MITSUBISHI ELECTRIC
Changes for the Better

PLC

Podstawy programowania (Tekst strukturalny)

W ramach niniejszego kursu przedstawiony zostanie sposób tworzenia podstawowych programów wykorzystywanych do sterowania sterownikami programowalnymi MELSEC.
W ramach niniejszego kursu tekst strukturalny (ST) jest stosowany do tworzenia programu.

L(FOD)00544POL

Wprowadzenie

Przeznaczenie kursu

W ramach niniejszego kursu przedstawiony zostanie sposób tworzenia podstawowych programów w tekście strukturalnym (ST) wykorzystywanych do sterowania sterownikami programowalnymi MELSEC.

Ukończenie poniższych kursów lub posiadanie odpowiedniej wiedzy jest warunkiem wstępnym uczestnictwa w niniejszym kursie:

Programming Basic (Podstawy programowania)

Posiadanie wiedzy lub doświadczenia z językiem programowania C lub BASIC może pomóc w zrozumieniu treści niniejszego kursu.

Wprowadzenie

Struktura kursu

Treść tego kursu posiada następującą strukturę.

Rozdział 1 – Przegląd tekstu strukturalnego

W rozdziale tym przedstawiono funkcje i odpowiednie zastosowanie tekstu strukturalnego (ST).

Rozdział 2 – Podstawowe zasady programów w ST

W rozdziale tym przedstawiono podstawowe zasady stosowane podczas tworzenia programów w ST.

Rozdział 3 – Tworzenie programów do sterowania wejściami i wyjściami

W rozdziale tym przedstawiono sposób tworzenia programów do sterowania wejściami i wyjściami.

Rozdział 4 – Operacje arytmetyczne

W rozdziale tym przedstawiono sposób tworzenia programów w wykorzystaniem operacji arytmetycznych.

Rozdział 5 – Rozgałęzienia warunkowe

W rozdziale tym przedstawiono rozgałęzienia warunkowe.

Rozdział 6 – Zapisywanie i przetwarzanie danych

W rozdziale tym przedstawiono sposób zapisywania zwięzłych programów w pamięci i przetwarzania danych.

Rozdział 7 – Przetwarzanie danych typu ciąg znaków

W rozdziale tym przedstawiono metody przetwarzania danych typu ciąg znaków.

Test końcowy

Ocena zaliczająca: 60% lub więcej

Wprowadzenie

Jak korzystać z tego narzędzia do e-learningu

Przejdź do następnej strony		Przejdź do następnej strony.
Wróć do poprzedniej strony		Wróć do poprzedniej strony.
Przejdź do żądanej strony		Wyświetli się „Spis treści” umożliwiający przejście do żądanej strony.
Zakończ naukę		Zakończ naukę.

Wprowadzenie

Wskazówki dotyczące użytkowania

Zalecenia dotyczące bezpieczeństwa

Jeśli uczysz się, korzystając z rzeczywistych produktów, prosimy o dokładne przeczytanie zasad bezpieczeństwa zawartych w odpowiednich instrukcjach obsługi.

Środki ostrożności dla tego kursu

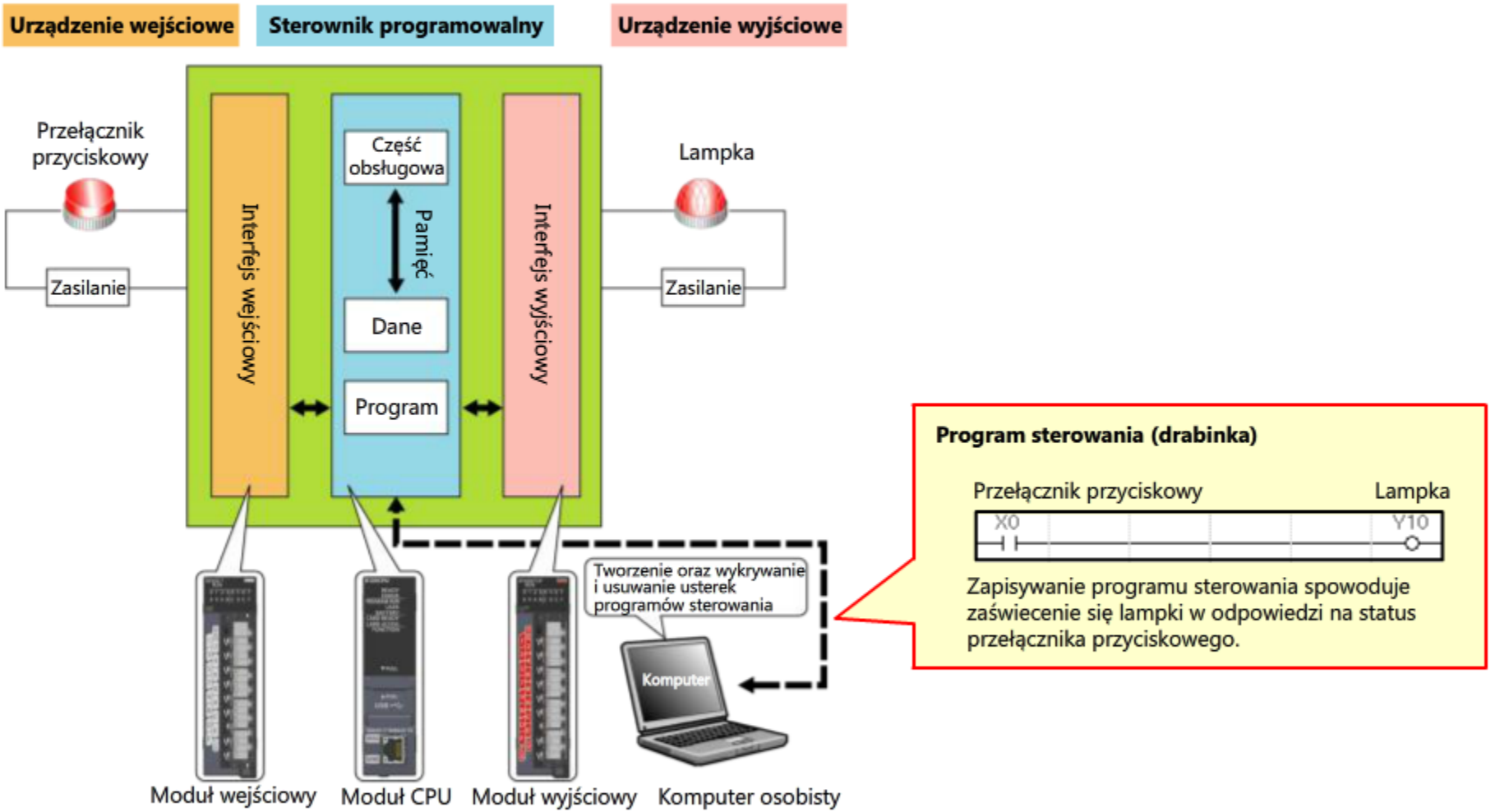
Ekrany wyświetlane dla oprogramowania inżynierskiego MELSOFT, którego używasz, mogą się różnić od przedstawionych w tym kursie.
W kursie tym używane są symbole drabinki MELSOFT GX Works3 w celu tworzenia programów.

Rozdział 1 Przegląd tekstu strukturalnego

W rozdziale tym przedstawiono funkcje i odpowiednie zastosowanie tekstu strukturalnego (ST).

1.1 Programy sterujące

Poniższy rysunek przedstawia konfigurację systemu sterownika programowalnego.
Sterowniki programowalne działają zgodnie z ich programami.
Działanie sterowników programowalnych można skonfigurować zgodnie z wymaganiami poprzez utworzenie programów.



Języki programowania sterowników programowalnych są definiowane przez międzynarodowy standard opracowany przez International Electrotechnical Commission (IEC) (Międzynarodowa Komisja Elektrotechniczna).

1.2 Funkcje ST i porównanie z innymi językami programowania IEC

IEC 61131 jest międzynarodowym standardem systemów sterowników programowalnych.
Języki programowania sterowników programowalnych są znormalizowane przez IEC 61131-3. ST jest jednym ze znormalizowanych języków programowania.
Każdy język oferuje różne funkcje w celu dostosowania danego zastosowania i umiejętności programistów.

Poniższa tabela przedstawia listę języków programowania IEC 61131-3.

Język programowania	Funkcje
Ladder **D**iagram (LD) (schemat drabinkowy)	• Symbole styków i cewek są stosowane w celu utworzenia programu podobnego do obwodu elektrycznego. • Wykonywanie programu jest łatwe i zrozumiałe, nawet dla osób początkujących.
Structured **T**ext (ST) (tekst strukturalny)	• Programy zapisywane są jako tekst (znaki). • ST jest łatwym językiem do nauczenia dla osób mających doświadczenie w pisaniu programów w języku programowania C lub BASIC. • Wzory obliczeniowe są podobne do wyrażeń matematycznych, które są łatwe do zrozumienia. • ST doskonale sprawdza się w przypadku przetwarzania danych.
Function **B**lock **D**iagram (FBD) (schemat bloków funkcyjnych)	• Programy są zapisywane poprzez ustalanie bloków z różnymi funkcjami i wskazywanie powiązań pomiędzy tymi blokami. • FBD poprawia czytelność, ponieważ cała operacja jest niezwykle czytelna.
Sequential **F**unction **C**hart (SFC) (sieć działań)	• Warunki i procesy są zapisywane jako schematy. • Wykonywanie programu jest łatwe do zrozumienia.
Instruction **L**ist (IL) (lista instrukcji)	• IL jest zbliżone do języka maszynowego. • IL jest obecnie rzadko wykorzystywane.

W niniejszym kursie przedstawiono sposób zapisywania podstawowych programów sterowania za pomocą ST.

1.3 Podsumowanie

W rozdziale tym przedstawiono:

- Związek pomiędzy systemami sterowników programowalnych a programami sterowania
- Międzynarodowy standard programów sterowania
- Funkcje ST

Ważne kwestie do rozważenia:

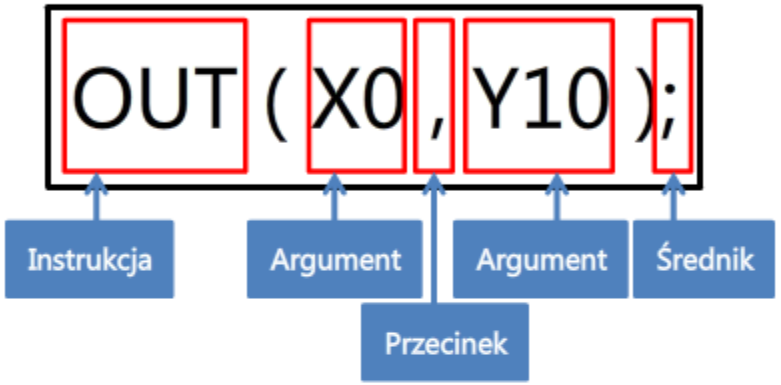
Związek pomiędzy systemami sterowników programowalnych a programami sterowania	• Sterowniki programowalne działają zgodnie z programami sterowania. • Działanie sterowników programowalnych można skonfigurować zgodnie z wymaganiami poprzez utworzenie programów sterowania.
Międzynarodowy standard programów sterowania	• ST jest jednym z języków programowania IEC. • Inne języki programowania IEC, włączając LD, FBD, SFC i IL, gdzie każdy z nich oferuje różne funkcje, pozwalające na dopasowanie do danego zastosowania i umiejętności programistów.
Funkcje ST	• ST jest łatwym językiem do nauczenia dla osób mających doświadczenie w pisaniu programów w języku C lub BASIC. • Obliczenia, takie jak dodawanie i odejmowanie, mogą być zapisywane jako typowo stosowane wyrażenia matematyczne, które są łatwe do zrozumienia. • ST doskonale sprawdza się w przypadku przetwarzania danych.

Rozdział 2 Podstawowe zasady programów zapisanych w ST

W rozdziale tym przedstawiono podstawowe zasady stosowane podczas tworzenia programów w ST.

2.1 Przykładowy program podstawowy (wyrażenie sterowania wejściami i wyjściami)

W punkcie tym przedstawiono przykład podstawowego programu napisanego z użyciem ST.
W poniższym przykładowym programie wyjście Y10 włącza się, gdy wejście X0 włącza się, a wyjście Y10 wyłącza się, gdy wejście X0 wyłącza się.



