KFKI-73-50

Lovas Istvánné

SYMBOLANG - egy FORTRAN szubrutinrendszer polinomok szimbolikus kezelésére

Hungarian Academy of Sciences

CENTRAL RESEARCH INSTITUTE FOR PHYSICS

BUDAPEST

KFKI-73-50

SYMBOLANG

EGY FORTRAN SZUBRUTINRENDSZER POLINOMOK SZIMBOLIKUS KEZELÉSÉRE

Lovas Istvánné
Központi Fizikai Kutató Intézet, Budapest
Számitástechnikai Főosztály

KIVONAT

A SYMBOLANG egy polinomok szimbolikus kezelésére szolgáló szubrutin csomag, mely a SLIP szimmetrikus listakezelő rendszerre épül. A rendszer rutinjainak segitségével elvégezhető szimbolikus müveletek: összeadás, szorzás, differenciálás, integrálás, behelyettesités, egyszerüsités, csonkitás, stb.
Az eredetileg A. Lapidus és M.Goldstein által kidolgozott rendszert az ICL 1905 számológépre alkalmaztuk.

РЕЗЮМЕ

SYMBOLANG - это набор подпрограмм, используемый для символических операций с полиномами. Он построен на базе системы обработки симметрических списков SLIP. С помощью подпрограмм системы могут быть осуществлены символические операции: сложение, умножение, дифференцирование, интегрирование, замещение, сокращение, отбрасывание и т.д.
Оригинал системы разработанной A. Lapidus и M. Goldstein мы применили для вычислительной машины ICL 1905.

ABSTRACT

SYMBOLANG is a package of FORTRAN subroutines for the symbolic manipulation of polynomials, which is based on the SLIP symmetric list processor. Symbolic addition, multiplication, differentiation, integration, substitution, simplification, truncation, etc. of polynomials are possible within the system. The system originally developed by A. Lapidus and M. Goldstein was implemented with some extensions on an ICL 1905 computer.

1. Bevezetés

Az elmult években egyre inkább az érdeklődés előterébe került a számológépek alkalmazása nem numerikus feladatok megoldására. Ennek a területnek egyik jelentős ága a matematikai kifejezések formális kezelése, az ugynevezett algebrai manipuláció.

Az algebrai manipulátor programokkal szimbolikusan ábrázolt kifejezéseken végezhetünk müveleteket.

A következőkben egy algebrai manipulációs feladatokra készült rendszert ismertetünk, a SYMBOLANG szubrutinrendszert, amely valós együtthatóju polinomok szimbolikus kezelésére szolgál. Segitségével elvégezhetők különböző polinom-müveletek: polinomok összeadása, szorzása, n-edik hatványra való emelése, differenciálása, stb., továbbá különböző átalakitások: polinom behelyettesitése egy változó helyébe, a lehetséges egyszerüsitések elvégzése, stb.

Egy algebrai manipulációs rendszernél alapvető fontosságu a belső ábrázolás módjának, az adatstrukturának a megválasztása. A problémát az jelenti, hogy mig pl. a FORTRAN nyelvben két szám /konstans, vagy változó/ összeszorzásának eredménye ismét egyetlen szám, addig a SYMBOLANG rendszerben két polinom szorzásának eredménye egy polinom, amelyről nem tudjuk előre, hogy hány tagból fog állni /mivel az összevonásnál több tag kieshet/. Ebből következik, hogy az algebrai manipulációs programok dinamikus adatbázist igényelnek, amelyben az adatok elérése a fizikai tárolás sorrendjétől függetlenül, az adatok belső összefüggése alapján történik.

Ilyen tárológazdálkodást biztositanak a különböző listakezelő nyelvek, ill. rendszerek /LISP, SLIP, IPL-V/. Ezért az algebrai kifejezések szimbolikus kezelése általában valamilyen listakezelő rendszeren alapul.

