MITSUBISHI ELECTRIC
Changes for the Better

# PLC

# Dasar-Dasar Pemrograman (Structured Text)

Kursus ini membahas cara membuat program dasar yang dibuat untuk mengontrol pengontrol terprogram MELSEC. Structured text (ST) digunakan untuk deskripsi program dalam kursus ini.



L(FOD)00549IND

Pendahuluan

# Tujuan kursus ini

Kursus ini menjelaskan cara membuat program kontrol dalam structured text (ST) untuk pengontrol terprogram MELSEC.

Penyelesaian kursus berikut atau memiliki pengetahuan yang setara merupakan prasyarat sebelum mengambil kursus ini:

Dasar-Dasar Pemrograman

Memiliki pengetahuan atau pengalaman dengan bahasa pemrograman C atau BASIC dapat membantu memahami isi dari kursus ini.

Pendahuluan

# Struktur Kursus

Berikut adalah daftar isi kursus.

**Bab 1 - Tinjauan structured text**

Bab ini membahas fitur dan aplikasi structured text (ST) yang sesuai.

**Bab 2 - Aturan dasar program ST**

Bab ini menguraikan aturan dasar yang digunakan untuk membuat program dalam ST.

**Bab 3 - Membuat program kontrol I/O**

Bab ini menguraikan cara membuat program kontrol I/O.

**Bab 4 – Operasi aritmetika**

Bab ini menguraikan cara membuat program operasi aritmetika.

**Bab 5 - Percabangan konsidional**

Bab ini menguraikan tentang percabangan kondisional.

**Bab 6 - Penyimpanan dan penanganan data**

Bab ini menguraikan tentang cara menulis program ringkas untuk menyimpan dan menangani data.

**Bab 7 – Penanganan data string**

Bab ini menguraikan metode untuk menangani data string.

**Tes Akhir**

Nilai kelulusan: 60% atau lebih tinggi

**Pendahuluan**

# Cara menggunakan alat e-Learning ini

| | | |
|---|---|---|
| Buka halaman berikutnya |  | Buka halaman berikutnya. |
| Kembali ke halaman sebelumnya |  | Kembali ke halaman sebelumnya. |
| Beralih ke halaman yang diinginkan |  | "Daftar Isi" akan ditampilkan, memungkinkan Anda untuk menavigasi ke halaman yang diinginkan. |
| Keluar dari kursus |  | Keluar dari kursus. |

# Peringatan penggunaan

## Petunjuk keselamatan

Saat Anda belajar dengan memakai produk sebenarnya, bacalah dengan cermat petunjuk keselamatan pada panduan yang sesuai.

## Petunjuk keselamatan dalam kursus ini

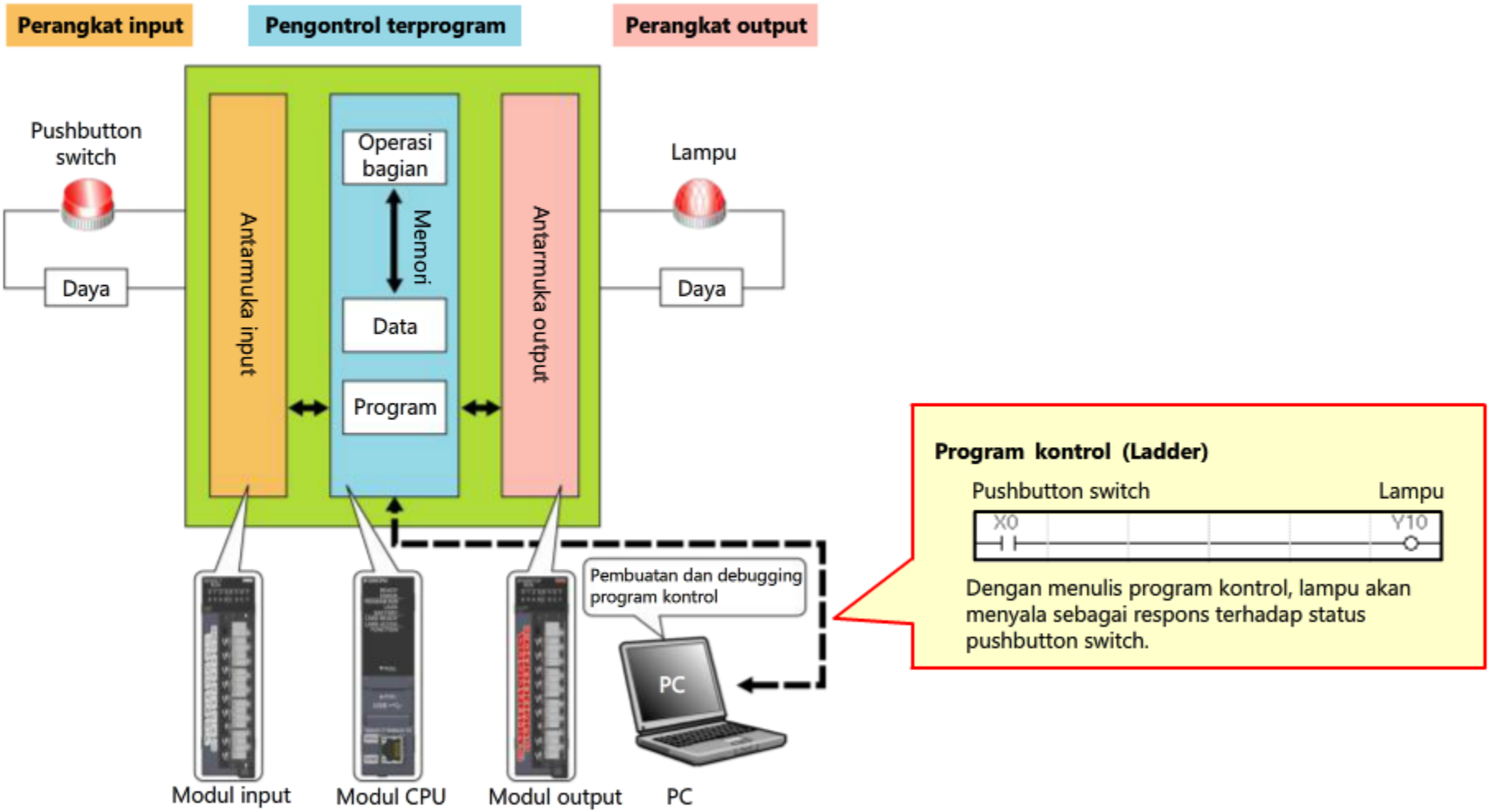
Layar yang ditampilkan pada perangkat lunak teknik MELSOFT yang Anda gunakan mungkin berbeda dengan yang ada di dalam kurus ini.
Kursus ini menggunakan simbol ladder MELSOFT GX Works3 untuk membuat program.

# Bab 1 Tinjauan structured text

Bab ini membahas fitur dan aplikasi structured text (ST) yang sesuai.

# 1.1 Kontrol program

Gambar berikut ini mengilustrasikan konfigurasi sistem PLC.
Pengontrol terprogram beroperasi sesuai dengan program kontrol.
Operasi pengontrol terprogram dapat dikonfigurasi seperti yang diinginkan dengan membuat program kontrol.




Bahasa pemrograman untuk pengontrol terprogram ditentukan oleh standar internasional yang dikembangkan oleh International Electrotechnical Commission (IEC) (Komisi Elektroteknik Internasional).

# 1.2 Fitur ST dan perbandingan dengan bahasa pemrograman IEC lain

IEC 61131 adalah standar internasional untuk sistem PLC.
Bahasa pemrograman untuk pengontrol terprogram distandarkan oleh IEC 61131-3. ST adalah salah satu bahasa pemrograman standar.
Setiap bahasa memiliki fitur berbeda untuk mengakomodasi aplikasi dan keterampilan programmer Anda.

Tabel berikut mencantumkan fitur IEC 61131-3 bahasa pemrograman.

| bahasa pemrograman | Fitur |
|---|---|
| **L**adder **D**iagram (LD)<br>(Diagram ladder) | • Simbol untuk kontak dan kumparan digunakan untuk membuat program yang mirip dengan sirkuit elektrik.<br>• Aliranprogram mudah untuk diikuti dan dipahami, bahkan untuk pemula. |
| **S**tructured **T**ext (ST)<br>(Teks Terstruktur) | • Program ditulis dalam teks (karakter).<br>• ST mudah dipelajari bagi orang yang sudah berpengalaman menulis program dalam bahasa pemrograman C atau BASIC.<br>• Rumus perhitungan mirip dengan ekspresi matematika, yang mudah dipahami.<br>• ST cocok untuk penanganan data. |
| **F**unction **B**lock **D**iagram (FBD)<br>(Diagram blok fungsi) | • Program ditulis dengan menyusun blok dengan berbagai fungsi dan mengindikasikan hubungan di antara blok.<br>• FBD Meningkatkan keterbacaan karena seluruh operasi dapat dilihat dengan mudah. |
| **S**equential **F**unction **C**hart (SFC)<br>(Diagram fungsi sekuensial) | • Kondisi dan proses ditulis sebagai diagram aliran.<br>• Aliran program mudah dipahami. |
| **I**nstruction **L**ist (IL)<br>(Daftar instruksi) | • IL mirip dengan bahasa mesin.<br>• IL jarang digunakan sekarang ini. |

Kursus ini menjelaskan cara untuk menulis program kontrol dasar menggunakan ST.

# 1.3 Ringkasan

Isi dari bab ini adalah:

- Hubungan antara sistem PLC dan program kontrol
- Standar internasional untuk program kontrol
- Fitur ST

Poin penting untuk dipertimbangkan:

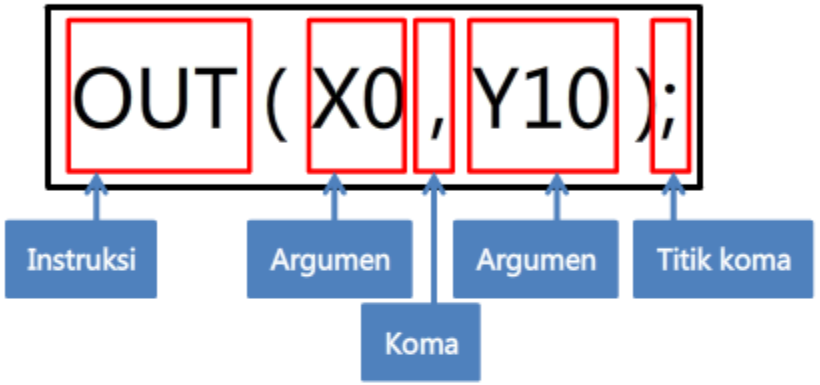
| | |
|---|---|
| Hubungan antara sistem PLC dan program kontrol. | • Pengontrol terprogram beroperasi sesuai dengan program kontrol.<br>• Operasi pengontrol terprogram dapat dikonfigurasi seperti yang diinginkan dengan membuat program kontrol. |
| Standar internasional untuk program kontrol | • ST adalah salah satu bahasa pemrograman IEC.<br>• Bahasa pemrograman IEC lainnya antara lain LD, FBD, SFC, dan IL, masing-masing memiliki fitur berbeda untuk mengakomodasi aplikasi dan keterampilan programmer Anda. |
| Fitur ST | • ST mudah dipelajari bagi orang yang sudah berpengalaman menulis program dalam bahasa C atau BASIC.<br>• Kalkulasi seperti penambahan dan pengurangan bisa ditulis sebagai ekspresi matematika yang lazim digunakan, yang mudah dipahami.<br>• ST cocok untuk penanganan data. |

# Bab 2 Aturan dasar program dalam ST

Bab ini menguraikan aturan dasar yang digunakan untuk membuat program dalam ST.

# 2.1 Contoh program dasar (Pernyataan kontrol I/O)

Bagian ini mengilustrasikan contoh program ST dasar.
Pada contoh program berikut, output Y10 menyala ketika input X0 menyala, dan Y10 mati ketika X0 mati.