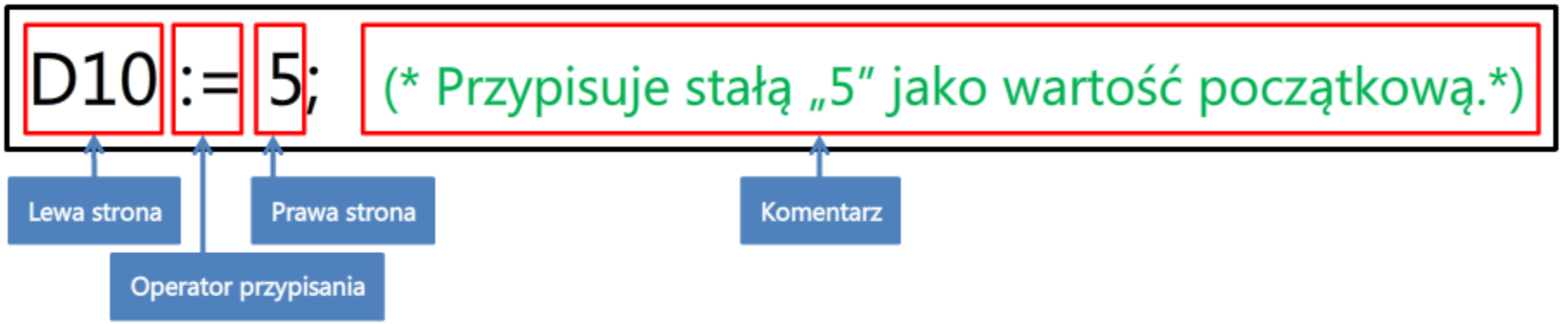
Instrukcja określa operację do wykonania.
Argumenty są zapisywane w nawiasach po instrukcji.
Argumenty są używane w celu opisania zmiennych, wyrażeń arytmetycznych i wartości stałych.
W przypadku sterowników programowalnych MELSEC adresy pamięci urządzeń CPU mogą być stosowane jako zmienne.

Liczba argumentów zależy od instrukcji.
Kilka argumentów oddziela się przecinkiem (,).

Jedna linia przedstawiona powyżej stanowi jedno wyrażenie. Każde wyrażenie jest zakończone średnikiem (;).
Program jest zapisywany jako połączenie wielu wyrażeń.

2.2 Przykładowy program podstawowy (wyrażenie przypisania)

Następny przykład przedstawia program, w którym zastosowano wyrażenie przypisania.
Poniższe wyrażenie przypisuje stałą dziesiętną „5" do zmiennej „D10".



Dla tego wyrażenie przypisania użyto operatora przypisania (:=). Należy pamiętać, że dwukropek (:) znajduje się po lewej stronie znaku równości (=).
Operator przypisania przypisuje wartość po prawej stronie do lewej strony.

Dodanie komentarza do programu sprawi, że działanie programu stanie się bardziej zrozumiałe. Umieść komentarze pomiędzy dwoma gwiazdkami (* *).

2.3 Zapis numeryczny

W przykładowym programie przedstawionym na poprzedniej stronie wartość dziesiętna została przypisana do zmiennej.

Czasami wartości inne niż dziesiętne, takie jak binarne i szesnastkowe są stosowane do sterowania sekwencyjnego.
Poniższa tabela przedstawia listę typów zapisów numerycznych stosowanych w ST sterowników programowalnych MELSEC.

Typ zapisu numerycznego	Metoda zapisu	Przykład
Binarny	Dodaj prefiks „2#”.	**2#**11010
Ósemkowy	Dodaj prefiks „8#”.	**8#**32
Dziesiętny	Bezpośrednie wejście	26
	Dodaj prefiks „K”.	**K**26
Szesnastkowy	Dodaj prefiks „16#”.	**16#**1A
	Dodaj prefiks „H”	**H**1A

Poniżej przedstawiono przykładowe programy do przypisania wartości do zmiennych.

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

2.3.1 Zapis bitowy

Bity reprezentują warunki true/false (prawda/fałsz), takie jak statusy on/off (wł./wył.) sygnałów. Bity reprezentują również ustanowienie/brak ustanowienia warunków.
W ST bity nie mogą być zapisywane jako „ON" i „OFF". Są one wyrażane jako „1" (ON) i „0" (OFF). Bity mogą być również wyrażane jako „TRUE" i „FALSE".

Poniższa tabela zawiera listę różnych typów zapisów.

Stan	ON	OFF
	True	False
Zapis numeryczny	1	0
Zapis True/False	TRUE	FALSE

Poniżej przedstawiono przykłady przypisania wartości do zmiennych typu bit.

Zapis numeryczny

```
X0 := 1;
```

=

Zapis True/False

```
X0 := TRUE;
```

Zapis numeryczny

```
X0 := 0;
```

=

Zapis True/False

```
X0 := FALSE;
```

2.4 Sekwencja wykonywania programu

Wyrażenie w ST są wykonywane w kolejności od góry do dołu.

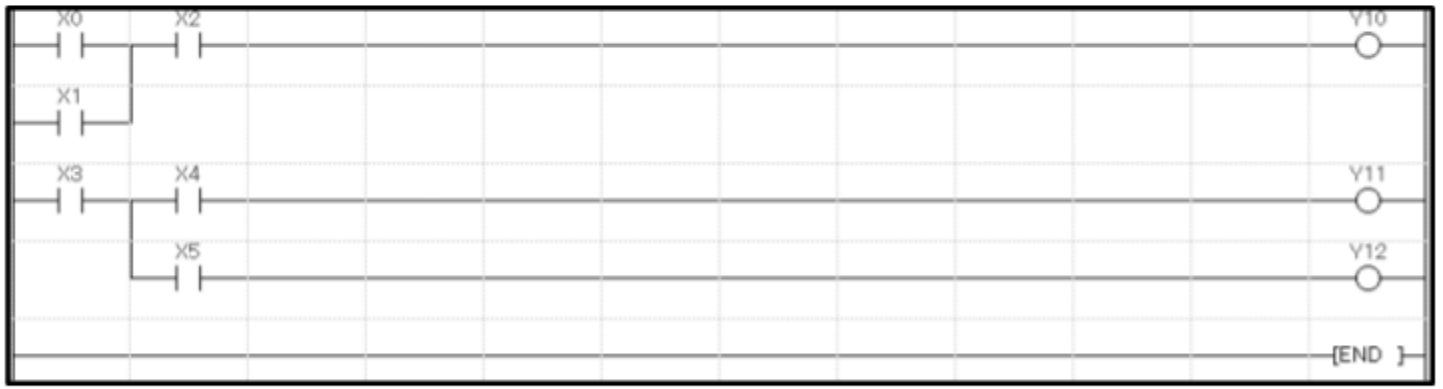
Przykład programu ST

```
Y10 := (X0 OR X1) AND X2; (* Wykonywane w pierwszej kolejności *)
Y11 := X3 AND X4;         (* Wykonywane w drugiej kolejności *)
Y12 := X3 AND X5;         (* Wykonywane w trzeciej kolejności. Nie wymaga wyrażenie END na końcu.*)
```

Wykonywanie powtarzalne

*Pomimo że wyrażenie END jest wymagane na końcu programu zapisanego w języku LD, w przypadku ST nie jest to konieczne.

Poniższy program drabinkowy przedstawia tę samą operację, co co przykładowy program ST powyżej.



Wykonywanie powtarzalne

Podobnie jak program zapisany w języku LD, instrukcje w ST są wielokrotnie wykonywane – po osiągnięciu ostatniej instrukcji, ponownie program powraca do pierwszej instrukcji.

2.5 Podsumowanie

W rozdziale tym przedstawiono:

- Podstawowy program w ST
- Format wyrażenia przypisania
- Zapis numeryczny
- Sekwencja wykonywania programu
- Komentarz

Ważne kwestie do rozważenia:

Podstawowy program ST	• Wyrażenie jest najmniejszym elementem programów ST. • Każde Wyrażenie jest zakończone średnikiem (;). • Program jest zapisywany jako połączenie wielu wyrażeń.
Format wyrażenia przypisania	• W przypadku wyrażeń przypisania stosowany jest operator przypisania (:=).
Zapis numeryczny	• Typy zapisu numerycznego w ST • W ST zamiast zapisu „ON" i „OFF" stosowane są wartości „1" i „0". • W ST wartości bitu można również zapisać jako „TRUE" i „FALSE".
Sekwencja wykonywania programu	• Programy tworzone w ST są wykonywane w kolejności od góry do dołu. • Jak w przypadku programów LD, proces wykonywania programów ST jest wielokrotnie powtarzany – po ukończeniu procesu program ponownie powraca do początku.
Komentarz	• Dodanie komentarza do programu sprawi, że działanie programu stanie się bardziej zrozumiałe. • Komentarze są zawarte pomiędzy dwoma gwiazdkami (* *).

Rozdział 3 Tworzenie programów sterowania wejściami i wyjściami

W rozdziale tym przedstawiono sposób tworzenia programów sterowania wyrażenie w ST.

3.1 Programy sterowania wejściami i wyjściami

Poniżej przedstawiono przykładowy program sterowania wejściami i wyjściami sterownika programowalnego.

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

- Instrukcja wyjścia
- Warunek wykonania (argument)
- Urządzenie wyjściowe (argument)

„OUT" oznacza instrukcję wyjścia. Argument określa warunek wykonania oraz urządzenie, do której skierowane jest wyjście. Jeśli warunek wykonania X0 został spełniony, urządzenie Y10 zostanie włączone.

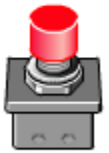
Kliknij przełącznik wejścia przedstawiony poniżej. Przełącznik wejścia X0 zostanie włączony.

- Jeśli przełącznik wejścia X0 zostanie włączony, lampka wyjścia Y10 zostanie włączona.
- Jeśli przełącznik wejścia X0 zostanie wyłączony, lampka wyjścia Y10 zostanie wyłączona.

Przykładowy program sterowania wejściami i wyjściami zapisany w ST

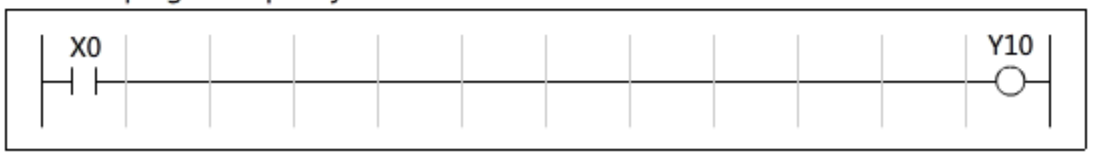
```
OUT(X0,Y10);
```

Przełącznik wejścia X0



Lampka wyjścia Y10

Taki sam program zapisany w LD



3.1 Programy sterowania wejściami i wyjściami

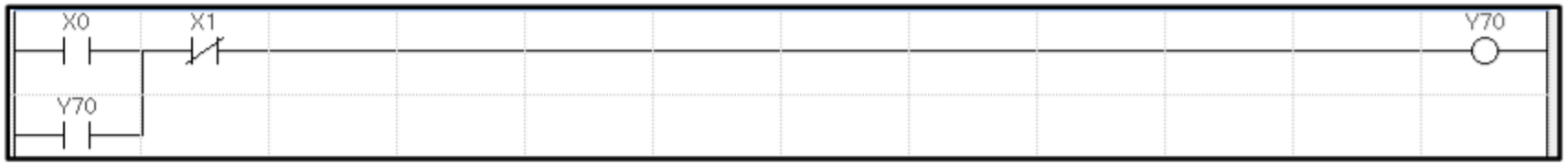
Podobnie do LD, dostępnych jest wiele instrukcji poza instrukcją OUT, takie jak instrukcje sterowania wejściami i wyjściami oraz instrukcje przetwarzania danych.
W celu uzyskania szczegółowych informacji na temat dostępnych instrukcji w ST patrz podręcznik programowania.

Uwaga, zapisanie „OUT(X0, Y10);" jako „Y10 := X0;" spowoduje wykonanie tej samej operacji.

```
Y10 := X0;  (* Taka sama operacja, jako „OUT(X0, Y10);" *)
```

3.2 Łączenie wielu warunków

Poniższy program drabinkowy przedstawia obwód samoutrzymujący się.



Ten sam program może zostać zapisany w ST w następujący sposób.

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

Operator logiczny

Zgodnie z powyższym, operatory logiczne są stosowane w ST w celu łączenia wielu warunków.

Poniższa tabela zawiera listę operatorów logicznych.

Operator	Znaczenie
OR	Logiczne LUB
AND	Logiczne I
NOT	Logiczne zaprzeczenie
XOR	Nierównoważność

3.3 Definiowanie znaczenia zmiennych

Korzystanie z ST w przypadku sterowników programowalnych MELSEC zarówno urządzenia, jak i etykiety mogą być przypisywane jako aliasy do zmiennych.
Użytkownicy mogą korzystać z etykiet zgodnie z zastosowaniami.

Jeśli etykieta powiązana z zastosowaniem zostanie przypisana, dana operacja jest łatwiejsza do zrozumienia.

```
Y10 := (X0 OR X1) AND X2;     (* Zapisane przy użyciu nazw urządzeń *)
```

```
Lamp := (Switch0 OR Switch1) AND Switch2;     (* Zapisane przy użyciu etykiet *)
```

Etykiety można nazywać za pomocą oprogramowania inżynierskiego MELSOFT.

Kolejne przykładowe programy przedstawione w niniejszym kursie zostały opisane za pomocą etykiet.

3.4 Podsumowanie

W rozdziale tym przedstawiono:

Przykładowe programy sterowania wejściami i wyjściami

- Operatory logiczne są stosowane w ST w celu łączenia wielu warunków.
- Nazwy urządzeń i etykiet można stosować jako nazwy zmiennych.

