MITSUBISHI ELECTRIC
Changes for the Better

PLC

Základy programování (Strukturovaný text)

Tento kurz vysvětluje, jak vytvářet základní programy pro řízení programovatelných řadičů MELSEC.
K popisu programu je v tomto kurzu použit strukturovaný text (ST).

L(FOD)00545CZE

Úvod

Účel kurzu

Tento kurz pomocí strukturovaného textu (ST) vysvětluje, jak vytvářet řídicí programy pro programovatelné řadiče MELSEC.

Předběžnou podmínkou absolvování tohoto kurzu je dokončení následujícího kurzu nebo přehled o ekvivalentních znalostech:

Programming Basic (Základy programování)

Znalost programovacích jazyků C nebo BASIC nebo zkušenosti s nimi vám mohou pomoci s pochopením obsahu tohoto kurzu.

Úvod Struktura kurzu

Obsah tohoto kurzu je následující.

Kapitola 1 – Přehled strukturovaného textu

Tato kapitola popisuje funkce a vhodné aplikace strukturovaného textu (ST).

Kapitola 2 – Základní pravidla programů ST

Tato kapitola popisuje základní pravidla sloužící k vytváření programů v ST.

Kapitola 3 – Vytváření I/O řídicích programů

Tato kapitola popisuje, jak vytvořit vstupně-výstupní řídicí programy.

Kapitola 4 – Aritmetické operace

Tato kapitola popisuje, jak vytvořit programy pro aritmetické operace.

Kapitola 5 – Podmíněné větvení

Tato kapitola popisuje podmíněné větvení.

Kapitola 6 – Ukládání dat a manipulace s nimi

Tato kapitola popisuje, jak psát stručné programy k ukládání dat a manipulaci s nimi.

Kapitola 7 – Manipulace s řetězcovými daty

Tato kapitola popisuje metody, jak manipulovat s řetězcovými daty.

Závěrečný test

Hodnocení ke splnění: 60 % nebo vyšší

Úvod Jak používat tento elektronický výukový nástroj

Přejdete na následující stránku		Přejdete na následující stránku.
Zpět na předchozí stránku		Zpět na předchozí stránku.
Přesunutí na požadovanou stránku		Zobrazí se „Obsah“, pomocí kterého můžete přejít na požadovanou stránku.
Ukončit výuku		Ukončit výuku.

Úvod Upozornění ohledně použití

Bezpečnostní opatření

Pokud se učíte používáním aktuálních produktů, pozorně si prosím přečtěte bezpečnostní opatření v odpovídajících příručkách.

Preventivní opatření v tomto kurzu

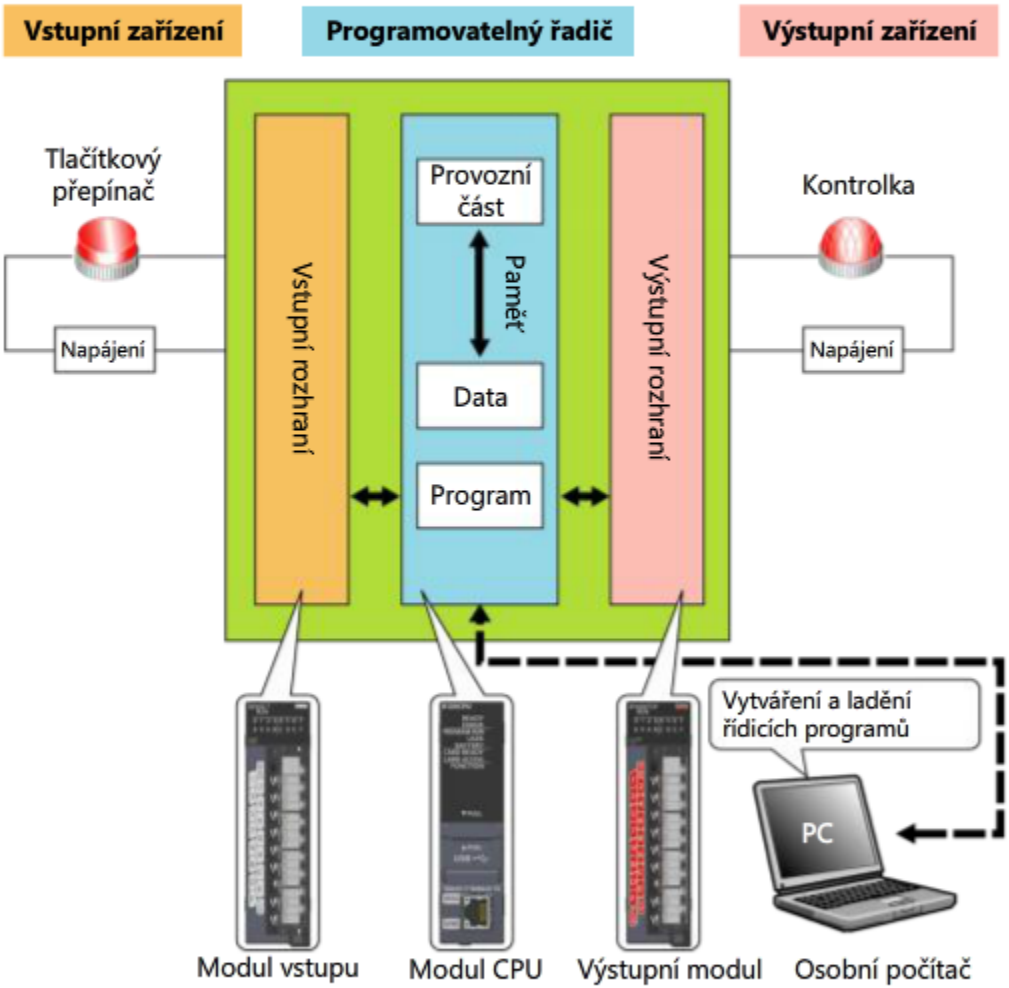
Zobrazené obrazovky inženýrského softwaru MELSOFT, který používáte, se mohou lišit od těch v tomto kurzu.
Tento kurz používá při vytváření programů symboly žebříkového diagramu ze softwaru MELSOFT GX Works3.

1. kapitola Přehled strukturovaného textu

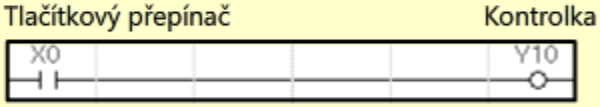
Tato kapitola popisuje funkce a vhodné aplikace strukturovaného textu (ST).

1.1 Řídicí programy

Následující obrázek ilustruje konfiguraci systému programovatelného kontroléru.
Programovatelné řadiče pracují v souladu s řídicími programy.
Provoz programovatelných řadičů lze vytvořením řídicích programů nakonfigurovat dle stanovených požadavků.



Řídicí program (Žebříkový diagram)



Po napsání řídicího programu se kontrolka rozsvítí v reakci na stav tlačítkového přepínače.

Programovací jazyky pro programovatelné řadiče jsou definovány mezinárodními normami vyvinutými International Electrotechnical Commission (IEC) (mezinárodní elektrotechnická komise).

1.2 Vlastnosti ST a porovnání s ostatními programovacími jazyky dle normy IEC

IEC 61131 je mezinárodní normou pro systémy programovatelného kontroléru.
Programovací jazyky pro programovatelné řadiče standardizuje norma IEC 61131-3. ST patří mezi standardní programovací jazyky.
Každý jazyk má odlišné vlastnosti lišící se dle požadované aplikace a schopností programátora.

Následující tabulka uvádí vlastnosti programovacích jazyků IEC 61131-3.

Vlastnosti programovacího	jazyka
Ladder **D**iagram (LD) (Žebříkový diagram)	• Symboly pro kontakty a cívky slouží k vytvoření programu připomínajícího elektrický obvod. • Programový tok mohou snadno pochopit a sledovat i začátečníci.
Structured **T**ext (ST) (Strukturovaný text)	• Programy jsou napsány textově (znaky). • ST snadno pochopí ti, kdo mají zkušenosti s psaním programů v programovacích jazycích C nebo BASIC. • Výpočetní vzorce jsou podobné matematickým výrazům, takže není složité je pochopit. • ST je vhodný k manipulaci s daty.
Function **B**lock **D**iagram (FBD) (Funkční blokový diagram)	• Programy jsou psány pomocí umisťování bloků s různými funkcemi a stanovením vztahů mezi bloky. • FBD zlepšuje čitelnost, protože máte přehled o celé operaci.
Sequential **F**unction **C**hart (SFC) (Sekvenční funkční schéma)	• Podmínky a procesy jsou zapsány pomocí tokových diagramů. • Programový tok je snadno pochopitelný.
Instruction **L**ist (IL) (Seznam instrukcí)	• IL je podobný strojovému jazyku. • IL se dnes používá jen velmi zřídka.

Tento kurz popisuje, jak psát základní řídicí programy pomocí ST.

1.3 Shrnutí

Obsah této kapitoly je následující:

- Vztahy mezi systémy programovatelných kontrolérů a řídicími programy
- Mezinárodní norma pro řídicí programy
- Vlastnosti ST

Důležité body ke zvážení:

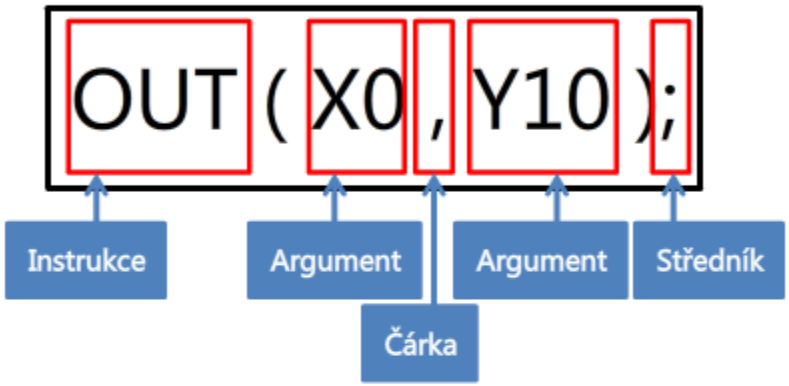
Vztahy mezi systémy programovatelných kontrolérů a řídicími programy	• Programovatelné řadiče pracují v souladu s řídicími programy. • Provoz programovatelných řadičů lze vytvořením řídicích programů nakonfigurovat dle stanovených požadavků.
Mezinárodní norma pro řídicí programy	• ST patří mezi programovací jazyky dle normy IEC. • Mezi další programovací jazyky dle IEC patří LD, FBD, SFC a IL, které mají odlišné vlastnosti vyhovující odlišným aplikacím a úrovni programátorských dovedností.
Vlastnosti ST	• ST snadno pochopí ti, kdo mají zkušenosti s psaním programů v jazycích C nebo BASIC. • Výpočty, například sčítání a odčítání, lze napsat jako běžně používané matematické výrazy, které je snadné pochopit. • ST je vhodný k manipulaci s daty.

2. kapitola Základní pravidla programů v ST

Tato kapitola popisuje základní pravidla sloužící k vytváření programů v ST.

2.1 Příklad základního programu (I/O řídicí příkaz)

Tato část popisuje příklad základního programu ST.
V následujícím vzorovém programu se aktivuje výstup Y10, když bude aktivován vstup X0. Výstup Y10 se deaktivuje, když se bude deaktivován vstup X0.



