Procedury v jazycích ILE

Vladimír Župka, 2021

Úvod

V systému IBM i před zavedením programového modelu ILE existovaly dva programové modely: OPM (Original Program Model) a EPM (Extended Program Model). Programový model OPM se týkal jazyků RPG, Cobol a CL, model EPM se týkal jazyků Fortran, Pascal a C. Zatímco jazyky Fortran a Pascal ze systému IBM i prakticky zmizely, jazyk C zůstal a k němu přibyl jazyk C++.

Programový model OPM je charakteristický tím, že každý program je jednolitý celek (nezná moduly) a ke spojení s jinými programy používá *dynamické* volání (dynamic call). Dynamické volání programů klade velké nároky na výpočetní čas. Program je totiž objekt, a jako takový jej systém nejprve musí nalézt v knihovně a musí zkontrolovat oprávnění uživatele k jeho spuštění. Pak jej teprve zavede do paměti a inicializuje. Dynamické volání ovšem funguje i v modelu ILE.

Model ILE byl zaveden zejména kvůli jazykům C a C++, aby je zefektivnil a umožnil spojování modulů. Model ILE byl pak zaveden i do jazyků RPG, Cobol a CL s různým stupněm využití jeho možností.

Programy v jazycích modelu ILE, tedy C, C++, RPG, Cobol a CL, se skládají z *modulů*. Moduly se skládají z *procedur*. Hlavnímu programu se říká hlavní procedura a ostatním procedurám zapsaným v témže modulu se říká *podprocedura* (subprocedure). Procedura může při vyvolání přijímat vstupní parametry a při návratu vracet výstupní parametry a případnou hodnotu. Vrací-li hodnotu, říká se jí *funkce*[1].

V modelu ILE existuje tzv. statické volání (static call). To znamená, že v době, kdy se moduly spojují do programu, jsou známy adresy procedur a ty se dosadí do volajících příkazů. Statické volání tak probíhá mnohem rychleji než dynamické, protože odpadají výše zmíněné zdlouhavé operace.

Terminologie se v jednotlivých jazycích trochu liší:

- V jazyku C a C++ se všechny procedury (i hlavní) nazývají *funkce*.
- Jazyk RPG zná všechny výše zmíněné pojmy, funkce je podprocedura, která vrací hodnotu, hlavní procedura je hlavní program.
- Jazyk Cobol nezná pojem “subprocedure”, ale “ILE procedure”, přičemž pojem “procedure” znamená program (většinou hlavní proceduru).
- Jazyk CL zná jen pojem procedura; nezná pojem “subprocedure”, protože CL program může mít jen jedinou (hlavní) proceduru.

[1] Procedura nemusí mít parametry, ani nemusí vracet hodnotu.

Za příklad ve všech jazycích si zvolíme výpočet obsahu obecného trojúhelníka, podle Heronova vzorce:

```
plocha = sqrt(s(s-a)(s-b)(s-c)),
```

kde

```
s = (a + b + c) / 2.
```

Trojúhelník je určen třemi stranami *a*, *b*, *c*, které musí splňovat podmínky

```
a < b + c,
b < a + c,
c < a + b
```

tedy že každá strana musí být menší než součet obou ostatních.

Procedury v jazyku C

V jazyku C a C++ se všechny procedury nazývají funkce. Původně každá funkce musela vracet hodnotu, což mělo odpovídat matematickému pojmu funkce. Časem však z definice jazyka tato podmínka zmizela a funkce již hodnotu vracet nemusí (v takovém případě však musí být zadáno slovo *void* jako typ "prázdné vracené hodnoty").

Funkce *obsah()* bude mít tři parametry (v jazyku C zvané argumenty): strany trojúhelníka. Zkontroluje strany, zda splňují podmínky, a když budou v pořádku, spočítá obsah. Vypočtený obsah vrátí jako hodnotu. Všechny hodnoty budou uvedeny jako dekadická čísla v pakovaném tvaru na 15 číslic, z toho 5 desetinných míst. V jazyku ILE/C to bude typ *decimal(15, 5)*.[2]

Nejprve ukážeme příklad, kdy kombinujeme funkci obsah() i main() do jednoho zdrojového členu, tedy modulu. Ten se kompiluje příkazem CrtBndC, tedy volbou 14 v PDM, kdy vzniká provizorní modul v knihovně QTEMP, který se "spojí" do výsledného programu a pak se smaže.

