PRAKTICKÉ VYUŽITÍ PROGRAMOVÝCH MODULŮ A JEJICH DOKUMENTACE

Ing.Zdaněk Tomka
Ústav vývoje a racionalizace železničního stavebnictví Brno

Příspěvek se zabývá praktickým využíváním modulů v relativní a zaváděcí formě, které jsou volány z řídícího programu v jazyce COBOL, zařazením této techniky do metodiky modulárního programování, dále pak dokumentací těchto modulů a na závěr jsou uvedeny příklady praktického využití modulů v našem VS.

1. Úvod

Vytváření modulů v relativní a zaváděcí formě je jedna z technik modulárního programování. V tomto příspěvku se budeme zabývat využitím takovýchto modulů za podmínky, že hlavním programovacím jazykem je jazyk COBOL a tudíž moduly budou volány z řídícího programu vytvořeného v jazyku COBOL.

Relativní a zaváděcí moduly jsou stavebnicové programové prvky, ze kterých vytváříme programový celek tak, že tyto moduly voláme z řídícího programu instrukcí CALL. Na tyto stavební prvky jsou kladeny určité nároky. Je to především podmínka jednoho vstupu do modulu a jednoho výstupu z něho. Moduly by měly být obecné, aby se daly využívat při různých aplikacích. Jsou však vhodné i moduly jednoúčelové, které usnadňují kolektivní práci a zvyšují srozumitelnost programu, zvláště u složitějších zadání.

Zadání modulů (podprogramů) vzniká dekompozicí úlohy metodou "Top-down", t.j. postupným zjemňováním problému. Moduly jsou pak volány z řídícího programu a z těchto modulů můžeme volat opět moduly na nižší úrovni. Z podprogramů na nejnižší úrovni se pak vracíme stejnou cestou zpět do řídící části programu. Vzniká tak hierarchická struktura programu. Tato struktura by měla mít však maximálně tři úrovně, aby program byl srozumitelný.

Volané podprogramy mohou být vypracovány v těchto různých jazycích: COBOL, ASSEMBLER, FORTRAN, PL1.
Volání podprogramu v jazyce PL1 je však komplikované a nedoporučuje se i v literatuře IBM. U jazyka PL1 jsou odlišné konvence

při předávání parametrů oproti běžným systémovým - předávání parametrů se děje přes informační vektory. Pro vyrovnání tohoto prostředí je zpravidla nutné volat další assemblerovský program. Proto pro volané podprogramy doporučujeme používat jen jazyky COBOL, ASSEMBLER, FORTRAN.

2. Zařazení techniky vytváření modulů v relativní a zaváděcí formě do metodiky modulárního programování

Do metodiky modulárního programování patří tyto základní techniky:

a) vytváření modulů ve zdrojové formě
b) vytváření modulů v relativní formě
c) vytváření modulů v zaváděcí formě
d) vytváření obecných parametrických programů.

a) Moduly ve zdrojové formě.

Jedná se o části strukturovaného programu, které splňují nejdůležitější požadavek na modul - jeden vstup, jeden výstup. V cobolském textu programu je to sekce, odstavec nebo kombinace obou, které jsou volány příkazem PERFORM. Hovoříme o vnitřní modularitě programu. Při víceúčelovém využití pracujeme s takovýmto modulem jako s textem uloženým na zdrojové knihovně a pomocí cobolského příkazu COPY nebo ./ IN generátoru zdrojových programů jej vkládáme před kompilací do textu programu. Další možnost je generování zdrojových částí programu generátorem.

b) Moduly v relativní formě.

Relativní modul vzniká jako produkt kompilátoru určité programové části - podprogramu. Vzniklý modul si můžeme uložit na relativní nebo zdrojovou knihovnu a později jej využít nebo jej můžeme přímo přes dočasnou knihovnu sestavit s řídícím programem v rámci jobu do jednoho programu. Relativní modul je volán z volajícího programu příkazem CALL.

c) Moduly v zaváděcí formě.

Tyto moduly vznikají kompilací a sestavením určité programové části - podprogramu. Vzniklý modul v zaváděcí formě můžeme uložit na knihovnu zaváděcích modulů a později jej sestavit s řídícím programem do jednoho programu. Moduly jsou volány

z řídícího programu příkazem CALL.

d) Obecné parametrické programy.

Jsou to obecné programy, u kterých můžeme vyvolávat různé funkce určité třídy úloh zadáním podmínek, které se zadávají pomocí parametrů. Programy jsou k dispozici na knihovně v zaváděcí formě a není tudíž nutná při aplikaci kompilace a sestavení. U této techniky se jedná o modularitu na vyšší úrovni, než je úroveň jednotlivého programu.

