MITSUBISHI ELECTRIC
Changes for the Better

A PLC
programozás alapjai (strukturált szöveg)

A tanfolyam a MELSEC programozható vezérlőknél használt alapvető programok létrehozását tárgyalja.
A strukturált szöveget (ST) ebben a tanfolyamban programleírásra használjuk.

L(FOD)00546HUN

Bevezetés A tanfolyam célja

Ez a tanfolyam a MELSEC programozható vezérlők strukturált szövegű (ST) vezérlőprogramjainak létrehozását mutatja be.

A következő tanfolyam elvégzése vagy azzal egyenértékű ismeret megléte a tanfolyam megkezdésének követelménye:

Programming Basic (Programozás alapjai)

A C vagy BASIC programnyelvekkel kapcsolatos ismeret vagy jártasság segíthet a tanfolyam tartalmának könnyebb megértésében.

Bevezetés A tanfolyam felépítése

Itt találja a tananyagban foglalt témaköröket.

1. fejezet – A strukturált szöveg áttekintése

Ebben a fejezetben a strukturált szöveg (ST) jellemzői és megfelelő alkalmazásai kerülnek bemutatásra.

2. fejezet – Az ST-programok alapvető szabályai

Ebben a fejezetben az ST-nyelven létrehozott programok alapvető szabályait mutatjuk be.

3. fejezet – I/O-vezérlőprogramok létrehozása

Ebben a fejezetben bemutatjuk, hogyan hozhatók létre I/O-vezérlő programok.

4. fejezet – Aritmetikai műveletek

Ebben a fejezetben az aritmetikai műveletek létrehozásának módját mutatjuk be.

5. fejezet – Feltételes elágazások

Ebben a fejezetben a feltételes elágazásokat mutatjuk be.

6. fejezet – Tárolás és adatkezelés

Ebben a fejezetben bemutatjuk, hogy az adatok tárolására és kezelésére hogyan írhatók tömör programok.

7. fejezet – Karakterlánc típusú adatok kezelése

Ebben a fejezetben a karakterlánc típusú adatok kezelési módjait tárgyaljuk.

Záró teszt

Ponthatár: 60% vagy afölött

Bevezetés Hogyan használjuk ezt az e-learning eszközt

Tovább a következő oldalra		Tovább a következő oldalra.
Vissza az előző oldalra		Vissza az előző oldalra.
Ugrás a kívánt oldalra		Megjelenik a „Tartalomjegyzék”, ahol lehetőség van a kívánt oldal elérésére.
Kilépés a kurzusból		Kilépés a kurzusból.

Bevezetés

A használatra vonatkozó óvintézkedések

Biztonság

Amikor a tényleges termékek használatával tanul, figyelmesen olvassa el a biztonsági óvintézkedéseket a vonatkozó kézikönyvben.

Megjegyzések a tananyag tartalmával kapcsolatban

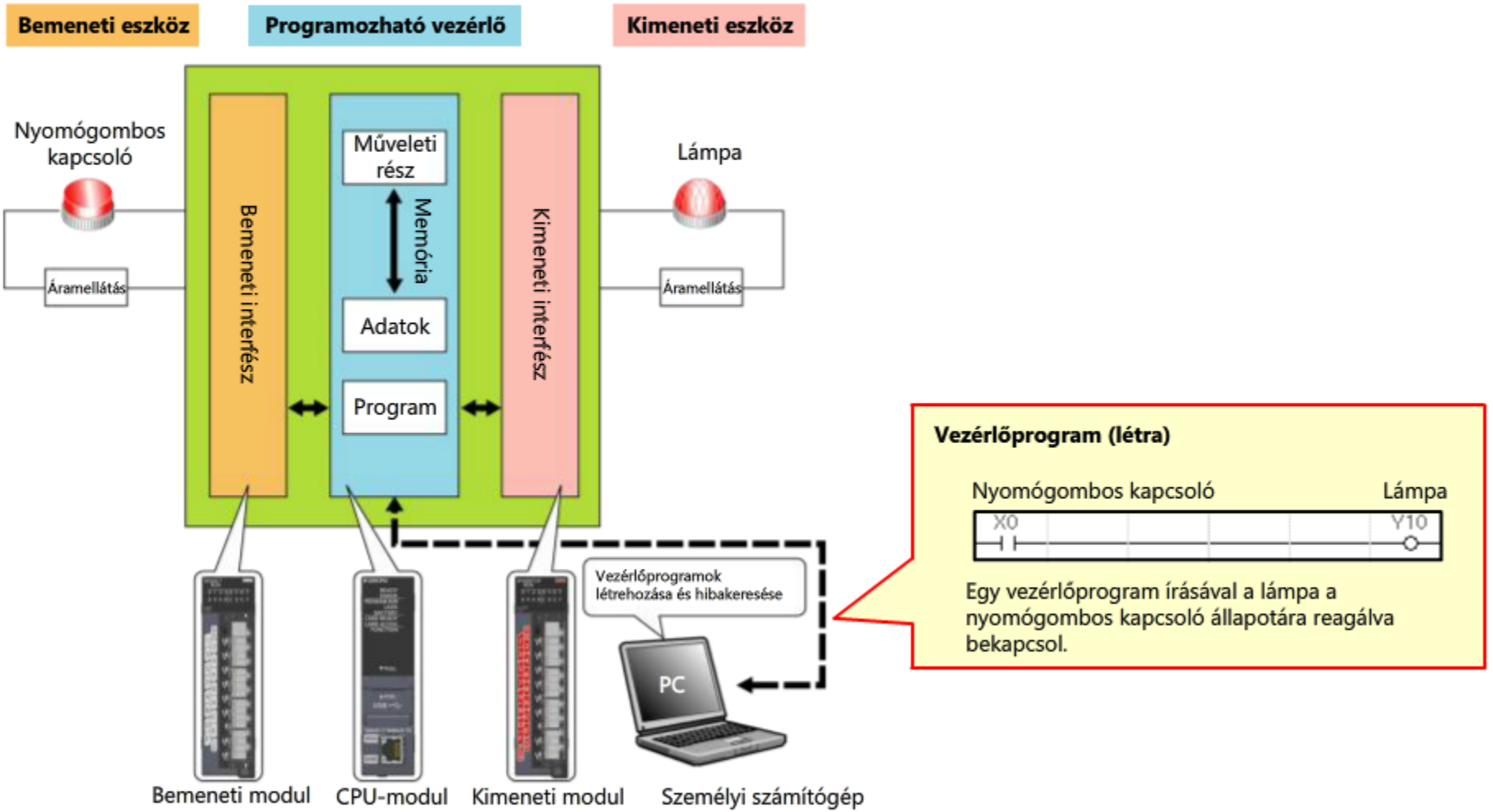
Az Ön által használt MELSOFT tervezői szoftverben megjelenő képernyők eltérhetnek a tananyagban szereplőktől.
A tanfolyam a programok létrehozásához a MELSOFT GX Works3 létraszimbólumait használja.

1. fejezet A strukturált szöveg áttekintése

Ebben a fejezetben a strukturált szöveg (ST) jellemzői és megfelelő alkalmazásai kerülnek bemutatásra.

1.1 Vezérlőprogramok

Az alábbi ábrán egy programozható vezérlőrendszer konfigurációja látható.
A programozható vezérlők a vezérlőprogramoknak megfelelően működnek.
A programozható vezérlők működése a leírtaknak megfelelően vezérlőprogramok létrehozásával konfigurálható.



A programozható vezérlők programnyelveit az International Electrotechnical Commission (IEC) (Nemzetközi Elektrotechnikai Bizottság (IEC)) által kifejlesztett nemzetközi szabvány határozza meg.

1.2 Az ST jellemzői és összehasonlítása más IEC-programnyelvekkel

Az IEC 61131 egy, a programozható vezérlőrendszerekre vonatkozó nemzetközi szabvány.
A programozható vezérlők programnyelveit az IEC 61131-3 szabványosítja. Az ST a standard programnyelvek közé tartozik.
Az egyes nyelvek különböző jellemzőkkel rendelkeznek, melyek az alkalmazásnak és a programozói készségnek megfelelően választhatók ki.

Az alábbi táblázatban az IEC 61131-3 szerinti programnyelvek jellemzőinek listája látható.

Programnyelv	Jellemzők
Ladder **D**iagram (LD) (Létradiagram (LD))	• Az érintkezők és tekercsek szimbólumait használva egy elektromos áramkörre hasonlító program készíthető. • A program folyamata könnyen követhető és megérthető, még kezdők számára is.
Structured **T**ext (ST) (Strukturált szöveg (ST))	• A programok írása szövegesen (karakterekkel) történik. • Az ST a C vagy a BASIC nyelvű programozásban jártasok számára könnyen elsajátítható. • A számítási képletek matematikai kifejezésekre hasonlítanak, így könnyen megérthetők. • Az ST adatkezelésre alkalmas.
Function **B**lock **D**iagram (FBD) (Funkcióblokk-diagram (FBD))	• Programok a különböző funkciójú blokkok elrendezésével és a köztük lévő kapcsolat megadásával írhatók. • Az FBD javítja az olvashatóságot, mivel a teljes műveletet könnyen átláthatóvá teszi.
Sequential **F**unction **C**hart (SFC) (Szekvenciális funkcióábra (SFC))	• A feltételek és műveletek írása folyamatábraként történik. • A program folyamata könnyen megérthető.
Instruction **L**ist (IL) (Utasítási lista (IL))	• Az IL a gépi nyelvre hasonlít. • Az IL manapság már alig használatos.

A tanfolyam bemutatja, hogy ST-ben hogyan írhatók alapvető vezérlőprogramok.

1.3 Összefoglalás

A fejezet tartalma:

- A programozható vezérlőrendszerek és a vezérlőprogramok kapcsolata
- A vezérlőprogramokra vonatkozó nemzetközi szabvány
- Az ST jellemzői

A következő fontos szempontokat kell figyelembe venni:

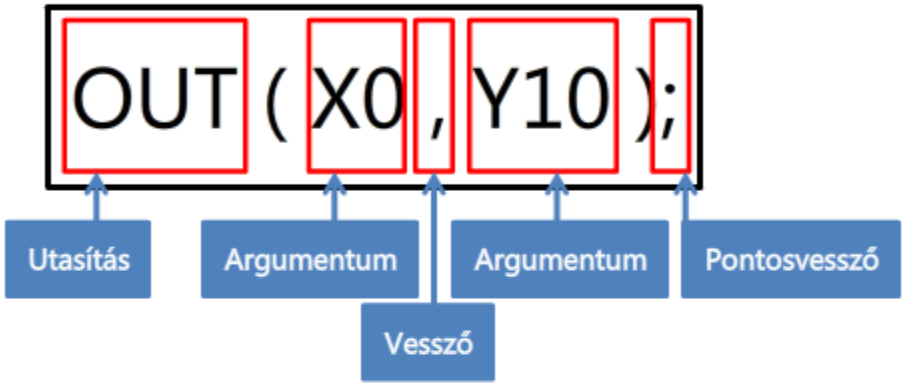
A programozható vezérlőrendszerek és a vezérlőprogramok kapcsolata	• A programozható vezérlők a vezérlőprogramoknak megfelelően működnek. • A programozható vezérlők működése a leírtaknak megfelelően vezérlőprogramok létrehozásával konfigurálható.
A vezérlőprogramokra vonatkozó nemzetközi szabvány	• Az ST az IEC-programnyelvek közé tartozik. • IEC-programnyelvek még továbbá az LD, az FBD, az SFC és az IL, melyek mindegyike különböző, az alkalmazáshoz és a programozói készséghez illeszkedő jellemzőkkel rendelkezik.
Az ST jellemzői	• Az ST könnyen elsajátítható azok számára, akik jártasak a C vagy a BASIC nyelvű programok írásában. • A számítások (például összeadás és kivonás) jellemzően használt matematikai kifejezésekként írhatók le, így könnyen megérthetők. • Az ST adatkezelésre alkalmas.