Modul s funkcemi obsah() a main()

```
#include <decimal.h>
#include <math.h>
#include <stdio.h>

// prototyp funkce obsah
decimal(15, 5) obsah (decimal(15, 5), decimal(15, 5), decimal(15, 5));

// hlavní funkce main()
int main ()
{
   decimal (15, 5) plocha = obsah (3.0D, 4.0D, 5.0D);  // volání funkce obsah
   printf ("plocha = %D(15,5)", plocha);
}

// funkce obsah()
decimal(15, 5) obsah (decimal(15, 5) a, decimal(15, 5) b, decimal(15, 5) c)
{
   decimal(15, 5) s;
   if (a >= b + c || b >= a + c || c >= a + b)
   {
      return -1.0D;    // chybně určený trojúhelník
   }
   s = (a + b + c) / 2;
   return sqrt(s * (s - a) * (s - b) * (s - c));
}
```

Konstanta typu decimal se označuje velkým písmenem D těsně za číslem, např. 3.0D.[3]

Z funkce `obsah()` se volá funkce druhé odmocniny `sqrt()` (standardní funkce jazyka C). V jejím průběhu se převádějí čísla z typu decimal na double a zpět, protože prototyp funkce je

```
double sqrt(double x);
```

---

[2] Typ *decimal* je rozšířením jazyka C v Systému i. V ostatních systémech neexistuje. Tam se místo toho používá typů *float* nebo *double*. My používáme typ decimal, abychom umožnili volání funkce s konstantními argumenty i z jazyka CL.

3 V jazyku C++ ale takový typ konstanty není. Tam by se napsalo `__D("3.0")`.

V příkladu je funkce `main()` zapsána jako první, funkce `obsah()` jako druhá. Proto je nutné předem zapsat prototyp funkce `obsah()`, aby se dala vyvolat ve funkci `main()`. Kdybychom obrátili pořadí funkcí, mohli bychom prototyp vynechat.

Program po spuštění vypíše výsledek

```
plocha = 6.00000.
```

Trojúhelník o stranách 3, 4, 5 je pravoúhlý, proto dvojnásobek jeho obsahu je dán součinem jeho odvěsen: 3 x 4 = 12.

Jiný způsob, jak sestavit program, je osamostatnit funkce `obsah()` a `main()` do samostatných zdrojových členů zkompilovat je do samostatných modulů příkazem `CRTCMOD` a spojit je příkazem `CRTPGM`.

Modul OBSAH_PRC s funkcí `obsah()`

```
#include <decimal.h>
#include <math.h>

decimal(15, 5) obsah (decimal(15, 5) a, decimal(15, 5) b, decimal(15, 5) c)
{
   decimal(15, 5) s;

   if (a >= b + c || b >= a + c || c >= a + b)
   {
      return -1.0D;    // chybně určený trojúhelník
   }
   s = (a + b + c) / 2;
   return sqrt(s * (s - a) * (s - b) * (s - c));
}
```

Modul OBSAH_CALL s funkcí `main()`

Funkci `obsah()` voláme z hlavní funkce `main()`, proto musíme uvést prototyp.

```
#include <stdio.h>
#include <decimal.h>

// prototyp funkce obsah
decimal(15, 5) obsah (decimal(15, 5), decimal(15, 5), decimal(15, 5));

// hlavní funkce main()
int main ()
{
   decimal (15, 5) plocha = obsah (3.0D, 4.0D, 5.0D);  // volání funkce obsah
   printf ("plocha = %D(15,5)", plocha);
}
```

Oba moduly kompilujeme příkazem `CRTCMOD`, čili volbou 15 v PDM. Vzniknou tak moduly, které spojíme příkazem

```
CRTPGM PGM(*CURLIB/OBSAH_CALL) MODULE(OBSAH_CALL OBSAH_PRC)
```

do programu `OBSAH_CALL` v běžné knihovně. Po jeho spuštění se vypíše výsledek

```
plocha = 6.00000.
```

Procedury v jazyku RPG

Hlavnímu programu v RPG se říká hlavní procedura a ostatním procedurám zapsaným v témže modulu se říká *podprocedura* (subprocedure). Podprocedura, která vrací hodnotu se nazývá *funkce*.

Nejprve ukážeme příklad, kdy kombinujeme hlavní proceduru i funkci obsah do jednoho zdrojového členu, tedy modulu. Ten se kompiluje příkazem `CRTBNDRPG`, tedy volbou 14 v PDM, kdy vzniká provizorní modul v knihovně `QTEMP`, který se "spojí" do výsledného programu a pak se smaže.

