Ing. ZDENĚK TOMKA

LABORATOŘ POČÍTACÍCH STROJŮ, VÚT BRNO

Tř. Obránců míru 21

VYUŽITÍ FORTRANSKÝCH PROCEDUR V JAZYCE COBOL na počítači TESLA

Obsah:

1. Úvod.

Při programování v jazyce Cobol mohou nastat případy, kdy je ve výpočtech třeba uplatnit jiné funkce než základní aritmetické operace (výpočet logaritmu, sinu atd., které se v jazyce Cobol nedají počítat).

Tento problém lze řešit přímým použitím procedur v jazyku Fortran, z kterých jsou při kompilaci vytvořeny moduly v BAR a které jsou sestaveny s moduly vzniklými kompilací cobolovské části, takže vytváří jediný program v HSFT. V cobolovském textu potom vyvoláváme subroutine příkazem CALL. Tímto způsobem se velmi rozšíří použitelnost jazyka Cobol a to hlavně v oblasti problémů, kde ve vědecko-technických výpočtech zpracováváme velké množství dat.

2. Základní principy řešení problému volání fortranských procedur v programu Cobol.

2.1. Sestavení programu

Aby bylo možné sestavení modulů vytvořených z fortranských procedur a modulů vytvořených překladem hlavního cobolovského textu, je třeba kompilovat ještě hlavní fiktivní program v jazyku Fortran, který má tu funkci, že pomocí něj se přisestaví moduly v BAR z knihovny, které umožní další sestavení.
Možné řešení hlavního fikt. programu.

```
c FIKTIVNÍ HLAVNÍ PROGRAM
  COMMON /A3/ M1, M2
  CALL COP
  STOP
  END
```

Při tomto řešení vždy po sestavení programu začíná program na instrukcích hlavního fiktivního programu a to i tehdy, jestliže kompilujeme cobolovský text po kompilaci fortranského fiktivního programu. Startovací adresa je vždy určena k programu v jazyce Fortran. Proto je třeba ve fiktivním hlavním programu vyvolat část programu cobolovskou instrukcí CALL COP, kde COP je název modulu přeloženého z cobolovského textu a odpovídá názvu u zápisu PROGRAM-ID.

Složení příkazových štítků pro sestavení programu.

```
. JOB
. CCBOL
  TEXT COBOL
. EOF
. FORTRAN
TEXT FIKTIVNÍHO PROGRAMU VE FORTRANU
TEXT FORTRANOVSKÉ SUBROUTINE
. EOF
. LOAD NASE, NENT
. EOF
```

2.2. Sdílení paměti pro společné proměnné obou částí programu

Mějme proměnné N1, N2, které se vyskytují jak v cobolovské části programu, kde jim přiřazujeme určitou hodnotu, tak i ve fortranské proceduře, kde jsou buď skutečnými parametry procedury nebo jsou přímo použity v bezparametrové proceduře. Potom tyto proměnné musí být popsány v obou částech programu, tak, aby sdílely stejnou část paměti.

V cobolovském textu jsou proto popsány jako externí proměnné. Proměnné popíšeme v DATA DIVISION takto:

```
LINKAGE SECTION.
01 A3.
02 N1 PIC X (4).
02 N2 PIC X (4).
```

A3 je označení skupiny použitých proměnných. Toto řešení zajišťuje dělitelnost čtyřmi počátečních adres binárních údajů v pevné desetinné čárce. Popis společných proměnných musí mít vždy tento obraz. PICTURE X (4). Tento popis proměnné odpovídá binárnímu zobrazení čísla s pevnou řádovou čárkou fortranských proměnných.

V oddíle ENVIRONMENT DIVISION stanovíme, že proměnné N1, N2 podřízené skupinovému údaji A3 jsou vnější takto:

LINKAGE SECTION.
EXTERNAL. A3.