2. fejezet Az ST-programok alapvető szabályai

Ebben a fejezetben az ST-nyelven létrehozott programok alapvető szabályait mutatjuk be.

2.1 Egyszerű példaprogram (I/O-vezérlési utasítás)

Ez a rész egy egyszerű ST-programra mutat be egy példát.
Az alábbi példaprogramban ha az X0 bemenet bekapcsol, akkor az Y10 kimenet is bekapcsol, illetve ha az X0 kikapcsol, akkor az Y10 is kikapcsol.



Az utasítások határozzák meg a végrehajtandó műveletet.
Az argumentumok az utasítást követő zárójelek közé kerülnek.
Az argumentumok változókat, aritmetikai kifejezéseket és állandó értékeket írnak le.
A MELSEC programozható vezérlőkkel a CPU-modul eszközei mint változók használhatók.

Az argumentumok száma az utasítástól függ.
Az argumentumokat vesszővel (,) kell elválasztani.

A fent látható sor egy utasításnak felel meg. Az egyes utasítások végére pontosvesszőt (;) kell tenni.
A program írása utasítások kombinálásával történik.

2.2 Egyszerű példaprogram (hozzárendelési utasítás)

A következő példa egy olyan programot jelenít meg, amely hozzárendelési utasítást használ.
Az alábbi utasítás az „5” decimális állandót a „D10” változóhoz rendeli.

A hozzárendelési utasításhoz egy hozzárendelési operátort (:=) kell használni. Figyelje meg, hogy az egyenlőségjel (=) bal oldalán egy kettőspont (:) található.
A hozzárendelési utasításokkal a jobb oldali érték a bal oldalihoz rendelhető.

Megjegyzés hozzáfűzése a programhoz könnyebben érthetővé teszi a műveletet. A megjegyzéseket mindkét oldalon csillaggal (* *) zárja le.

2.3 Numerikus jelölés

Az előző oldalon szereplő példaprogrammal egy decimális értéket egy változóhoz rendeltünk.

A szekvenciális vezérléshez időnként a decimálistól eltérő, például bináris vagy hexadecimális értékeket használnak.
Az alábbi táblázatban a MELSEC programozható vezérlőknél az ST-nyelvben használt numerikus jelölési típusok láthatók.

Numerikus jelölés típusa	Jelölés módja	Példa
Bináris	„2#" előtag hozzáadása.	**2#**11010
Oktális	„8#" előtag hozzáadása.	**8#**32
Decimális	Közvetlen bemenet	26
	„K" előtag hozzáadása.	**K**26
Hexadecimális	„16#" előtag hozzáadása.	**16#**1A
	„H" előtag hozzáadása.	**H**1A

Az alább látható példaprogramok értékek változókhoz hozzárendelését végzik.

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

2.3.1 Bites jelölés

A bitek true/false (igaz/hamis) állapotot jelölnek, például a jelek on/off (be/ki) helyzetét, illetve az állapotok meglétét/hiányát.
ST-ben a bitek „ON" és „OFF" értékkel nem írhatók le. Ezeket az „1" (ON) és a „0" (OFF) értékek jelölik. A bitek „TRUE" és „FALSE" értékként is kifejezhetők.

Az alábbi táblázat a jelölések különböző típusait tartalmazza.

Állapot	ON	OFF
	True	False
Numerikus jelölés	1	0
True/false jelölés	TRUE	FALSE

Az alábbiakban néhány példa látható az értékek bittípusú változókhoz való hozzárendelésére.

Numerikus jelölés

```
X0 := 1;
```

=

True/false jelölés

```
X0 := TRUE;
```

Numerikus jelölés

```
X0 := 0;
```

=

True/false jelölés

```
X0 := FALSE;
```

2.4 Program végrehajtási szekvencia

Az ST-utasítások végrehajtása fentről lefelé haladó sorrendben történik.

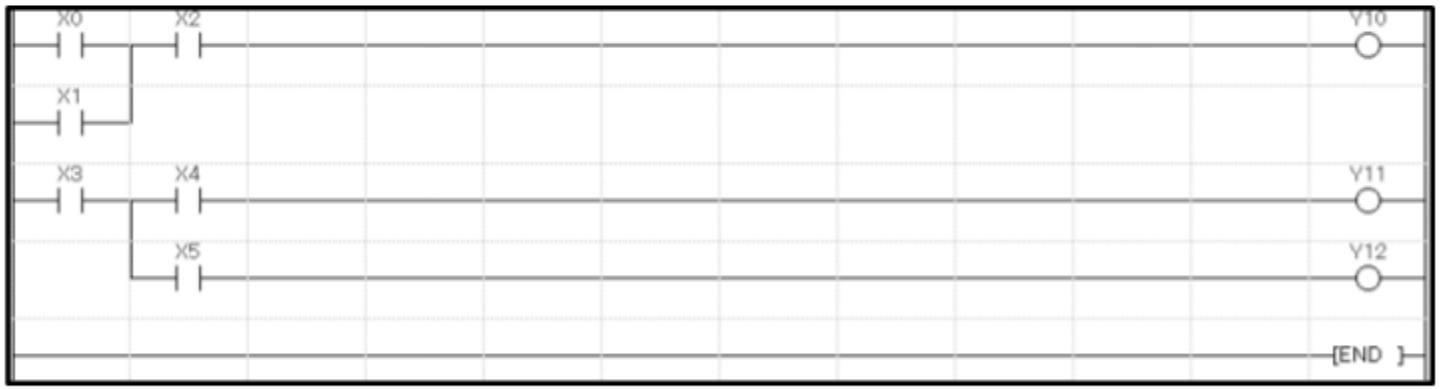
ST-példaprogram

```
Y10 := (X0 OR X1) AND X2;      (* Először végrehajtva *)
Y11 := X3 AND X4;              (* Másodjára végrehajtva *)
Y12 := X3 AND X5;              (* Harmadjára végrehajtva. A végén nincs szükség END utasításra.*)
```

Ismétlődő végrehajtás

* Bár LD-ben a program végén szerepelnie kell egy END utasításnak, erre ST-ben nincs szükség.

Az alábbi létraprogram ugyanannak a műveletnek felel meg, mint a fenti ST-példaprogram.



Ismétlődő végrehajtás

Az LD-hez hasonlóan az utasítások végrehajtása ST-ben is ismétlődik; az utolsó utasítás elérésével visszatér az elsőre.

2.5 Összefoglalás

A fejezet tartalma:

- Egyszerű ST-program
- Hozzárendelési utasítás formátuma
- Numerikus jelölés
- Program végrehajtási szekvencia
- Megjegyzés

A következő fontos szempontokat kell figyelembe venni:

Egyszerű ST-program	• Az utasítások az ST-programok legkisebb elemei. • Az egyes utasítások végére pontosvesszőt (;) kell tenni. • A program írása utasítások kombinálásával történik.
Hozzárendelési utasítás formátuma	• A hozzárendeléshez hozzárendelési operátort (:=) kell használni.
Numerikus jelölés	• Numerikus jelölés típusai ST-ben • ST-ben a bitek értékét „ON" és „OFF" jelölések helyett „1" és „0" jelöli. • A bitértékek ST-ben „TRUE" és „FALSE" alakban is megadhatók.
Program végrehajtási szekvencia	• Az ST-ben létrehozott programok végrehajtása fentről lefelé haladó sorrendben történik. • Az LD-programokhoz hasonlóan a program folyamata, amint eléri a program végét, ismétlődően visszatér a program elejére.
Megjegyzés	• Megjegyzés hozzáfűzése a programhoz könnyebben érthetővé teszi a műveletet. • A megjegyzéseket csillagok (* *) fogják közre.

3. fejezet I/O-vezérlő programok létrehozása

Ez a fejezet bemutatja az I/O-vezérlő programok létrehozásának módját ST-ben.

3.1 I/O-vezérlő programok

Az alábbiakban egy programozható vezérlő I/O-vezérlési példaprogramja látható.

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

- OUT: Kimeneti utasítás
- X0: Végrehajtási feltétel (argumentum)
- Y10: Kimeneti eszköz (argumentum)

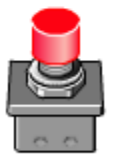
Az „OUT" a kimeneti utasítás. Az argumentumok végrehajtási feltételeket határoznak meg, valamint azt az eszközt, amelyre a kimenet irányul.
Ha az X0 végrehajtási feltétel teljesül, az Y10 eszköz bekapcsol.

Kattintson a lent látható bemeneti gombra. Az X0 bemeneti kapcsoló bekapcsol.

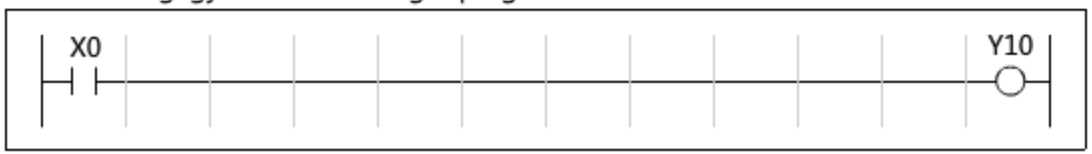
- Az X0 bemeneti kapcsoló bekapcsolásával az Y10 kimeneti lámpa is bekapcsol.
- Az X0 bemeneti kapcsoló kikapcsolásával az Y10 kimeneti lámpa is kikapcsol.

ST-ben megírt I/O-vezérlő példaprogram

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



Az ezzel megegyező LD-ben megírt program



3.1 I/O-vezérlő programok

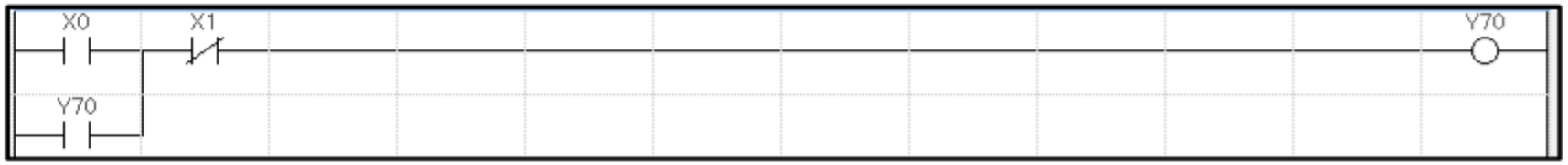
Az LD-hez hasonlóan az OUT mellett számos más utasítás is elérhető, mint például az I/O-vezérlő utasítások és az adatfeldolgozási utasítások.
További információkat az ST-ben elérhető utasításokról programozható vezérlőjének programozási kézikönyvében talál.

Figyelje meg, hogy az „OUT(X0, Y10);" sort „Y10 := X0;" formában leírva ugyanaz a művelet történik.

```
Y10 := X0;  (* Ugyanaz a művelet, mint az „OUT(X0, Y10);" esetén. *)
```

3.2 Több feltétel kombinációja

Az alábbi létraprogram egy öntartó áramkört ábrázol.



Ugyanez a program ST-ben az alábbi módon írható le.

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

Logikai operátor

Ahogy az fent látható, a logikai operátorok ST-ben több feltételből álló kombinációk létrehozására alkalmazhatók.

Az alábbi táblázat a logikai operátorokat tartalmazza.

Operátor	Jelentés
OR	Logikai VAGY
AND	Logikai ÉS
NOT	Logikai tagadás
XOR	Kizáró VAGY

3.3 A változók jelentésének definiálása

Az ST-nyelvet MELSEC programozható vezérlővel használva az eszközök és a címkék egymással felcserélhetően rendelhetőek változókhoz.
A felhasználók az alkalmazásnak megfelelően címkéket is használhatnak.

Egy, az alkalmazáshoz kapcsolódó címke kiosztásával a művelet könnyebben megérthetővé válik.

```
Y10 := (X0 OR X1) AND X2;     (* Eszköznevekkel megírva. *)
```

```
Lamp := (Switch0 OR Switch1) AND Switch2;     (* Címkékkel megírva. *)
```

A címkék a MELSOFT tervezői szoftver segítségével nevezhetők el.