Instruksi menentukan operasi yang akan dijalankan.
Argumen ditulis dalam tanda kurung setelah instruksi.
Argumen digunakan untuk menguraikan variabel, ekspresi aritmetika, dan nilai konstanta.
Dengan pengontrol terprogram MELSEC, perangkat modul CPU dapat digunakan sebagai variabel.

Jumlah argumen bergantung pada instruksi.
Argumen yang lebih dari satu dipisahkan oleh koma (,).

Satu jalur yang ditunjukkan di atas mewakili satu pernyataan. Setiap pernyataan diakhiri dengan titik koma (;).
Program ditulis dengan mengombinasikan pernyataan.

# 2.2 Contoh program dasar (Pernyataan pemilihan)

Contoh berikut ini mengilustrasikan program yang menggunakan pernyataan pemilihan.
Pernyataan berikut ini memilih konstanta desimal "5" ke variabel "D10".


Operator pemilihan (:=) digunakan untuk pernyataan pemilihan ini. Perhatikan bahwa titik dua (:) ditempatkan di sebelah kiri tanda sama dengan (=).
Operator pemilihan menetapkan nilai sisi kanan ke sisi kiri.

Menambahkan komentar ke program membuat operasi lebih mudah dipahami. Masukkan komentar di antara dua bintang (* *).

# 2.3 Notasi numerik

Dengan contoh program di halaman sebelumnya, nilai desimal dipilih ke variabel.

Kadang nilai selain desimal seperti biner dan heksadesimal digunakan untuk kontrol sekuensial.
Tabel berikut ini mencantumkan tipe notasi numerik yang digunakan dalam ST digunakan untuk pengontrol terprogram MELSEC.

| Tipe notasi numerik | Metode notasi | Contoh |
|---|---|---|
| Biner | Tambahkan awalan "2#". | **2#**11010 |
| Octal | Tambahkan awalan "8#". | **8#**32 |
| Desimal | Input langsung | 26 |
| | Tambahkan awalan "K". | **K**26 |
| Heksadesimal | Tambahkan awalan "16#". | **16#**1A |
| | Tambahkan awalan "H" | **H**1A |

Contoh program untuk menetapkan nilai ke variabel ditunjukkan di bawah ini.

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

# 2.3.1 Notasi bit

Bit mewakili kondisi true/false (benar/salah) seperti status sinyal on/off (nyala/mati). Bit juga mewakili kondisi pendirian/non-pendirian.
Dalam ST, bit tidak dapat ditulis sebagai "ON" dan "OFF". Bit diekspresikan sebagai "1" (ON) dan "0" (OFF). Bit juga dapat diekspresikan sebagai "TRUE" dan "FALSE".

Tabel berikut ini mencantumkan berbagai tipe notasi.

| Status | ON | OFF |
|---|---|---|
| | True | False |
| Notasi numerik | 1 | 0 |
| Notasi true/false | TRUE | FALSE |

Berikut ini beberapa contoh memilih nilai ke variabel tipe bit.

Notasi numerik

```
X0 := 1;
```

=

Notasi true/false

```
X0 := TRUE;
```

Notasi numerik

```
X0 := 0;
```

=

Notasi true/false

```
X0 := FALSE;
```

# 2.4 Urutan menjalankan program

Pernyataan ST dijalankan dengan urutan dari atas ke bawah.

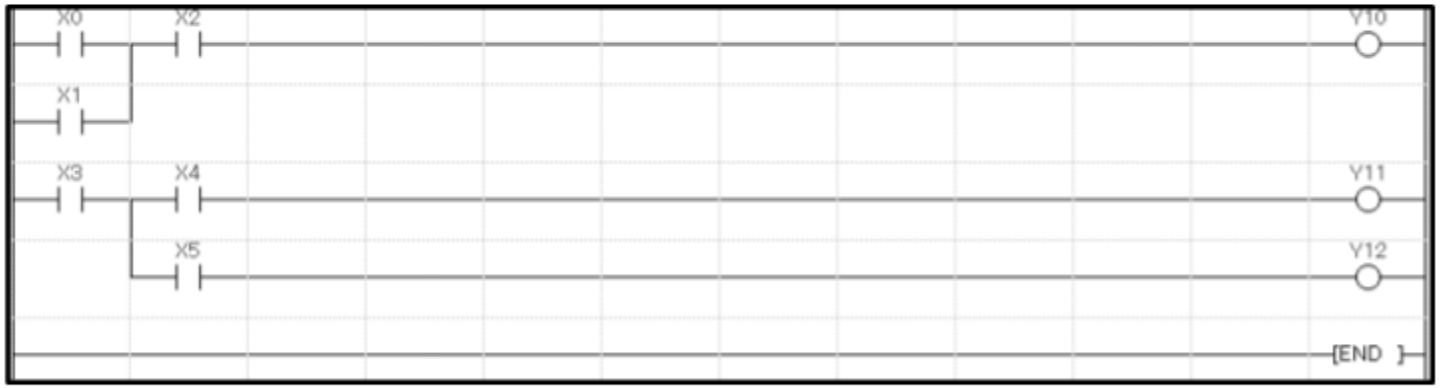
Contoh program ST

```
Y10 := (X0 OR X1) AND X2;        (* Pertama dijalankan *)
Y11 := X3 AND X4;                (* Kedua dijalankan *)
Y12 := X3 AND X5;    (* Ketiga dijalankan. Tidak memerlukan pernyataan END di bagian akhir.*)
```

Dijalankan berulang

*Meski pernyataan END diperlukan di bagian akhir program dalam LD, pernyataan END tidak diperlukan di ST.

Program Ladder berikut ini menyatakan operasi yang sama seperti contoh program ST di atas.




Dijalankan berulang

Seperti dalam kasus LD, instruksi dalam ST dijalankan berulang dengan kembali ke instruksi pertama setelah mencapai instruksi terakhir.

# 2.5 Ringkasan

Isi dari bab ini adalah:

- Program ST dasar
- Format pernyataan pemilihan
- Notasi numerik
- Urutan menjalankan program
- Komentar

Poin penting untuk dipertimbangkan:

| | |
|---|---|
| Program ST dasar | • Pernyataan adalah elemen minimum program ST.<br>• Setiap pernyataan diakhiri dengan titik koma (;).<br>• Program ditulis dengan mengombinasikan pernyataan. |
| Format pernyataan pemilihan | • Operator pemilihan (:=) digunakan untuk pemilihan. |
| Notasi numerik | • Tipe notasi numerik dalam ST<br>• "1" dan "0" digunakan untuk nilai bit dalam ST, bukan notasi "ON" dan "OFF".<br>• Nilai bit dapat pula dinyatakan sebagai "TRUE" dan "FALSE" dalam ST. |
| Urutan menjalankan program | • Program yang dibuat dalam ST dijalankan dengan urutan dari atas ke bawah.<br>• Seperti program LD, program ST berproses berulang, kembali ke awal program setelah mencapai akhir program. |
| Komentar | • Menambahkan komentar ke program membuat operasi lebih mudah dipahami.<br>• Komentar dimasukkan di antara dua bintang (* *). |

# Bab 3 Membuat program kontrol I/O

Bab ini menjelaskan cara membuat program kontrol I/O dalam ST.

# 3.1 Program kontrol I/O

Yang berikut ini adalah contoh program untuk kontrol I/O suatu pengontrol terprogram.

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

- OUT: Instruksi output
- X0: Kondisi untuk menjalankan (argumen)
- Y10: Perangkat output (argumen)

"OUT" adalah instruksi output. Argumen menentukan kondisi untuk dijalankan dan perangkat tujuan output.
Bila kondisi untuk dijalankan X0 dipenuhi, perangkat Y10 menyala.

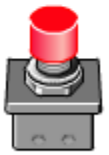
Klik input switch yang ditunjukkan di bawah ini. Input switch X0 menyala.

- Bila input switch X0 menyala, output lampu Y10 menyala.
- Bila input switch X0 mati, output lampu Y10 mati.

Contoh program kontrol I/O yang ditulis dalam ST

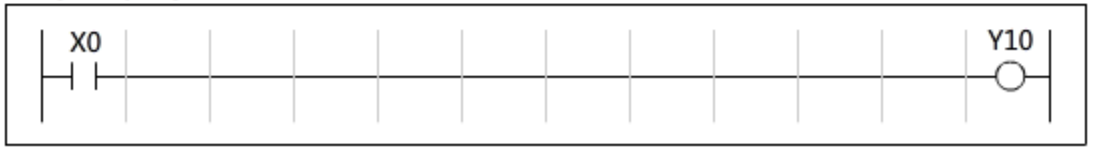
```
OUT(X0,Y10);
```

Switch input X0



Lampu output Y10

Program yang sama ditulis dalam LD



# 3.1 Program kontrol I/O

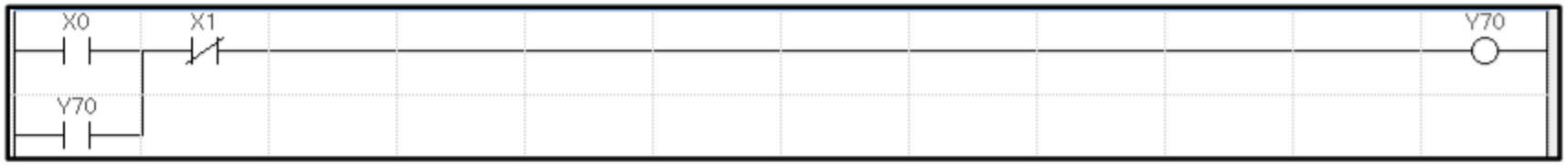
Mirip dengan LD, ada banyak instruksi yang tersedia selain dari instruksi OUT seperti instruksi kontrol I/O dan instruksi pemrosesan data.
Baca manual pemrograman dari pengontrol terprogram Anda untuk informasi selengkapnya tentang instruksi yang tersedia di ST.

Perhatikan bahwa menulis "OUT(X0, Y10);" sebagai "Y10 := X0;" menghasilkan operasi yang sama.

```
Y10 := X0;  (* Operasi yang sama "OUT(X0, Y10);" *)
```

# 3.2 Mengombinasikan beberapa kondisi

Program ladder berikut ini mewakili sebuah sirkuit yang memelihara diri sendiri.




Program yang sama dapat ditulis dalam ST sebagai berikut.

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

Operator logika

Seperti ditunjukkan di atas, operator logika digunakan untuk mengombinasikan beberapa kondisi di ST.

Tabel berikut ini mencantumkan operator logika.

| Operator | Makna |
|---|---|
| OR | Logika OR |
| AND | Logika AND |
| NOT | Logika negasi |
| XOR | Exclusive OR |

# 3.3 Menentukan makna pada variabel

Menggunakan ST dengan pengontrol terprogram MELSEC, perangkat maupun label dapat ditetapkan sebagai alias pada variabel. Pengguna dapat menggunakan label sesuai dengan aplikasi.

Bila suatu label yang terkait dengan aplikasi ditetapkan, operasi lebih mudah dipahami.

```
Y10 := (X0 OR X1) AND X2;     (* Ditulis menggunakan nama perangkat *)
```

```
Lamp := (Switch0 OR Switch1) AND Switch2;     (* Ditulis menggunakan label *)
```

Label dapat dinamai menggunakan perangkat lunak teknik MELSOFT.

Contoh program berikutnya dalam kursus ini diuraikan menggunakan label.

# 3.4 Ringkasan

Isi dari bab ini adalah:

Contoh program kontrol I/O

- Operator logika digunakan untuk mengombinasikan beberapa kondisi dalam ST.
- Nama perangkat dan label dapat digunakan sebagai nama variabel.