Ważne kwestie do rozważenia:

Łączenie wielu warunków	• Operatory logiczne są stosowane w ST w celu łączenia warunków.
Definiowanie znaczenia zmiennych	• Jeśli etykieta powiązana z zastosowaniem zostanie przypisana, dana operacja jest łatwiejsza do zrozumienia.

Rozdział 4 Operacje arytmetyczne

W rozdziale tym przedstawiono sposób tworzenia arytmetycznych programów sterowania.

- Wyrażenie operacji arytmetycznych
- Określanie typu danych odpowiadających zakresom numerycznym
- Nadawanie nazw zmiennym w celu uniknięcia niezgodności w typach danych

4.1 Podstawowe operacje arytmetyczne

Niniejszy przykładowy program podsumowuje wielkość produkcyjną dwóch oddzielnych linii produkcyjnych.
Prawa strona równania jest operacją arytmetyczną zawierającą zmienne i operatory arytmetyczne.

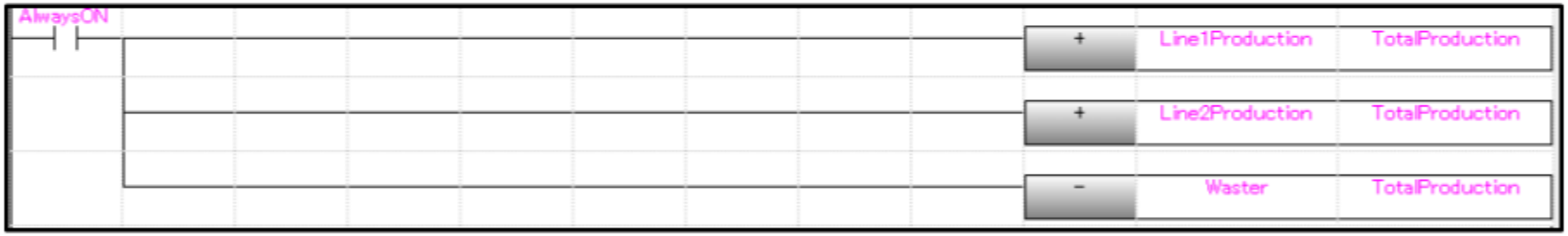
Przykładowy program arytmetyczny zapisany w ST

Operator dodawania

Operator odejmowania

```
TotalProduction := Line1Production + Line2Production - Waster;
(* Wartość całkowita wielkości produkcyjnej dwóch linii produkcyjnych, odejmij liczbę produktów uszkodzonych od wartości całkowitej i przypisz uzyskaną wartość. *)
```

Poniżej przedstawiono ten sam program zapisany w LD.



Zgodnie z powyższym program musi zostać zapisany przy użyciu 3 linii w drabince. Jednak dzięki ST może zostać zapisany w 1 linii.

Poniższa tabela zawiera listę podstawowych operatorów arytmetycznych.

Operator	Znaczenie
+	Dodawanie
-	Odejmowanie
*	Mnożenie
/	Dzielenie

4.2 Typy danych zmiennych

Typ danych musi zostać określony dla każdej zmiennej poprzez zdefiniowanie zakresu wartości, który zostanie przetworzony.
Typami danych wartości numerycznej w ST są bit, liczba całkowita i liczby rzeczywiste.

Wśród typów danych stosowanych w ST poniższa tabela przedstawia typy danych użyte w niniejszym kursie.

Typ danych		Zakres danych
Bit		Stan WŁ./WYŁ. urządzeń bitowych oraz stan prawda/fałsz wyników wykonania
Integer (Liczba całkowita)	Word (unsigned) (Słowo danych (nieprzypisane))	0–65 535
	Word (signed) (Słowo danych (przypisane))	−32 768–32 767
	Double-word (unsigned) (Słowo podwójne (nieprzypisane))	0–4 294 967 295
	Double-word (signed) (Słowo podwójne przypisane)	−2 147 483 648–2 147 483 647

W przypadku korzystania z liczby całkowitej jako typu danych wybierz słowo lub słowo podwójne zgodnie z zakresem danych i wybierz typ ze znakiem lub bez znaku zgodnie z wymogiem przetworzenia wartości ujemnych.
Określ typ danych zmiennej, gdy nazwa etykiety zostanie ustawiona przy pomocy oprogramowania inżynierskiego MELSOFT.

4.3 Nazwy zmiennych przedstawiających typy danych

Używając różnych typów danych po lewej i prawej stronie równania przypisania może spowodować błąd kompilacji lub nieoczekiwany wynik.
Poniżej przedstawiono przykład takiego przypadku.

```
ValueA := ValueB;      (* ValueA: Liczba całkowita Word  ValueB: Liczba całkowita Double-word *)
```

Liczba całkowita Double-word nie może zostać przypisana do liczby całkowitej typu Word. Jednakże w tym przypadku typ danych nie jest rozpoznawalny.

Istnieje możliwość dodania prefiksów przedstawiających typ danych do nazw zmiennych, aby zapewnić wzrokową identyfikację typów danych.
Tego typu nazywanie zmiennej znane jest jako notacja węgierska.

Typ danych		Zakres danych	Prefiks	Rozszerzenie prefiksu
Bit		Stan WŁ./WYŁ. urządzeń bitowych oraz stan prawda/fałsz wyników wykonania	b	**B**it (Bit)
Integer (Liczba całkowita)	Word (unsigned) (Słowo danych (bez znaku))	0–65 535	u	**u**nsigned word (słowo bez znaku)
	Word (signed) (Słowo danych (ze znakiem))	−32 768–32 767	w	signed **w**ord (słowo danych ze znakiem)
	Double-word (unsigned) (Słowo podwójne (bez znaku))	0–4 294 967 295	ud	**u**nsigned **d**ouble-word (słowo podwójne ze znakiem)
	Double-word (signed) (Słowo podwójne ze znakiem)	−2 147 483 648–2 147 483 647	d	signed **d**ouble-word (słowo podwójne ze znakiem)

Przykładowy program na górze strony może zostać zapisany jako następująca notacja węgierska:

```
wValueA := dValueB;  (* Zmienna typu Double-word nie może zostać przypisana do zmiennej typu Word. *)
```

Dzięki notacji węgierskiej niezgodności typu danych można zidentyfikować podczas procesu zapisywania programu.

W dalszej części kursu nazwy zmiennych podane w przykładzie zostały zapisane w notacji węgierskiej.

4.4 Podsumowanie

W rozdziale tym przedstawiono:

- Wyrażenie operacji arytmetycznych
- Określanie typu danych odpowiadających zakresom numerycznym
- Dodawanie nazw zmiennych przedstawiających typy danych

Ważne kwestie do rozważenia:

Podstawowe operacje arytmetyczne	• Operatory stosowane zwykle w podstawowych językach programowania mogą zostać użyte w ST do wyrażania obliczeń.
Typy danych zmiennych	• Typ danych musi zostać określony dla każdej zmiennej poprzez zdefiniowanie zakresu wartości, który zostanie przetworzony.
Dodawanie nazw zmiennych przedstawiających typy danych	• Opisywanie nazw zmiennych za pomocą notacji węgierskiej umożliwia identyfikację niezgodności w typach danych zmiennej podczas zapisywania programów.

Rozdział 5 Operacje warunkowe

Programy sterowania zawierają również sekcje kodu, w których aktualne przetwarzanie zmienia się zgodnie z określonymi warunkami.
W rozdziale tym przedstawiono rozgałęzienia warunkowe.

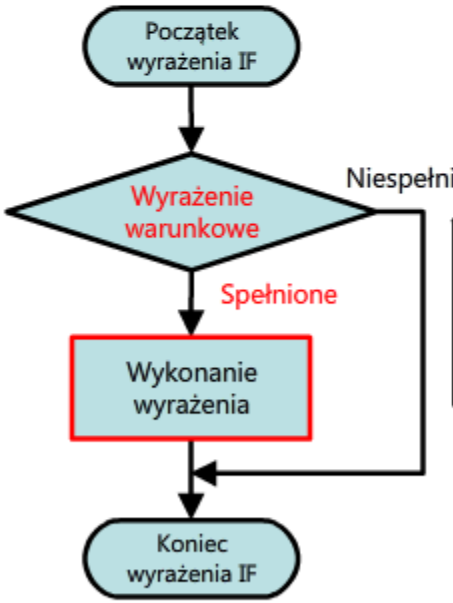
5.1 Operacje warunkowe (IF)

Wyrażenia IF są stosowane do operacji warunkowych. Składnia wyrażenia IF została opisana poniżej.

```
IF wyrażenie warunkowe THEN
  Wyrażenie wykonywane;        (* Wyrażenie jest wykonywane, jeśli wyrażenie warunkowe zostało spełnione. *)
END_IF;                        (* END_IF; musi być umieszczane na końcu wyrażenia IF. *)
```

W ramach tego programu wyrażenie jest wykonywane, gdy wyrażenie warunkowe zostanie spełnione.
Wyrażenie nie jest wykonywane, gdy wyrażenie warunkowe nie zostanie spełnione.

Poniższy rysunek przedstawia przebieg operacji w ramach niniejszego przykładowego programu.

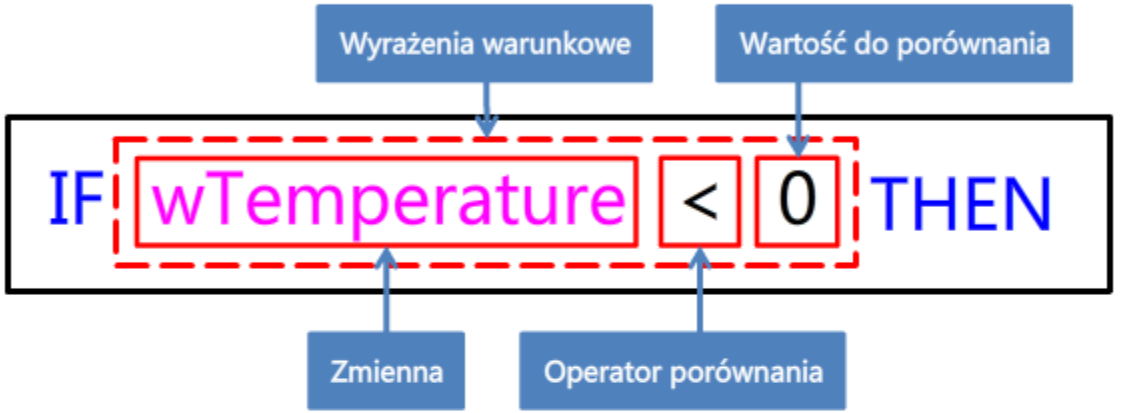


Poniższy przykład przedstawia rozgałęzienie programu poprzez porównywanie wartości zmiennych.
W przykładowym programie nagrzewnica włącza się, gdy temperatura w panelu sterowania spadnie poniżej 0 stopni.

```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Nagrzewnica włącza się, gdy temperatura w panelu sterowania spadnie poniżej 0 stopni. *)
END_IF;
```

5.1.1 Zapisywanie wyrażeń warunkowych

Na poprzedniej stronie opisano wyrażenie warunkowe „wTemperature < 0", które oznacza „gdy wartość zmiennej wTemperature jest mniejsza niż 0".
Tak jak w tym wyrażeniu, wyrażenia warunkowe stosują operatory porównania w celu przedstawienia zależności pomiędzy zmiennymi i wartościami do porównania.



Po lewej i prawej stronie operatora porównania wartości zostały zapisane jako zmienne lub stałe porównania.

Poza porównywaniem zmiennych i stałych wyrażenia warunkowe mogą być zapisywane w celu porównania zmiennych i przeprowadzenia operacji logicznych wyników porównania lub zmiennych typu bit.

Porównywanie zmiennych

- uValue1 <= uValue2

Operacja logiczna dwóch wyników porównania

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

Operacja logiczna dwóch zmiennych typu bit

- bSwitch0 OR bSwitch1

Poniższa tabela zawiera listę typów operatorów porównania.

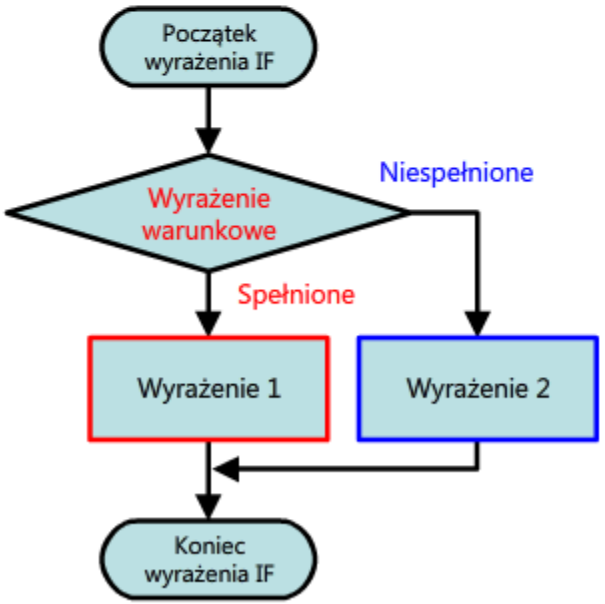
Operator	Znaczenie
>	Większe niż
<	Mniejsze niż
>=	Większe lub równe
<=	Mniejsze lub równe
=	Równe
<>	Nierówne

5.1.2 Alternatywne operacje warunkowe wyrażenia IF (ELSE)

Proste wyrażenia IF (patrz punkt 5.1) są stosowane w celu wykonywania wyrażenia, gdy wyrażenie warunkowe zostanie spełnione. Aby wykonać inne wyrażenie, gdy wyrażenie warunkowe nie zostało spełnione, należy użyć wyrażenia ELSE.

```
IF conditional expression THEN
  Wyrażenie 1;          (* Wyrażenie 1 zostanie wykonywane, jeśli wyrażenie warunkowe zostanie spełnione. *)
  ELSE
  Wyrażenie 2;          (* Wyrażenie 2 zostanie wykonane, jeśli wyrażenie warunkowe nie zostanie spełnione *)
END_IF;
```

Poniższy rysunek przedstawia przebieg operacji w przypadku zastosowania wyrażenia ELSE.