A tanfolyam további részében a példaprogramok kifejtése címkék segítségével fog történni.

3.4 Összefoglalás

A fejezet tartalma:

I/O-vezérlő példaprogramok

- A logikai operátorok ST-ben több feltételből álló kombinációk létrehozására alkalmazhatók.
- Az eszköznevek és a címkék változók neveiként használhatók.

A következő fontos szempontokat kell figyelembe venni:

Több feltétel kombinációja	• A logikai operátorok ST-ben a feltételek kombinációinak létrehozására alkalmazhatók.
A változók jelentésének definiálása	• Egy, az alkalmazáshoz kapcsolódó címke kiosztásával a művelet könnyebben megérthetővé válik.

4. fejezet Aritmetikai műveletek

Ebben a fejezetben az aritmetikai műveletek létrehozásának módját mutatjuk be.

- Az aritmetikai műveletek bemutatása
- A numerikus tartományoknak megfelelő adattípusok meghatározása
- Változók elnevezése az adattípus-inkonzisztencia elkerülése érdekében

4.1 Az alapvető aritmetikai műveletek

Ez a példaprogram két külön gyártósor legyártott mennyiségét összegzi.
Az egyenlet jobb oldala aritmetikai művelet, benne változókkal és aritmetikai operátorokkal.

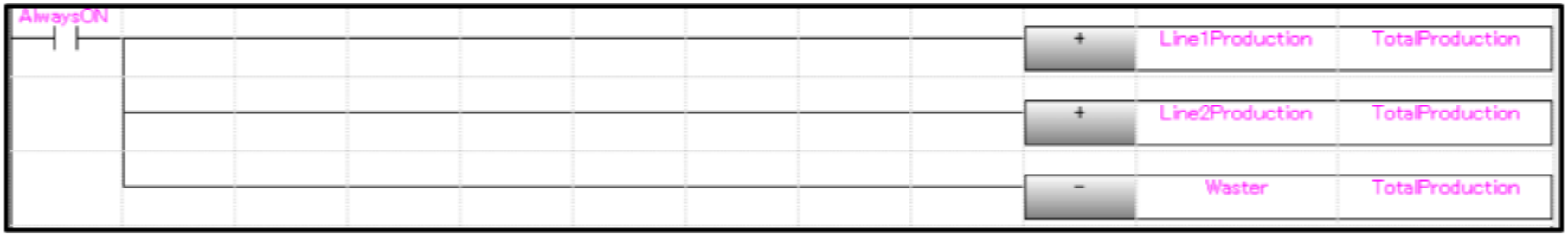
ST-ben megírt aritmetikai példaprogram

Összeadás operátor

Kivonás operátor

```
TotalProduction := Line1Production + Line2Production - Waster;
(* Összegzi a két gyártósor termelési mennyiségét, az összegből levonja a hibás termékek számát, és hozzárendeli a kapott értéket. *)
```

Alább ugyanez a program látható LD-ben megírva.



Amint az fent is látható, a program megírásához létrában 3 sort kell használni, ST-ben azonban 1 sor is elég.

Az alábbi táblázat az alapvető aritmetikai operátorokat tartalmazza.

Operátor	Jelentés
+	Összeadás
-	Kivonás
*	Szorzás
/	Osztás

4.2 Változói adattípusok

A kezelendő értéktartomány meghatározásához minden egyes változóhoz meg kell adni egy adattípust.
Az ST-ben használt numerikus értékű adattípusok a bit, az integer és a valós szám típusok.

Az alábbi táblázatban az ST-ben használt adattípusok közül az ezen a tanfolyamon használt adattípusok láthatók.

Adattípus		Adattartomány
Bit		Biteszközök BE/KI állapota és végrehajtási eredmények igaz/hamis állapota
Integer	Szó (előjel nélküli)	0–65 535
	Szó (előjeles)	−32 768–32 767
	Duplaszó (előjel nélküli)	0–4 294 967 295
	Duplaszó (előjeles)	−2 147 483 648–2 147 483 647

Az integer típus használatakor a szó vagy duplaszó típust az adattartománynak megfelelően válassza ki, illetve az előjeles vagy előjel nélküli típust attól függően, hogy szükség van-e negatív számok kezelésére.
A változók adattípusát akkor határozza meg, ha a címkenév a MELSOFT tervezői szoftverben be van állítva.

4.3 Adattípust megjelenítő változónevek

Egy hozzárendelési egyenlet bal és jobb oldalán eltérő adattípusok használata fordítási hibához vagy nem várt eredményhez vezethet.
Ilyen esetre láthatunk lent egy példát.

```
ValueA := ValueB;       (* ValueA: szavas integer, ValueB: dupla szavas integer *)
```

Dupla szavas integer nem rendelhető hozzá egy szavas integerhez. Ebben az esetben azonban az adattípus nem ismerhető fel.

A változónévhez az adattípust megjelenítő előtagok rendelhetők, amivel az adattípusok vizuálisan azonosíthatóvá válnak.
A változók ilyen típusú elnevezése magyar jelölésként ismert.

Adattípus		Adattartomány	Előtag	Előtag kifejtése
Bit		Biteszközök BE/KI állapota és végrehajtási eredmények igaz/hamis állapota	b	**B**it (bit)
Integer	Szó (előjel nélküli)	0–65 535	u	**u**nsigned word (előjel nélküli szó)
	Szó (előjeles)	–32 768–32 767	w	signed **w**ord (előjeles szó)
	Duplaszó (előjel nélküli)	0–4 294 967 295	ud	**u**nsigned **d**ouble-word (előjel nélküli duplaszó)
	Duplaszó (előjeles)	–2 147 483 648–2 147 483 647	d	signed **d**ouble-word (előjeles duplaszó)

Az oldal tetején feltüntetett példaprogram magyar jelöléssel az alábbiak szerint írható le:

```
wValueA := dValueB; (* Dupla szavas változó nem rendelhető hozzá egy szavas változóhoz. *)
```

Magyar jelölés használatával a program írása közben azonosíthatók az adattípusokban jelentkező inkonzisztenciák.

A tanfolyam hátralévő részében a példákban szereplő változónevek magyar jelölésűek lesznek.

4.4 Összefoglalás

A fejezet tartalma:

- Az aritmetikai műveletek bemutatása
- A numerikus tartományoknak megfelelő adattípusok meghatározása
- Adattípust megjelenítő változónevek hozzáadása

A következő fontos szempontokat kell figyelembe venni:

Az alapvető aritmetikai műveletek	• Számítások kifejezésére az ST-ben az általános programnyelvekkel közös operátorok használhatók.
Változói adattípusok	• A kezelendő értéktartomány meghatározásához minden egyes változóhoz meg kell adni egy adattípust.
Adattípust megjelenítő változónevek hozzáadása	• A változónevek magyar jelöléssel történő leírása programírás közben lehetővé teszi a változók adattípusában jelentkező inkonzisztenciák azonosítását.

5. fejezet Feltételes elágazások

A vezérlőprogramok olyan kódrészeket is tartalmaznak, amelyekben a tényleges feldolgozás a meghatározott feltételek függvényében változik.
Ebben a fejezetben a feltételes elágazásokat mutatjuk be.

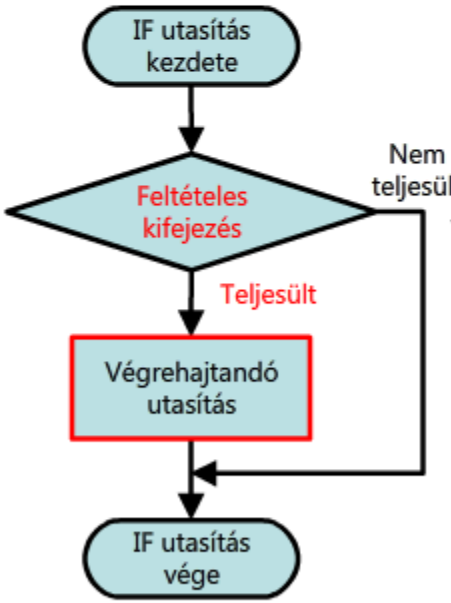
5.1 Feltételes elágazások (IF)

A feltételes elágazásokhoz IF (jelentése „ha”) utasításokat használnak. Az IF utasítások meghatározása alább látható.

```
IF conditional expression THEN
  Execution statement;          (* Az utasítás akkor hajtódik végre, ha a feltételes kifejezés teljesül. *)
END_IF;                         (* Az IF utasítás végét az END_IF; sorral kell lezárni. *)
```

Ebben a példaprogramban egy utasítás akkor kerül végrehajtásra, ha a feltételes kifejezés teljesül. Ha a feltételes kifejezés nem teljesül, az utasítás nem hajtódik végre.

Az alábbi ábra bemutatja a művelet folyamatát ebben a példaprogramban.

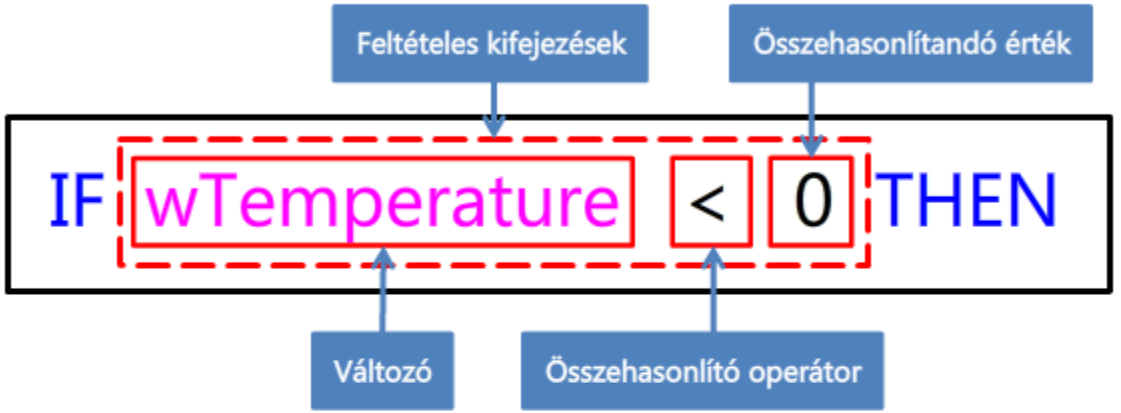


A következő példa a program változók értékeinek összehasonlítása általi elágazását szemlélteti. A példaprogramban a fűtőelem bekapcsol, ha a vezérlőpulton a hőmérséklet 0 fok alá esik.

```
IF wTemperature < 0 THEN
  bHeater := 1; (* A fűtőelem bekapcsol, ha a vezérlőpulton a hőmérséklet 0 fok alá esik. *)
END_IF;
```

5.1.1 Feltételes kifejezések írása

Az előző oldalon a „wTemperature < 0" feltételes kifejezését mutatta be, melynek jelentése: ha a wTemperature változó értéke kisebb, mint 0.
Mint ennél a kifejezésnél is látható, a feltételes kifejezések összehasonlító operátorokat használnak a változók és az összehasonlítandó értékek közötti viszony megjelenítésére.



Az összehasonlító operátorok bal és jobb oldalán az értékek mint változók vagy összehasonlítási állandók jelennek meg.

A változók és állandók összehasonlításán túl a feltételes kifejezések a változók összehasonlítására és az összehasonlítási eredmény logikai műveletének vagy bittípusú változók végrehajtására is használhatók.

Változók összehasonlítása

- uValue1 <= uValue2

Logikai művelet két összehasonlítási eredmény között

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

Logikai művelet két bittípusú változó között

- bSwitch0 OR bSwitch1

Az alábbi táblázat az összehasonlító operátorok típusait tartalmazza.

