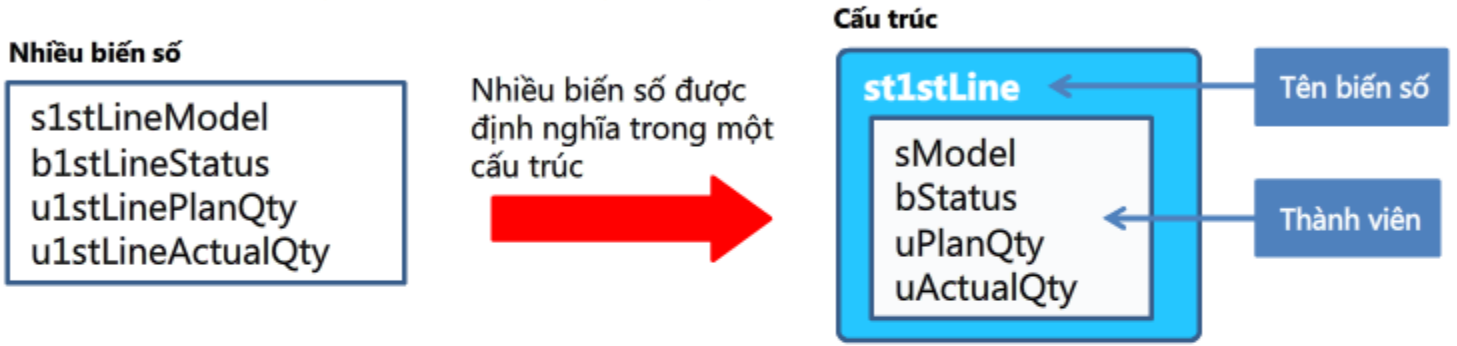
Dây chuyền sản xuất thứ hai

s2ndLineModel
b2ndLineStatus
u2ndLinePlanQty
u2ndLineActualQty

Khi số dây chuyền sản xuất tăng lên, số biến số cần xử lý cũng tăng theo. Khi đó, chương trình sẽ dài hơn và khó đọc hơn.

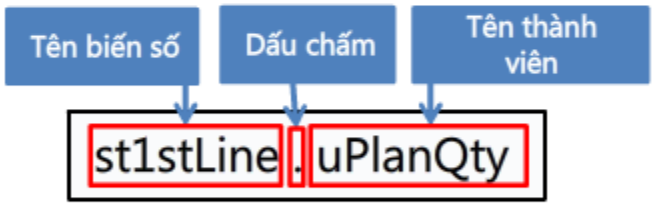
6.3 Lưu trữ dữ liệu liên quan (Cấu trúc)

Sử dụng cấu trúc cho phép một tên biến số đại diện cho nhiều biến số liên quan đến một dây chuyền sản xuất.
Như vậy, cấu trúc được sử dụng để tổ chức, lưu trữ và xử lý dữ liệu theo lô đối với điều kiện và thông số kỹ thuật của các vật thể thực như thiết bị, trang thiết bị và chi tiết gia công.



Structure variable (biến số cấu trúc) chứa tiền tố "st" để thể hiện rằng đây là một cấu trúc.
Các biến số riêng lẻ được định nghĩa bằng cấu trúc được gọi là thành viên. Kiểu dữ liệu của mỗi thành viên có thể khác nhau.

Có thể chỉ định từng thành viên trong mảng cấu trúc sau số thành tố của mảng bằng dấu chấm phía trước tên thành viên.

