MITSUBISHI ELECTRIC
Changes for the Better

Softvér PLC
základné informácie o programovaní (štruktúrovaný text)

Tento kurz sa zameriava na to, ako vytvárať základné programy slúžiace na riadenie programovateľných radičov MELSEC.
V tomto kurze sa na popisy programu využíva štruktúrovaný text (jazyk ST).

Úvod Cieľ kurzu

Tento kurz vysvetľuje, ako vytvárať riadiace programy v štruktúrovanom texte (jazyku ST) pre programovateľné radiče MELSEC.

Pred absolvovaním tohto kurzu musíte absolvovať nasledujúci kurz, prípadne musíte mať ekvivalentné znalosti:

Programming Basics (Základné informácie o programovaní)

Znalosti alebo skúsenosti s programovacím jazykom C alebo BASIC vám môžu pomôcť pochopiť obsah tohto kurzu.

Úvod Štruktúra kurzu

Obsah tohto kurzu je nasledujúci.

Kapitola 1 – Prehľad štruktúrovaného textu

Táto kapitola opisuje funkcie a vhodné aplikácie štruktúrovaného textu (jazyka ST).

Kapitola 2 – Základné pravidlá programov v jazyku ST

Táto kapitola opisuje základné pravidlá používané na vytváranie programov v jazyku ST.

Kapitola 3 – Vytváranie riadiacich programov I/O

Táto kapitola opisuje postup vytvárania riadiacich programov I/O.

Kapitola 4 – Aritmetické operácie

Táto kapitola opisuje postup vytvárania programov aritmetických operácií.

Kapitola 5 – Podmienené vetvenie

Táto kapitola opisuje podmienené vetvenie.

Kapitola 6 – Ukladanie a spracúvanie údajov

Táto kapitola opisuje postup písania stručných programov na ukladanie a spracúvanie údajov.

Kapitola 7 – Spracúvanie reťazcových údajov

Táto kapitola opisuje metódy spracúvania reťazcových údajov.

Záverečný test

Miera úspešnosti testu: 60% alebo viac

Úvod Používanie tohto nástroja elektronického kurzu

Prechod na nasledujúcu obrazovku		Prechod na nasledujúcu obrazovku.
Návrat na predchádzajúcu obrazovku		Návrat na predchádzajúcu obrazovku.
Prechod na požadovanú obrazovku		Zobrazí sa „Obsah", pomocou ktorého budete môcť prejsť na požadovanú obrazovku.
Ukončenie kurzu		Ukončenie kurzu.

Úvod

Upozornenia týkajúce sa používania

Bezpečnostné opatrenia

Ak sa učíte pomocou skutočných produktov, dôkladne si prečítajte bezpečnostné opatrenia v príslušných návodoch.

Opatrenia v tomto kurze

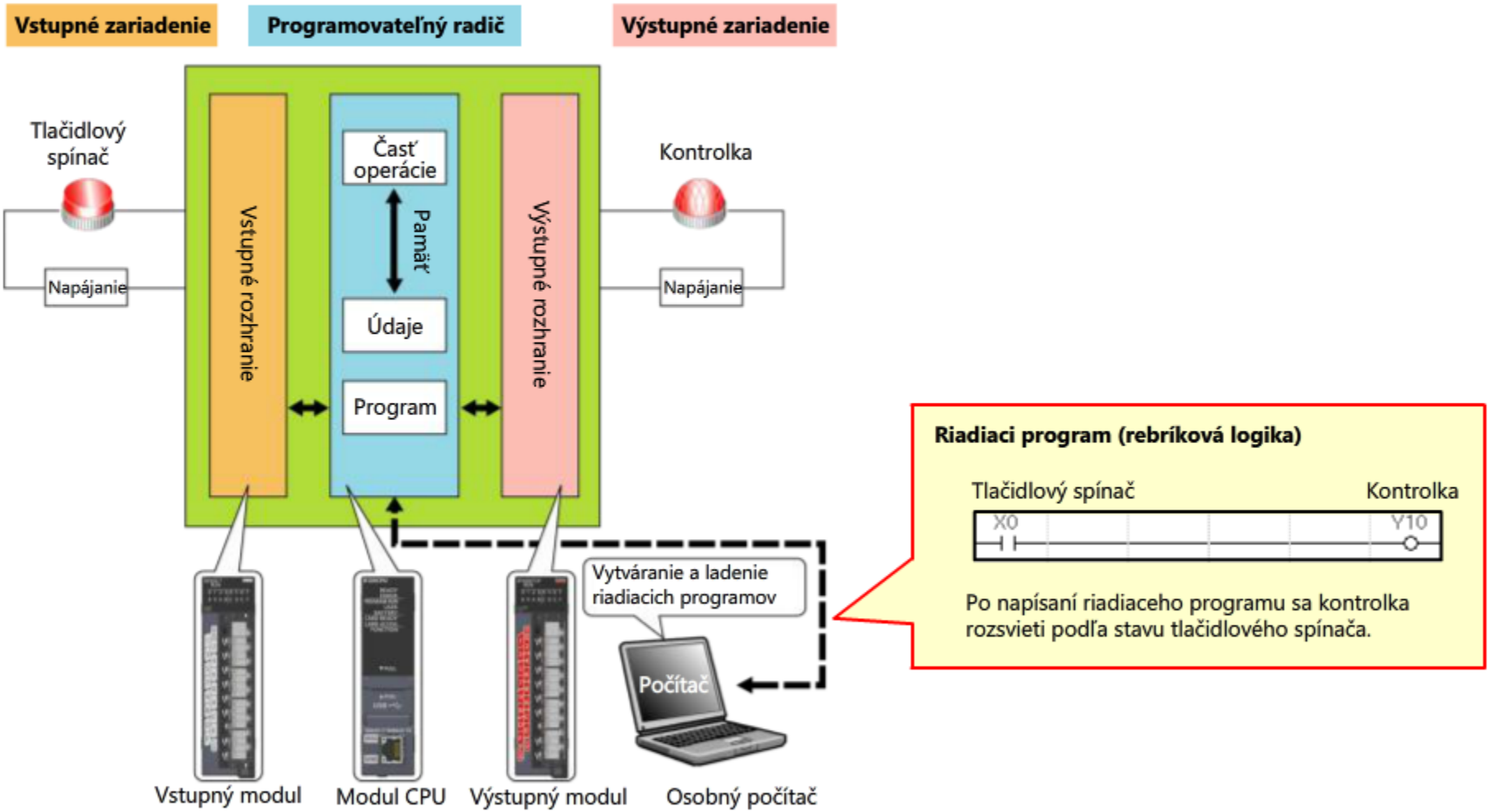
Zobrazené obrazovky používaného technického softvéru MELSOFT sa môžu líšiť od obrazoviek zobrazených v tomto kurze.
V tomto kurze sa na vytváranie programov používajú symboly rebríkovej logiky softvéru MELSOFT GX Works3.

Kapitola 1 Prehľad štruktúrovaného textu

Táto kapitola opisuje funkcie a vhodné aplikácie štruktúrovaného textu (jazyka ST).

1.1 Riadiace programy

Na nasledujúcom obrázku je zobrazená konfigurácia systému programovateľného radiča.
Programovateľné radiče fungujú na základe riadiacich programov.
Operácie programovateľných radičov možno konfigurovať podľa potreby vytvorením riadiacich programov.



Programovacie jazyky pre programovateľné radiče definuje medzinárodná norma, ktorú vytvorila organizácia International Electrotechnical Commission (IEC) (Medzinárodná elektrotechnická komisia).

1.2 Funkcie jazyka ST a porovnanie s inými programovacími jazykmi IEC

IEC 61131 je medzinárodná norma pre systémy programovateľných radičov.
Programovacie jazyky pre programovateľné radiče sa štandardizujú podľa normy IEC 61131-3. ST je jedným zo štandardných programovacích jazykov.
Každý jazyk má iné funkcie, ktoré zodpovedajú vašej aplikácii a programátorským zručnostiam.

V nasledujúcej tabuľke je uvedený zoznam funkcií programovacích jazykov IEC 61131-3.

Programovací jazyk	Funkcie
Ladder **D**iagram (LD) (Rebríkový diagram (LD))	• Symboly pre kontakty a cievky slúžia na vytvorenie programu pripomínajúceho elektrický obvod. • Krokmi programu sa možno jednoducho riadiť a sú zrozumiteľné aj pre začiatočníkov.
Structured **T**ext (ST) (Štruktúrovaný text (ST))	• Programy sa píšu ako text (znaky). • Jazyk ST sa ľahko naučia používať programátori so skúsenosťami s písaním programov v programovacom jazyku C alebo BASIC. • Výpočtové vzorce sú podobné matematickým výrazom, ktoré sú jednoducho zrozumiteľné. • Jazyk ST je vhodný na spracúvanie údajov.
Function **B**lock **D**iagram (FBD) (Diagram funkčného bloku (FBD))	• Programy sa píšu usporiadaním blokov s rôznymi funkciami a označením vzťahov medzi blokmi. • Jazyk FBD zlepšuje čitateľnosť, pretože možno jednoducho zobraziť celú operáciu.
Sequential **F**unction **C**hart (SFC) (Sekvenčná funkčná schéma (SFC))	• Podmienky a procesy sa píšu ako tokové diagramy. • Kroky programu sú jednoducho zrozumiteľné.
Instruction **L**ist (IL) (Zoznam inštrukcií (IL))	• Jazyk IL je podobný strojovému jazyku. • Jazyk IL sa už v súčasnosti používa len zriedka.

Tento kurz opisuje postup písania základných riadiacich programov pomocou jazyka ST.

1.3 Súhrn

Obsah tejto kapitoly:

- Súvzťažnosť medzi systémami programovateľných radičov a riadiacimi programami
- Medzinárodná norma pre riadiace programy
- Funkcie jazyka ST

Dôležité body na zváženie:

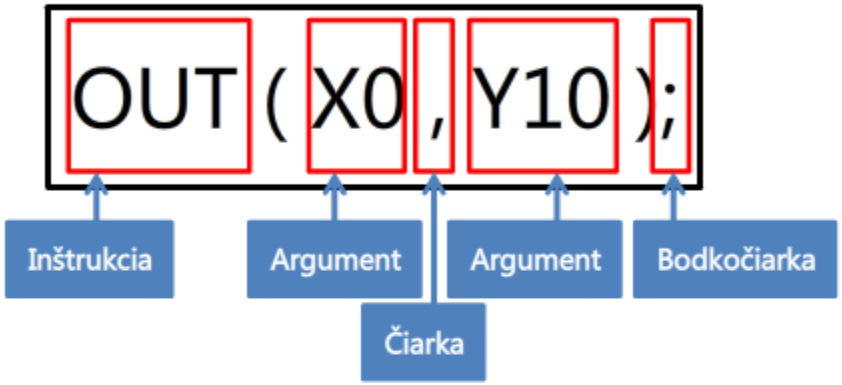
Súvzťažnosť medzi systémami programovateľných radičov a riadiacimi programami	• Programovateľné radiče fungujú na základe riadiacich programov. • Operácie programovateľných radičov možno konfigurovať podľa potreby vytvorením riadiacich programov.
Medzinárodná norma pre riadiace programy	• ST je jedným z programovacích jazykov IEC. • Ďalšími programovacími jazykmi IEC sú jazyky LD, FBD, SFC a IL, pričom každý z nich má iné funkcie, ktoré zodpovedajú vašej aplikácii a programátorským zručnostiam.
Funkcie jazyka ST	• Jazyk ST sa ľahko naučia používať programátori so skúsenosťami s písaním programov v jazyku C alebo BASIC. • Výpočty, napríklad sčítanie a odčítanie, možno napísať ako bežne používané matematické výrazy, ktoré sú jednoducho zrozumiteľné. • Jazyk ST je vhodný na spracúvanie údajov.

