MITSUBISHI ELECTRIC
Changes for the Better

PLC
Programlama Temelleri (Yapılandırılmış Metin)

Bu kurs, MELSEC programlanabilir kontrolörleri kontrol etmek için kullanılan temel programların nasıl oluşturulacağını açıklar.
Yapılandırılmış metin (ST) bu kurstaki program açıklamaları için kullanılır.

L(FOD)00553TUR

Giriş

Kursun amacı

Bu kursta MELSEC programlanabilir kontrolörler için yapılandırılmış metinde (ST) kontrol programlarının nasıl oluşturulacağını açıklanır.

Bu kursu görmeden önce, aşağıdaki kursun tamamlanması veya o kursun dengi bilgiye sahip olmak bir ön gerekliliktir:

Programlama Temelleri

C veya BASIC programlama dilini bilmek veya bu dillerde deneyim sahibi olmak, kursun içeriğini anlamaya yardımcı olabilir.

Giriş Kursun Yapısı

Bu kursun içeriği aşağıdaki gibidir.

Bölüm 1 - Yapılandırılmış metnin genel görünümü

Bu bölümde yapılandırılmış metnin (ST) özellikleri ve uygun uygulamalar açıklanır.

Bölüm 2 - ST programlarının temel kuralları

Bu bölümde ST'de programları oluşturmak için kullanılan temel kurallar açıklanır.

Bölüm 3 - G/Ç kontrol programlarını oluşturma

Bu bölümde G/Ç kontrol programlarının nasıl oluşturulacağı açıklanır.

Bölüm 4 - Aritmetik operasyonlar

Bu bölümde aritmetik operasyon programlarının nasıl oluşturulacağı açıklanır.

Bölüm 5 - Koşullu dallanma

Bu bölümde koşullu dallanma açıklanır.

Bölüm 6 - Veri saklama ve işleme

Bu bölümde veriyi saklamak ve işlemek için kısa programların nasıl yazılacağı açıklanır.

Bölüm 7 - Dize verisinin işlenmesi

Bu bölümde dize verisini işleme yöntemleri açıklanır.

Son Test

Geçme notu: %60 ve üstü

Giriş — Bu e-Eğitim aracının kullanımı

Sonraki sayfaya git		Sonraki sayfaya git.
Önceki sayfaya dön		Önceki sayfaya dön.
İstenen sayfaya ulaş		"İçindekiler Tablosu" görüntülenerek istediğiniz sayfaya ulaşabilmenizi sağlar.
Eğitimden çık		Eğitimden çık.

Giriş

Kullanım önlemleri

Güvenlik önlemleri

Gerçek ürünleri kullanarak öğrendiğinizde, lütfen ilgili kılavuzlardaki güvenlik önlemlerini dikkatlice okuyun.

Bu kurstaki önlemler

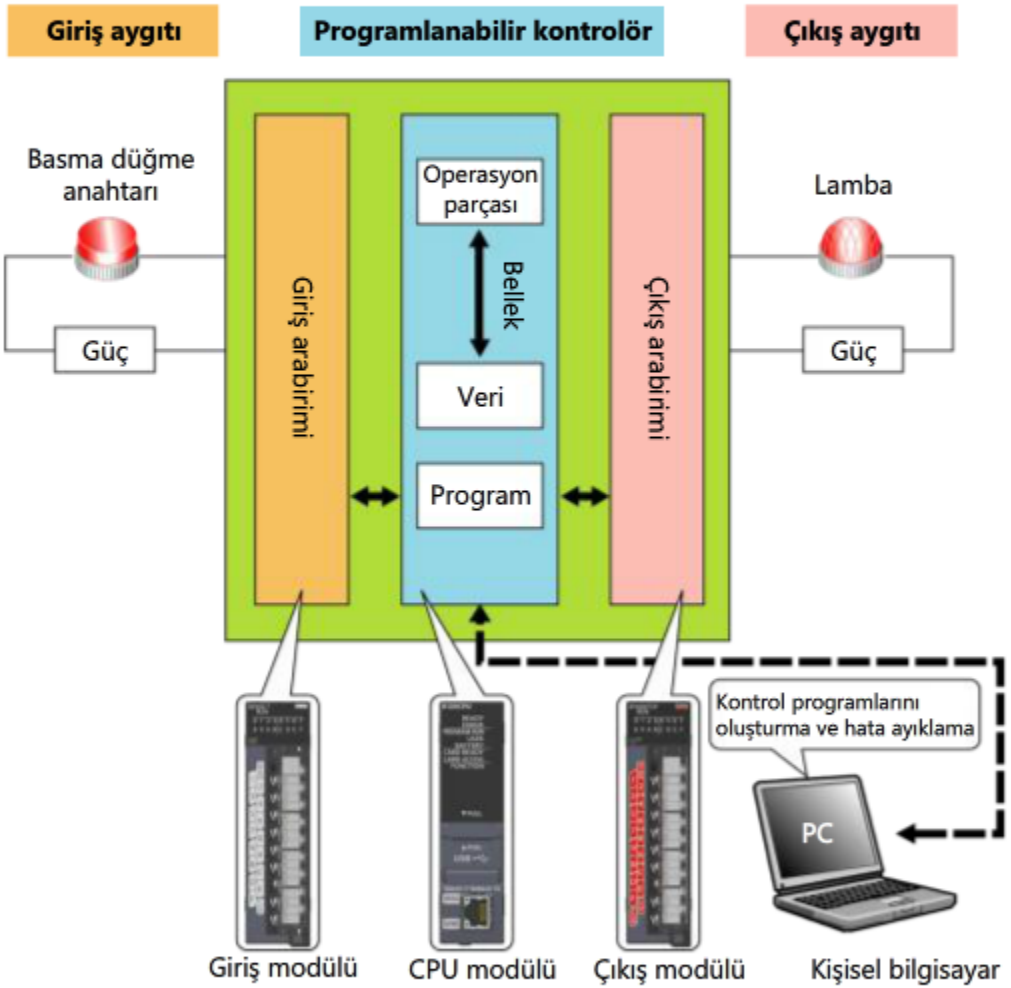
Kullandığınız MELSOFT mühendislik yazılımında görüntülenen ekranlar bu kurstakilerden farklı olabilir.
Bu kursta programlar oluşturulurken MELSOFT GX Works3 merdiven sembolleri kullanılır.

Bölüm 1 Yapılandırılmış metnin genel görünümü

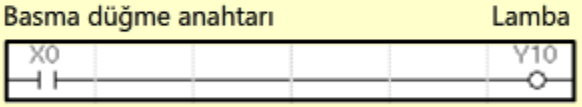
Bu bölümde yapılandırılmış metnin (ST) özellikleri ve uygun uygulamalar açıklanır.

1.1 Kontrol programları

Aşağıdaki şekilde, programlanabilir kontrolör sisteminin yapılandırılması gösterilmektedir.
Programlanabilir kontrolörler, kontrol programlarına göre çalışır.
Programlanabilir kontrolörlerin çalışması, kontrol programları oluşturularak istenen şekilde yapılandırılır.



Kontrol programı (Merdiven)



Bir kontrol programı yazıldığında basma düğme anahtarın durumuna yanıt olarak lamba açılacaktır.

Programlanabilir kontrolörlerin programlama dilleri International Electrotechnical Commission (IEC) (Uluslararası Elektroteknik Komisyonu) tarafından geliştirilen uluslararası standarda göre tanımlanır.

1.2 ST özellikleri ve diğer IEC programlama dilleriyle karşılaştırma

IEC 61131, programlanabilir kontrolör sistemlerinin uluslararası standardıdır.
Programlanabilir kontrolörlerin programlama dilleri IEC 61131-3 ile standartlaştırılır. ST, standart programlama dillerinden biridir.
Her dilin, uygulamanızı ve programlayıcıların becerisini saklayan farklı özellikleri vardır.

Aşağıdaki tablo, IEC 61131-3 programlama dillerinin özelliklerini listeler.

Programlama dili	Özellikler
Ladder **D**iagram (LD) (Merdiven Diyagram)	• Kontaklar ve bobinlerin sembolleri, bir elektrik devresine benzeyen bir program oluşturmak için kullanılır. • Program akışını takip etmek ve anlamak, yeni başlayanlar için bile kolaydır.
Structured **T**ext (ST) (Yapılandırılmış Metin)	• Programlar metin olarak (karakter) yazılır. • C veya BASIC programlama dilinde program yazma deneyimi olanlar ST dilini kolayca öğrenebilir. • Hesaplama formülleri, kolay anlaşılan matematik ifadelerine benzer. • ST, veri işleme için uygundur.
Function **B**lock **D**iagram (FBD) (İşlev Bloğu Diyagramı)	• Programlar, farklı işlevleri olan blokları düzenleyerek ve bloklar arasındaki ilişkileri belirterek yazılır. • Tüm operasyon kolayca görülebildiği için FDB, okunabilirliği iyileştirir.
Sequential **F**unction **C**hart (SFC) (Sıralı İşlev Tablosu)	• Koşullar ve süreçler, akış tabloları olarak yazılır. • Program akışını anlamak kolaydır.
Instruction **L**ist (IL) (Yönerge Listesi)	• IL, makine diline benzer. • IL günümüzde nadiren kullanılır.

Bu kursta ST kullanarak nasıl temel kontrol programlarının yazılacağı açıklanır.

1.3 Özet

Bu bölümün içeriği şunlardır:

- Programlanabilir kontrolör sistemleri ve kontrol programları arasındaki ilişki
- Kontrol programlarının standardı
- ST'nin Özellikleri

Dikkat edilecek önemli noktalar:

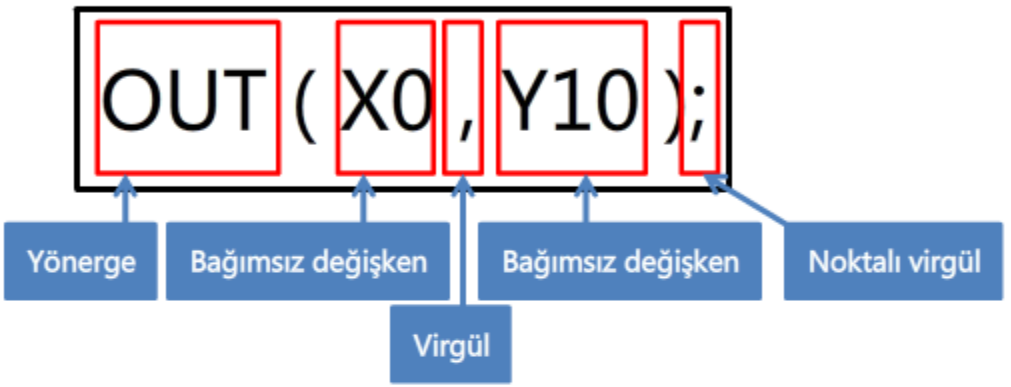
Programlanabilir kontrolör sistemleri ve kontrol programları arasındaki ilişki	• Programlanabilir kontrolörler, kontrol programlarına göre çalışır. • Programlanabilir kontrolörlerin çalışması, kontrol programları oluşturularak istenen şekilde yapılandırılır.
Kontrol programlarının standardı	• ST, IEC programlama dillerinden biridir. • Diğer IEC programlama dillerine LD, FBD, SFC ve IL dahildir ve bunların her birinin uygulamanızı ve programlayıcıların becerisini saklayan farklı özellikleri vardır.
ST'nin Özellikleri	• C veya BASIC dilinde program yazma deneyimi olanlar ST dilini kolayca öğrenebilir. • Toplama ve çıkarma gibi hesaplamalar genellikle kolay anlaşılan yaygın matematik ifadeleri kullanılarak yazılabilir. • ST, veri işleme için uygundur.

Bölüm 2 ST'deki programların temel kuralları

Bu bölümde ST'de programları oluşturmak için kullanılan temel kurallar açıklanır.

2.1 Temel program örneği (G/Ç kontrol ifadesi)

Bu bölümde temel ST programının bir örneği gösterilmektedir.
Aşağıdaki örnek programda giriş X0 açıldığında çıkış Y10 açılır ve X0 kapatıldığında Y10 kapatılır.