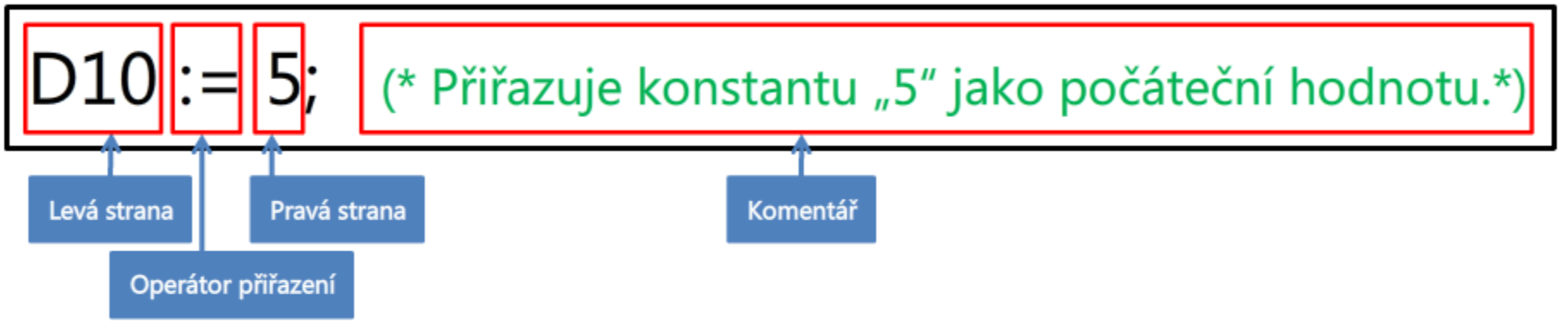
Instrukce definuje operaci, která se má spustit.
Argumenty jsou napsány v závorkách za instrukcí.
Argumenty lze použít k popisu proměnných, aritmetických výrazů a hodnot konstant.
U programovatelných řadičů MELSEC lze jako proměnné používat zařízení modulu CPU.

Počet argumentů závisí na instrukci.
Jednotlivé argumenty se oddělují čárkami (,).

Výše uvedená řádka představuje jeden příkaz. Každý příkaz končí středníkem (;).
Program je tvořen kombinací příkazů.

2.2 Příklad základního programu (přiřazovací příkaz)

Další příklad ilustruje program používající příkaz přiřazení.
Následující příkaz přiřazuje konstantu „5" v desítkové soustavě k proměnné „D10".



K příkazu přiřazení se používá operátor přiřazení (:=). Povšimněte si, že nalevo od znaku rovná se (=) se nachází dvojtečka (:).
Operátor přiřazení přiřadí hodnotu na pravé straně proměnné na levé straně.

Přidání komentářů do programu zvyšuje pochopitelnost operací. Komentáře vkládejte mezi dvě hvězdičky (* *).

2.3 Numerický zápis

U vzorového programu na předchozí straně byla proměnné přiřazena hodnota v decimální soustavě.

Někdy se při sekvenčním řízení používají také jiné než decimální hodnoty, například hodnoty binární nebo hexadecimální. V následující tabulce jsou uvedeny typy numerických zápisů používané v ST pro programovatelné automaty MELSEC.

Typ numerického zápisu	Způsob zápisu	Příklad
Binární	Přidejte prefix „2#“.	**2#**11010
Oktalová	Přidejte prefix „8#“.	**8#**32
Decimální	Přímý vstup	26
	Přidejte prefix „K“.	**K**26
Hexadecimální	Přidejte prefix „16#“.	**16#**1A
	Přidejte prefix „H“	**H**1A

Níže najdete příklady programu, který přiřadí hodnoty k proměnným.

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

2.3.1 Bitový zápis

Bity představují podmínky true/false (pravda/nepravda), například stavy on/off (zapnut/vypnut) u signálů.
Bity také reprezentují splnění/nesplnění podmínek.
V ST nelze bity zapisovat jako „ON" a „OFF". Je nutné je zapsat jako „1" (ON) a „0" (OFF). Bity lze také zapsat jako „TRUE" a „FALSE".

V následující tabulce jsou uvedeny různé typy zápisu.

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

Následuje několik příkladů přiřazení hodnot k proměnným bitového typu.

Numerický zápis

```
X0 := 1;
```

=

Zápis true/false

```
X0 := TRUE;
```

Numerický zápis

```
X0 := 0;
```

=

Zápis true/false

```
X0 := FALSE;
```

2.4 Sekvence vykonávání programu

Příkazy ST se vykonávají v pořadí shora dolů.

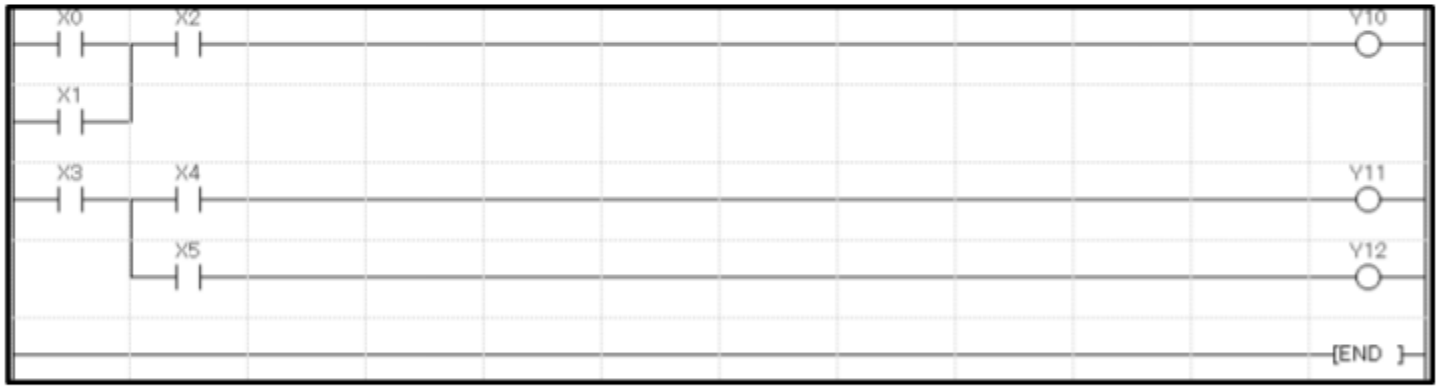
Příklad programu ST

```
Y10 := (X0 OR X1) AND X2;        (* Spustí se jako první *)
Y11 := X3 AND X4;                (* Spustí se jako druhý *)
Y12 := X3 AND X5;   (* Spustí se jako třetí. Nevyžaduje na konci příkaz END.*)
```

Opakované provádění

*Příkaz END je sice nutné použít na konci programu v LD, v ST to nutné není.

Následující žebříkový program představuje stejnou operaci jako v příkladu programu ST uvedeném výše.



Opakované provádění

Stejně jako v případě LD jsou instrukce ST vykonávány opakovaně návratem k první instrukci po dosažení poslední instrukce.

2.5 Shrnutí

Obsah této kapitoly je následující:

- Základní program ST
- Formát příkazu přiřazení
- Numerický zápis
- Sekvence vykonávání programu
- Komentář

Důležité body ke zvážení:

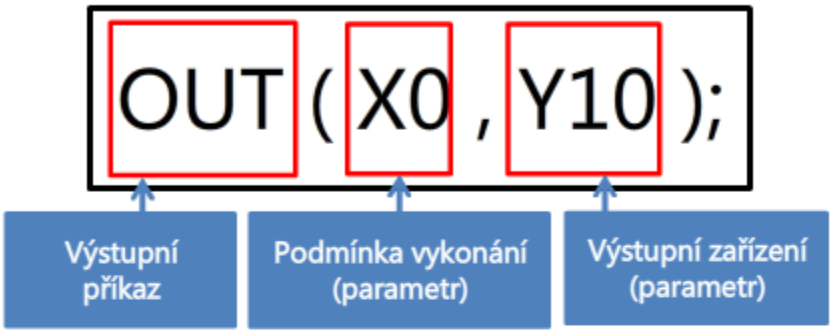
Základní program ST	• Příkaz je nejmenším prvkem programů ST. • Každý příkaz končí středníkem (;). • Program je tvořen kombinací příkazů.
Formát příkazu přiřazení	• Pro přiřazování se používá operátor přiřazení (:=).
Numerický zápis	• Typ numerického zápisu v ST • V ST se pro bitové hodnoty používá „1" a „0" namísto zápisu „ON" a „OFF". • Bitové hodnoty lze v ST zapsat také jako „TRUE" a „FALSE".
Sekvence vykonávání programu	• Programy vytvořené v ST se vykonávají v pořadí shora dolů. • Stejně jako u programů v LD se programy v ST zpracovávají po dosažení konce procesu opakovaně od začátku programu.
Komentář	• Přidání komentářů do programu zvyšuje pochopitelnost operací. • Komentáře jsou ohraničeny dvěma hvězdičkami (* *).

3. kapitola Vytváření I/O řídicích programů

Tato kapitola popisuje, jak vytvořit vstupně-výstupní řídicí programy v ST.

3.1 I/O řídicí programy

Následuje příklad programu pro I/O řízení programovatelného řadiče.



„OUT" je výstupním příkazem. Argument specifikuje podmínku provedení a zařízení, na které je výstup směřován. Když je uspokojena podmínka provedení X0, zařízení Y10 se zapne.

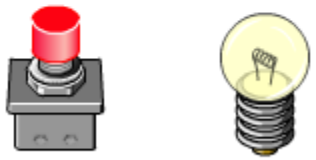
Klikněte na níže zobrazený vstupní přepínač. Vstupní přepínač X0 se zapne.

- Když je vstupní přepínač X0 zapnutý, zapne se výstupní kontrolka Y10.
- Když se vypne vstupní přepínač X0, vypne se také výstupní kontrolka Y10.

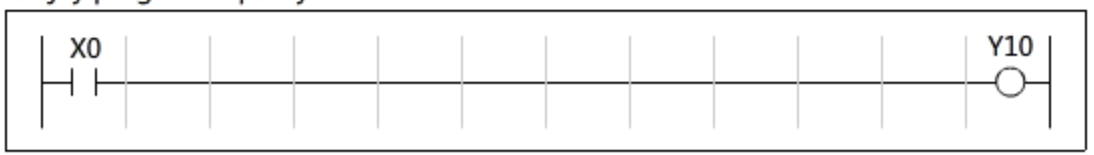
Vzor vstupně-výstupního programu napsaného v ST

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

Vstupní přepínač X0 Výstupní kontrolka Y10



Stejný program napsaný v LD



3.1 I/O řídicí programy

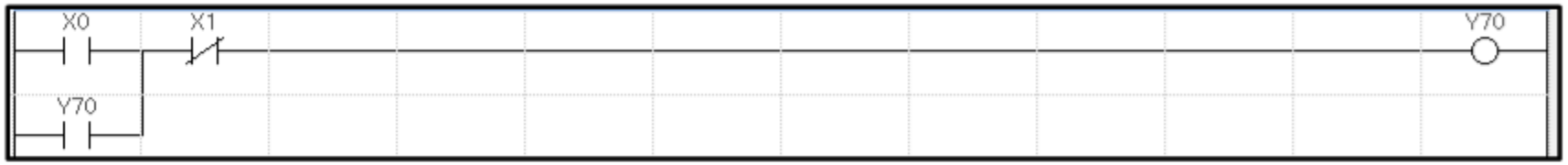
Podobně jako v LD je kromě OUT k dispozici řada instrukcí, například řídicí příkazy pro vstup/výstup a instrukce zpracování dat.
Více informací o instrukcích dostupných v ST naleznete v programátorském manuálu svého programovatelného řadiče.

Všimněte si, že zápis „OUT(X0, Y10);“ jako „Y10 := X0;“ provede stejnou operaci.

```
Y10 := X0;  (* Stejná operace jako „OUT(X0, Y10);“ *)
```

3.2 Kombinování více podmínek

Následující žebříkový program představuje obvod, který se sám udržuje.



Ten samý program lze v ST zapsat následovně.

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

Logický operátor

Ve výše uvedeném příkladu se logické operátory v ST používají ke zkombinování několika podmínek.

V následující tabulce je uveden seznam logických operátorů.

Operátor	Význam
OR	Logické NEBO
AND	Logické A
NOT	Logická negace
XOR	Vylučovací NEBO

3.3 Definování významu proměnných

Pomocí ST lze u programovatelných řadičů MELSEC přiřadit zařízení i symbolické proměnné jako aliasy proměnných.
Uživatelé mohou používat symbolické proměnné dle příslušné aplikace.

