RNDr. Jaroslav HROUDA, CSc.
VÚ technicko-ekonomický chemického průmyslu, Praha

SEGMENTARIZACE JAKO METODA PROGRAMOVÁNÍ

Předmětem tohoto příspěvku je racionalizace tvorby programů - s omezením na jazyk PL/I. Program budeme chápat trojím způsobem, a to jako:

a) posloupnost příkazů programovacího jazyka,
b) strukturu funkcí,
c) posloupnost znakových vět (text).

Pojetí (a) je základem programátorského i strojového zpracování; lze z něho odvodit všechny funkce programu. Universálnost tohoto pojetí však někdy vede k příliš velkým a kompaktním celkům.

Význam pojetí (b) spočívá v tom, že umožňuje dekompozici programu na relativně samostatné jednodušší části (moduly). K podpoře funkčního pojetí programu má jazyk PL/I některé základní prostředky. Jsou to především interní a externí procedury a standardní funkce. Dále je to možnost vkládání úseků programů makroprocesorovým příkazem %INCLUDE (makroprocesor neboli preprocesor PL/I je standardní součástí kompilátoru). Důsledné použití těchto prostředků však naráží na hledisko provozní efektivnosti, které nelze zanedbat.

Pojetí (c) vystupuje do popředí při technickém zpracování programu. Máme tím na mysli pořízení programu na paměťovém médiu včetně reprodukce opakujících se nebo hotových (typových) částí. Dále to jsou úpravy programového textu při ladění a údržbě programu a vývoj programových variant.

V tomto příspěvku chceme referovat o jedné koncepci, která umožňuje - podle našeho názoru a zkušenosti - důsledněji realizovat výše zmíněné trojí pojetí programu v rámci jazyka PL/I, než to umožňují základní prostředky jazyka. Je to koncepce segmentarizace programů na úrovni makroprocesoru, kterou spolu s podpůrnými softwarovými prostředky nazýváme metodou segmentárního programování.

1. Podstata metody

Segmentem rozumíme jakýkoliv souvislý úsek zdrojového textu, který obsahuje jednak příkazy jazyka PL/I, jednak reference na jiné segmenty. Každý segment je opatřen jménem. Zdrojový program se vytváří ze segmentů procesem postupného vkládání. Tento proces začíná od segmentu, který je určen jako hlavní. Ostatní segmenty jsou vedlejší. Každá reference na segment se nahradí textem příslušného segmentu. Proces končí, když žádný z vložených segmentů již neobsahuje reference na další segmenty (předpokládá se tedy nerekursivnost referencí).

Popsaná inserční metoda je v programování všeobecně známá. V jazyku PL/I, jak už bylo uvedeno, ji lze realizovat makroprocesorem PL/I, tj. reference na segmenty mají formu příkazů %INCLUDE. Pro určité technické obtíže (vkládané úseky musí být samostatnými knihovními členy) se však v PL/I této možnosti využívá hlavně pro přenos částí zdrojových textů mezi programy (např. deklarace struktur). V metodě segmentárního programování jsme se pokusili uplatnit princip vkládání segmentů i v rámci programu samého. Segmenty jednoho programu lineárně sdružujeme do jednoho textového celku, který nazýváme preprogram. Tento text je uložen jako jeden člen ve zdrojové knihovně. Generace zdrojového programu procesem vkládání se uskuteční makroprocesorem až po rozkladu preprogramu na jednotlivé segmenty.

Kromě segmentů preprogramu lze používat i segmentů uložených v knihovnách jako samostatné členy - tzv. samostatné segmenty. Rozlišujeme samostatné segmenty standardní (s obecnou použitelností) a uživatelské (vytvořené pro speciální účel). Hlavní segment má vždy jméno $@ a musí být umístěn v preprogramu.

Vzájemné vztahy segmentů vytvořené referencemi %INCLUDE představují tzv. segmentární strukturu programu. Obecně je tato struktura síťová necyklická.

Zabývejme se nyní důležitou otázkou vymezení segmentů. Hlediska, která při tom uplatníme, vycházejí z pojetí (b) a (c), o nichž jsme pojednali v úvodu.

1) Funkční hledisko: segment realizuje určitou dílčí nebo hlavní funkci programu. Např. to může být

- čtení a kontrola vstupních dat,
- sumace položek,
- tisk stránky sestavy.

2) Textové hledisko: do segmentu se začlení příkazy, které budou jako celek použity v nějaké textové manipulaci. Např. to může být

- skupina příkazů převzatých z jiného programu (formou samostatného segmentu),
- skupina příkazů, které se v programu opakují (stačí je tedy pořídit pouze jednou),
- skupina příkazů, které se nahradí při odvozování varianty programu,
- skupina příkazů, které bude možno dočasně vynechat při ladění programu (pro úsporu paměti a kompilačního času),
- skupina příkazů, které bude možno umístit na 1 až 2 tiskové strany, aby je bylo možno přehlédnout jedním pohledem a spolehlivě verifikovat.