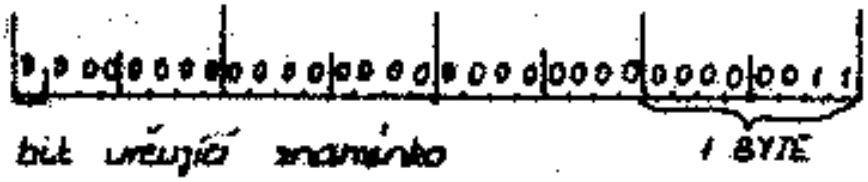
Ve fortranské proceduře musí být společné proměnné popsány jako proměnné označené identifikátorem v části COMMON
COMMON /A3/ N1, N2.

Proměnné N1, N2 nelze v popisu COMMON uvést v neoznačeném bloku COMMON N1, N2, poněvadž adresy proměnných vymezené cobolovským popisem a fortranským popisem by byly posunuty o 4 byty.

2.3. Převod čísla ze znakové formy do binárního zobrazení v pevné desetinné čárce.

Aby bylo možno ve fortranské subroutině počítat, je třeba, aby v paměti byly hodnoty společných proměnných N1, N2 zobrazeny v binární formě s pevnou řádovou čárkou.

Příklad binárního zobrazení:
binární zobrazení čísla 3 v pevné des. čárce



Mějme za úkol přečíst v cobolovské části prógramu 2 čísla a provést s nimi výpočet ve fortranské proceduře. V Cobolu jsou čtena čísla z děrných štítků a ukládána do paměti dle popisu souboru ve znakové formě.

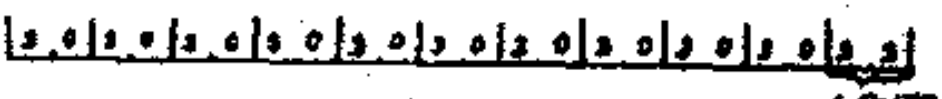
Popis souboru pro čtení N5:

```
1 N1.
2 Q1 PIC S9 (11).
2 Q2 PIC S9 (11).
```

Obrazem S 9(11) můžeme zobrazit číslo, které má 11 řádů, tedy dostatečně velké pro běžné použití.

Příklad zobrazení čísla ve znakové formě:

zobrazení čísla 3 ve znakové formě



11 BYTE

Pro zajištění převodu čísla ve znakové formě do binárního zobrazení použijeme podprogramu operačního systému /CFDEB.
Tento podprogram převádí čísla v dekadicky zhuštěné formě na číslo v binární formě v pevné řádové čárce.
Je třeba zajistit tedy nejdříve v cobolovské části převod čísla ze znakové formy do dekadicky zhuštěné formy takto:

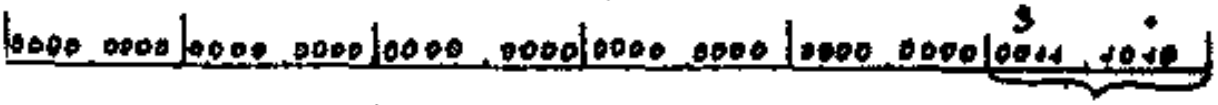
Ve WORKING - STORAGE SECTION popíšeme nové proměnné v dek. zhuštěné formě.

```
77 P1 PIC S9 (11) COMP.
77 P2 PIC S9 (11) COMP.
```

Převod provedeme v části PROCEDURE DIVISION
příkazy MOVE Q1 TO P1.
MOVE Q2 TO P2.

Příklad zobrazení čísla v dekadicky zhuštěné formě.

Zobrazení čísla 3 v dek. zhuštěné formě:



6 BYTE

1 číslice je zobrazena na 1/2 bytu

Dále provedeme převod P1, P2 v dekadicky zhuštěné formě na N1, N2, které již budou v binární formě příkazy

```
CALL "/SFDEB " USING P1, N1.
CALL "/CFDEB " USING P2, N2.
```

a máme tak připravené podmínky pro spuštění fortr. procedury.

2.4. Převod čísla z binární formy v pevné řádové čárce do znakové formy.

Postup při tomto řešení je opačný než jak byl popsán v předcházejícím odstavci.
Pro převod z binární formy do dekadicky zhuštěné formy používáme podprogram operačního systému /CFBID takto:

```
CALL"/CFBID " USING P2, N2.
```

Proměnná P2 je popsána v dekadicky zhuštěné formě.
Proměnná N2 je popsána v binární formě a pevnou řádovou čárkou.
Převod do znakové formy pak provedeme v Cobolovské části takto:

```
MOVE P2 TO P1.
```

P1 je popsán ve znakové formě.