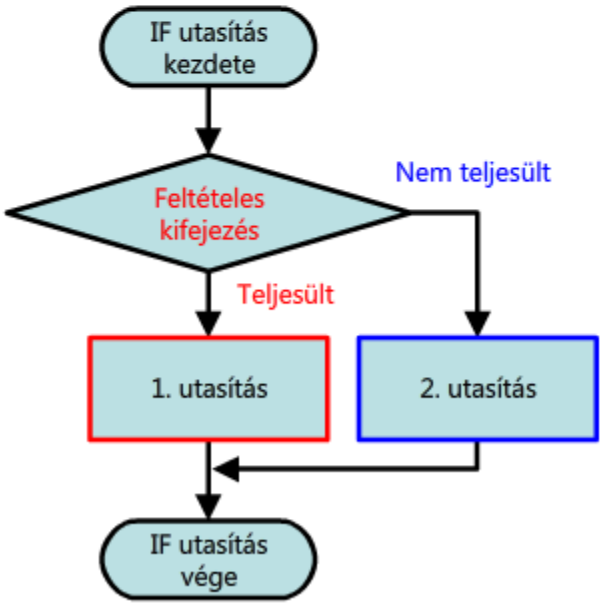
Operátor	Jelentés
>	Nagyobb, mint
<	Kisebb, mint
>=	Nagyobb vagy egyenlő
<=	Kisebb vagy egyenlő
=	Egyenlő
<>	Nem egyenlő

5.1.2 Az IF utasítás kivételes elágazása (ELSE)

Az utasítások végrehajtására egyszerű IF utasításokat (ld. 5.1) használnak a feltételes kifejezés teljesülésekor.
Egy eltérő utasítás végrehajtására, ha a feltételes kifejezés nem teljesült, használjuk az ELSE utasítást.

```
IF conditional expression THEN
  Execution statement 1;          (* Az 1. utasítás akkor hajtódik végre, ha a feltételes kifejezés teljesül. *)
  ELSE
  Execution statement 2;          (* A 2. utasítás akkor hajtódik végre, ha a feltételes kifejezés nem teljesül. *)
END_IF;
```

Az alábbi ábra bemutatja a művelet folyamatát ELSE utasítás használatakor.



Az alábbi példaprogram különböző utasításokat hajt végre attól függően, hogy a feltétel teljesült-e.

Az 5.1. fejezet példaprogramjának hátulütője, hogy a fűtőelem a 0 fok elérése után is folyamatosan növeli a hőmérsékletet. Az alábbi program ezzel szemben lekapcsolja a fűtőelemet, ha a „wTemperature" átlépi a „0" fokot.

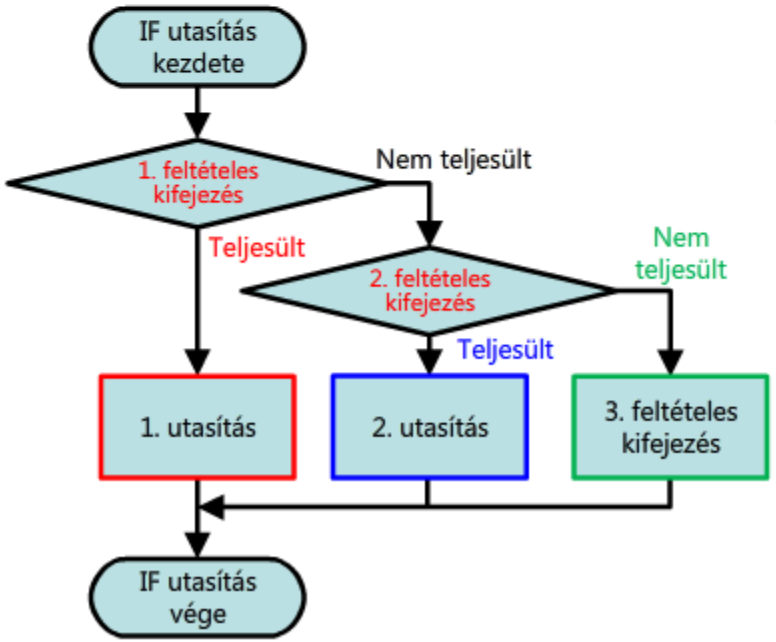
```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Bekapcsolja a fűtőelemet, ha a hőmérséklet 0 fok alá esik. *)
  ELSE
  bHeater := 0;  (* Lekapcsolja a fűtőelemet, ha a hőmérséklet eléri vagy átlépi a 0 fokot. *)
END_IF;
```

5.1.3 Az IF utasítás további elágazása (ELSEIF)

Az eltérő utasítások végrehajtására ELSE utasításokat használnak, ha a feltételes kifejezés nem teljesül.
További feltételes elágazások ELSIF utasításokkal adhatók hozzá, melynek szerepe, hogy ha az előző feltételes kifejezés nem teljesült, akkor a program egy másik feltételes kifejezést is ellenőrizzen.

```
IF Conditional expression 1 THEN
  Execution statement 1;     (* Az 1. utasítás akkor hajtódik végre, ha az 1. feltételes kifejezés teljesül. *)
  ELSIF Conditional expression 2 THEN
  Execution statement 2;     (* A 2. utasítás akkor hajtódik végre, ha az 1. feltételes kifejezés nem teljesül és a 2. feltételes kifejezés teljesül. *)
  ELSE
  Execution statement 3;     (* A 3. utasítás akkor hajtódik végre, ha az 1. és a 2. feltételes kifejezések nem teljesülnek. *)
END_IF;
```

Az alábbi ábra bemutatja a művelet folyamatát ELSEIF utasítás használatakor.



Az ELSIF utasítást hozzáadtuk az 5.1.2. fejezetben használt példaprogramhoz, hogy megbirkózhassunk azzal a helyzettel, amikor a hőmérséklet átlépi a 40 fokot.

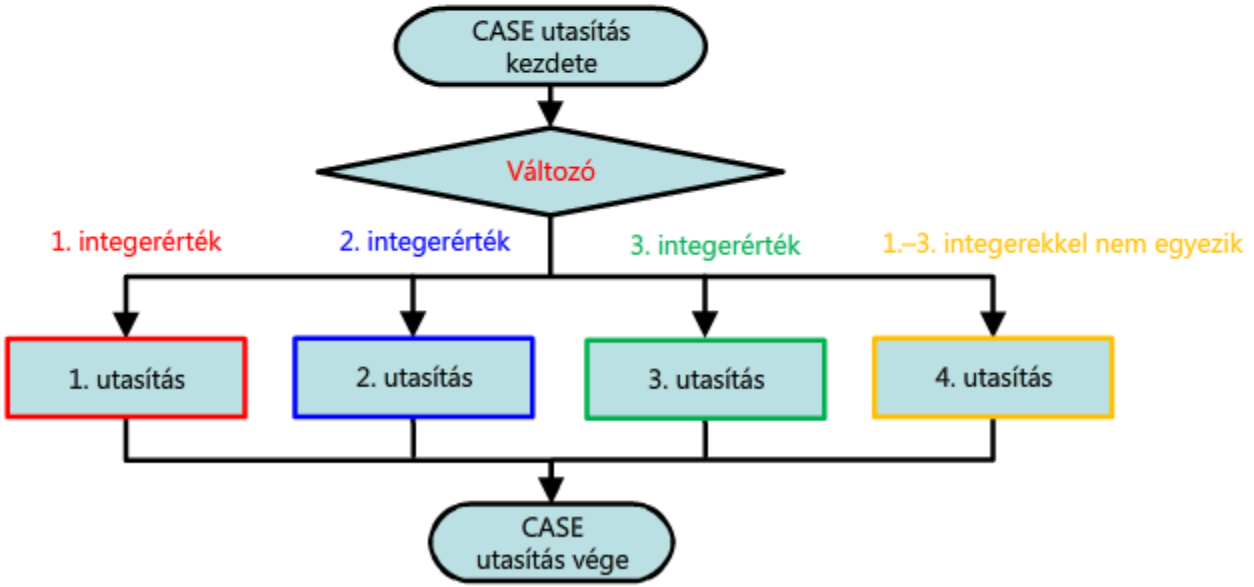
```
IF wTemperature < 0 THEN
   bHeater := 1; (* Bekapcsolja a fűtőelemet, ha a hőmérséklet 0 fok alá esik. *)
   bCooler := 0; (* Kikapcsolja a hűtőegységet, ha a hőmérséklet 0 fok alá esik. *)
   ELSIF 40 < wTemperature THEN
   bHeater := 0; (* Lekapcsolja a fűtőelemet, ha a hőmérséklet átlépi a 40 fokot. *)
   bCooler := 1; (* Bekapcsolja a hűtőegységet, ha a hőmérséklet átlépi a 40 fokot. *)
   ELSE
   bHeater := 0; (* Lekapcsolja a fűtőelemet, ha az előző feltételek egyike sem teljesül. *)
   bCooler := 0; (* Lekapcsolja a hűtőegységet, ha az előző feltételek egyike sem teljesül. *)
END_IF;
```

5.2 Feltételes elágazások integerértékek szerint (CASE)

Az IF utasításokat elágazásokhoz használják attól függően, hogy a feltételes kifejezések teljesültek-e.
A CASE utasításokat integerértékek szerinti elágazásokhoz használják.
Az alábbi ábra bemutatja, hogyan írható CASE utasítás.

```
CASE Variable OF
    Integer value 1: Execution statement 1;     (* A 1. utasítás akkor hajtódik végre, ha a változó megegyezik a 1. integerértékkel. *)
    Integer value 2: Execution statement 2;     (* A 2. utasítás akkor hajtódik végre, ha a változó megegyezik a 2. integerértékkel. *)
    Integer value 3: Execution statement 3;     (* A 3. utasítás akkor hajtódik végre, ha a változó megegyezik a 3. integerértékkel. *)
    ELSE 4. végrehajtandó utasítás;             (* A 4. utasítás akkor hajtódik végre, ha a változó egyik integerértékkel sem egyezik meg. *)
END_CASE;                                       (* A CASE utasítást „END_CASE;" sorral kell lezárni. *)
```

Az alábbi ábra bemutatja a művelet folyamatát CASE utasítás használatakor.



5.2.1 Példaprogram CASE utasításra

A CASE utasítás végrehajtását a példaprogram műveletén keresztül mutatjuk be.

Kattintson gombra a következő oldalra ugráshoz.
Az animáció ismételt megtekintéséhez kattintson a „Lejátszás" gombra.

Lejátszás


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

Súly:	uSize	Osztály
0–20 kg	1	M
21–30 kg	2	L
31–40 kg	3	XL
41 kg és felette	4	Oversize

5.3 Összefoglalás

A fejezet tartalma:

- Feltételes elágazások IF utasítással
- Feltételes kifejezések írása
- Feltételes elágazások integerértékek szerint (CASE utasítás)

A következő fontos szempontokat kell figyelembe venni:

IF utasítás	• IF utasítással a program egy feltételes kifejezés teljesülésekor elágazik. • Ha a feltételes kifejezés nem teljesül, elágazás létrehozásához az ELSE utasítás használható. • Ha az IF utasításban a feltételes kifejezés nem teljesül, újabb elágazás hozzáadásához az ELSEIF utasítás használható.
Feltételes kifejezés	• A feltételes kifejezések összehasonlító operátorok segítségével a változók és az összehasonlítandó értékek közötti viszonyt jelenítik meg.
CASE utasítás	• A CASE utasításokat integerértékek szerinti elágazásokhoz használják.

6. fejezet Tárolás és adatkezelés

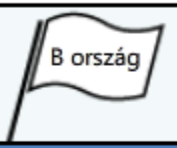
Az I/O-vezérlési alkalmazás mellett a programozható vezérlőket mint a termelési rendszer központját nagy mértékű adatfeldolgozásra is használják.
Ehhez az adatokat tárolni kell, illetve szükség szerinti olvasni kell tudni őket.
Ez a fejezet bemutatja, hogy az adatok tárolására és feldolgozására hogyan írhatók tömör programok.