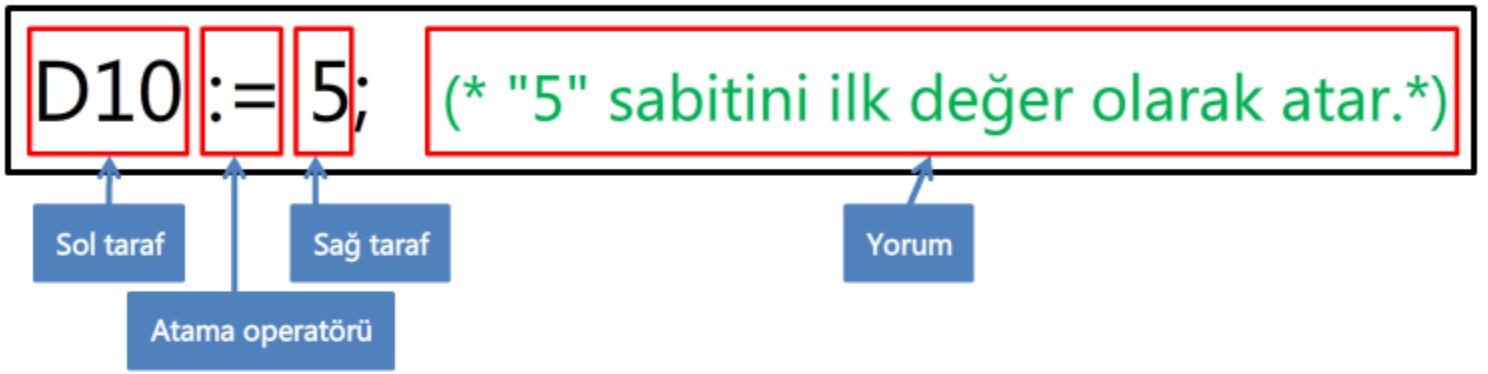
Yürütülecek operasyonu bir yönerge tanımlar.
Bağımsız değişkenler bir yönergenin ardından parantez içinde yazılır.
Bağımsız değişkenler; değişkenleri, aritmetik ifadeleri ve sabit değerleri açıklamak için kullanılır.
MELSEC programlanabilir kontrolörlerde CPU modülü aygıtları değişkenler olarak kullanılabilir.

Bağımsız değişkenlerin sayısı yönergelere bağlıdır.
Çoklu bağımsız değişkenler virgülle (,) ayrılır.

Yukarıda gösterilen bir satır, bir ifadeyi temsil eder. Her ifade, noktalı virgülle (;) sona erer.
Bir program, ifadeleri birleştirerek yazılır.

2.2 Temel program örneği (Atama ifadesi)

Sonraki örnekte bir atama ifadesi kullanan bir program gösterilmektedir.
Aşağıdaki ifade, ondalık sabiti "5"'i "D10" değişkenine atar.



Atama ifadesi için bir atama operatörü (:=) kullanılır. (:) sütununun, eşit işaretinin sol tarafına yerleştirildiğini unutmayın (=).
Bir atama operatörü, sağ taraftaki değeri sol tarafa atar.

Bir programa bir yorum eklemek, operasyonu daha anlaşılabilir kılar. Yorumları iki yıldız arasına dahil edin (* *).

2.3 Sayısal notasyon

Önceki sayfadaki örnek programla değişkene ondalık bir değer atanmıştır.

Bazen, sıralı kontrol için ikili ve onaltılık gibi ondalıklardan başka değerler kullanılır.
Aşağıdaki tabloda MELSEC programlanabilir kontrolörler için ST'de kullanılan sayısal notasyon tiplerini listelenir.

Sayısal notasyon tipi	Notasyon yöntemi	Örnek
İkili	Bir "2#" ön eki ekleyin.	**2#**11010
Sekizli	Bir "8#" ön eki ekleyin.	**8#**32
Ondalık	Doğrudan giriş	26
	Bir "K" ön eki ekleyin.	**K**26
Onaltılık	Bir "16#" ön eki ekleyin.	**16#**1A
	Bir "H" ön eki ekleyin	**H**1A

Değerleri değişkenlere atamak için program örnekleri aşağıda gösterilmiştir.

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

2.3.1 Bit notasyonu

Bitler, sinyallerin on/off (açık/kapalı) durumları gibi true/false (doğru/yanlış) koşulları temsil eder. Bitler ayrıca kurma/kurmama koşullarını da temsil eder.
ST'de bitler "ON" ve "OFF" olarak yazılamaz. Bunlar "1" (ON) ve "0" (OFF) olarak ifade edilir. Bitler ayrıca "TRUE" ve "FALSE" olarak ifade edilebilir.

Aşağıdaki tabloda farklı notasyon tipleri listelenir.

Durum	ON	OFF
	True	False
Sayısal notasyon	1	0
True/false notasyonu	TRUE	FALSE

Burada bit tipi değişkenlere bazı değer atama örnekleri verilmektedir.

Sayısal notasyon

```
X0 := 1;
```

=

True/false notasyonu

```
X0 := TRUE;
```

Sayısal notasyon

```
X0 := 0;
```

=

True/false notasyonu

```
X0 := FALSE;
```

2.4 Program yürütme sıralaması

ST ifadeleri yukarıdan aşağıya yürütülür.

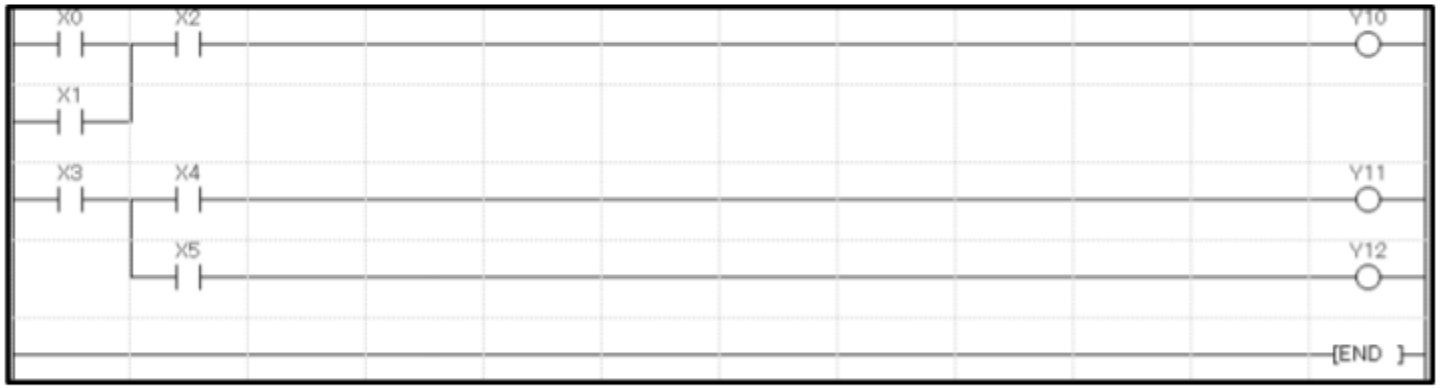
ST program örneği

```
Y10 := (X0 OR X1) AND X2;        (* İlk olarak yürütülür *)
Y11 := X3 AND X4;                (* İkinci olarak yürütülür *)
Y12 := X3 AND X5;    (* Üçüncü olarak yürütülür. Sonda bir END ifadesi gerektirmez.*)
```

Tekrarlanan yürütme

*LD'de programın sonunda END ifadesi gerekse bile, ST'de bu gereksizdir.

Aşağıdaki merdiven programı yukarıdaki ST program örneğiyle aynı operasyonu temsil eder.



Tekrarlanan yürütme

LD'de olduğu gibi ST'deki yönergeler son yönergeye ulaştıktan sonra ilk yönergeye geri dönüp tekrarlanarak yürütülür.

2.5 Özet

Bu bölümün içeriği şunlardır:

- Temel ST programı
- Atama ifadesi formatı
- Sayısal notasyon
- Program yürütme sıralaması
- Yorum

Dikkat edilecek önemli noktalar:

Temel ST programı	• ST programlarının minimum elemanı bir ifadedir. • Her ifade, noktalı virgülle (;) sona erer. • Bir program, ifadeleri birleştirerek yazılır.
Atama ifadesi formatı	• Atamalar için bir atama operatörü (:=) kullanılır.
Sayısal notasyon	• ST'de sayısal notasyon tipleri • ST'de bit değerleri için "ON" ve "OFF" notasyonları yerine "1" ve "0" kullanılır. • Bit değerleri ST'de ayrıca "TRUE" ve "FALSE" olarak da belirtilir.
Program yürütme sıralaması	• ST'de oluşturulan programlar yukarıdan aşağıya yürütülür. • LD programlarda olduğu gibi ST programlar sürecin sonuna ulaşıldığında programın başına dönerek tekrar tekrar işlenir.
Yorum	• Bir programa bir yorum eklemek, operasyonu daha anlaşılabilir kılar. • Yorumlar iki yıldız imi arasına dahil edilir (* *).

Bölüm 3 G/Ç kontrol programları oluşturma

Bu bölümde G/Ç kontrol programlarının ST'de nasıl oluşturulacağı açıklanır.

3.1 G/Ç kontrol programları

Aşağıdaki bir programlanabilir kontrolörün G/Ç kontrolünün program örneğidir.

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

- OUT: Çıkış yönergesi
- X0: Yürütme koşulu (bağımsız değişken)
- Y10: Çıkış aygıtı (bağımsız değişken)

"OUT", çıkış yönergesidir. Bir bağımsız değişken, bir yürütme koşulunu ve çıkışın yönlendirildiği aygıtı belirtir. Yürütme koşulu X0 yerine getirildiğine Y10 aygıtı açılır.

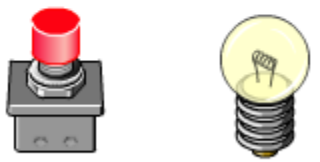
Aşağıda gösterilen giriş anahtarına tıklayın. Giriş anahtarı X0 açılır.

- Giriş anahtarı X0 açıldığında, çıkış lambası Y10 açılır.
- Giriş anahtarı X0 kapatıldığında, çıkış lambası Y10 kapatılır.

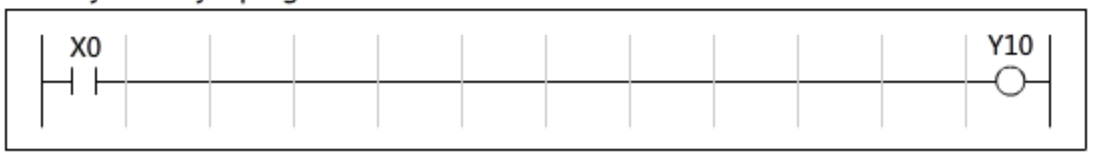
ST'de yazılmış G/Ç kontrol programı örneği

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

Giriş anahtarı X0 Çıkış lambası Y10



LD'de yazılan aynı program



3.1 G/Ç kontrol programları

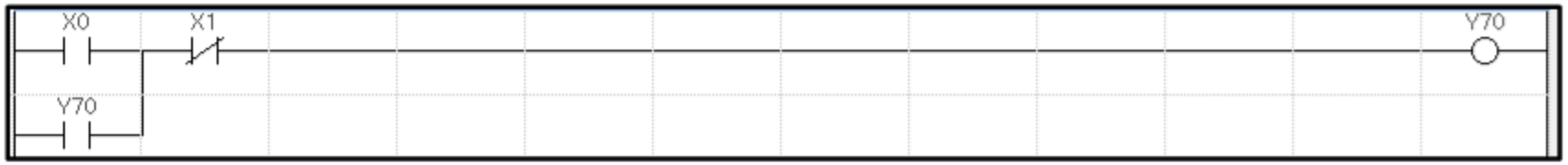
LD'ye benzer şekilde, OUT yönergesinin yanı sıra G/Ç kontrolü yönergeleri ve veri işleme yönergeleri gibi pek çok yönerge bulunmaktadır.
ST'de bulunan yönergeler hakkında daha fazla bilgi için programlanabilir kontrolörünüzün programlama kılavuzuna bakın.

Unutmayın ki "OUT(X0, Y10);" ifadesini "Y10 := X0;" olarak yazmak aynı operasyonu oluşturur.

```
Y10 := X0;  (* "OUT(X0, Y10);" ile aynı operasyon *)
```

3.2 Çoklu koşulları birleştirme

Aşağıdaki merdiven programı, kendinden tutunan bir devreyi temsil eder.



Aynı program aşağıdaki gibi ST'de yazılabilir.

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

Mantıksal operatör

Yukarıda gösterildiği gibi mantıksal operatörler ST'de çoklu koşulları birleştirmek için kullanılır.

