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).
Úvod

Čel kurzu

Tento kurz pomocí strukturovaného textu (ST) vysvětluje, jak vytvářet řídící 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 řídících programů**
Tato kapitola popisuje, jak vytvořit vstupně-výstupní řídící 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
<table>
<thead>
<tr>
<th>Úvod</th>
<th>Jak používat tento elektronický výukový nástroj</th>
</tr>
</thead>
<tbody>
<tr>
<td>Přejdete na následující stránku</td>
<td>Přejdete na následující stránku.</td>
</tr>
<tr>
<td>Zpět na předchozí stránku</td>
<td>Zpět na předchozí stránku.</td>
</tr>
<tr>
<td>Přesunutí na požadovanou stránku</td>
<td>Zobrazí se „Obsah“, pomocí kterého můžete přejít na požadovanou stránku.</td>
</tr>
<tr>
<td>Ukončit výuku</td>
<td>Ukončit výuku.</td>
</tr>
</tbody>
</table>
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 Řídící programy
1.2 Vlastnosti ST a porovnání s ostatními programovacími jazyky dle normy IEC
1.1 Řídicí programy


**Vstupní zařízení**

**Programovatelný řádící**

**Výstupní zařízení**

řídicí program (Žebříkový diagram)

<table>
<thead>
<tr>
<th>Tlačítkový přepínač</th>
<th>Kontrolka</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0</td>
<td>Y10</td>
</tr>
</tbody>
</table>

Po napsání řídicího programu se kontrolka rozhodí 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


Následující tabulka uvádí vlastnosti programovacích jazyků IEC 61131-3.

<table>
<thead>
<tr>
<th>Vlastnosti programovacího jazyka</th>
<th>jazyka</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ladder Diagram</strong> (LD) (Žebříkový diagram)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Symboly pro kontakty a cívky slouží k vytvoření programu připomínajícího elektrický obvod.</td>
</tr>
<tr>
<td></td>
<td>• Programový tok mohou snadno pochopit a sledovat i začátečníci.</td>
</tr>
<tr>
<td><strong>Structured Text</strong> (ST) (Strukturovaný text)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Programy jsou napsány textově (znaky).</td>
</tr>
<tr>
<td></td>
<td>• ST snadno pochopí ti, kdo mají zkušenosti s psaním programů v programovacích jazycích C nebo BASIC.</td>
</tr>
<tr>
<td></td>
<td>• Výpočetní vzorce jsou podobné matematickým výrazům, takže není složité je pochopit.</td>
</tr>
<tr>
<td></td>
<td>• ST je vhodný k manipulaci s daty.</td>
</tr>
<tr>
<td><strong>Function Block Diagram</strong> (FBD) (Funkční blokový diagram)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Programy jsou psány pomocí usměrňování bloků s různými funkcemi a stanovením vztahů mezi bloky.</td>
</tr>
<tr>
<td></td>
<td>• FBD zlepšuje čitelnost, protože máte přehled o celé operaci.</td>
</tr>
<tr>
<td><strong>Sequential Function Chart</strong> (SFC) (Sekvenční funkční schéma)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Podmínky a procesy jsou zapsány pomocí tokových diagramů.</td>
</tr>
<tr>
<td></td>
<td>• Programový tok je snadno pochopitelný.</td>
</tr>
<tr>
<td><strong>Instruction List</strong> (IL) (Seznam instrukcí)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• IL je podobný strojovému jazyku.</td>
</tr>
<tr>
<td></td>
<td>• IL se dnes používá jen velmi zřídka.</td>
</tr>
</tbody>
</table>

Tento kurz popisuje, jak psát základní řídící programy pomocí ST.
1.3 Shrnutí

Obsah této kapitoly je následující:

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

Důležité body ke zvážení:

<table>
<thead>
<tr>
<th>Vztahy mezi systémy programovatelných kontrolérů a řídicími programy</th>
<th>Programovatelné řadiče pracují v souladu s řídicími programy. Provoz programovatelných řadičů lze vytvořením řídicích programů nakonfigurovat dle stanovených požadavků.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mezinárodní norma pro řídicí programy</td>
<td>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.</td>
</tr>
<tr>
<td>Vlastnosti ST</td>
<td>ST snadno pochopí ti, kdo mají zkušenosti s psáním 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.</td>
</tr>
</tbody>
</table>
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 řídící příkaz)
2.2 Příklad základního programu (přířazovací příkaz)
2.3 Numerický zápis
2.4 Sekvence vykonávání programu
Příklad základního programu (I/O řídící 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.