A SYMBOLANG szubrutinrendszer a SLIP listakezelő rendszerre épül ([1],[2]). Mivel a SLIP a FORTRAN nyelvbe van beágyazva, bizonyos értelemben a FORTRAN nyelv egy kibővitését jelenti. A SYMBOLANG a FORTRAN egy további bővitéseként fogható fel. Igy a rendszer felhasználójának a FORTRAN nyelv numerikus lehetőségei is rendelkezésére állnak.

A SYMBOLANG rendszert eredetileg Lapidus és Goldstein ([3],[4]) dolgozta ki. Az irodalom alapján implementáltuk a Központi Fizikai Kutató Intézet ICL 1905-ös számológépére. Célunk egy müszaki és tudományos feladatok megoldására jól használható rendszer megvalósitása volt.

A rendszer mindazon gépeken használható, amelyeken a SLIP is implementálva van, igy a CDC 3300 gépen is.

A szubrutincsomag szerkezeti felépitése nagy flexibilitást biztosit, ezért könnyen bővithető további rutinokkal /a SLIP nyelv ismeretében/. Már az implementáció során több uj rutint illesztettünk a rendszerhez.

A következő ismertetést ugy épitettük fel, hogy az a SLIP rendszer ismerete nélkül is érthető. Azok számára, akik a rendszert mélyebben meg akarják ismerni, vagy tovább akarják fejleszteni, a 2.2 pontban irtuk le a belső ábrázolás formáját.

2. Kifejezések ábrázolása a SYMBOLANG-ban.

2.1. Külső ábrázolás

A SYMBOLANG operandusai kifejezések, melyek számokat és szimbolikus változókat tartalmazhatnak. A kifejezések valós együtthatóju polinomok lehetnek.

Egy kifejezés vagy egy tagból áll, vagy végesszámu tag összege. A kifejezéseket zárójelmentes formában kell megadni, a rendszerben a zárójel használata nem megengedett.

Egy tag vagy konstans, vagy egy együtthatóból és végesszámu szimbolikus változó szorzatából áll, minden változóhoz tartoznia kell egy kitevőnek.

A kitevőt mindig valós számként ábrázoljuk /akkor is meg kell adni, ha 1.0-gyel egyenlő/.

Megjegyezzük, hogy a SYMBOLANG rendszer legtöbb rutinja nem használja ki azt, hogy formálisan kezelt kifejezés egész kitevőjü polinom, hanem megengedi azt, hogy a szereplő kitevők tetszőleges valós számok legyenek. Csak egyes rutinok /pl. SUBST, INTEGR/ használják ki a kifejezés speciális alakját. Az egyes rutinok ismertetésénél megadjuk a kitevőkre vonatkozó megszoritásokat.

Egy SYMBOLANG programban egy kifejezésre a kifejezés nevén keresztül hivatkozunk. A név egy FORTRAN változó, melyet csak a SYMBOLANG szubrutinokban használhatunk fel.

A kifejezés nevét egy SYMBOLANG utasitással - a NEWEXP függvény hivásával - generáljuk. /A függvény értéke a nevet adja eredményül./ Ezután a kifejezés nevéhez - ujabb SYMBOLANG utasitásokkal - hozzárendeljük a kifejezés tagjainak nevét.

A tag neve, a kifejezés nevéhez hasonlóan, szintén FORTRAN változó, melyet csak SYMBOLANG utasitásban használhatunk fel, és ugyancsak egy függvény /a NEWTERM függvény/ hivásával kell generálni.

A felhasználó általában a tag nevét, a kifejezés előállitásán kivül, nem használja.