Poin penting untuk dipertimbangkan:

| | |
|---|---|
| Mengombinasikan beberapa kondisi | • Operator logika digunakan untuk mengombinasikan kondisi di ST. |
| Menentukan makna pada variabel | • Bila suatu label yang terkait dengan aplikasi ditetapkan, operasi lebih mudah dipahami. |

# Bab 4 Operasi aritmetika

Bab ini menguraikan cara membuat program operasi aritmetika.

- Menguraikan operasi aritmetika
- Menentukan tipe data yang terkait dengan rentang numerik
- Menamai variabel guna menghindari inkonsistensi tipe data

# 4.1 Operasi aritmetika dasar

Contoh program ini menjumlahkan volume produksi dua lini produksi terpisah.
Sisi kanan persamaan adalah operasi aritmetika yang mengandung variabel dan operator aritmetika.

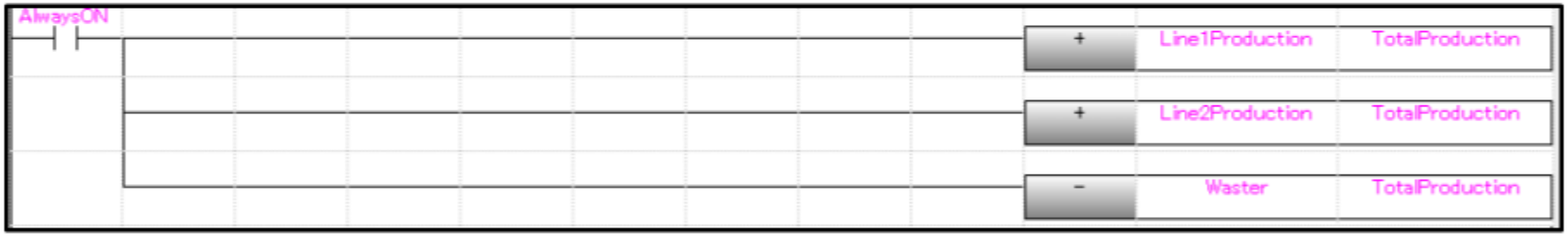
Contoh program artimetika yang ditulis dalam ST

Operator penambahan

Operator pengurangan

```
TotalProduction := Line1Production + Line2Production - Waster;
(* Volume produksi total dari dua lini produksi, kurangi jumlah produk cacat dari total, dan tetapkan nilai yang didapatkan. *)
```

Program yang sama ditulis dalam LD seperti ditunjukkan di bawah ini.




Seperti yang ditunjukkan di atas, program harus ditulis dengan menggunakan 3 jalur dalam Ladder, tapi dengan ST, bisa ditulis dalam 1 jalur.

Tabel berikut ini mencantumkan operator aritmetika dasar.

| Operator | Makna |
|---|---|
| + | Penambahan |
| - | Pengurangan |
| * | Perkalian |
| / | Pembagian |

# 4.2 Tipe data variabel

Tipe data harus ditentukan untuk masing-masing variabel guna menentukan rentang nilai yang akan ditangani.
Tipe data untuk nilai numerik yang digunakan dalam ST adalah tipe bit, integer, dan bilangan asli.

Di antara tipe data yang digunakan di ST, tabel di bawah ini menunjukkan tipe data yang digunakan dalam kursus ini.

| Tipe data | | Rentang data |
|---|---|---|
| Bit | | Status ON/OFF perangkat bit dan status true/false hasil eksekusi |
| Integer | Word (unsigned) | 0 - 65.535 |
| | Word (signed) | -32.768 - 32.767 |
| | Double-word (unsigned) | 0 - 4.294.967.295 |
| | Double-word (signed) | -2.147.483.648 - 2.147.483.647 |

Ketika menggunakan tipe integer, pilih tipe word atau double-word sesuai dengan rentang data, dan pilih tipe signed atau unsigned sesuai dengan keperluannya untuk menangani nilai negatif.
Tentukan tipe data variabel bila nama label ditetapkan menggunakan perangkat lunak teknik MELSOFT.

# 4.3 Nama variabel yang menyatakan tipe data

Menggunakan tipe data yang berbeda di sisi kiri dan kanan suatu persamaan pemilihan dapat menyebabkan kesalahan kompilasi atau hasil yang tidak diharapkan.
Di bawah ini adalah contoh kasus seperti itu.

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

Double-word integer tidak dapat ditetapkan pada word integer. Namun, dalam kasus ini, tipe data tidak dapat dikenali.

Awalan yang menyatakan tipe data bisa ditambahkan ke nama variabel agar tipe data dapat diidentifikasi secara visual.
Tipe penamaan variabel seperti ini dikenal sebagai notasi Hungaria.

| Tipe data | | Rentang data | Awalan | Ekspansi awalan |
|---|---|---|---|---|
| Bit | | Status ON/OFF perangkat bit dan status true/false hasil eksekusi | b | **B**it |
| Integer | Word (unsigned) | 0 - 65.535 | u | **u**nsigned word (kata tidak ditandatangani) |
| | Word (signed) | -32.768 - 32.767 | w | signed **w**ord (kata ditandatangani) |
| | Double-word (unsigned) | 0 - 4.294.967.295 | ud | **u**nsigned **d**ouble-word (kata ganda tidak ditandatangani) |
| | Double-word (signed) | -2.147.483.648 - 2.147.483.647 | d | signed **d**ouble-word (kata ganda ditandatangani) |

Contoh program di bagian atas halaman ini dapat ditulis sebagai berikut menggunakan notasi Hungaria:

```
wValueA := dValueB;  (* Variabel Double-word tidak dapat ditetapkan ke variabel word. *)
```

Dengan menggunakan notasi Hungaria, inkonsistensi tipe data dapat diidentifikasi dalam proses menulis program.

Di bagian selanjutnya kursus ini, nama variabel dalam contoh program ditulis dalam notasi Hungaria.

# 4.4 Ringkasan

Isi dari bab ini adalah:

- Menguraikan operasi aritmetika
- Menentukan tipe data yang terkait dengan rentang numerik
- Menambahkan nama variabel yang menyatakan tipe data

Poin penting untuk dipertimbangkan:

| | |
|---|---|
| Operasi aritmetika dasar | • Operator yang biasa digunakan dalam bahasa pemrograman umum dapat digunakan dalam ST untuk mengekspresikan perhitungan. |
| Tipe data variabel | • Tipe data harus ditentukan untuk masing-masing variabel guna menentukan rentang nilai yang akan ditangani. |
| Menambahkan nama variabel yang menyatakan tipe data | • Menguraikan nama variabel menggunakan notasi Hungaria memungkinkan identifikasi inkonsistensi dalam tipe data variabel saat menulis program. |

# Bab 5 Percabangan kondisional

Program kontrol juga mengandung bagian dari kode di mana pemrosesan aktual berubah sesuai dengan kondisi yang ditetapkan. Bab ini menguraikan tentang percabangan kondisional.

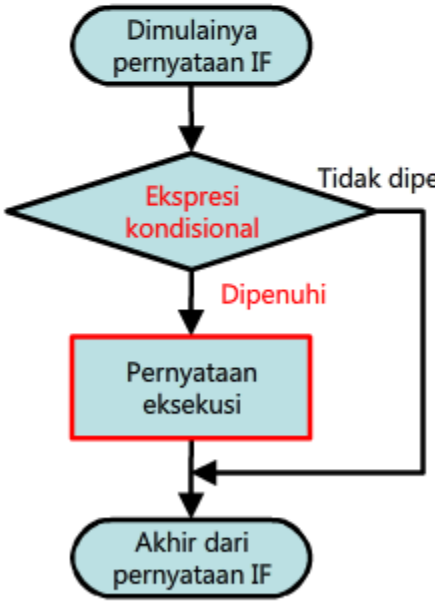
# 5.1 Percabangan kondisional (IF)

Pernyataan IF digunakan untuk percabangan kondisional. Pernyataan IF diuraikan sebagai berikut.

```
IF conditional expression THEN
    Execution statement;          (* Pernyataan dijalankan jika ekspresi kondisional dipenuhi. *)
END_IF;                           (* END_IF; harus ditempatkan di bagian akhir pernyataan IF. *)
```

Dalam contoh program ini, pernyataan dijalankan jika ekspresi kondisional dipenuhi. Pernyataan tidak dijalankan jika ekspresi kondisional tidak dipenuhi.

Gambar berikut ini mengilustrasikan alur operasi dalam contoh program ini.

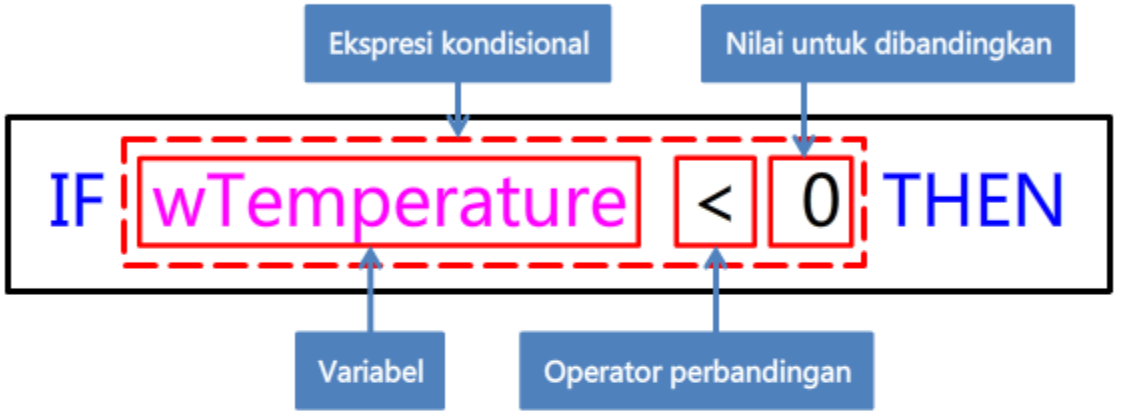



Contoh berikut ini mengilustrasikan percabangan program dengan membandingkan nilai variabel.
Dalam contoh program, pemanas menyala bila suhu dalam panel kontrol turun menjadi di bawah 0 derajat.

```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Pemanas menyala bila suhu di panel kontrol turun menjadi di bawah 0 derajat. *)
END_IF;
```

# 5.1.1 Menulis ekspresi kondisional

Halaman sebelumnya menjelaskan ekspresi kondisional "wTemperature < 0", yang artinya "bila nilai variabel wTemperature lebih kecil dari 0".
Seperti ekspresi ini, ekspresi kondisional menggunakan operator perbandingan untuk menyatakan hubungan antara variabel dan nilai untuk dibandingkan.




Di sisi kiri dan kanan operator perbandingan, nilai ditulis sebagai variabel atau konstanta untuk perbandingan.

Selain untuk membandingkan variabel dan konstanta, ekspresi kondisional dapat ditulis untuk membandingkan variabel, dan menjalankan operasi logika hasil perbandingan atau variabel tipe bit.

Membandingkan variabel

- uValue1 <= uValue2

Operasi logika untuk dua hasil perbandingan

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

Operasi logika untuk dua variabel tipe bit

- bSwitch0 OR bSwitch1

Tabel berikut ini mencantumkan tipe operator perbandingan.