Kapitola 2 Základné pravidlá programov v jazyku ST

Táto kapitola opisuje základné pravidlá používané na vytváranie programov v jazyku ST.

2.1 Príklad základného programu (riadiaci príkaz I/O)

Táto časť obsahuje príklad základného programu v jazyku ST.
V nasledujúcom príklade programu sa výstup Y10 zapne, keď sa zapne vstup X0 a výstup Y10 sa vypne, keď sa vypne vstup X0.



Inštrukcia definuje operáciu, ktorá sa má vykonať.
Argumenty sa píšu za inštrukciou v zátvorkách.
Argumenty opisujú premenné, aritmetické výrazy a konštantné hodnoty.
V prípade programovateľných radičov MELSEC možno ako premenné používať zariadenia modulu CPU.

Počet argumentov závisí od príslušnej inštrukcie.
Viaceré argumenty sú oddelené čiarkami (,).

Jeden riadok zobrazený vyššie predstavuje jeden príkaz. Každý príkaz končí bodkočiarkou (;).
Program sa píše kombinovaním príkazov.

2.2 Príklad základného programu (príkaz pridelenia)

V nasledujúcom príklade je zobrazený program využívajúci príkaz pridelenia.
Nasledujúci príkaz prideľuje desiatkovú konštantu „5“ premennej „D10“.

Pre tento príkaz pridelenia sa používa operátor pridelenia (:=). Upozorňujeme, že dvojbodka (:) sa píše naľavo od znaku rovná sa (=). Príkaz pridelenia prideľuje hodnotu z pravej strany hodnote z ľavej strany.

Pridaním komentára do programu bude operácia zrozumiteľnejšia. Komentáre píšte medzi dve hviezdičky (* *).

2.3 Číselný zápis

V príklade programu na predchádzajúcej strane bola premennej pridelená desatinná hodnota.

Niekedy sa na sekvenčné riadenie používajú iné ako desatinné hodnoty, napríklad binárne a hexadecimálne.
V nasledujúcej tabuľke je uvedený zoznam typov číselných zápisov používaných v jazyku ST pre programovateľné radiče MELSEC.

Typ číselného zápisu	Metóda zápisu	Príklad
Binárny	Pridanie predpony „2#“.	**2#**11010
Osmičkový	Pridanie predpony „8#“.	**8#**32
Desatinný	Priamy vstup	26
	Pridanie predpony „K“.	**K**26
Hexadecimálny	Pridanie predpony „16#“.	**16#**1A
	Pridanie predpony „H“	**H**1A

Nižšie sú uvedené príklady programov na pridelenie hodnôt premenným.

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

2.3.1 Bitový zápis

Bity predstavujú podmienky true/false (pravdivé/nepravdivé), napríklad stavy signálov on/off (zapnuté/vypnuté).
Bity zároveň predstavujú vytvorenie/nevytvorenie podmienok.
V jazyku ST nemožno bity zapisovať ako hodnoty „ON" a „OFF". Tieto hodnoty sú vyjadrené ako „1" (ON) a „0" (OFF).
Bity tiež možno vyjadriť ako hodnoty „TRUE" a „FALSE".

V nasledujúcej tabuľke je uvedený zoznam rôznych typov zápisov.

Stav	ON	OFF
	True	False
Číselný zápis	1	0
Zápis hodnoty True/false	TRUE	FALSE

Nasleduje niekoľko príkladov pridelenia hodnôt bitovým premenným.

Číselný zápis

```
X0 := 1;
```

=

Zápis hodnoty True/false

```
X0 := TRUE;
```

Číselný zápis

```
X0 := 0;
```

=

Zápis hodnoty True/false

```
X0 := FALSE;
```

2.4 Spúšťacia sekvencia programu

Príkazy v jazyku ST sa vykonávajú v poradí zhora nadol.

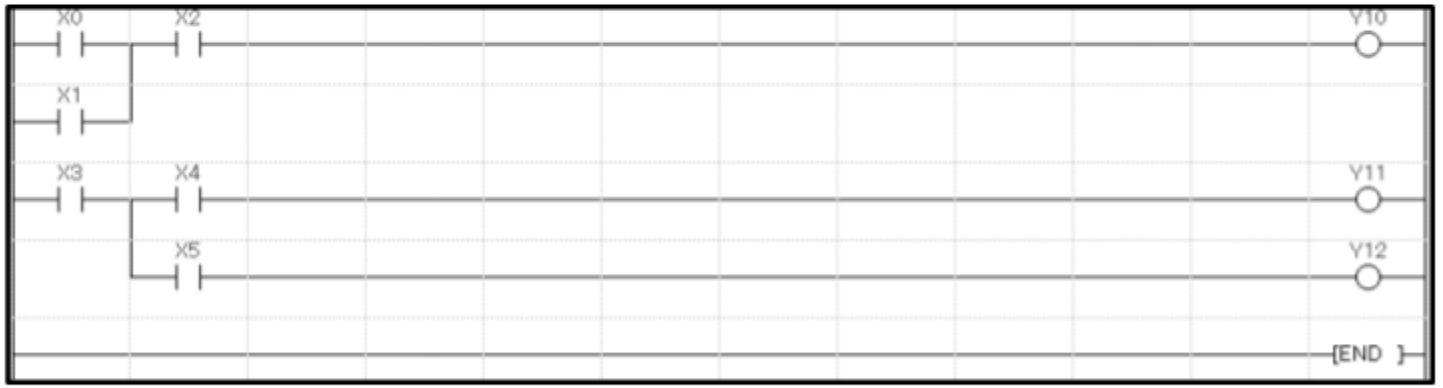
Príklad programu v jazyku ST

```
Y10 := (X0 OR X1) AND X2;        (* Vykonaný prvý *)
Y11 := X3 AND X4;                (* Vykonaný druhý *)
Y12 := X3 AND X5;   (* Vykonaný tretí. Nevyžaduje na konci príkaz KONIEC.*)
```

Opakované vykonanie

* Hoci je v jazyku LD na konci programu potrebný príkaz KONIEC, v jazyku ST potrebný nie je.

Nasledujúci program v rebríkovej logike predstavuje rovnakú operáciu ako príklad programu v jazyku ST uvedený vyššie.



Opakované vykonanie

Rovnako ako v jazyku LD, aj v jazyku ST sa inštrukcie vykonávajú opakovane po vykonaní poslednej inštrukcie návratom na prvú inštrukciu.

2.5 Súhrn

Obsah tejto kapitoly:

- Základný program v jazyku ST
- Formát príkazu pridelenia
- Číselný zápis
- Spúšťacia sekvencia programu
- Komentár

Dôležité body na zváženie:

Základný program v jazyku ST	• Príkaz je minimálnym prvkom programov v jazyku ST. • Každý príkaz končí bodkočiarkou (;). • Program sa píše kombinovaním príkazov.
Formát príkazu pridelenia	• Pre príkazy sa používa operátor pridelenia (:=).
Číselný zápis	• Typy číselných zápisov v jazyku ST • V jazyku ST sa ako bitové hodnoty namiesto zápisu „ON" a „OFF" používajú hodnoty „1" a „0". • Bitové hodnoty tiež možno v jazyku ST vyjadriť ako hodnoty „TRUE" a „FALSE".
Spúšťacia sekvencia programu	• Programy vytvorené v jazyku ST sa spúšťajú v poradí zhora nadol. • Rovnako ako v prípade programov v jazyku LD, aj programy v jazyku ST sa vykonávajú opakovane návratom na začiatok programu po dokončení procesu.
Komentár	• Pridaním komentára do programu bude operácia zrozumiteľnejšia. • Komentáre píšte medzi dve hviezdičky (* *).

Kapitola 3 Vytváranie riadiacich programov I/O

Táto kapitola opisuje postup vytvárania riadiacich programov I/O v jazyku ST.

3.1 Riadiace programy I/O

Nižšie je uvedený príklad programu na riadenie I/O programovateľného radiča.

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

Výstupná inštrukcia	Podmienka vykonania (argument)	Výstupné zariadenie (argument)

Výstupnou inštrukciou je hodnota „OUT". Argument špecifikuje podmienku vykonania a zariadenie, pre ktoré je výstup určený. Po splnení podmienky vykonania vstupu X0 sa zapne zariadenie Y10.

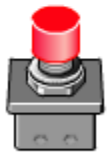
Kliknite na prepínač vstupu zobrazený nižšie. Prepínač vstupu X0 sa zapne.

- Po zapnutí prepínača vstupu X0 sa rozsvieti výstupná kontrolka Y10.
- Po vypnutí prepínača vstupu X0 výstupná kontrolka Y10 zhasne.

Príklad riadiaceho programu I/O napísaného v jazyku ST

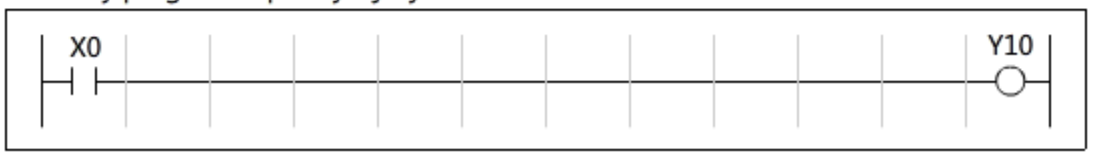
```
OUT(X0,Y10);
```

Vstupný prepínač X0



Výstupná kontrolka Y10

Rovnaký program napísaný v jazyku LD



3.1 Riadiace programy I/O

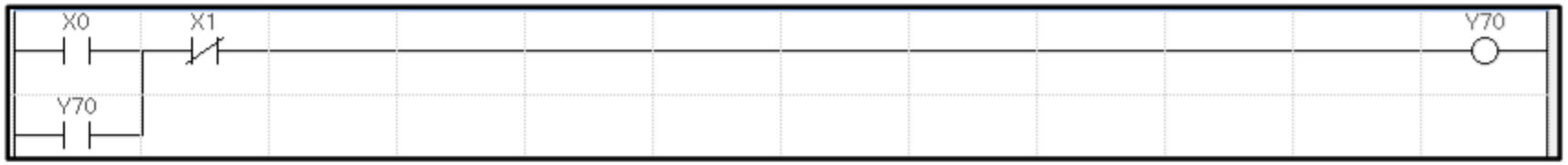
Podobne ako v jazyku LD, aj v tomto prípade sú okrem inštrukcie VÝSTUP k dispozícii rôzne inštrukcie, napríklad inštrukcie riadenia I/O a inštrukcie na spracovanie údajov.
Ďalšie informácie o inštrukciách dostupných v jazyku ST nájdete v príručke programovania k programovateľnému radiču.

Upozorňujeme, že ak príkaz „OUT(X0, Y10);“ zapíšete ako „Y10 := X0;“, spustí sa rovnaká operácia.

```
Y10 := X0;  (* Rovnaká operácia ako v prípade príkazu „OUT(X0, Y10);“ *)
```

3.2 Kombinovanie viacerých podmienok

Nasledujúci program v rebríkovej logike predstavuje automaticky udržiavaný obvod.



Rovnaký program možno v jazyku ST zapísať takto.

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

Logický operátor

Ako je zobrazené vyššie, logické operátory sa v jazyku ST používajú na kombinovanie viacerých podmienok.

V nasledujúcej tabuľke je uvedený zoznam logických operátorov.

Operátor	Význam
OR	Logické ALEBO
AND	Logické A
NOT	Logická negácia
XOR	Exkluzívne OR

3.3 Definovanie významu premenných

Pri používaní jazyka ST s programovateľnými radičmi MELSEC možno zariadenia aj štítky prideľovať premenným ako aliasy. Používatelia môžu používať štítky podľa aplikácií.