A tag nevéhez - SYMBOLANG utasitással - rendeljük hozzá az adatokat. A szimbolikus változók neve egy Hollerith konstans /az ICL 1900-as és a CDC 3300-as reprezentációban legfeljebb 8 karakter lehet/. A space karakterek is szignifikánsak /pl. ▽EPSILON és EPSILON▽ két különböző változót jelent/. A ▽ jel a szóköz /space/ karakter jele az ICL 1900-as gépi reprezentációban.
Az adatokat a következő sorrendben kell a tagokhoz rendelni:

1. Együttható Lebegőpontos számként kell megadni. Mindig szerepelnie kell, még abban az esetben is, ha az együttható 1.0.
2. Első változó neve Hollerith konstansként adhatjuk meg.
3. Első változó kitevője Lebegőpontos számként kell megadni. Mindig szerepelnie kell, még abban az esetben is, ha a kitevő 1.0.
4. Második változó neve
5. Második változó kitevője
. .
. .
2N. N-edik változó neve
2N+1. N-edik változó kitevője

* * * *

A következő pont a SYMBOLANG adatstrukturájának leirását tartalmazza. A felhasználó, ha nem kiván a listaszerkezetekkel foglalkozni, kihagyhatja a 2.2 pontot, melyet csak a teljesség kedvéért helyeztünk itt el. A későbbiek megértéséhez elolvasása nem szükséges.

2.2. Belső ábrázolás

A SYMBOLANG alapját a dinamikus tárfelosztással rendelkező SLIP listakezelő rendszer képezi, mely speciális adatstrukturán, szimmetrikus listákon operál.

A listák elemekből állnak, minden listának van egy kitüntetett eleme, a lista feje. A listafejen keresztül a listára való hivatkozás egyértelmü. A listára való hivatkozás az ugynevezett listanévvel történik, a listanév tartalmazza a listafej gépi cimét. A listanév lehet FORTRAN program valamely változója, vagy lehet egy listaelem is.

A listafejen kivül kétféle listaelem létezik:

- atom
- kapcsolóelem

Az atom az adat hordozója. Az atom lehet egész vagy valós szám, vagy Hollerith konstans.

A kapcsolóelem olyan listaelem, mely adatként egy lista nevét, azaz egy listafej gépi cimét tartalmazza.

A listákból listastrukturákat épitettünk fel. Egy Y listáról azt mondjuk, hogy az X lista allistája, ha X tartalmaz egy olyan kapcsolóelemet, mely az Y lista fejére mutat.

A SYMBOLANG-ban egy polinomot egy listastrukturával ábrázolunk, amelyen a polinom minden tagjának egy - egy allista felel meg. A tagot ábrázoló allista a következő sorrendben tartalmazza az adatokat:

1. elem - A tag együtthatója /lebegőpontos szám/.
2. elem - Első változó neve /Hollerith konstans/.
3. elem - Az első változó kitevője /lebegőpontos szám/.
4. elem - Második változó neve.

5. elem - Második változó kitevője.

.

.

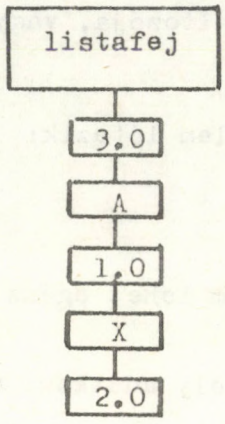
2N. elem - N-edik változó neve.

2N+1. elem - N-edik változó kitevője.

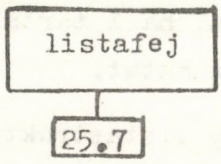
Például a

$3\,ax^2$

tag listán való ábrázolása a következő:



Egy konstans tag ábrázolása: 25.7



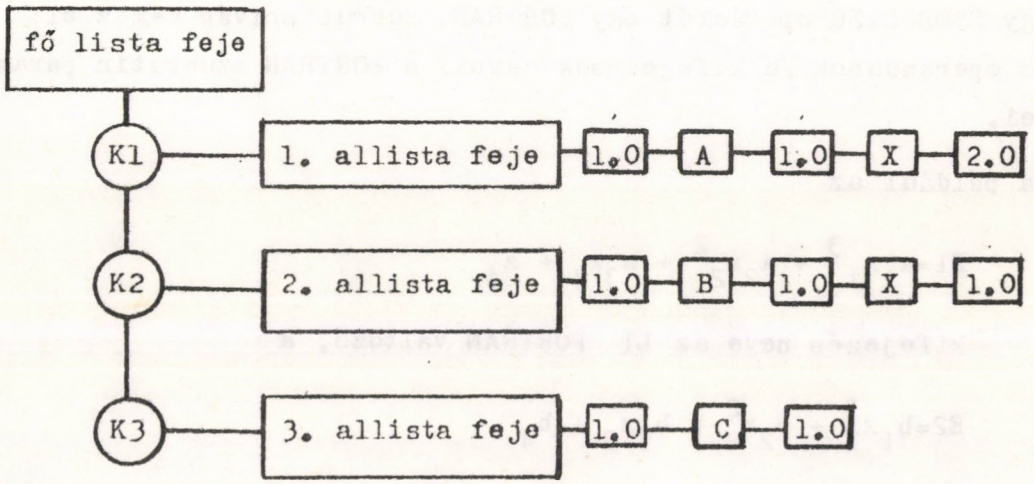
Egy tagot tartalmazó lista elemeinek a száma mindig 2N+1, ahol N a tagban szereplő változók száma.

A SYMBOLANG-ban egy kifejezést egy listastruktura ábrázol. A kifejezést ábrázoló lista a főlista, amelyen a kifejezés minden tagja allistaként helyezkedik el. A főlistán csak kapcsolóelemek vannak, ezek a kapcsolóelemek mutatnak az egyes tagokat tartalmazó allisták fejére.

Például az

$$ax^2 + bx + c$$

kifejezés ábrázolása a következő:



K1, K2, K3 a kapcsolóelemeket jelentik.

Egy tagból álló kifejezés fólistája egyetlen kapcsolóelemet tartalmaz, mely a tagot tartalmazó allista fejére mutat. Minden allista csak egy fólistához tartozhat, keresztreferenciákat a SYMBOLANG nem enged meg.

Egy SYMBOLANG programban egy kifejezésre való hivatkozás azt jelenti, hogy hivatkozunk arra a fólistára, melynek allistái a kifejezés tagjait tartalmazzák.

A hivatkozás a fólista nevén keresztül történik. A fólista nevét mint FORTRAN változót használjuk, és kifejezésnévnek hívjuk. A tagokat tartalmazó allisták nevét tagnévnek hívjuk. /Általában a programban a tagokat tartalmazó allisták nevére, mint FORTRAN változóra nincs szükség./

3. A SYMBOLANG program szerkezete.

3.1. SYMBOLANG utasitások.

A SYMBOLANG operátorai kifejezéseken /polinomokon/ értelmezett müveletek, /két polinom összeadása, szorzása stb./
Egy SYMBOLANG operációt egy FORTRAN szubrutinhivás végez el; az operandusok /a kifejezések nevei/ a FORTRAN szubrutin paraméterei.

Ha például az

$$E1=a_1x_1^3 + a_2x_2^2 + a_3x_3 + a_4$$

kifejezés neve az L1 FORTRAN változó, a

$$E2=b_1x_1^3 + b_2x_2^2 + b_3x_3 + b_4$$

kifejezés neve az L2 FORTRAN változó, akkor az

E3 = E1 * E2

müveletet a

CALL SUMPY (L1, L2, L3)

FORTRAN utasitás végzi el, ahol L3 a szorzat - polinomot tartalmazó kifejezés neve.

A SYMBOLANG utasitásokat - rutinokat - három nagy csoportra osztjuk.