Poniższy przykładowy program wykonuje różne wyrażenia w zależności od tego, czy warunek został spełniony.

Przykładowy program przedstawiony w punkcie 5.1 ma wadę polegającą na tym, że nagrzewnica podnosi temperaturę, nawet po osiągnięciu 0 stopni. Jednakże następujący program wyłącza nagrzewnicę, gdy „wTemperature" przekroczy 0 stopni.

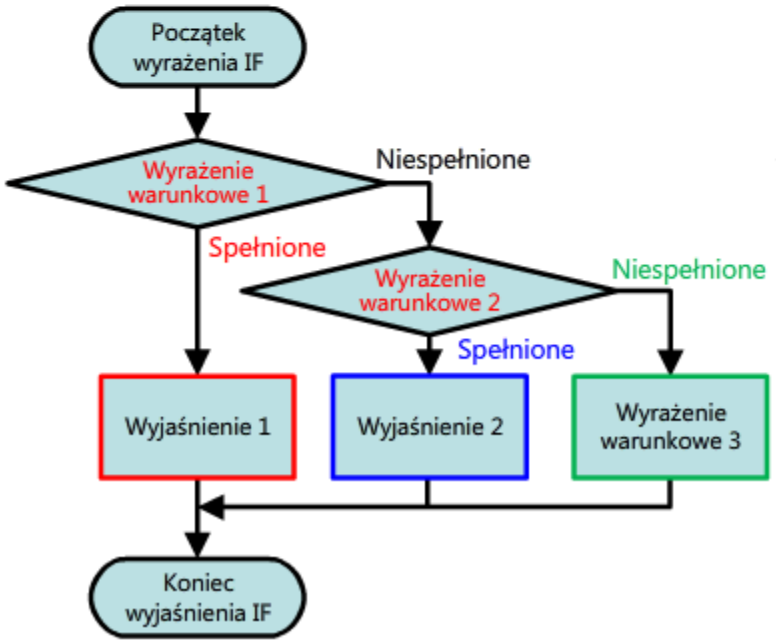
```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Włącza nagrzewnicę, gdy temperatura spadnie poniżej 0 stopni. *)
  ELSE
  bHeater := 0;  (* Wyłącza nagrzewnicę, gdy temperatura osiągnie lub przekroczy 0 stopni *)
END_IF;
```

5.1.3 Warunki dodatkowe wyrażenia IF (ELSIF)

Wyrażenia ELSE są stosowane w celu wykonywania innego wyrażenia, gdy wyrażenie warunkowe nie zostanie spełnione. Istnieje możliwość dodania następnej operacji warunkowej poprzez użycie wyrażeń ELSIF, co oznacza, że jeśli poprzednie wyrażenie warunkowe nie zostanie spełnione, wówczas inne wyrażenie warunkowe zostanie sprawdzone.

```
IF Wyrażenie warunkowe 1 THEN
  Wyrażenie 1;      (* Wyrażenie 1 zostanie wykonane, jeśli wyrażenie warunkowe 1 zostanie spełnione. *)
  ELSIF Wyrażenie Warunkowe 2 THEN
  Wyrażenie 2;      (* Wyrażenie 2 zostanie wykonane, jeśli wyrażenie warunkowe 1 nie zostanie spełnione, a wyrażenie warunkowe 2 zostanie spełnione. *)
  ELSE
  Wyrażenie 3;      (* Wyrażenie 3 zostanie wykonane, jeśli wyrażenia warunkowe 1 i 2 nie zostaną spełnione. *)
END_IF;
```

Poniższy rysunek przedstawia przebieg operacji w przypadku zastosowania wyrażenia ELSEIF.



Wyrażenie ELSIF jest dodawane do przykładowego programu przedstawionego w punkcie 5.1.2, aby poradzić sobie z przypadkiem przekroczenia temperatury powyżej 40 stopni.

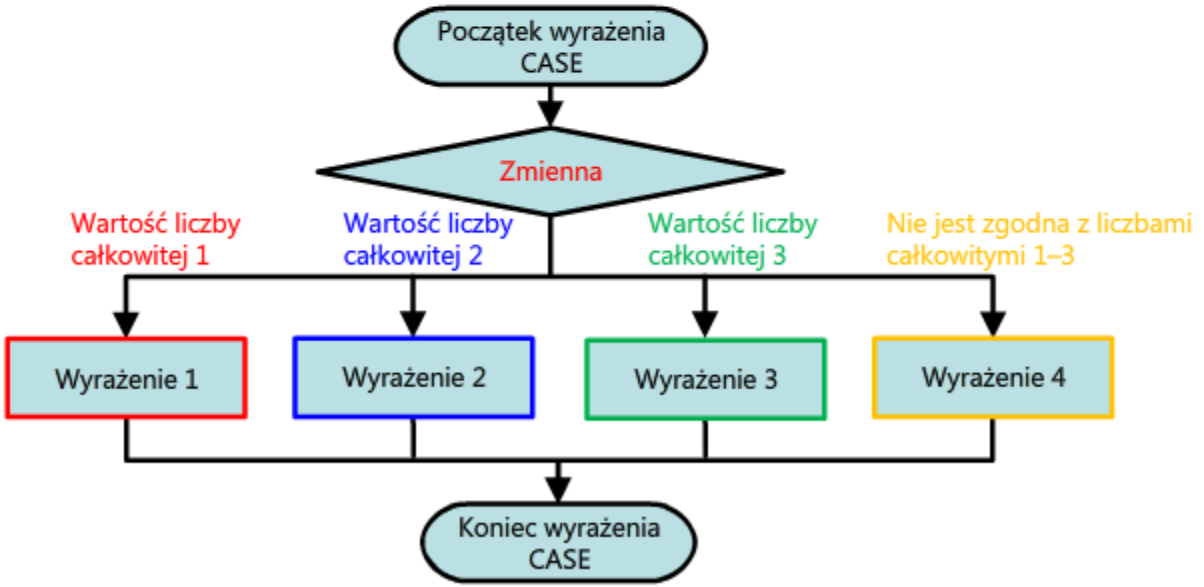
```
IF wTemperature < 0 THEN
   bHeater := 1; (* Włącza nagrzewnicę, gdy temperatura spadnie poniżej 0 stopni. *)
   bCooler := 0; (*  Wyłącza chłodnicę, gdy temperatura spadnie poniżej 0 stopni. *)
   ELSIF 40 < wTemperature THEN
   bHeater := 0; (* Wyłącza nagrzewnicę, gdy temperatura przekroczy 40 stopni. *)
   bCooler := 1; (* Włącza chłodnicę, gdy temperatura przekroczy 40 stopni. *)
   ELSE
   bHeater := 0; (*  Wyłącza nagrzewnicę, gdy żaden z powyższych warunków nie został spełniony. *)
   bCooler := 0; (*  Wyłącza chłodnicę, gdy żaden z powyższych warunków nie został spełniony. *)
END_IF;
```

5.2 Operacje warunkowe zgodnie z wartościami liczb całkowitych (CASE)

Wyrażenia IF są stosowane do rozgałęzień w zależności od tego, czy wyrażenia warunkowe zostały lub nie zostały spełnione.
Wyrażenia CASE są stosowane do rozgałęzień zgodnie z wartościami liczb całkowitych.
Poniższy rysunek przedstawia sposób zapisu wyrażenia CASE.

```
CASE Zmienna OF
    Wartość całkowita 1: Wyrażenie 1;   (* Wyrażenie 1 zostanie wykonane, gdy zmienna jest zgodna z wartością liczby całkowitej 1. *)
    Wartość całkowita 2: Wyrażenie 2;   (* Wyrażenie 2 zostanie wykonane, gdy zmienna jest zgodna z wartością liczby całkowitej 2. *)
    Wartość całkowita 3: Wyrażenie 3;   (* Wyrażenie 3 zostanie wykonane, gdy zmienna jest zgodna z wartością liczby całkowitej 3. *)
    ELSE Wyrażenie 4;                   (* Wyrażenie 4 zostanie wykonane, gdy zmienna nie jest zgodna z żadną wartością liczby całkowitej. *)
END_CASE;                               (* „END_CASE;" musi zostać wstawione na końcu wyrażenia CASE. *)
```

Poniższy rysunek przedstawia przebieg operacji w przypadku zastosowania wyrażenia CASE.



5.2.1 Przykładowy program z wyrażeniem CASE

Wykonanie wyrażenia CASE zostało opisane w oparciu o przykładową operację programu.

Kliknij , aby przejść do następnej strony.
Aby ponownie odtworzyć animację, kliknij poniżej przycisk „Odtwórz".

Odtwórz


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

Waga	uSize	Klasa
0 do 20 kg	1	M
21 do 30 kg	2	L
31 do 40 kg	3	XL
41 kg i więcej	4	Oversize

5.3 Podsumowanie

W rozdziale tym przedstawiono:

- Rozgałęzienia warunkowe z wyrażeniami IF
- Zapisywanie wyrażeń warunkowych
- Operacje warunkowe zgodnie z wartościami liczb całkowitych (Wyrażenie CASE)

Ważne kwestie do rozważenia:

Wyrażenie IF	• W przypadku wyrażenia IF program zostanie rozgałęziony, gdy wyrażenie warunkowe zostanie spełnione. • Wyrażenie ELSE jest wykorzystywane do operacji, gdy wyrażenie warunkowe nie zostanie spełnione. • Wyrażenie ELSIF jest wykorzystywane w celu dodania innego warunku, gdy wyrażenie warunkowe w wyrażeniu IF nie zostanie spełnione.
Wyrażenie warunkowe	• Wyrażenia warunkowe przedstawiają zależności pomiędzy zmiennymi a wartościami w celu porównania przy użyciu operatorów porównania.
Wyrażenie CASE	• Wyrażenia CASE są stosowane do rozgałęzień zgodnie z wartościami liczb całkowitych.

Rozdział 6 Zapisywanie i przetwarzanie danych

Tak samo, jak w przypadku zastosowań aplikacji do sterowania we/wy, obecnie do przetwarzania większej ilości danych stanowiących serce systemów produkcyjnych wykorzystuje się sterowniki programowalne.
W celu przetworzenia dużej ilości danych, muszą one zostać w pierwszej kolejności zapisane, a następnie odczytane w razie takiej konieczności.
W rozdziale tym przedstawiono sposób zapisywania zwięzłych programów w pamięci i przetwarzania danych.

- Do ustalenia kolejności i organizacji zmiennych wykorzystywane są tablice.
- Do organizacji powiązanych danych wykorzystywane są struktury danych.
- Programy przetwarzania w pętli skutecznie przetwarzają tablice przy użyciu pętli FOR.

Istnieje możliwość tworzenia programów zwięzłych umożliwiających przechowywanie i przetwarzanie danych za pomocą tablic, struktur danych i pętli FOR.

6.1 Ustawianie kolejności i przechowywanie danych (tablica)

Przy użyciu tablic istnieje możliwość przetworzenia wielu wartości za pomocą jednej zmiennej.
W przykładzie poniżej dane dot. wielkości produkcji zakładu produkcyjnego przemysłu motoryzacyjnego są przechowywane wg kraju przeznaczenia.

Miejsce przeznaczenia	Kraj A	Kraj B	Kraj C
Wielkość produkcyjna	35	75	65

Dane dot. wielkości produkcyjne wg. kraju przeznaczenia są przypisane do zmiennej. W przypadku braku korzystania z tablic od każdej zmiennej musi zostać utworzone miejsce przeznaczenia.
Dzięki zastosowaniu tablic wielkość produkcyjna dla wielu miejsc przeznaczenia może zostać przypisana i przechowywana w ramach jednej zmiennej.