Aşağıdaki tabloda mantıksal operatörler listelenmektedir.

Operatör	Anlamı
OR	Mantıksal VEYA
AND	Mantıksal VE
NOT	Mantıksal değilleme
XOR	Özel kullanım VEYA

3.3 Değişkenlere anlamlar tanımlama

ST'yi MELSEC programlanabilir kontrolörle kullanarak hem aygıtlar hem de etiketler, değişkenlere diğer ad olarak atanabilir. Kullanıcılar uygulamalara göre etiketleri kullanabilir.

Uygulamayla ilgili bir etiket atandığında, operasyonu anlamak daha kolay olur.

```
Y10 := (X0 OR X1) AND X2;     (* Aygıt adları kullanılarak yazılır *)
```

```
Lamp := (Switch0 OR Switch1) AND Switch2;     (* Etiketler kullanılarak yazılmış *)
```

Etiketler MELSOFT mühendislik yazılımı kullanılarak adlandırılabilir.

Bu kurstaki sonraki program örnekleri etiketler kullanılarak açıklanmıştır.

3.4 Özet

Bu bölümün içeriği şunlardır:

G/Ç kontrol programı örnekleri

- Mantıksal operatörler ST'de çoklu koşulları birleştirmek için kullanılır.
- Aygıt adları ve etiketleri, değişken adları olarak kullanılabilir.

Dikkat edilecek önemli noktalar:

Çoklu koşulları birleştirme	• Mantıksal operatörler ST'de koşulları birleştirmek için kullanılır.
Değişkenlere anlamlar tanımlama	• Uygulamayla ilgili bir etiket atandığında, operasyonu anlamak daha kolay olur.

Bölüm 4 Aritmetik operasyonlar

Bu bölümde aritmetik operasyon programlarının nasıl oluşturulacağı açıklanır.

- Aritmetik operasyonları açıklama
- Sayısal aralıklarla ilgili veri tiplerini belirtme
- Veri tipi tutarsızlıklarını önlemek için değişkenleri adlandırmak

4.1 Temel aritmetik operasyonlar

Bu örnek program, iki ayrı üretim hattının üretim hacminin toplamını alır.
Denklemin sağ tarafı, değişkenleri ve aritmetik operasyonları içeren bir aritmetik operasyondur.

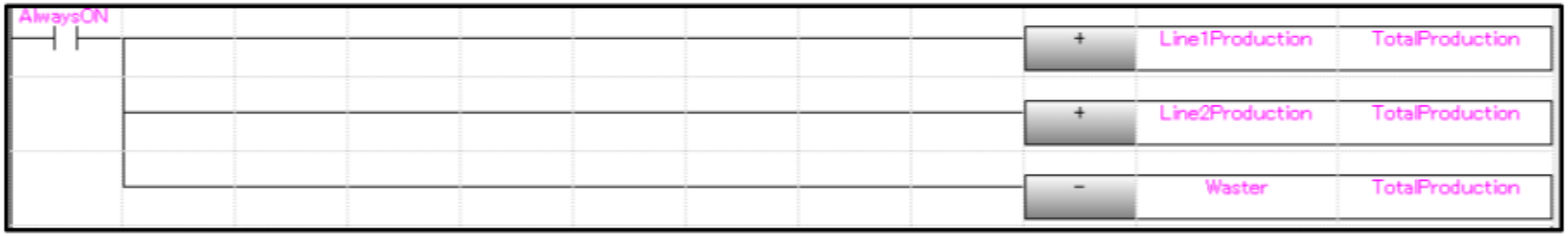
ST'de yazılmış örnek aritmetik program

Toplama operatörü

Çıkarma operatörü

```
TotalProduction := Line1Production + Line2Production - Waster;
(* İki üretim hattının üretim hacminin toplamını alın, hatalı ürünlerin sayısını
toplamdan çıkarın ve edinilen değeri atayın. *)
```

LD'de yazılmış aynı program aşağıdaki gibidir.



Yukarıda gösterildiği gibi, program Merdivende 3 satır kullanılarak yazılmalıdır ama ST'de 1 satırda yazılabilir.

Aşağıdaki tabloda temel aritmetik operatörler listelenmektedir.

Operatör	Anlamı
+	Toplama
-	Çıkarma
*	Çarpma
/	Bölme

4.2 Değişkenlerin veri tipleri

İşlenecek veri aralıklarını tanımlamak için her değişkene bir veri tipi tanımlanmalıdır.
ST'de kullanılan sayısal değerlerin veri tipleri bit, tamsayı ve gerçek sayı tipleridir.

ST'de kullanılan veri tiplerinin yanı sıra aşağıdaki tabloda bu kursta kullanılan veri tipleri gösterilir.

Veri tipi		Veri aralığı
Bit		Bit cihazlarının AÇIK/KAPALI durumu ve yürütme sonuçlarının doğru/yanlış durumu
Tamsayı	Sözcük (imzasız)	0 - 65.535
	Sözcük (imzalı)	-32.768 - 32.767
	Çift sözcük (imzasız)	0 - 4.294.967.295
	Çift sözcük (imzalı)	-2.147.483.648 - 2.147.483.647

Tamsayı tipi kullanılırken, veri aralığına göre sözcük veya çift sözcük tipini seçin ve negatif değerleri işleme gerekliliğine göre imzalı veya imzasız tipi seçin.
Etiket adı ayarlandığında MELSOFT mühendislik yazılımını kullanarak bir değişkenin veri tipini belirtin.

4.3 Veri tiplerini temsil eden değişken adları

Bir atama denkleminin sol ve sağ taraflarındaki farklı veri tiplerini kullanmak bir derleme hatasına veya beklenmeyen sonuca yol açabilir.
Aşağıda böyle bir durumun örneği verilmiştir.

```
ValueA := ValueB;      (* ValueA: Sözcük tamsayı  ValueB: Çift sözcük tamsayı *)
```

Çift sözcüklü tamsayı bir sözcük tamsayısına atanamaz. Ancak bu durumda veri tipi fark edilemez.

Veri tipini temsil eden ön ekler, veri tiplerini görsel olarak tanımlanabilir kılmak için değişken adlarına eklenebilir.
Bu tür değişken adlandırmasına Macar notasyonu denir.

Veri tipi		Veri aralığı	Ön ek	Ön ekin uzatması
Bit		Bit cihazlarının AÇIK/KAPALI durumu ve yürütme sonuçlarının doğru/yanlış durumu	b	**B**it
Tamsayı	Sözcük (imzasız)	0 - 65.535	u	**u**nsigned word (imzasız sözcük)
	Sözcük (imzalı)	-32.768 - 32.767	w	signed **w**ord (imzalı sözcük)
	Çift sözcük (imzasız)	0 - 4.294.967.295	ud	**u**nsigned **d**ouble-word (imzasız çift sözcük)
	Çift sözcük (imzalı)	-2.147.483.648 - 2.147.483.647	d	signed **d**ouble-word (imzalı çift sözcük)

Bu sayfanın başındaki örnek program, Macar notasyonu kullanılarak aşağıdaki gibi yazılabilir:

```
wValueA := dValueB;  (* Çift sözcük değişkeni bir sözcük değişkenine atanamaz. *)
```

Macar notasyonunu kullanarak, veri tipi tutarsızlıkları program yazma sürecinde tanımlanabilir.

Kursun geri kalanında örnekteki değişken adları Macar notasyonunda yazılır.

4.4 Özet

Bu bölümün içeriği şunlardır:

- Aritmetik operasyonları açıklama
- Sayısal aralıklarla ilgili veri tiplerini belirtme
- Veri tiplerini temsil eden değişken adlarını eklemek

Dikkat edilecek önemli noktalar:

Temel aritmetik operasyonlar	• Genel programlama dillerinde yaygın olan operatörler ST'de hesaplamaları ifade etmek için kullanılabilir.
Değişkenlerin veri tipleri	• İşlenecek veri aralıklarını tanımlamak için her değişkene bir veri tipi tanımlanmalıdır.
Veri tiplerini temsil eden değişken adlarını eklemek	• Macar notasyonunu kullanarak değişken adlarını açıklamak, program yazarken veri tiplerindeki tutarsızlıkların tanımlanmasını sağlar.

Bölüm 5 Koşullu dallanma

Kontrol programları ayrıca gerçek işlemenin belirtilen koşullara göre değiştiği kod bölümlerini içerir.
Bu bölümde koşullu dallanma açıklanır.

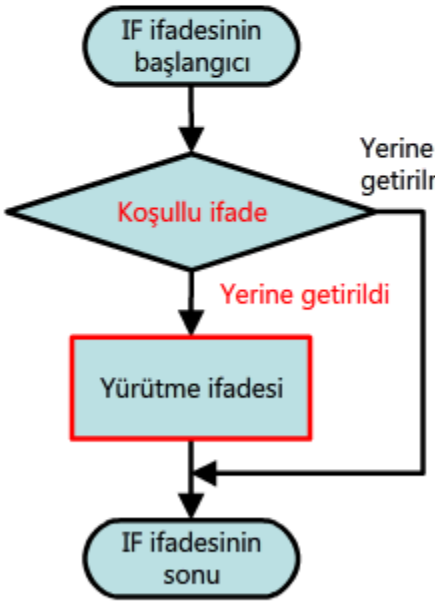
5.1 Koşullu dallanma (IF)

IF ifadeleri koşullu dallanma için kullanılır. IF ifadeleri aşağıdaki gibi açıklanır.

```
IF conditional expression THEN
    Execution statement;          (* Koşullu ifade yerine getirilirse ifade yürütülür. *)
END_IF;                           (* END_IF; IF ifadelerinin sonuna yerleştirilmelidir. *)
```

Bu örnek programda koşullu ifade yerine getirilirse bir ifade yürütülür. Koşullu ifade yerine getirilmezse ifade yürütülmez.

Aşağıdaki şekilde bu örnek programdaki operasyon akışı gösterilmektedir.

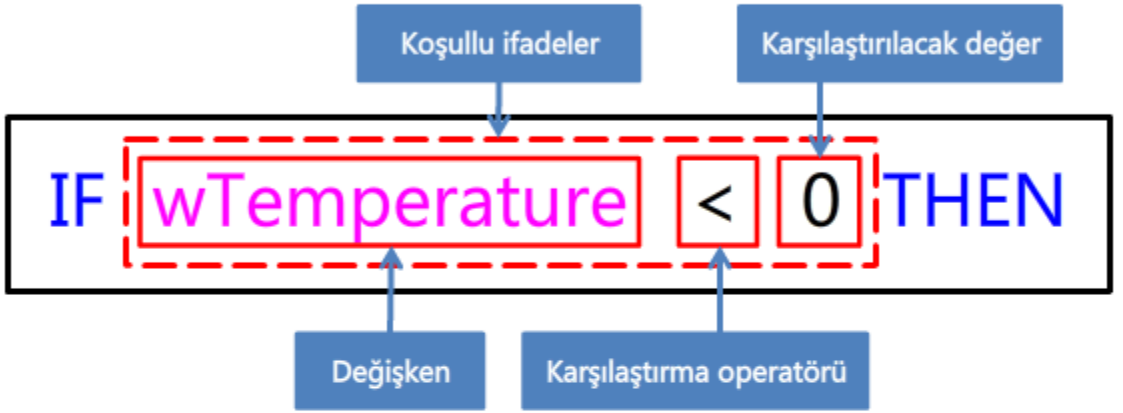


Aşağıdaki örnekte, değişken değerleri karşılaştırılarak programın dallanması gösterilmektedir. Örnek programda, kontrol panelinde sıcaklık 0 derecenin altına düştüğünde ısıtıcı açılır.

```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Kontrol panelindeki sıcaklık 0 derecenin altına düştüğünde ısıtıcı açılır. *)
END_IF;
```

5.1.1 Koşullu ifadeleri yazma

Önceki sayfada "wTemperature değişkeninin değeri 0'dan düşük olduğunda" anlamına gelen "wTemperature < 0" koşullu ifadesi açıklanmıştı.
Bu ifadede olduğu gibi, koşullu ifadelerde karşılaştırılacak değişkenler ve değerlerin arasındaki ilişkiyi temsil etmek için karşılaştırma operatörleri kullanılır.



Karşılaştırma operatörünün sol ve sağ taraflarında değerler, karşılaştırma için değişkenler veya sabitler olarak yazılır.