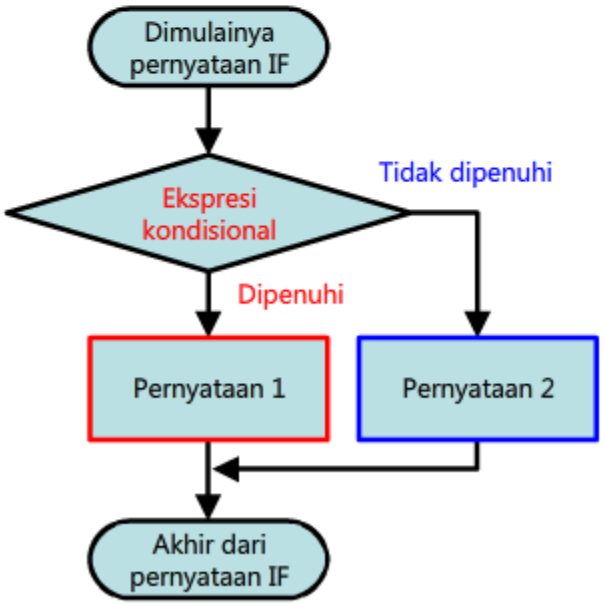
| Operator | Makna |
|---|---|
| > | Lebih besar dari |
| < | Lebih kecil dari |
| >= | Lebih besar dari atau sama dengan |
| <= | Lebih kecil dari atau sama dengan |
| = | Sama dengan |
| <> | Tidak sama dengan |

# 5.1.2 Percabangan pengecualian untuk pernyataan IF (ELSE)

Pernyataan IF sederhana (lihat 5.1) digunakan untuk menjalankan pernyataan bila ekspresi kondisional dipenuhi. Untuk menjalankan pernyataan berbeda bila ekspresi kondisional tidak dipenuhi, statement ELSE digunakan.

```
IF conditional expression THEN
  Execution statement 1;          (* Pernyataan 1 dijalankan jika ekspresi kondisional dipenuhi. *)
  ELSE
  Execution statement 2;          (* Pernyataan 2 dijalankan jika ekspresi kondisional tidak dipenuhi *)
END_IF;
```

Gambar berikut ini mengilustrasikan aliran operasi bila menggunakan pernyataan ELSE.




Contoh program berikut menjalankan berbagai pernyataan bergantung pada apakah kondisi dipenuhi.

Contoh program di 5.1 memiliki kekurangan yakni bahwa pemanas terus menaikkan suhu meski suhu telah mencapai 0 derajat. Namun, program berikut ini mematikan pemanas bila "wTemperature" melebihi 0 derajat.

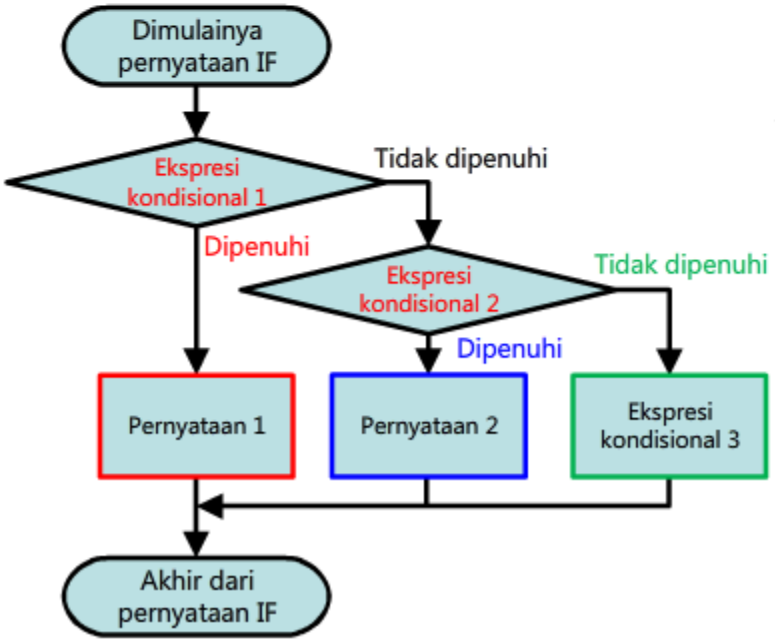
```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Menyalakan pemanas bila suhu turun menjadi di bawah 0 derajat. *)
  ELSE
  bHeater := 0;  (* Mematikan pemanas bila suhu mencapai atau melebihi 0 derajat *)
END_IF;
```

# 5.1.3 Percabangan tambahan untuk pernyataan IF (ELSIF)

Pernyataan ELSE digunakan untuk menjalankan pernyataan berbeda bila ekspresi kondisional tidak dipenuhi. Percabangan tambahan dapat ditambahkan dengan menggunakan pernyataan ELSIF, yang berarti bahwa jika ekspresi kondisional sebelumnya tidak dipenuhi, maka ekspresi kondisional diperiksa.

```
IF Conditional expression 1 THEN
  Execution statement 1;      (* Pernyataan 1 dijalankan jika ekspresi kondisional 1 dipenuhi. *)
  ELSIF Conditional expression 2 THEN
  Execution statement 2;      (* Pernyataan 2 dijalankan jika ekspresi kondisional 1 tidak dipenuhi dan ekspresi kondisional 2 dipenuhi. *)
  ELSE
  Execution statement 3;      (* Pernyataam 3 dijalankan jika ekspresi kondisional 1 dan 2 tidak dipenuhi. *)
END_IF;
```

Gambar berikut ini mengilustrasikan aliran operasi saat menggunakan pernyataan ELSEIF.




Pernyataan ELSIF ditambahkan ke contoh program yang diilustrasikan di 5.1.2 untuk mengatasi kasus ketika suhu melebihi 40 derajat.

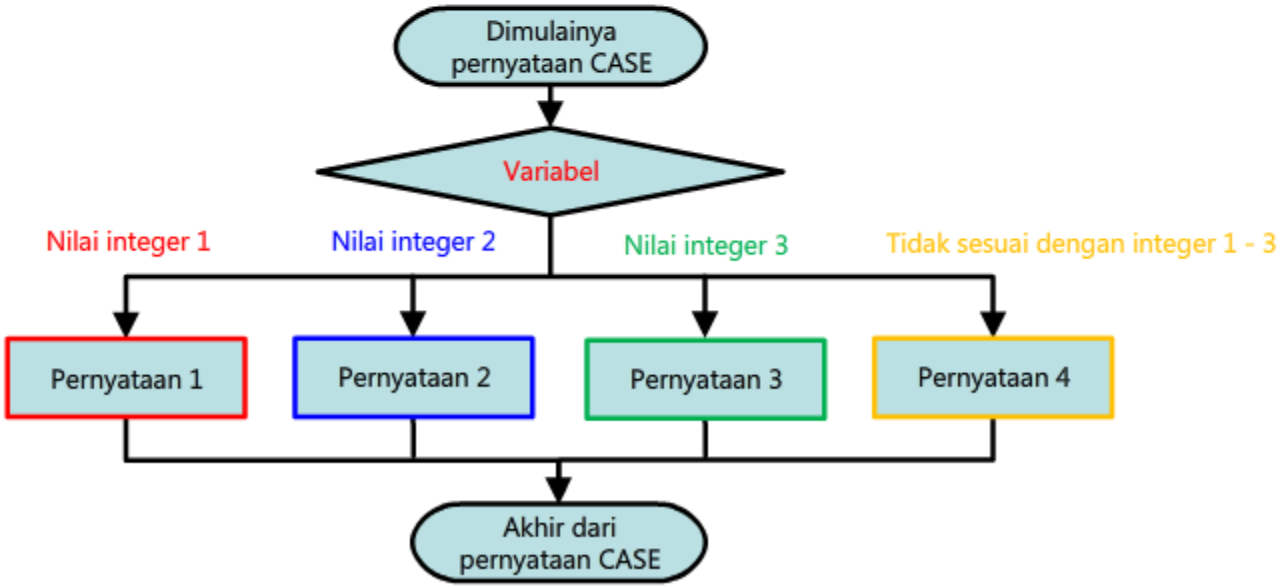
```
IF wTemperature < 0 THEN
   bHeater := 1; (* Menyalakan pemanas ketika suhu turun menjadi di bawah 0 derajat. *)
   bCooler := 0; (* Mematikan pendingin ketika suhu turun menjadi di bawah 0. *)
   ELSIF 40 < wTemperature THEN
   bHeater := 0; (* Mematikan pemanas jika suhu melebihi 40 derajat. *)
   bCooler := 1; (* Menyalakan pendingin jika suhu melebihi 40 derajat. *)
   ELSE
   bHeater := 0; (* Mematikan pemanas jika tidak satu pun kondisi sebelumnya dipenuhi. *)
   bCooler := 0; (* Mematikan pendingin jika tidak satu pun kondisi sebelumnya dipenuhi. *)
END_IF;
```

# 5.2 Percabangan kondisional sesuai dengan nilai integer (CASE)

Pernyataan IF digunakan untuk percabangan bergantung pada apakah ekspresi kondisional dipenuhi atau tidak.
Pernyataan CASE digunakan untuk percabangan sesuai dengan nilai integer.
Gambar berikut ini mengilustrasikan bagaimana pernyataan CASE ditulis.

```
CASE Variable OF
    Integer value 1: Execution statement 1;      (* Pernyataan 1 dijalankan bila variabel sesuai dengan nilai integer 1. *)
    Integer value 2: Execution statement 2;      (* Pernyataan 2 dijalankan bila variabel sesuai dengan nilai integer 2. *)
    Integer value 3: Execution statement 3;      (* Pernyataan 3 dijalankan bila variabel sesuai dengan nilai integer 3. *)
    ELSE Execution statement 4;     (* Pernyataan 4 dijalankan jika variabel tidak sesuai dengan salah satu dari nilai integer. *)
END_CASE;                           (* "END_CASE;" harus ditempatkan di bagian akhir pernyataan CASE. *)
```

Gambar berikut ini mengilustrasikan alur operasi saat menggunakan pernyataan CASE.



# 5.2.1 Contoh program untuk pernyataan CASE

Eksekusi pernyataan CASE dijelaskan menggunakan operasi contoh program.

Klik untuk melanjutkan ke halaman berikutnya.
Untuk menonton animasinya lagi, klik tombol "Putar".

Putar


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

| Berat | uSize | Kelas |
|---|---|---|
| 0 hingga 20 kg | 1 | M |
| 21 hingga 30 kg | 2 | L |
| 31 hingga 40 kg | 3 | XL |
| 41 kg lebih | 4 | Oversize |

# 5.3 Ringkasan

Isi dari bab ini adalah:

- Percabangan kondisional dengan pernyataan IF
- Menulis ekspresi kondisional
- Percabangan kondisional sesuai dengan nilai integer (statement CASE)

Poin penting untuk dipertimbangkan:

| | |
|---|---|
| Pernyataan IF | • Dengan pernyataan IF, program dicabangkan bila ekspresi kondisional dipenuhi.<br>• Pernyataan ELSE digunakan untuk percabangan bila ekspresi kondisional tidak dipenuhi.<br>• Pernyataan ELSIF digunakan untuk menambahkan percabangan lain bila ekspresi kondisional dalam pernyataan IF tidak dipenuhi. |
| Ekspresi kondisional | • Ekspresi kondisional menyatakan hubungan antara variabel dan nilai untuk membandingkan menggunakan operator perbandingan. |
| Pernyataan CASE | • Pernyataan CASE digunakan untuk percabangan sesuai dengan nilai integer. |

# Bab 6 Penyimpanan dan penanganan data

Seperti untuk aplikasi kontrol I/O, pengontrol terprogram sekarang ini digunakan untuk memproses sejumlah besar data sebagai inti sistem produksi.
Untuk memproses data dalam jumlah besar, data harus disimpan lalu dibaca sebagaimana perlu.
Bab ini menjelaskan tentang cara menulis program ringkas untuk menyimpan dan memproses data.

- Array digunakan untuk mengurutkan dan mengatur variabel.
- Struktur data digunakan untuk mengatur variabel terkait.
- Program pemrosesan loop secara efektif memproses array menggunakan FOR.

Program singkat untuk menyimpan dan menangani data dapat dibuat dengan menggunakan array, struktur data, dan pernyataan FOR.

# 6.1 Pengurutan dan penyimpanan data (Array)

Dengan menggunakan array, beberapa nilai dapat ditangani hanya dengan satu variabel.
Dalam contoh berikut ini, data volume produksi di pabrik produksi mobil disimpan menurut negara destinasi.

| Destinasi | Negara A | Negara B | Negara C |
|---|---|---|---|
| Volume produksi | 35 | 75 | 65 |

Data volume produksi menurut negara destinasi ditetapkan ke sebuah variabel. Tanpa menggunakan array, satu variabel harus dibuat untuk masing-masing destinasi.
Namun dengan menggunakan array, volume produksi untuk beberapa destinasi bisa ditetapkan pada dan disimpan dalam satu variabel.