Realizace metody pomocí makroprocesoru PL/I umožňuje provádět parametrizaci segmentů ve smyslu náhrady textu. Prakticky se této možnosti využívá v případě, že segment obsahující explicitní deklaraci (např. návěští) je použit opakovaně v úrovni jednoho bloku, aby se tak vyloučila nežádoucí duplikace.

2. Realizace metody

V předchozím paragrafu jsme zavedli pojem preprogramu. Ve fyzické formě obsahuje preprogram tyto součásti:

- segmenty, tj. úseky zdrojového textu PL/I včetně makroprocesorových příkazů.
- titulní řádky segmentů ve tvaru "jméno segmentu - poznámka". Jméno segmentu musí začínat znakem S v 1.slabice věty.
- komentářové řádky ve tvaru "C text" v libovolném počtu a libovolném místě. Symbol C je umístěn v 1.slabice věty.

Nyní popíšeme způsob zpracování preprogramu. Jsou k dispozici tři na sebe navazující funkce.

1) Edice preprogramu, při níž se provádí výběr, případně úprava vět pro zpracování. Rozlišují se dva způsoby:

a) distribuční edice, při které se vybrané věty rozdělí po segmentech a založí pod svými jmény jako členy pomocné knihovny. Z preprogramu se převezmou buď všechny segmenty (standard) nebo pouze vyjmenované segmenty, nebo se vynechají vyjmenované segmenty. Při přebírce vět lze vynechat - pro účely ladění - komentářové řádky nebo i všechny příkazy PL/I (vznikne prázdný segment). Titulní a komentářové řádky se při edici upravují na poznámky podle syntaxe PL/I a navíc se připojují poznámky s označením konce segmentů. Původní preprogram zůstává v knihovně beze změny.

b) kopírovací edice, při které je výsledkem nový preprogram, v němž se provedou požadované úpravy (titulní a komentářové řádky zůstávají v původním tvaru). Navíc oproti distribuční edici lze provádět i výběr vět podle intervalů pořadových čísel.

Oba způsoby edice preprogramu se uskutečňují speciálním programem PREEDIT1. V distribuční edici se k tvorbě pomocné knihovny využívá ještě služebního programu IEBUPDTE. Distribuční edice je základem pro generaci programu pomocí makroprocesoru; kopírovací edice má služební účel, jak bude uvedeno dále.

2) Makrogenerace a kompilace. Vstupním textem je hlavní segment, tj. člen SØ z pomocné knihovny, další členy z pomocné knihovny a samostatné segmenty. Výsledkem je zdrojový text, který je vstupem do kompilace.

3) Edice výpisu kompilátoru. Účelem této funkce je upravit výpis kompilátoru tak, aby odkazy na řádky zdrojového (generovaného) textu a čísla zdrojových příkazů byly převeditelné do segmentů a vět preprogramu. Děje se to pomocí speciálního programu POSTEDT1.

Při praktickém použití je zpracování preprogramu řízeno prostřednictvím katalogizovaných procedur řídícího jazyka operačního systému. Tyto procedury jsou zařazeny do širšího systému procedur a knihoven, tzv. systému pro racionalizaci programování (SRP). Podrobnosti viz v [1],[2]; zde jen stručně z hlediska našeho tématu:

V systému SRP jsou zřízeny knihovny Un.SOURCE, Un.OBJECT a Un.LOAD (n=1,2,...) pro zdrojové texty, cílové a zaváděcí moduly, dále knihovny SRP1.SP1.SOURCE a SRP1.SP1.LOAD pro standardní programy, knihovna standardních segmentů SRP1.SS1.SOURCE a knihovna katalogizovaných procedur SRP1.PROCLIB. V knihovnách Un.SOURCE mohou být ukládány zdrojové programy, preprogramy i samostatné uživatelské segmenty.