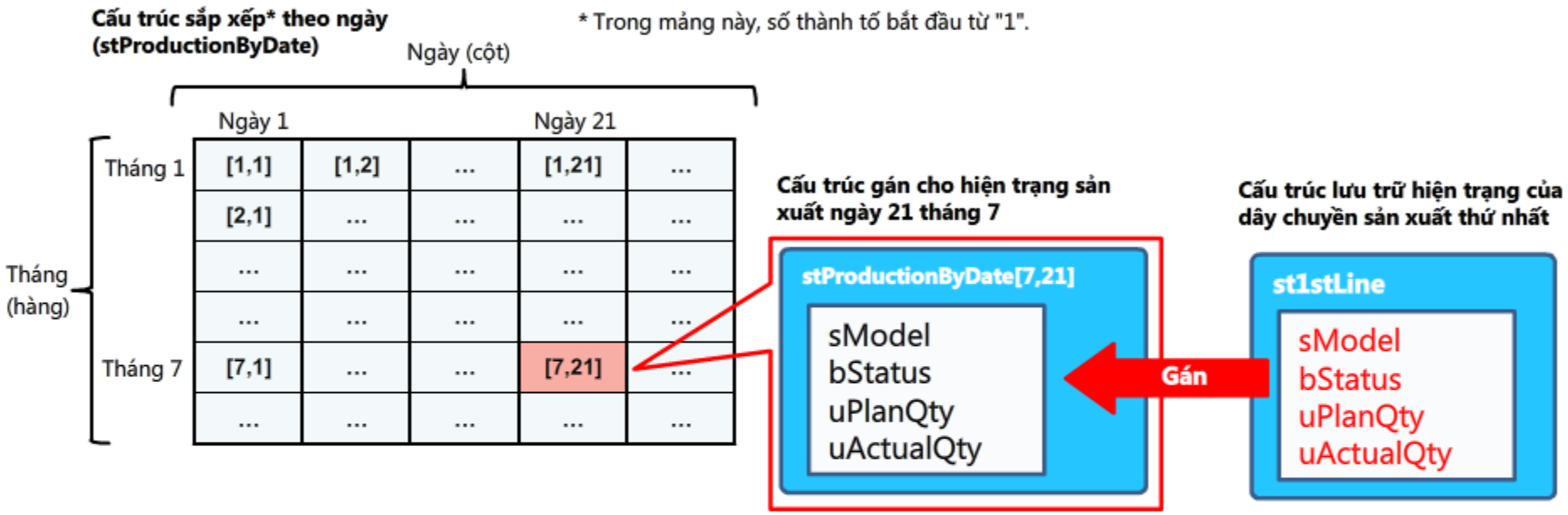


Trong chương trình ví dụ sau, một hằng số được gán cho một thành viên của biến số cấu trúc cho dây chuyền sản xuất thứ nhất.

```
st1stLine.uPlanQty := 150;
 (* Đặt sản lượng mục tiêu trong ngày cho dây chuyền sản xuất thứ nhất là 150. *)
```

6.3.1 Lưu trữ mảng cấu trúc

Cấu trúc có thể được tạo thành mảng.
Trong ví dụ sau, hiện trạng sản xuất được lưu trữ theo ngày.

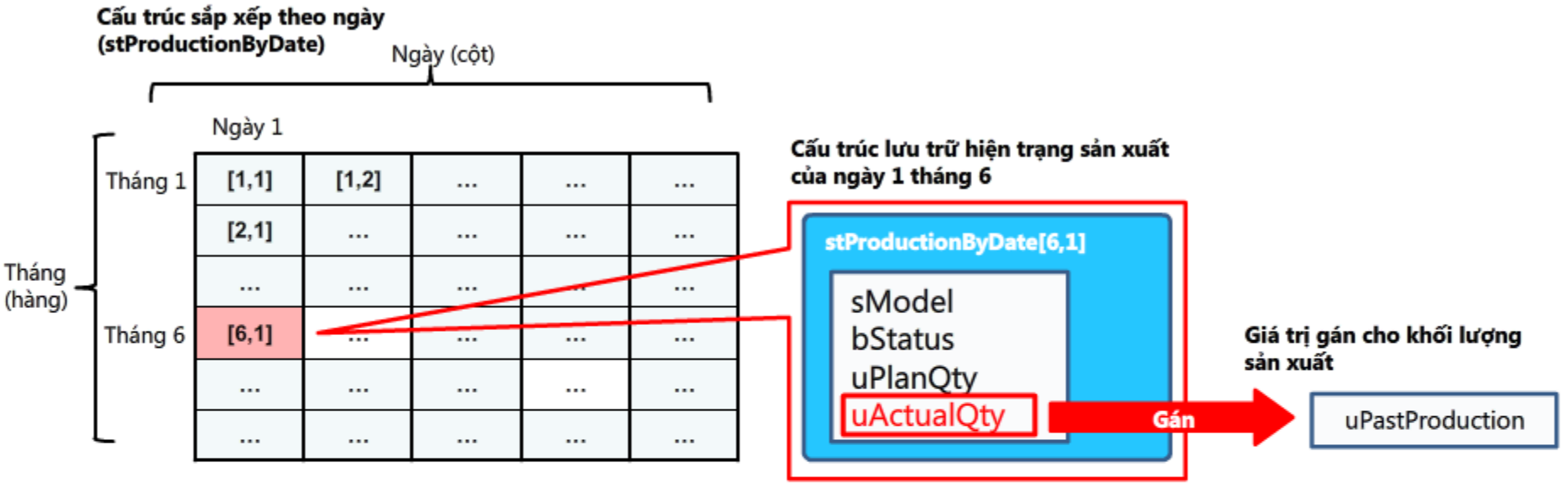



```
stProductionByDate[7,21] := st1stLine;
(* Hiện trạng sản xuất của ngày 21 tháng 7 được lưu trữ trong cấu trúc sắp xếp theo ngày (stProductionByDate). *)
```

Như vậy, khi gán cấu trúc không cần chỉ định riêng từng thành viên.