Při přiřazení symbolické proměnné odpovídající příslušné aplikaci, bude operace pochopitelnější.

```
Y10 := (X0 OR X1) AND X2;     (* Zápis za využití názvů zařízení *)
```

```
Lamp := (Switch0 OR Switch1) AND Switch2;     (* Zápis pomocí symbolické proměnné *)
```

Symbolickou proměnnou lze pojmenovat pomocí inženýrského softwaru MELSOFT.

Další příklady programu v tomto kurzu budou popsány pomocí symbolické proměnné.

3.4 Shrnutí

Obsah této kapitoly je následující:

Příklady I/O řídicích programů

- Logické operátory se v ST používají ke zkombinování několika podmínek.
- Jako názvy proměnných lze použít názvy zařízení a symbolické proměnné.

Důležité body ke zvážení:

Kombinování více podmínek	• Ke kombinování podmínek se v ST používají logické operátory.
Definování významu proměnných	• Při přiřazení symbolické proměnné odpovídající příslušné aplikaci, bude operace pochopitelnější.

4. kapitola Aritmetické operace

Tato kapitola popisuje, jak vytvořit programy pro aritmetické operace.

- Popis aritmetických operací
- Specifikace datových typů odpovídajících numerickým rozsahům
- Pojmenování proměnných, aby nedocházelo k nekonzistencím v datovém typu

4.1 Základní aritmetické operace

Tento vzorový program sčítá výrobní objem dvou samostatných výrobních linek.
Pravá strana rovnice je aritmetickou operací obsahující proměnné a aritmetické operace.

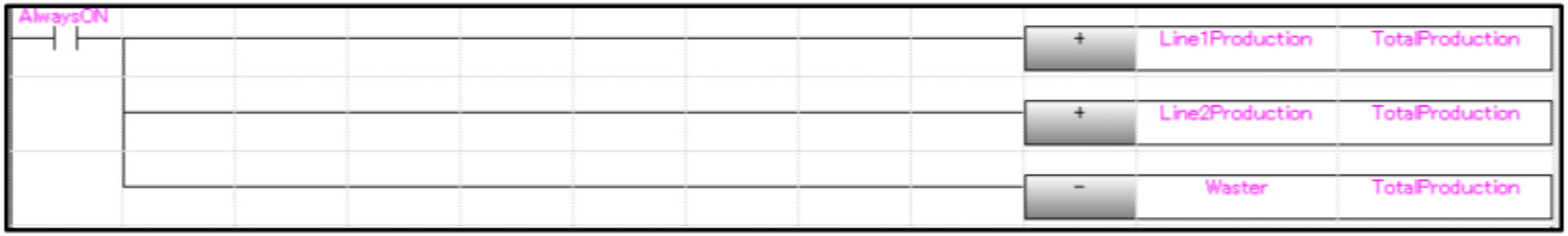
Vzor aritmetického programu napsaného v ST

Operátor sčítání

Operátor odčítání

```
TotalProduction := Line1Production + Line2Production - Waster;
(* Sečtení produkčního objemu dvou výrobních linek, odečtení počtu vadných
výrobků a přiřazení výsledné hodnoty. *)
```

Stejný program zapsaný v LD naleznete níže.



Jak je vidět výše, program je nutné v žebříkovém diagramu zapsat na 3 řádky, ale v ST k zápisu stačí 1 řádka.

V následující tabulce je uveden seznam základních aritmetických operátorů.

Operátor	Význam
+	Sčítání
-	Odčítání
*	Násobení
/	Dělení

4.2 Datové typy proměnných

U každé proměnné musí být zadán datový typ, aby došlo k definování rozsahu hodnot, s nimiž se bude manipulovat. Datovými typy pro číselné hodnoty, které se v ST používají jsou bity, celá čísla a reálná čísla.

V níže uvedené tabulce naleznete datové typy ST, které jsou použity v tomto kurzu.

Typ dat		Datový rozsah
Bit		Stav ON/OFF bitových zařízení a stav true/false pro výsledky provedení operace
Integer (celé číslo)	Word (slovo bez znaménka)	0 - 65 535
	Word (slovo se znaménkem)	−32 768 – 32 767
	Double-word (dvojité slovo bez znaménka)	0 - 4 294 967 295
	Double-word (Dvojité slovo se znaménkem)	−2 147 483 648 – 2 147 483 647

Při použití typu celého čísla vyberte slovo nebo dvojité slovo podle požadovaného datového rozsahu a vyberte typ se znaménkem nebo bez znaménka podle toho, zda potřebujete manipulovat se zápornými hodnotami. Datový typ proměnné zadejte v případě, že je pomocí inženýrského softwaru MELSOFT nastaven název symbolické proměnné.

4.3 Názvy proměnných představujících datové typy

Použití odlišných datových typů na levé a pravé straně přiřazovací rovnice může vést k chybě kompilace nebo neočekávaným výsledkům.
Níže následuje příklad takového případu.

```
ValueA := ValueB;      (* ValueA: celé číslo typu Word ValueB: celé číslo typu Double-word *)
```

Celé číslo typu double-word nelze přiřadit k celému číslu typu word. V tomto případě ale datový typ nelze rozpoznat.

K názvům proměnných lze přiřazovat prefixy představující datový typ, aby bylo možné datové typy odhalit na první pohled. Tomuto způsobu pojmenovávání proměnných se říká maďarský zápis.

Typ dat		Datový rozsah	Prefix	Rozšíření prefixu
Bit		Stav ON/OFF bitových zařízení a stav true/false pro výsledky provedení operace	b	**B**it (bit)
Integer (celé číslo)	Word (slovo bez znaménka)	0 – 65 535	u	**u**nsigned word (slovo bez znaménka)
	Word (slovo se znaménkem)	–32 768 – 32 767	w	signed **w**ord (slovo se znaménkem)
	Double-word (dvojité slovo bez znaménka)	0 – 4 294 967 295	ud	**u**nsigned **d**ouble-word (dvojité slovo bez znaménka)
	Double-word (Dvojité slovo se znaménkem)	–2 147 483 648 – 2 147 483 647	d	signed **d**ouble-word (dvojité slovo se znaménkem)

Vzorový program v horní části této stránky lze zapsat pomocí maďarského zápisu takto:

```
wValueA := dValueB;  (* Proměnná typu double-word nemůže být přiřazena proměnné typu word. *)
```

Při použití maďarského zápisu lze nekonzistentní datové typy identifikovat již při psaní programu.

Ve zbytku tohoto kurzu budou vzorové názvy proměnných zapsány maďarským zápisem.

4.4 Shrnutí

Obsah této kapitoly je následující:

- Popis aritmetických operací
- Specifikace datových typů odpovídajících numerickým rozsahům
- Přidávání názvů proměnných představujících datové typy

Důležité body ke zvážení:

Základní aritmetické operace	• K vyjádření výpočtu lze v ST používat operátory běžné v obvyklých programovacích jazycích.
Datové typy proměnných	• U každé proměnné musí být zadán datový typ, aby došlo k definování rozsahu hodnot, s nimiž se bude manipulovat.
Přidávání názvů proměnných představujících datové typy	• Popis názvů proměnných pomocí maďarského zápisu umožňuje identifikovat nekonzistentní datové typy proměnných již při psaní programu.

5. kapitola Podmíněné větvení

Řídicí programy také obsahují části kódu, ve kterých se konkrétní způsob zpracování liší dle zadaných podmínek. Tato kapitola popisuje podmíněné větvení.

5.1 Podmíněné větvení (IF)

Příkazy IF slouží k podmíněnému větvení. Následuje popis příkazu IF.

```
IF conditional expression THEN
  Execution statement;          (* Příkaz je proveden v případě uspokojení podmiňovacího výrazu. *)
END_IF;                         (* END_IF; je nutné zapsat na konec příkazů IF. *)
```

V tomto vzorovém programu se příkaz provede v případě splnění podmiňovacího výrazu. Pokud nebude podmiňovací výraz splněn, příkaz se neprovede.

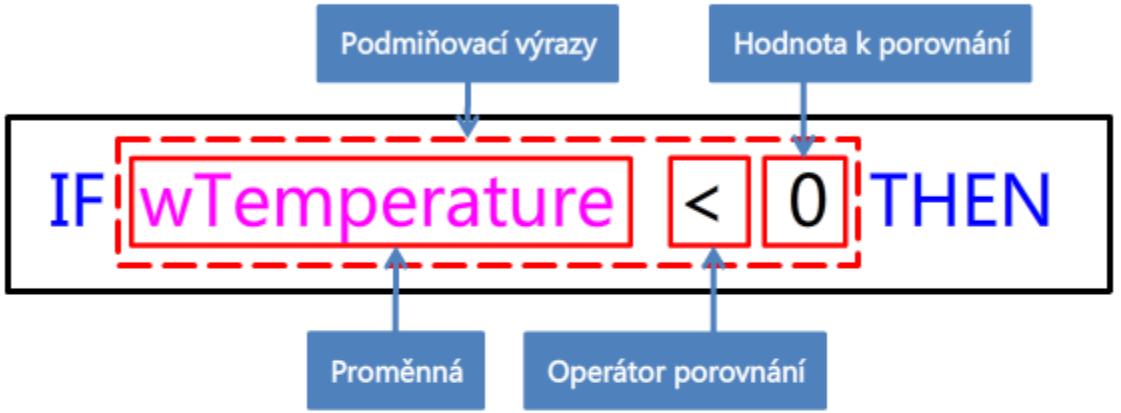
Následující obrázek ilustruje operační tok tohoto vzorového programu.

Následující příkaz ilustruje větvení programu při porovnání hodnot proměnných.
Ve vzorovém programu se ohřívač zapne ve chvíli, kdy teplota na ovládacím panelu klesne pod 0 stupňů.

```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Ohřívač se zapne ve chvíli, kdy teplota na ovládacím panelu klesne pod 0 stupňů. *)
END_IF;
```

5.1.1 Zápis podmiňovacích výrazů

Na předchozí straně se nachází popis podmiňovacího výrazu „wTemperature < 0“, který znamená „pokud bude hodnota proměnné wTemperature nižší než 0“.
Podmiňovací výrazy stejně jako tento výraz používají porovnávací operátory k vyjádření vztahu mezi proměnnými a srovnávacími hodnotami.



Na levé a pravé straně operátoru porovnání mohou být hodnoty k porovnání zapsány jako proměnné nebo jako konstanty.

Kromě porovnávání proměnných a konstant lze podmiňovací výrazy zapsat i tak, aby docházelo k porovnání proměnných a provedení logických operací na základě výsledků porovnání nebo hodnoty proměnné bitového typu.

Porovnávání proměnných

- uValue1 <= uValue2

Logická operace pro dva výsledky porovnání

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

Logická operace pro dvě proměnné bitového typu

- bSwitch0 OR bSwitch1

V následující tabulce je uveden seznam typů operátorů porovnání.

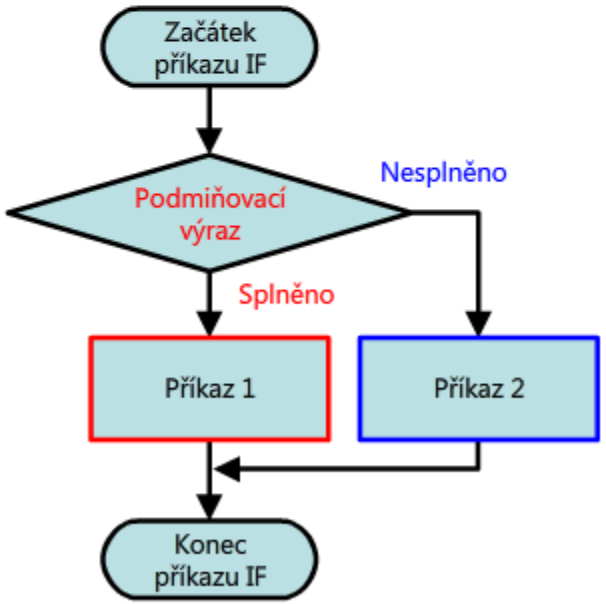
Operátor	Význam
>	Větší než
<	Menší než
>=	Větší nebo rovno
<=	Menší nebo rovno
=	Rovná se
<>	Nerovná se

5.1.2 Větvení při nesplnění příkazu IF (ELSE)

Jednoduché příkazy IF (viz 5.1) slouží k provedení příkazu při splnění podmiňovacího výrazu.
Za účelem vykonání odlišného příkazu, když podmiňovací výraz splněn není, se používá výraz ELSE.

```
IF conditional expression THEN
  Execution statement 1;          (* Příkaz 1 se provede v případě, že je podmiňovací výraz splněn. *)
  ELSE
  Execution statement 2;          (* Příkaz 2 se provede v případě, že podmiňovací výraz splněn není *)
END_IF;
```

Následující obrázek ilustruje operační tok při použití příkazu ELSE.