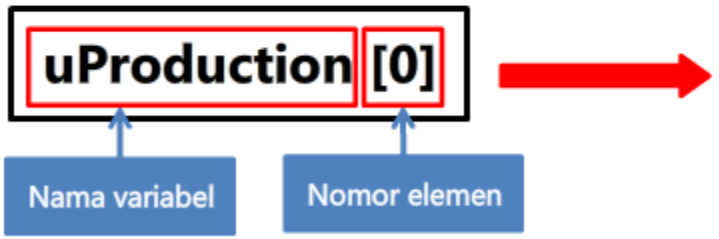
**Tidak menggunakan array**

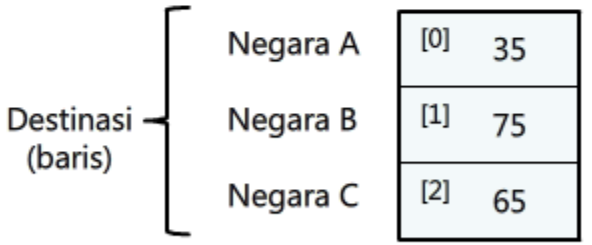
uProductionA
uProductionB
uProductionC

**Menggunakan array**

uProduction

Variabel individual dalam array ditetapkan menggunakan nomor elemen. Nomor elemen dimulai dari nol [0].




Dalam contoh program berikut, variabel volume produksi yang direncanakan untuk Negara A dipilih.

```
uShowProductionPlan := uProduction[0];
(* Menentukan nomor elemen untuk negara A. *)
```

# 6.1.1 Array matriks

Berikutnya, data warna cat digunakan sebagai tambahan pada data destinasi.

| Destinasi | Negara A | | | Negara B | | | Negara C | | |
|---|---|---|---|---|---|---|---|---|---|
| Warna cat |  | | | | | | | | |
| Volume produksi | 10 | 5 | 20 | 15 | 40 | 20 | 25 | 30 | 10 |
| | 35 totalnya | | | 75 totalnya | | | 65 totalnya | | |

Seperti yang diilustrasikan dalam tabel berikut, data dapat dipisah dan disimpan menurut warna cat (kolom) untuk masing-masing negara destinasi (baris).

Warna cat (kolom)

| Destinasi (baris) | Merah | Kuning | Biru |
|---|---|---|---|
| Negara A | [0,0] 10 | [0,1] 5 | [0,2] 20 |
| Negara B | [1,0] 15 | [1,1] 40 | [1,2] 20 |
| Negara C | [2,0] 25 | [2,1] 30 | [2,2] 10 |

Nomor elemen menyatakan destinasi

Nomor elemen menyatakan warna cat

Array yang mengatur data ke dalam baris dan kolom dengan cara ini dikenal sebagai array matriks.
Nomor elemen yang menyatakan baris dan kolom dipisahkan oleh koma.

**Variabel array (array matriks)**

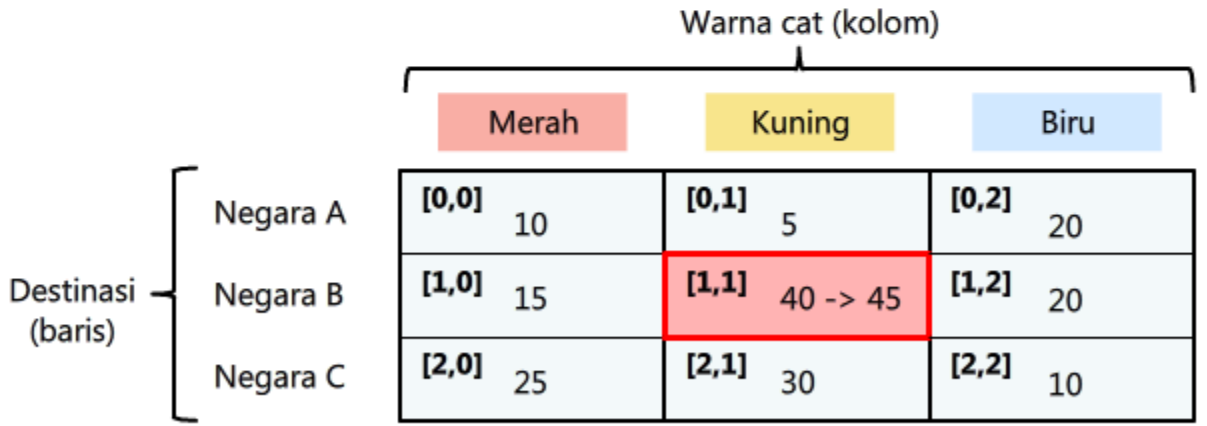
uProduction[1,1]

# 6.1.2 Pemilihan array matriks

Menggunakan array matriks, contoh program berikut ini menetapkan jumlah mobil yang harus diproduksi mendesak selain dari rencana volume produksi mobil kuning untuk Negara B.

```
uAdditionalProduction := 5;
uProduction[1,1] := uProduction[1,1] + uAdditionalProduction;
(* Menambahkan jumlah produksi tambahan (5 unit) ke rencana volume produksi awal. *)
```

| Destinasi | Negara A | Negara B (Tambahan 5 mobil) | Negara C |
|---|---|---|---|
| Warna cat |  |  |  |
| Volume produksi | 10 \| 5 \| 20 | 15 \| 40 \| 20 | 25 \| 30 \| 10 |
| | 35 totalnya | 75 totalnya | 65 totalnya |



# 6.1.3 Memproses informasi yang disimpan dalam array matriks

Menggunakan array matriks, contoh program berikut ini menghitung volume produksi total yang direncanakan untuk semua warna cat untuk Negara C dan menetapkan nilai tersebut ke suatu variabel.

```
uProductionToday := uProduction[2,0] + uProduction[2,1] +uProduction[2,2];
(* Menghitung volume produksi total yang direncanakan untuk hari ini untuk semua warna cat untuk
Negara C dan memilih nilai tersebut ke "uProductionToday". *)
```

| Destinasi | Negara A | | | Negara B | | | Negara C | | |
|---|---|---|---|---|---|---|---|---|---|
| Warna cat |  | | | | | | | | |
| Volume produksi | 10 | 5 | 20 | 15 | 45 | 20 | 25 | 30 | 10 |
| | 35 totalnya | | | 80 totalnya | | | 65 totalnya | | |

Warna cat (kolom)

| Destinasi (baris) | Merah | Kuning | Biru |
|---|---|---|---|
| Negara A | [0,0] 10 | [0,1] 5 | [0,2] 20 |
| Negara B | [1,0] 15 | [1,1] 45 | [1,2] 20 |
| Negara C | [2,0] 25 | [2,1] 30 | [2,2] 10 |

# 6.2 Looping (FOR)

Contoh program di halaman sebelumnya (rencana volume produksi untuk hari ini dipilih) ditunjukkan lagi di bawah ini.

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

Dengan contoh program ini, bila jumlah warna cat meningkat, akan ditambahkan lebih banyak variabel.
Maka, ekspresi menjadi semakin panjang, membuatnya lebih sulit dibaca.

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

Dalam kasus ini, pernyataan loop dapat digunakan untuk membuat kode yang lebih bersih.

Pernyataan loop mencakup pernyataan FOR, WHILE, dan REPEAT. kursus ini membahas pernyataan FOR.

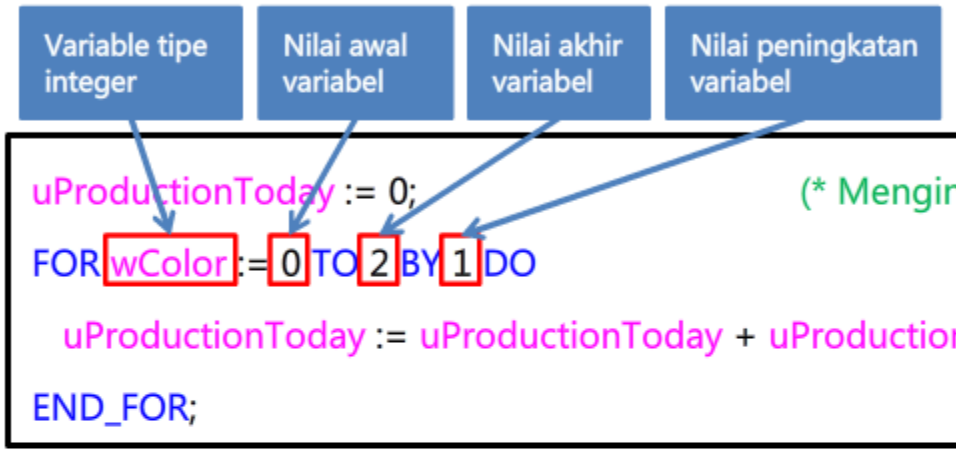
Pernyataan FOR dijelaskan sebagai berikut.

```
FOR variable := initial value TO final value BY increments DO
  Execution statement; (* Pernyataan dijalankan dalam loop hingga variabel mencapai nilai akhir. *)
END_FOR;               (* END_FOR; harus ditempatkan di bagian akhir pernyataan FOR. *)
```

Pernyataan diulang hingga nilai akhir variabel dicapai dan kode "END_FOR;" dijalankan.

# 6.2 Looping (FOR)

Menggunakan pernyataan FOR, contoh program berikut ini mendapatkan volume produksi yang direncanakan untuk semua warna cat untuk Negara C.




```
uProductionToday := 0;                          (* Menginisialisasi variabel. *)
FOR wColor := 0 TO 2 BY 1 DO
   uProductionToday := uProductionToday + uProduction[2,wColor]; (* Menambahkan volume produksi yang direncanakan. *)
END_FOR;
```

Menggunakan pernyataan FOR, variabel "wColor" meningkat satu mulai dari nilai awal nol dan pernyataan diulang hingga variabel mencapai nilai akhir dua.
Variabel "wColor" ditentukan sebagai nomor elemen kedua dalam array "uProduction" yang diuraikan dalam pernyataan dijalankan.
Nilai variabel "wColor" meningkat setiap kali pernyataan diulang. Rencana volume produksi untuk masing-masing warna cat ditambahkan setiap kali untuk mendapatkan total.

Contoh program ini dijalankan dalam loop tiga kali. (Pertama kali: merah [0] => Kedua kali: kuning [1] => Ketiga kali: biru [2])

Operasi program ini diilustrasikan di halaman berikutnya.

# 6.2 Looping (FOR)

Eksekusi pernyataan FOR diuraikan menggunakan operasi contoh program.

**Array perkiraan volume produksi**

| | Merah | Kuning | Biru |
|---|---|---|---|
| Negara A | [0,0] 10 | [0,1] 5 | [0,2] 20 |
| Negara B | [1,0] 15 | [1,1] 45 | [1,2] 20 |
| Negara C | [2,0] 25 | [2,1] 30 | [2,2] 10 |

Klik untuk melanjutkan ke halaman berikutnya.
Untuk menonton animasinya lagi, klik tombol "Putar".

Putar

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 Menyimpan data terkait (Struktur)

Struktur memungkinkan satu nama variabel untuk menyatakan beberapa variabel terkait.
Dalam contoh berikut, status lini produksi mobil ditampilkan di Andon (papan tampilan).

Tabel berikut ini mencantumkan nama variabel, nilai, dan tipe data yang terkait dengan item yang ditampilkan.

| Item | Nama variabel | Nilai | Tipe data Variabel |
|---|---|---|---|
| Model | sModel | 'ST TRUCK' | String teks |
| Status | bStatus | 'dalam produksi' | Tipe bit |
| Volume target produksi untuk hari ini | uPlanQty | '100' | Tipe word integer (unsigned) |
| Volume produksi saat ini | uActualQty | '88' | Tipe word integer (unsigned) |


Jika struktur tidak digunakan, nama variabel harus diganti untuk setiap lini bila ada beberapa lini produksi.
Berikut ini ditunjukkan contoh nama variabel menurut lini produksi.