Po pridelení štítka súvisiaceho s príslušnou aplikáciou je operácia zrozumiteľnejšia.

```
Y10 := (X0 OR X1) AND X2;     (* Zapísané pomocou názvov zariadení *)
```

```
Lamp := (Switch0 OR Switch1) AND Switch2;     (* Zapísané pomocou štítkov *)
```

Štítky možno pomenúvať pomocou technického softvéru MELSOFT.

Ďalšie príklady programov v tomto kurze sú opísané pomocou štítkov.

3.4 Súhrn

Obsah tejto kapitoly:

Príklady riadiacich programov I/O

- Logické operátory sa v jazyku ST používajú na kombinovanie viacerých podmienok.
- Názvy zariadení a štítky možno používať ako názvy premenných.

Dôležité body na zváženie:

Kombinovanie viacerých podmienok	• Logické operátory sa v jazyku ST používajú na kombinovanie podmienok.
Definovanie významu premenných	• Po pridelení štítka súvisiaceho s príslušnou aplikáciou je operácia zrozumiteľnejšia.

Kapitola 4 Aritmetické operácie

Táto kapitola opisuje postup vytvárania programov aritmetických operácií.

- Opis aritmetických operácií
- Špecifikácia typov údajov zodpovedajúcich číselným rozsahom
- Pomenúvanie premenných s cieľom predchádzať nekonzistentným typom údajov

4.1 Základné aritmetické operácie

Tento príklad programu uvádza celkový objem výroby dvoch samostatných výrobných liniek.
V pravej časti rovnice je aritmetická operácia obsahujúca premenné a aritmetické operátory.

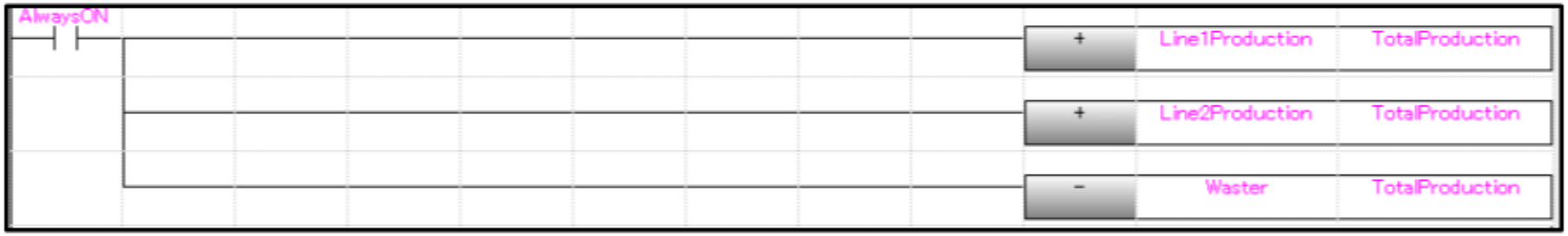
Príklad aritmetického programu zapísaného v jazyku ST

Operátor sčítania

Operátor odčítania

```
TotalProduction := Line1Production + Line2Production - Waster;
(* Celkový objem výroby dvoch výrobných liniek mínus počet chybných výrobkov
a následné pridelenie získanej hodnoty. *)
```

Nižšie je zobrazený rovnaký program zapísaný v jazyku LD.



Ako vidno v príklade vyššie, program musí byť v rebríkovej logike zapísaný pomocou 3 riadkov, no v jazyku ST ho možno zapísať v 1 riadku.

V nasledujúcej tabuľke je uvedený zoznam základných aritmetických operátorov.

Operátor	Význam
+	Sčítanie
–	Odčítanie
*	Násobenie
/	Delenie

4.2 Typy údajov premenných

Pre každú premennú musí byť špecifikovaný typ údajov, aby sa mohol definovať rozsah spracúvaných hodnôt.
Typy údajov pre číselné hodnoty používané v jazyku ST sú bitové údaje, celé čísla a reálne čísla.

V tabuľke nižšie sú uvedené typy údajov používané v tomto kurze, ktoré sa okrem iných používajú v jazyku ST.

Typ údajov		Rozsah údajov
Bitové		Stav ZAPNUTÉ/VYPNUTÉ bitových zariadení a stav pravdivé/nepravdivé výsledkov vykonania
Celé čísla	Slovo (nepriradené)	0 – 65 535
	Slovo (priradené)	−32 768 – 32 767
	Dvojslovo (nepriradené)	0 – 4 294 967 295
	Dvojslovo (priradené)	−2 147 483 648 – 2 147 483 647

Pri používaní typu celého čísla vyberte typ slova alebo dvojslova podľa rozsahu údajov a podľa potreby spracúvať záporné hodnoty vyberte typ priradené alebo nepriradené.
Pri nastavovaní názvu štítka pomocou technického softvéru MELSOFT špecifikujte typ údajov premennej.

4.3 Názvy premenných predstavujúce typy údajov

Použitie rôznych typov údajov v ľavej a pravej časti rovnice pridelenia môže spôsobiť chybu alebo neočakávaný výsledok. Nižšie je uvedený príklad takého prípadu.

```
ValueA := ValueB;      (* ValueA: slovo, celé číslo ValueB: dvojslovo, celé číslo *)
```

Celé číslo vyjadrené ako dvojslovo nemožno prideliť celému číslu vyjadrenému ako slovo. V tomto prípade však nie je typ údajov rozpoznateľný.

Predpony predstavujúce typ údajov možno pridávať k názvom premenných, aby bolo možné typy údajov vizuálne identifikovať. Tento typ pomenúvania premenných je známy ako maďarský zápis.

Typ údajov		Rozsah údajov	Predpona	Rozpísaný názov predpony
Bitové		Stav ZAPNUTÉ/VYPNUTÉ bitových zariadení a stav pravdivé/nepravdivé výsledkov vykonania	b	**B**it (Bitové)
Celé čísla	Slovo (nepriradené)	0 – 65 535	u	**u**nsigned word (nepriradené slovo)
	Slovo (priradené)	−32 768 – 32 767	w	signed **w**ord (priradené slovo)
	Dvojslovo (nepriradené)	0 – 4 294 967 295	ud	**u**nsigned **d**ouble-word (nepriradené dvojslovo)
	Dvojslovo (priradené)	−2 147 483 648 – 2 147 483 647	d	signed **d**ouble-word (priradené dvojslovo)

Príklad programu vo vrchnej časti tejto strany možno pomocou maďarského zápisu zapísať takto:

```
wValueA := dValueB;  (* Premennú vyjadrenú ako dvojslovo nemožno prideliť premennej vyjadrenej ako slovo. *)
```

Použitím maďarského zápisu možno pri písaní programu identifikovať nekonzistentné typy údajov.

V zvyšnej časti kurzu sú názvy premenných v príklade zapísané pomocou maďarského zápisu.

4.4 Súhrn

Obsah tejto kapitoly:

- Opis aritmetických operácií
- Špecifikácia typov údajov zodpovedajúcich číselným rozsahom
- Pridávanie názvov premenných predstavujúcich typy údajov

Dôležité body na zváženie:

Základné aritmetické operácie	• V jazyku ST možno na vyjadrenie výpočtov používať operátory bežne používané vo všeobecných programovacích jazykoch.
Typy údajov premenných	• Pre každú premennú musí byť špecifikovaný typ údajov, aby sa mohol definovať rozsah spracúvaných hodnôt.
Pridávanie názvov premenných predstavujúcich typy údajov	• Použitie maďarského zápisu na opísanie názvov premenných umožňuje identifikovať nekonzistentné typy údajov premenných pri písaní programu.

Kapitola 5 Podmienené vetvenie

Riadiace programy obsahujú tiež časti kódu, v ktorom sa reálne spracovanie mení v súlade so špecifikovanými podmienkami.
Táto kapitola opisuje podmienené vetvenie.

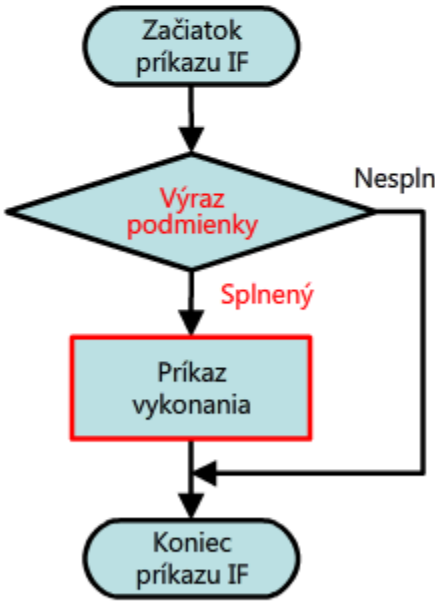
5.1 Podmienené vetvenie (IF)

Príkazy IF sa používajú na podmienené vetvenie. Príkazy IF sú opísané nižšie.

```
IF conditional expression THEN
Execution statement;          (* Príkaz sa vykoná, ak je splnený výraz podmienky. *)
END_IF;                       (* Príkaz END_IF; sa musí nachádzať na konci príkazov IF. *)
```

V tomto príklade programu sa príkaz vykoná, keď je splnený výraz podmienky. Keď nie je splnený výraz podmienky, príkaz sa nevykoná.

Na nasledujúcom obrázku je zobrazený priebeh operácie v tomto príklade programu.

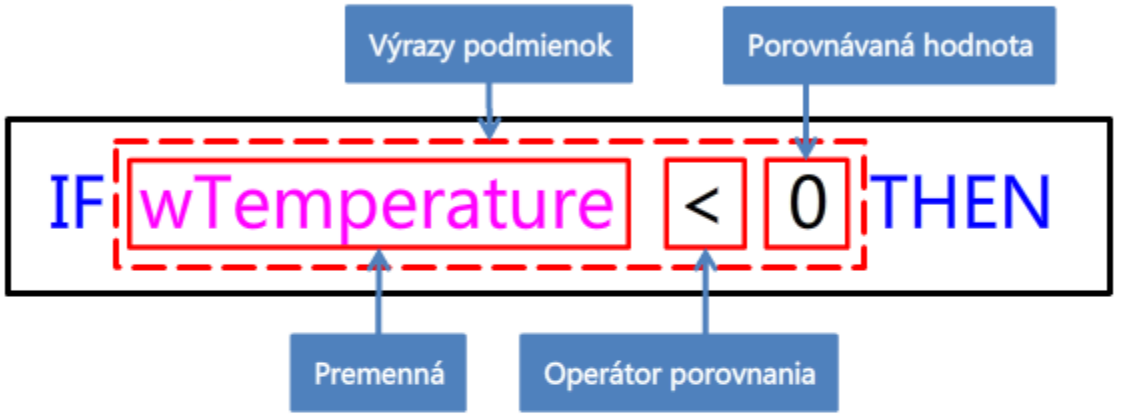


V nasledujúcom príklade je zobrazené vetvenie programu porovnávaním hodnôt premenných. V tomto príklade programu sa ohrievač zapne, keď teplota na ovládacom paneli klesne pod 0 stupňov.

```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Ohrievač sa zapne, keď teplota na ovládacom paneli klesne pod 0 stupňov. *)
END_IF;
```

5.1.1 Zápis výrazov podmienok

Na predchádzajúcej strane bol opísaný výraz podmienky „wTemperature < 0“, ktorá znamená „keď je hodnota premennej wTemperature menšia ako 0“.
Podobne ako v tomto výraze sa vo výrazoch podmienok na vyjadrenie súvzťažnosti medzi porovnávanými premennými a hodnotami používajú operátory porovnávania.



V ľavej a pravej časti operátora porovnania sú hodnoty zapísané ako premenné alebo konštanty na účely porovnania.

Okrem porovnávania premenných a konštánt možno výrazy podmienok písať na porovnávanie premenných a na vykonávanie logických operácií s výsledkami porovnávania alebo bitovými premennými.