Za příklad si opět zvolíme výpočet obsahu obecného trojúhelníka podle Heronova vzorce.

Modul s hlavní procedurou a podprocedurou obsah

```
CTL-OPT MAIN (OBSAH_B);  // určuje lineární hlavní proceduru
CTL-OPT DFTACTGRP(*NO);  // nebo např. ACTGRP(*NEW)
CTL-OPT BndDir('QC2LE'); // potřebné pro funkce z jazyka C

// lineární hlavní procedura - hlavní program
DCL-PROC OBSAH_B;  // hlavní procedura má podobu podprocedury
    dsply OBSAH ( 3: 4: 5 );  // statické volání podprocedury bez prototypu
END-PROC;

// podprocedura
DCL-PROC  Obsah;  // převádí se do velkých písmen
    // rozhraní podprocedury
    DCL-PI *N   packed(15: 5);        // vracená hodnota
        a    packed(15: 5) value;     // tři parametry
        b    packed(15: 5) value;
        c    packed(15: 5) value;
    END-PI;
    // prototyp standardní funkce sqrt z jazyka C
    dcl-pr Sqrt float(8)  extproc('sqrt');
        *n   float(8)   value;
    end-pr;
    dcl-s  s  packed(15: 5);
    if a >= b + c or b >= a + c or c >= a + b;
       return -1;
    endif;
    s = (a + b + c) / 2;      // poloviční součet stran
    return Sqrt(s * (s - a) * (s - b) * (s - c)); // statické volání funkce Sqrt
END-PROC OBSAH;
```

Příkaz `CTL-OPT` obsahuje dvě důležité specifikace. Parametr `DFTACTGRP(*NO)` znamená, že program poběží v pojmenované aktivační skupině se standardním jménem QILE. Parametr `BNDDIR('QC2LE')` umožňuje přístup ke standardním a unixovým funkcím jazyků C a C++ (zde k funkci druhé odmocniny - `sqrt()`).

Hlavní procedura musí být zapsána jako první, podprocedura obsah jako druhá. V hlavní proceduře musí být zapsán prototyp podprocedury `obsah`.[4]

Popis podprocedury - funkce `obsah()` je uveden příkaem `DCL-PROC` a zakončen příkazem `END-PROC`.

---

4 V jednom modulu může být ovšem zapsáno více podprocedur.

Všimněme si, že parametry funkce v prototypu (PR) i v popisu rozhraní (PI – procedure interface) jsou označeny slovem VALUE, což znamená, že proceduře se předává *hodnota* parametru. Bez tohoto označení by se předávala *adresa* parametru.

V prototypu je také uveden parametr EXTPROC, který zadává jméno funkce v doslovném tvaru s malými písmeny.

Ve funkci druhé odmocniny `sqrt()` proběhne konverze čísel z pakovaného dekadického typu na typ pohyblivé řádové čárky (typ `float(8)` totožný s typem `double` z jazyka C) a zpět. Prototypu funkce v jazyku C

```
double sqrt(double x);
```

odpovídá prototyp v jazyku RPG

```
    dcl-pr Sqrt float(8)  extproc('sqrt');
        *n   float(8)   value;
    end-pr;
```

Program po spuštění vypíše výsledek

```
DSPLY  plocha = 6.00000.
```

Trojúhelník o stranách 3, 4, 5 je pravoúhlý, proto dvojnásobek jeho obsahu je dán součinem jeho odvěsen: 3 x 4 = 12.

Jiný způsob, jak sestavit program, je osamostatnit proceduru `obsah()` a hlavní proceduru do samostatných zdrojových členů, zkompilovat je do modulů příkazem CRTRPGMOD a spojit je příkazem CRTPGM.

Modul OBSAH_PRC s procedurou 'obsah'

```
CTL-OPT NOMAIN; // modul nemá hlavní proceduru

// podprocedura (funkce)
DCL-PROC  OBSAH  EXPORT ;  // podrocedura je volána z jiného modulu
    // rozhraní procedury - procedure interface
    DCL-PI *N      packed(15: 5)     // vracená hodnota
                   EXTPROC('obsah'); // malá písmena!
        a    packed(15: 5) value;    // parametry
        b    packed(15: 5) value;
        c    packed(15: 5) value;
    END-PI;
    dcl-s  s  packed(15: 5);  // poloviční součet stran
    if a >= b + c or b >= a + c or c >= a + b;
       return -1;
    endif;
    s = (a + b + c) / 2;      // poloviční součet stran
    return %sqrt(s * (s - a) * (s - b) * (s - c));  // výpočet obsahu
END-PROC obsah;
```