6.3.2 Đọc mảng cấu trúc

Trong ví dụ sau, khối lượng sản xuất được đọc từ một cấu trúc sắp xếp theo ngày rồi được gán cho một biến số.




```
uPastProduction := stProductionByDate[6,1].uActualQty;
(* Gán khối lượng sản xuất của ngày 1 tháng 6 cho biến số uPastProduction. *)
```

Có thể chỉ định từng thành viên trong mảng cấu trúc bằng cách thêm dấu chấm (.) và tên thành viên vào số thành tố của mảng.

6.4 Tổng kết

Nội dung của chương này như sau:

- Tổng quan về mảng và cách sử dụng
- Xử lý vòng lặp bằng câu lệnh FOR
- Tổng quan về cấu trúc và cách sử dụng

Những điểm quan trọng cần cân nhắc:

Mảng	• Sử dụng mảng có thể xử lý nhiều giá trị bằng một biến số. • Các biến số riêng lẻ trong mảng được chỉ định bằng số thành tố được thêm vào cuối tên biến số.
Câu lệnh FOR	• Câu lệnh vòng lặp được sử dụng trong các chương trình cần vận hành lặp lại. • Câu lệnh FOR được sử dụng để lặp lại quá trình vận hành đến khi các điều kiện kết thúc vận hành theo vòng lặp được thỏa mãn. Câu lệnh trước câu lệnh "END_FOR;" được thực hiện lặp đi lặp lại.
Cấu trúc	• Cấu trúc cho phép một tên biến số đại diện cho nhiều biến số liên quan. Cấu trúc có thể bao gồm các biến số thuộc nhiều kiểu dữ liệu khác nhau. • Có thể chỉ định từng biến số, hay thành viên, được xác định trong cấu trúc bằng cách thêm dấu chấm và tên thành viên vào sau tên biến số của cấu trúc.

Chương 7 Xử lý dữ liệu chuỗi

Trong một số trường hợp, bộ điều khiển khả trình sử dụng dữ liệu chuỗi để gửi lệnh đi hoặc nhận hồi tiếp từ các thiết bị được kết nối như máy đọc mã vạch, bộ điều khiển nhiệt độ hoặc cân điện tử. Vì những mục đích này, cần nối hoặc tách dữ liệu chuỗi theo yêu cầu.

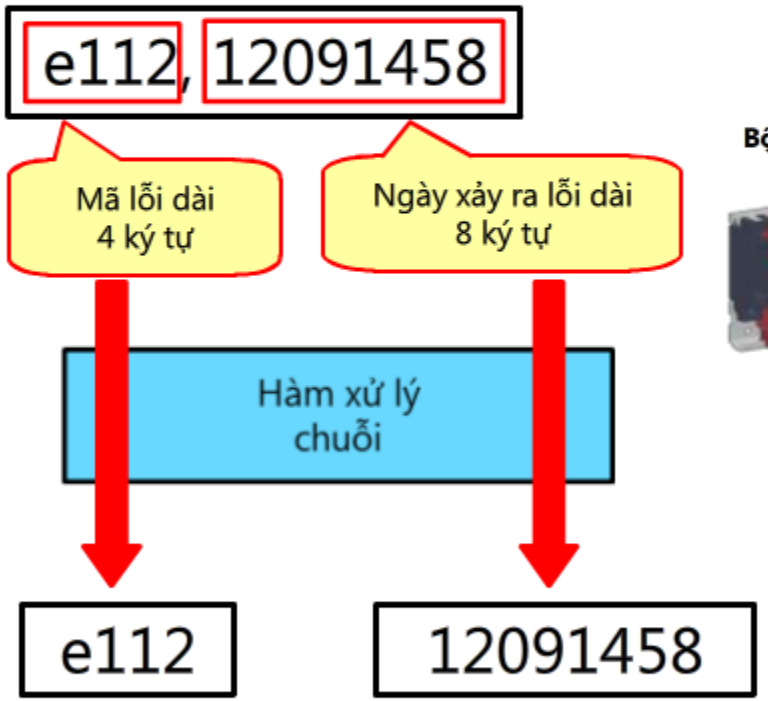
Chương này mô tả cách xử lý dữ liệu chuỗi.

7.1 Ví dụ về xử lý dữ liệu chuỗi

Để ví dụ cho xử lý chuỗi, ví dụ sau đây minh họa một tình huống đọc dữ liệu bằng máy đọc mã vạch.
Các hàm (một loại lệnh) được sử dụng để xử lý chuỗi.

Như minh họa bên dưới, các chuỗi mà máy đọc mã vạch đọc được đều chứa một mã lỗi dài cố định 4 ký tự và dữ liệu về ngày, tháng, giờ và phút dài cố định 8 ký tự.
Ví dụ chương trình xử lý chuỗi sẽ được mô tả bằng hệ thống này.