Z hlediska systému SRP jsou preprogramy i samostatné segmenty zdrojovými texty. Lze na ně aplikovat všechny procedury určené k modifikacím, kopírování atp. Speciálně pro segmentární programování jsou určeny dvě skupiny procedur:

a) procesní P-procedury, kterými se řídí distribuční edice preprogramu a navazující procesy (makrogenerace, kompilace atd.). Např. v pracovním kroku

```
//    EXEC DOP1CL2,LIB=U1,NAME1=PPA,NAME2=LDA
//P.IN DD *
    S3            PL1=0
EXC S5
    S5A     S5
```

se provedou veškeré operace potřebné k tomu, aby z preprogramu PPA v knihovně U1.SOURCE byl vytvořen zaváděcí modul LDA v knihovně U1.LOAD. Přitom segment S3 se zpracuje jako prázdný, segment S5 bude vyloučen a segment S5A přejmenován na S5.

b) služební P-procedury, kterými se řídí kopírovací edice preprogramu. Těchto procedur se využívá k formálním textovým úpravám preprogramů, k přípravě samostatných segmentů a přípravě dokumentace. Např. v pracovním kroku

```
//    EXEC PEDSRCE,LIB1=U1,LIB2=U3,NAME=PPA
//P.IN DD *
    000360  001450  SEGMX
    S5A             SEGMY
```

se z preprogramu PPA v knihovně U1.SOURCE extrahují dva úseky textu a zřídí se jako samostatné segmenty SEGMX a SEGMY v knihovně U3.SOURCE.

Důležitou součástí koncepce segmentárního programování jsou standardní segmenty soustředěné v knihovně SRP1.SS1.SOURCE. V současné době jsou v této knihovně k dispozici segmenty určené

pro podporu ladění, pro vstup znakových dat a pro tisky metodou RECORD. Zmíníme se trochu podrobněji jen o prvním z nich - segmentu UBGPACK1: Je to soubor makroprocesorových funkcí, pomocí nichž lze trasovat segmentární strukturu programu. Např. na začátku segmentu S4 se uvede

@DOWN1(S4,3,)

a na konci segmentu

@UP1(S4,3,DATA(I,J,K)).

Tyto funkce se v makroprocesorové fázi rozvinou - v závislosti na parametru DEBUG - buď do prázdných příkazů nebo do příkazů, které budou produkovat hlášení o vstupu do segmentu a výstupu ze segmentu s tiskem obsahu proměnných I,J,K, a to nejvýše 3-krát.

3. Aplikace a efektivnost metody

Typické použití metody segmentárního programování postupuje asi v těchto krocích: Programátor navrhne segmentární strukturu programu. Vychází při tom z funkčního rozboru úlohy. Některé segmenty může převzít hotové jako samostatné segmenty (deklarace struktur, typové prvky). Nové segmenty kóduje přímo do preprogramu. Takto připravený preprogram založí do zdrojové knihovny, např. U1.SOURCE a zahájí ladění. Pomocí procedur systému SRP provádí dočasné nebo trvalé úpravy zdrojového textu a řídí výběrové zapojování nebo vyřazování segmentů; může tak volit úspornou strategii ladění po částech. Po odladění programu je vhodné připojit segment s podrobnými dokumentačními údaji.

V metodě segmentárního programování lze dobře využívat techniky ladění známé jako "top-down development". Segmenty na nižší úrovni významnosti se na začátku ladění ponechají dočasně prázdné - uvede se pouze jejich titulní řádek, případně reference na ladicí funkce. Tyto nehotové segmenty se pak postupně doplňují až v průběhu ladění.

Další výhodou je možnost systematické tvorby variant programu; lze postupovat dvěma způsoby:

1) Generovat variantní programy výběrem z jednoho společného preprogramu, který se v této souvislosti jeví jako reservoir segmentů. K tomuto účelu se používá distribuční edice preprogramu.

2) Vytvářet variantní preprogramy tak, že se provede výběr segmentů pomocí kopírovací edice a připojí se nové segmenty.

Závěrem se ještě zmíníme o efektivnosti metody segmentárního programování. Z hlediska programátorského použití je možno říci, že metoda je pojmově jednoduchá a že obsahuje malý počet omezení (např. na rozdíl od metody strukturovaného programování). Účelným použitím metody segmentarizace lze dosáhnout úspory lidské práce a kvalitnějšího softwarového produktu.

Z hlediska nároků na strojový čas je třeba upozornit, že zpracování se prodlužuje v přípravné fázi o čas nutný na pomocné operace a makrogeneraci. Podíl tohoto prodloužení činí v průměru 100% čistého času kompilace se silným kolísáním v závislosti na délce programu (klesá) a počtu segmentů (vzrůstá). V provozní fázi však nevznikají žádné vyšší nároky na strojový čas.

Z hlediska softwarové realizace je metoda velmi efektivní, protože v maximální míře navazuje na existující vybavení - operační systém OS/360, resp. OS/EC, služební programy a kompilátor PL/I(F).

$R = V/N$ koef. rentability

V - výhody

N - náklady

Literatura:

[1] Hrouda, J.: Systém pro racionalizaci programování (SRP). Uživatelská příručka. VÚTECHP, Praha, ZN 3/80.

[2] Hrouda, J.: Systém pro racionalizaci programování. MAA, 1980, č. 12, s. 464-465.