Değişkenleri ve sabitleri karşılaştırmaya ek olarak koşullu ifadeler değişkenleri karşılaştırmak ve karşılaştırma sonuçlarının veya bit tipi değişkenlerin mantıksal operasyonlarını gerçekleştirmek için yazılabilir.

Değişkenleri karşılaştırma
- uValue1 <= uValue2

İki karşılaştırma sonucu için mantıksal operasyon
- (10 < uValue) AND (uValue <= 50)

İki bit tipi değişken için mantıksal operasyon
- bSwitch0 OR bSwitch1

Aşağıdaki tabloda karşılaştırma operatörlerinin tipleri listelenmektedir.

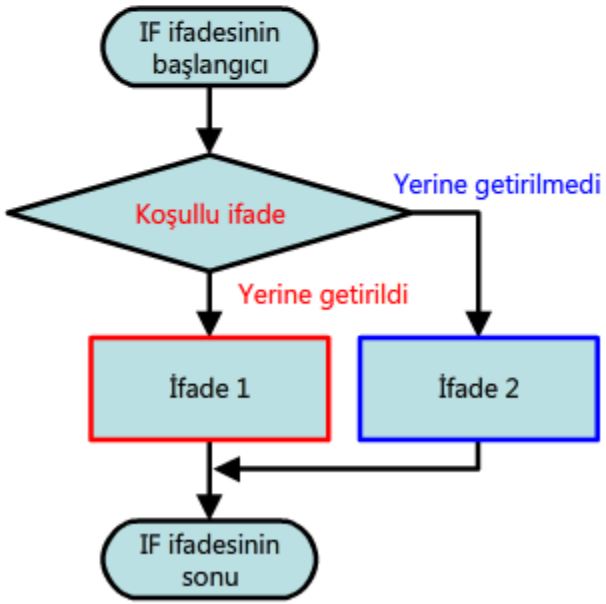
Operatör	Anlamı
>	Daha büyük
<	Daha küçük
>=	Daha büyük veya eşit
<=	Daha küçük veya eşit
=	Eşittir
<>	Eşit değildir

5.1.2 IF ifadesi için özel dallanma (ELSE)

Koşullu ifade yerine getirildiğinde bir ifadeyi yürütmek için basit IF ifadeleri (5.1'e bakın) kullanılır.
Koşullu ifade yerine getirilmediğinde farklı bir ifadeyi yürütmek için bir ELSE ifadesi kullanılır.

```
IF conditional expression THEN
  Execution statement 1;          (* Koşullu ifade yerine getirildiğinde 1 ifadesi yürütülür.  *)
  ELSE
  Execution statement 2;          (* Koşullu ifade yerine getirilmediğinde 2 ifadesi yürütülür *)
END_IF;
```

Aşağıdaki şekilde ELSE ifadesi kullanılırken operasyon akışı gösterilmektedir.



Aşağıdaki örnek program, koşulun yerine getirilip getirilmediğine göre farklı ifadeleri yürütür.

5.1'deki örnek programda, 0 dereceye ulaştıktan sonra bile ısıtıcının sıcaklığı arttırdığı bir kusur vardır. Ancak aşağıdaki program "wTemperature" 0 dereceyi aştığında ısıtıcıyı kapatır.

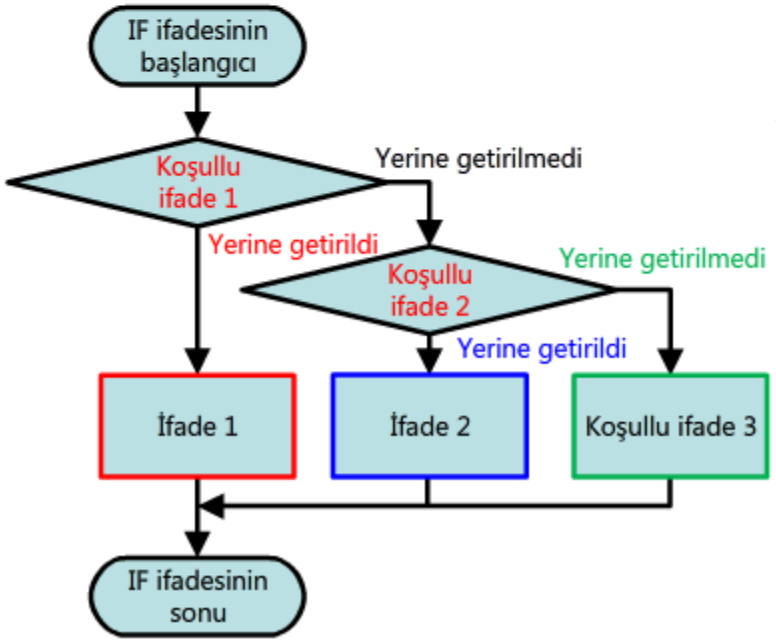
```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Sıcaklık 0 derecenin altına düştüğünde ısıtıcıyı açar. *)
  ELSE
  bHeater := 0;  (* Sıcaklık 0 dereceye ulaştığında veya 0 dereceyi aştığında ısıtıcıyı kapatır *)
END_IF;
```

5.1.3 IF ifadesi için ek dallanma (ELSIF)

Koşullu ifade yerine getirilmediğinde farklı bir ifadeyi yürütmek için ELSE ifadeleri kullanılır.
ELSIF ifadelerini kullanarak başka bir koşullu dallanma eklenebilir, bu önceki koşullu ifadenin yerine getirilmediği anlamına gelir ve başka bir koşullu ifadenin kontrol edilir.

```
IF Conditional expression 1 THEN
    Execution statement 1;      (* Koşullu ifade 1 yerine getirildiğinde 1 ifadesi yürütülür. *)
ELSIF Conditional expression 2 THEN
    Execution statement 2;      (* Eğer koşullu ifade 1 yerine getirilmezse ve koşullu ifade 2 yerine getirilirse İfade 2 yürütülür. *)
ELSE
    Execution statement 3;      (* Eğer koşullu ifade 1 ve 2 yerine getirilmezse ifade 3 yürütülür. *)
END_IF;
```

Aşağıdaki şekilde ELSIF ifadesi kullanılırken operasyon akışı gösterilmektedir.



Sıcaklık 40 dereceyi aştığında bu durumla başa çıkmak için ELSIF ifadesi şekil 5.1.2'de gösterilen program örneğine eklenir.

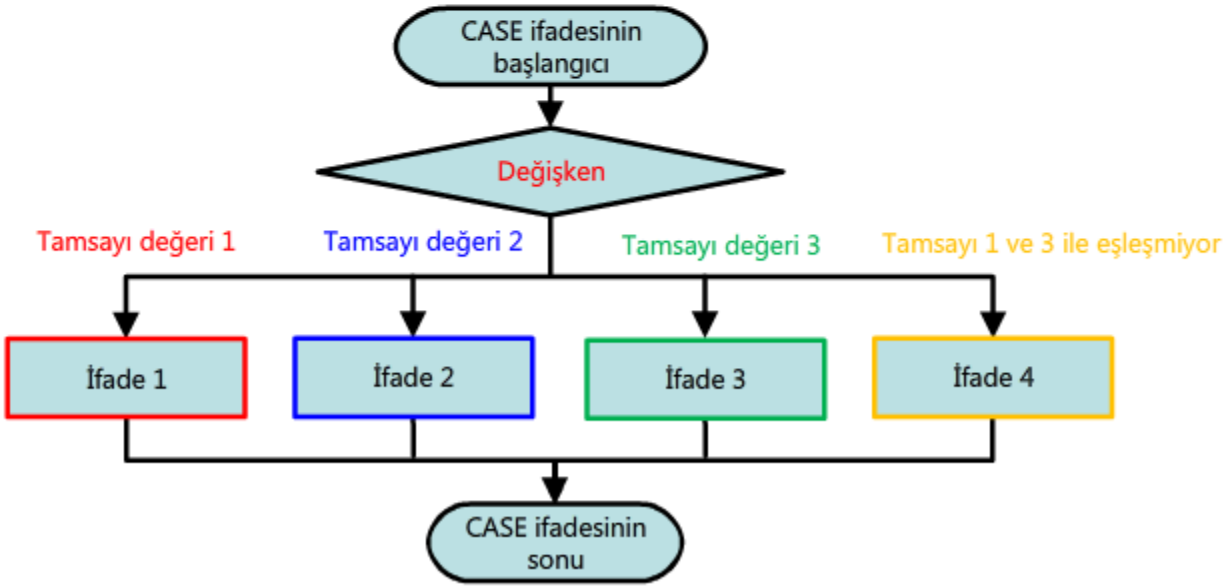
```
IF wTemperature < 0 THEN
   bHeater := 1; (* Sıcaklık 0 derecenin altına düştüğünde ısıtıcıyı açar. *)
   bCooler := 0; (* Sıcaklık 0 derecenin altına düştüğünde soğutucuyu kapatır. *)
   ELSIF 40 < wTemperature THEN
   bHeater := 0; (* Sıcaklık 40 dereceyi aştığında ısıtıcıyı kapatır. *)
   bHeater := 1; (* Sıcaklık 40 dereceyi aştığında soğutucuyu açar. *)
   ELSE
   bHeater := 0; (* Önceki koşulların hiçbiri yerine getirilmezse ısıtıcıyı kapatır. *)
   bCooler := 0; (* Önceki koşulların hiçbiri yerine getirilmezse soğutucuyu kapatır. *)
END_IF;
```

5.2 Tamsayı değerlerine göre koşullu dallanma (CASE)

IF ifadeleri koşullu ifadelerin yerine getirilip getirilmediğine bakılmaksızın dallanma için kullanılır.
CASE ifadeleri tamsayı değerlerine göre dallanma için kullanılır.
Aşağıdaki şekilde, bir CASE ifadesinin nasıl yazılacağı gösterilmektedir.

```
CASE Variable OF
    Integer value 1: Execution statement 1;      (* Değişken, tamsayı değeri 1 ile eşleştiğinde ifade 1 yürütülür. *)
    Integer value 2: Execution statement 2;      (* Değişken, tamsayı değeri 2 ile eşleştiğinde ifade 2 yürütülür. *)
    Integer value 3: Execution statement 3;      (* Değişken, tamsayı değeri 3 ile eşleştiğinde ifade 3 yürütülür. *)
    ELSE Execution statement 4;                  (* Değişken tamsayı değerlerinin hiçbiriyle eşleşmezse ifade 4 yürütülür.  *)
END_CASE;    (* END_CASE; CASE ifadesinin sonuna yerleştirilmelidir. *)
```

Aşağıdaki şekilde CASE ifadesi kullanılırken operasyon akışı gösterilmektedir.



5.2.1 CASE ifadesi için örnek program

CASE ifadesi yürütmesi, örnek program operasyonu kullanılarak açıklanır.

Sonraki sayfaya geçmek için üzerine tıklayın.
Animasyonu tekrar izlemek için "Oynat" düğmesine tıklayın.

Oynat


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

Ağırlık	uSize	Sınıf
0 ila 20 kg	1	M
21 ila 30 kg	2	L
31 ila 40 kg	3	XL
41 kg ve üstü	4	Oversize

5.3 Özet

Bu bölümün içeriği şunlardır:

- IF ifadeleriyle koşullu dallanma
- Koşullu ifadeleri yazma
- Tamsayı değerlerine göre koşullu dallanma (CASE ifadesi)

Dikkat edilecek önemli noktalar:

IF ifadesi	• IF ifadesiyle, bir koşullu ifade yerine getirildiğinde program dallanır. • Koşullu ifade yerine getirilmezse dallanma için bir ELSE ifadesi kullanılır. • IF ifadesindeki koşullu ifade yerine getirilmezse başka bir dallanma eklemek için bir ELSIF ifadesi kullanılır.
Koşullu ifade	• Koşullu ifadeler, karşılaştırma operatörlerini kullanarak karşılaştırma yapmak için değişkenler ve değerler arasındaki ilişkiyi temsil eder.
CASE ifadesi	• CASE ifadeleri tamsayı değerlerine göre dallanma için kullanılır.

Bölüm 6 Veri saklama ve işleme

Üretim sistemlerinin temeli olarak büyük miktarda veriyi işlemek için günümüzde G/Ç kontrol uygulamalarında olduğu gibi programlanabilir kontrolörler kullanılır.
Büyük miktarda veriyi işlemek için veri saklanmalı ve sonra gerektiği gibi okunmalıdır.
Bu bölümde veriyi saklamak ve işlemek için kısa programların nasıl yazılacağı açıklanır.