Porovnávanie premenných

- uValue1 <= uValue2

Logická operácia pre dva výsledky porovnávania

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

Logická operácia pre dve bitové premenné

- bSwitch0 OR bSwitch1

V nasledujúcej tabuľke je uvedený zoznam typov operátorov porovnania.

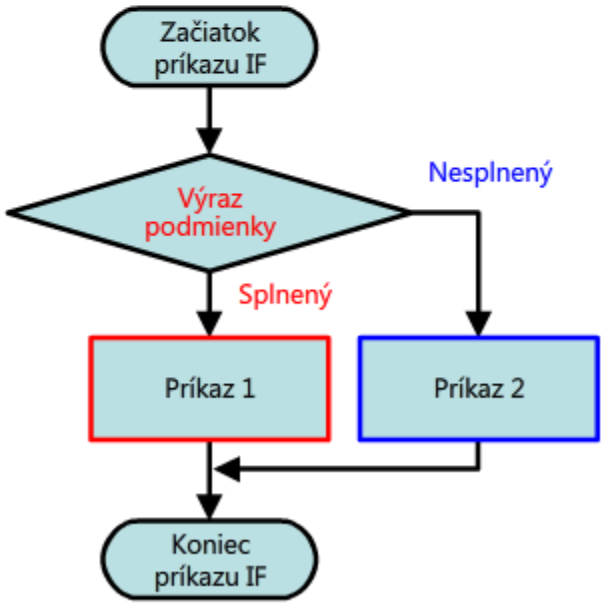
Operátor	Význam
>	Väčšie ako
<	Menšie ako
>=	Väčšie alebo rovnaké ako
<=	Menšie alebo rovnaké ako
=	Rovná sa
<>	Nerovná sa

5.1.2 Výnimka vetvenia príkazu IF (ELSE)

Jednoduché príkazy IF (pozrite si časť 5.1) slúžia na vykonanie príkazu, keď je splnený výraz podmienky.
Ak nie je splnený výraz podmienky a má sa vykonať iný príkaz, používa sa príkaz ELSE.

```
IF conditional expression THEN
  Execution statement 1;          (* Príkaz 1 sa vykoná, ak je splnený výraz podmienky. *)
ELSE
  Execution statement 2;          (* Príkaz 2 sa vykoná, ak nie je splnený príkaz podmienky *)
END_IF;
```

Na nasledujúcom obrázku je zobrazený priebeh operácie pri použití príkazu ELSE.



V nasledujúcom príklade programu sa vykonajú rôzne príkazy v závislosti od toho, či je splnená podmienka.

Príklad programu v časti 5.1 má ten nedostatok, že ohrievač zvyšuje teplotu aj po dosiahnutí 0 stupňov. V nasledujúcom programe sa však ohrievač vypne, keď hodnota „wTemperature" prekročí 0 stupňov.

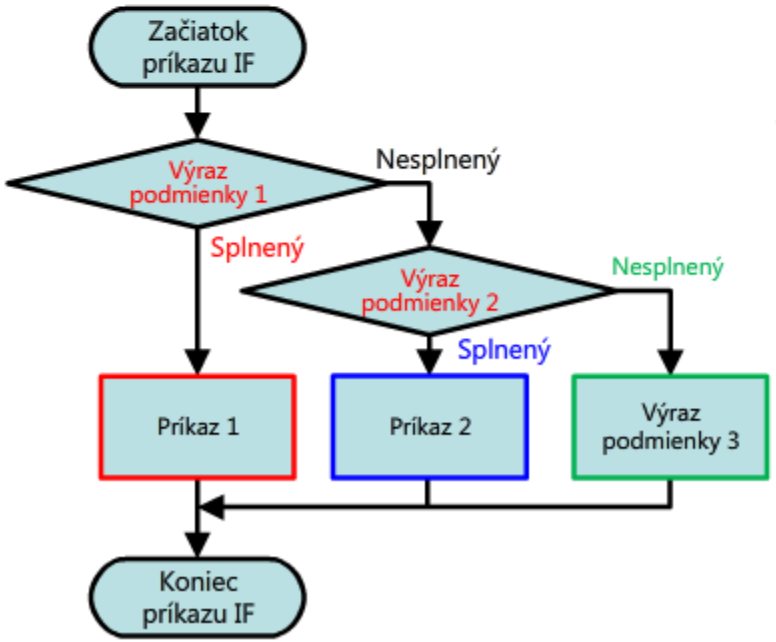
```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Zapne ohrievač, keď teplota klesne pod 0 stupňov. *)
  ELSE
  bHeater := 0;  (* Vypne ohrievač, keď teplota dosiahne alebo prekročí 0 stupňov *)
END_IF;
```

5.1.3 Ďalšie vetvenie príkazu IF (ELSIF)

Príkazy ELSE slúžia na vykonanie iného príkazu, keď nie je splnený výraz podmienky.
Pomocou príkazov ELSIF možno pridávať ďalšie podmienené vetvenie, čo znamená, že ak nie je splnený predchádzajúci výraz podmienky, overí sa ďalší výraz podmienky.

```
IF Conditional expression 1 THEN
  Execution statement 1;      (* Príkaz 1 sa vykoná, ak je splnený výraz podmienky 1. *)
  ELSIF Conditional expression 2 THEN
  Execution statement 2;      (* Príkaz 2 sa vykoná, ak výraz podmienky 1 nie je splnený a výraz podmienky 2 je splnený. *)
  ELSE
  Execution statement 3;      (* Príkaz 3 sa vykoná, ak výrazy podmienok 1 a 2 nie sú splnené. *)
END_IF;
```

Na nasledujúcom obrázku je zobrazený priebeh operácie pri použití príkazu ELSEIF.



Príkaz ELSIF sme pridali do príkladu programu v časti 5.1.2, aby sme vyriešili situáciu, keď teplota prekročí 40 stupňov.

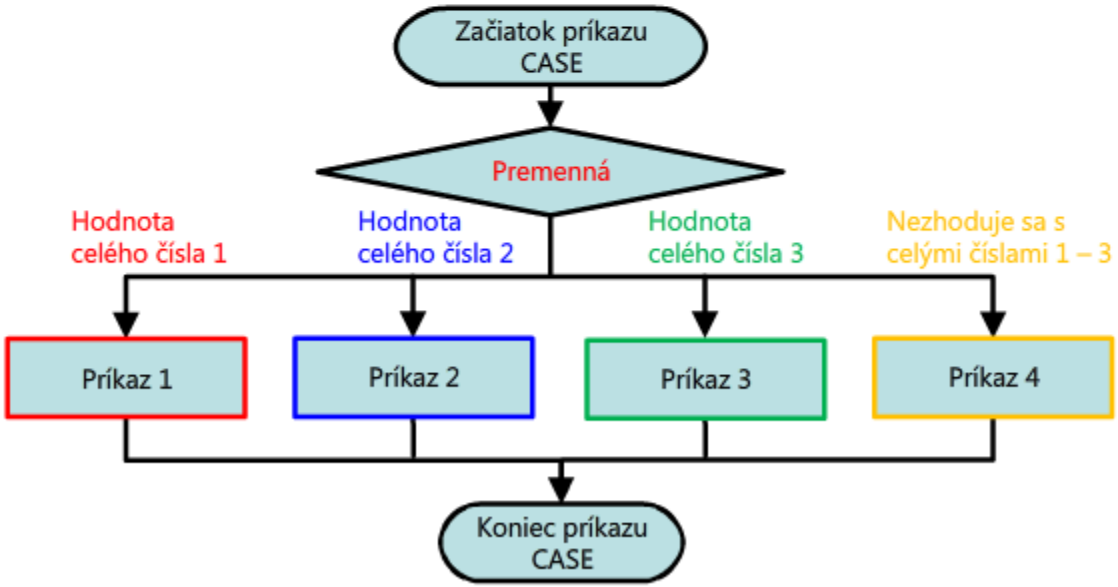
```
IF wTemperature < 0 THEN
   bHeater := 1; (* Zapne ohrievač, keď teplota klesne pod 0 stupňov. *)
   bCooler := 0; (*  Vypne chladič, keď teplota klesne pod 0 stupňov. *)
   ELSIF 40 < wTemperature THEN
   bHeater := 0; (* Vypne ohrievač, ak teplota prekročí 40 stupňov. *)
   bCooler := 1; (* Zapne chladič, ak teplota prekročí 40 stupňov. *)
   ELSE
   bHeater := 0; (*  Vypne ohrievač, ak nie je splnená žiadna z predchádzajúcich podmienok. *)
   bCooler := 0; (*  Vypne chladič, ak nie je splnená žiadna z predchádzajúcich podmienok. *)
END_IF;
```

5.2 Podmienené vetvenie podľa hodnôt celých čísel (CASE)

Príkazy IF slúžia na vetvenie v závislosti od toho, či sú splnené výrazy podmienok.
Príkazy CASE slúžia na vetvenie podľa hodnôt celých čísel.
Na nasledujúcom obrázku je zobrazený postup zápisu príkazu CASE.

```
CASE Variable OF
    Integer value 1: Execution statement 1;  (* Príkaz 1 sa vykoná, keď sa premenná zhoduje s hodnotou celého čísla 1. *)
    Integer value 2: Execution statement 2;  (* Príkaz 2 sa vykoná, keď sa premenná zhoduje s hodnotou celého čísla 2. *)
    Integer value 3: Execution statement 3;  (* Príkaz 3 sa vykoná, keď sa premenná zhoduje s hodnotou celého čísla 3. *)
    ELSE Execution statement 4;              (* Príkaz 4 sa vykoná, ak sa premenná nezhoduje so žiadnou z hodnôt celého čísla. *)
END_CASE;                                    (* Príkaz „END_CASE;" sa musí nachádzať na konci príkazu CASE. *)
```

Na nasledujúcom obrázku je zobrazený priebeh operácie pri použití príkazu CASE.



5.2.1 Príklad programu s príkazom CASE

Vykonanie príkazu CASE je opísané pomocou operácie príkladu programu.

Kliknutím na tlačidlo pokračujte na ďalšiu stranu.

Ak chcete prehrať animáciu znova, kliknite na tlačidlo „Prehrať".

Prehrať


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

Hmotnosť	uSize	Označenie
0 až 20 kg	1	M
21 až 30 kg	2	L
31 až 40 kg	3	XL
41 kg a viac	4	Oversize

5.3 Súhrn

Obsah tejto kapitoly:

- Podmienené vetvenie s príkazmi IF
- Zápis výrazov podmienok
- Podmienené vetvenie podľa hodnôt celých čísel (príkaz CASE)

Dôležité body na zváženie:

Príkaz IF	• Prostredníctvom príkazu IF sa program vetví, keď je splnený výraz podmienky. • Príkaz ELSE slúži na vetvenie, keď nie je splnený výraz podmienky. • Príkaz ELSIF slúži na pridanie ďalšieho vetvenia, keď nie je splnený výraz podmienky príkazu IF.
Výraz podmienky	• Výrazy podmienok predstavujú súvzťažnosť medzi premennými a hodnotami na účely porovnávania pomocou operátorov porovnania.
Príkaz CASE	• Príkazy CASE slúžia na vetvenie podľa hodnôt celých čísel.

Kapitola 6 Ukladanie a spracúvanie údajov

Rovnako ako v prípade riadiacich aplikácií I/O, aj súčasné programovateľné radiče slúžia na spracúvanie veľkých objemov údajov, ktoré tvoria jadro výrobných systémov.
Ak sa majú spracúvať veľké objemy údajov, údaje musia byť uložené a potom sa musia podľa potreby čítať.
Táto kapitola opisuje postup písania stručných programov na ukladanie a spracúvanie údajov.

- Polia slúžia na vytváranie sekvencií a organizovanie premenných.
- Štruktúry údajov slúžia na organizovanie súvisiacich premenných.
- Programy spracúvajúce slučky efektívne spracúvajú polia pomocou príkazov FOR.