Z výše uvedených přístupů k modulárnímu programování si si programátor vybírá nejvhodnější možnost podle konkrétního zadání úlohy, důležité však je pohlížet na zadání jako na celek a postupovat metodou postupného zjemňování problému. Nejvýhodnější a nejčastější je aplikování techniky vnitřní modularity. Pro části programů, které by nezvládl jazyk COBOL je vhodné používat podprogramy v jiných jazycích, které mohou být potom jako relativní moduly v rámci jobu sestaveny s řídícím programem do jednoho zaváděcího modulu. Pro častěji se opakující funkce v programech je vhodné si tuto funkci nebo algoritmus připravit jako modul v zaváděcí formě. Moduly v relativní a zaváděcí formě také usnadňují ladění rozsáhlejších programů tím, že je můžeme odladit samostatně. Uvedené techniky modulárního programování lze vhodně kombinovat.

3. Moduly v jazyku COBOL

Hlavní zásady při předávání řízení a proměnných mezi programem řídícím a podprogramem (modulem):

- formální parametry (externí proměnné) se deklarují v podprogramu v LINKAGE SECTION na úrovni 01 nebo 77 a zapisujeme je také v doložce USING zápisu PROCEDURE DIVISION nebo ENTRY v podprogramu
- skutečné parametry jsou deklarovány ve WORKING-STORAGE SECTION nebo ve FILE SECTION řídícího programu a zapisujeme je v doložce USING příkazu CALL
- předávané proměnné musí mít stejné zobrazení a při rozdílné délce musí dát programátor pozor, aby nepřemazal obsah jiných proměnných v oblasti řídící části programu.

- do podprogramu se předá řízení buď na první instrukci PROCEDURE DIVISION, jestliže v příkazu CALL uvedeme jméno v PROGRAM-ID modulu nebo na instrukci za příkazem ENTRY modulu, jestliže v příkazu CALL uvedeme jméno obsažené v tomto zápisu ENTRY.

Příklad zápisu řídícího programu a podprogramu:

řídící program	podprogram
: FILE SECTION. : 02 PS1 PIC 99. : WORKING-STORAGE SECTION. 77 PS2 PIC XX. : PROCEDURE DIVISION. : CALL "CRSUB1" USING PS1 PS2 : CALL "A1" USING PS1 PS2. :	: PROGRAM-ID. CRSUB1. : LINKAGE SECTION. 01 PF2 PIC XX. 01 PF1 PIC 99. : PROCEDURE DIVISION USING PF1 PF2 : ENTRY "A1" USING PF1 PF2. :

Moduly můžeme uchovat v relativní nebo zaváděcí formě k pozdějšímu sestavení s řídícím programem. Lepší je však ai připravit moduly v zaváděcí formě. Moduly v jazyku COBOL jsou vhodné pro častěji se opakující . funkce nebo algoritmy v programech. Jednoúčelové podprogramy je lépe řešit vnitřní modularitou programu (příkaz PERFORM). Také opakující se algoritmy v rámci jednoho programu je lépe řešit vnitřní modularitou.

4. Moduly v jazyku ASSEMBLER

Předávání řízení a proměnných mezi řídícím programem a podprogramem se děje podle konvencí OS.

Obsazení registrů:

R1 ... adresa seznamu adres parametrů

R13... adresa oblasti úshovy
Je to adresa oblasti volajícího programu,do kterého podprogram uschovává obsahy registrů.

R14... adresa návratu do volajícího programu

R15... adresa vstupního bodu volaného programu
Při návratu můžeme použít registr jako registr návratového kódu.

Adresy jednotlivých předávaných proměnných můžeme získat instrukcí LM. Například pro 2 parametry použijeme instrukci
LM 3,4,0(1) ,která způsobí, že v registru 3 máme adresu prvního parametru, v registru 4 adresu druhého parametru.

V podprogramu si definujeme rovněž oblast úschovy o velikosti 18 slov. Př. SAVEAR DS 18F
Na adresu třetího bytu v oblasti úschovy podprogramu uložíme adresu oblasti úschovy volajícího programu (R13) a před návratem z podprogramu obnovíme registr 13 a potom zbývající registry.

Příklad zápisu řídícího programu a podprogramu:

řídící program	podprogram
.	M1 CSECT
FILE SECTION.	USING *,12 BÁZOVÝ REGISTR
:	:
02 PS1 PIC XX.	(M1 ..) VSTUPNÍ BOD
:	:
WORKING-STORAGE SECTION.	SAVE (14,12) ÚSCHOVA REGISTRŮ
77 PS2 PIC 99.	LR 12,15
PROCEDURE DIVISION.	ST 13,SAVEAR+4 ULOŽENÍ ADR. OBLASTI ÚSCHOVY
.	LM 3,4,0(1)
CALL "M1" USING PS1 PS2.	:
:	vlastní podprogram
CALL "R1" USING PS1 PS2.	:
:	L 13,SAVEAR+4 OBNOVA R13
	RETURN (14,12),RC=0 OBNOVA REG.
	(ENTRY R1)
	SAVEAR DS 18F
	:

Moduly v jazyku assembler jsou vhodné pro části zadání programu, na které jazyk COBOL nestačí. Jednoúčelové procedury můžeme sestavit s přeloženým řídícím programem v rámci jobu, přes relativní moduly do jednoho programu. U víceúčelových procedur je vhodné si tyto podprogramy připravit v zaváděcí formě. Takovéto moduly připravují většinou specialisté, dobře seznámení s jazykem assembler.