- A változók felosztására és rendezésére tömböket használnak.
- Az összefüggő változók elrendezésére adatstruktúrákat használnak.
- A ciklikus feldolgozású programok a FOR utasítás segítségével hatásosan dolgozzák fel a tömböket.

A tömbök, az adatstruktúrák és a FOR utasítás használatával az adatok tárolására és kezelésére tömör programok hozhatók létre.

6.1 Adatok rendezése és tárolása (tömb)

Tömböket használva egy változóval több érték is kezelhető.
Az alábbi példa az egyes gépjárműgyártó üzemek termelési mennyiségi adatait célországonként tárolja.

Célország			
Termelési mennyiség	35	75	65

A célországonkénti termelési mennyiségi adatokat egy változóhoz rendelik hozzá. Tömbök használata nélkül minden egyes célországhoz külön változót kell létrehozni.
Azonban tömböket használva több célország termelési mennyiségi adatai egy változóhoz rendelhetők, és abban tárolhatók.

Tömb nélkül

uProductionA
uProductionB
uProductionC

Tömbbel

uProduction

Az egyes változók a tömbön belül elemszámok segítségével határozhatóak meg. Az elemszámok [0] értéktől kezdődnek.

uProduction [0]

Változónév — Elemszám

Célország (sor)		
A ország	[0]	35
B ország	[1]	75
C ország	[2]	65

Az alábbi példaprogramban az A ország tervezett termelési mennyiségének változója kerül hozzárendelésre.

```
uShowProductionPlan := uProduction[0];
(* Az A ország elemszámát határozza meg. *)
```

6.1.1 Mátrixtömb

A következő lépésben a céladatok mellé vegyük fel a festési színek adatait!

Célország									
Festés színe									
Termelési mennyiség	10	5	20	15	40	20	25	30	10
	Összesen 35			Összesen 75			Összesen 65		

Ahogy az alábbi ábrán is látható, az adatok szétválaszthatók, és az egyes célországok (sor) festési színei (oszlop) szerint tárolhatók.

Festés színe (oszlop)

Célország (sor)	Piros	Sárga	Kék
A ország	[0,0] 10	[0,1] 5	[0,2] 20
B ország	[1,0] 15	[1,1] 40	[1,2] 20
C ország	[2,0] 25	[2,1] 30	[2,2] 10

A célországot jelölő elemszám

A festés színét jelölő elemszám

Az adatokat ily módon sorokba és oszlopokba rendező tömböket mátrixtömböknek nevezzük.
A sorokat és oszlopokat jelölő elemszámokat vesszővel választjuk el.

Tömbváltozó (mátrixtömb)

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

6.1.2 Mátrixtömbök hozzárendelése

Az alábbi példaprogram mátrixtömböket használva a sürgősen legyártandó autók számát hozzárendeli a sárga személygépkocsik B országban tervezett gyártási mennyiségéhez.

```
uAdditionalProduction := 5;
uProduction[1,1] := uProduction[1,1] + uAdditionalProduction;
(* Hozzáadja a további termelési mennyiséget (5 egység) az eredetileg tervezett termelési mennyiséghez. *)
```

Festés színe (oszlop)

Célország (sor)	Piros	Sárga	Kék
A ország	[0,0] 10	[0,1] 5	[0,2] 20
B ország	[1,0] 15	[1,1] 40 -> 45	[1,2] 20
C ország	[2,0] 25	[2,1] 30	[2,2] 10

6.1.3 A mátrixtömbökben tárolt információk feldolgozása

Az alábbi példaprogram mátrixtömböket használva kiszámítja a C ország összes festési színére tervezett teljes termelési mennyiséget, és a kapott értéket hozzárendeli egy változóhoz.

```
uProductionToday := uProduction[2,0] + uProduction[2,1] +uProduction[2,2];
(* Kiszámítja a C ország összes festési színére, mára tervezett teljes termelési mennyiséget, és a kapott értéket hozzárendeli az „uProductionToday” változóhoz. *)
```

Célország	A ország			B ország			C ország		
Festés színe									
Termelési mennyiség	10	5	20	15	45	20	25	30	10
	Összesen 35			Összesen 80			Összesen 65		

Festés színe (oszlop)

Célország (sor)	Piros	Sárga	Kék
A ország	[0,0] 10	[0,1] 5	[0,2] 20
B ország	[1,0] 15	[1,1] 45	[1,2] 20
C ország	[2,0] 25	[2,1] 30	[2,2] 10

6.2 Ciklusutasítás (FOR)

Itt ismét az előző oldalon használt példaprogram (hozzárendelt mára tervezett termelési mennyiség) látható.

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

Ezzel a példaprogrammal ha a festési színek száma nő, több változó hozzáadására van szükség. Ezzel a kifejezés hossza is nő, ami nehezebbé teszi az olvasását.

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

Ilyen az esetben a tisztább kód létrehozása érdekében ciklusutasítások használhatók.

A ciklusutasítások közé a FOR, a WHILE és a REPEAT utasítások tartoznak. Ez a tanfolyam csak a FOR utasítást tárgyalja.

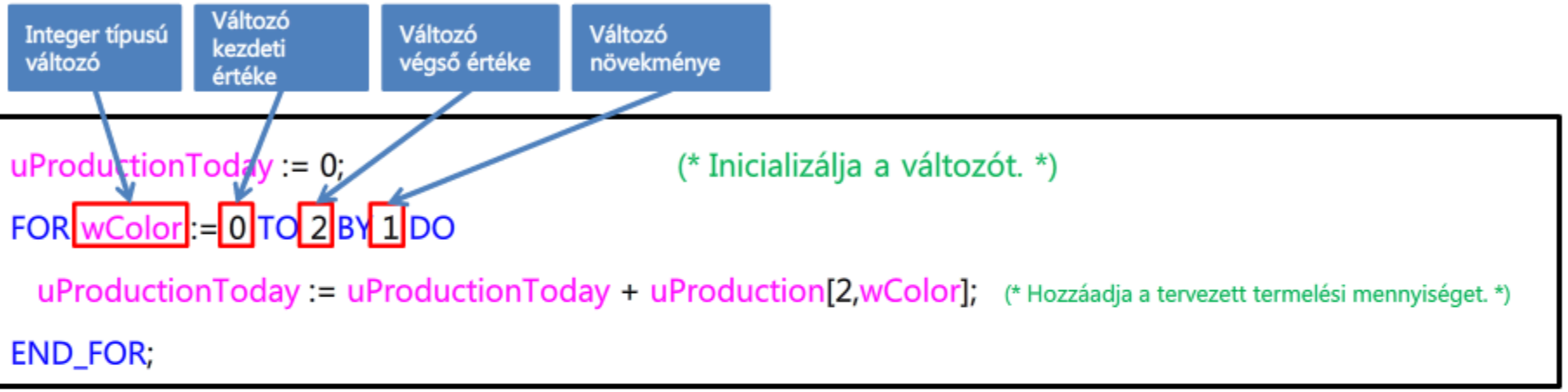
A FOR utasítások meghatározása alább látható.

```
FOR variable := initial value TO final value BY increments DO
   Execution statement; (* Az utasítás ciklikusan végrehajtódik, amíg a változó eléri a végső értéket. *)
END_FOR;                (* A FOR utasítás végét az END_FOR; sorral kell lezárni.  *)
```

Az utasítás addig ismétlődik, amíg a változó eléri a végső értékét, és az „END_FOR” utasítás végrehajtódik.

6.2 Ciklusutasítás (FOR)

Az alábbi példaprogram a FOR utasítást használva éri el a C országban az összes festési színre tervezett termelési mennyiséget.



A FOR utasítás segítségével a „wColor" változó a nulla kezdeti értékről indulva eggyel nő, és az utasítás addig ismétlődik, amíg a változó eléri a végső értéket, a kettőt.
A „wColor" változó a végrehajtási utasításban leírt „uProduction" tömbben a második elemszámként szerepel.
A „wColor" változó értéke az utasítás minden egyes megismétlésével eggyel nő. Az teljes mennyiség eléréséhez az egyes festési színek tervezett termelési mennyisége minden alkalommal hozzáadódik.

Ez a példaprogram ciklikusan háromszor hajtódik végre (először a piros [0] => másodszor a sárga [1] => harmadszor a kék [2]).

A program művelete a következő oldalon kerül bemutatásra.

6.2 Ciklusutasítás (FOR)

A FOR utasítás végrehajtását a példaprogram műveletén keresztül mutatjuk be.

A becsült termelési mennyiség tömbje

	Piros	Sárga	Kék
A ország	[0,0] 10	[0,1] 5	[0,2] 20
B ország	[1,0] 15	[1,1] 45	[1,2] 20
C ország	[2,0] 25	[2,1] 30	[2,2] 10

Kattintson a gombra a következő oldalra ugráshoz.
Az animáció ismételt megtekintéséhez kattintson a „Lejátszás" gombra.