3.1.1. Algebrai tipusu müveletek /rutinok/.

A következő müveletek végezhetők el az ide tartozó rutinokkal. /A zárójelben álló nagybetüs szavak a megfelelő SYMBOLANG szubrutinok nevei. A rutinokat a 4. fejezetben ismertetjük részletesen./

- Két kifejezés összeadása /ADD/
- Két kifejezés kivonása /SUBTR/

- Egy kifejezés szorzása konstanssal /CMULT/
- Két kifejezés szorzása /SUMPY/
- Egy kifejezés osztása konstanssal /DVSUMF/
- Egy kifejezés osztása szimbolikus változóval /DVSUMH/
- Behelyettesités /SBST/
- Egyszerüsités /SMPL/
- Egy polinom differenciálása /DERIV/
- Egy polinom integrálása /INTEGR/
- Csonkitás /magasabb foku tagok elhagyása/ /TRUNC/
- Lineáris egyismeretlenes egyenlet megoldása /SOLVE/
- Egy polinom értékének kiszámitása adott helyen /VALUE/

3.1.2. A kifejezéseket előállitó utasitások /rutinok/

Az ide tartozó rutinok segitségével generálhatjuk a kifejezéseket, illetve törölhetjük őket. Az utóbbira azért van szükség, hogy a feleslegessé vált kifejezések által lefoglalt memóriaterületet ujra szabaddá tehessük.

Egy kifejezés előállitását mindig a kifejezés nevének generálásával kell kezdeni.
A kifejezés nevét a

```
NAME = NEWEXP(A)
```

FORTRAN utasitás szolgáltatja.
Ezután a kifejezés nevéhez hozzá kell rendelni a kifejezés tagjainak a nevét. Ha a tagok nevei pl. NAMT1, NAMT2, a hozzárendelést a

```
CALL PUT (NAMT1, NAME)
CALL PUT (NAMT2, NAME)
```

utasitások végzik el.

Egy tag nevét a

```
NAMT1 = NEWTERM( A )
```

utasitás szolgáltatja. Ezután a tag nevéhez hozzá kell rendelni az

adatokat a megfelelő sorrendben. Ha pl. a $3x^2$ tagot kivánjuk előállitani a hozzárendelést a

```
CALL PUT( 3.0, NAMT1)
CALL PUT( 1HX, NAMT1)
CALL PUT( 2.0, NAMT1)
```

utasitások végzik el. A programozónak kell ügyelnie a megfelelő sorrendre /együttható, 1. változó, 1. változó kitevője, stb./
A következő programrészlet előállitja az $E = 3x^2 + 2$ kifejezést.

```
      .
      .
C     A KIFEJEZÉS NEVE = NAME
      NAME = NEWEXP ( A )
C     AZ ELSŐ TAG GENERÁLÁSA
      NTAG1 = NEWTERM ( A )
      CALL PUT( 3.0, NTAG1)
      CALL PUT( 1HX, NTAG1)
      CALL PUT( 2.0, NTAG1)
C     AZ ELSŐ TAG NEVET HOZZÁRENDELÜK A KIFEJEZÉSHEZ
      CALL PUT( NTAG1, NAME)
C     MÁSODIK TAG GENERÁLÁSA
      NTAG2 = NEWTERM ( A )
      CALL PUT( 2.0, NTAG2)
C     MÁSODIK TAG NEVET HOZZÁRENDELJÜK A KIFEJEZÉSHEZ
      CALL PUT ( NTAG2, NAME )
      .
      .
```

Az adatokat beolvashatjuk input perifériáról is a READTERM függvény segitségével. A függvény meghatározott formátum szerint olvassa be az adatokat és hozzárendeli a függvényérték által meghatározott tag nevéhez.