5. Moduly v jazyku FORTRAN

Při předávání řízení a proměnných mezi řídícím programem a podprogramem v jazyku FORTRAN platí tyto základní vztahy:

- hodnoty skutečných proměnných (parametrů) z řídícího programu jsou předávány do podprogramů pomocí formálních parametrů

SUBROUTINE

- předávání se děje podle pořadí v klausuli USING příkazu CALL a podle pořadí formálních parametrů v zápisu SUBROUTINE. Hodnoty odpovídajících si proměnných dle pořadí se ztotožní
- zobrazení v paměti odpovídajících si proměnných musí být stejné (viz tabulka):

COBOL	FORTRAN	DÉLKA B
PIC 9(2až4) COMP	INTEGER*2	2
PIC 9(5až9) COMP	INTEGER nebo proměnné I až N	4
PIC 9(..) COMP-1	REAL nebo proměnné jiné než I až N	4
PIC 9(..) COMP-2	REAL*8	8

- skutečné proměnné se deklarují ve FILE SECTION nebo WORKING-STORAGE SECTION a mají USAGE COMP, COMP-1, COMP-2
- návrat do řídícího programu zabezpečíme příkazem RETURN.

Příklad zápisu řídícího programu a podprogramu:

řídící program	podprogram
: WORKING-STORAGE SECTION. 77 PS1 PIC 9(5) COMP. 77 PS2 PIC 9(5) COMP-1. PROCEDURE DIVISION. : CALL "FPROC" USING PS1 PS2 :	SUBROUTINE FPROC(K1,S2) : S2 = SQRT(FLOAT(K1)) : RETURN END

Fortranské podprogramy jsou vhodné pro řešení částí programů, ve kterých jsou zadány jiné funkce než základní aritmetické. (Např. logaritmus,odmocnina,atd.). Většinou se jedná o jednoúčelovou proceduru, kterou můžeme sestavit v rámci jobu s přeloženým řídícím programem, přes relativní moduly do jednoho programu. Pro víceúčelové procedury si můžeme tento podprogram připravit v zaváděcí formě a uložit na knihovnu k pozdějšímu využití. Subroutine bývají většinou jednoduché, vyvolávají se v nich jen potřebné aritmetické funkce. Programátor tedy nemusí znát podrobně jazyk FORTRAN, ale stačí většinou reprezentační příklad a seznam vnitřních funkcí fortranu.

6. Dokumentace modulů

Při vytváření modulů v zaváděcí, případně relativní formě pro obecné využití je třeba zpracovat dokumentaci k tomuto modulu. K vytvořenému textu podprogramu si zpracuje programátor běžnou programovou dokumentaci. Důležitý je však "Návod k použití modulu", který by měl mít jednotnou formu a měl by obsahovat tyto části:

a) Titulní list modulu.
b) Detailní popis funkcí modulu.
c) Popis parametrů.
d) Podmínky použití.
e) Příklad použití modulu.

Takovýto modul s dokumentací můžeme pak zařadit mezi programové typové prvky ASŘ a používat je v rámci subsystému, VS nebo i k širšímu použití.

Evidence typových prvků se provádí v "Knize typových prvků", která se vede v týmech pro jednotlivé subsystémy. Tato kniha má hlavní význam, aby jednotlivým typovým prvkům byla přidělena jednoznačná jména a má formu:

Označení typ. prvku	jazyk	stručná charakteristika	autor	zahá-jení	ukon-čení	forma
R01	C	výpočet kontr. čísla	Bystrý	14.5	15.6	L
R02	A	test bit.řetězce	Tomka	2.6	2.7	L

Označení typového prvku má skladbu

X XX(X)
- X — zkratka subsystému ASŘ, ve kterém prvek vznikne
- XX — pořadové číslo typového prvku v rámci subsystému
- (X) — číslo, udávající pořadí doplňující vstupního bodu

- jazyk C(COBOL), A(ASSEMBLER), F(FORTRAN)
- forma L(zaváděcí modul), R(relativní modul), S(zdrojový modul), O(obecný parametrický program), G(generátor textů programu)

a) Titulní list modulu.