- Değişkenleri sıralamak ve düzenlemek için diziler kullanılır.
- İlgili değişkenleri organize etmek için veri yapıları kullanılır.
- Döngü işleme programları, FOR ifadelerini kullanan dizileri etkili bir şekilde işler.

Veriyi saklamak ve işlemek için diziler, veri yapıları ve FOR ifadeleri kullanılarak kısa programlar oluşturulabilir.

6.1 Veriyi sıralama ve saklama (Dizi)

Dizileri kullanarak bir değişkenle çoklu değerleri işleyebilirsiniz.
Aşağıdaki örnekte bir otomotiv üretim fabrikasındaki üretim hacmi verisi, hedef ülkeye göre saklanmaktadır.

Hedef	A Ülkesi	B Ülkesi	C Ülkesi
Üretim hacmi	35	75	65

Hedef ülkeye göre üretim hacmi verisi bir değişkene atanmıştır. Her hedef için dizileri kullanmadan bir değişken oluşturulmalıdır.
Ancak dizileri kullanarak çeşitli hedeflerin üretim hacmi tek bir değişkene atanabilir ve burada saklanabilir.

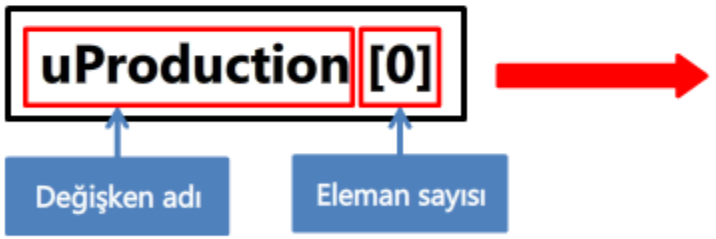
Dizi kullanmadan

uProductionA
uProductionB
uProductionC

Dizi kullanarak

uProduction

Dizideki bireysel değişkenler, eleman sayıları kullanılarak belirtilmiştir. Eleman sayıları sıfırdan [0] başlar.



Hedef (satır)

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

Aşağıdaki program örneğinde A Ülkesi için planlanan üretim hacmi değişkeni atanmıştır.

```
uShowProductionPlan := uProduction[0];
(* A ülkesi için eleman sayısını belirtir. *)
```

6.1.1 Matris dizisi

Sonra, hedef verisine ek olarak boya rengi verisi kullanılır.

Hedef	A Ülkesi			B Ülkesi			C Ülkesi		
Boya rengi									
Üretim hacmi	10	5	20	15	40	20	25	30	10
	toplam 35			toplam 75			toplam 65		

Aşağıdaki tabloda gösterildiği gibi, veri her hedef ülke için (satır) boya rengine (sütun) göre ayrılabilir ve saklanabilir.

Boya rengi (sütun)

Hedef (satır)	Kırmızı	Sarı	Mavi
A Ülkesi	[0,0] 10	[0,1] 5	[0,2] 20
B Ülkesi	[1,0] 15	[1,1] 40	[1,2] 20
C Ülkesi	[2,0] 25	[2,1] 30	[2,2] 10

Hedefi temsil eden eleman sayısı

Boya rengini temsil eden eleman sayısı

Veriyi bu şekilde satırlara ve sütunlara göre düzenleyen diziler, matris dizileri olarak bilinir.
Satırları ve sütunları temsil eden eleman sayıları virgülle ayrılır.

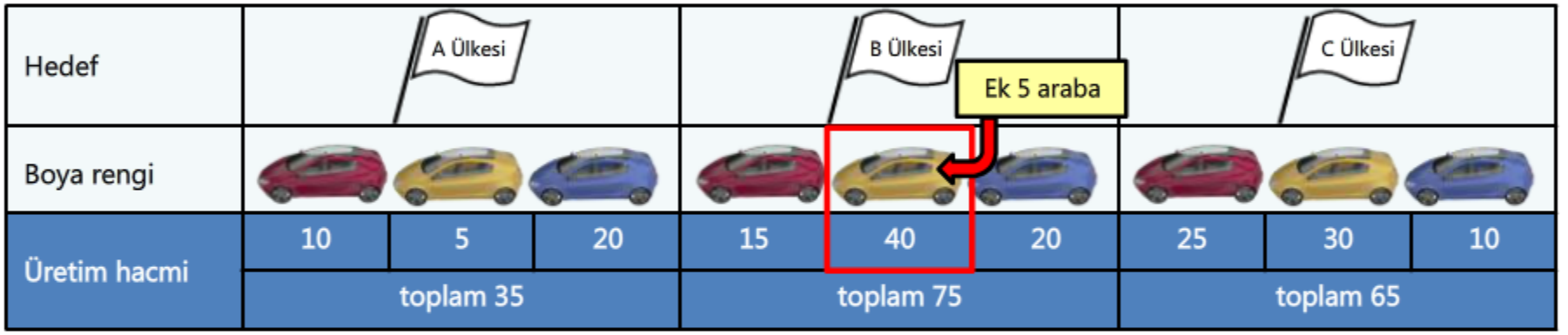
Dizi değişkeni (matris dizisi)

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

6.1.2 Matris dizi ataması

Aşağıdaki örnek program, matris dizileri kullanarak B Ülkesi için sarı otomobilin planlanan üretim hacmine ek olarak acil üretilmesi gereken araba sayısını atar.

```
uAdditionalProduction := 5;
uProduction[1,1] := uProduction[1,1] + uAdditionalProduction;
(* İlk planlanan üretim hacmine ek üretim miktarını (5 birim) ekler. *)
```



Boya rengi (sütun)

Hedef (satır)	Kırmızı	Sarı	Mavi
A Ülkesi	[0,0] 10	[0,1] 5	[0,2] 20
B Ülkesi	[1,0] 15	[1,1] 40 -> 45	[1,2] 20
C Ülkesi	[2,0] 25	[2,1] 30	[2,2] 10

6.1.3 Matris dizilerinde saklanan işlem bilgisi

Aşağıdaki örnek program C Ülkesinde tüm boya renkleri için planlanan toplam üretim hacmini Matris dizileri kullanarak hesaplar ve değeri bir değişkene atar.

```
uProductionToday := uProduction[2,0] + uProduction[2,1] +uProduction[2,2];
(* C ülkesinde bugün için tüm boya renklerinin toplam üretim hacmini hesaplar ve değeri
"uProductionToday" öğesine atar. *)
```

Hedef	A Ülkesi			B Ülkesi			C Ülkesi		
Boya rengi									
Üretim hacmi	10	5	20	15	45	20	25	30	10
	toplam 35			toplam 80			toplam 65		

Boya rengi (sütun)

Hedef (satır)	Kırmızı	Sarı	Mavi
A Ülkesi	[0,0] 10	[0,1] 5	[0,2] 20
B Ülkesi	[1,0] 15	[1,1] 45	[1,2] 20
C Ülkesi	[2,0] 25	[2,1] 30	[2,2] 10

6.2 Döngü (FOR)

Önceki sayfadaki örnek program (bugünün planlanan üretim hacmi atanır) aşağıda tekrar gösterilmektedir.

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

Bu program örneğinde boya renklerinin sayısı arttığında daha fazla değişken eklenecektir. Ardından, ifade daha uzun hale gelir ve okunması zorlaşır.

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

Bu durumda, daha net bir kod oluşturmak için döngü ifadeleri kullanılabilir.

Döngü ifadelerine FOR, WHILE ve REPEAT ifadeleri dahildir. Bu kurs, FOR ifadelerini kapsar.

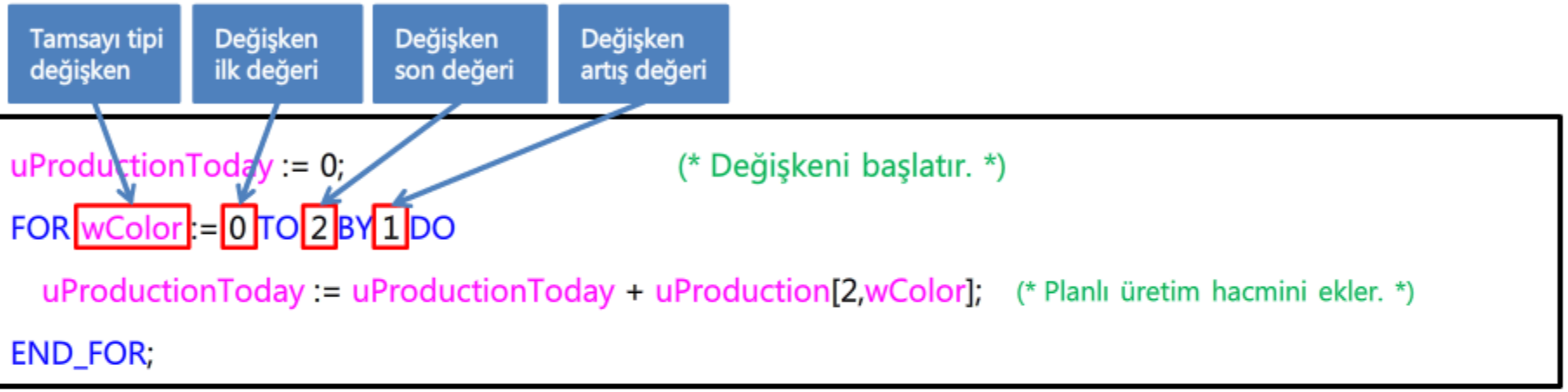
FOR ifadeleri aşağıdaki gibi açıklanır.

```
FOR variable := initial value TO final value BY increments DO
   Execution statement; (* Değişken son değere ulaşana kadar ifade döngüde yürütülür. *)
END_FOR;                (* END_FOR; FOR ifadelerinin sonuna yerleştirilmelidir.  *)
```

Değişkenin son değerine ulaşılana ve "END_FOR;" kodu yürütülene kadar ifade tekrarlanır.

6.2 Döngü (FOR)

Aşağıdaki örnek programda bir FOR ifadesi kullanılarak C Ülkesindeki tüm boya renklerinin planlı üretim hacmi elde edilir.



FOR ifadesi kullanılarak "wColor" değişkeni ilk değer olan sıfırdan başlayarak birer birer artar ve değişken, son değer ikiye ulaşana kadar ifade tekrarlanır.
"wColor" değişkeni, yürütme ifadesinde açıklanan "uProduction" dizisindeki ikinci eleman sayısı olarak belirtilir.
İfade her tekrarlandığında "wColor" değişkeninin değeri artar. Toplamı elde etmek için her boya renginin planlı üretim hacmi eklenir.

Bu örnek program üç defa bir döngüde yürütülür. (Birinci defa: kırmızı [0] => İkinci defa: sarı [1] => Üçüncü defa: mavi [2])

Bu programın operasyonu sonraki sayfada gösterilmektedir.

6.2 Döngü (FOR)

Program örneğinin operasyonu kullanılarak FOR ifadesinin yürütülüşü açıklanmıştır.

Tahmini üretim hacmi dizisi

	Kırmızı	Sarı	Mavi
A Ülkesi	[0,0] 10	[0,1] 5	[0,2] 20
B Ülkesi	[1,0] 15	[1,1] 45	[1,2] 20
C Ülkesi	[2,0] 25	[2,1] 30	[2,2] 10

Sonraki sayfaya geçmek için  üzerine tıklayın.
Animasyonu tekrar izlemek için "Oynat" düğmesine tıklayın.