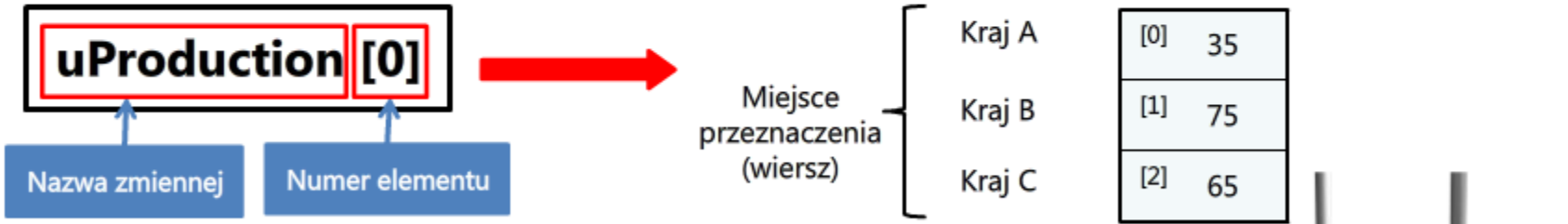
Brak zastosowania tablicy

```
uProductionA
uProductionB
uProductionC
```

Zastosowanie tablicy

```
uProduction
```

Pojedyncze zmienne w tablicy są określane za pomocą numeru elementu. Numery elementów rozpoczynają się od zera [0].



W poniższym przykładowym programie zmienna zaplanowanej wielkości produkcyjnej dla kraju A została przypisana.

```
uShowProductionPlan := uProduction[0];
(* określa numer elementu dla kraju A. *)
```

6.1.1 Tablica macierzy

Ponadto w przypadku danych dot. miejsca przeznaczenia stosowane są kolory zaznaczenia.

Miejsce przeznaczenia	Kraj A			Kraj B			Kraj C		
Kolor lakieru									
Wielkość produkcyjna	10	5	20	15	40	20	25	30	10
	35 łącznie			75 łącznie			65 łącznie		

Jak przedstawiono w poniższej tabeli dane mogą być oddzielone i przechowywane wg koloru zaznaczenia (kolumna) dla każdego kraju przeznaczenia (wiersz).

Kolor lakieru (kolumna)

Miejsce przeznaczenia (wiersz)	Czerwony	Żółty	Niebieski
Kraj A	[0,0] 10	[0,1] 5	[0,2] 20
Kraj B	[1,0] 15	[1,1] 40	[1,2] 20
Kraj C	[2,0] 25	[2,1] 30	[2,2] 10

Numer elementu określa miejsce przeznaczenia

Numer elementu określa Kolor lakieru

Tablice organizujące dane wg wierszy i kolumn w takim przypadku są zwane tablicami macierzy. Numery elementów określające wiersze i kolumny są oddzielane przecinkami.

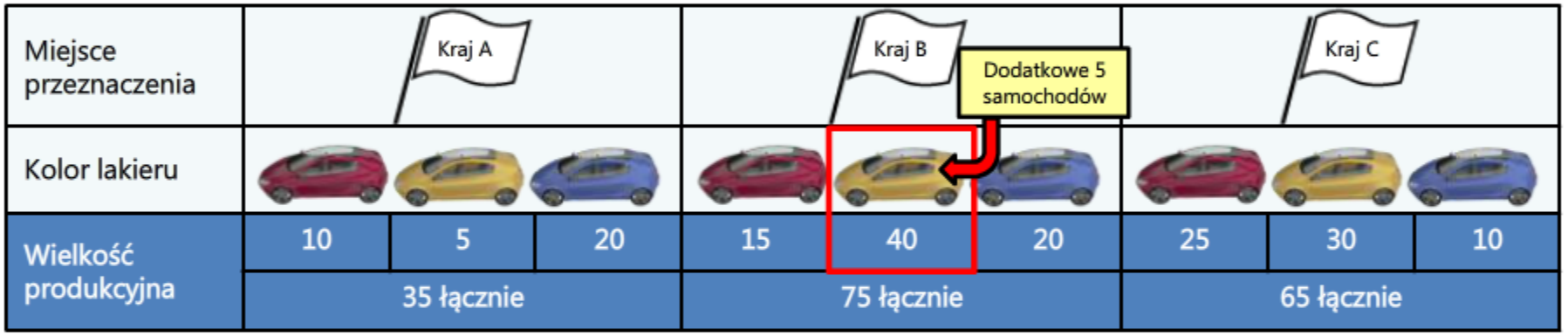
Zmienna tablicy (tablica macierzy)

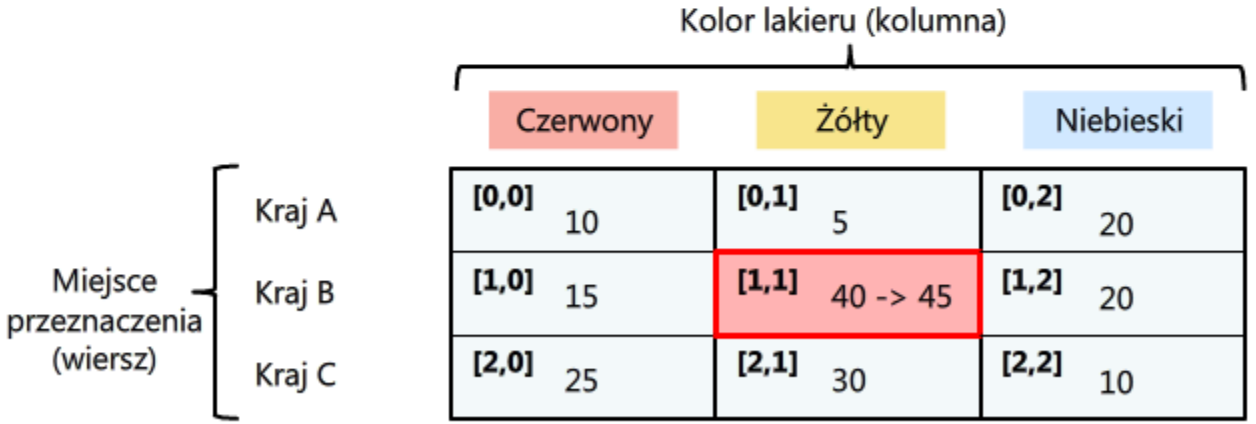
```
uProduction[1,1]
```

6.1.2 Przypisywanie tablicy macierzy

Korzystając z tablic macierzy poniższy przykładowy program przypisuje liczbę samochodów, które muszą zostać pilnie wyprodukowane dodatkowo do zaplanowanej wielkości produkcyjnej żółtych samochodów z kraju B.

```
uAdditionalProduction := 5;
uProduction[1,1] := uProduction[1,1] + uAdditionalProduction;
(* Dodaje dodatkową wielkość produkcyjną (5 sztuk) do wstępnie zaplanowanej wielkości produkcyjnej. *)
```





6.1.3 Przetwarzanie informacji zapisanych w tablicach macierzy

Przy użyciu tablic macierzy poniższy przykładowy program oblicza całkowitą zaplanowaną wielkość produkcyjną dla wszystkich kolorów lakierów dla kraju C i przypisuje wartość do zmiennej.

```
uProductionToday := uProduction[2,0] + uProduction[2,1] +uProduction[2,2];
(* Oblicza całkowitą wielkość produkcyjną zaplanowaną na dzisiaj dla wszystkich kolorów lakierów dla kraju C i przypisuje wartość do "uProductionToday". *)
```

Miejsce przeznaczenia	Kraj A			Kraj B			Kraj C		
Kolor lakieru									
Wielkość produkcyjna	10	5	20	15	45	20	25	30	10
	35 łącznie			80 łącznie			65 łącznie		

Kolor lakieru (kolumna)

Miejsce przeznaczenia (wiersz)	Czerwony	Żółty	Niebieski
Kraj A	[0,0] 10	[0,1] 5	[0,2] 20
Kraj B	[1,0] 15	[1,1] 45	[1,2] 20
Kraj C	[2,0] 25	[2,1] 30	[2,2] 10

6.2 Przetwarzanie w pętli (FOR)

Przykładowy program na poprzedniej stronie (zaplanowana wielkość produkcyjna na dzisiaj została przypisana) został ponownie przedstawiony poniżej.

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

W ramach tego przykładowego programu zwiększenie liczby kolorów zaznaczenia spowoduje dodanie większej liczby zmiennych. Wówczas wyrażenie staje się dłuższe, co utrudnia jego odczytanie.

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

W takim przypadku istnieje możliwość zastosowania wyrażeń pętli w celu utworzenia czytelniejszego kodu.

Wyrażenia pętli obejmują wyrażenia FOR, WHILE i REPEAT. W kursie tym zastosowano wyrażenia FOR.

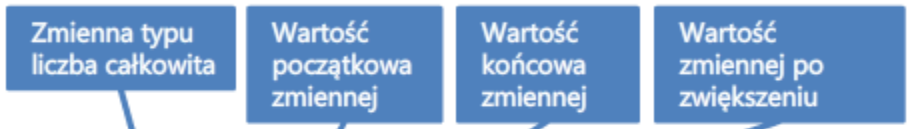
Wyrażenia FOR zostały opisane poniżej.

```
FOR Zmienna := Wartość początkowa TO Wartość końcowa BY inkrementacja DO
  Wyrażenie;          (* Wyrażenie jest wykonywane w pętli do czasu aż zmienna osiągnie wartość końcową. *)
END_FOR;              (* END_FOR; musi być umieszczane na końcu pętli FOR.  *)
```

Wyrażenie jest powtarzane do czasu aż osiągnięta zostanie wartość końcowa zmiennej i zostanie wykonane kod „END_FOR;".

6.2 Przetwarzanie w pętli (FOR)

Dzięki stosowaniu wyrażenia FOR poniższy przykładowy program otrzymuje zaplanowaną wielkość produkcyjną dla wszystkich kolorów lakieru dla kraju C.




```
uProductionToday := 0;                                    (* Inicjalizuje zmienną. *)
FOR wColor := 0 TO 2 BY 1 DO
  uProductionToday := uProductionToday + uProduction[2,wColor];   (* Dodaje zaplanowaną wielkość produkcyjną. *)
END_FOR;
```

Przy użyciu pętli FOR zmienna „wColor" zwiększa się o jeden, rozpoczynając od wartości początkowej wynoszącej zero i Wyrażenie jest powtarzane do momentu, gdy zmienna ta osiągnie wartość końcową wynoszącą dwa.
Zmienna „wColor" została określona jako drugi numer elementu w tablicy „uProduction" opisanej w wykonaniu wyrażenia. Wartość zmiennej „wColor" wzrasta po każdym powtórzeniu wyrażenia. Zaplanowana wielkość produkcyjna dla każdego koloru zaznaczenia jest dodawana za każdym razem w celu uzyskania wielkości całkowitej.

Ten przykładowy program jest wykonywane w pętli trzy razy. (Pierwszy raz: kolor czerwony [0] => Drugi raz: kolor żółty [1] => Trzeci raz: kolor niebieski [2])

Zasada działania niniejszego programu została przedstawiona na następnej stronie.

6.2 Przetwarzanie w pętli (FOR)

Wykonanie pętli FOR zostało przedstawione w oparciu o działanie przykładowego programu.

Tablica szacunkowej wielkości produkcyjnej

	Kolor czerwony	Kolor żółty	Kolor niebieski
Kraj A	[0,0] 10	[0,1] 5	[0,2] 20
Kraj B	[1,0] 15	[1,1] 45	[1,2] 20
Kraj C	[2,0] 25	[2,1] 30	[2,2] 10

Kliknij , aby przejść do następnej strony.
Aby ponownie odtworzyć animację, kliknij poniżej przycisk „Odtwórz".