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

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 řádku představuje jeden příkaz. Každý příkaz končí středníkem (;).
Program je tvořen kombinací příkazů.
Další příklad ilustruje program používající příkaz přiřazení.
Následující příkaz přiřazuje konstantu „5“ v desítkové soustavě k proměnné „D10“.

```plaintext
D10 := 5; (* Přiřazuje konstantu „5“ jako počáteční hodnotu.*)
```

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.

<table>
<thead>
<tr>
<th>Typ numerického zápisu</th>
<th>Způsob zápisu</th>
<th>Příklad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binární</td>
<td>Přidejte prefix „2#“.</td>
<td>2#11010</td>
</tr>
<tr>
<td>Oktalová</td>
<td>Přidejte prefix „8#“.</td>
<td>8#32</td>
</tr>
<tr>
<td>Decimální</td>
<td>Přímý vstup</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Přidejte prefix „K“.</td>
<td>K26</td>
</tr>
<tr>
<td>Hexadecimální</td>
<td>Přidejte prefix „16#“.</td>
<td>16#1A</td>
</tr>
<tr>
<td></td>
<td>Přidejte prefix „H“</td>
<td>H1A</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Stav</th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Numerický zápis</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Zápis True/false</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

Následuje několik příkladů přiřazení hodnot k proměnným bitového typu.

**Numerický zápis**

\[
X_0 := 1; = \quad X_0 := \text{TRUE};
\]

**Numerický zápis**

\[
X_0 := 0; = \quad X_0 := \text{FALSE};
\]
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.*)

*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.

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:

<table>
<thead>
<tr>
<th>Základní program ST</th>
<th>Příkaz je nejmenším prvkem programů ST. Každý příkaz končí středníkem (;). Program je tvořen kombinací příkazů.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formát příkazu přiřazení</td>
<td>Pro přiřazování se používá operátor přiřazení (:=).</td>
</tr>
<tr>
<td>Numerický zápis</td>
<td>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“.</td>
</tr>
<tr>
<td>Sekvence vykonávání programu</td>
<td>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.</td>
</tr>
<tr>
<td>Komentář</td>
<td>Přidání komentářů do programu zvyšuje pochopitelnost operací. Komentáře jsou ohraničeny dvěma hvězdičkami (* *).</td>
</tr>
</tbody>
</table>
3. kapitola Vytváření I/O řídicích programů

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

3.1 I/O řídicí programy
3.2 Kombinování více podmínek
3.3 Definování významu proměnných
3.1 I/O řídící programy

Následuje příklad programu pro I/O řízení programovatelného řadiče.

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

Výstupní příkaz: OUT
Podmínka vykonání: (X0)
Výstupní zařízení: (Y10)

„OUT“ je výstupním příkazem. Argument specifikuje podmínku provedení a zařízení, na které je výstup směrován. 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);
```

Stejný program napsaný v LD

```
X0
```

```
Y10
```
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 instrukčních 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.

\[ Y70 := (X0 \text{ OR } Y70) \text{ AND } \text{NOT } X1; \]

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

V následující tabulce je uveden seznam logických operátorů.

<table>
<thead>
<tr>
<th>Operátor</th>
<th>Význam</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR</td>
<td>Logické NEBO</td>
</tr>
<tr>
<td>AND</td>
<td>Logické A</td>
</tr>
<tr>
<td>NOT</td>
<td>Logická negace</td>
</tr>
<tr>
<td>XOR</td>
<td>Vyučovací NEBO</td>
</tr>
</tbody>
</table>
3.3 Definování významu proměnných

Pomocí ST lze u programovatelných řadičů MELSEC přidat 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.

\[
Y10 := (X0 \text{ OR } X1) \text{ AND } X2; \quad (* \text{ Zápis za využití názvů zařízení } *)
\]

\[
\text{Lamp} := (\text{Switch0 OR Switch1}) \text{ AND } \text{Switch2}; \quad (* \text{ Zápis pomocí symbolické proměnné } *)
\]

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:

<table>
<thead>
<tr>
<th>Kombinování více podmínek</th>
<th>Ke kombinování podmínek se v ST používají logické operátory.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definování významu proměnných</td>
<td>Při přiřazení symbolické proměnné odpovídající příslušné aplikaci, bude operace pochopitelnější.</td>
</tr>
</tbody>
</table>
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
4.2 Datové typy proměnných
4.3 Názvy proměnných představujících datové typy
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

```
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.