Vyplněný formulář "Titulní list modulu", který je dále uveden, slouží jednak jako úvodní list dokumentace "Návod k použití modulu", jednak může být zařazen do centrálně vedené

kartotéky vybraných modulů.

Vysvětlení k jednotlivým částem formuláře:

- hlavní vstupní bod

  Název hlavního vstupního bodu je jméno ze zápisu PROGRAM-ID, CSECT nebo SUBROUTINE modulu, je shodné s označením typového prvku a pod tímto jménem je také modul ukládán na knihovnu.

- vstupní body

  Jedná se o další vstupní body modulu dané použitím zápisu ENTRY. Každý vstupní bod je považován za nový typový prvek a jeho název je shodný s označením tohoto typového prvku. Pro každý takovýto vstupní bod bude vyplněn jeden "Titulní list modulu", kde v části"vstupní body"vyplníme jen popisovaný vstupní bod a v části "účel typového prvku" uvedeme jen funkce, vztahující se k popisovanému vstupnímu bodu. Při popisování hlavního vstupního bodu uvedeme názvy všech doplňujících vstupních bodů, s příslušnými dalšími, doplňujícími informacemi.

- účel typového prvku

  Zde uvedeme, pro jaký účel byl modul vytvořen, stručný popis funkce modulu, seznam jmen použitých datových struktur ze zdrojové knihovny (jedná se o struktury potřebné ve volajícím programu při využívání modulu) a jiné podstatné informace důležité pro předběžné seznámení s modulem a jeho využitelnosti. Jestliže bude modul doplněn strukturami, pak tyto struktury uložíme na zdrojové knihovně pod názvem odvozeným od jména modulu: XXX W X

  - X: pořadové číslo struktury pro daný modul
  - W: znak W udává, že se jedná o strukturu
  - XXX: jméno modulu (hlavní vstupní bod)

b) Detailní popis funkcí modulu

V této části popíšeme detailně a přesně funkce modulu a funkce, odpovídající jednotlivým vstupním bodům. Při složitějších algoritmech můžeme popis doplnit vývojovým diagramem.

c) Popis parametrů

Zde uvedeme a popíšeme předávané proměnné mezi řídícím programem a podprogramem (skutečné a formální parametry). Tyto parametry můžeme rozdělit na vstupní do modulu, vystupující z modulu a pomocné.

d) Podmínky použití

V této části popíšeme, jakým způsobem funkci modulu nebo jeho podfunkcí (danými doplňujícími vstupními body) vyvoláme. Dále uvedeme, jak použijeme parametry nebo připravené struktury dat. Je-li to nutné, uvedeme také softwarové a hardwarové omezení pro volání modulu.

e) Příklad použití modulu

Návod doložíme příkladem výpočtu řídícího programu, který modul volá.

7. Příklady vytvořených modulů

Ve VS ÚVAR ŽS jsme vytvořili a používáme například tyto moduly v zaváděcí formě:

R02 - modul pro výpočet a kontrolu kontrolní číslice číselného údaje algoritmem "modulo 10".

R04 - modul pro umožnění zápisu do jednotlivých bitů. Takto lze vytvářet bitové řetězce, což jazyk COBOL neumožňuje. Pro záznam na magnetické médium můžeme při zápisu indikátorů nějakých stavů do věty, ušetřit až 8-násobek délky těchto informací.

R05 - modul pro umožnění testování jednotlivých bitů. Jedná se o opačnou funkci, kterou má modul R04.

R06 - modul pro třídění ve vnitřní paměti metodou Quick-sort dle více klíčových položek. Podprogram se hodí pro třídění malých souborů dat a pro přetřídění tabulek před příkazem SEARCH.

E01 - modul pro tisk statistických údajů na konci výpočtu programu. Podprogram usnadňuje práci výstupní kontroly při rutinních výpočtech.

Předpokládáme, že na tvorbě dalších modulů se budou podílet programátoři ze všech týmů jednotlivých subsystémů. Pro opakující se algoritmy v zadání programů ASŘ budou vytvářeny moduly, které jako stavební kameny programové realizace budou vybírány k centrálnímu použití. Tyto moduly budou rovněž poskytovány jiným výpočetním střediskům, zejména VS rezortu dopravy. Naopak náš ústav bude přebírat vhodné obecné typové prvky od jiných organisací pro svou potřebu.

ÚVOR ŽS BRNO OS CC 1033	TITULNÍ LIST POUŽITÍ MODULŮ	Autor:	
		Datum:	
Subsystém:	Symb. označení typového prvku:	Původní jméno programu:	

Forma modulu:	
Programovací jazyk:	
Knihovna:	

HLAVNÍ VSTUPNÍ BOD

VSTUPNÍ BODY

Název	Poznámka

Účel typového prvku (stručně funkce):