Ví dụ về dữ liệu chuỗi đọc được từ máy đọc mã vạch



Mã lỗi được tách riêng.
7.3 Tách chuỗi (LEFT)

Ngày và giờ xảy ra lỗi (14:58, ngày 9 tháng 12) được tách riêng.
7.4 Tách chuỗi (MID)

7.2 Gán chuỗi

Trước khi giải thích cách tách chuỗi, mục này sẽ mô tả các kiểu dữ liệu chuỗi.

Các kiểu dữ liệu chuỗi có thể sử dụng với bộ điều khiển khả trình được liệt kê trong bảng sau đây.

Kiểu dữ liệu	Kiểu ký tự có thể xử lý	Tiền tố trong ký hiệu Hungary	Mở rộng tiền tố
Chuỗi	Chuỗi ký tự chữ và số và số (ASCII) hoặc tiếng Nhật (Shift-JIS)	s	**s**tring (chuỗi)
Chuỗi [Unicode]	Chuỗi chứa nhiều ngôn ngữ và ký hiệu khác nhau	ws	**w**ide **s**tring (chuỗi rộng)

Kiểu chuỗi sử dụng tùy thuộc vào thiết bị được kết nối với bộ điều khiển khả trình hoặc ngôn ngữ tương ứng.
Chương này mô tả các kiểu chuỗi văn bản khác nhau.

Khi gán một chuỗi dữ liệu kiểu chuỗi cho một biến số kiểu chuỗi, cần đặt chuỗi trong dấu nháy đơn (').

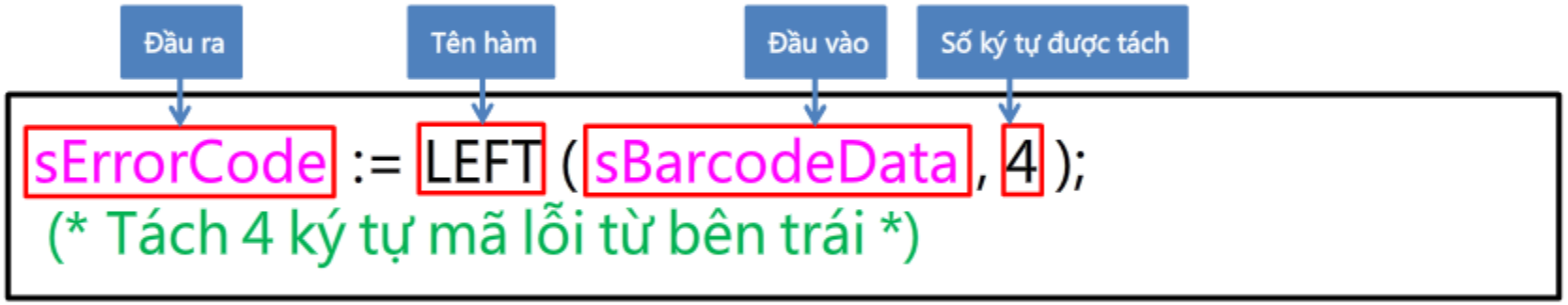
```
sDefault := 'e112,12091458';  (* Gán chuỗi *)
```

7.3 Tách chuỗi (LEFT)

Mã lỗi "e112" được tách từ biến số kiểu chuỗi "sBarcodeData" chứa chuỗi "e112,12091458".

Tên biến số	Chuỗi được lưu trữ
sBarcodeData	e112, 12091458

Hàm LEFT chỉ tách số ký tự được chỉ định bắt đầu từ bên trái của chuỗi đầu vào.
Sau đây là minh họa một chương trình ví dụ.



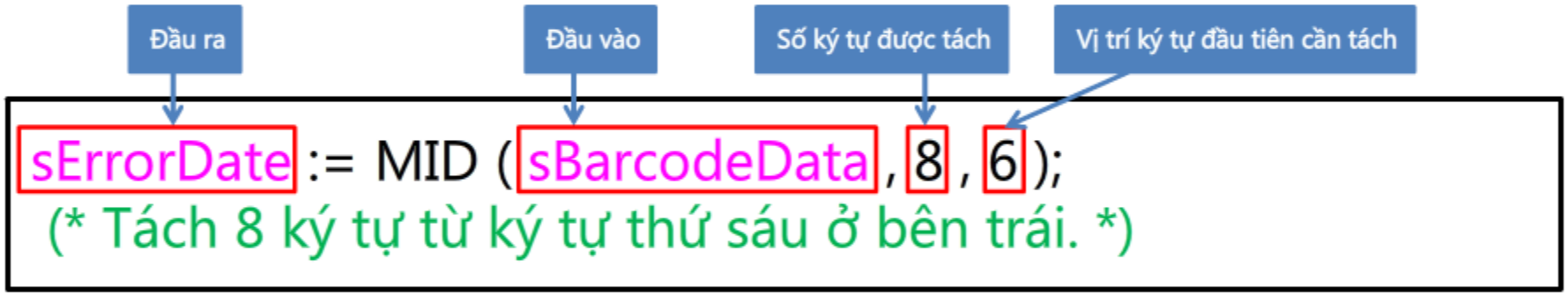
Bốn ký tự được tách riêng từ bên trái. Giá trị "e112", chính là chuỗi biểu diễn mã lỗi, được gán cho bên trái.