![Program diagram]

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

V následující tabulce je uveden seznam základních aritmetických operátorů.

<table>
<thead>
<tr>
<th>Operátor</th>
<th>Význam</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Sčítání</td>
</tr>
<tr>
<td>-</td>
<td>Odčítání</td>
</tr>
<tr>
<td>*</td>
<td>Násobení</td>
</tr>
<tr>
<td>/</td>
<td>Dělení</td>
</tr>
</tbody>
</table>
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.

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

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.

<table>
<thead>
<tr>
<th>Typ dat</th>
<th>Datový rozsah</th>
<th>Prefix</th>
<th>Rozšíření prefixu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>Stav ON/OFF bitových zařízení a stav true/false pro výsledky provedení operace</td>
<td>b</td>
<td>Bit (bit)</td>
</tr>
<tr>
<td>Integer (celé číslo)</td>
<td>Word (slovo bez znaménka)</td>
<td>u</td>
<td>unsigned word (slovo bez znaménka)</td>
</tr>
<tr>
<td></td>
<td>Word (slovo se znaménkem)</td>
<td>w</td>
<td>signed word (slovo se znaménkem)</td>
</tr>
<tr>
<td></td>
<td>Double-word (dvojité slovo bez znaménka)</td>
<td>ud</td>
<td>unsigned double-word (dvojité slovo bez znaménkem)</td>
</tr>
<tr>
<td></td>
<td>Double-word (Dvojité slovo se znaménkem)</td>
<td>d</td>
<td>signed double-word (dvojité slovo se znaménkem)</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Základní aritmetické operace</th>
<th>K vyjádření výpočtu lze v ST používat operátory běžné v obvyklých programovacích jazycích.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datové typy proměnných</td>
<td>U každé proměnné musí být zadán datový typ, aby došlo k definování rozsahu hodnot, s nimiž se bude manipulovat.</td>
</tr>
<tr>
<td>Přidávání názvů proměnných představujících datové typy</td>
<td>Popis názvů proměnných pomocí maďarského zápisu umožňuje identifikovat nekonzistentní datové typy proměnných již při psaní programu.</td>
</tr>
</tbody>
</table>
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)
5.2 Podmíněné větvení dle celočíselných hodnot (CASE)
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 ilustrovaný 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.

- Podmiňovací výrazy
- Hodnota k porovnání

IF wTemperature < 0 THEN

- Proměnná
- Operátor porovnání

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.

<table>
<thead>
<tr>
<th>Operátor</th>
<th>Význam</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>Větší než</td>
</tr>
<tr>
<td>&lt;</td>
<td>Menší než</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Větší nebo rovno</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Menší nebo rovno</td>
</tr>
<tr>
<td>=</td>
<td>Rovná se</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Nerovná se</td>
</tr>
</tbody>
</table>
Větvení při nesplnění příkazu IF (ELSE)

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

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

```c
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 větvění u příkazu IF (ELSIF)

Příkazy ELSE slouží k provedení příkazu při nesplnění podmiňovacího výrazu.
Dalšího podmíněného větvění 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 ELSIF.

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 stupnů. *)
  bCooler := 0; (* Vypne chladič, když teplota klesne pod 0 stupnů. *)
ELSIF 40 < wTemperature THEN
  bHeater := 0; (* Vypne ohřívač, pokud teplota přesáhne 40 stupňů. *)
  bCooler := 1; (* Zapne chladič, když teplota přesáhne 40 stupnů. *)
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.

```plaintext
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 „Přehráte“.

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

<table>
<thead>
<tr>
<th>Hmotnost</th>
<th>uSize</th>
<th>Velikost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 až 20 kg</td>
<td>1</td>
<td>M</td>
</tr>
<tr>
<td>21 až 30 kg</td>
<td>2</td>
<td>L</td>
</tr>
<tr>
<td>31 až 40 kg</td>
<td>3</td>
<td>XL</td>
</tr>
<tr>
<td>41 kg a více</td>
<td>4</td>
<td>Oversize</td>
</tr>
</tbody>
</table>
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 zdůraznění:

| 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 řídících 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)
6.2 Vytváření smyček (FOR)
6.3 Ukládání souvisejících dat (struktury)
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.

<table>
<thead>
<tr>
<th>Cíl</th>
<th>Země A</th>
<th>Země B</th>
<th>Země C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Výrobní objem</td>
<td>35</td>
<td>75</td>
<td>65</td>
</tr>
</tbody>
</table>

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].