Pomocou polí, štruktúr údajov a príkazov FOR možno vytvárať stručné programy na ukladanie a spracúvanie údajov.

6.1 Vytváranie sekvencií a ukladanie údajov (pole)

Pomocou polí môže jedna premenná spracovať viacero hodnôt.
V nasledujúcom príklade sa údaje o objeme výroby v závode na výrobu automobilov ukladajú v cieľovej krajine.

Cieľová krajina	Krajina A	Krajina B	Krajina C
Objem výroby	35	75	65

Údaje o objeme výroby príslušnej cieľovej krajiny sa prideľujú určitej premennej. Bez použitia polí sa musí pre každú cieľovú krajinu vytvoriť jedna premenná.
Pomocou polí však možno objem výroby vo viacerých cieľových krajinách prideliť jednej premennej a uložiť v nej tieto údaje.

Bez použitia poľa

uProductionA
uProductionB
uProductionC

S použitím poľa

uProduction

Jednotlivé premenné v poli sú špecifikované pomocou čísel prvkov. Čísla prvkov začínajú nulou [0].

V nasledujúcom príklade programu je pridelená premenná plánovaného objemu výroby v krajine A.

```
uShowProductionPlan := uProduction[0];
(* Špecifikuje číslo prvku pre krajinu A. *)
```

6.1.1 Maticové pole

V tejto časti sa okrem údajov o cieľovej krajine používajú aj údaje o farbe náteru.

Cieľová krajina									
Farba náteru									
Objem výroby	10	5	20	15	40	20	25	30	10
	Celkovo 35			Celkovo 75			Celkovo 65		

Ako je zobrazené v nasledujúcej tabuľke, údaje možno oddeliť a uložiť podľa farby náteru (stĺpec) pre jednotlivé cieľové krajiny (riadok).

Farba náteru (stĺpec)

Cieľová krajina (riadok)	Červená	Žltá	Modrá
Krajina A	[0,0] 10	[0,1] 5	[0,2] 20
Krajina B	[1,0] 15	[1,1] 40	[1,2] 20
Krajina C	[2,0] 25	[2,1] 30	[2,2] 10

Číslo prvku predstavuje cieľovú krajinu

Číslo prvku predstavuje farbu náteru

Polia, v ktorých sa týmto spôsobom organizujú údaje do riadkov a stĺpcov, sú známe ako maticové polia. Čísla prvkov predstavujúce riadky a stĺpce sú oddelené čiarkami.

Premenná poľa (maticové pole)

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

6.1.2 Pridelenie maticového poľa

Pomocou maticových polí je v nasledujúcom príklade programu pridelený počet automobilov, ktorý sa musí urgentne vyrobiť navyše k plánovanému objemu výroby žltých automobilov v krajine B.

```
uAdditionalProduction := 5;
uProduction[1,1] := uProduction[1,1] + uAdditionalProduction;
(* Pridá ďalší objem výroby (5 jednotiek) k pôvodnému plánovanému objemu výroby. *)
```

Farba náteru (stĺpec)

Cieľová krajina (riadok)	Červená	Žltá	Modrá
Krajina A	[0,0] 10	[0,1] 5	[0,2] 20
Krajina B	[1,0] 15	[1,1] 40 -> 45	[1,2] 20
Krajina C	[2,0] 25	[2,1] 30	[2,2] 10

6.1.3 Spracovanie informácií uložených v maticových poliach

Pomocou maticových polí sa v nasledujúcom príklade programu vypočíta celkový objem výroby plánovaný pre všetky farby náteru v krajine C a príslušnej premennej sa pridelí konkrétna hodnota.

```
uProductionToday := uProduction[2,0] + uProduction[2,1] +uProduction[2,2];
(* Vypočíta celkový plánovaný objem výroby na dnes pre všetky farby náteru v krajine C a pridelí príslušnú hodnotu premennej „uProductionToday“. *)
```

Cieľová krajina	Krajina A			Krajina B			Krajina C		
Farba náteru									
Objem výroby	10	5	20	15	45	20	25	30	10
	Celkovo 35			Celkovo 80			Celkovo 65		

Farba náteru (stĺpec)

Cieľová krajina (riadok)	Červená	Žltá	Modrá
Krajina A	[0,0] 10	[0,1] 5	[0,2] 20
Krajina B	[1,0] 15	[1,1] 45	[1,2] 20
Krajina C	[2,0] 25	[2,1] 30	[2,2] 10

6.2 Vytváranie slučiek (FOR)

Nižšie sa znova zobrazuje príklad programu z predchádzajúcej strany (pridelený plánovaný objem výroby na dnes).

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

V tomto príklade programu platí, že keď sa zvýši počet farieb náteru, pridajú sa ďalšie premenné. Následne je výraz dlhší a ťažšie čitateľný.

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

V tomto prípade možno na vytvorenie zreteľnejšieho kódu použiť príkazy slučky.

Príkazy slučky zahŕňajú príkazy FOR, WHILE a REPEAT. V tomto kurze sa zameriame na príkazy FOR.

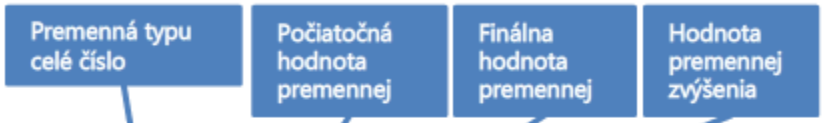
Príkazy FOR sú opísané nižšie.

```
FOR variable := initial value TO final value BY increments DO
  Execution statement; (* Príkaz sa vykonáva v slučke, kým premenná nedosiahne finálnu hodnotu. *)
END_FOR;               (* Príkaz END_FOR; sa musí nachádzať na konci príkazov FOR.  *)
```

Príkaz sa opakuje, kým sa nedosiahne finálna hodnota premennej, a vykoná sa kód „END_FOR;".

6.2 Vytváranie slučiek (FOR)

Pomocou príkazu FOR sa v nasledujúcom príklade programu dosiahne plánovaný objem výroby pre všetky farby náteru v krajine C.




```
uProductionToday := 0;                          (* Inicializuje príslušnú premennú. *)
FOR wColor:=0 TO 2 BY 1 DO
  uProductionToday := uProductionToday + uProduction[2,wColor];   (* Pridá plánovaný objem výroby. *)
END_FOR;
```

Použitím príkazu FOR sa hodnota premennej „wColor" v porovnaní s počiatočnou hodnotou nula zvýši o jednu a príkaz sa opakuje, kým premenná nedosiahne finálnu hodnotu dva.
Premenná „wColor" je v poli „uProduction" opísanom v príkaze vykonania špecifikovaná ako druhé číslo prvku.
Hodnota premennej „wColor" sa zvyšuje po každom opakovaní príkazu. Plánovaný objem výroby každej farby náteru sa pridá po každom dosiahnutí celkového objemu.

Tento príklad programu sa zopakuje trikrát v slučke. (Prvýkrát: červená [0] => druhýkrát: žltá [1] => tretíkrát: modrá [2])

Operácia tohto programu je zobrazená na ďalšej strane.

6.2 Vytváranie slučiek (FOR)

Vykonanie príkazu FOR je opísané prostredníctvom operácie príkladu programu.

Pole odhadovaného objemu výroby

	Červená	Žltá	Modrá
Krajina A	[0,0] 10	[0,1] 5	[0,2] 20
Krajina B	[1,0] 15	[1,1] 45	[1,2] 20
Krajina C	[2,0] 25	[2,1] 30	[2,2] 10

Kliknutím na tlačidlo pokračujte na ďalšiu stranu.
Ak chcete prehrať animáciu znova, kliknite na tlačidlo „Prehrať".