7.4 Tách chuỗi (MID)

Thời gian xảy ra lỗi "12091458" được tách từ biến số kiểu chuỗi "sBarcodeData" chứa chuỗi "e112,12091458".

Tên biến số	Chuỗi được lưu trữ
sBarcodeData	e112,12091458

Hàm MID tách số ký tự được chỉ định từ vị trí bắt đầu đã chỉ định của chuỗi đầu vào.
Sau đây là minh họa một chương trình ví dụ.



Trong ví dụ này, một chuỗi 8 ký tự được tách riêng, bắt đầu từ ký tự thứ sáu. Giá trị "12091458", chính là chuỗi biểu diễn thời gian xảy ra lỗi, được gán cho bên trái.

7.5 Tổng kết

Nội dung của chương này như sau:

- Các phương pháp gán chuỗi cho biến số kiểu chuỗi
- Các hàm tách chuỗi (LEFT và MID)

Những điểm quan trọng cần cân nhắc:

Gán chuỗi	• Để gán chuỗi cho một biến số kiểu chuỗi, cần đặt chuỗi trong dấu nháy đơn ('). • Sử dụng kiểu chuỗi hoặc kiểu chuỗi [Unicode] tùy theo thiết bị được kết nối với bộ điều khiển khả trình hoặc ngôn ngữ tương ứng.
Các hàm xử lý chuỗi	• Các hàm được dùng để xử lý chuỗi.

7.6 Tổng kết khóa học

Khóa học đã trình bày những kiến thức cơ bản về cách tạo chương trình bằng ST.
Khóa học e-learning này đến đây là kết thúc.

Các chương trình viết bằng ST được tạo bằng phần mềm kỹ thuật MELSOFT.
Để biết chi tiết về các bước cụ thể như nhập dữ liệu, chỉnh sửa, lưu và biên soạn chương trình bằng phần mềm kỹ thuật MELSOFT, hãy tham khảo tài liệu sau.

- Khóa học Mitsubishi FA e-Learning "MELSOFT GX Works3 (Structured Text)" (MELSOFT GX Works3 (Văn bản có cấu trúc)) (sắp phát hành)
- Hướng dẫn vận hành phần mềm kỹ thuật MELSOFT

Để biết thêm thông tin về ST, hãy tham khảo tài liệu sau.

- Sách hướng dẫn lập trình bộ điều khiển khả trình

Để biết thông tin về các lệnh và hàm cho ứng dụng, hãy tham khảo tài liệu sau.

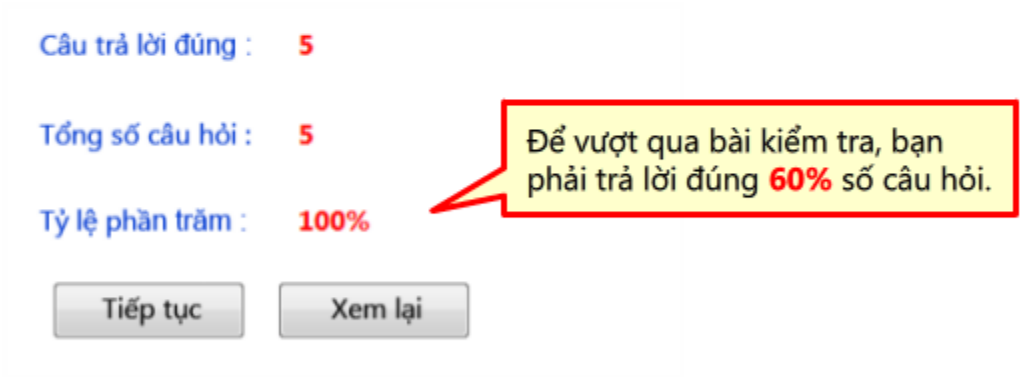
- Hướng dẫn lập trình bộ điều khiển khả trình

Bài kiểm tra Bài kiểm tra cuối khóa

Vì bạn đã hoàn thành tất cả các bài học trong khóa học Kiến thức cơ bản về lập trình (Văn bản có cấu trúc), bạn đã sẵn sàng tham gia bài kiểm tra cuối khóa. Nếu bạn không rõ về bất cứ chủ đề nào được trình bày, vui lòng nhân cơ hội này xem xét lại các chủ đề đó.
Có tổng cộng 12 câu hỏi (20 mục) trong Bài kiểm tra cuối khóa này.
Bạn có thể làm bài kiểm tra cuối khóa nhiều lần tùy thích.