<table>
<thead>
<tr>
<th>uProduction [0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Název proměnné</td>
</tr>
</tbody>
</table>

**Cíl (řádek)**

- Země A [0]: 35
- Země B [1]: 75
- Země C [2]: 65

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

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

Nyní kromě údaje o cíli uložíme také údaj o barvě.

<table>
<thead>
<tr>
<th>Cíl</th>
<th>Země A</th>
<th>Země B</th>
<th>Země C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barva laku</td>
<td><img src="image1" alt="Barva laku" /></td>
<td><img src="image2" alt="Barva laku" /></td>
<td><img src="image3" alt="Barva laku" /></td>
</tr>
<tr>
<td>Výrobní objem</td>
<td>10</td>
<td>15</td>
<td>25</td>
</tr>
</tbody>
</table>

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

![Barva laku diagram](image4)

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

Proměnná pole (dvourozměrné pole): `uProduction[1,1]`
### 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.

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

<table>
<thead>
<tr>
<th>Cíl</th>
<th>Země A</th>
<th>Země B</th>
<th>Země C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barva laku</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Výrobní objem</td>
<td>10</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>20</td>
<td>10</td>
</tr>
</tbody>
</table>

**Dalších 5 vozů**

**Table:**

<table>
<thead>
<tr>
<th>Cíl (řádek)</th>
<th>Země A</th>
<th>Země B</th>
<th>Země C</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,0]</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[0,1]</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[0,2]</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[1,0]</td>
<td>15</td>
<td>[1,1]</td>
<td>[1,2]</td>
</tr>
<tr>
<td></td>
<td>40 -&gt; 45</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>[2,0]</td>
<td>25</td>
<td>[2,1]</td>
<td>[2,2]</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

**Barva laku (sloupec):**

- Červená
- Žlutá
- Modrá
6.1.3 Zpracování informací uložených ve více rozměrných polích

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

\[
u_{ProductionToday} := u_{Production}[2,0] + u_{Production}[2,1] + u_{Production}[2,2];
\]

(* Vypočítá celkový objem plánované produkce pro dnešní den pro všechny barvy laků pro zemi C a hodnotu poté přirádí do proměnné „u_{ProductionToday}“. *)

<table>
<thead>
<tr>
<th>Cíl</th>
<th>Země A</th>
<th>Země B</th>
<th>Země C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barva laku</td>
<td><img src="Cars.png" alt="Cars" /></td>
<td><img src="Cars.png" alt="Cars" /></td>
<td><img src="Cars.png" alt="Cars" /></td>
</tr>
<tr>
<td>Výrobní objem</td>
<td>35 celkem</td>
<td>80 celkem</td>
<td>65 celkem</td>
</tr>
</tbody>
</table>

![Barva laku (sloupec)](Barva.png)

![Země C: Dnešní plánovaný objem výroby: 65](Země.png)
6.2 Vytváření smyček (FOR)

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

\[
u\text{ProductionToday} := u\text{Production}[2,0] + u\text{Production}[2,1] + u\text{Production}[2,2];
\]

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

\[
u\text{ProductionToday} := u\text{Production}[2,0] + u\text{Production}[2,1] + u\text{Production}[2,2] + u\text{Production}[2,3] + u\text{Production}[2,4] + u\text{Production}[2,5] ...\]

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.

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

Výraz je opakovan, dokud nebude dosažena finální hodnota proměnné a proveden kód „END_FOR;“. 
Vytváření smyček (FOR)

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

```c
uProductionToday := 0;
FOR wColor := 0 TO 2 BY 1 DO
    uProductionToday := uProductionToday + uProduction[2,wColor]; (* Zvyšuje objem plánované výroby. *)
END_FOR;
```

Za využití příkazu FOR se proměnná „wColor“ navýšuje 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.


Operace tohoto programu jsou znázorněny na další straně.
6.2 *Vytváření smyček (FOR)*

Provdění příkazu FOR je popsáno pomocí operací příkladu programu.