**Lini produksi pertama**

s1stLineModel
b1stLineStatus
u1stLinePlanQty
u1stLineActualQty

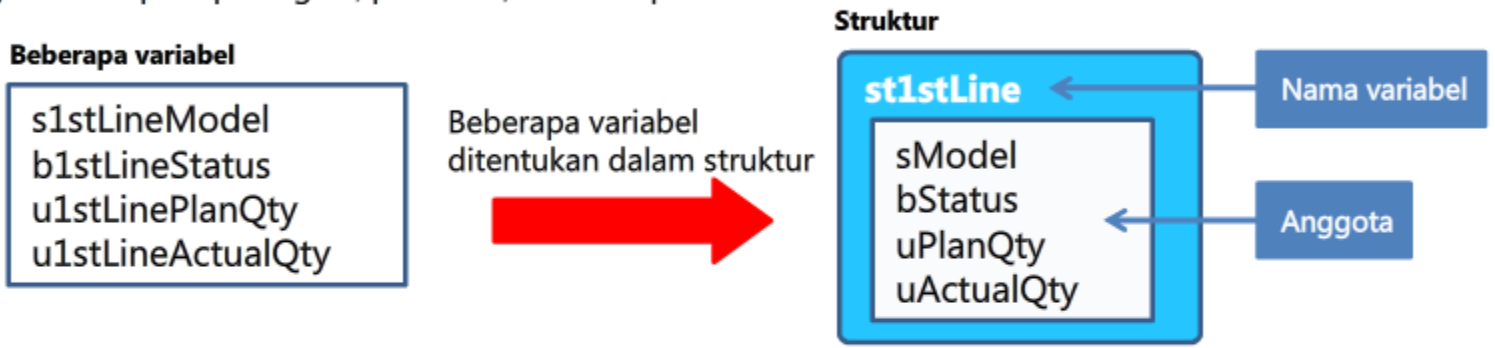
**Lini produksi kedua**

s2ndLineModel
b2ndLineStatus
u2ndLinePlanQty
u2ndLineActualQty

Bila jumlah lini produksi meningkat, jumlah variabel untuk ditangani juga akan meningkat. Maka, program menjadi lebih panjang dan lebih sulit dibaca.

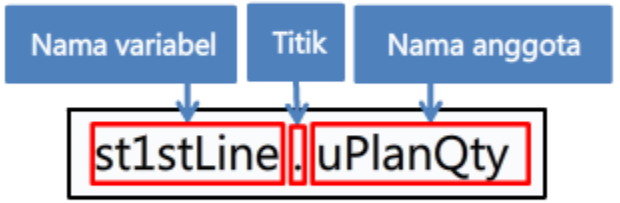
# 6.3 Menyimpan data terkait (Struktur)

Menggunakan struktur memungkinkan satu nama variabel untuk menyatakan beberapa variabel yang terkait dengan satu lini produksi.
Seperti ini, struktur digunakan untuk mengatur, menyimpan, dan menangani data dalam batch untuk kondisi dan spesifikasi objek fisik seperti perangkat, peralatan, dan workpiece.




**St**ructure variable (variabel struktur) mengandung awalan "st" untuk menyatakan bahwa ini adalah struktur.
Variabel individual yang ditentukan oleh struktur dikenal sebagai anggota. Tipe data masing- masing anggota dapat berbeda.

Masing-masing anggota array struktur dapat ditentukan setelah nomor elemen array menggunakan titik sebelum nama anggota.

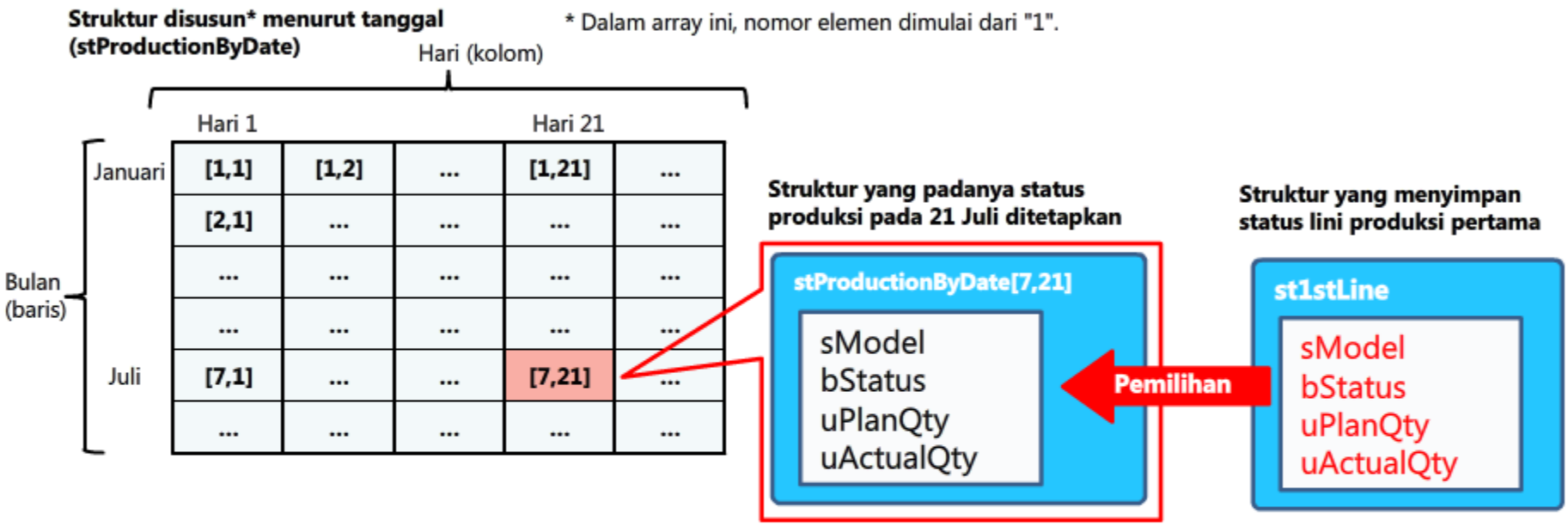



Dalam contoh program, konstanta ditetapkan ke anggota variabel struktur untuk lini produksi pertama.

```
st1stLine.uPlanQty := 150;
 (* Menetapkan target produksi hari ini untuk lini produksi pertama menjadi 150. *)
```

# 6.3.1 Menyimpan array struktur

Struktur dapat dibuat sebagai array.
Dalam contoh berikut ini, status produksi disimpan menurut tanggal.

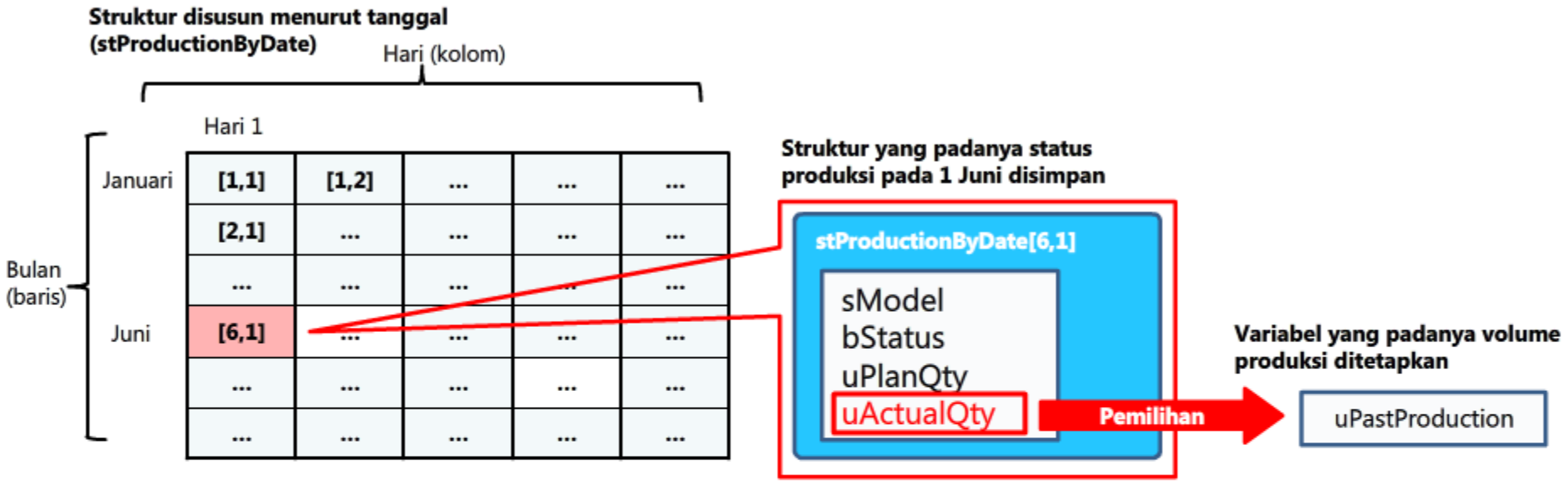



```
stProductionByDate[7,21] := st1stLine;
(* Status produksi pada 21 Juli disimpan dalam struktur yang disusun
menurut tanggal (stProductionByDate). *)
```

Seperti ini, anggota tidak perlu ditetapkan secara individual untuk struktur pemilihan.

# 6.3.2 Membaca array struktur

Dalam contoh berikut ini, volume produksi dibaca dari struktur yang disusun menurut tanggal lalu ditetapkan ke suatu variabel.




```
uPastProduction := stProductionByDate[6,1].uActualQty;
(* Memilih volume produksi pada 1 Juni pada variabel uPastProduction. *)
```

Masing-masing anggota array struktur dapat ditentukan dengan menambahkan sebuah titik (.) dan nama anggota ke nomor elemen array tersebut.

# 6.4 Ringkasan

Isi dari bab ini adalah:

- Gambaran umum dan penggunaan array
- Pemrosesan loop menggunakan pernyataan FOR
- Gambaran umum dan penggunaan struktur

Poin penting untuk dipertimbangkan:

| | |
|---|---|
| Array | • Beberapa nilai dapat ditangani oleh satu variabel dengan menggunakan array.<br>• Variabel individual dalam array ditentukan oleh nomor elemen yang ditambahkan ke akhir nama variabel. |
| Pernyataan FOR | • Pernyataan loop digunakan dalam program bila operasi berulang diinginkan.<br>• Pernyataan FOR digunakan untuk mengulangi operasi hingga kondisi untuk akhir operasi loop dipenuhi. Pernyataan sebelum pernyataan "END_FOR;" dijalankan secara berulang. |
| Struktur | • Struktur memungkinkan satu nama variabel untuk menyatakan beberapa variabel terkait. Struktur dapat mencakup variabel dari tipe data berbeda.<br>• Variabel individual, atau anggota, yang ditentukan dalam struktur ditentukan dengan menambahkan sebuah titik dan nama anggota setelah nama variabel struktur. |

# Bab 7 Penanganan data string

Dalam beberapa kasus, pengontrol terprogram menggunakan data string untuk mengirim perintah ke atau menerima umpan balik dari perangkat terhubung seperti pembaca barcode, pengontrol suhu, atau timbangan elektronik. Untuk tujuan tersebut, perlu untuk bergabung atau mengekstrak data string sebagaimana diperlukan.