Làm thế nào ghi điểm bài kiểm tra
Sau khi chọn câu trả lời, đừng quên nhấp vào nút **Câu trả lời**. Câu trả lời của bạn sẽ bị mất nếu bạn tiếp tục mà không nhấp vào nút Câu trả lời. (Coi như là câu hỏi chưa được câu trả lời.)

Kết quả điểm số
Số lượng câu trả lời đúng, số lượng câu hỏi, tỷ lệ câu trả lời đúng, và kết quả đạt/hỏng sẽ xuất hiện trên trang điểm số.



- Nhấp vào nút **Tiếp tục** để thoát khỏi bài kiểm tra.
- Nhấp vào nút **Xem lại** để xem lại bài kiểm tra. (Kiểm tra câu trả lời đúng)
- Nhấp vào nút **Thử lại** để làm lại bài kiểm tra.

Bài kiểm tra

Bài kiểm tra cuối khóa 1

Đặc điểm của văn bản có cấu trúc (ST)
Chọn câu mô tả không đúng về ST.

- ST dễ học đối với người có kinh nghiệm viết chương trình bằng ngôn ngữ C hoặc BASIC.
- Các phép tính như phép cộng và phép trừ có thể được viết dưới dạng biểu thức toán học thông thường.
- Sử dụng các biểu tượng tiếp điểm và cuộn cảm để tạo ra chương trình giống một mạch điện.
- ST thích hợp để xử lý dữ liệu.

Câu trả lời | Quay lại

Bài kiểm tra

Bài kiểm tra cuối khóa 2

Những nguyên tắc cơ bản của ST
Chọn câu lệnh đúng được viết bằng ST.

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

Câu trả lời Quay lại

Bài kiểm tra

Bài kiểm tra cuối khóa 3

Chú thích mô tả
Chọn chú thích đúng được viết trong ST.

- ' Gán giá trị 1 cho biến số.
- (* Gán giá trị 1 cho biến số. *)
- { Gán giá trị 1 cho biến số. }
- <!-- Gán giá trị 1 cho biến số. -->

Câu trả lời Quay lại

Bài kiểm tra

Bài kiểm tra cuối khóa 4

Trình tự thực hiện chương trình viết bằng ST
*Giá trị ban đầu của "uTotalProduction" là "100". Giá trị của biến số "uTotalProduction" sẽ là "101" sau khi chương trình ví dụ sau đây được xử lý. Chọn hiện trạng đúng của "uTotalProduction" sau khi quá trình diễn ra được vài giây.

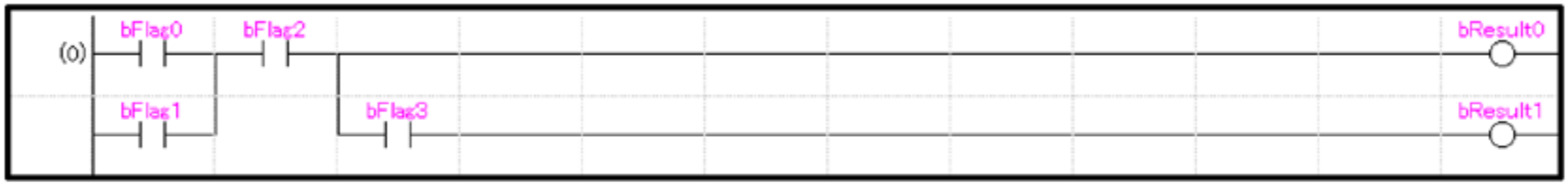
```
uTotalProduction := uTotalProduction + 1;
```

- ○ Giá trị vẫn là 101.
- ○ Giá trị tiếp tục thay đổi.

Câu trả lời | Quay lại

Bài kiểm tra Bài kiểm tra cuối khóa 5

Kết hợp nhiều điều kiện
Chọn ví dụ chương trình viết bằng ST đúng biểu diễn cùng một quy trình vận hành như ví dụ chương trình được viết bằng LD sau đây.



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

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

Câu trả lời Quay lại

Bài kiểm tra Bài kiểm tra cuối khóa 6

Mô tả câu lệnh IF trong ST
Quá trình vận hành sau đây được thực hiện bằng chương trình ví dụ bên dưới.
•Nếu nhiệt độ giảm còn 5 độ trở xuống, bộ phát nhiệt sẽ bật và bộ làm mát sẽ tắt.
•Nếu nhiệt độ vượt quá 50 độ, bộ phát nhiệt sẽ tắt và bộ làm mát sẽ bật.
•Nếu nhiệt độ không áp dụng cho các câu lệnh trên, cả bộ phát nhiệt và bộ làm mát đều sẽ tắt.