Následující vzorový program provádí odlišné příkazy podle toho, zda je podmínka splněna nebo ne.

Vzorový program v příkladu 5.1 měl slabinu v tom, že ohřívač zvyšoval teplotu i po dosažení 0 stupňů. Následující program ale ohřívač vypne, jakmile teplota „wTemperature" 0 stupňů přesáhne.

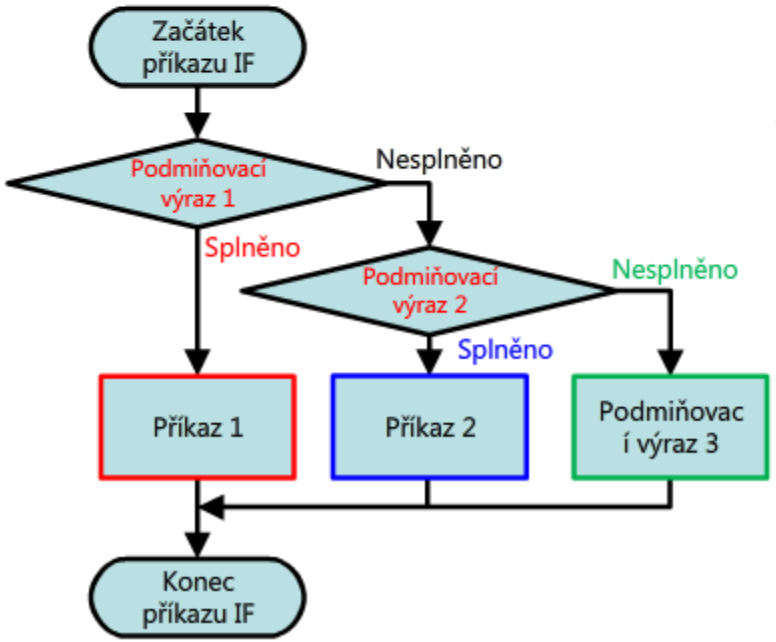
```
IF wTemperature < 0 THEN
  bHeater := 1;  (* Zapne ohřívač, když teplota klesne pod 0 stupňů. *)
  ELSE
  bHeater := 0;  (* Vypne ohřívač, když teplota dosáhne nebo překročí 0 stupňů *)
END_IF;
```

5.1.3 Další větvení u příkazu IF (ELSIF)

Příkazy ELSE slouží k provedení příkazu při nesplnění podmiňovacího výrazu.
Dalšího podmíněného větvení lze dosáhnout pomocí příkazů ELSIF. Pokud při použití tohoto příkazu nedojde ke splnění předchozího podmiňovacího výrazu, zkontroluje se následující podmiňovací výraz.

```
IF Conditional expression 1 THEN
  Execution statement 1;      (* Příkaz 1 se provede v případě splnění podmiňovacího výrazu 1. *)
  ELSIF Conditional expression 2 THEN
  Execution statement 2;      (* Příkaz 2 se provede v případě nesplnění podmiňovacího výrazu 1 a splnění podmiňovacího výrazu 2. *)
  ELSE
  Execution statement 3;      (* Příkaz 3 se provede v případě nesplnění podmiňovacích výrazů 1 a 2. *)
END_IF;
```

Následující obrázek ilustruje operační tok při použití příkazu ELSEIF.



Příkaz ELSIF je přidán do vzorového programu v sekci 5.1.2, aby vyřešil případ, kdy teplota přesáhne 40 stupňů.

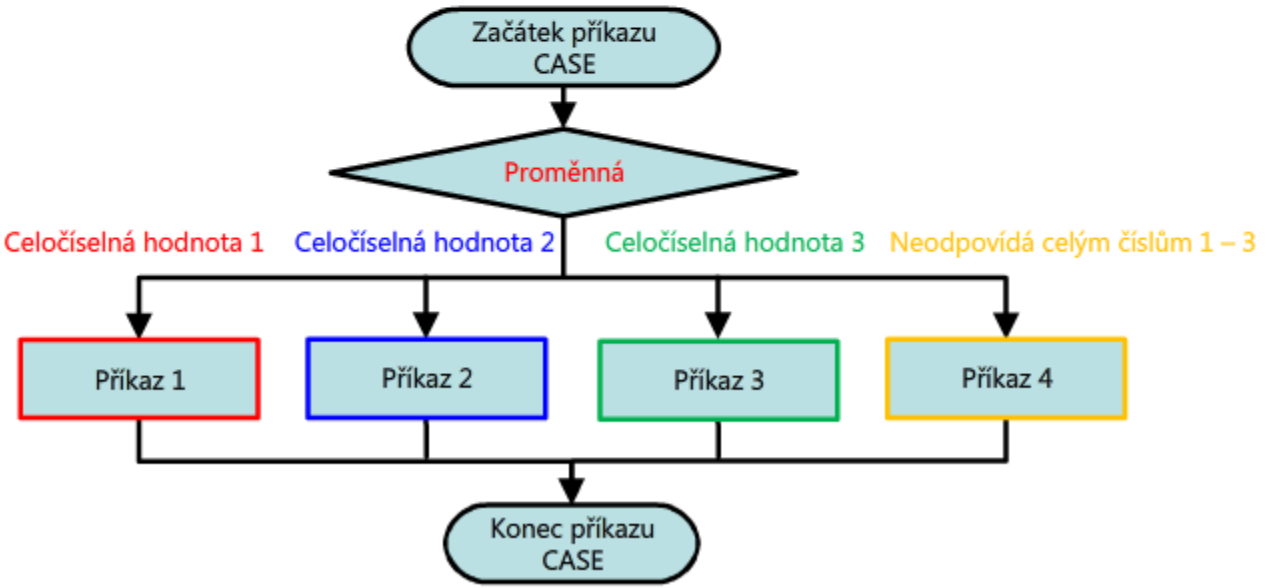
```
IF wTemperature < 0 THEN
   bHeater := 1; (* Zapne ohřívač, když teplota klesne pod 0 stupňů. *)
   bCooler := 0; (*  Vypne chladič, když teplota klesne pod 0 stupňů. *)
   ELSIF 40 < wTemperature THEN
   bHeater := 0; (* Vypne ohřívač, pokud teplota přesáhne 40 stupňů. *)
   bCooler := 1; (* Zapne chladič, když teplota přesáhne 40 stupňů. *)
   ELSE
   bHeater := 0; (*  Vypne ohřívač v případě nesplnění ani jedné z výše uvedených podmínek. *)
   bCooler := 0; (*  Vypne chladič v případě nesplnění ani jedné z výše uvedených podmínek. *)
END_IF;
```

5.2 Podmíněné větvení dle celočíselných hodnot (CASE)

Příkazy IF slouží k větvení založenému na splnění nebo nesplnění podmiňovacích výrazů.
Příkazy CASE slouží k větvení na základě celočíselných hodnot.
Následující obrázek ilustruje způsob zápisu příkazu CASE.

```
CASE Variable OF
    Integer value 1: Execution statement 1;     (* Příkaz 1 se provede v případě, že se celočíselná hodnota rovná 1. *)
    Integer value 2: Execution statement 2;     (* Příkaz 2 se provede v případě, že se celočíselná hodnota rovná 2. *)
    Integer value 3: Execution statement 3;     (* Příkaz 3 se provede v případě, že se celočíselná hodnota rovná 3. *)
    ELSE Execution statement 4;                 (* Příkaz 4 se provede v případě, že proměnná neodpovídá žádné z uvedených celočíselných hodnot. *)
END_CASE;                                       (* „END_CASE;" je nutné uvést na konec příkazu CASE. *)
```

Následující obrázek ilustruje operační tok při použití příkazu CASE.



5.2.1 Příklad programu s využitím příkazu CASE

Provedení příkazu CASE je popsáno na běhu vzorového programu.

Kliknutím na ▶ můžete pokračovat na další stranu. Za účelem opakovaného přehrání animace klikněte na tlačítko „Přehrát".

Přehrát


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

Hmotnost	uSize	Velikost
0 až 20 kg	1	M
21 až 30 kg	2	L
31 až 40 kg	3	XL
41 kg a více	4	Oversize

5.3 Shrnutí

Obsah této kapitoly je následující:

- Podmíněné větvení s příkazy IF
- Zápis podmiňovacích výrazů
- Podmíněné větvení dle celočíselných hodnot (příkaz CASE)

Důležité body ke zvážení:

Příkaz IF	• Pomocí příkazu IF se program větví při splnění podmiňovacího výrazu. • Příkaz ELSE se používá v případě nesplnění podmiňovacího výrazu. • Příkaz ELSIF se používá k přidání dalšího větvení pro případ nesplnění podmiňovacího výrazu v rámci příkazu IF.
Podmiňovací výraz	• Podmiňovací výrazy představují vztah mezi proměnnými a hodnotami za využití operátorů porovnávání.
Příkaz CASE	• Příkazy CASE slouží k větvení na základě celočíselných hodnot.

6. kapitola Ukládání dat a manipulace s nimi

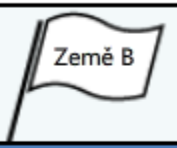
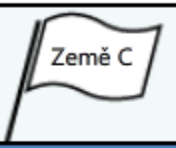
Kromě vstupně-výstupních řídicích aplikací se programovatelné řadiče v současnosti používají v jádrech výrobních systémů také ke zpracování velkého množství dat.
Za účelem zpracování velkého množství dat je nutné tato data uložit a dle potřeby číst.
Tato kapitola popisuje, jak psát stručné programy k ukládání a zpracování dat.

- K řazení a organizaci proměnných se používají pole.
- K organizování souvisejících proměnných se používají datové struktury.
- Programy běžící ve smyčkách mohou pole efektivně zpracovávat pomocí příkazů FOR.

Pomocí polí, datových struktur a příkazů FOR lze psát stručné programy k ukládání a zpracování dat.

6.1 Řazení a ukládání dat (pole)

Při využití polí lze pomocí jediné proměnné obsluhovat více hodnot.
V následujícím příkladu jsou data o výrobním objemu v oboru automobilové výroby ukládána na základě cílové země.

Cíl			
Výrobní objem	35	75	65

Proměnné jsou přiřazena data o výrobním objemu dle cílové země. Bez použití polí by musela být pro každou destinaci vytvořena vlastní proměnná.
Při použití polí lze ale výrobní objem pro více destinací přiřadit a uložit do jediné proměnné.

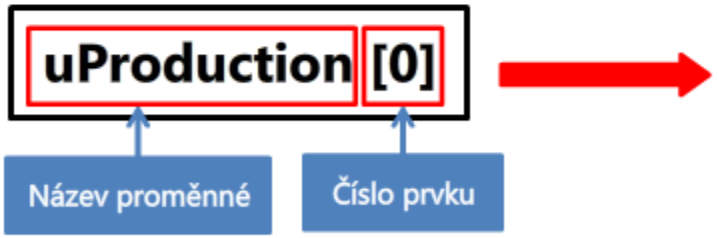
Bez použití pole

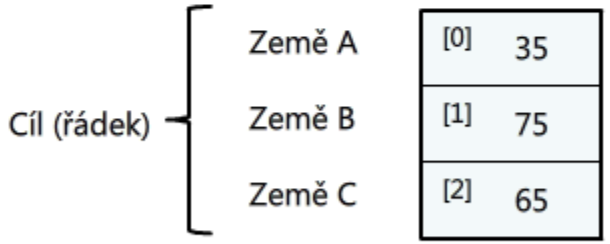
uProductionA
uProductionB
uProductionC

S použitím pole

uProduction

Jednotlivé proměnné v poli jsou odlišeny číslem prvku. Číslování prvků začíná od nuly [0].