2.5. Vlastní volání procedury a předávání parametrů

Subroutinu budeme volat tímto Cobolovským příkazem

```
CALL " NAZEVS " [USING N1, N2 ...]
```

" NAZEVS " je jméno volané subroutiny
N1, N2 jsou skutečné parametry.
Předávání skutečných parametrů subroutině :

Subroutině s parametry předáváme skutečné parametry takto:
do RN1 musí být uložen počet parametrů

a) do dvouparametrové procedury
   v RN2 musí být adresa 1. parametru
   v RN3 musí být adresa 2. parametru

b) pro víceparametrovou proceduru
   v RN2 musí být uložena adresa tabulky adres parametrů,
   kde na každou adresu jsou reservovány 2 byty.

I. Možnost USING instrukce CALL nám umožňuje předávat skutečné parametry fortranské proceduře takto:

a) při použití procedury do dvou parametrů ukládá adresu
   1. parametru do RN2.
   adresu 2. parametru do RN3.

b) při použití procedury s více než dvěma parametry vytvoří tabulku adres parametrů a adresu tabulky uloží do RN2.

   V tabulce jsou rezervovány 4 byty pro 1. adresu.

Srovnáním s požadavky subroutiny vidíme, že indikací USING v instrukci CALL můžeme bez komplikací předávat parametry proceduře s maximálně dvěma parametry. Při volání více parametrové procedury tabulka vytvořená indikací USING neodpovídá tabulce, kterou požaduje subroutina. Adresy jsou uloženy na 4 Byte místo na 2 Byte. Obejití této skutečnosti při řešení použití víceparametrových procedur je ukázáno v další části popisu.

II. Z důvodu tohoto rozporu byl pro předávání parametrů vytvořen zvláštní modul v BAR/ CPPAPn, který je součástí operačního systému.
Tento modul dle expres informací č. 28 by měl předávat skutečné parametry 1 až 6 parametrové subroutině.

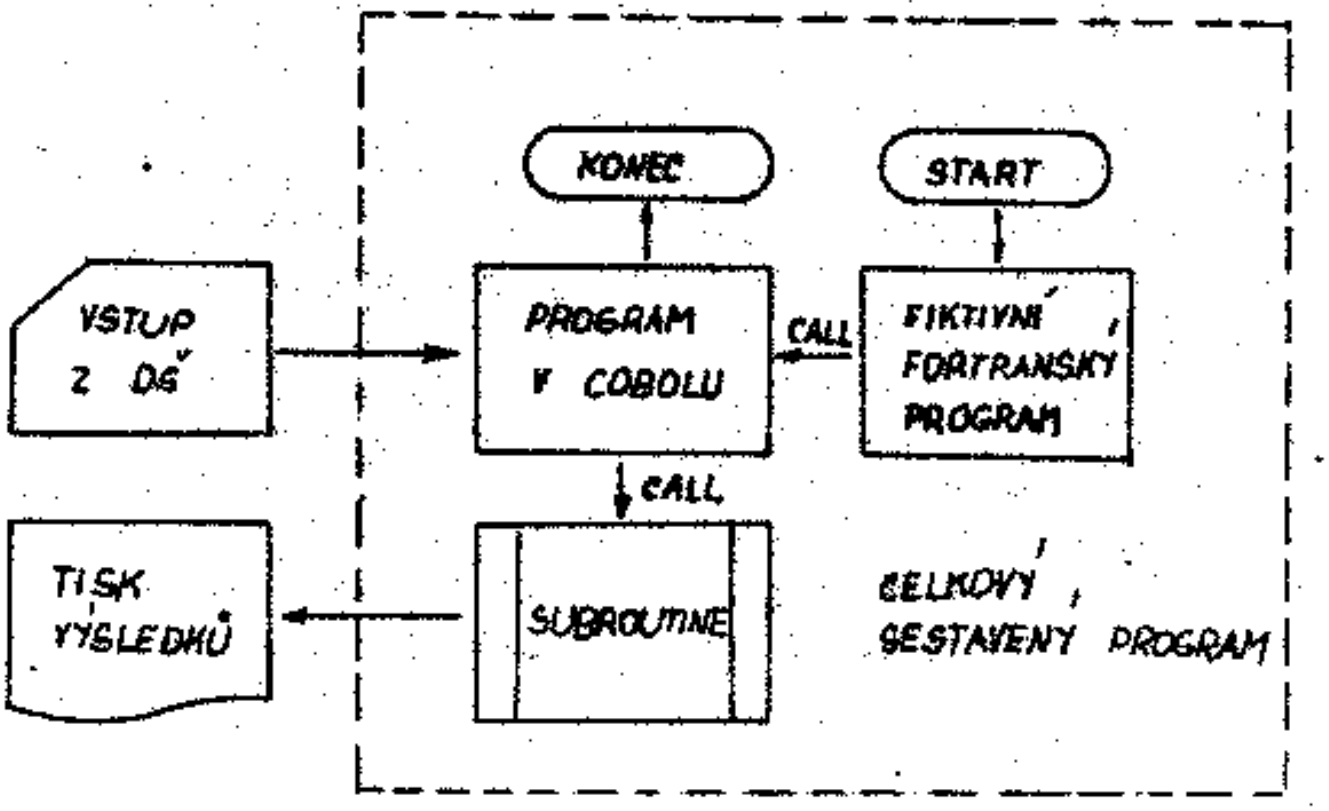
Ve skutečnosti však nepředával při odzkoušení parametry více než pro 2 parametrovou proceduru.
Proto jsme vytvořili program v APS, z kterého byl vytvořen modul v BAR pod názvem FOC, jež může doplnit knihovnu systému. Tento modul FOC umožňuje pracovat se subroutine s prakticky neomezeným počtem parametrů (50).