Odtwórz

Number of repetition of the loop: 3

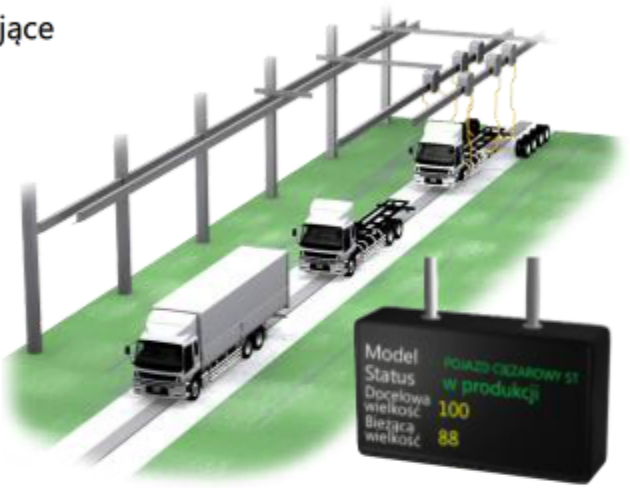
```
uProductionToday := 0;

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

6.3 Przechowywanie powiązanych danych (struktura)

Struktura umożliwia nazwie jednej zmiennej reprezentowanie wielu powiązanych zmiennych.
W poniższym przykładzie status samochodowej linii produkcyjnej został wyświetlony na Andon (tablica świetlna).

Poniższa tabela zawiera listę nazw zmiennych, wartości oraz typy danych odpowiadające wyświetlonym pozycjom.

Pozycja	Nazwa zmiennej	Wartość	Typ danych zmiennej
Model	sModel	„POJAZD CIĘŻAROWY ST"	Ciąg tekstowy
Status	bStatus	„w produkcji"	Typ bitowy
Docelowa wielkość produkcyjna na dzisiaj	uPlanQty	„100"	Słowo typu liczba całkowita (bez znaku)
Bieżąca wielkość produkcyjna	uActualQty	„88"	Słowo typu liczba całkowita (nieprzypisane)



Jeśli struktura nie jest wykorzystywana w programie, nazwy zmiennych muszą być zmieniane w przypadku każdej linii, gdy stosowanych jest wiele linii produkcyjnych.
Poniżej przedstawiono przykłady nazw zmiennych wg linii produkcyjnej.

Pierwsza linia produkcyjna

s1stLineModel
b1stLineStatus
u1stLinePlanQty
u1stLineActualQty

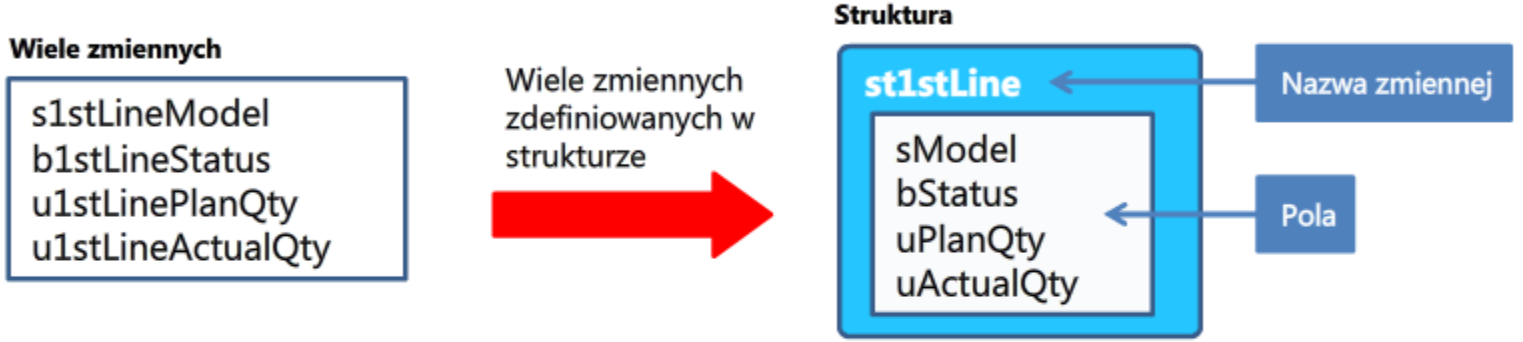
Druga linia produkcyjna

s2ndLineModel
b2ndLineStatus
u2ndLinePlanQty
u2ndLineActualQty

Jeśli liczba linii produkcyjnych zwiększy się, liczba zmiennych do przetworzenia również zwiększy się.
Wówczas program staje się dłuższy i bardziej skomplikowany do odczytania.

6.3 Przechowywanie powiązanych danych (struktura)

Korzystanie ze struktur umożliwia nazwie jednej zmiennej reprezentowanie wielu zmiennych w ramach jednej linii produkcyjnej. Jak poniżej, struktury są stosowane w celu zorganizowania, przechowywania i przetwarzania danych w partiach wg warunków i specyfikacji fizycznych obiektów, takich jak urządzenia, sprzęt i przedmioty obrabiane.



Structure variable (Zmienna struktury) zawiera prefiks „st" oznaczający, że to jest struktura.
Indywidualne zmienne zdefiniowane przez strukturę są znane jako pola. Typy danych każdego pola mogą różnić się od siebie.

Każde pole tablic struktur można określić po numerze pola tablicy za pomocą kropki przed nazwą pola.

W poniższym przykładowym programie stała została przypisana do pola zmiennej struktury dla pierwszej linii produkcyjnej.

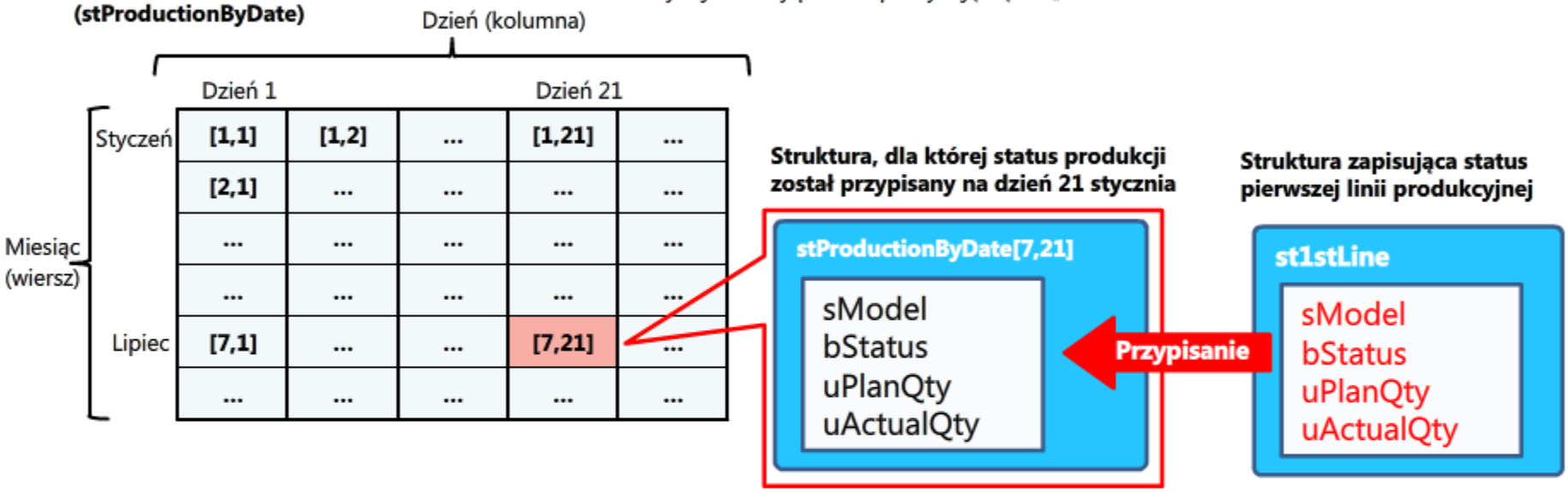
```
st1stLine.uPlanQty := 150;
(* Ustawia codzienną docelową wielkość produkcyjną pierwszej linii produkcyjnej na wartość 150. *)
```

6.3.1 Zapisywanie tablic struktur

Struktury można tworzyć w formie tablic.
W poniższym przykładzie status produkcji został zapisany wg daty.

Struktura rozplanowana* wg daty (stProductionByDate)

* W tablicy tej numery pola rozpoczynają się od „1".

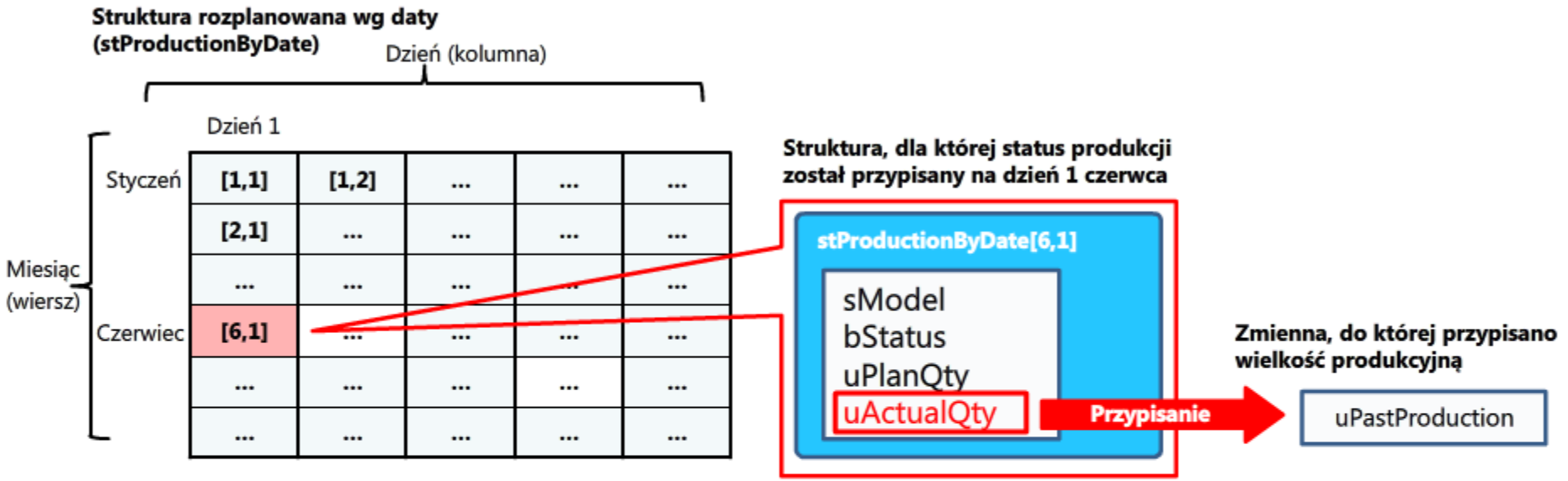



```
stProductionByDate[7,21] := st1stLine;
(* Status produkcji na dzień 21 stycznia został zapisany w strukturze
rozplanowanej wg daty (stProductionByDate). *)
```

Jak poniżej, pola nie wymagają indywidualnego określenia w ramach przypisania struktury.

6.3.2 Odczytywanie tablic struktur

W poniższym przykładzie wielkość produkcyjna jest odczytywana ze struktury rozplanowanej wg daty, a następnie jest przypisywana do zmiennej.