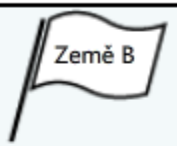
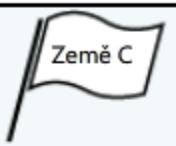


V následujícím příkladu programu je přiřazena hodnota proměnné plánovaného výrobního objemu pro zemi A.

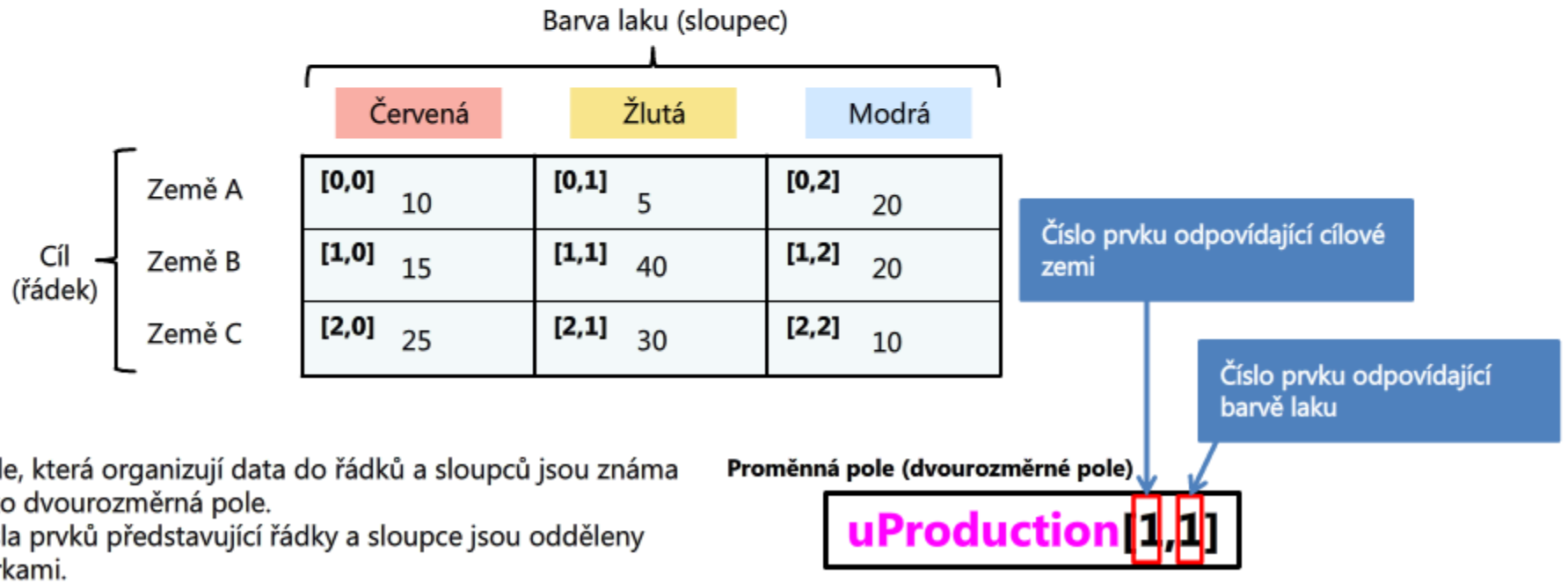
```
uShowProductionPlan := uProduction[0];
(* Specifikuje číslo prvku pro zemi A. *)
```

6.1.1 Dvourozměrné pole

Nyní kromě údaje o cíli uložíme také údaj o barvě.

Cíl									
Barva laku									
Výrobní objem	10	5	20	15	40	20	25	30	10
	35 celkem			75 celkem			65 celkem		

Jak je vidět v následující tabulce, lze data oddělovat a ukládat na základě barvy laku (sloupec), pro každou cílovou zemi (řádek).

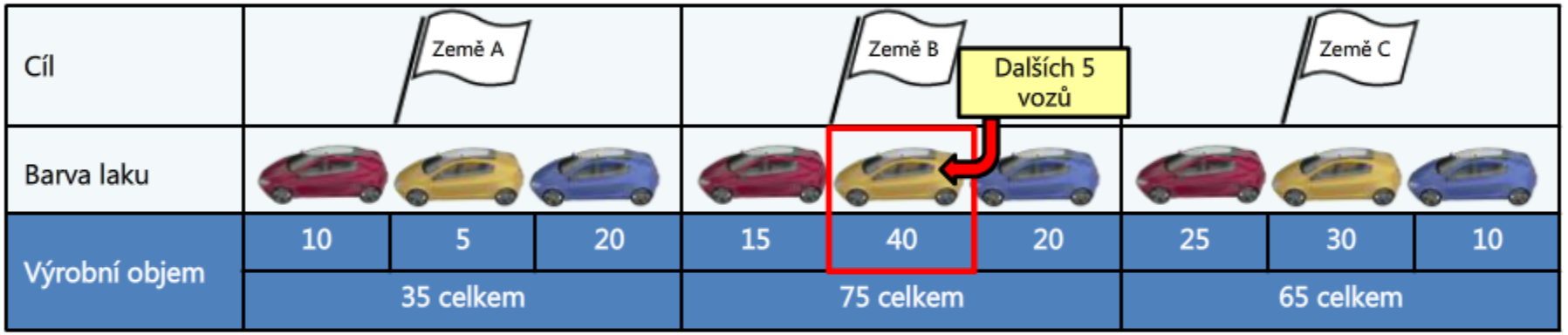


Pole, která organizují data do řádků a sloupců jsou známa jako dvourozměrná pole.
Čísla prvků představující řádky a sloupce jsou odděleny čárkami.

6.1.2 Přiřazení dvourozměrného pole

Následující vzorový program za využití dvourozměrného pole přiřadí k plánovanému produkčnímu objemu žlutých automobilů pro zemi B počet vozů, které je nutné vyrobit urgentně.

```
uAdditionalProduction := 5;
uProduction[1,1] := uProduction[1,1] + uAdditionalProduction;
(* Navýší původně naplánovaný výrobní objem o další produkci (5 kusů). *)
```



Barva laku (sloupec)

Cíl (řádek)	Červená	Žlutá	Modrá
Země A	[0,0] 10	[0,1] 5	[0,2] 20
Země B	[1,0] 15	[1,1] 40 -> 45	[1,2] 20
Země C	[2,0] 25	[2,1] 30	[2,2] 10

6.1.3 Zpracování informací uložených ve vícerozměrných polích

Následující vzorový program využívá dvourozměrná pole k výpočtu celkového výrobního objemu naplánovaného pro všechny barvy vozů pro zemi C a hodnotu pak přiřadí proměnné.

```
uProductionToday := uProduction[2,0] + uProduction[2,1] +uProduction[2,2];
(* Vypočítá celkový objem plánované produkce pro dnešní den pro všechny barvy laků pro zemi C a
hodnotu poté přiřadí do proměnné „uProductionToday“. *)
```

Cíl	Země A	Země B	Země C
Barva laku			
Výrobní objem	10 | 5 | 20	15 | 45 | 20	25 | 30 | 10
	35 celkem	80 celkem	65 celkem

Barva laku (sloupec)

Cíl (řádek)	Červená	Žlutá	Modrá
Země A	[0,0] 10	[0,1] 5	[0,2] 20
Země B	[1,0] 15	[1,1] 45	[1,2] 20
Země C	[2,0] 25	[2,1] 30	[2,2] 10

6.2 Vytváření smyček (FOR)

Vzorový program z předchozí strany (je přiřazen plánovaný výrobní objem pro dnešní den) je níže uveden znovu.

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

Tento příklad programu přidává další proměnné spolu s tím, jak stoupá počet barev laků. Výraz se pak prodlužuje a zhoršuje se jeho čitelnost.

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

V tomto případě lze příkaz pro smyčku použít k vytvoření čistšího kódu.

Mezi příkazy pro vytváření smyček patří příkazy FOR, WHILE a REPEAT. Tento kurz se zabývá příkazy FOR.

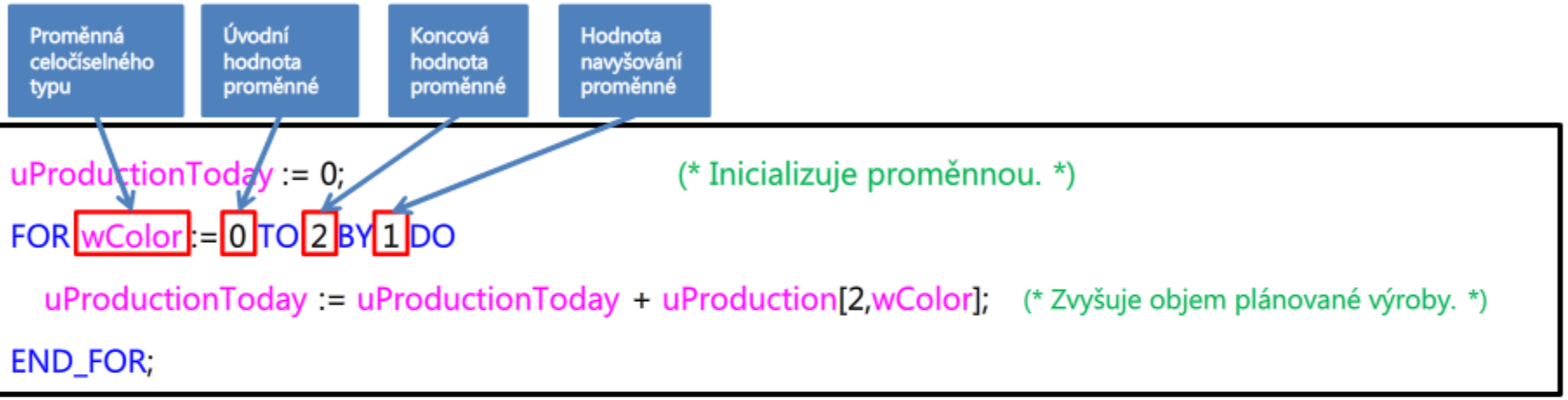
Následuje popis příkazů FOR.

```
FOR variable := initial value TO final value BY increments DO
  Execution statement; (* Příkaz je prováděn ve smyčce, dokud hodnota proměnné nedosáhne koncové hodnoty. *)
END_FOR;               (* END_FOR; je nutné zapsat na konec příkazů FOR.  *)
```

Výraz je opakován, dokud nebude dosažena finální hodnota proměnné a proveden kód „END_FOR;".

6.2 Vytváření smyček (FOR)

Následující vzorový program využívá příkaz FOR k získání plánovaného výrobního objemu všech barev laku určených pro zemi C.



Za využití příkazu FOR se proměnná „wColor" navyšuje o jedničku od úvodní hodnoty nula. Příkaz se opakuje, dokud hodnota proměnné nedosáhne koncové hodnoty dva.
Proměnná „wColor" je v příkazu k provedení specifikována jako druhé číslo prvku v poli „uProduction".
Hodnota proměnné „wColor" se zvyšuje vždy, když je příkaz zopakován. Při získávání součtu se vždy přičte plánovaný objem výroby pro každou barvu laku.

Tento vzorový program je proveden ve smyčce třikrát za sebou. (Poprvé: červená [0] => Podruhé: žlutá [1] => Potřetí: modrá [2])

Operace tohoto programu jsou znázorněny na další straně.

6.2 Vytváření smyček (FOR)

Provádění příkazu FOR je popsáno pomocí operací příkladu programu.