3. Použití bezparametrové procedury.

Použití bezparametrové procedury si ukážeme na příkladech.

Příklad
Zadání:
Čtěte dvojice čísel N1, N2 z DS a pro každou dvojici proveďte výpočet dle vzorce P3 = $N1^2 - N2^2$ a vytiskněte na tiskárnu výsledek P3. DS čtete v jazyku Cobol. Výpočet a tisk výsledků proveďte v Subroutině.

schema:



Řešení:

- popis souboru pro čtení DŠ ve FILE SECTION.

```
R1.
2 Q1 PIC S9 (11).
2 Q2 PIC S9 (11).
```

- Popis proměnných ve zhuštěném dekadickém tvaru ve WORKING - STORAGE SECTION.

```
EXTERNAL. A3.
```

v DATA DIVISION je třeba popsat proměnné takto:

```
LINKAGE SECTION.

1 A3.
  2 N1 PIC X(4).
  2 N2 PIC X(4).
```

ve FORTRANU:

```
COMMON /A3/ N1, N2
```

- Převod ze znakové do dekadicky zhuštěné formy:

```
MOVE Q1 TO P1.
MOVE Q2 TO P2.
```

- Převod z dekadicky zhuštěné formy do binární formy v pevné řádové čárce:

```
CALL "/CFDEB " USING P1, N1.
CALL "/CFDEB " USING P2, N2.
```

- Uložení nuly do RN1 (počet parametru je 0) se provede instrukcí v APS:

```
ENTER COBOL.
```

- volání vlastní procedury:

```
CALL "ODM ".
```

- V subroutině se vypočte hodnota P 3, vytiskne se a program se vrací na další cobolovské čtení

```
P3 = SQRT (FLOAT (N1 * * 2) - FLOAT (N2* * 2))
WRITE (3,10) P3
```

Příklad č. 2.

Zadání: Je shodno s předchozím až na to, že výsledná hodnota N3 se převede zpět do tvaru znakového a vytiskne se cobolovsky.