Lejátszás

Number of repetition of the loop: 3

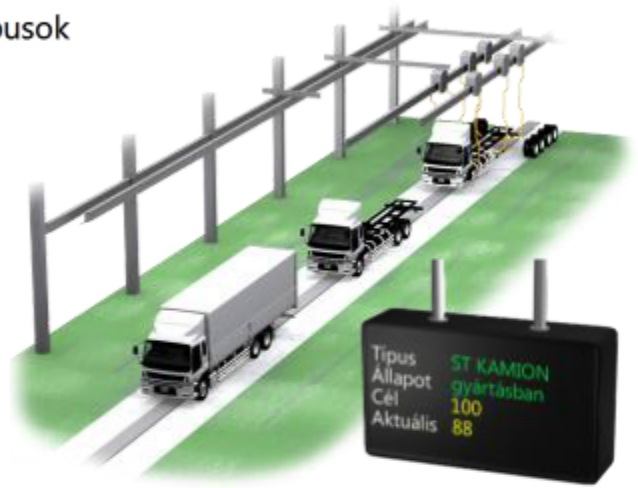
```
uProductionToday := 0;

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

6.3 Összefüggő adatok tárolása (struktúrák)

A struktúra lehetővé teszi, hogy egy változónév több összefüggő változót jelenítsen meg.
Az alábbi példában a személygépkocsi-gyártósorok állapota egy Andon kijelzőtáblán jelenik meg.

Az alábbi táblázatban a kijelzett tételnek megfelelő változónevek, értékek és adattípusok listája látható.

Elem	Változónév	Érték	Változó adattípusa
Típus	sModel	'ST TRUCK'	Szöveges karakterlánc
Állapot	bStatus	'gyártásban'	Bittípusú
Kitűzött termelési mennyiség mára	uPlanQty	'100'	Integer típusú szó (előjel nélküli)
Aktuális termelési mennyiség	uActualQty	'88'	Integer típusú szó (előjel nélküli)



Struktúra használata nélkül ha több gyártósor is van, a változóneveket minden sornál meg kell változtatni.
Az alábbiakban gyártósorok szerinti változónevekre adott példák láthatók.

Első gyártósor

s1stLineModel
b1stLineStatus
u1stLinePlanQty
u1stLineActualQty

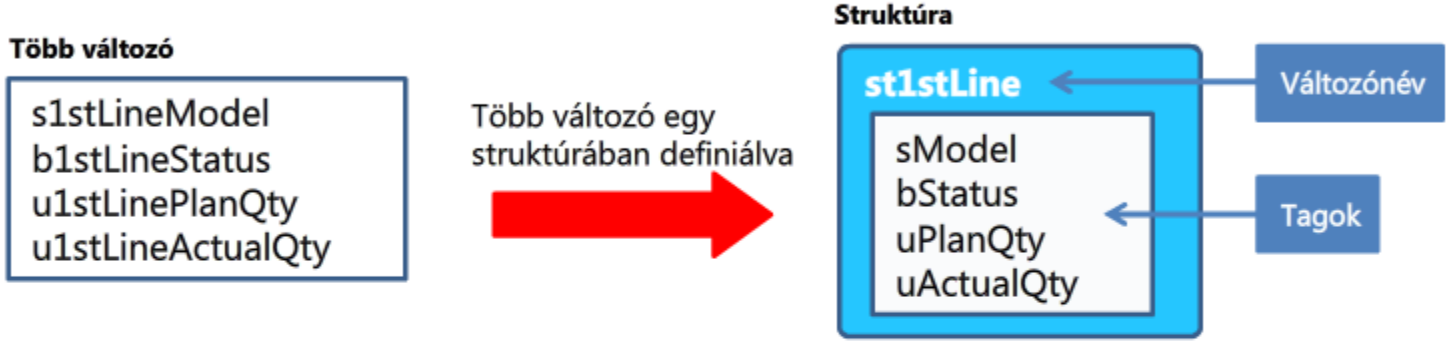
Második gyártósor

s2ndLineModel
b2ndLineStatus
u2ndLinePlanQty
u2ndLineActualQty

A kezelendő változók száma a gyártósorok számával együtt növekszik. Ezzel a program hossza is nő, ami nehezebbé teszi az olvasását.

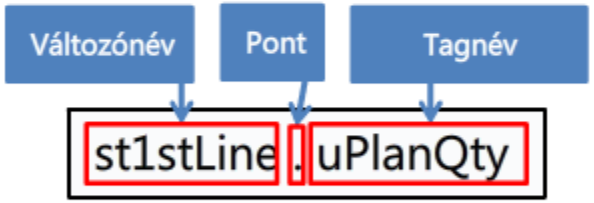
6.3 Összefüggő adatok tárolása (struktúrák)

A struktúrák használata lehetővé teszi, hogy egy változónév több, egy gyártósorhoz tartozó, összefüggő változót jelenítsen meg. A struktúrákat ekképpen fizikai objektumok (pl. készülékek, berendezések és munkadarabok) állapotainál és műszaki jellemzőinél az adatok kötegben történő szervezésére, tárolására és kezelésére használják.



A **st**ructure variable (struktúraváltozó) egy „st" előtagot tartalmaz, ami jelzi, hogy a változó egy struktúra.
A struktúra által definiált különálló változókat tagoknak nevezzük. Az egyes tagok adattípusa eltérő lehet.

A struktúratömb egyes tagjai a tömb elemszáma után egy, a tagnév elé helyezett pont segítségével határozhatók meg.



Az alábbi példaprogramban az első gyártósor struktúraváltozójának egyik tagjához egy állandót rendelünk.

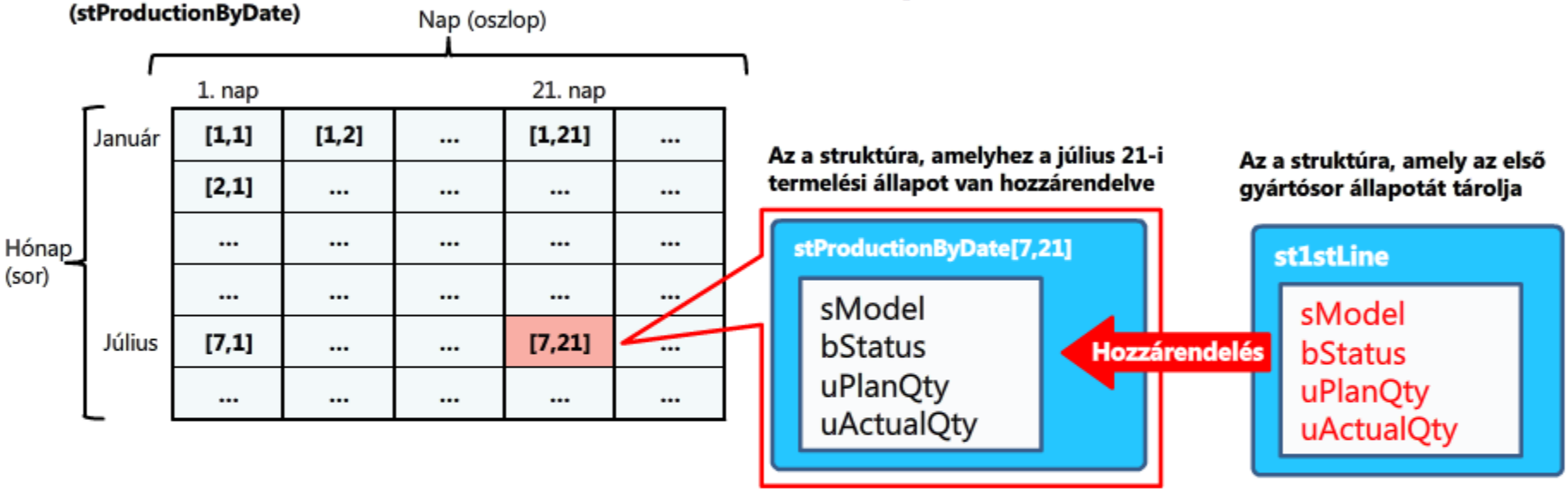
```
st1stLine.uPlanQty := 150;
 (* Az első gyártósor mai céltermelését 150-re állítja. *)
```

6.3.1 Struktúratömbök tárolása

Struktúrák létrehozhatók tömbökként.
Az alábbi példában a termelési állapot tárolása dátum szerint történik.

Dátum szerint rendezett* struktúra (stProductionByDate)

* Ebben a tömbben az elemszám „1”-től kezdődik.

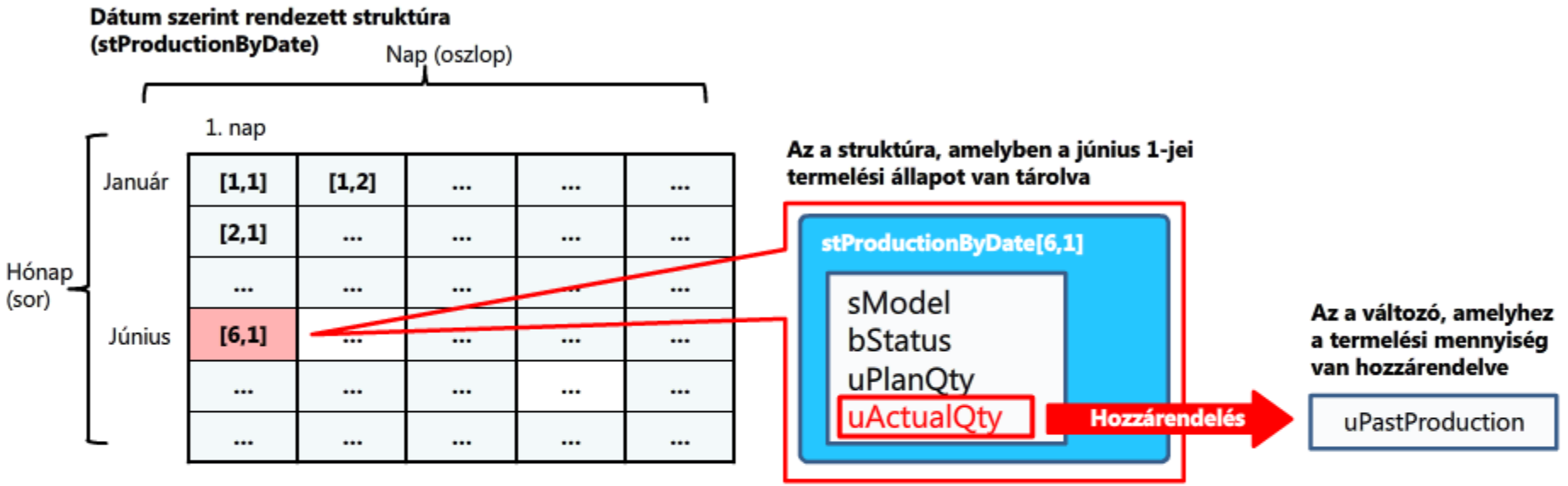



```
stProductionByDate[7,21] := st1stLine;
 (* A július 21-i termelési állapot a dátum szerint elrendezett
(stProductionByDate) struktúrában van tárolva. *)
```

Ily módon a tagokat a struktúrához rendeléskor nem kell külön meghatározni.

6.3.2 Struktúratömbök olvasása

Az alábbi példában a rendszer a termelési mennyiséget kiolvassa a dátum szerint rendezett struktúrából, majd hozzárendeli egy változóhoz.