```
uPastProduction := stProductionByDate[6,1].uActualQty;
(* Przypisuje wielkość produkcyjną na dzień 1 czerwca do zmiennej uPastProduction. *)
```

Każde pole tablic struktur można określić poprzez dodanie kropki (.) i nazwy pola do numeru pola tablicy.

6.4 Podsumowanie

W rozdziale tym przedstawiono:

- Przegląd i zastosowanie tablic
- Przetwarzanie w pętli przy użyciu pętli FOR
- Przegląd i zastosowanie struktur

Ważne kwestie do rozważenia:

Tablica	• Dzięki użyciu tablicy istnieje możliwość przetworzenia wielu wartości za pomocą jednej zmiennej. • Indywidualne zmienne w tablicach są określane poprzez dodanie numerów pól na końcu nazw zmiennych.
Pętla FOR	• Wyrażenia pętli są stosowane w programach, gdy wymagane jest powtarzanie operacji. • Wyrażenia FOR są stosowane do powtarzania operacji do momentu spełnienia warunków końca operacji pętli. Wyrażenie przed Wyrażeniem „END_FOR;" są wykonywane wielokrotnie.
Struktura	• Struktury umożliwiają nazwie jednej zmiennej reprezentowanie wielu powiązanych zmiennych. Struktury mogą zawierać zmienne różnych typów danych. • Indywidualne zmienne lub pola definiowane w strukturze są określane poprzez dodawanie kropki i nazwy pola po nazwie zmiennej struktury.

Rozdział 7 Przetwarzanie danych typu ciąg znaków

W niektórych przypadkach sterowniki programowalne stosują dane typu ciąg znaków w celu wysyłania poleceń lub otrzymywania sprzężenia zwrotnego z podłączonych urządzeń, takich jak czytniki kodów kreskowych, sterowniki temperatur lub wagi elektroniczne. Do tego tupu celów konieczne jest dołączenie lub pozyskanie danych typu ciąg znaków zgodnie z wymaganiami.

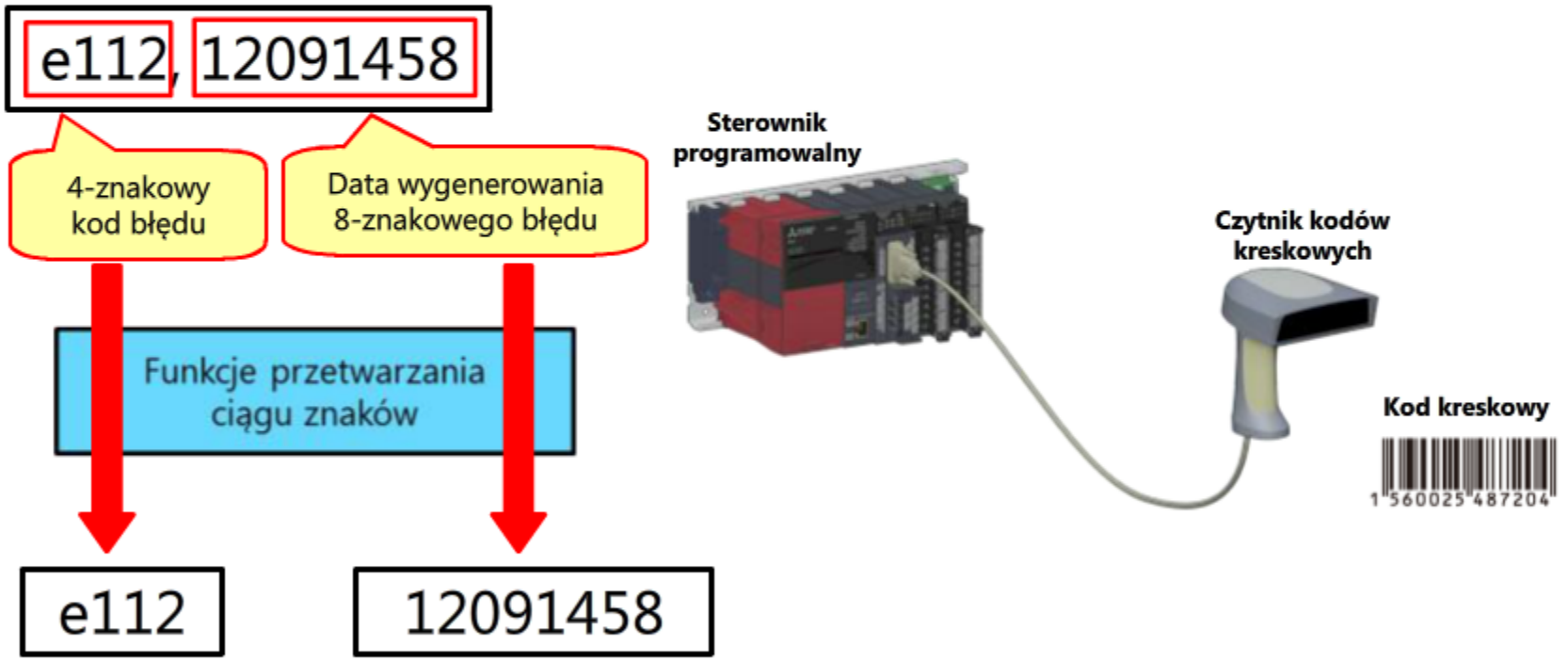
W rozdziale tym przedstawiono sposób przetwarzania danych typu ciąg znaków.

7.1 Przykładowe przetwarzanie danych typu ciąg znaków

W ramach przykładowego przetwarzania danych typu ciąg znaków przedstawiono scenariusz, w którym dane są odczytywane z czytnika kodów kreskowych.
W celu przetworzenia danych typu ciąg znaków stosowane są funkcje (typ instrukcji).

Zgodnie z poniższą instrukcją dane typu ciąg znaków odczytywane przez czytnik kodów kreskowych zawierają kod błędu o stałej długości 4 znaków oraz dane miesiąca, daty, godziny i minut o stałej długości 8 znaków.
Przykładowy program przetwarzania danych typu ciąg znaków zostanie przedstawiony przy użyciu tego systemu.

Przykładowy odczyt danych typu ciąg znaków z czytnika kodów kreskowych



Pozyskano kod błędu.
7.3 Pozyskiwanie ciągów znaków (LEFT)

Pozyskano datę i godzinę wystąpienia błędu (14:58, 9 grudnia).
7.4 Pozyskiwanie ciągów znaków (MID)

7.2 Przypisanie ciągu znaków

Przed Wyrażeniem sposobu pozyskiwania ciągu znaków w punkcie tym przedstawiono opis typów danych dla ciągów znaków.

Typy danych dla ciągów znaków, które można użyć w sterownikach programowalnych zostały wymienione w poniższej tabeli.

Typ danych	Można przetworzyć dane typu znak	Prefiksy notacji węgierskiej	Rozszerzenie prefiksu
Ciąg znaków	Ciągi znaków alfanumerycznych i liczbowych (ASCII) lub japońskich (Shift-JIS)	s	**s**tring (ciąg znaków)
Ciąg znaków [Unicode]	Ciągi znaków wielu różnych języków i symboli	ws	**w**ide **s**tring (szeroki ciąg znaków)

Typ ciągu znaków, jaki należy użyć, zależy od urządzenia podłączonego do sterownika programowalnego lub odpowiadającego języka.
W rozdziale tym przedstawiono różne typy tekstowych ciągów znaków.

Po przypisaniu typu ciągu znaków do zmiennej ciągu znaków, zamknij ciąg znaków w pojedynczym cudzysłowie (').

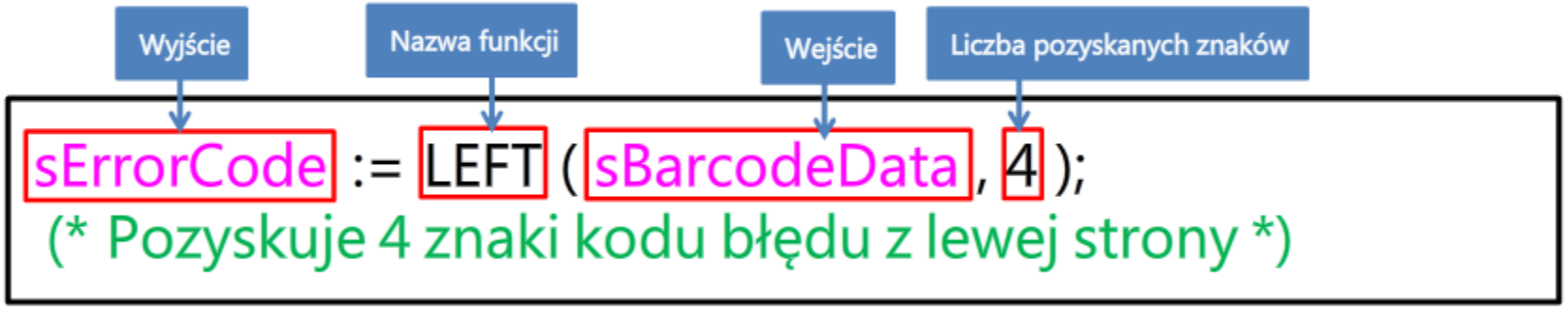
```
sDefault := 'e112,12091458';  (* Przypisanie ciągu znaków *)
```

7.3 Pozyskiwanie ciągów znaków (LEFT)

Kod błędu „e112" został pozyskanie ze zmiennej ciągu znaków „sBarcodeData", która zwierała ciąg znaków „e112,12091458".

Nazwa zmiennej	Pozyskany ciąg znaków
sBarcodeData	e112, 12091458

Funkcja LEFT pozyskuje wyłącznie określoną liczbę znaków rozpoczynającą się z lewej strony ciągu znaków wejścia.
Poniżej przedstawiono przykładowy program.



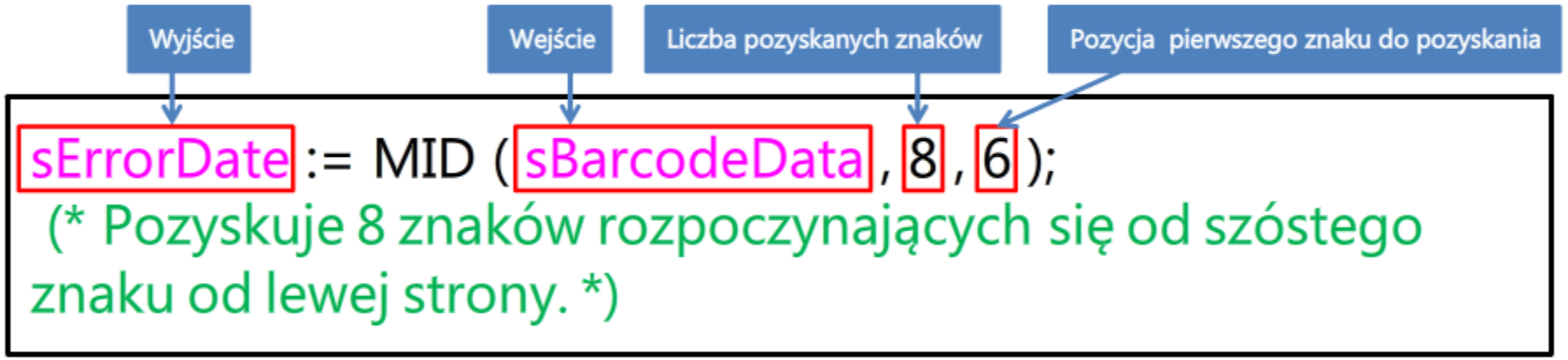
Pozyskano cztery znaki z lewej strony. Wartość „e112" stanowiąca ciąg znaków określających kod błędu została przypisana z lewej strony.

7.4 Pozyskiwanie ciągów znaków (MID)

Czas generowania błędu „12091458" został pozyskany ze zmiennej ciągu znaków „sBarcodeData", która zwiera ciąg znaków „e112,12091458".

Nazwa zmiennej	Pozyskany ciąg znaków
sBarcodeData	e112,12091458

Funkcja MID pozyskuje wyłącznie określoną liczbę znaków od określonego miejsca początkowego w ciągu znaków wejścia. Poniżej przedstawiono przykładowy program.



W przykładzie tym 8-znakowy ciąg znaków został pozyskany, rozpoczynając od szóstego znaku. Wartość „12091458" stanowiąca ciąg znaków określających czas wystąpienia błędu został przypisany z lewej strony.

7.5 Podsumowanie

W rozdziale tym przedstawiono:

- Metody przypisywania ciągu znaków do zmiennych ciągów znaków
- Funkcje pozyskujące ciągi znaków (LEFT i MID)

Ważne kwestie do rozważenia:

Przypisanie ciągu znaków	• Aby przypisać ciąg znaków do zmiennej ciągu znaków, zamknij ciąg znaków w pojedynczym cudzysłowie ('). • Użyj typu ciągu znaków lub typu ciągu znaków [Unicode] zgodnie z urządzeniem podłączonym do sterownika programowalnego lub odpowiadającego języka.
Funkcje do przetwarzania ciągu znaków	• Funkcje umożliwiają przetwarzanie ciągu znaków.

7.6 Podsumowanie kursu

W rozdziale tym przedstawiono podstawowe informacje dotyczące sposobu tworzenia programów w ST.
Oznacza to ukończenie kursu e-learningowego.

Programy ST są tworzone przy użyciu oprogramowania inżynierskiego MELSOFT.
W celu uzyskania szczegółowych informacji na temat określonych funkcji, takich jak wprowadzanie danych, edytowanie, zapisywanie i łączenie programów za pomocą oprogramowania inżynierskiego MELSOFT należy zapoznać się z poniższą dokumentacją.

- Mitsubishi FA e-Learning Course „MELSOFT GX Works3 (Structured Text)" (Narzędzie do e-Learningu Mitsubishi FA „MELSOFT GX Works3 (tekst strukturalny)") (zostanie wkrótce wydane)
- Instrukcja obsługi oprogramowania inżynierskiego MELSOFT

W celu zyskania szczegółowych informacji na temat ST należy zapoznać się z poniższą dokumentacją.

- Przewodnik programowania sterownika programowalnego

W celu zyskania szczegółowych informacji na temat instrukcji i funkcji aplikacji należy zapoznać się z poniższą dokumentacją.

- Instrukcja programowania sterownika programowalnego

Test — Test końcowy

Po zakończeniu wszystkich etapów kursu Podstawy programowania (tekst strukturalny) możesz teraz przystąpić do testu końcowego. W razie niejasności w zakresie któregokolwiek z tematów, wykorzystaj tę możliwość do ponownego zapoznania się z tymi zagadnieniami.
Test końcowy składa się z 12 pytań (20 elementów).
Możesz zdawać test końcowy dowolną ilość razy.

Jak rozwiązywać test
Po wybraniu odpowiedzi upewnij się, że przycisk **Odpowiedź** został kliknięty. Twoja odpowiedź zostanie utracona, jeśli będziesz kontynuować bez kliknięcia przycisku Odpowiedź. (Zostanie potraktowana jako pytanie, na które nie udzielono odpowiedzi.)

Punktacja końcowa
Liczba prawidłowych odpowiedzi, liczba pytań, procent prawidłowych odpowiedzi i wynik zaliczony/niezaliczony pojawią się na stronie wyniku.

Prawidłowe odpowiedzi: 4

Wszystkie pytania: 4

Procent prawidłowych odpowiedzi: 100%

Aby zaliczyć test, musisz odpowiedzieć poprawnie na **60%** pytań.

Kontynuuj | Przeglądaj

- Kliknij przycisk **Kontynuuj**, aby zakończyć test.
- Kliknij przycisk **Przeglądaj**, aby przeglądać test. (Sprawdzenie prawidłowych odpowiedzi)
- Kliknij przycisk **Spróbuj ponownie**, aby powtórzyć test.

Test
Test końcowy 1

Charakterystyka tekstu strukturalnego (ST)
Wybierz nieprawidłowy opis ST.

- ST jest łatwym językiem do nauczenia dla osób mających doświadczenie w pisaniu programów w języku C lub BASIC.
- Obliczenia, takie jak dodawanie i odejmowanie, mogą być zapisywane jako typowo stosowane wyrażenia matematyczne.
- Symbole styków i cewek są stosowane w celu utworzenia programu podobnego do obwodu elektrycznego.
- ST doskonale sprawdza się w przypadku przetwarzania danych.

Odpowiedź | Wstecz

Test

Test końcowy 2

Podstawowe zasady ST
Wybierz prawidłowe wyrażenie zapisane w ST.

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

Odpowiedź Wstecz

Test

Test końcowy 3

Opisywanie komentarzy
Wybierz prawidłowy komentarz zapisany w ST.

- ' Przypisuje wartość 1 do zmiennej.
- (* Przypisuje wartość 1 do zmiennej. *)
- { Przypisuje wartość 1 do zmiennej. }
- <!-- Przypisuje wartość 1 do zmiennej. -->

Odpowiedź Wstecz

Test końcowy 4

Sekwencja wykonywania programu ST
*Wartość początkowa „uTotalProduction” wynosi „100”. Wartość zmiennej „uTotalProduction” będzie wynosiła „101” po przetworzeniu przykładowego programu. Wybierz prawidłowy status „uTotalProduction” po kilku sekundach.

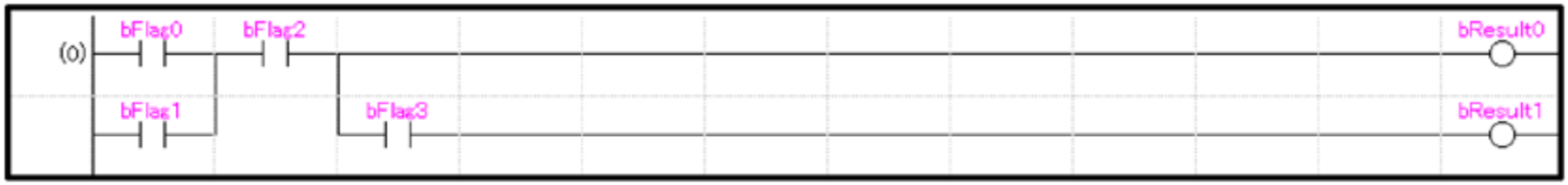
```
uTotalProduction := uTotalProduction + 1;
```

- ○ Wartość pozostaje
- ○ Wartość zmienia się.

Odpowiedź Wstecz

Test końcowy 5

Łączenie wielu warunków
Wybierz prawidłowy przykładowy program ST przedstawiający tę samą operację, jak następujący przykładowy program zapisany w LD.



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

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

Odpowiedź | Wstecz

Test

Test końcowy 6

Opis wyjaśnień IF w ST
Następująca operacja jest wykonywana przez poniższy przykładowy program.
- Jeśli temperatura spadnie do 5 stopni lub niżej, nagrzewnica włączy się, a chłodnica wyłączy.
- Jeśli temperatura przekroczy 50 stopni, nagrzewnica wyłączy się, a chłodnica włączy.
- Jeśli temperatura nie jest zgodna z powyższymi wyjaśnieniami, zarówno nagrzewnica, jak i chłodnica pozostaną wyłączone.

*Nazwy zmiennych: Temperatura (wTemperature), nagrzewnica (bHeater) i chłodnica (bCooler)
Dokonaj prawidłowego wyboru każdej pustej sekcji przykładowego programu.