V příkazu CTL-OPT jsme zapsali slovo NOMAIN, aby modul neobsahoval hlavní proceduru, která by zabírala zbytečně mnoho místa. Podprocedura `obsah()` je uvedena příkaem DCL-PROC a zakončen příkazem END-PROC. V úvodním příkazu je slovo EXPORT, které je nutné, aby procedura byla k dispozici jiným modulům (zde modulu OBSAH_CALL). Před zápisem podprocedury musí být uveden její prototyp.

Modul OBSAH_CALL – hlavní procedura

Z hlavní procedury voláme podproceduru `obsah()`, proto musíme uvést její prototyp.

```
// Program bez RPG cyklu
CTL-OPT MAIN(OBSAH_CALL) ; // určuje lineární hlavní proceduru

// lineární hlavní procedura
DCL-PROC OBSAH_CALL;
    // prototyp
    DCL-PR OBSAH   packed(15: 5)     // vracená hodnota
                   EXTPROC('obsah'); // malá písmena!
        *n    packed(15: 5) value;   // parametry
        *n    packed(15: 5) value;
        *n    packed(15: 5) value;
    END-PR;
    dsply obsah ( 3: 4: 5 ); // statické volání podprocedury
END-PROC;
```

Spojení modulů

Oba moduly kompilujeme příkazem CRTRPGMOD, čili volbou 15 v PDM. Vzniknou tak moduly, které spojíme příkazem

```
CRTPGM PGM(*CURLIB/OBSAH_CALL) MODULE(OBSAH_CALL OBSAH_PRC)
```

do programu OBSAH_CALL v běžné knihovně. Po jeho spuštění se vypíše výsledek

```
DSPLY  plocha = 6.00000.
```

Procedury v jazyku Cobol

Za příklad si opět zvolíme výpočet obsahu obecného trojúhelníka podle Heronova vzorce.

Nejprve ukážeme případ, kdy kombinujeme hlavní program i funkci *obsah* do jednoho zdrojového členu, tedy modulu.

Modul s hlavní procedurou a procedurou `obsah`

```
 PROCESS OPTIONS NOMONOPRC.
*   hlavní program
*   --------------
 IDENTIFICATION DIVISION. PROGRAM-ID. OBSAH_HLA.
 WORKING-STORAGE SECTION.
 01  a       PACKED-DECIMAL PICTURE S9(10)V9(5) VALUE 3.0.
 01  b       PACKED-DECIMAL PICTURE S9(10)V9(5) VALUE 4.0.
 01  c       PACKED-DECIMAL PICTURE S9(10)V9(5) VALUE 5.0.
 01  plocha  PACKED-DECIMAL PICTURE S9(10)V9(5).
 PROCEDURE DIVISION.
     CALL LINKAGE TYPE IS PROCEDURE "obsah",
        USING  BY VALUE a BY VALUE b BY VALUE c
        RETURNING plocha.
     DISPLAY "plocha = " plocha.

*   funkce obsah - jako "vložený program"
*   -------------------------------------
 IDENTIFICATION DIVISION. PROGRAM-ID. "obsah".
 WORKING-STORAGE SECTION.
 01  s             PACKED-DECIMAL PICTURE S9(10)V9(5).
 01  soucin-dec    PACKED-DECIMAL PICTURE S9(10)V9(5).
 01  plocha-dec    PACKED-DECIMAL PICTURE S9(10)V9(5).
 01  soucin-double COMP-2.
 01  plocha-double COMP-2.
 LINKAGE SECTION.
 01  a       PACKED-DECIMAL PICTURE S9(10)V9(5) VALUE 3.0.
 01  b       PACKED-DECIMAL PICTURE S9(10)V9(5) VALUE 4.0.
 01  c       PACKED-DECIMAL PICTURE S9(10)V9(5) VALUE 5.0.
 PROCEDURE DIVISION USING BY VALUE a BY VALUE b BY VALUE c
        RETURNING plocha-dec.
     IF a >= b + c OR b >= a + c OR c >= a + b
        MOVE -1.0 TO plocha-dec
        GOBACK
     END-IF.
     COMPUTE  s = (a + b + c) / 2.
     COMPUTE  soucin-double = s * (s - a) * (s - b) * (s - c).
     CALL PROCEDURE "sqrt" USING BY VALUE soucin-double
        RETURNING plocha-double.
     MOVE plocha-double TO plocha-dec.
 END PROGRAM "obsah".
 END PROGRAM OBSAH_HLA.
```