```
uPastProduction := stProductionByDate[6,1].uActualQty;
(* Hozzárendeli a június 1-jei termelési mennyiséget az uPastProduction változóhoz. *)
```

A struktúratömb egyes elemei egy pontnak (.) és a tagnévnek a tömb elemszámához történő hozzáfűzésével határozhatók meg.

6.4 Összefoglalás

A fejezet tartalma:

- A tömbök áttekintése és használata
- Ciklikus feldolgozás FOR utasítással
- A struktúrák áttekintése és használata

A következő fontos szempontokat kell figyelembe venni:

Tömb	• Egy változó tömbök használatával több értéket is képes kezelni. • A tömbökben lévő különálló változók a változó nevének végén megadott elemszámokkal határozhatók meg.
FOR utasítás	• A ciklikus utasításokat ismétlődő műveletet igénylő programokban használják. • A FOR utasításokat a műveletek olyan ismétlésére használják, amely a ciklikus művelet befejezési feltételeinek teljesüléséig tart. Az „END_FOR" utasítás előtt álló utasítások ismétlődően végrehajtódnak.
Struktúra	• A struktúrák lehetővé teszik, hogy egy változónév több összefüggő változót jelenítsen meg. A struktúrák különböző adattípusú változókat tartalmazhatnak. • A struktúrákban definiált különálló változók (vagy tagok) meghatározása a struktúra változóneve után egy pont és a tagnév megadásával történik.

7. fejezet Karakterlánc típusú adatok kezelése

Egyes esetekben a programozható vezérlők karakterláncadatokat használnak, hogy parancsokat küldjenek vagy visszajelzést fogadjanak olyan csatlakoztatott eszközöktől, mint például a vonalkód-leolvasók, a hőmérséklet-vezérlők vagy az elektromos mérlegek. A karakterláncadatok összekapcsolása vagy kinyerése ilyen célokra lehet szükséges.

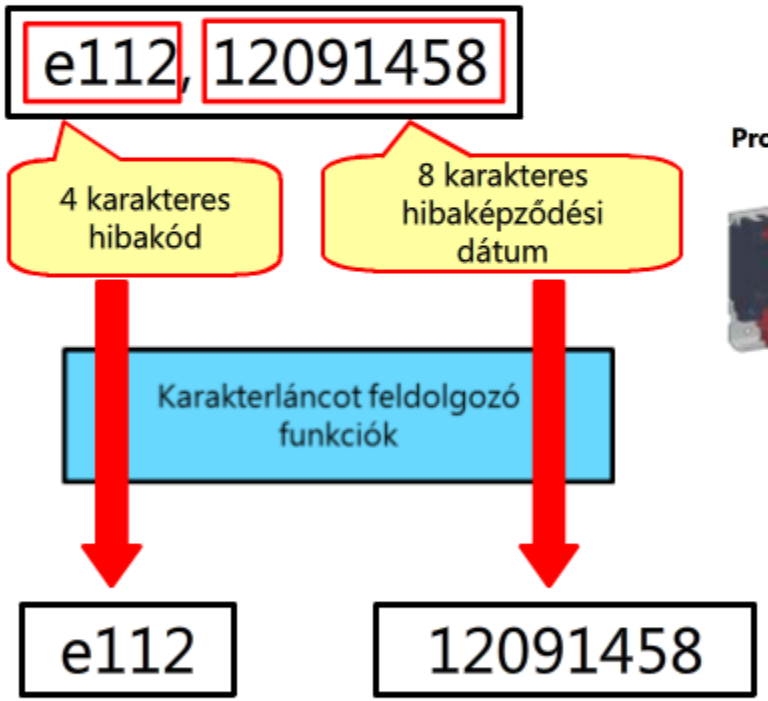
Ez a fejezet a karakterláncadatok kezelésének módját mutatja be.

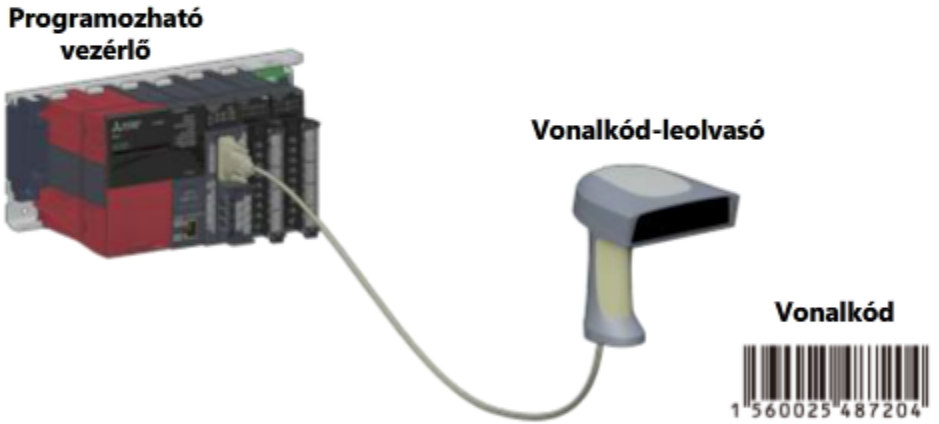
7.1 A karakterláncadatok kezelési példája

A karakterláncok feldolgozási példájaként itt egy olyan szituáció látható, amelyben egy vonalkód-leolvasóból olvasunk ki adatokat. A karakterláncok feldolgozására funkciókat (utasítások egy fajtáját) használunk.

Ahogy az lent látható, a vonalkód-leolvasó által beolvasott karakterláncok egy 4 karakteres rögzített hosszúságú hibakódot és egy 8 karakteres rögzített hosszúságú, hónapot, napot, órát és percet tartalmazó adatot tartalmaznak. A karakterláncokat feldolgozó példaprogram ezen a rendszeren keresztül kerül bemutatásra.

Példa vonalkód-leolvasóból kiolvasott karakterláncadatokra





A rendszer kinyert egy hibakódot.
7.3 A karakterláncok kinyerése (LEFT)

A hiba előfordulásának dátuma és ideje (14:58, december 9.) kinyerve.
7.4 A karakterláncok kinyerése (MID)

7.2 A karakterláncok hozzárendelése

Mielőtt a karakterláncok kinyerésére rátérnénk, ebben a részben bemutatjuk a karakterláncok adattípusait.

A programozható vezérlővel együtt használható karakterlánc-adattípusok az alábbi listában találhatók.

Adattípus	Feldolgozható karaktertípusok	Magyar jelölésű előtag	Előtag kifejtése
Karakterlánc	Alfanumerikus karakterek és számok (ASCII) vagy japán (Shift-JIS) karakterláncok	s	**s**tring (karakterlánc)
Karakterlánc [Unicode]	Számos nyelv és szimbólum karakterláncai	ws	**w**ide **s**tring (széles karakterlánc)

A használandó karakterlánc típusa függ a programozható vezérlőhöz csatlakoztatott eszköztől, illetve a vonatkozó nyelvtől. Ez a fejezet különböző típusú szöveges karakterláncokat mutat be.

Ha egy karakterlánc-változóhoz egy karakterlánc típusú karakterláncot rendel, a karakterláncot tegye aposztrófok (') közé.

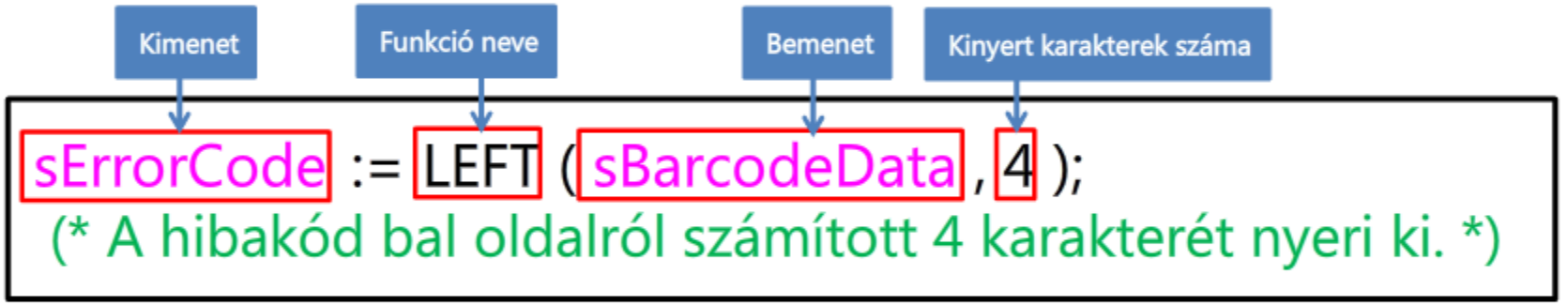
```
sDefault := 'e112,12091458';  (* Karakterlánc hozzárendelése *)
```

7.3 A karakterláncok kinyerése (LEFT)

Az „sBarcodeData” karakterlánc-változóból az „e112” hibakód lett kinyerve, amely változó az „e112,12091458” karakterláncot tartalmazza.

Változónév	Tárolt karakterlánc
sBarcodeData	e112, 12091458

A LEFT funkcióval csak a bemeneti karakterlánc bal oldalán kezdődő, megadott számú karakterek nyerhetők ki.
Az alábbiakban egy példaprogram látható.



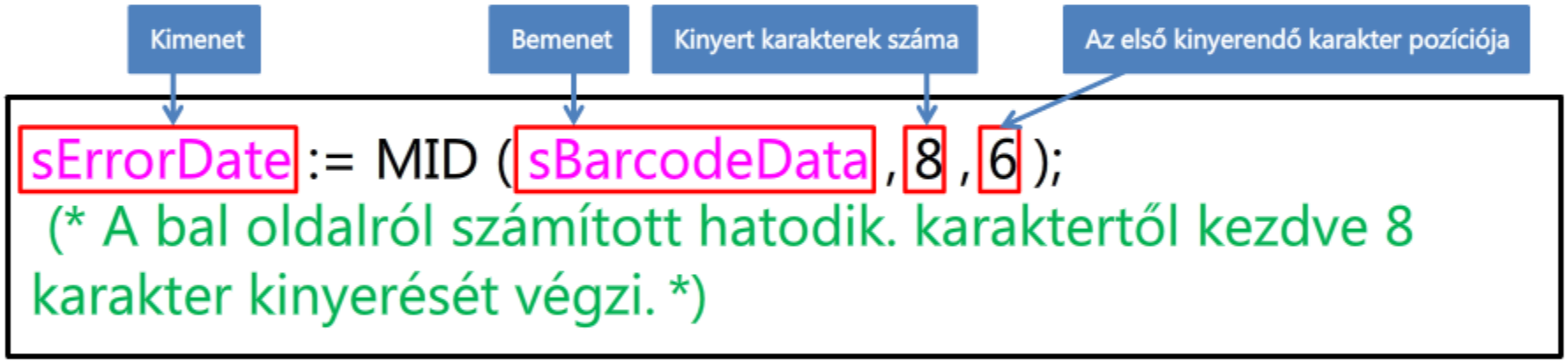
A bal oldalról négy karakter került kinyerésre. Az „e112” érték, amely a hibakódot megjelenítő karakterlánc, hozzárendelődik a kifejezés bal oldalához.

7.4 A karakterláncok kinyerése (MID)

Az „sBarcodeData” karakterlánc-változóból az „12091458” hibakeletkezési idő lett kinyerve, amely változó az „e112,12091458” karakterláncot tartalmazza.

Változónév	Tárolt karakterlánc
sBarcodeData	e112,12091458

A MID funkcióval a bemeneti karakterlánc meghatározott helyétől meghatározott számú karakter nyerhetők ki.
Az alábbiakban egy példaprogram látható.



Ebben a példában egy 8 karakteres karakterlánc kerül kivonásra a hatodik karaktertől indulva. Az „12091458” érték, amely a hiba előfordulási idejét megjelenítő karakterlánc, hozzárendelődik a kifejezés bal oldalához.

7.5 Összefoglalás

A fejezet tartalma:

- Karakterláncok karakterlánc-változókhoz történő hozzárendelési eljárása
- Karakterláncok kinyerésére szolgáló funkciók (LEFT és MID)

A következő fontos szempontokat kell figyelembe venni:

A karakterláncok hozzárendelése	• Ha egy karakterláncot hozzá szeretne rendelni egy karakterlánc-változóhoz, a karakterláncot tegye aposztrófok (') közé. • Mind a karakterlánc, mind pedig karakterlánctípust [Unicode] a programozható vezérlőhöz csatlakoztatott eszköznek vagy a vonatkozó nyelvnek megfelelően használja.
A karakterláncok kezelését végző funkciók	• A funkciók a karakterláncok kezelésére szolgálnak.

7.6 A tanfolyam összefoglalása

A tanfolyam bemutatta a programok ST-nyelven való létrehozásának alapjait.
Ezzel elérkeztünk az e-learning tanfolyam végére.

ST-programok a MELSOFT tervezői szoftver segítségével hozhatók létre.
A konkrét lépéseket, úgymint adatok megadása, illetve programok szerkesztése, mentése és fordítása a MELSOFT tervezői szoftverrel az alábbi helyeken találja.

- Mitsubishi FA e-Learning tanfolyam „MELSOFT GX Works3 (Structured Text)"
  (MELSOFT GX Works3 (strukturált szöveg)) (hamarosan kiadásra kerül)
- A MELSOFT tervezői szoftver üzemeltetési útmutatója

Az ST-vel kapcsolatos további információkat az alábbi helyen találja.

- Programozható vezérlője programozási útmutatója

Az utasításokkal és funkciókkal kapcsolatos információkat alkalmazása számára az alábbi helyen találja.

- Programozható vezérlőjének programozási kézikönyve

Teszt — Záró teszt

Most, hogy elvégezte a Programozás alapjai (strukturált szöveg) kurzus minden leckéjét, készen áll a záró tesztre. Ha valami nem világos a témával kapcsolatban, használja ki a lehetőséget az ilyen témák áttekintésére.
Ebben a záró tesztben összesen 12 kérdés (20 elem) található.
A záró tesztet annyiszor végezheti el, ahányszor csak akarja.

A teszt pontozása
A válasz kiválasztása után feltétlenül kattintson a **Válasz** gombra. A választ a rendszer nem rögzíti, ha a Válasz gombra való kattintás nélkül lép tovább. (A kérdés megválaszolatlanként lesz rögzítve.)

Pontozási eredmények
A pontszám oldalon a helyes válaszok száma, a kérdések száma, a helyes válaszok százalékaránya és a teszt sikeres/sikertelen eredménye jelenik meg.

Helyes válaszok: 4

Összes kérdés: 4

Százalék: 100%

A teszt teljesítéséhez a válaszok **60%**-ának kell helyesnek lennie.

Továbblépés | Áttekintés

- Kattintson a **Továbblépés** gombra a tesztből való kilépéshez.
- Kattintson az **Áttekintés** gombra a teszt áttekintéséhez. (Helyes válasz ellenőrzése)
- Kattintson az **Újra** gombra a teszt megismétléséhez.

Teszt Záró teszt 1

A strukturált szöveg (ST) jellemzői
Válassza ki az ST-vel kapcsolatos helytelen állítást!

- Az ST könnyen elsajátítható azok számára, akik jártasak a C vagy a BASIC nyelvű programok írásában.
- A számítások (például összeadás és kivonás) jellemzően használt matematikai kifejezésekként írhatók le.
- Az érintkezők és tekercsek szimbólumait használva egy elektromos áramkörre hasonlító program készíthető.
- Az ST adatkezelésre alkalmas.

Válasz Vissza

Teszt — Záró teszt 2

Az ST alapjai
Válassza ki az ST-ben helyesen megírt utasítást!

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

Válasz | Vissza

Teszt — Záró teszt 3

Leíró megjegyzések
Válassza ki az ST-ben helyesen megírt megjegyzést!

- ' A változóhoz „1" értéket rendel.
- (* A változóhoz „1" értéket rendel. *)
- { A változóhoz „1" értéket rendel. }
- <!-- A változóhoz „1" értéket rendel. -->

Válasz | Vissza

Teszt Záró teszt 4

ST-program végrehajtási szekvenciája
*Az „uTotalProduction” kezdeti értéke 100. Az „uTotalProduction” értéke az alábbi példaprogram feldolgozása után „101” lesz. Válassza ki az „uTotalProduction” néhány másodperccel későbbi állapotát!

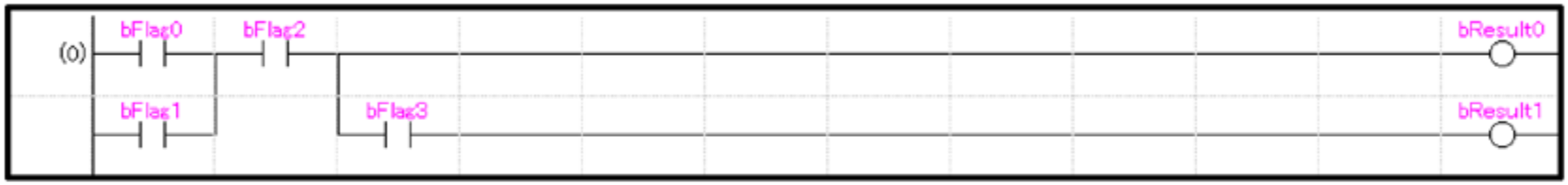
```
uTotalProduction := uTotalProduction + 1;
```

- ○ Az érték 101 marad.
- ○ Az érték továbbra is változik.

Válasz Vissza

Teszt — Záró teszt 5

Több feltétel kombinációja
Válassza ki a helyes ST-példaprogramot, amely megegyezik az alábbi LD-ben megírt példaprogram műveletével!



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

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

Válasz | Vissza

Teszt — Záró teszt 6

Az IF utasítás leírása ST-ben
A lenti példaprogram az alábbi művelet hajtja végre.
•Ha a hőmérséklet 5 fokra vagy az alá esik, a fűtőelem bekapcsol, a hűtőegység pedig kikapcsol.
•Ha a hőmérséklet túllépi az 50 fokot, a fűtőelem kikapcsol, a hűtőegység pedig bekapcsol.
•Ha a hőmérséklet a fenti állítások egyikének sem felel meg, a fűtőelem és a hűtőegység is kikapcsol.

*Változónevek: hőmérséklet (wTemperature), fűtőelem (bHeater) és hűtőegység (bCooler)
Adja meg a helyes választ a példaprogram üresen hagyott helyein!