*Tên biến số: Nhiệt độ (wTemperature), bộ phát nhiệt (bHeater) và bộ làm mát (bCooler)
Chọn lựa chọn đúng cho mỗi phần còn trống của chương trình ví dụ.

```
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--

Câu trả lời Quay lại

Bài kiểm tra **Bài kiểm tra cuối khóa 7**

Câu lệnh CASE

Chọn câu trả lời đúng cho mỗi câu (Q1 đến Q5) mô tả câu lệnh CASE sau đây.

Câu lệnh CASE được sử dụng để phân nhánh dựa vào giá trị của (Q1).
Trong chương trình ví dụ sau, khi giá trị của (Q2) là 25, biến số (Q3) được gán giá trị (Q4). Khi giá trị của biến số (Q2) không bằng 10, 25 hoặc 8, biến số (Q3) được gán giá trị (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--

Câu trả lời | Quay lại

Bài kiểm tra **Bài kiểm tra cuối khóa 8**

Mảng viết bằng ST và câu lệnh lặp lại
Chương trình ví dụ sau đây tính tổng khối lượng sản xuất theo kế hoạch của tất cả các mẫu được đưa đến Quốc gia Y rồi gán giá trị này cho một biến số. Chọn phần trong mảng được đọc sau khi câu lệnh FOR được thực hiện trong một vòng lặp 3 lần.

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

Mảng sử dụng để lưu trữ số lượng đơn vị dự tính sản xuất cho mỗi mẫu và điểm đến (uProduction)

Điểm đến (hàng)	Mẫu (cột)			
	Mẫu 1	Mẫu 2	Mẫu 3	Mẫu 4
Quốc gia X	[0,0]	[0,1]	[0,2] C	[0,3]
Quốc gia Y	[1.0]	[1.1] A	[1.2] D	[1.3] E
Quốc gia Z	[2,0]	[2,1] B	[2,2]	[2,3]

- A
- B
- C
- D

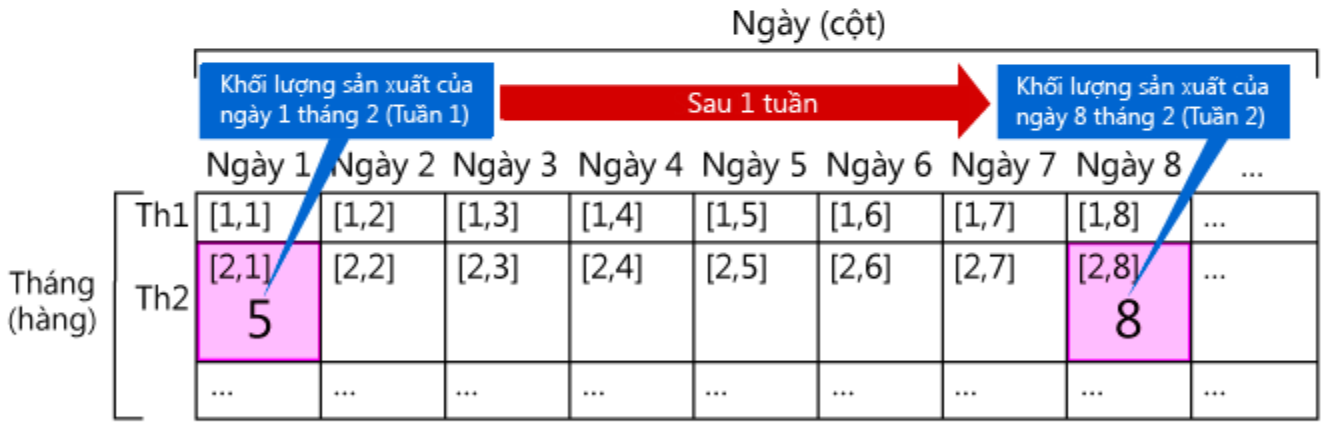
Câu trả lời Quay lại

Bài kiểm tra cuối khóa 9

Mảng viết bằng ST và câu lệnh lặp lại
Chương trình ví dụ sau đây thu được tổng khối lượng sản xuất vào cùng các ngày trong tuần. Tổng sau 4 tuần thu được từ mảng lưu trữ khối lượng sản xuất mỗi ngày. Chọn số đúng cho chương trình ví dụ.