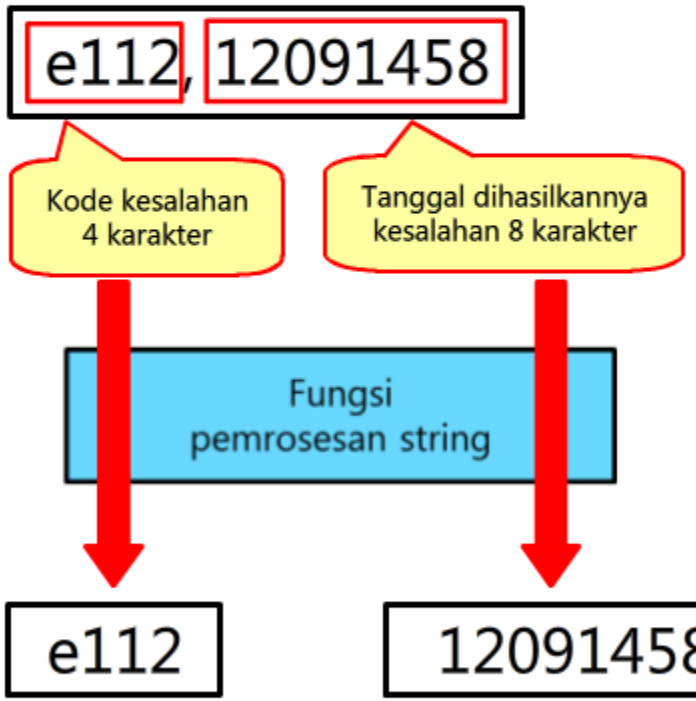
Bab ini menjelaskan cara menangani data string.

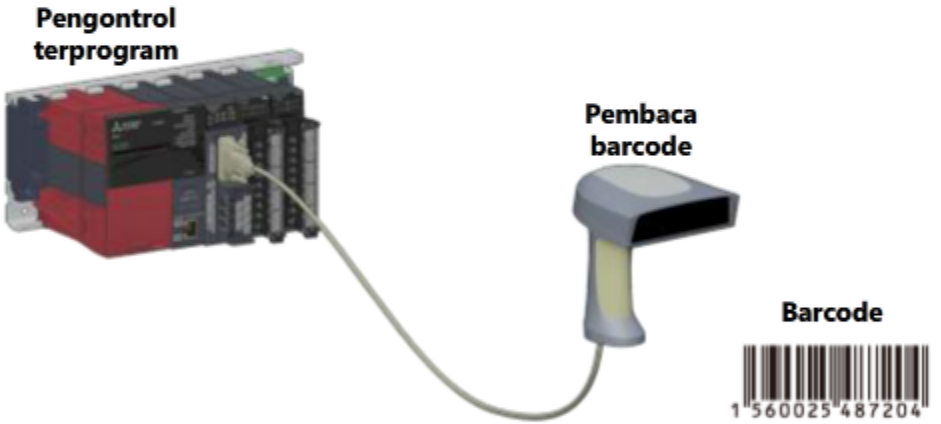
# 7.1 Contoh penanganan data string

Sebagai contoh pemrosesan string, contoh ini mengilustrasikan skenario di mana data dibaca dari pembaca barcode. Fungsi (sebuah tipe instruksi) digunakan untuk memproses string.

Sebagaimana diilustrasikan di bawah ini, string yang dibaca oleh barcode mengandung kode kesalahan dengan panjang tetap 4 karakter dan bulan dengan panjang tetap 8 karakter, tanggal, waktu, dan data menit.
Contoh program pemrosesan string akan dijelaskan menggunakan sistem ini.

**Contoh data string yang dibaca dari pembaca barcode**




Kode kesalahan diekstrak.
7.3 Mengekstrak string (LEFT)

Tanggal dan waktu terjadinya kesalahan (14:58, 9 Desember) diekstrak.
7.4 Mengekstrak string (MID)

# 7.2 Pemilihan string

Sebelum menjelaskan tentang cara mengekstrak string, bagian ini menjelaskan tipe data untuk string.

Tipe data untuk string yang dapat digunakan dengan pengontrol terprogram tercantum di tabel berikut ini.

| Tipe data | Tipe karakter dapat diproses | Awalan notasi Hungaria | Ekspansi awalan |
|---|---|---|---|
| String | String karakter alfanumerik dan nomor (ASCII) atau Jepang (Shift-JIS) | s | **s**tring (string) |
| String [Unicode] | String dari berbagai bahasa dan simbol | ws | **w**ide **s**tring (string lebar) |

Tipe string yang digunakan bergantung pada perangkat yang terhubung ke pengontrol terprogram atau bahasa terkait.
Bab ini menjelaskan tentang berbagai tipe string teks.

Bila tipe string ditetapkan ke sebuah variabel string, masukkan string tersebut dalam tanda kutip tunggal (').

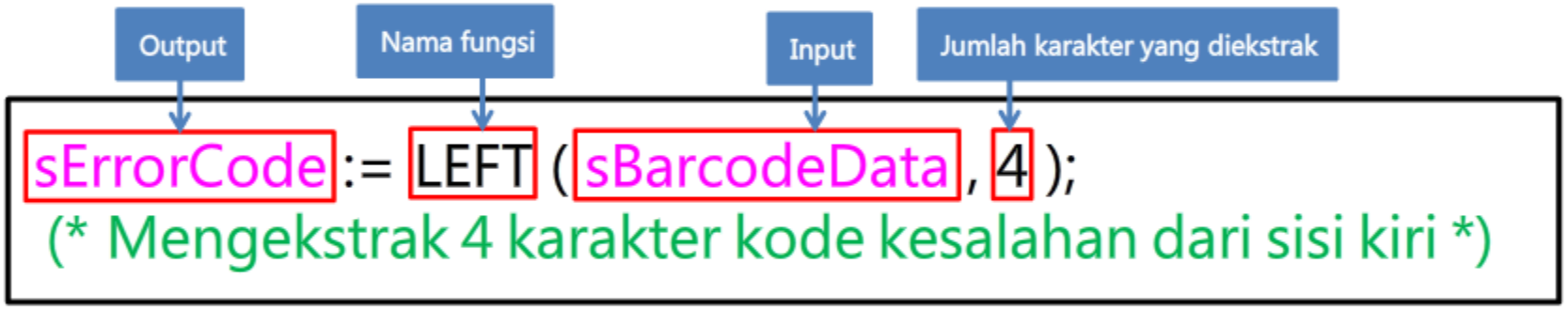
```
sDefault := 'e112,12091458';  (* String pemilihan *)
```

# 7.3 Mengekstrak string (LEFT)

Kode kesalahan "e112" diekstrak dari variabel string "sBarcodeData" yang mengandung string "e112,12091458".

| Nama variabel | String yang disimpan |
|---|---|
| sBarcodeData | e112, 12091458 |

Fungsi LEFT mengekstrak hanya sejumlah karakter yang ditetapkan dari sisi kiri string input.
Yang berikut ini mengilustrasikan contoh program.



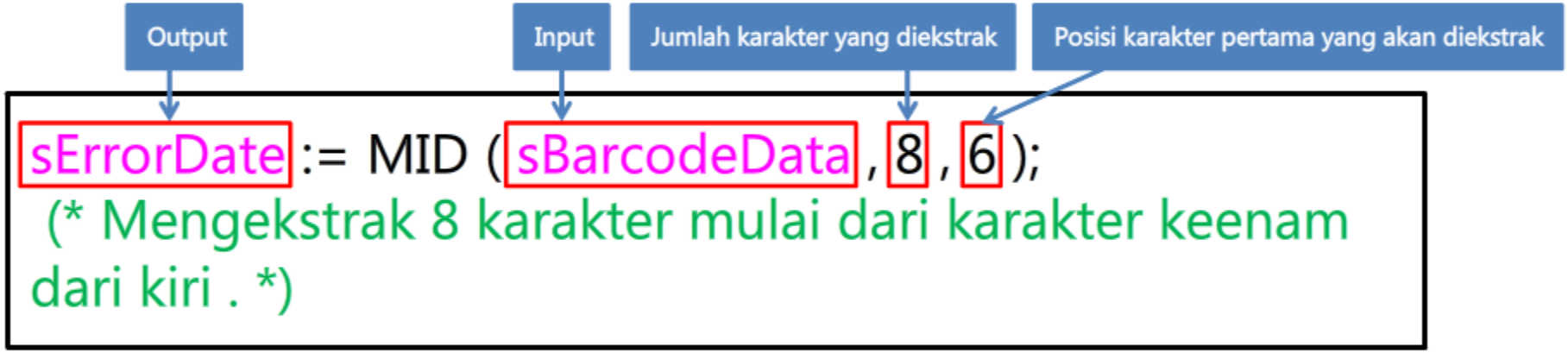

Empat karakter diekstrak dari kiri. Nilai "e112", yang merupakan string yang mewakili kode kesalahan, ditetapkan ke sisi kiri.

# 7.4 Mengekstrak string (MID)

Waktu dihasilkannya kesalahan "12091458" diekstrak dari variabel string "sBarcodeData" yang mengandung string "e112,12091458".

| Nama variabel | String yang disimpan |
|---|---|
| sBarcodeData | e112,12091458 |

Fungsi MID mengekstrak jumlah karakter yang ditetapkan yang dimulai di posisi mulai yang ditetapkan dalam input string. Yang berikut ini mengilustrasikan contoh program.




Dalam contoh ini, string 8 karakter diekstrak mulai dari karakter keenam. Nilai "12091458", yang merupakan string yang menyatakan waktu terjadinya kesalahan, ditetapkan ke sisi kiri.

# 7.5 Ringkasan

Isi dari bab ini adalah:

- Metode pemilihan ke variabel string
- Fungsi yang mengekstrak string (LEFT dan MID)

Poin penting untuk dipertimbangkan:

| | |
|---|---|
| Pemilihan string | • Untuk memilih string ke variabel string, masukkan string dalam tanda kutip tunggal (').<br>• Gunakan tipe string atau tipe [Unicode] string ke perangkat yang dihubungkan ke pengontrol terprogram atau ke bahasa terkait. |
| Fungsi untuk menangani string | • Fungsi digunakan untuk menangani string. |

# 7.6 Ringkasan kursus

Kursus ini membahas dasar-dasar cara membuat program di ST.
Ini adalah bagian akhir dari kursus e-learning ini.

Program ST dibuat menggunakan perangkat lunak teknik MELSOFT.
Untuk detail tentang langkah-langkah spesifik seperti memasukkan data, mengedit, menyimpan, mengompilasi program dengan perangkat lunak teknik MELSOFT, baca yang berikut.

- Mitsubishi FA e-Learning Course "MELSOFT GX Works3 (Structured Text)" (MELSOFT GX Works3 (Teks Terstruktur)) (akan segera dirilis)
- Panduan pengoperasian perangkat lunak teknik MELSOFT

Untuk informasi lebih lanjut tentang ST, baca yang berikut ini.

- Buku panduan pemrograman pengontrol terprogram

Untuk informasi tentang instruksi dan fungsi untuk aplikasi Anda, baca yang berikut ini.

- Panduan pemrograman pengontrol terprogram

# Tes Akhir

Setelah menyelesaikan semua pelajaran kursus Dasar-Dasar Pemrograman (Teks Terstruktur) , Anda sudah siap untuk mengikuti tes akhir. Jika Anda masih kurang memahami salah satu topik yang dibahas, gunakan kesempatan ini untuk mengulas topik tersebut.
Total terdapat 12 pertanyaan (20 item) dalam Tes Akhir ini.
Anda dapat mengikuti tes akhir sesering mungkin.

**Cara menilai tes**
Setelah memilih jawaban, pastikan untuk mengeklik tombol **Jawab**. Jawaban akan hilang jika Anda melanjutkan tanpa mengeklik tombol Jawab. (Dianggap sebagai pertanyaan belum dijawab.)

**Hasil penilaian**
Jumlah jawaban yang benar, jumlah pertanyaan, persentase jawaban yang benar, dan hasil lulus/gagal akan ditampilkan pada halaman nilai.

| | |
|---|---|
| Jawaban yang benar: | **5** |
| Jumlah total pertanyaan: | **5** |
| Persentase: | **100%** |

Agar lulus tes, Anda harus menjawab **60%** pertanyaan dengan benar.