```
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;
```

K1 --Select--

K2 --Select--

K3 --Select--

K4 --Select--

K5 --Select--

Válasz Vissza

Teszt — Záró teszt 7

CASE utasítások

Válassza ki a helyes kifejezéseket (K1–K5) a CASE utasítás alábbi állításaiban!

A CASE utasításokat elágazásokhoz használjuk a (K1) értéke szerint.
Az alábbi példaprogramban, ha a (K2) értéke 25, a (K3) változó értékéhez (K4) rendelődik hozzá.
Ha a (K2) változó értéke nem 10, 25 vagy 8, a (K3) változóhoz (K5) rendelődik hozzá.

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

K1 --Select--

K2 --Select--

K3 --Select--

K4 --Select--

K5 --Select--

Válasz Vissza

Teszt — Záró teszt 8

ST-tömbök és ismétlődő utasítások
Az alábbi példaprogram összegzi az összes, Y országba kirendelt típus tervezett termelési mennyiségét, majd a kapott értéket egy változóhoz rendeli. Válassza ki a tömb azon részét, amelynek kiolvasása a FOR utasítás 3 ciklusú végrehajtását követi!

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

A legyártott egységek becsült számának típusonkénti és célországonkénti tárolását végző tömb (uProduction)

Célország (sor)	Típus (oszlop) 1. típus	2. típus	3. típus	4. típus
X ország	[0,0]	[0,1]	[0,2] C	[0,3]
Y ország	[1.0]	[1.1] A	[1.2] D	[1.3] E
Z ország	[2,0]	[2,1] B	[2,2]	[2,3]

- A
- B
- C
- D

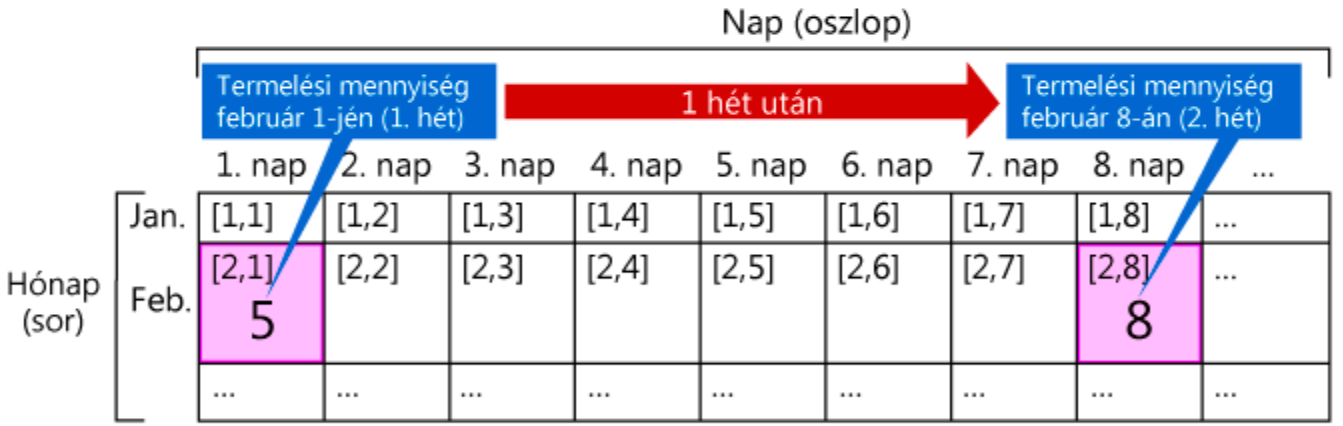
Válasz | Vissza

Teszt — Záró teszt 9

ST-tömbök és ismétlődő utasítások
A teljes termelési mennyiség minden hét ugyanazon napján az alábbi példaprogramba kerül. A 4 hét utáni teljes összeg a napi termelési mennyiséget tároló tömbből származik. Válassza ki a példaprogramhoz a helyes számot!

```
uTotalProduction := 0;
FOR wOnceAWeek := 1 TO ■ BY 7 DO
  uTotalProduction := uTotalProduction + uProductionByDate[2,wOnceAWeek];
END_FOR;
(* Kivonja és összeadja a termelési mennyiséget a hét azonos napjain 4 héten keresztül február 1-jétől kezdve. *)
```

A termelési mennyiséget naponként tároló tömb (uProductionByDate)



- 22
- 21
- 4
- 28

Válasz | Vissza

Teszt — Záró teszt 10

Struktúrák jellemzői ST-ben
Válasszon ki egy struktúrákkal kapcsolatos helytelen állítást!

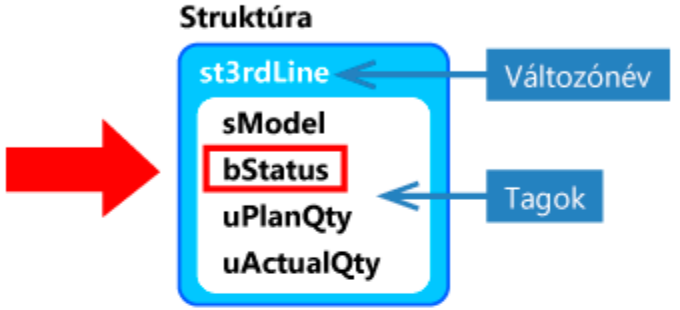
- A struktúrákat adatok eszközökön való olyan feltételek szerinti rendezésére és tárolására használják, mint az állapot és a műszaki jellemzők.
- A nagy mennyiségű adatot feldolgozó programok struktúrákat használva tömörebbre írhatók.
- A struktúrában definiált minden tagnak egyező adattípusúnak kell lennie.
- Az egyazon struktúrán belüli tagokhoz külön meghatározás nélkül értékek rendelhetők.

Válasz | Vissza

Teszt — Záró teszt 11

Tagok meghatározása struktúrákhoz ST-ben
Az alábbi struktúra egy személygépkocsi-gyártósorral kapcsolatos változók rendezésére szolgál.
Válassza ki a helyes választ a „bStatus” tag meghatározásához ebben a struktúrában!

Paraméter	Változónév
Típus	sModel
Állapot	bStatus
Termelési cél az aktuális napra	uPlanQty
Aktuális termelési szám	uActualQty

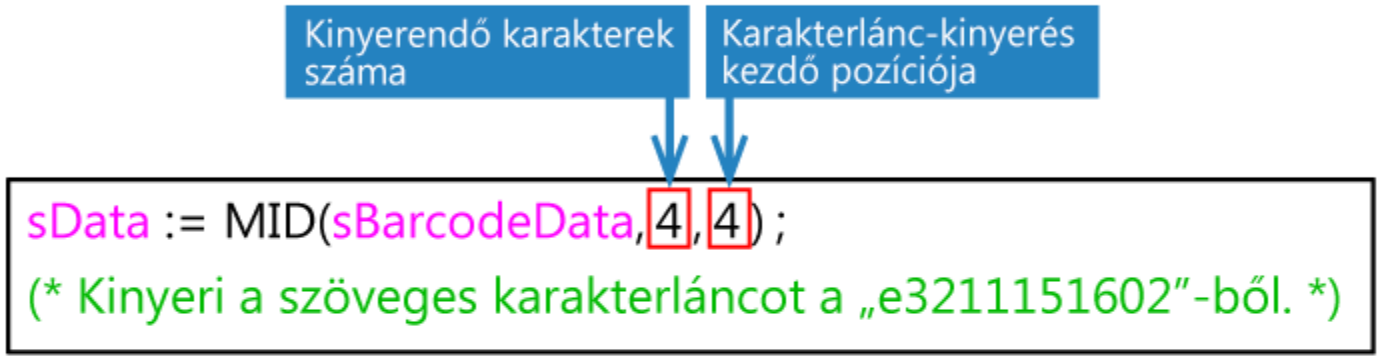


- st3rdLine.bStatus
- st3rdLine->bStatus
- st3rdLine[bStatus]
- st3rdLine[1]

Válasz | Vissza

Teszt — Záró teszt 12

Karakterláncok kezelése ST-ben
Az alábbi példaprogram az „sBarcodeData” változóban tárolt „e3211151602” karakterláncból egy meghatározott karakterláncot nyer ki. A MID funkció a meghatározott kiindulási helytől kezdve meghatározott számú karaktert nyer ki. Válassza ki a helyesen kinyert karakterláncot!



- 1151
- 1602
- e321
- 1115

Válasz | Vissza

Teszt Tesztpontszám

Befejezte a záró tesztet. Az eredményei a következők.
A záró teszt befejezéséhez lépjen a következő oldalra.

Helyes válaszok:	12
Összes kérdés:	12
Százalék:	100%

Továbblépés Áttekintés

Gratulálunk! A teszt sikerült.

Ön elvégezte a Programozás alapjai (strukturált szöveg) tanfolyamot.

Köszönjük, hogy részt vett kurzuson.

Reméljük, élvezte a tananyagot, és a kurzuson szerzett információk hasznosak lesznek az Ön számára a jövőben.

A kurzust annyiszor tekintheti meg, ahányszor csak akarja.

Áttekintés Bezárás