```
uTotalProduction := 0;
FOR wOnceAWeek := 1 TO ■ BY 7 DO
  uTotalProduction := uTotalProduction + uProductionByDate[2,wOnceAWeek];
END_FOR;
(* Tách và tính tổng khối lượng sản xuất trong cùng một ngày của tuần trong vòng 4 tuần kể từ ngày 1 tháng 2. *)
```

Mảng lưu trữ khối lượng sản xuất mỗi ngày (uProductionByDate)



- 22
- 21
- 4
- 28

Câu trả lời | Quay lại

Bài kiểm tra

Bài kiểm tra cuối khóa 10

Đặc điểm của cấu trúc trong ST
Chọn câu mô tả không đúng về cấu trúc.

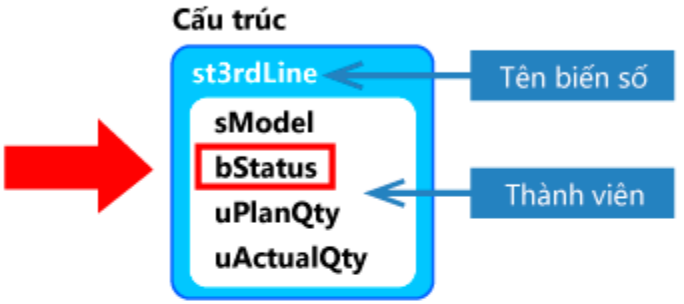
- Cấu trúc được sử dụng để tổ chức và lưu trữ dữ liệu trên thiết bị theo điều kiện như hiện trạng và thông số kỹ thuật.
- Có thể viết các chương trình ngắn gọn để xử lý khối lượng dữ liệu lớn nếu sử dụng cấu trúc.
- Tất cả các thành viên được xác định trong cấu trúc phải có chung kiểu dữ liệu.
- Có thể gán giá trị cho các thành viên trong cùng một cấu trúc mà không cần chỉ định riêng lẻ.

Câu trả lời | Quay lại

Bài kiểm tra Bài kiểm tra cuối khóa 11

Chỉ định thành viên cho cấu trúc trong ST
Cấu trúc sau đây tổ chức các biến số liên quan đến một dây chuyền sản xuất ô tô.
Chọn mô tả đúng để chỉ định thành viên "bStatus" trong cấu trúc này.

Tham số	Tên biến số
Mẫu	sModel
Hiện trạng	bStatus
Sản lượng mục tiêu của ngày hiện tại	uPlanQty
Số sản phẩm hiện tại	uActualQty

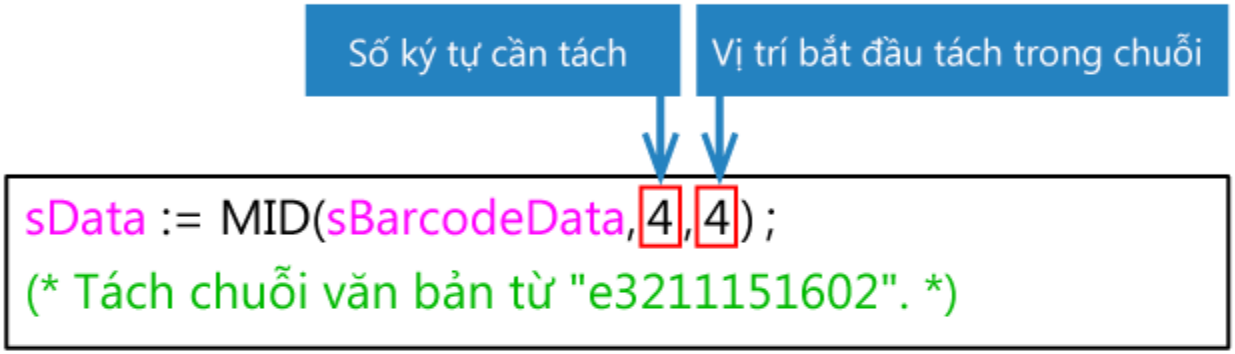


- st3rdLine.bStatus
- st3rdLine->bStatus
- st3rdLine[bStatus]
- st3rdLine[1]

Câu trả lời Quay lại

Bài kiểm tra **Bài kiểm tra cuối khóa 12**

Xử lý chuỗi trong ST
Chương trình ví dụ sau đây tách riêng một chuỗi cụ thể từ chuỗi "e3211151602" được lưu trữ trong biến số "sBarcodeData". Hàm MID tách số ký tự được chỉ định bắt đầu từ vị trí đã chỉ định.
Chọn chuỗi được tách đúng.



- 1151
- 1602
- e321
- 1115

Câu trả lời Quay lại

Bài kiểm tra Điểm kiểm tra

Bạn đã hoàn thành Bài kiểm tra cuối khóa. Kết quả của bạn như sau.
Để kết thúc Bài kiểm tra cuối khóa, hãy tiếp tục tới trang tiếp theo.

Số câu trả lời đúng : 12

Tổng số câu hỏi : 12

Tỷ lệ phần trăm : 100%

Tiếp tục Xem lại

Xin chúc mừng. Bạn đã vượt qua bài kiểm tra.

Bạn đã hoàn thành khóa học Kiến thức cơ bản về lập trình (Văn bản có cấu trúc).

Cảm ơn bạn đã tham gia khóa học này.

Chúng tôi hy vọng bạn thích các bài học và những thông tin bạn có được trong khóa học này sẽ hữu ích trong tương lai.

Bạn có thể xem lại khóa học này nhiều lần tùy ý.

Xem lại Đóng