Program se kompiluje příkazem CRTBNDCBL, tedy volbou 14 v PDM, kdy vzniká provizorní modul v knihovně `QTEMP`, který se "spojí" do výsledného programu a pak se smaže. Je ovšem třeba zadat parametr `BNDDIR(QC2LE)`, aby se připojila C funkce `"sqrt"`:

```
CRTBNDCBL PGM(OBSAH_HLA) BNDDIR(QC2LE).
```

Modul obsahuje hlavní program `OBSAH_HLA` a vnořený (nested) program `"obsah"`. Program `"obsah"` je funkce, protože vrací hodnotu. V terminologii jazyka ILE/Cobol je to "ILE

procedure”, což odpovídá obecnému pojmu “subprocedure”.[5] Pro naši ukázku jsme volili jméno v malých písmenech, jak je to obvyklé v jazyku C.

Příkaz `CALL [LINKAGE TYPE IS] PROCEDURE` v hlavním programu volá funkci `"obsah"`. V něm zadáme jméno funkce, parametry a návratovou hodnotu. Musíme ovšem zařídit, aby jméno funkce vystupovalo v doslovném znění, tj. s malými písmeny. Toho dosáhneme zápisem příkazu `PROCESS OPTIONS NOMONOPRC` na začátku zdrojového modulu nebo při kompilaci zadáním volby `*NOMONOPRC` v parametru `OPTION`. Bez tohoto parametru by se jméno převedlo do velkých písmen. Dekadické pakované parametry `a, b, c` se předávají hodnotou (`BY VALUE`), funkce vrací také dekadické číslo.

Ve funkci `"obsah"` se počítá součin dekadických čísel, který se při dosazení do výsledku konvertuje do čísla v pohyblivé čárce typu `COMP-2`, což je totéž jako typ `double` v jazyku C. Toto číslo je zapotřebí jako argument standardní funkce `"sqrt"` jazyka C, jejíž výsledek -– odmocnina – je také typu `double`. Příkaz `MOVE` pak toto číslo převede do dekadického pakovaného tvaru.

V jazyku Cobol se pro ILE proceduru neuvádí žádný prototyp, na rozdíl od jazyků C a RPG. Chybí zde tedy statická kontrola parametrů při kompilaci. Proto je nutné dbát na přesné dodržení počtu, typů a pořadí parametrů a typu návratové hodnoty; v případě chyby jsou výsledky nepředvídatelné.

Výsledek spuštění programu `OBSAH_HLA` se vypíše bez vyznačené desetinné tečky:

```
plocha = 000000000600000.
```

Trojúhelník o stranách 3, 4, 5 je pravoúhlý, proto dvojnásobek jeho obsahu je dán součinem jeho odvěsen: 3 x 4 = 12.

Jiný způsob, jak sestavit program, je osamostatnit proceduru “obsah” a hlavní proceduru do samostatných zdrojových členů, zkompilovat je do modulů příkazem `CRTCBLMOD` a spojit je příkazem `CRTPGM`.

5 V programu lze zadat více vnořených programů.

Modul OBSAH_PRC s procedurou obsah

```
 PROCESS OPTIONS NOMONOPRC.
*   funkce obsah - jako ILE procedura
*   ---------------------------------
 IDENTIFICATION DIVISION. PROGRAM-ID. "obsah".
 WORKING-STORAGE SECTION.
 01  s             PICTURE S9(10)V9(5).
 01  soucin-dec    PICTURE S9(10)V9(5).
 01  plocha-dec    PICTURE S9(10)V9(5).
 01  soucin-double COMP-2.
 01  plocha-double COMP-2.
 LINKAGE SECTION.
 01  a       PICTURE S9(10)V9(5).
 01  b       PICTURE S9(10)V9(5).
 01  c       PICTURE S9(10)V9(5).
 PROCEDURE DIVISION USING BY VALUE a BY VALUE b BY VALUE c
        RETURNING plocha-dec.
     IF a >= b + c OR b >= a + c OR c >= a + b
        MOVE -1.0 TO plocha-dec
        GOBACK
     END-IF.
     COMPUTE  s = (a + b + c) / 2.
     COMPUTE  soucin-double = s * (s - a) * (s - b) * (s - c).
     CALL PROCEDURE "sqrt" USING BY VALUE soucin-double
        RETURNING plocha-double.
     MOVE plocha-double TO plocha-dec.
 END PROGRAM "obsah".
```