Pole odhadovaného výrobního objemu

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

Kliknutím na  můžete pokračovat na další stranu.
Za účelem opakovaného přehrání animace klikněte na tlačítko „Přehrát".

Přehrát

Number of repetition of the loop: 3

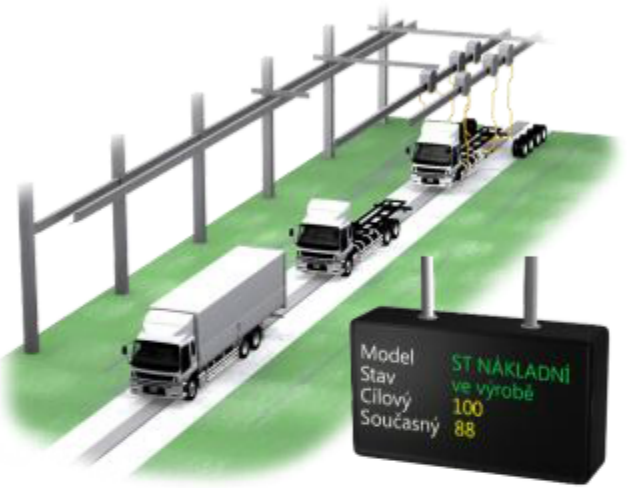
```
uProductionToday := 0;

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

6.3 Ukládání souvisejících dat (struktura)

Díky struktuře lze jeden název proměnné použít k reprezentaci několika souvisejících proměnných.
V následujícím příkladu je na Andonu (displeji) zobrazen stav automobilové výrobní linky.

Následující tabulka uvádí názvy proměnných, hodnoty a datové typy odpovídající zobrazeným položkám.

Položka	Název proměnné	Hodnota	Datový typ proměnné
Model	sModel	„ST TRUCK"	Textový řetězec
Stav	bStatus	„ve výrobě"	Bitový typ
Cílový objem výroby pro dnešní den	uPlanQty	„100"	Celočíselný typ word (bez znaménka)
Aktuální objem výroby	uActualQty	„88"	Celočíselný typ word (bez znaménka)



V případě nepoužití struktury by bylo nutné v případě existence více výrobních linek měnit názvy proměnných pro každou linku.
Následující příklad zobrazuje názvy proměnných podle výrobní linky.

První výrobní linka

s1stLineModel
b1stLineStatus
u1stLinePlanQty
u1stLineActualQty

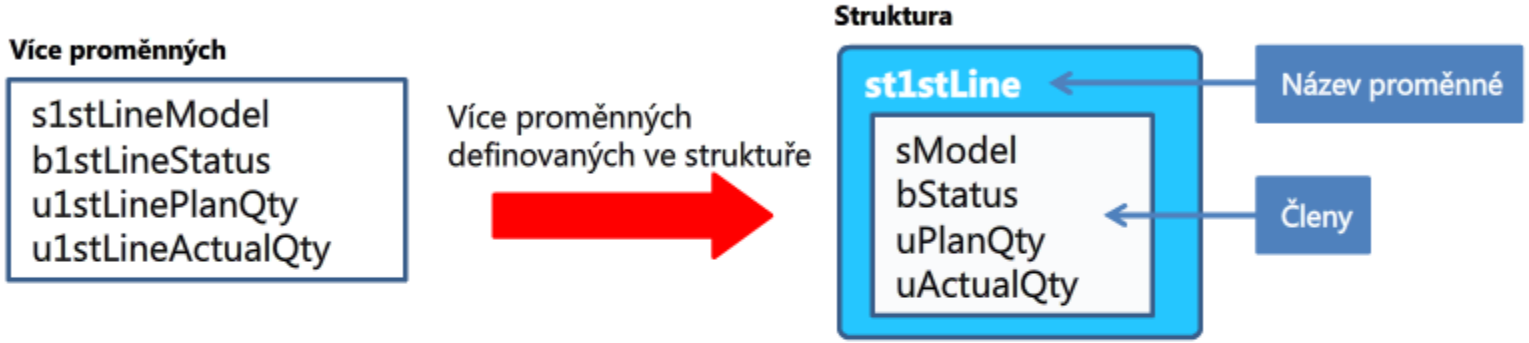
Druhá výrobní linka

s2ndLineModel
b2ndLineStatus
u2ndLinePlanQty
u2ndLineActualQty

Se zvětšujícím se počtem výrobních linek by se také zvyšoval počet proměnných, s nimiž by se pracovalo.
Program by se tím prodlužoval a zhoršila by se jeho čitelnost.

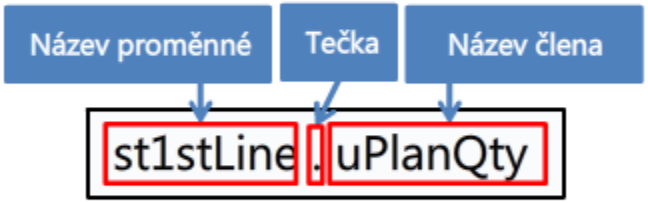
6.3 Ukládání souvisejících dat (struktura)

Díky strukturám je možné, aby jeden název proměnné představoval více proměnných souvisejících s jednou výrobní linkou. Tímto způsobem lze struktury využít k organizaci, ukládání a zpracování dat v dávkách dle podmínek a specifikací fyzických objektů, například zařízení, vybavení a obrobků.



Structure variable (strukturovaná proměnná) obsahuje prefix „st", který označuje, že se jedná o strukturu. Jednotlivé proměnné definované strukturou se označují jako členy. Datové typy jednotlivých členů se mohou lišit.

Každý člen strukturálního pole může být specifikován za číslem prvku v poli tak, že se před název člena dá tečka.

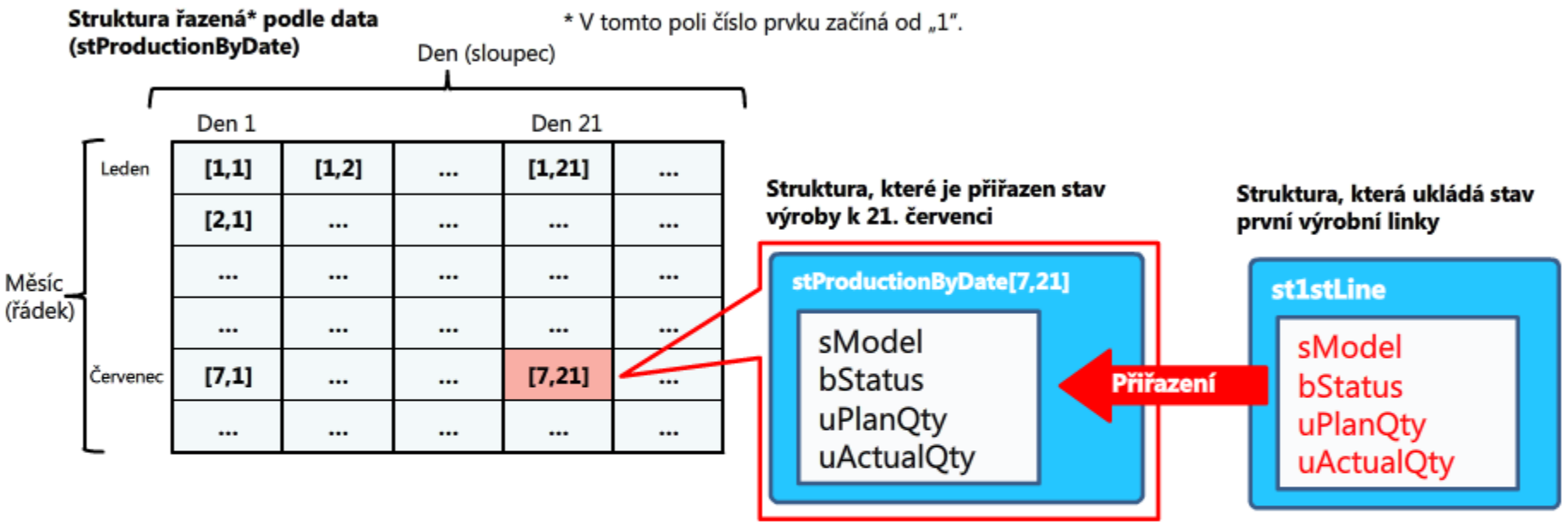


V následujícím vzorovém programu je členu strukturální proměnné přiřazena hodnota konstanty pro první výrobní linku.

```
st1stLine.uPlanQty := 150;
 (* Nastavuje dnešní cílovou produkci pro první výrobní linku na 150 kusů. *)
```

6.3.1 Ukládání strukturálních polí

Struktury lze vytvářet jako pole.
V následujícím příkladu se stav produkce ukládá podle data.

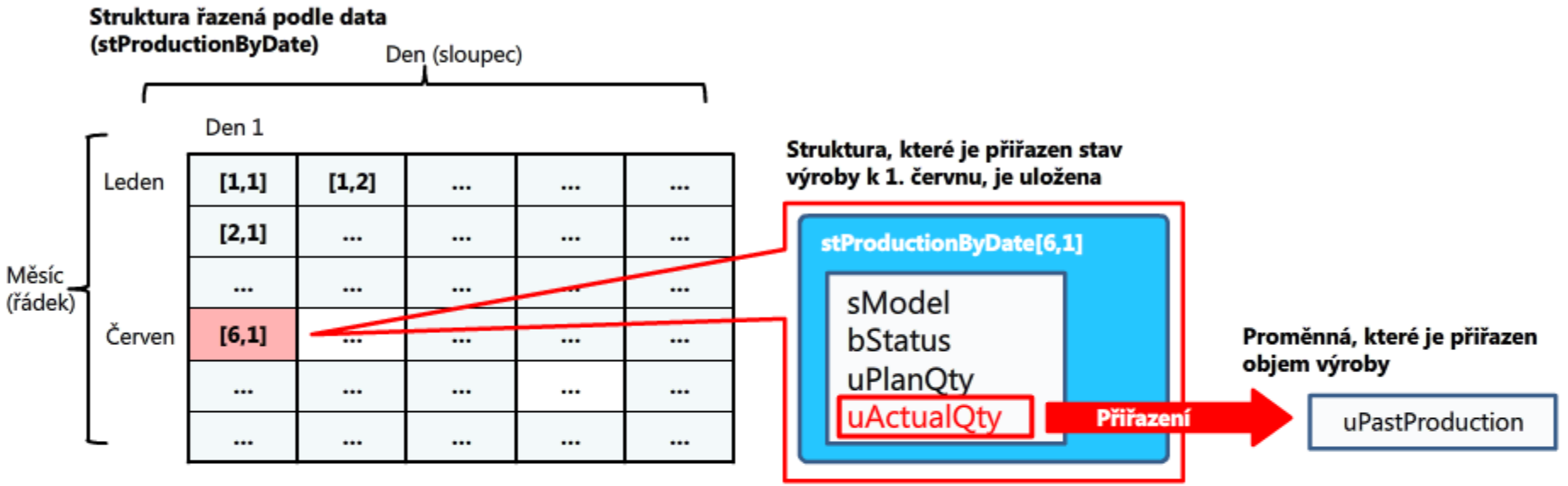



```
stProductionByDate[7,21] := st1stLine;
 (* Stav výroby k 21. červenci se uloží do struktury řazené podle data
(stProductionByDate). *)
```

Tímto způsobem není nutné členy pro přiřazení do struktury specifikovat samostatně.

6.3.2 Čtení strukturálních polí

V následujícím příkladu je načten objem výroby ze struktury řazené dle data a poté je hodnota přiřazena proměnné.