Oynat

Number of repetition of the loop: 3

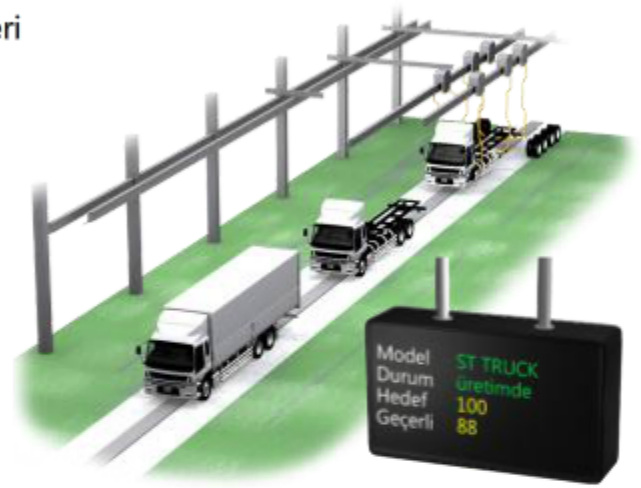
```
uProductionToday := 0;

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

6.3 İlgili veriyi saklama (Yapı)

Bir yapı, ilgili çoklu değişkenleri temsil etmek için bir değişken adının kullanılmasını sağlar.
Aşağıdaki örnekte Andon'da bir otomobil üretim hattının durumu gösterilmektedir (görüntü kartı).

Aşağıdaki tabloda, görüntülenen öğelerle ilgili değişken adları, değerler ve veri tipleri listelenmektedir.

Öğe	Değişken adı	Değer	Değişken Veri Tipi
Model	sModel	'ST TRUCK'	Metin dizesi
Durum	bStatus	'üretimde'	Bit tipi
Bugünün hedef üretim hacmi	uPlanQty	'100'	Tamsayı tipi Sözcük (imzasız)
Geçerli üretim hacmi	uActualQty	'88'	Tamsayı tipi Sözcük (imzasız)



Eğer bir yapı kullanılmazsa, çeşitli üretim hatları olduğunda değişken adları değiştirilmelidir.
Aşağıda, üretim hattına göre değişken adı örnekleri gösterilmektedir.

Birinci üretim hattı

s1stLineModel
b1stLineStatus
u1stLinePlanQty
u1stLineActualQty

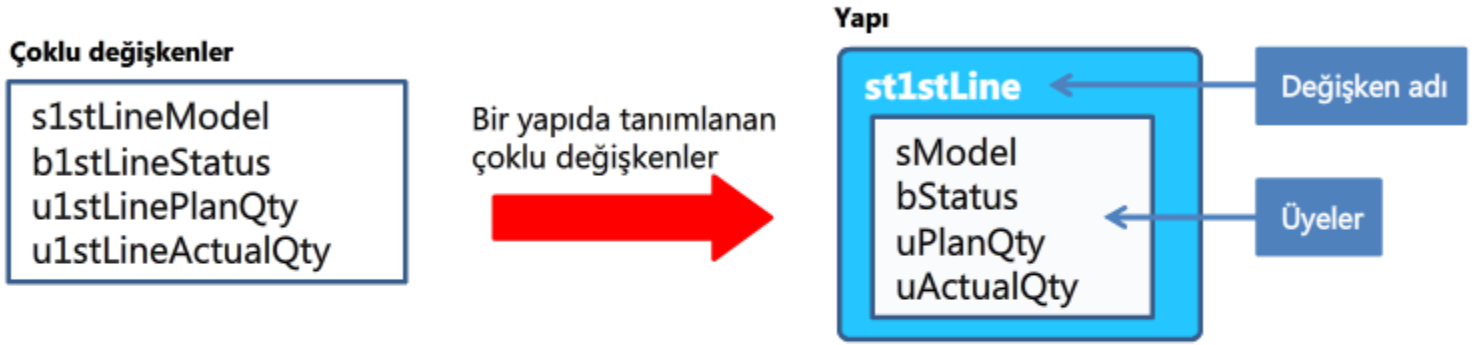
İkinci üretim hattı

s2ndLineModel
b2ndLineStatus
u2ndLinePlanQty
u2ndLineActualQty

Üretim hattı sayısı arttığında, işlenecek değişken sayısı da artacaktır. Ardından, program daha uzun hale gelir ve okunması zorlaşır.

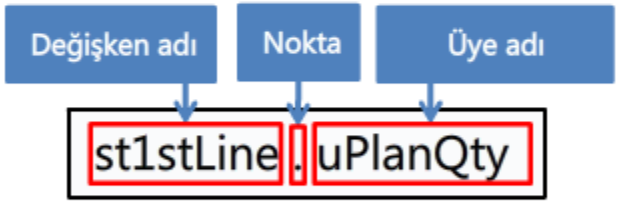
6.3 İlgili veriyi saklama (Yapı)

Yapıları kullanmak, bir üretim hattıyla ilgili çoklu değişkenlere bir değişken adı verilmesini sağlar.
Benzer şekilde yapılar veriyi aygıtlar, ekipman ve iş parçaları gibi fiziksel nesnelerin koşullarına ve özelliklerine göre seri halinde düzenlemek, saklamak ve işlemek için kullanılır.



Structure variable (yapı değişkeni), bunun bir yapı olduğunu temsil eden bir "st" ön ekini içerir.
Yapı tarafından tanımlanan bireysel değişkenler, üyeler olarak bilinir. Her üyenin veri tipleri farklı olabilir.

Yapı dizilerinin her üyesi, üye adının öncesinde bir nokta kullanılarak dizinin eleman sayısının ardından belirtilebilir.

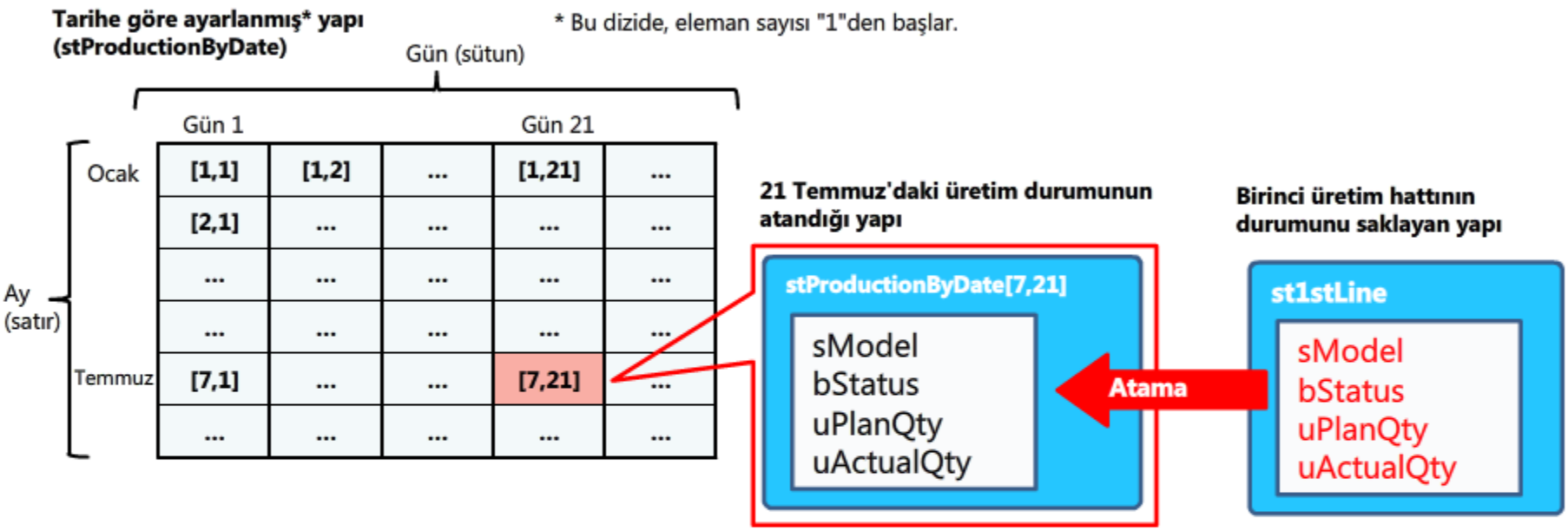


Aşağıdaki örnek programda sabit, ilk üretim hattındaki yapı değişkeninin bir üyesine atanmıştır.

```
st1stLine.uPlanQty := 150;
 (* İlk üretim hattında bugünün hedef üretimini 150'ye ayarlar. *)
```

6.3.1 Yapı dizilerini saklama

Yapılar, diziler olarak oluşturulabilir.
Aşağıdaki örnekte, üretim durumu tarihe göre saklanmıştır.

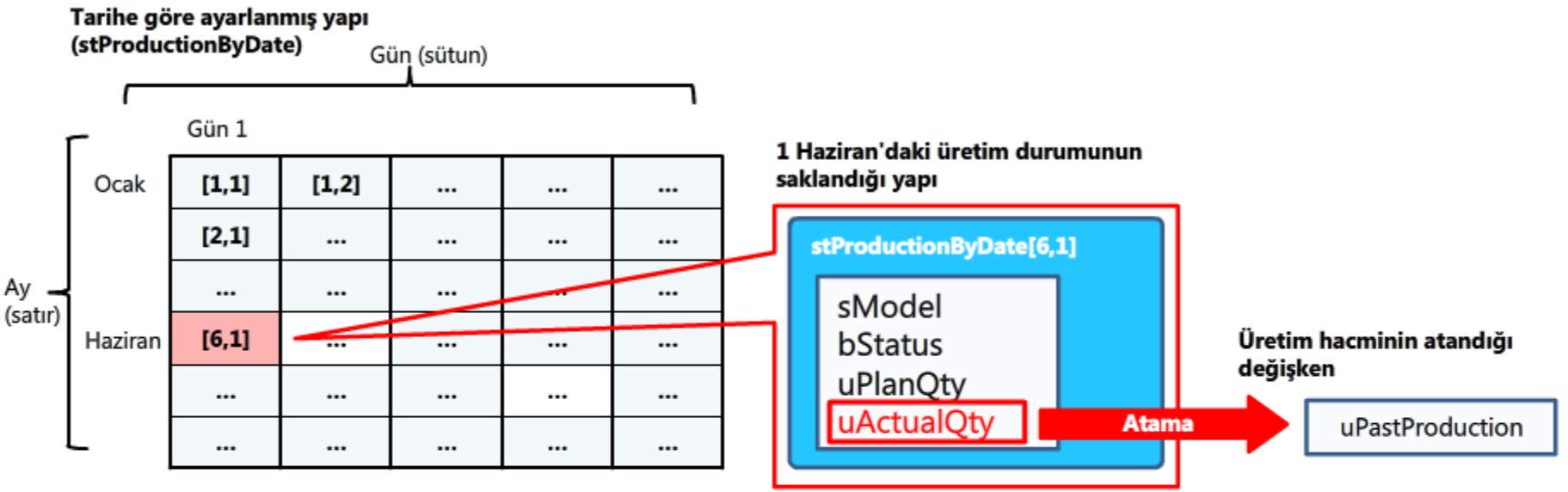



```
stProductionByDate[7,21] := st1stLine;
 (* 21 Temmuz'un üretim durumu, tarihe göre ayarlanmış yapıda saklanır
(stProductionByDate). *)
```

Benzer şekilde, üyelerin yapı ataması için bireysel olarak belirtilmesi gerekmez.

6.3.2 Yapı dizilerini okuma

Aşağıdaki örnekte, üretim hacmi tarihe göre düzenlenmiş ve sonra bir değişkene atanmış bir yapıdan okunur.