Prehrať

Number of repetition of the loop: 3

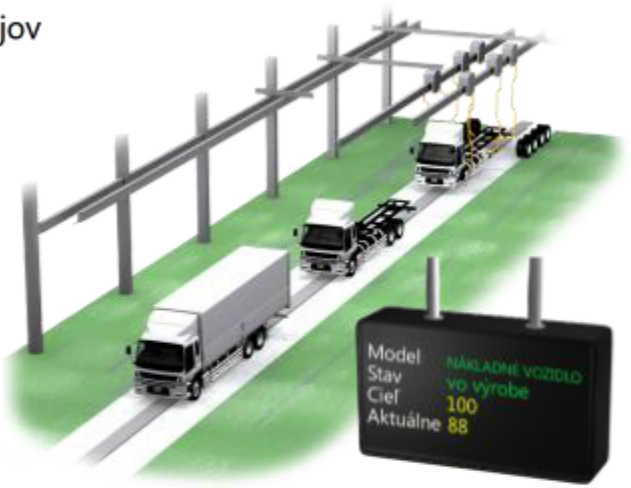
```
uProductionToday := 0;

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

6.3 Ukladanie súvisiacich údajov (štruktúra)

Štruktúra umožňuje, aby jeden názov premennej predstavoval viacero súvisiacich premenných.
V nasledujúcom príklade je stav automobilovej výrobnej linky zobrazený na displeji Andon (panelovom displeji).

V nasledujúcej tabuľke je uvedený zoznam názvov premenných, hodnôt a typov údajov zodpovedajúcich zobrazeným položkám.

Položka	Názov premennej	Hodnota	Typ údajov premennej
Model	sModel	„ST TRUCK "	Textový reťazec
Stav	bStatus	„vo výrobe"	Bitový typ
Cieľový objem výroby na dnes	uPlanQty	„100"	Typ celého čísla – slovo (nepriradené)
Aktuálny objem výroby	uActualQty	„88"	Typ celého čísla – slovo (nepriradené)



Ak sa nepoužíva štruktúra, v prípade existencie viacerých výrobných liniek sa musia názvy premenných meniť pre každú linku.
V nasledujúcej časti sú zobrazené príklady názvov premenných podľa výrobnej linky.

Prvá výrobná linka

s1stLineModel
b1stLineStatus
u1stLinePlanQty
u1stLineActualQty

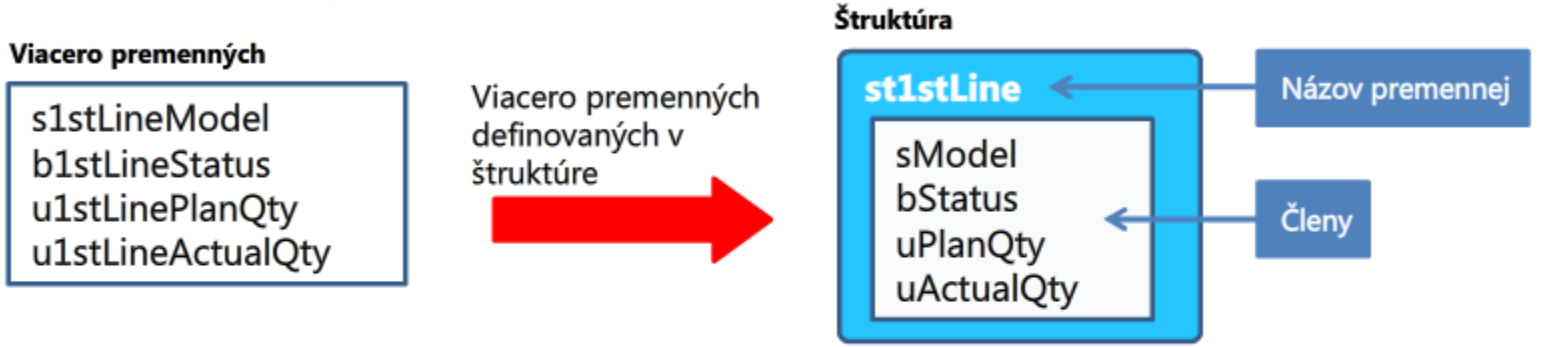
Druhá výrobná linka

s2ndLineModel
b2ndLineStatus
u2ndLinePlanQty
u2ndLineActualQty

Keď sa zvýši počet výrobných liniek, zároveň sa zvýši aj počet premenných, ktoré sa majú spracovať.
Následne je program dlhší a ťažšie čitateľný.

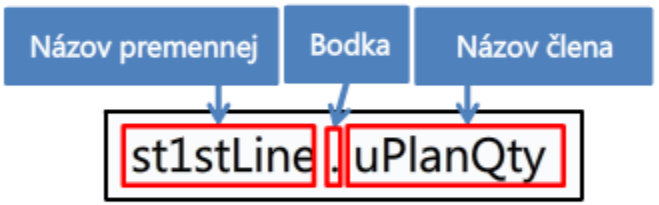
6.3 Ukladanie súvisiacich údajov (štruktúra)

Používanie štruktúr umožňuje, aby jeden názov premennej predstavoval viacero premenných súvisiacich s jednou výrobnou linkou. Týmto spôsobom sa štruktúry používajú na hromadné organizovanie, ukladanie a spracúvanie údajov súvisiacich s podmienkami a špecifikáciami fyzických objektov, ako sú napríklad zariadenia, vybavenie a obrobky.



Parameter **st**ructure variable (premenná štruktúry) obsahuje predponu „st“, ktorá signalizuje, že ide o štruktúru. Jednotlivé premenné definované štruktúrou sú známe ako členy. Typy údajov jednotlivých členov sa môžu líšiť.

Každého člena polí štruktúry možno špecifikovať po čísle prvku poľa, a to zadaním bodky pred názov člena.

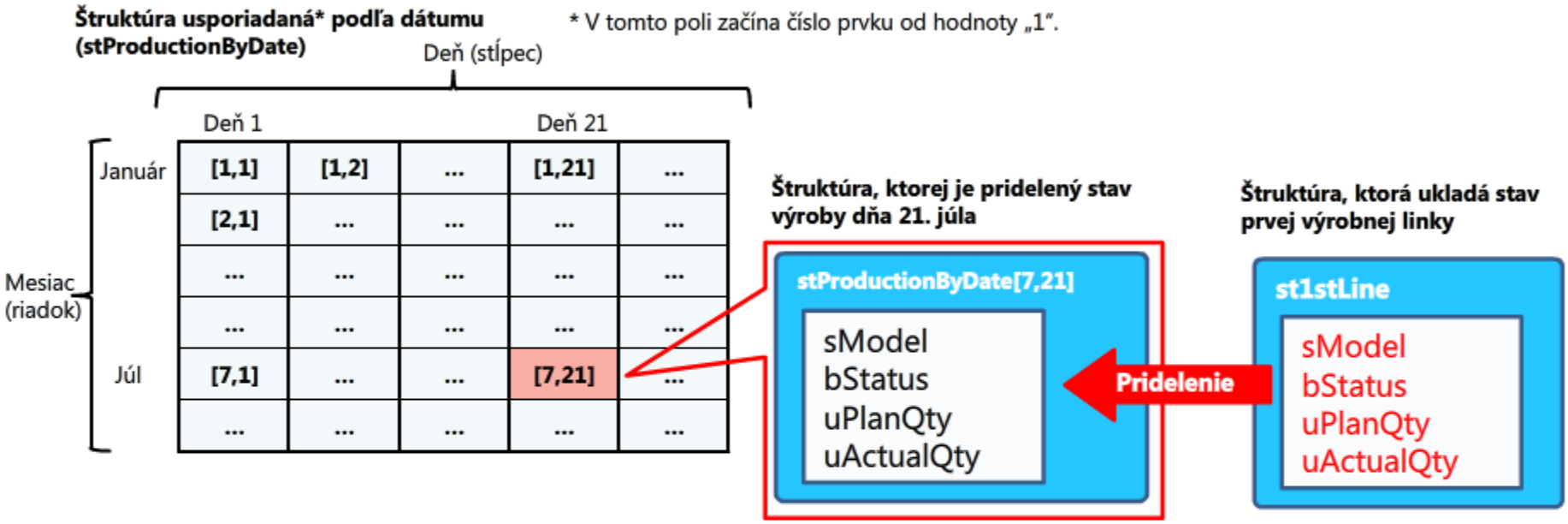


V nasledujúcom príklade programu je členovi premennej štruktúry pre prvú výrobnú linku pridelená konštanta.

```
st1stLine.uPlanQty := 150;
 (* Stanovuje dnešnú cieľovú výrobu prvej výrobnej linky na počet 150. *)
```

6.3.1 Ukladanie polí štruktúry

Štruktúry možno vytvárať ako polia.
V nasledujúcom príklade sa stav výroby ukladá podľa dátumu.

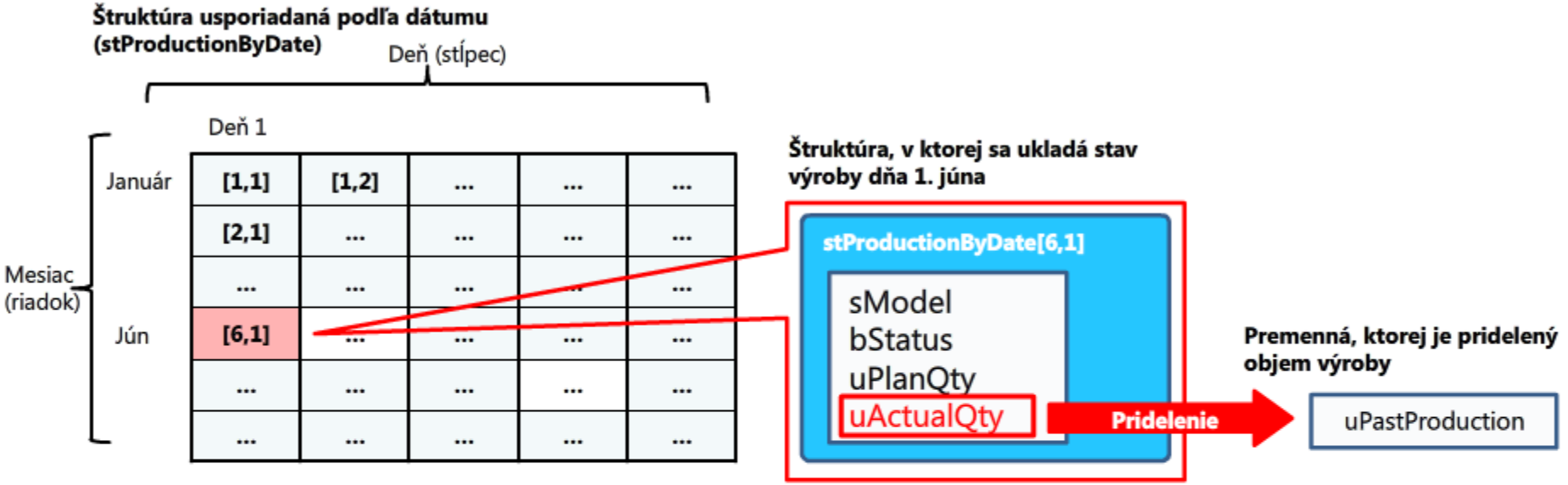



```
stProductionByDate[7,21] := st1stLine;
 (* Stav výroby dňa 21. júla je uložený v štruktúre usporiadanej podľa dátumu
(stProductionByDate). *)
```

Týmto spôsobom nie je potrebné individuálne špecifikovať členy pri prideľovaní štruktúry.

6.3.2 Čítanie polí štruktúry

V nasledujúcom príklade sa objem výroby číta zo štruktúry usporiadanej podľa dátumu a potom pridelenej k premennej.