```
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--

Odpowiedź Wstecz

Test końcowy 7

Wyjaśnienia CASE

Wybierz prawidłowe jedno wyrażenie dla każdej zmiennej (Q1 do Q5) dla następujących opisów wyrażeń CASE.

Wyjaśnienia CASE są stosowane do rozgałęzień zgodnie z wartością (Q1).
W poniższym przykładowym programie, gdy wartość (Q2) wynosi 25, zmiennej (Q3) jest przypisywana wartość (Q4).
Gdy wartość zmiennej (Q2) nie jest równa 10, 25 lub 8, zmiennej (Q3) jest przypisywana wartość (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--

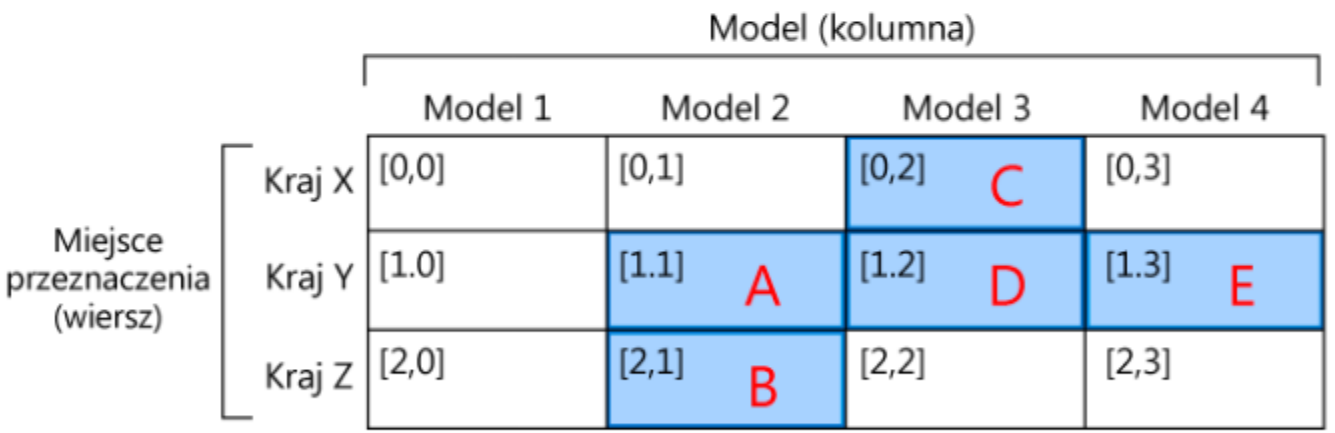
Odpowiedź Wstecz

Test — Test końcowy 8

Tablice ST i operacje powtarzające się
Poniższy przykładowy program sumuje zaplanowaną wielkość produkcyjną wszystkich modeli w kraju przeznaczenia Y, a następnie przypisuje tę wartość do zmiennej. Wybierz sekcję tablicy, która jest odczytywana po 3-krotnym wykonaniu wyjaśnienia FOR w pętli.

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

Tablica ma na celu zapisanie szacunkowej liczby sztuk wyprodukowanej na dany model i dane miejsce przeznaczenia (uProduction)



- A
- B
- C
- D

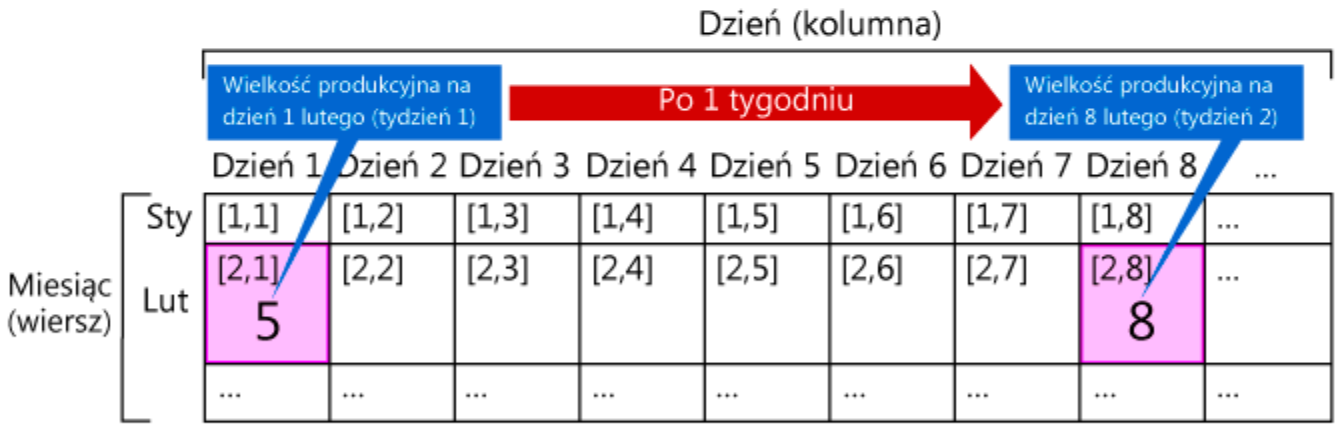
Odpowiedź | Wstecz

Test — Test końcowy 9

Tablice ST i wyjaśnienia powtarzające się
Poniższy przykładowy program uzyskuje całkowitą wielkość produkcyjną w tych samych dniach tygodnia. Wartość całkowita po 4 tygodniach jest uzyskiwana z tablicy zapisującej wielkość produkcyjną każdego dnia. Wybierz prawidłowy rysunek przykładowego programu.

```
uTotalProduction := 0;
FOR wOnceAWeek := 1 TO ■ BY 7 DO
  uTotalProduction := uTotalProduction + uProductionByDate[2,wOnceAWeek];
END_FOR;
(* Pozyskuje i sumuje wielkość produkcyjną w te same dni tygodnia przez 4 tygodnie począwszy od 1 lutego. *)
```

Tablica, która zapisuje dzienną wielkość produkcyjną (uProductionByDate)



- 22
- 21
- 4
- 28

Odpowiedź | Wstecz

Test
Test końcowy 10

Charakterystyka struktur w ST
Wybierz nieprawidłowy opis struktur.

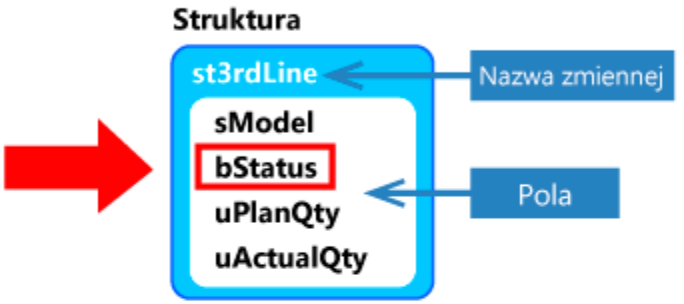
- Struktury są stosowane w celu zorganizowania i przechowywania danych na urządzeniach wg warunków, takich jak status i specyfikacja.
- Programy przetwarzające duże ilości danych mogą zastać zwięźle zapisane przy pomocy struktur.
- Wszystkie pola zdefiniowane w strukturze muszą zawierać ten sam typ danych.
- Wartości mogą być przypisywane do pól w tej samej strukturze bez konieczności ich indywidualnego określania.

Odpowiedź | Wstecz

Test
Test końcowy 11

Określanie pól struktur w ST
Poniższa struktura porządkuje zmienne powiązane z samochodową linią produkcyjną.
Wybierz prawidłowy opis określający pole „bStatus" w tej strukturze.

Parametr	Nazwa zmiennej
Model	sModel (sModel)
Status	bStatus (bStatus)
Produkcja docelowa w bieżącym dniu	uPlanQty (uWielProd)
Bieżąca wielkość produkcyjna	uActualQty (uBieżWiel)



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

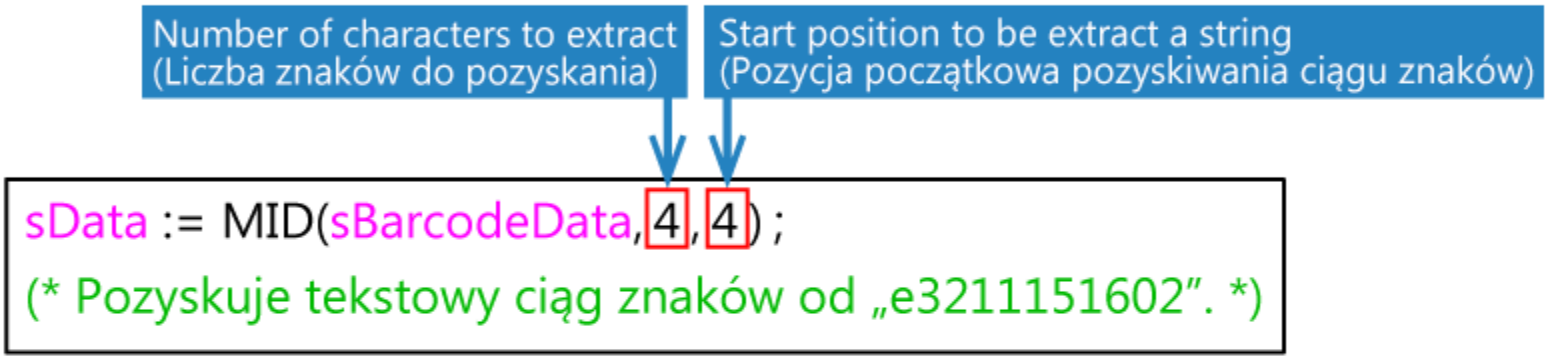
Odpowiedź Wstecz

Test — Test końcowy 12

Przetwarzanie ciągów znaków w ST

Poniższy przykładowy program pozyskuje określony ciąg znaków z ciągu znaków „e3211151602” zapisanego w zmiennej „sBarcodeData”. Funkcja MID pozyskuje określoną liczbę znaków od określonego miejsca początkowego w ciągu znaków.

Wybierz prawidłowo pozyskany ciąg znaków.



- 1151
- 1602
- e321
- 1115

Odpowiedź | Wstecz

Test Wynik testu

Test końcowy został zakończony. Twoje wyniki są przedstawione poniżej.
Aby zakończyć test końcowy, przejdź do następnej strony.

Prawidłowe odpowiedzi:	12
Wszystkie pytania:	12
Procent prawidłowych odpowiedzi:	100%

Kontynuuj Przeglądaj

Gratulacje. Test został zaliczony.

Kurs Podstawy programowania (tekst strukturalny) został ukończony.

Dziękujemy za wzięcie udziału w kursie.

Mamy nadzieję, że poruszone tematy były interesujące, a informacje uzyskane w trakcie tego kursu będą przydatne w przyszłości.

Możesz przeglądać kurs dowolną ilość razy.

Przeglądaj Zamknij