Řešení:

- Cobolovský popis tiskového recordu:

```
1 RT1.
2 Q3 PIC --------- 9,99.
```

- Popis proměnných ve zhušt. dek. tvaru ve WORKING - STORAGE SECTION.

```
77 P1 PIC S9(11) comp.
77 P2  -  "   -
77 P3  -  "   -
```

- Popis proměnné pro uložení výsledku ve znakové formě.

```
77 P4 PIC S9 (11).
```

- Popis společných proměnných v Cobolu:

```
01 A3.
   02 N1 PIC X(4).
   02 N2 PIC X(4).
   02 N3 PIC X(4).
```

ve fortranu:

```
COMMON /A3/, N1, N2, N3
```

Protože instrukce před vstupem do subroutiny byly již ukázány v předcházejícím příkladě, ukážeme si jen převod hodnoty výsledné proměnné zpět do znakové formy.

- výpočet v proceduře

```
P3 = SQRT (FLOAT (N1) - FLOAT (N2))
N3 = P3, 100
```

v poslední instrukci jednak převádíme hodnotu typu real P3 na hodnotu typu integer N3 a násobíme 100, abychom zachovali hodnotu čísla na 2 desetinná čísla.

- provedeme převod z binárního čísla v pevné řádové čárce na dekadicky zhuštěný tvar :

  CALL */CFBID * USING N3, P3

- Převedeme dekád. zhuštěný tvar proměnné P3 do znakové formy.

  MOVE P3 TO P4.

- Přesun tvarů výsledků ve znakové formě do tiskového recordu MOVE P4 TO Q3.

4. Výpočet se zápornými čísly a s desetinnými čísly.

Záporná čísla:
Při výpočtu fortranské procedury volané cobolskými programem můžeme pracovat se zápornými čísly. Obraz popisu rekordu na DŠ musí obsahovat indikaci S. (2 Q1 PIC S9(11).)
Taktéž popis proměnné ve zhuštěné formě musí obsahovat indikaci S. (02 PIC S9 (10) COMP.)

Na DŠ bude zakódováno záporné číslo dle zvyklostí TESLA COBOL tak, že poslední číslice záporného čísla je nahrazena odpovídajícím písmenem.

0 . . . . . P
1 . . . . . Q atd.

Desetinná čísla:
Při výpočtu fortranské procedury volané cobolovskými programem nedáváme pracovat přímo s desetinnými čísly. Převod čísla ze znakové formy do fortranské procedury lze provést jen do binárního tvaru čísla v pevné řádové čárce pomocí podprogramu /CFDEB. V systému není vypracován podprogram, který by převáděl číslo vyjádřené dekadicky zhuštěnou formou (případně znakovou formou)

do binárního tvaru v pohyblivé desetinné čárce. Z těchto důvodů musíme vstupovat do fortranské procedury jen s čísly typu integer a na výstupu zajistíme správnost desetinné čárky tím, že výsledek podělíme mocninami čísla 10, tak jak předpokládáme výsledný řád čísla.

Použití v popisu obrazu proměnné v Cobolovském programu indikace V jako pomyslné desetinné čárky nemá smysl.
např. Q1 PIC S9(10) V9.
Výsledek je stejný jako kdybychom napsali
Q1 PIC S9(11).

3. Volání subroutine se 2 parametry.

Použití volání 2 parametrové procedury si ukážeme na příkladech.
Nejprve budeme řešit zadání příkladu č.1, uvedeného dříve, kde však proměnné budou skutečnými parametry subroutine.
Je třeba zajistit předání skutečných parametrů instrukcí v APS

```
ENTER OBM - CODE.
LA I 1,2
ENTER COBOL.
```

a instrukcí v Cobolu.

```
CALL "ODM " USING N1, N2.
```

N1, N2 jsou skutečné parametry subroutiny. Fortranská procedura je potom popsána takto:

```
SUBROUTINE ODM (K1, K2)
```

kde K1, K2 jsou formální parametry.
Zadání je možno řešit také tak, že skutečné parametry nejsou popsány v části Cobol jako externí, ale jsou popsány přímo ve WORKING -STORAGE SECTION :

```
77 N1 PIC X(8).
77 N2 PIC X(4).
```

Toto řešení je možné, protože přenos parametrů je uskutečněn pomocí numerických registrů RN1, RN2, RN3.
Není zde však zajištěna dělitelnost čtyřmi počátečních a binárních údajů skutečných parametrů a tudíž není možné druhý způsob doporučit.