```
uPastProduction := stProductionByDate[6,1].uActualQty;
(* Prideľuje objem výroby dňa 1. júna premennej uPastProduction. *)
```

Každého člena polí štruktúry možno špecifikovať pridaním bodky (.) do názvu člena v čísle prvku poľa.

6.4 Súhrn

Obsah tejto kapitoly:

- Prehľad a používanie polí
- Spracúvanie slučiek pomocou príkazov FOR
- Prehľad a používanie štruktúr

Dôležité body na zváženie:

Pole	• Viaceré hodnoty možno spracúvať jednou premennou, a to použitím polí. • Jednotlivé premenné v poliach sú špecifikované číslami prvkov pridanými na koniec názvov premenných.
Príkaz FOR	• Príkazy slučky sa používajú v programoch, v ktorých sa vyžaduje opakovaná operácia. • Príkazy FOR slúžia na opakovanie operácie dovtedy, kým sa nesplnia podmienky na konci operácie slučky. Príkaz pred príkazom „END_FOR;" sa vykonáva opakovane.
Štruktúra	• Štruktúry umožňujú, aby jeden názov premennej predstavoval viacero súvisiacich premenných. Štruktúry môžu zahŕňať premenné rôznych typov údajov. • Jednotlivé premenné alebo členy definované v štruktúrach sa špecifikujú pridaním bodky a názvu člena po názve premennej štruktúry.

Kapitola 7 Spracúvanie reťazcových údajov

V niektorých prípadoch využívajú programovateľné radiče reťazcové údaje na odosielanie príkazov do pripojených zariadení, ako sú napríklad čítačky čiarových kódov, regulátory teploty alebo elektronické váhy, alebo na prijímanie spätnej väzby z týchto zariadení. Na takéto účely je potrebné reťazcové údaje spájať alebo extrahovať.

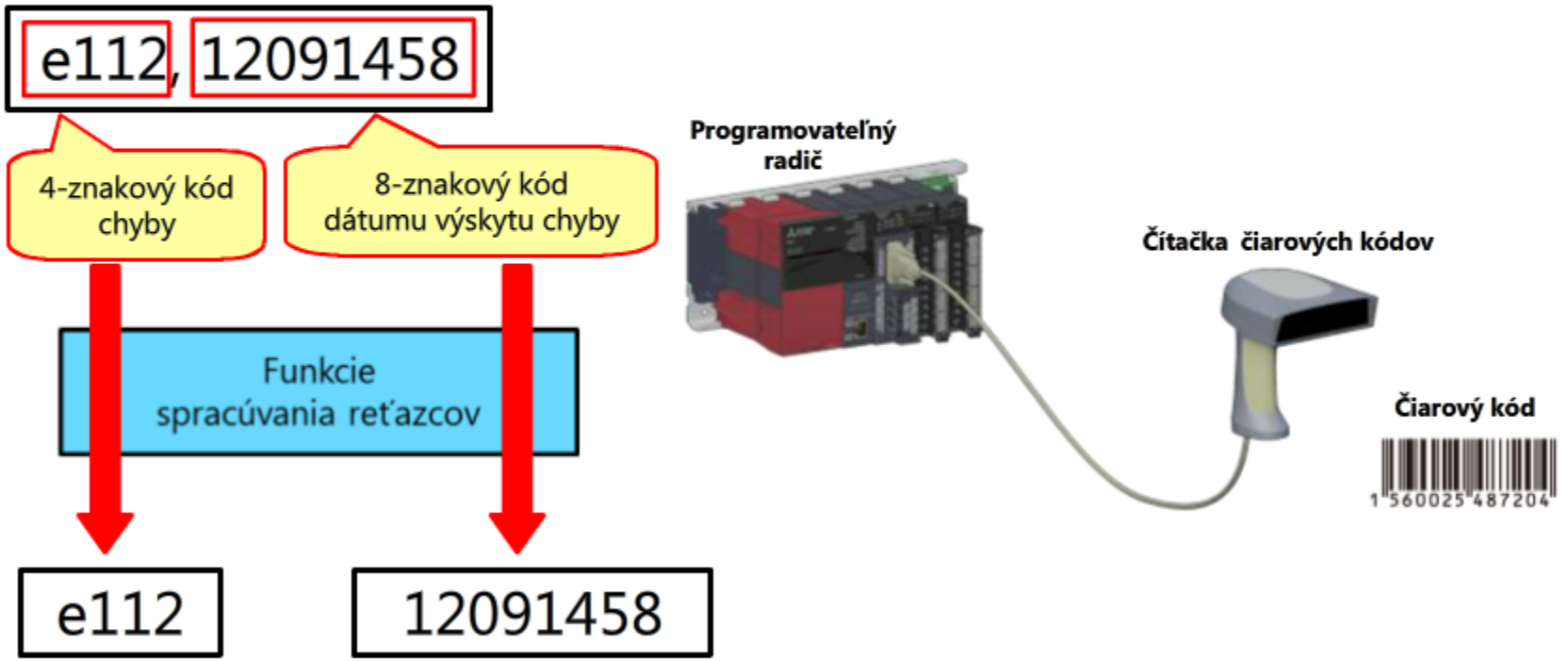
Táto kapitola opisuje postup spracúvania reťazcových údajov.

7.1 Príklad spracúvania reťazcových údajov

Ako príklad spracúvania reťazcov použijeme scenár, pri ktorom sa údaje čítajú z čítačky čiarových kódov.
Na spracúvanie reťazcov sa používajú funkcie (typ inštrukcií).

Ako je zobrazené nižšie, reťazce čítané pomocou čítačky čiarových kódov obsahujú 4-znakový kód chyby s pevnou dĺžkou a 8-znakové údaje o mesiaci, dátume, čase a minúte s pevnou dĺžkou.
Príklad programu spracúvania reťazcov opíšeme pomocou tohto systému.

Príklad čítania reťazcových údajov z čítačky čiarových kódov



Extrahuje sa kód chyby.
7.3 Extrahovanie reťazcov (LEFT)

Extrahuje sa dátum a čas výskytu chyby (14:58, 9. december).
7.4 Extrahovanie reťazcov (MID)

7.2 Pridelenie reťazca

Skôr ako si vysvetlíme postup extrahovania reťazcov, v tejto časti sa oboznámime s typmi údajov reťazcov.

Typy údajov reťazcov, ktoré možno používať s programovateľnými radičmi, sú uvedené v nasledujúcej tabuľke.

Typ údajov	Typ znaku možno spracovať	Predpony maďarského zápisu	Rozpísaný názov predpony
Reťazec	Reťazce alfanumerických znakov a čísla (ASCII) alebo japonské znaky (Shift-JIS)	s	**s**tring (reťazec)
Reťazec [Unicode]	Reťazce pozostávajúce z rôznych jazykov a symbolov	ws	**w**ide **s**tring (široký reťazec)

Typ reťazca, ktorý sa má použiť, závisí od zariadenia pripojeného k programovateľnému radiču alebo od príslušného jazyka.
Táto kapitola opisuje rôzne typy textových reťazcov.

Po pridelení typu reťazca premennej reťazca napíšte reťazec medzi jednoduché úvodzovky (').

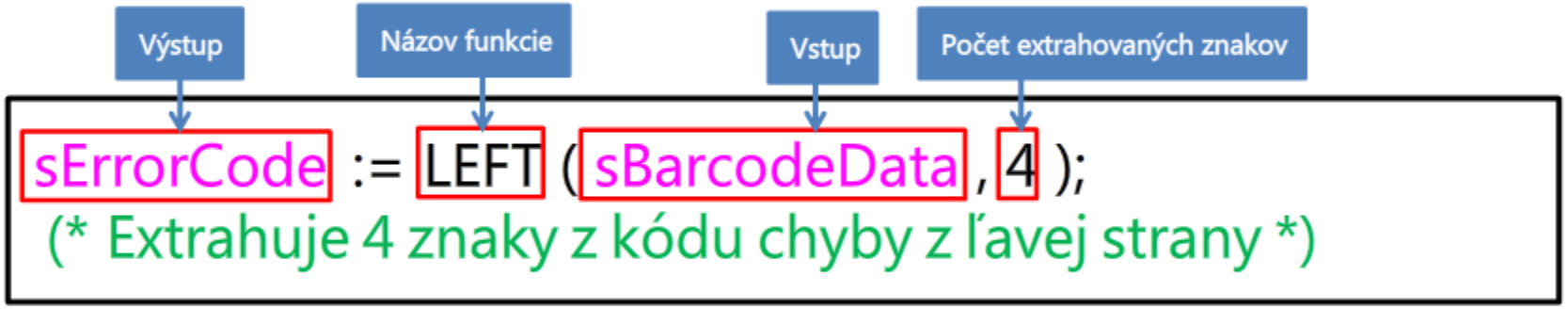
```
sDefault := 'e112,12091458';  (* Pridelenie reťazca *)
```

7.3 Extrahovanie reťazcov (LEFT)

Kód chyby „e112“ sa extrahuje z premennej reťazca „sBarcodeData“, ktorá obsahuje reťazec „e112,12091458“.

Názov premennej	Uložený reťazec
sBarcodeData	e112, 12091458

Funkcia LEFT extrahuje len špecifikovaný počet znakov, a to z ľavej strany vstupného reťazca.
Uvádzame príklad programu.



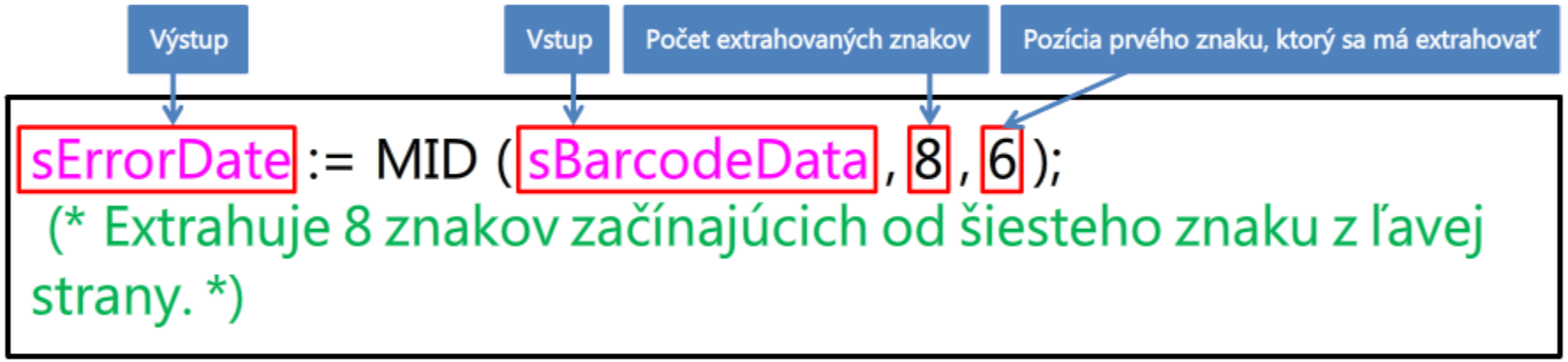
Extrahujú sa štyri znaky zľava. Hodnota „e112“, teda reťazec predstavujúci kód chyby, je pridelená ľavej strane.

7.4 Extrahovanie reťazcov (MID)

Čas výskytu chyby „12091458" sa extrahuje z premennej reťazca „sBarcodeData", ktorá obsahuje reťazec „e112,12091458".

Názov premennej	Uložený reťazec
sBarcodeData	e112,12091458

Funkcia MID extrahuje špecifikovaný počet znakov od špecifikovanej pozície začiatku vo vstupnom reťazci.
Uvádzame príklad programu.



V tomto príklade sa extrahuje 8-znakový reťazec začínajúci od šiesteho znaku. Hodnota „12091458", teda reťazec predstavujúci čas výskytu chyby, je pridelená ľavej strane.

7.5 Súhrn

Obsah tejto kapitoly:

- Metódy prideľovania reťazcov premenným reťazcov
- Funkcie na extrahovanie reťazcov (LEFT a MID)

Dôležité body na zváženie:

Pridelenie reťazca	• Ak chcete prideliť premennú reťazca, napíšte reťazec medzi jednoduché úvodzovky ('). • Použite príslušný typ reťazca alebo typ reťazca [Unicode] v závislosti od zariadenia pripojeného k programovateľnému radiču alebo od príslušného jazyka.
Funkcie na spracúvanie reťazcov	• Na spracúvanie reťazcov sa používajú funkcie.

7.6 Súhrn kurzu

Tento kurz sa zameriava na základné pravidlá vytvárania programov v jazyku ST.
Dostávame sa tak na koniec tohto elektronického kurzu.

Programy v jazyku ST sa vytvárajú pomocou technického softvéru MELSOFT.
Podrobnosti o špecifických krokoch, napríklad o zadávaní údajov či upravovaní, ukladaní a vytváraní programov pomocou technického softvéru MELSOFT, nájdete v nasledujúcich zdrojoch.

- Elektronický kurz Mitsubishi FA „MELSOFT GX Works3 (Structured Text)"
  (Softvér MELSOFT GX Works3 (štruktúrovaný text)) (dostupný už čoskoro)
- Prevádzková príručka k technickému softvéru MELSOFT

Ďalšie informácie o jazyku ST nájdete v nasledujúcich zdrojoch.

- Sprievodca programovaním k programovateľnému radiču

Informácie o inštrukciách a funkciách pre vašu aplikáciu nájdete v nasledujúcich zdrojoch.

- Príručka programovania k programovateľnému radiču

Test Záverečný test

Teraz, keď ste dokončili všetky lekcie kurzu Základné informácie o programovaní (štruktúrovaný text), ste pripravení na záverečný test. Ak si nie ste istí niektorými preberanými témami, využite túto príležitosť a zopakujte si ich.
Celkovo je v tomto záverečnom teste 12 otázok (20 položiek).
Záverečný test môžete absolvovať ľubovoľne veľakrát.

Hodnotenie testu

Po výbere odpovede kliknite na tlačidlo **Odpovedať**. Ak prejdete na ďalšiu otázku bez kliknutia na tlačidlo Odpovedať, vaša odpoveď sa nezapočíta. (Považuje sa za nezodpovedanú otázku.)

Výsledky testu

Na stránke výsledkov sa zobrazí počet odpovedí, percentuálna úspešnosť a výsledok úspešnosti/neúspešnosti absolvovania.

Správne odpovede: 4

Celkový počet otázok: 4

Percentuálna úspešnosť: 100%

Na úspešné absolvovanie testu musíte správne zodpovedať **60%** otázok.

Pokračovať Skontrolovať

- Kliknutím na tlačidlo **Pokračovať** sa test ukončí.
- Kliknutím na tlačidlo **Skontrolovať** si môžete test skontrolovať. (Kontrola správnych odpovedí)
- Kliknutím na tlačidlo **Znova** môžete test absolvovať znova.

Test Záverečný test 1

Znaky štruktúrovaného textu (jazyka ST)
Vyberte nesprávny opis jazyka ST.

- Jazyk ST sa ľahko naučia používať programátori so skúsenosťami s písaním programov v jazyku C alebo BASIC.
- Výpočty, napríklad sčítanie a odčítanie, možno napísať ako bežne používané matematické výrazy.
- Symboly pre kontakty a cievky slúžia na vytvorenie programu pripomínajúceho elektrický obvod.
- Jazyk ST je vhodný na spracúvanie údajov.

Odpovedať Späť

Test Záverečný test 2

Základné princípy jazyka ST
Vyberte správny príkaz napísaný v jazyku ST.

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

Odpovedať Späť

Test — Záverečný test 3

Opis komentárov
Vyberte správny komentár napísaný v jazyku ST.

- ' Prideľuje premennej hodnotu 1.
- (* Prideľuje premennej hodnotu 1. *)
- { Prideľuje premennej hodnotu 1. }
- <!-- Prideľuje premennej hodnotu 1. -->

Odpovedať Späť

Test Záverečný test 4

Spúšťacia sekvencia programu v jazyku ST
* Počiatočná hodnota premennej „uTotalProduction“ je „100“. Hodnota premennej „uTotalProduction“ po spracovaní nasledujúceho príkladu programu bude „101“. Vyberte správny stav premennej „uTotalProduction“ po uplynutí niekoľkých sekúnd.

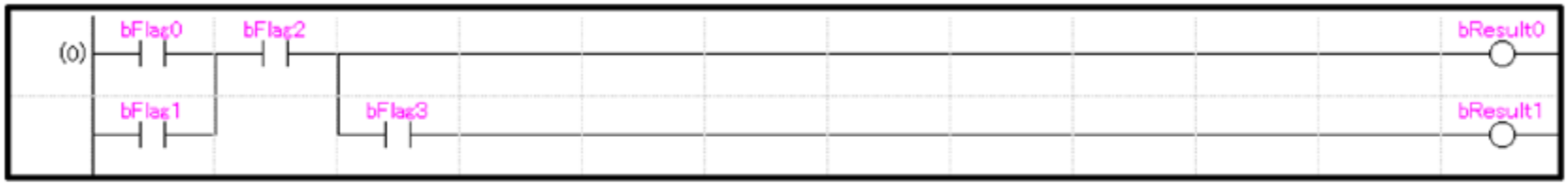
```
uTotalProduction := uTotalProduction + 1;
```

- ○ Hodnota 101 sa nezmení.
- ○ Hodnota sa neustále mení.

Odpovedať Späť

Test Záverečný test 5

Kombinovanie viacerých podmienok
Vyberte správny príklad programu v jazyku ST, ktorý predstavuje rovnakú operáciu ako nasledujúci príklad programu v jazyku LD.



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

Odpovedať Späť

Test Záverečný test 6

Opis príkazov IF v jazyku ST
Nasledujúca operácia sa vykoná v príklade programu uvedenom nižšie.
•Ak teplota klesne na 5 stupňov alebo menej, ohrievač sa zapne a chladič sa vypne.
•Ak teplota prekročí 50 stupňov, ohrievač sa vypne a chladič sa zapne.
•Ak sa teplota netýka vyššie uvedených príkazov, ohrievač aj chladič sa vypnú.

* Názvy premenných: teplota (wTemperature), ohrievač (bHeater) a chladič (bCooler)
Pre každú prázdnu časť príkladu programu vyberte správnu možnosť.