```
uPastProduction := stProductionByDate[6,1].uActualQty;
(* Přiřazuje objem výroby k 1. červnu proměnné uPastProduction. *)
```

Každého člena strukturálního pole lze specifikovat přidáním tečky (.) a názvu člena k číslu prvku pole.

6.4 Shrnutí

Obsah této kapitoly je následující:

- Přehled a použití polí
- Zpracování smyček pomocí příkazů FOR
- Přehled a používání struktur

Důležité body ke zvážení:

Pole	• Díky polím lze manipulovat řadou hodnot za využití jedné proměnné. • Jednotlivé proměnné v polích jsou určovány číslem prvku přidaným na konec názvu proměnné.
Příkaz FOR	• Příkazy k vytváření smyček se používají u programů, u kterých je nutné používat opakované operace. • Příkazy FOR slouží k opakování operace, dokud nebudou splněny podmínky na konci smyčky. Příkazy před příkazem „END_FOR;" se vykonávají opakovaně.
Struktura	• Díky strukturám lze jeden název proměnné použít k reprezentaci několika souvisejících proměnných. Struktury mohou obsahovat proměnné různých datových typů. • Jednotlivé proměnné nebo členy definované ve strukturách se specifikují přidáním tečky a názvu člena za název strukturální proměnné.

7. kapitola Manipulace s řetězcovými daty

V některých případech využívají programovatelné řadiče řetězcová data k zasílání příkazů nebo přijímání zpětné vazby od připojených zařízení, například od čteček čárových kódů, teplotních kontrolérů nebo elektronických vah. Za tímto účelem je nutné dle potřeby spojovat nebo extrahovat řetězcová data.

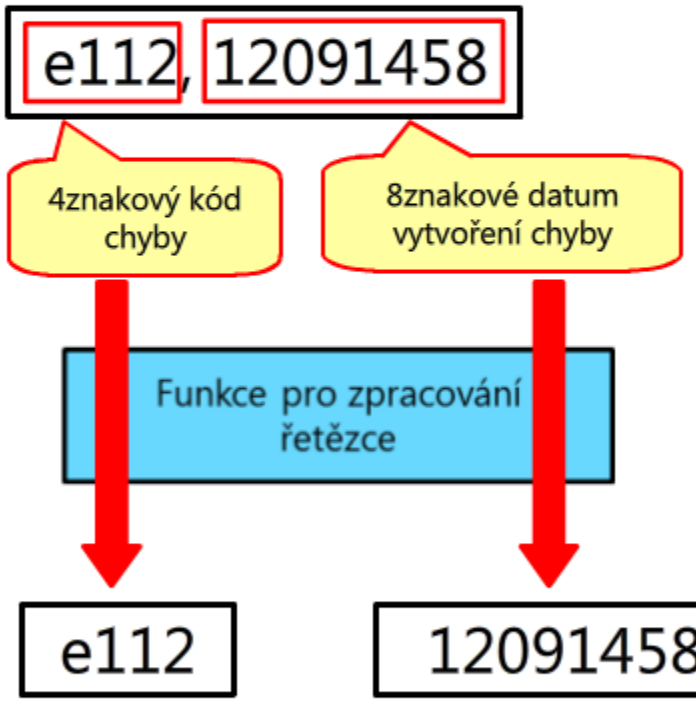
Tato kapitola popisuje, jak manipulovat s řetězcovými daty.

7.1 Příklad manipulace s řetězcovými daty

Příkladem zpracování řetězce je scénář, ve kterém jsou zpracovávána data z čtečky čárových kódů.
Ke zpracování řetězců jsou použity funkce (typ instrukcí).

V níže uvedené ilustraci řetězce přečtené čtečkou čárových kódů obsahují 4znakový chybový kód s pevnou délkou a 8znakový údaj o měsíci, datu a času s pevnou délkou.
Bude popsán příklad programu pro zpracování řetězce pracující s tímto systémem.

Příklad čtení řetězcových dat z čtečky čárových kódů



Je extrahován chybový kód.
7.3 Extrahování řetězců (LEFT)

Je extrahováno datum a čas výskytu chyby (14:58, 9. prosince).
7.4 Extrahování řetězců (MID)

7.2 Přiřazení řetězce

Než si vysvětlíme, jak extrahovat řetězce, popíše tato sekce datové typy řetězců.

V následující tabulce jsou uvedeny datové typy řetězců, které lze použít s programovatelnými řadiči.

Datový typ	Lze zpracovat znakový typ	Prefixy dle maďarského zápisu	Rozšíření prefixu
Řetězec	Řetězce alfanumerických znaků a čísel (ASCII) nebo japonských znaků (Shift-JIS)	s	**s**tring (řetězec)
String [Unicode]	Řetězce v řadě různých jazyků a s mnoha symboly	ws	**w**ide **s**tring (řetězec s rozšířenou znakovou sadou)

Použitý typ řetězce závisí na zařízení připojeném k programovatelnému řadiči a na použitém jazyku.
Tato kapitola popisuje různé typy textových řetězců.

Když je k řetězcové proměnné přiřazen řetězec řetězcového typu, uzavřete řetězec do jednoduchých uvozovek (').

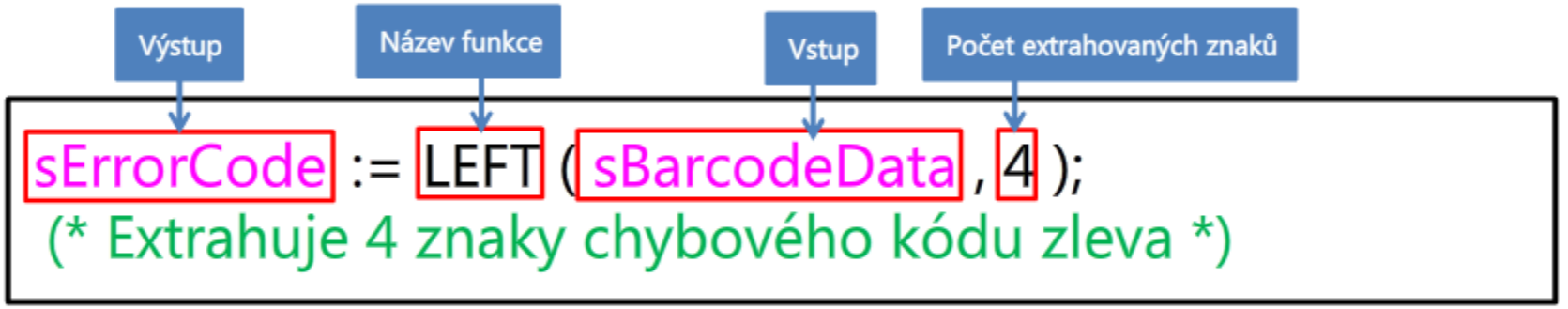
```
sDefault := 'e112,12091458';  (* Přiřazení řetězce *)
```

7.3 Extrahování řetězců (LEFT)

Je extrahován chybový kód „e112" z řetězcové proměnné „sBarcodeData" obsahující řetězec „e112,12091458".

Název proměnné	Uložený řetězec
sBarcodeData	e112, 12091458

Funkce LEFT extrahuje pouze určený počet znaků počínaje na levé straně vstupního řetězce.
Následuje ilustrace vzorového programu.



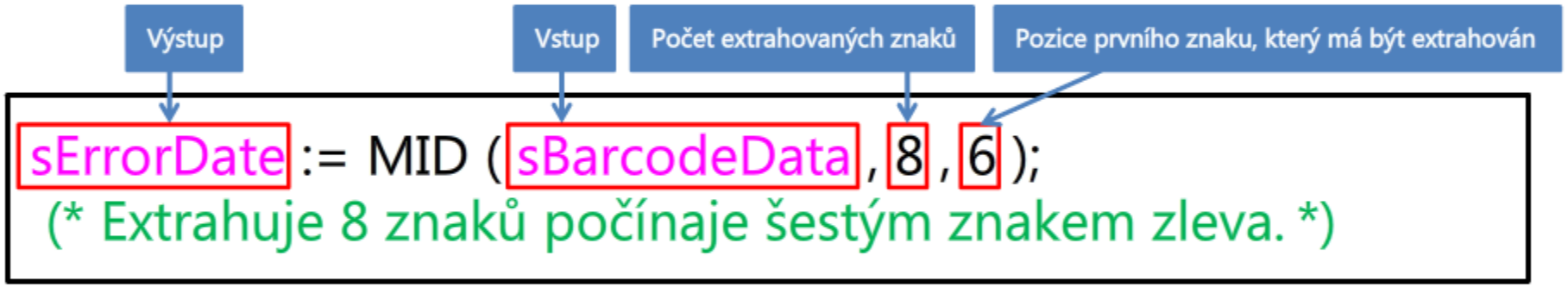
Jsou extrahovány čtyři znaky zleva. Levé straně je přiřazena hodnota „e112", což je řetězec představující chybový kód.

7.4 Extrahování řetězců (MID)

Je extrahován čas vytvoření chyby „12091458" z řetězcové proměnné „sBarcodeData" obsahující řetězec „e112,12091458".

Název proměnné	Uložený řetězec
sBarcodeData	e112,12091458

Funkce MID extrahuje určený počet znaků počínaje specifikovanou počáteční pozicí ve vstupním řetězci.
Následuje ilustrace vzorového programu.



V tomto příkladu je extrahován 8znakový řetězec počínaje šestým znakem. Levé straně je přiřazena hodnota „12091458", což je řetězec představující datum vytvoření chyby.

7.5 Shrnutí

Obsah této kapitoly je následující:

- Metody přiřazování řetězců řetězcovým proměnným
- Funkce načítající řetězce (LEFT a MID)

Důležité body ke zvážení:

Přiřazení řetězce	• Pro přiřazení řetězce k proměnné řetězcového typu, uzavřete řetězec do jednoduchých uvozovek ('). • Použijte typ string nebo string [Unicode] podle typu zařízení připojeného k programovatelnému řadiči nebo podle požadovaného jazyka.
Funkce pro manipulaci s řetězci	• K manipulaci s řetězci se používají funkce.

7.6 Shrnutí kurzu

Tento kurz se zabýval základy tvorby programů v ST.
Tím tento elektronický výukový kurz končí.

Programy v ST se vytváří pomocí inženýrského softwaru MELSOFT.
Podrobnosti o jednotlivých krocích, například o zadávání dat, úpravách, ukládání a kompilaci programů pomocí inženýrského softwaru MELSOFT naleznete v následujících materiálech.

- Elektronický výukový kurz Mitsubishi FA „MELSOFT GX Works3 (Structured Text)"
  (MELSOFT GX Works3 (Strukturovaný text)) (bude vydán brzy)
- Operační manuál k inženýrskému softwaru MELSOFT

Další informace o ST naleznete v následujících materiálech.

- Příručka k programování vašeho programovatelného řadiče

Informace o instrukcích a funkcích pro vaši aplikaci naleznete v následujících materiálech.

- Manuál k programování vašeho programovatelného řadiče

Test Závěrečný test

Když jste nyní dokončili všechny lekce kurzu Základy programování (Strukturovaný text), můžete podstoupit závěrečný test. Pokud si nejste jisti ohledně nějakého tématu, máte nyní možnost si jednotlivá témata zopakovat.
Tento závěrečný test obsahuje celkem 12 otázek (20 položek).
Závěrečný test můžete podstoupit kolikrát chcete.

Způsob provedení testu

Po vybrání odpovědi nezapomeňte kliknout na tlačítko **Odpověď**. Pokud nekliknete na tlačítko Odpověď, bude vaše odpověď ztracena. (Otázka bude tedy považována za nezodpovězenou.)

Hodnocení výsledků

Na stránce hodnocení se zobrazí počet správných odpovědí, počet otázek, procento správných odpovědí a výsledek úspěšný/neúspěšný.

Počet správných odpovědí: 4

Celkový počet odpovědí: 4

Procento: 100%

Abyste úspěšně složili tento test, musíte správně odpovědět na **60 %** otázek.

Pokračovat | Zkontrolovat

- Test můžete ukončit kliknutím na tlačítko **Pokračovat**.
- Test si můžete zkontrolovat kliknutím na tlačítko **Zkontrolovat**. (Kontrola správnosti odpovědí)
- Test si můžete zopakovat kliknutím na tlačítko **Znovu**.

Test Závěrečný test 1

Charakteristika strukturovaného textu (ST)
Zvolte nesprávný popis ST.

- ST snadno pochopí ti, kdo mají zkušenosti s psaním programů v jazycích C nebo BASIC.
- Výpočty, například sčítání a odčítání, lze napsat jako běžně používané matematické výrazy.
- Symboly pro kontakty a cívky slouží k vytvoření programu připomínajícího elektrický obvod.
- ST je vhodný k manipulaci s daty.

Odpovědět Zpět

Test Závěrečný test 2

Základní principy ST
Vyberte správný příkaz zapsaný v ST.

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

Odpovědět Zpět

Test — Závěrečný test 3

Popisné komentáře
Vyberte správný komentář zapsaný v ST.

- ' Přiřazuje proměnné hodnotu 1.
- (* Přiřazuje proměnné hodnotu 1. *)
- { Přiřazuje proměnné hodnotu 1. }
- <!-- Přiřazuje proměnné hodnotu 1. -->

Odpovědět Zpět

Test Závěrečný test 4

Sekvence vykonávání programu v ST
*Počáteční hodnota „uTotalProduction" je „100". Hodnota proměnné „uTotalProduction" bude po zpracování následujícího kódu „101". Vyberte správný stav „uTotalProduction" po uplynutí několika sekund.

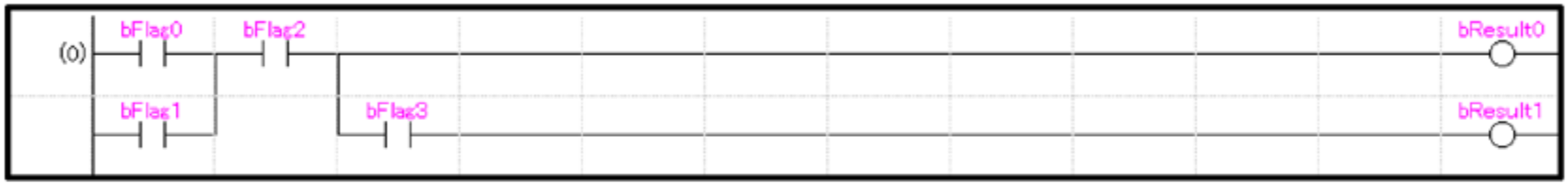
```
uTotalProduction := uTotalProduction + 1;
```

- ○ Hodnota zůstává 101.
- ○ Hodnota se neustále mění.

Odpovědět Zpět

Test — Závěrečný test 5

Kombinování více podmínek
Vyberte správný příklad programu v ST, který provádí stejnou operaci jako následující vzorový příklad v LD.



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

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

Odpovědět Zpět

Test — Závěrečný test 6

Popis příkazů IF v ST
Níže uvedený vzorový program provádí následující operaci.
- Pokud teplota klesne na hodnotu 5 stupňů nebo méně, ohřívač se zapne a chladič se vypne.
- Pokud teplota překročí 50 stupňů, ohřívač se vypne a chladič se zapne.
- Pokud teplota neodpovídá výše uvedeným příkazům, ohřívač i chladič se vypnou.

*Názvy proměnných: Teplota (wTemperature), ohřívač (bHeater) a chladič (bCooler)
Vyberte správnou možnost pro každou prázdnou sekci vzorového programu.