```
uPastProduction := stProductionByDate[6,1].uActualQty;
(* 1 Haziran'daki üretim hacmini uPastProduction değişkenine atar. *)
```

Yapı dizilerinin her üyesi, dizinin eleman sayısına bir nokta (.) ve bir üye adı ekleyerek belirtilebilir.

6.4 Özet

Bu bölümün içeriği şunlardır:

- Dizilerin genel görünümü ve kullanımı
- FOR ifadelerini kullanarak döngü işleme
- Yapıların genel görünümü ve kullanımı

Dikkat edilecek önemli noktalar:

Dizi	• Diziler kullanılarak çoklu değerler bir değişkenle işlenebilir. • Dizilerdeki bireysel değişkenler, değişken adlarının sonuna eklenen eleman sayılarıyla belirtilebilir.
FOR ifadesi	• Döngü ifadeleri tekrarlanan operasyonların istendiği programlarda kullanılır. • FOR ifadeleri, döngü operasyonunun sonundaki koşullar yerine getirilene kadar operasyonu tekrarlamak için kullanılır. "END_FOR;" ifadesinin öncesindeki ifade tekrarlanarak yürütülür.
Yapı	• Yapılar, ilgili çoklu değişkenleri temsil etmek için bir değişken adının kullanılmasını sağlar. Yapılara farklı veri tiplerinin değişkenleri dahil olabilir. • Yapılarda tanımlanan bireysel değişkenler veya üyeler, yapı değişken adından sonra bir nokta ve üye adı ekleyerek belirtilir.

Bölüm 7 Dize verisinin işlenmesi

Bazı durumlarda barkod okuyucular, sıcaklık kontrolörleri veya elektronik ölçekler gibi bağlı aygıtlara komut yollamak veya bunlardan geri bildirim almak için programlanabilir kontrolörlerde dize verisi kullanılır. Böyle amaçlar için dize verisini gerektiği gibi eklemek veya ayıklamak gereklidir.

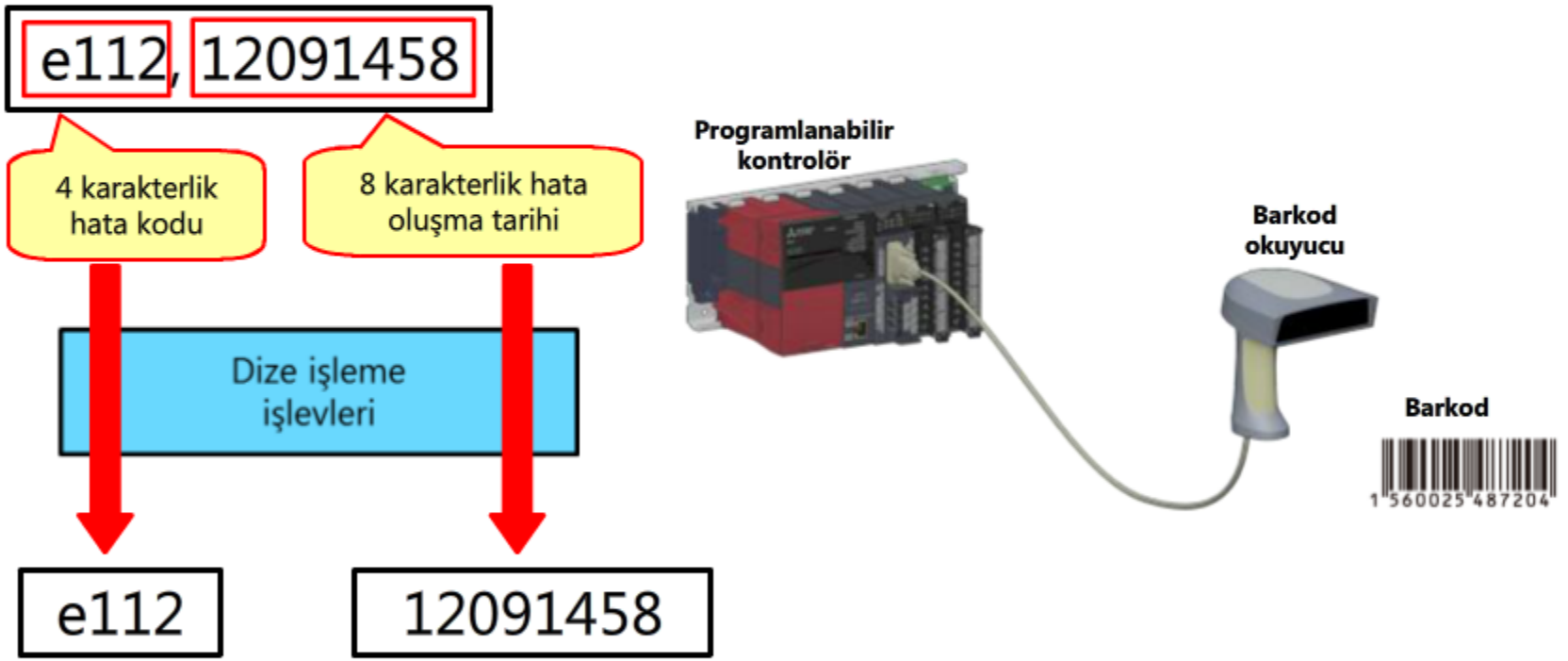
Bu bölümde dize verisinin nasıl işleneceği açıklanır.

7.1 Dize verisi işleme örneği

Dize işleme örneğinde, verinin bir barkod okuyucudan okunduğu bir senaryo gösterilmektedir.
İşlevler (yönerge tipi) dizeleri işlemek için kullanılır.

Aşağıda gösterildiği gibi, barkod okuyucu tarafından okunan dizelerde 4 karakterlik sabit uzunlukta hata kodu ve 8 karakterlik sabit uzunlukta ay, tarih, saat ve dakika verisi bulunur.
Dize işleme program örneği, bu sistem kullanılarak açıklanacaktır.

Barkod okuyucudan okunan örnek dize verisi



Bir hata kodu ayıklanır.
7.3 Dizeleri ayıklama (LEFT)

Hata oluşma tarihi ve zamanı (14:58, 9 Aralık) ayıklanır.
7.4 Dizeleri ayıklama (MID)

7.2 Dize ataması

Dizelerin nasıl ayıklanacağını açıklamadan önce bu bölümde dizelerin veri tipleri açıklanır.

Programlanabilir kontrolörlerle kullanılabilecek dizelerin veri tipleri aşağıdaki tabloda listelenmiştir.

Veri tipi	Karakter tipi işlenebilir	Macar notasyonu ön ekleri	Ön ekin uzatması
Dize	Alfasayısal karakter ve sayı dizeleri (ASCII) veya Japonca dizeler (Shift-JIS)	s	**s**tring (dize)
Dize [Unicode]	Farklı dillerin ve sembollerin dizeleri	ws	**w**ide **s**tring (geniş dize)

Kullanılacak dize tipi, programlanabilir kontrolöre veya ilgili dile bağlı aygıta göredir.
Bu bölümde farklı metin dizesi tipleri açıklanır.

Dize tipi bir dize, bir dize değişkenine atandığında, dizeyi tek tırnak işaretiyle kapsayın (').

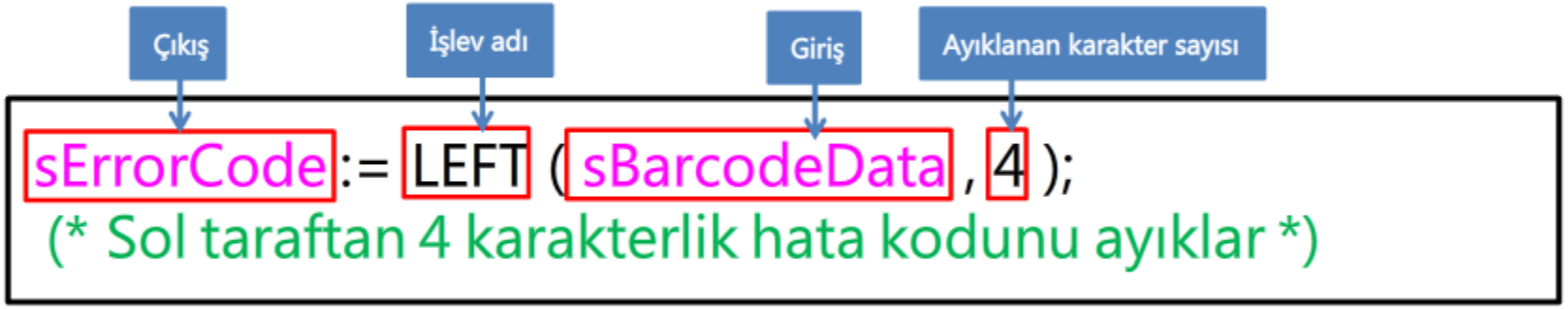
```
sDefault := 'e112,12091458';  (* Dize ataması *)
```

7.3 Dizeleri ayıklama (LEFT)

Hata kodu "e112", "e112,12091458" dizesini içeren "sBarcodeData" dize değişkeninden ayıklanır.

Değişken adı	Saklanan dize
sBarcodeData	e112, 12091458

LEFT işlevi giriş dizesinin sol tarafından başlayarak yalnızca belirtilen sayıdaki karakterleri ayıklar.
Aşağıda örnek bir program gösterilmektedir.



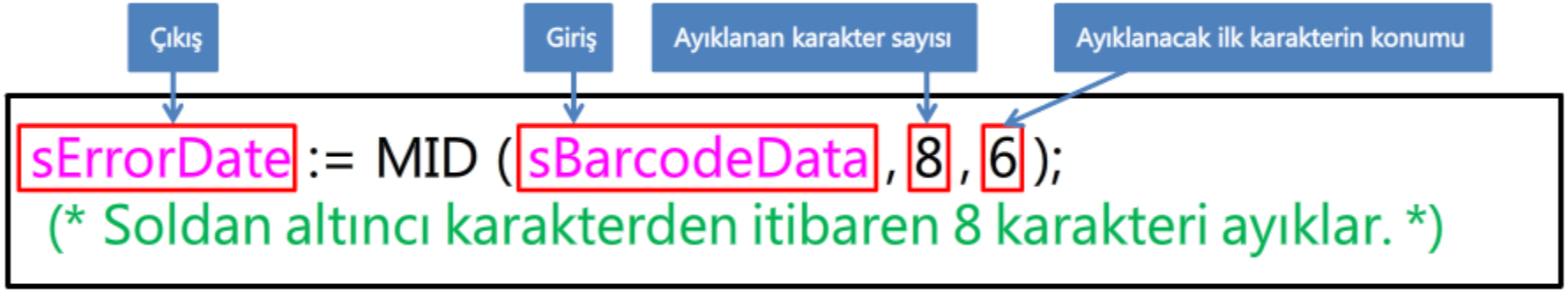
Soldan dört karakter ayıklanır. Hata kodunu temsil eden dizi olan "e112" değeri sol tarafa atanır.

7.4 Dizeleri ayıklama (MID)

Hata oluşturma zamanı "12091458", "e112,12091458" dizesini içeren "sBarcodeData" dize değişkeninden ayıklanır.

Değişken adı	Saklanan dize
sBarcodeData	e112,12091458

MID işlevi giriş dizesinin belirtilen başlangıç konumundan itibaren belirtilen sayıdaki karakterleri ayıklar.
Aşağıda örnek bir program gösterilmektedir.



Bu örnekte 8 karakterlik bir dize altıncı karakterden itibaren ayıklanır. Hata oluşma zamanını temsil eden dizi olan "12091458" değeri sol tarafa atanır.

7.5 Özet

Bu bölümün içeriği şunlardır:

- Dizeleri dize değişkenlerine atama yöntemleri
- Dizeleri ayıklayan işlevler (LEFT ve MID)

Dikkat edilecek önemli noktalar:

Dize ataması	• Bir dizeyi bir dize değişkenine atamak için, dizeyi tek tırnak işaretiyle kapsayın ('). • Programlanabilir kontrolöre veya ilgili dile bağlanan aygıta göre dize tipini veya dize [Unicode] tipini kullanın.
Dizeleri işlemek için işlevler	• Dizeleri işlemek için işlevler kullanılır.

7.6 Kurs özeti

Bu kursta, ST'de program oluşturmanın temelleri açıklanmıştır.
Böylece bu kursun sonuna geldik.

ST programları MELSOFT mühendislik yazılımı kullanılarak oluşturulur.
MELSOFT mühendislik yazılımıyla veri girme, düzenleme, kaydetme ve programları derleme gibi belirli adımların ayrıntıları için aşağıdakilere başvurun.

- Mitsubishi FA e-Learning Kursu "MELSOFT GX Works3 (Structured Text)" (MELSOFT GX Works3 (Yapılandırılmış Metin)) (yakında yayınlanacak)
- MELSOFT mühendislik yazılımınızın çalıştırma kılavuzu

ST hakkında daha fazla bilgi için aşağıdakilere başvurun.

- Programlanabilir kontrolörünüzün programlama el kitabı

Uygulamanızın yönergeleri ve işlevleri hakkında bilgi için aşağıdakilere başvurun.

- Programlanabilir kontrolörünüzün programlama kılavuzu

Test — Son Test

Artık Programlama Temelleri (Yapılandırılmış Metin) kursundaki tüm dersleri tamamladığınızdan, son teste girmeye hazırsınız. Ele alınan konulardan herhangi birini tam anlamadıysanız, lütfen bu konuları gözden geçirmek için bu fırsatı değerlendirin.
Bu Son Testte toplam 12 soru (20 madde) yer almaktadır.
Son testi istediğiniz sayıda uygulayabilirsiniz.

Testin puanlanması
Cevabı seçtikten sonra, **Cevapla** düğmesini tıkladığınızdan emin olun. Cevapla düğmesini tıklamadan ilerlemeniz durumunda cevabınız kaybolur. (Cevaplanmamış soru olarak değerlendirilir.)

Puan sonuçları
Doğru cevap sayısı, soru sayısı, doğru cevapların yüzdesi ve başarılı/başarısız sonucu puan sayfasında görüntülenir.

Doğru cevaplar: 5

Toplam soru: 5

Yüzde: 100%

Testi geçebilmek için, soruların **%60'ını** doğru cevaplamanız gerekir.

Devam Et | İncele

- Testten çıkmak için **Devam Et** düğmesini tıklayın.
- Testi incelemek için **İncele** düğmesini tıklayın. (Doğru cevap kontrolü)
- Testi tekrar yapmak için **Tekrar Dene** düğmesini tıklayın.

Test Son Test 1

Yapılandırılmış metnin (ST) özellikleri
ST'nin yanlış açıklamasını seçin.

- C veya BASIC dilinde program yazma deneyimi olanlar ST dilini kolayca öğrenebilir.
- Toplama ve çıkarma gibi hesaplamalar genellikle yaygın matematik ifadeleri kullanılarak yazılabilir.
- Kontaklar ve bobinlerin sembolleri, bir elektrik devresine benzeyen bir program oluşturmak için kullanılır.
- ST, veri işleme için uygundur.

Cevapla Geri

Test Son Test 2

ST'nin temel ilkeleri
ST'de yazılmış doğru ifadeyi seçin.

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

Cevapla Geri

Test Son Test 3

Yorumları açıklama
ST'de yazılmış doğru yorumu seçin.

- ' Değişkene 1 değerini atar.
- (* Değişkene 1 değerini atar. *)
- { Değişkene 1 değerini atar. }
- <!-- Değişkene 1 değerini atar. -->

Cevapla Geri

Test — Son Test 4

ST program yürütme sıralaması
*"uTotalProduction" başlangıç değeri "100"dür. Aşağıdaki örnek program işlendikten sonra "uTotalProduction" değişkeni değeri "101" olacaktır. Birkaç saniye geçtikten sonra doğru "uTotalProduction" durumunu seçin.

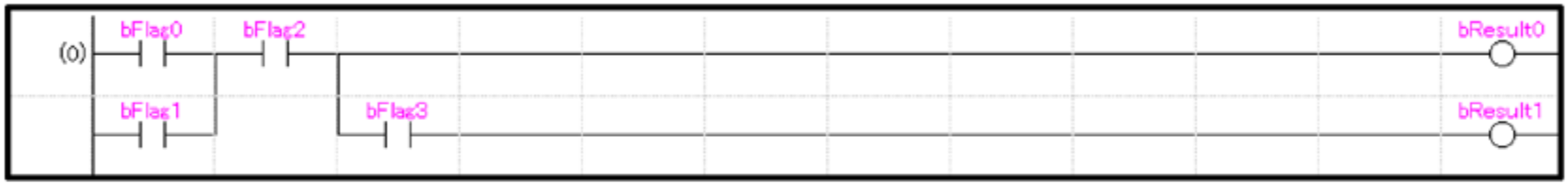
```
uTotalProduction := uTotalProduction + 1;
```

- ○ Değer 101'de kalır.
- ○ Değer değişmeye devam eder.

Cevapla Geri

Test — Son Test 5

Çoklu koşulları birleştirme
Aşağıda LD'deki program örneğindeki aynı operasyonu temsil eden doğru ST program örneğini seçin.



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

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

Cevapla Geri

Test — Son Test 6

ST'de IF ifadelerinin açıklaması
Aşağıdaki operasyon, aşağıdaki örnek program tarafından yürütülür.
- Eğer sıcaklık 5 dereceye veya daha aşağı inerse, ısıtıcı açılır ve soğutucu kapatılır.
- Eğer sıcaklık 50 dereceyi aşarsa, ısıtıcı kapatılır ve soğutucu açılır.
- Eğer sıcaklık yukarıdaki ifadeler için geçerli değilse hem ısıtıcı hem de soğutucu kapatılır.

*Değişken adları: Sıcaklık (wTemperature), ısıtıcı (bHeater) ve soğutucu (bCooler)
Örnek programın her boş kısmı için doğru seçimi yapın.