<table>
<thead>
<tr>
<th>Pole odhadovaného výrobního objemu</th>
<th>Červená</th>
<th>Žlutá</th>
<th>Modrá</th>
</tr>
</thead>
<tbody>
<tr>
<td>Země A</td>
<td>[0,0] 10</td>
<td>[0,1] 5</td>
<td>[0,2] 20</td>
</tr>
<tr>
<td>Země B</td>
<td>[1,0] 15</td>
<td>[1,1] 45</td>
<td>[1,2] 20</td>
</tr>
</tbody>
</table>

Kliknutím na můžete pokračovat na další stranu. Za účelem opakovaného přehrávání animace klikněte na tlačítko „Přehrát“.

```pseudocode
uProductionToday := 0;

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

Number of repetition of the loop: 3
# 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.

<table>
<thead>
<tr>
<th>Položka</th>
<th>Název proměnné</th>
<th>Hodnota</th>
<th>Datový typ proměnné</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>sModel</td>
<td>„ST TRUCK“</td>
<td>Textový řetězec</td>
</tr>
<tr>
<td>Stav</td>
<td>bStatus</td>
<td>„ve výrobě“</td>
<td>Bitový typ</td>
</tr>
<tr>
<td>Cílový objem výroby pro dnešní den</td>
<td>uPlanQty</td>
<td>„100“</td>
<td>Celočíselný typ word (bez znaménka)</td>
</tr>
<tr>
<td>Aktuální objem výroby</td>
<td>uActualQty</td>
<td>„88“</td>
<td>Celočíselný typ word (bez znaménka)</td>
</tr>
</tbody>
</table>

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 čítnost.
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.

**Více proměnných**

- s1stLineModel
- b1stLineStatus
- u1stLinePlanQty
- u1stLineActualQty

**Struktura**

```
struct st1stLine {
    sModel:
    bStatus:
    uPlanQty:
    uActualQty:
}
```

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.

```
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.

Struktura řazená podle data (stProductionByDate)

Den (sloupec)

<table>
<thead>
<tr>
<th>Leden</th>
<th>[1,1]</th>
<th>[1,2]</th>
<th>...</th>
<th>[1,21]</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2,1]</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>[7,1]</td>
<td>...</td>
<td>...</td>
<td>[7,21]</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Měsíc (řádek)

<table>
<thead>
<tr>
<th>Červenec</th>
<th>[7,1]</th>
<th>...</th>
<th>...</th>
<th>[7,21]</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2,1]</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Struktura, která je přiřazen stav výroby k 21. červenci

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.

<table>
<thead>
<tr>
<th>Struktura řazená podle data (stProductionByDate)</th>
<th>Den (sloupec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leden</td>
<td>[1,1]</td>
</tr>
<tr>
<td>Měsíc  Řádek</td>
<td></td>
</tr>
<tr>
<td>Červen</td>
<td>[6,1]</td>
</tr>
</tbody>
</table>

Struktura, které je přiřazen stav výroby k 1. červnu, je uložena

Proměnná, které je přiřazen objem výroby

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 Shrunutí

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 přídáním tečky a názvu člena za název strukturní proměnné. |
7. kapitola Manipulace s řetězovými daty

V některých případech využívají programovatelné řadiče řetězová 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 řetězová data.

Tato kapitola popisuje, jak manipulovat s řetězovými daty.

7.1 Příklad manipulace s řetězovými daty
7.2 Přiřazení řetězce
7.3 Extrahování řetězců (LEFT)
7.4 Extrahování řetězců (MID)
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

- e112, 12091458
  - 4znakový kód chyby
  - 8znakové datum vytvoření chyby

Funkce pro zpracování řetězce

- e112
- 12091458

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, popište tato sekce datové typy řetězců.

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

<table>
<thead>
<tr>
<th>Datový typ</th>
<th>Lze zpracovat znakový typ</th>
<th>Prefixy dle maďarského zápisu</th>
<th>Rozšíření prefixu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Řetězec</td>
<td>Řetězce alfanumerických znaků a čísel (ASCII) nebo japonských znaků (Shift-JIS)</td>
<td>s</td>
<td><strong>string</strong> (řetězec)</td>
</tr>
<tr>
<td>String [Unicode]</td>
<td>Řetězce v řadě různých jazyků a s mnoha symboly</td>
<td>ws</td>
<td><strong>wide string</strong> (řetězec s rozšířenou znakovou sadou)</td>
</tr>
</tbody>
</table>

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“.

<table>
<thead>
<tr>
<th>Název proměnné</th>
<th>Uložený řetězec</th>
</tr>
</thead>
<tbody>
<tr>
<td>sBarcodeData</td>
<td>e112, 12091458</td>
</tr>
</tbody>
</table>

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

\[
\text{sErrorCode} := \text{LEFT} (\text{sBarcodeData}, 4);
\]

(* Extrahuje 4 znaky chybového kódu zleva *)

Jsou extrahovány čtyři znaky zleva. Levé straně je přiřazena hodnota „e112“, což je řetězec představující chybový kód.
Extrahování řetězců (MID)

Je extrahován čas vytvoření chyby „12091458“ z řetězové proměnné „sBarcodeData“ obsahující řetězec „e112,12091458“.