```
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;
```

Otázka 1 --Select--

Otázka 2 --Select--

Otázka 3 --Select--

Otázka 4 --Select--

Otázka 5 --Select--

Odpovedať Späť

Test — Záverečný test 7

Príkazy CASE

V nasledujúcom opise príkazov CASE vyberte správnu možnosť pre každú otázku (Otázka 1 až Otázka 5).

Príkazy CASE slúžia na vetvenie podľa hodnoty (Otázka 1).
Keď je v nasledujúcom príklade programu hodnota (Otázka 2) 25, premenná (Otázka 3) je pridelená hodnote (Otázka 4).
Keď sa hodnota premennej (Otázka 2) nerovná 10, 25 ani 8, premenná (Otázka 3) je pridelená hodnote (Otázka 5).

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

Otázka 1 --Select--

Otázka 2 --Select--

Otázka 3 --Select--

Otázka 4 --Select--

Otázka 5 --Select--

Odpovedať Späť

Test — Záverečný test 8

Polia v jazyku ST a opakované príkazy
V nasledujúcom príklade programu sa uvádza celkový plánovaný objem výroby všetkých modelov v cieľovej krajine Y a potom sa táto hodnota prideľuje príslušnej premennej. Vyberte časť poľa, ktoré sa číta po vykonaní príkazu FOR v 3 slučkách.

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

Pole slúžiace na ukladanie odhadovaného počtu vyrobených jednotiek pre jednotlivé modely a cieľové krajiny (uProduction)

Cieľová krajina (riadok)	Model 1	Model 2	Model 3	Model 4
Krajina X	[0,0]	[0,1]	[0,2] C	[0,3]
Krajina Y	[1.0]	[1.1] A	[1.2] D	[1.3] E
Krajina Z	[2,0]	[2,1] B	[2,2]	[2,3]

Model (stĺpec)

- A
- B
- C
- D

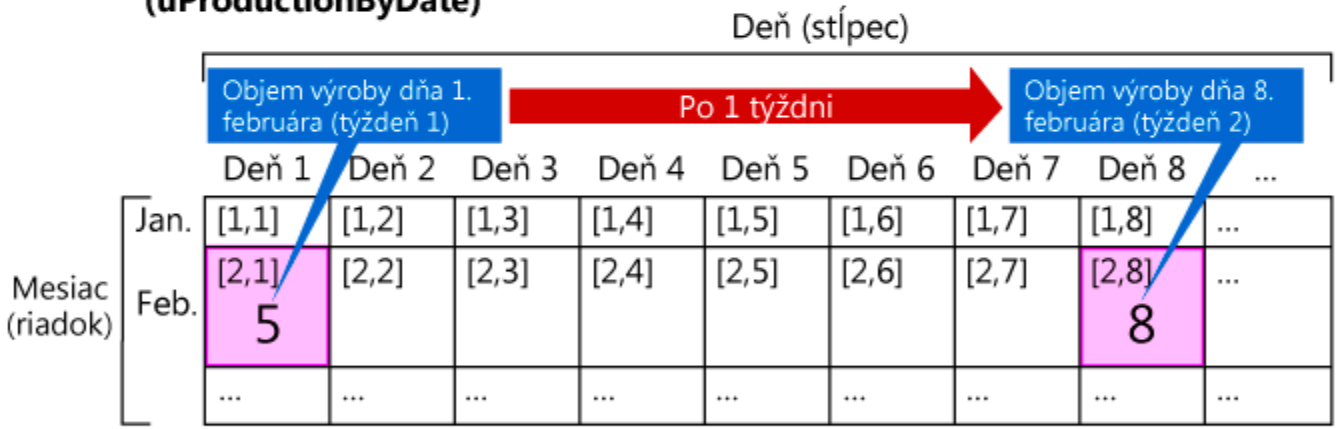
Odpovedať | Späť

Test — Záverečný test 9

Polia v jazyku ST a opakované príkazy
Nasledujúci príklad programu uvádza celkový objem výroby v rovnaké dni v týždni. Celkový objem za 4 týždne sa získal z poľa, v ktorom sa ukladá objem výroby za deň. Vyberte v príklade programu správne číslo.

```
uTotalProduction := 0;
FOR wOnceAWeek := 1 TO ■ BY 7 DO
  uTotalProduction := uTotalProduction + uProductionByDate[2,wOnceAWeek];
END_FOR;
(* Extrahuje a spočíta celkový objem výroby v rovnaké dni v týždni za 4 týždne začínajúce od 1. februára. *)
```

Pole, v ktorom sa ukladajú údaje o objeme výroby na deň (uProductionByDate)



- 22
- 21
- 4
- 28

Odpovedať Späť

Test Záverečný test 10

Znaky štruktúr v jazyku ST
Vyberte nesprávny opis štruktúr.

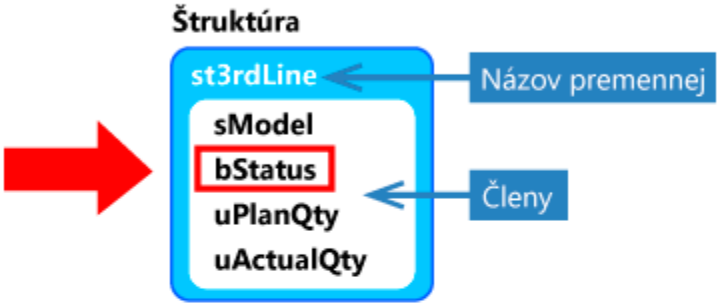
- Štruktúry slúžia na organizovanie a ukladanie údajov v zariadeniach, a to podľa podmienok, napríklad podľa stavu a špecifikácií.
- Programy spracúvajúce veľké objemy údajov možno napísať stručne pomocou štruktúr.
- Všetky členy definované v štruktúre musia mať rovnaký typ údajov.
- Hodnoty možno prideľovať členom v rovnakej štruktúre bez toho, aby ich bolo potrebné špecifikovať individuálne.

Odpovedať Späť

Test — Záverečný test 11

Špecifikovanie členov štruktúr v jazyku ST
Nasledujúca štruktúra organizuje premenné súvisiace s automobilovou výrobnou linkou.
Vyberte správny opis špecifikácie člena „bStatus“ v tejto štruktúre.

Parameter	Názov premennej
Model	sModel
Stav	bStatus
Cieľová výroba na dnes	uPlanQty
Aktuálny objem výroby	uActualQty

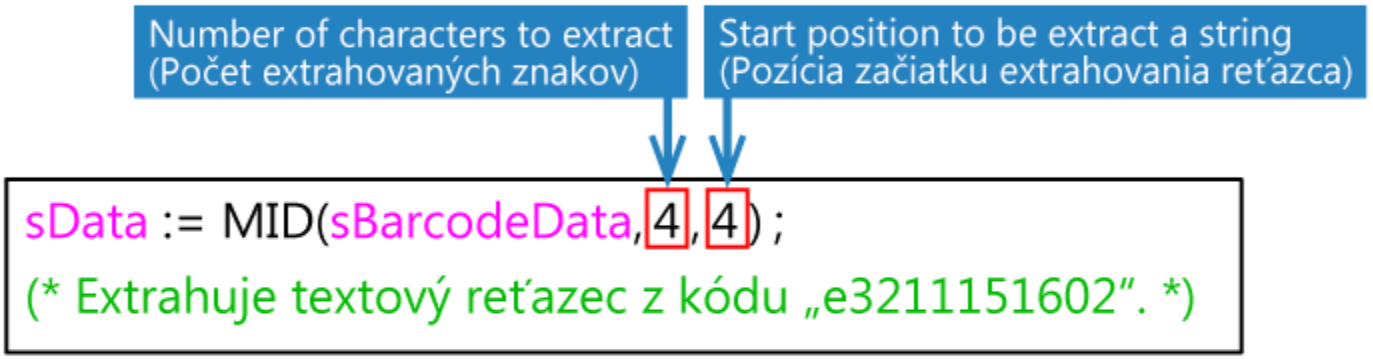


- st3rdLine.bStatus
- st3rdLine->bStatus
- st3rdLine[bStatus]
- st3rdLine[1]

Odpovedať Späť

Test — Záverečný test 12

Spracúvanie reťazcov v jazyku ST
V nasledujúcom príklade programu sa extrahuje špecifický reťazec z reťazca „e3211151602" uloženého v premennej „sBarcodeData". Funkcia MID extrahuje špecifikovaný počet znakov začínajúci od špecifikovaného bodu začiatku.
Vyberte správne extrahovaný reťazec.



- 1151
- 1602
- e321
- 1115

Odpovedať | Späť

Test Vyhodnotenie testu

Dokončili ste záverečný test. Vaše výsledky sú uvedené nižšie.
Ak chcete ukončiť záverečný test, prejdite na ďalšiu stranu.

Správne odpovede:	12
Celkový počet otázok:	12
Percentuálna úspešnosť:	100%

Pokračovať Skontrolovať

Blahoželáme. Uspeli ste v teste.

Dokončili ste kurz Základné informácie o programovaní (štruktúrovaný text).

Ďakujeme, že ste absolvovali tento kurz.

Veríme, že sa vám lekcie páčili a informácie získané v tomto kurze budú pre vás v budúcnosti užitočné.

Kurz môžete absolvovať podľa potreby viac krát.

Skontrolovať Zavrieť