```
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;
```

S1 --Select--

S2 --Select--

S3 --Select--

S4 --Select--

S5 --Select--

Cevapla Geri

Test — Son Test 7

CASE ifadeleri

Aşağıdaki CASE ifadesi açıklamalarına göre her biri (S1'den S5'e kadar) için doğru seçimi yapın.

CASE ifadeleri (S1) değerine göre dallanma için kullanılır.
Aşağıdaki örnek programda (S2) değeri 25 olduğunda, (S3) değişkeni (S4) değerine atanır. (S2) değişkeninin değeri 10, 25 veya 8'e eşit olmadığında (S3) değişkeni (S5) değerine atanır.

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

S1 --Select--

S2 --Select--

S3 --Select--

S4 --Select--

S5 --Select--

Cevapla Geri

Test — Son Test 8

ST dizileri ve tekrarlanan ifadeler
Aşağıdaki örnek program, Y Ülkesine gidecek tüm modellerin planlanan üretim hacminin toplamını alır ve bu değeri bir değişkene atar. Dizinin, FOR ifadesinin ardından okunan ve döngüde 3 defa yürütülen kısmını seçin.

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

Model ve hedef bazında tahmini üretilen birim sayısını saklamak için kullanılan dizi (uProduction)

Hedef (satır)	Model (sütun) Model 1	Model 2	Model 3	Model 4
X Ülkesi	[0,0]	[0,1]	[0,2] C	[0,3]
Y Ülkesi	[1.0]	[1.1] A	[1.2] D	[1.3] E
Z Ülkesi	[2,0]	[2,1] B	[2,2]	[2,3]

- A
- B
- C
- D

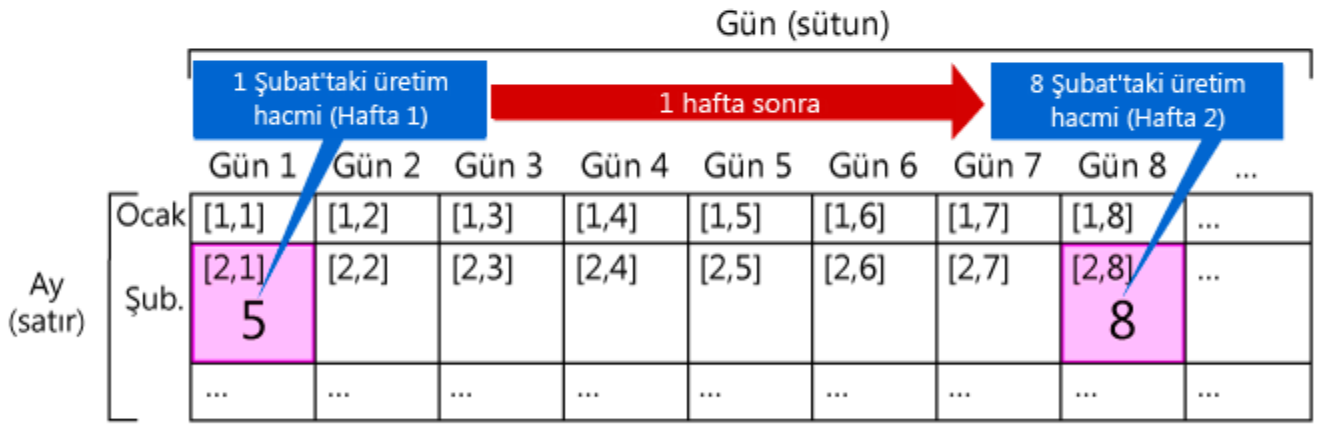
Cevapla Geri

Test — Son Test 9

ST dizileri ve tekrarlanan ifadeler
Aşağıdaki örnek program, haftanın aynı günlerindeki toplam üretim hacmini alır. 4 haftanın toplamı, günlük üretim hacmini saklayan diziden alınır. Örnek program için doğru rakamı seçin.

```
uTotalProduction := 0;
FOR wOnceAWeek := 1 TO ■ BY 7 DO
  uTotalProduction := uTotalProduction + uProductionByDate[2,wOnceAWeek];
END_FOR;
(* 1 Şubat'tan başlayarak 4 haftanın aynı günlerindeki üretim hacmini ayıklar ve toplamını alır. *)
```

Günlük üretim hacmini saklayan dizi (uProductionByDate)



- 22
- 21
- 4
- 28

Cevapla | Geri

Test Son Test 10

ST'deki yapıların özellikleri
Yapıların yanlış açıklamasını seçin.

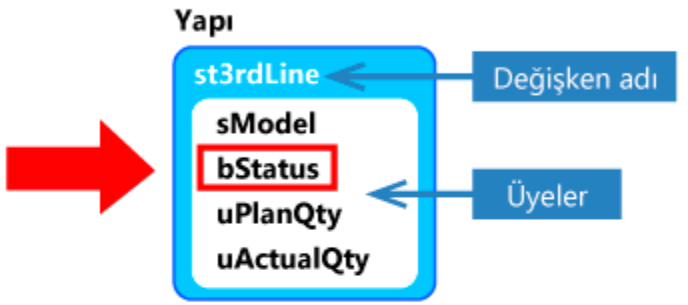
- Yapılar; durum ve özellikler gibi koşullara göre veriyi aygıtlarda düzenlemek ve saklamak için kullanılır.
- Büyük miktarda veri işleyen programlar, yapılar kullanılarak kısa bir şekilde yazılabilir.
- Yapıda tanımlanan tüm üyelerin aynı veri tipi olmalıdır.
- Değerler, bireysel olarak belirtilmeden aynı yapıdaki üyelere atanabilir.

Cevapla Geri

Test — Son Test 11

ST'deki yapılar için üyeleri belirtmek
Aşağıdaki yapı, bir otomobil üretim hattıyla ilgili değişkenleri düzenler.
Bu yapıda "bStatus" üyesini belirtmek için doğru açıklamayı seçin.

Parametre	Değişken adı
Model	sModel
Durum	bStatus
Geçerli günün hedef üretimi	uPlanQty
Geçerli üretim sayısı	uActualQty

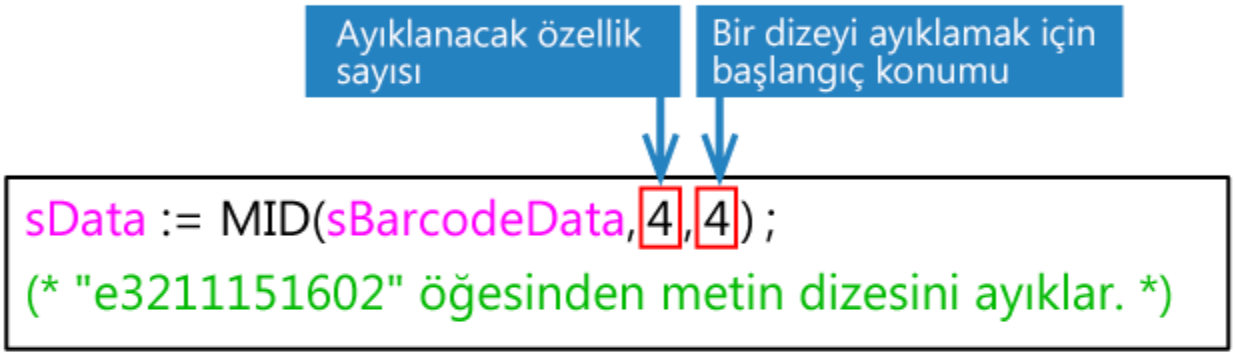


- st3rdLine.bStatus
- st3rdLine->bStatus
- st3rdLine[bStatus]
- st3rdLine[1]

Cevapla Geri

Test — Son Test 12

ST'de dizeleri işleme
Aşağıdaki örnek program, "sBarcodeData" değişkeninde saklanan "e3211151602" dizesinden belirli bir dizeyi ayıklar.
MID işlevi, belirtilen başlangıç konumundan itibaren belirtilen sayıdaki karakterleri ayıklar.
Ayıklanan dizeyi doğru olarak seçin.



- 1151
- 1602
- e321
- 1115

Cevapla | Geri

Test Puanı

Son Testi tamamladınız. Sonuç alanınız aşağıda gösterildiği gibidir.
Son Testi sonlandırmak için bir sonraki sayfaya ilerleyin.

Doğru cevaplar:	12
Toplam soru:	12
Yüzde:	100%

Devam Et | İncele

Tebrikler. Testi geçtiniz.

Programlama Temelleri (Yapılandırılmış Metin) kursunu tamamladınız.

Bu kursa katıldığınız için teşekkür ederiz.

Derslerden keyif almış olmanızı ve bu kursta edindiğiniz bilgilerin gelecekte faydalı olmasını umarız.

Kursu istediğiniz kadar çok gözden geçirebilirsiniz.

İncele Kapat