<table>
<thead>
<tr>
<th>Název proměnné</th>
<th>Uložený řetězec</th>
</tr>
</thead>
<tbody>
<tr>
<td>sBarcodeData</td>
<td>e112,12091458</td>
</tr>
</tbody>
</table>

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

Výstup: Vstup, Počet extrahovaných znaků, Pozice prvního znaku, který má být extrahován

```
sErrorDate := MID (sBarcodeData, 8, 6);
(* Extrahuje 8 znaků počínaje šestým znakem zleva. *)
```

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)” (MELOSOFT 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 tématata zopakovat. Tento závěrečný test obsahuje celkem 12 otázek (20 položek). Závěrečný test můžete podstoupit kolikrát chcete.

Způsob provedení testu
Po vybraní 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.

• 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.
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.
Základní principy ST
Vyberte správný příkaz zapsaný v ST.

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

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

- Hodnota zůstává 101.
- Hodnota se neustále mění.
Kombinování více podmínek
Vyberte správný příklad programu v ST, který provádí stejnou operaci jako následující vzorový příklad v LD.

\[
\text{bResult0} := (\text{bResult0 OR bFlag1}) \text{ AND bFlag2};
\]
\[
\text{bResult1} := \text{bResult0 AND bFlag3};
\]

\[
\text{bResult0} := (\text{bFlag0 OR bFlag2}) \text{ AND bFlag1};
\]
\[
\text{bResult1} := \text{bResult0 AND bFlag3};
\]
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 prázdnu sekcí vzorového programu.
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--
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)

<table>
<thead>
<tr>
<th></th>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
<th>Model 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Země X</td>
<td>[0,0]</td>
<td>[0,1]</td>
<td>[0,2]</td>
<td>[0,3]</td>
</tr>
<tr>
<td>Země Y</td>
<td>[1.0]</td>
<td>[1.1]</td>
<td>[1.2]</td>
<td>[1.3]</td>
</tr>
<tr>
<td>Země Z</td>
<td>[2.0]</td>
<td>[2.1]</td>
<td>[2.2]</td>
<td>[2,3]</td>
</tr>
</tbody>
</table>
Pole v ST a opakované příkazy
Následující vzorový program získává celkový výrobní objem ve stejné dnechným týdnu. Z pole, které ukládá denní produkční objemy je získán součet za 4 týdny. Vyberte správné číslo, které vzorový program vrátí.

```pseudocode
uTotalProduction := 0;
FOR wOnceAWeek := 1 TO 4 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)**

<table>
<thead>
<tr>
<th>Den</th>
<th>Produkční objem dne 1. února (tyden 1)</th>
<th>Po 1 týdnu</th>
<th>Produkční objem dne 8. února (tyden 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Den 1</td>
<td>[1,1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Den 2</td>
<td>[1,2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Den 3</td>
<td>[1,3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Den 4</td>
<td>[1,4]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Den 5</td>
<td>[1,5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Den 6</td>
<td>[1,6]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Den 7</td>
<td>[1,7]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Den 8</td>
<td>[1,8]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Leden</td>
<td>[2,1]</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Únor</td>
<td>[2,2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2,3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2,4]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2,5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2,6]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2,7]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2,8]</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Odpověď: 8
Charakteristiky struktur v ST
Vyberte nesprávný popis struktur.

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

<table>
<thead>
<tr>
<th>Parametr</th>
<th>Název proměnné</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>sModel</td>
</tr>
<tr>
<td>Stav</td>
<td>bStatus</td>
</tr>
<tr>
<td>Cílová produkce pro současný den</td>
<td>uPlanQty</td>
</tr>
<tr>
<td>Aktuální objem výroby</td>
<td>uActualQty</td>
</tr>
</tbody>
</table>

Název proměnné: st3rdLine.bStatus
Členové: sModel, uPlanQty, uActualQty

Odpověď: st3rdLine.bStatus
Manipulace s řetězcem 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.

```
sData := MID(sBarcodeData, 4, 4);  (* Extrahuje textový řetězec z „e3211151602“. *)
```

- 1151
- 1602
- e321
- 1115
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