- Na příkladě č. 3 bude řešení výpočtu pomocí 2 parametrové procedury, kde jeden parametr je vstupní, druhý výstupní.

Příklad č. 3

Zadání:

Máme číst hodnotu N1 z DŠ, v části Cobolovského programu provést výpočet dle vzorce P5 = $\sqrt{2 * N1}$ a Cobolovsky vytisknout výsledek. Tento výpočet opakujeme pro více hodnot N1.

Řešení:

- popíšeme proměnnou ve znakové formě pro uložení hodnoty z DŠ

```
1 R1.
  2 Q1 PIC S9(11).
```

- popíšeme editovanou alfanumerickou proměnnou pro tiskový record

```
1 RTI.
  2 Q3 PIC - - - - - - - - - - 9,99.
```

- popíšeme pro vstupní i výstupní parametr pomocnou proměnnou v dek. zhuštěném tvaru

```
77 P1 PIC S9(11) COMP.
77 P3 PIC S9(11) COMP.
```

- popíšeme pomocnou proměnnou pro výpočtenou hodnotu ve znakové formě

```
77 P4 PIC S9(11).,
```

- vymezíme oblast paměti pro skutečné parametry v Cobolu:

```
ENVIRONMENT DIVISION.
LINKAGE SECTION.
EXTERNAL A3.
DATA DIVISION.
LINKAGE SECTION.
```

```
1 A3.
  2 N1 PIC X(4).
  2 N3 PIC X(4).
```

ve fortranu:

```
COMMON /A3/ N1, N3
```

- provedeme převod ze znakové formy vstupního parametru do dekadicky zhuštěné formy

```
MOVE Q1 TO P1.
```

- převedeme vstupní parametr z dek. zhuštěné formy do binárního zobrazení v pevné řádové čárce

```
CALL "/CFDEB " USING P1, N1.
```

- provedeme předání skutečných parametrů

I. způsob :

```
ENTER OWN-CODE.
. LA I 1,2
  ENTER COBOL.
  CALL "ODM " USING N1,N3.
```

II. způsob :

```
ENTER OWN - CODE CFPAR2, USING N1, N3
CALL " ODM ".
```

- provedeme výpočet o subroutině

```
P5 = SQRT (FLOAT ( 2  K1))
K3 = P5  100
```

K1, K3 jsou formální parametry

- převedeme skutečný výstupní parametr z binárního tvaru do dekadicky zhuštěného tvaru

```
CALL "/CFBID " USING N3, P3.
```

- převedeme výstupní parametr z dekadicky zhuštěného tvaru do znakové formy

```
MOVE P3 TO P4.
```

- přesuneme výstupní parametr do tiskového recordu a vytiskneme

```
MOVE P4 TO Q3.
WRITE RTI.
```

6. Volání víceparametrové procedury

Při využití víceparametrové fortranské procedury je hlavní problém předávat této proceduře skutečné parametry. Modulem /CFPAR by mělo být umožněno předání 6 parametrů. Při ověření však bylo možno pomocí tohoto modulu pracovat jen s maximálně dvojparametrovou procedurou. Předávat prakticky neomezený počet parametrů je však možno tím, že voláme námi vytvořený podprogram FOC.

Volání víceparametrové subroutine s využitím modulu FOC

FOC je název modulu v BAR, vytvořeného přeložením programu v APS, kterým zabezpečujeme předání prakticky neomezeného počtu skutečných parametrů fortranské progeduře. Modul FOC je možno uložit na systémovou MP.

Postup při výpočtu, ve kterém voláme 3-parametrovou proceduru a předání parametrů uskutečňujeme pomocí modulu FOC, uloženého na systémové MP si ukážeme na příkladu č.6.

Příklad č. 6.

Zadání: Přečtete hodnoty N1, N2, N3 z děrných štítků v jazyce Cobol, ze kterých vypočtete a vytiskněte v subroutině hodnoty P5 dle vztahu P5=N1-N2+N3.