```
IF wTemperature [Q1] 5 [Q2]
    bHeater := 1;
    bCooler := 0;
[Q3] 50 [Q4] wTemperature [Q2]
    bHeater := 0;
    bCooler := 1;
[Q5]
    bHeater := 0;
    bCooler := 0;
END_IF;
```

Q1 --Select--
Q2 --Select--
Q3 --Select--
Q4 --Select--
Q5 --Select--

Odpovědět Zpět

Test — Závěrečný test 7

Příkazy CASE

Vyberte správnou možnost pro každý (Q1 až Q5) z následujících popisů příkazů CASE.

Příkazy CASE slouží k větvení na základě hodnoty (Q1).
Když v následujícím vzorovém programu hodnota (Q2) dosáhne 25, pak je proměnné (Q3) přiřazena hodnota (Q4). Když hodnota proměnné (Q2) není rovna 10, 25 nebo 8, je proměnné (Q3) přiřazena hodnota (Q5).

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

Q1 --Select--

Q2 --Select--

Q3 --Select--

Q4 --Select--

Q5 --Select--

Odpovědět Zpět

Test — Závěrečný test 8

Pole v ST a opakované příkazy
Následující vzorový program sčítá plánované produkční objemy pro všechny modely určené pro zemi Y a poté přiřadí tuto hodnotu proměnné. Vyberte část pole, která je přečtena poté, co je příkaz FOR třikrát proveden ve smyčce.

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

Pole sloužící k uložení odhadovaného počtu vyrobených kusů dle modelu a místa určení (uProduction)

Cíl (řádek)	Model 1	Model 2	Model 3	Model 4
Země X	[0,0]	[0,1]	[0,2] C	[0,3]
Země Y	[1.0]	[1.1] A	[1.2] D	[1.3] E
Země Z	[2,0]	[2,1] B	[2,2]	[2,3]

- A
- B
- C
- D

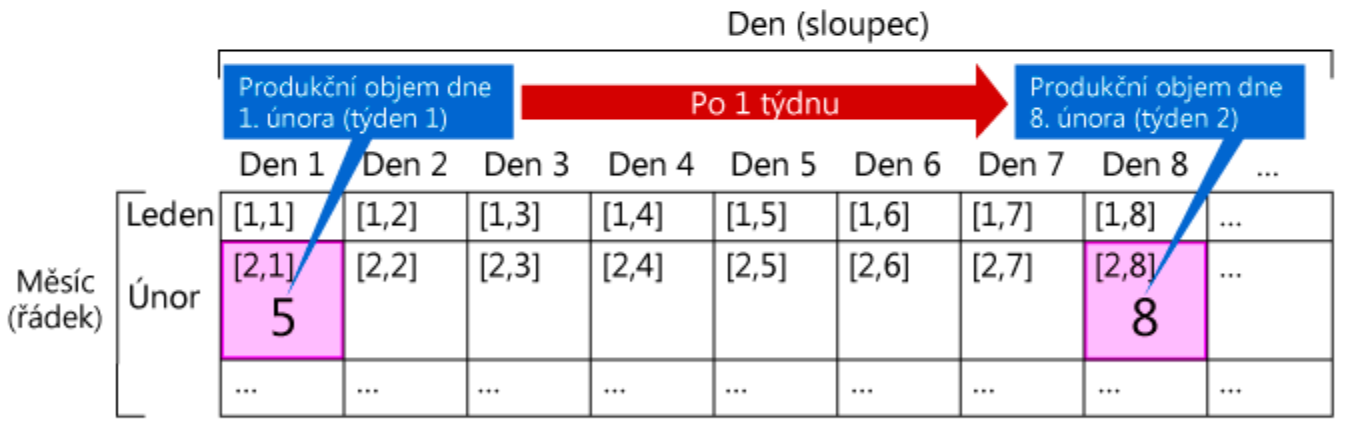
Odpovědět | Zpět

Test — Závěrečný test 9

Pole v ST a opakované příkazy
Následující vzorový program získává celkový výrobní objem ve stejných dnech týdnu. Z pole, které ukládá denní produkční objemy je získán součet za 4 týdny. Vyberte správné číslo, které vzorový program vrátí.

```
uTotalProduction := 0;
FOR wOnceAWeek := 1 TO ■ BY 7 DO
  uTotalProduction := uTotalProduction + uProductionByDate[2,wOnceAWeek];
END_FOR;
(* Načte součty výrobních objemů pro stejné dny v týdnu v průběhu období 4 týdnů počínaje 1. únorem. *)
```

Pole, které ukládá produkční objemy podle dnů (uProductionByDate)



- 22
- 21
- 4
- 28

Odpovědět Zpět

Test Závěrečný test 10

Charakteristiky struktur v ST
Vyberte nesprávný popis struktur.

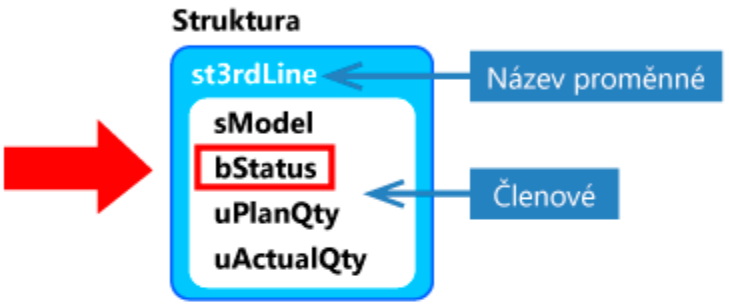
- Struktury slouží k organizaci a ukládání dat na zařízeních dle různých podmínek, například dle stavu a specifikací.
- Pomocí struktur lze úsporně psát programy, které zpracovávají velká množství dat.
- Všechny členy definované ve struktuře musí mít stejný datový typ.
- Členům ve stejné struktuře lze přiřazovat hodnoty bez nutnosti jejich individuální specifikace.

Odpovědět Zpět

Test Závěrečný test 11

Specifikace členů pro struktury v ST
Následující struktura organizuje proměnné související s automobilovou výrobní linkou.
Vyberte správný popis specifikující člena „bStatus" v rámci této struktury.

Parametr	Název proměnné
Model	sModel
Stav	bStatus
Cílová produkce pro současný den	uPlanQty
Aktuální objem výroby	uActualQty

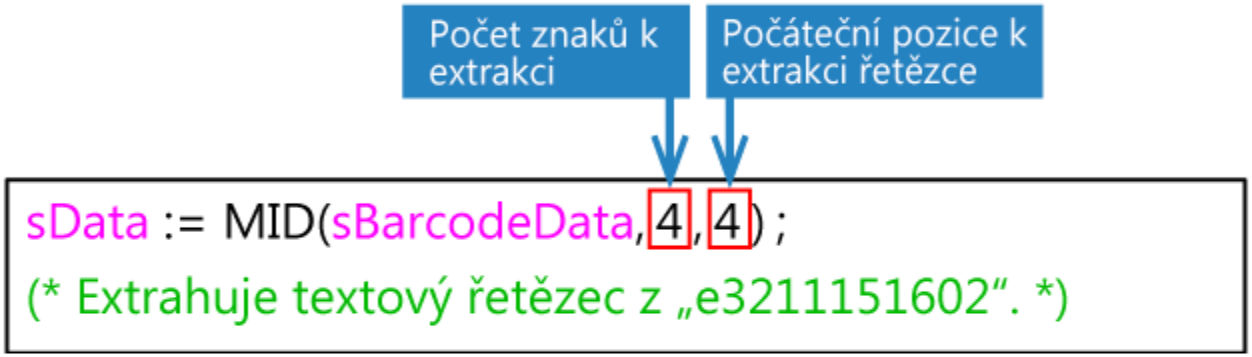


- st3rdLine.bStatus
- st3rdLine->bStatus
- st3rdLine[bStatus]
- st3rdLine[1]

Odpovědět Zpět

Test — Závěrečný test 12

Manipulace s řetězci v ST
Následující program extrahuje specifický řetězec z řetězce „e3211151602“ uloženého v proměnné „sBarcodeData“.
Funkce MID extrahuje určený počet znaků počínaje specifikovanou počáteční pozicí.
Vyberte správný extrahovaný řetězec.



- 1151
- 1602
- e321
- 1115

Odpovědět Zpět

Test Hodnocení testu

Dokončili jste závěrečný test. Vaše výsledky jsou následující.
Závěrečný test ukončíte přechodem na další stránku.

Počet správných odpovědí: 12

Celkový počet odpovědí: 12

Procento: 100%

Pokračovat Zkontrolovat

Gratulujeme. Úspěšně jste prošli v testu.

Dokončili jste kurz Základy programování (Strukturovaný text).

Děkujeme za vaši účast v tomto kurzu.

Doufáme, že se vám lekce líbily a že informace získané v průběhu tohoto kurzu vám budou užitečné.

Celý kurz si můžete projít kolikrát chcete.

Zkontrolovat

Zavřít