Modul OBSAH_CALL – hlavní procedura

```
 PROCESS OPTIONS NOMONOPRC.
*   hlavní program - volá funkci "obsah"
*   ------------------------------------
 IDENTIFICATION DIVISION. PROGRAM-ID. OBSAH_HLA.
 WORKING-STORAGE SECTION.
 01  a       PICTURE S9(10)V9(5) VALUE 3.0.
 01  b       PICTURE S9(10)V9(5) VALUE 4.0.
 01  c       PICTURE S9(10)V9(5) VALUE 5.0.
 01  plocha  PICTURE S9(10)V9(5).
 PROCEDURE DIVISION.
     CALL LINKAGE TYPE IS PROCEDURE "obsah",
        USING  BY VALUE a BY VALUE b BY VALUE c
        RETURNING plocha.
     DISPLAY "plocha = " plocha.
 END PROGRAM OBSAH_HLA.
```

Spojení modulů a spuštění programu

Oba moduly kompilujeme příkazem CrtCblMod, čili volbou 15 v PDM. Vzniknou tak moduly, které spojíme příkazem

```
CRTPGM PGM(*CURLIB/OBSAH_CALL) MODULE(OBSAH_CALL OBSAH_PRC) BNDDIR(QC2LE)
```

do programu OBSAH_CALL v běžné knihovně. Parametr BndDir(QC2LE) je nutný kvůli připojení funkce *sqrt()* jazyka C. Po spuštění programu se vypíše výsledek bez vyznačené desetinné tečky:

```
plocha = 000000000600000.
```

Procedury v jazyku CL

V jazyku CL není psaní procedur typu funkce možné, protože CL program nemůže vracet hodnotu. Navíc není možné specifikovat vstupní parametry jako předávané hodnotou jako jsme to zvolili v našem příkladu.

Proto uvedeme jen CL program OBSAH_CALL, který volá funkci *obsah* vytvořenou z modulu OBSAH_PRC vzniklého *z libovolného předchozího příkladu*.

```
DCL         VAR(&A) TYPE(*DEC) LEN(15 5) VALUE(3)
DCL         VAR(&B) TYPE(*DEC) LEN(15 5) VALUE(4)
DCL         VAR(&C) TYPE(*DEC) LEN(15 5) VALUE(5)
DCL         VAR(&PLOCHA) TYPE(*DEC) LEN(15 5)
DCL         VAR(&PLOCHA_C) TYPE(*CHAR) LEN(16)

CALLPRC     PRC('obsah') PARM((&A *BYVAL) (&B *BYVAL) +
              (&C *BYVAL)) RTNVAL(&PLOCHA)

CHGVAR      VAR(&PLOCHA_C) VALUE(&PLOCHA)
SNDPGMMSG   MSG('plocha = ' *BCAT &PLOCHA_C) TOPGMQ(*EXT)
```

Volání funkce provádíme příkazem CALLPRC (Call Procedure). Jméno funkce musíme uvést malými písmeny v apostrofech, parametry &A, &B, &C předáme hodnotou (*BYVAL) a návratovou hodnotu &PLOCHA zadáme v parametru RTNVAL.

Vzhledem k tomu, že parametry jsou dekadická pakovaná čísla o rozměrech (15 5), můžeme příkaz CALLPRC napsat také s konstantními parametry takto:

```
CALLPRC     PRC('obsah') PARM((3 *BYVAL) (4 *BYVAL) +
              (5 *BYVAL)) RTNVAL(&PLOCHA)
```

V jazyku CL rovněž neuvádíme žádný prototyp. Ani zde tedy neprobíhá při kompilaci programu žádná (statická) kontrola volání, nesprávně zadané parametry nebo návratová proměnná se projeví *až při výpočtu*!

Program musíme zkompilovat příkazem CRTCLMOD (volbou 15 v PDM) a spojit příkazem CRTPGM:

```
CRTPGM PGM(*CURLIB/OBSAH_CALL) MODULE(OBSAH_CALL OBSAH_PRC) BNDDIR(QC2LE).
```

Po spuštění programu se vždy vypíše výsledek

```
plocha = 000000006.00000,
```

ať už je spojen s modulem OBSAH_PRC z kteréhokoliv jazyka (C, RPG, Cobol).