Řešení:

- popíšeme vstupní record DS

```
1 R1.
  2 Q1 PIC S9(11).
  2 Q2 PIC S9(11).
  2 Q3 PIC S9(11).
```

- popíšeme vstupní proměnné v dekadicky zhuštěném tvaru

```
77 P1 PIC S9(11) COMP.
77 P2 PIC S9(11) COMP.
77 P3 PIC S9(11) COMP.
```

- vymezíme společnou část paměti pro vstupní proměnné v binárním tvaru

v Cobolu:

```
LINKAGE SECTION.
1 A3.
  2 N1 PIC X(4).
  2 N2 PIC X(4).
  2 N3 PIC X(4).
```

ve fortranu:

```
COMMON/A3/N1,N2,N3
```

- převod vstupních proměnných ze znakové formy do dekadicky zhuštěného tvaru

```
MOVE Q1 TO P1.
MOVE Q2 TO P2.
MOVE Q3 TO P3.
```

- převod vstupních proměnných z dekad. zhuštěného tvaru do binární formy

```
CALL "/CFDEB" USING P1,N1.
CALL "/CFDEB" USING P2,N2.
CALL "/CFDEB" USING P3,N3.
```

- předání parametrů proceduře

```
ENTER OWN-CODE.
.   LAI   1,3
ENTER COBOL.
```

Tímto zápisem uložíme do RN1 počet parametrů procedury, v našem případě 3.

```
CALL "FOC" USING N1,N2,N3.
```

Pomocí modulu v BAR FOC vytvoříme potřebnou tabulku adres skutečných parametrů procedury a její adresu uložíme v RN2.

- volání subroutine

```
CALL "ODM".
```

- popis subroutine

```
SUBROUTINE ODM(K1,K2,K3)
```

K1,K2,K3 jsou formální parametry procedury.

- výpočet a tisk v subroutine

```
P5=K1-K2-K3
WRITE (3,10)P5
```

7. Volání procedury, kde parametrem je pole.

Při volání fortranovských procedur může být parametrem procedury pole. Využitím pole jako parametru procedury můžeme zmenšit počet potřebných parametrů v proceduře a vyhnout se tak komplikacím, které se vyskytují u více parametrové procedury.
Použití si ukážeme na řešení příkladu č. 5.

Příklad č. 5

Zadání: Přečtěte hodnoty Q1, Q2, Q3 z DS, vytvořte z těchto údajů pole M1, které bude skutečným parametrem fortranské procedury, ve které vypočteme a vytiskneme hodnotu

P3 = M1(1) - M1(2) + M1(3).

Řešení:

- popíšeme vstupní record DS

```
1 M1.
  2 Q1 PIC S9(11).
  2 Q2 PIC S9(11).
  2 Q3 PIC S9(11).
```

- popíšeme vstupní proměnné v dekadicky zhuštěném tvaru

```
77 P1 PIC S9(11) COMP.
77 P2 PIC S9(11) COMP.
77 P3 PIC S9(11) COMP.
```

- vyrovnáme pro pole M1, které je skutečným parametrem část paměti takto

v Cobolu:

```
LINKAGE SECTION.
1 A3.
  2 M1 PIC X(4) OCCURS 3 TIMES.
```

ve Fortranu:

```
COMMON /A3/ M1 (3)
```

- procedura bude popsána takto:

```
SUBROUTINE ODM (K1)
DIMENSION K1(3)
```

- provedeme převod ze vstupních hodnot ze znakové formy do dekadicky zhuštěné formy

```
MOVE Q1 TO P1
```

- provedeme převod vstupních hodnot z dekadicky zhuštěné formy do pole N1, kde budou hodnoty v binárním tvaru s pevnou desetinnou čárkou

```
CALL "/CFDEB " USING P1, N1 (1).
CALL "/CFDEB " USING P2, N1 (2).
```

- provedeme předání skutečného parametru

```
ENTER OWN - CODE.
. LAI  1,1
ENTER COBOL.
CALL " ODM " USING A3.
```

- provedeme výpočet v subroutine a vytiskneme vypočtenou hodnotu P5 = K1(1) - K1(2) * K1(3)

```
WRITE (3, 10) P5
```

K1 je pole, které je formálním parametrem.