Lanjutkan | Tinjau

- Klik tombol **Lanjutkan** untuk keluar dari tes.
- Klik tombol **Tinjau** untuk meninjau tes. (Jawaban yang benar dicentang)
- Klik tombol **Coba lagi** untuk mengikuti kembali tes.

# Tes

## Tes Akhir 1

Karakteristik structured text (ST)
Pilih deskripsi ST yang tidak benar.

- ST mudah dipelajari bagi orang yang sudah berpengalaman menulis program dalam bahasa C atau BASIC.
- Perhitungan seperti penambahan dan pengurangan bisa ditulis seperti ekspresi matematika yang lazim digunakan.
- Simbol untuk kontak dan kumparan digunakan untuk membuat program yang mirip dengan sirkuit elektrik.
- ST cocok untuk penanganan data.

Jawab Kembali

# Tes

## Tes Akhir 2

Prinsip-prinsip dasar ST
Pilih pernyataan yang benar yang ditulis dalam ST.

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

Jawab Kembali

# Tes

## Tes Akhir 3

Komentar yang menjelaskan
Pilih komentar yang benar yang ditulis dalam ST.

- ' Memilih nilai 1 ke variabel.
- (* Memilih nilai 1 ke variabel. *)
- { Memilih nilai 1 ke variabel. }
- <!-- Memilih nilai 1 ke variabel. -->

Jawab Kembali

# Tes

## Tes Akhir 4

Urutan menjalankan program ST
*Nilai awal "uTotalProduction" adalah "100". Nilai variabel "uTotalProduction" akan menjadi "101" setelah contoh program berikut diproses. Pilih status "uTotalProduction" yang benar setelah selang beberapa detik.

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

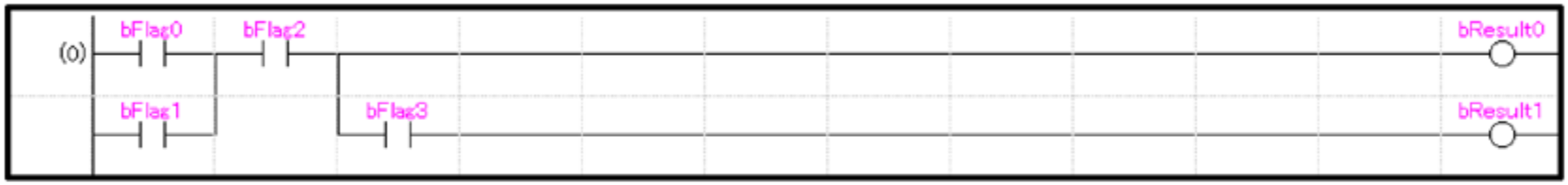
- ○ Nilai tetap di 101.
- ○ Nilai terus berubah.

Jawab | Kembali

# Tes

## Tes Akhir 5

Mengombinasikan beberapa kondisi
Pilih contoh program ST yang benar yang menyatakan operasi yang sama seperti contoh program berikut dalam LD.




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

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

Jawab  Kembali

# Tes

## Tes Akhir 6

Deskripsi pernyataan IF di ST
Operasi berikut ini dijalankan oleh contoh program di bawah ini.
- Jika suhu turun menjadi 5 derajat atau kurang, pemanas menyala dan pendingin mati.
- Jika suhu melebihi 50 derajat, pemanas mati dan pendingin menyala.
- Jika suhu tidak berlaku pada pernyataan di atas, baik pemanas maupun pendingin mati.

*Nama variabel: Suhu (wTemperature), pemanas (bHeater), dan pendingin (bCooler)
Pilih pilihan yang benar untuk tiap bagian yang kosong dari contoh program.

```
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--

Jawab Kembali

# Tes

## Tes Akhir 7

Pernyataan CASE

Pilih yang benar untuk masing-masing deskripsi (Q1 hingga Q5) pernyataan CASE berikut ini.

Pernyataan CASE digunakan untuk percabangan sesuai dengan nilai (Q1).
Dalam contoh program berikut ini, bila nilai (Q2) adalah 25, variabel (Q3) diberi nilai (Q4). Bila nilai variabel (Q2) tidak sama dengan 10, 25, atau 8, variabel (Q3) diberi nilai (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--

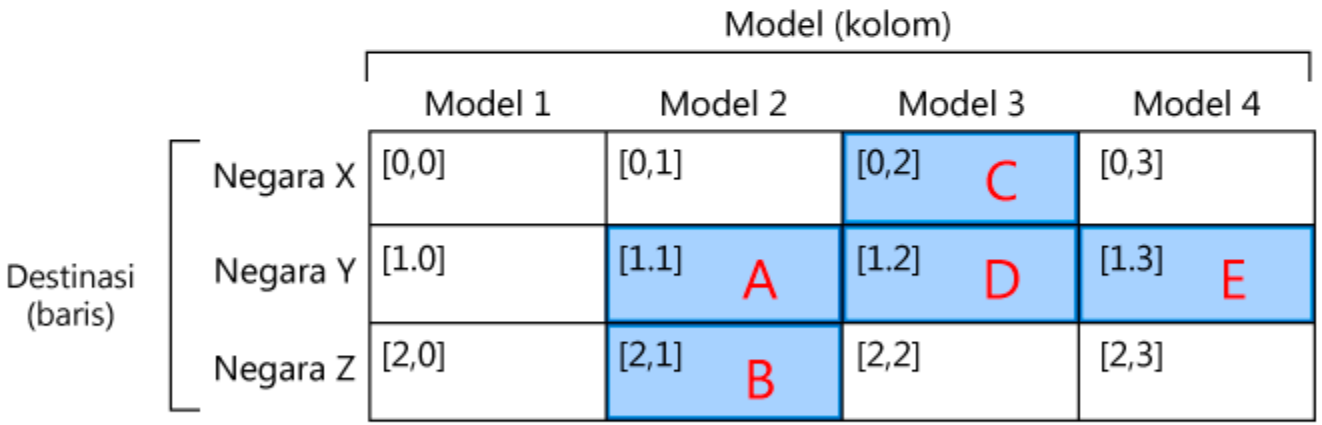
Jawab Kembali

# Tes Akhir 8

Array ST dan statement berulang
Contoh program berikut ini menjumlahkan volume rencana produksi dari semua model yang ditetapkan untuk Negara Y lalu menetapkan nilai ini ke variabel. Pilih bagian dari array yang dibaca setelah pernyataan FOR dijalankan dalam loop 3 kali.

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

**Array digunakan untuk menyimpan perkiraan jumlah unit yang diproduksi per model dan destinasi (uProduction)**




- A
- B
- C
- D

Jawab | Kembali

# Tes Akhir 9

Array ST dan pernyataan berulang
Contoh program berikut ini mendapatkan total volume produksi pada hari yang sama dalam minggu. Total selama 4 minggu didapatkan dari array yang menyimpan volume produksi per hari. Pilih angka yang benar untuk contoh program tersebut.

```
uTotalProduction := 0;
FOR wOnceAWeek := 1 TO ■ BY 7 DO
  uTotalProduction := uTotalProduction + uProductionByDate[2,wOnceAWeek];
END_FOR;
(* Ekstrak dan total volume produksi pada hari yang sama dalam seminggu selama 4 minggu mulai dari 1 Februari. *)
```

**Array yang menyimpan volume produksi per hari (uProductionByDate)**


- 22
- 21
- 4
- 28

Jawab  Kembali

# Tes

## Tes Akhir 10

Karakteristik struktur dalam ST
Pilih deskripsi struktur yang tidak benar.

- Struktur digunakan untuk mengatur dan menyimpan data pada perangkat menurut kondisi seperti status dan spesifikasi.
- Program yang memproses data dalam jumlah besar dapat dengan ringkas ditulis menggunakan struktur.
- Semua anggota yang ditentukan dalam struktur harus memiliki tipe data yang sama.
- Nilai bisa ditetapkan ke anggota dalam struktur yang sama tanpa menentukan satu per satu.

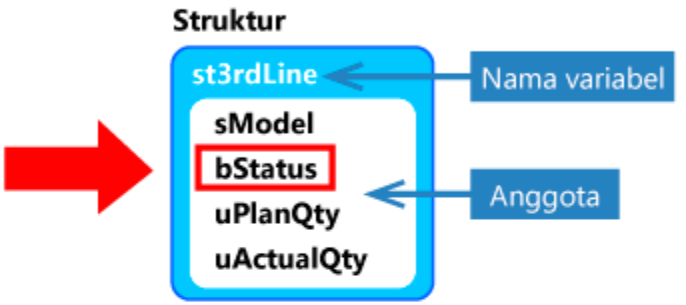
Jawab Kembali

# Tes

## Tes Akhir 11

Menentukan anggota untuk struktur dalam ST
Struktur berikut ini mengatur variabel-variabel yang terkait dengan lini produksi mobil.
Pilih deskripsi yang benar dengan menentukan "bStatus" anggota dalam structure ini.

| Parameter | Nama variabel |
|---|---|
| Model | sModel |
| Status | bStatus |
| Target produksi untuk hari ini | uPlanQty |
| Jumlah produksi saat ini | uActualQty |



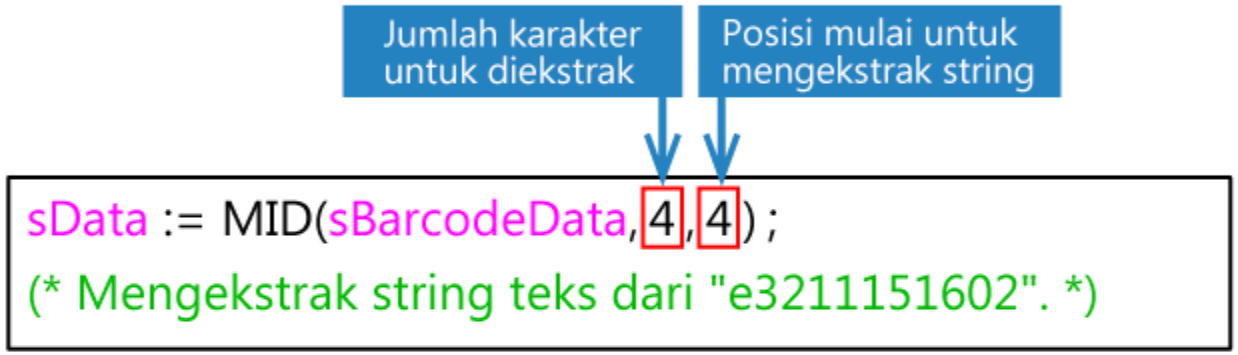

- st3rdLine.bStatus
- st3rdLine->bStatus
- st3rdLine[bStatus]
- st3rdLine[1]

Jawab Kembali

# Tes

## Tes Akhir 12

Penanganan string dalam ST
Contoh program berikut ini mengekstrak string tertentu dari string "e3211151602" yang disimpan dalam variabel "sBarcodeData". Fungsi MID mengekstrak jumlah karakter yang ditetapkan yang dimulai di posisi mulai yang ditetapkan.
Pilih string yang diekstrak dengan benar.




- 1151
- 1602
- e321
- 1115

Jawab Kembali

# Tes

## Skor Tes

Anda telah menyelesaikan Tes Akhir. Hasil Anda adalah sebagai berikut.
Untuk mengakhiri Tes Akhir, lanjutkan ke halaman berikutnya.

Jawaban yang benar : 12

Pertanyaan total : 12

Persentase : 100%

Lanjutkan Tinjau

**Selamat. Anda lulus tes ini.**

Anda telah menyelesaikan kursus Dasar-Dasar Pemrograman (Teks Terstruktur).

Terima kasih telah mengikuti kursus ini.

Kami harap Anda menikmati pelajaran, dan kami harap informasi yang diperoleh dalam kursus ini dapat bermanfaat di masa mendatang.

Anda dapat mengulas kursus ini sesering yang Anda inginkan.

Tinjau Tutup