Példa egy 5 tagból álló kifejezés beolvasására.

```
C       A KIFEJEZÉS NEVE = NAME
        NAME = NEWEXP( A )
        DO 1 I = 1,5
        R = READTERM(K)
C       A FÜGGVÉNY AZ R NEVÜ TAGHOZ HOZZÁRENDELTE A
C       BEOLVASOTT ADATOKAT
        CALL PUT( R, NAME )
C       AZ R NEVÜ TAGOT HOZZÁRENDELTÜK A KIFEJEZÉSHEZ
1       CONTINUE
```

A felhasználó általában kifejezésekkel dolgozik, a kifejezés tagjainak nevére nincs szükség. A fenti példában az R FORTRAN változó a ciklus első lépésében az első tag nevét tartalmazza, miután az első tag nevét hozzárendeltük a kifejezéshez, az első tag nevére nincs többé szükségünk, az R FORTRAN változót felhasználhatjuk a második tag nevének ideiglenes tárolására, stb. /A megelőző példában ugyanigy felhasználhattuk volna az NTAG1 FORTRAN változót ismételten is, a kurrens tag nevének tárolására./

Ha egy kifejezésre nincs tovább szükségünk, például két polinom összeszorzása után csak a szorzat — polinommal fogunk tovább foglalkozni, a feleslegessé vált kifejezések által lefoglalt memóriaterületet célszerü felszabaditani. Ezt a

CALL DELEXP(NAME)

utasitás kiadásával kérhetjük. Miután a NAME nevü kifejezés által lefoglalt memóriaterületet felszabaditottuk, a NAME kifejezésre nem hivatkozhatunk. A NAME nevü FORTRAN változó nem tartalmaz többe kifejezésnevet, a továbbiakban újból felhasználhatjuk /akár mint egy később generálandó kifejezés nevét, akár mint "közönséges" FORTRAN változót/.

Egy kifejezés előállitását, mindig a kifejezés nevének generálása kell, hogy megelőzze. Ha például két kifejezést az L1 és L2-t összeszorozzuk, egy uj kifejezés keletkezik. Ezért a CALL SUMPY(L1, L2, L3) utasitást meg kell előznie az L3 kifejezésnevet generáló utasitásnak, az L3 = NEWEXP(A) utasitásnak.

Pl:

```
      .
      .
      L3 = NEWEXP ( A )
      .
      .
      CALL SUMPY ( L1, L2, L3)
```

Ha azonban például az igy előállitott L3 = L1 * L2 kifejezéshez hozzá akarjuk adni az L4 * L5 szorzatot, a

```
      CALL SUMPY( L4, L5, L3 )
```

utasitás a szorzáson kivül az összeadást is elvégzi /összevonás nélkül/.

A rutinok leirásában a 4. pontban az a meghatározás, hogy az „eredmény-kifejezés hozzáadódik az L3 kifejezéshez", azt jelenti, hogy, ha az L3 kifejezésnévhez valamilyen módon már rendeltünk tagokat, akkor tényleges összeadás történik /összevonás nélkül/, ha hozzárendelés még nem történt, ugy a hozzáadás hozzárendelést jelent.

Példa: Legyenek L1, L2, L3, L4 kifejezések, melyeket valamilyen módon, pl. beolvasással előállitottunk. Szükségünk van az LX = L1 * L2 + L3 * L4 kifejezésre, és az eredeti kifejezések közül csak L3-ra.

```
      .
      .
C     AZ EREDMÉNYKIFEJEZÉS NEVÉNEK ELŐÁLLITÁSA
      LX = NEWEXP ( A )
```

```
C       SZORZÁS: LX = L1 * L2
        CALL SUMPY( L1, L2, LX )
C       FELESLEGES KIFEJEZÉSEK TÖRLÉSE
        CALL DELEXP( L1 )
        CALL DELEXP( L2 )
C       SZORZÁS: LX = LX + L3 * L4
        CALL SUMPY( L3, L4, LX )
C       FELESLEGES KIFEJEZÉS TÖRLÉSE
        CALL DELEXP( L4 )
C       ÖSSZEVONÁS
        CALL SMPL( LX )
        .
        .
```

Kifejezés kinyomtatására szolgál a WREXP rutin, mely tagonként nyomtatja ki a kifejezést, a konstansokat és kitevőket lebegőpontos, a változókat karakter formában.

Megjegyezzük, hogy a SYMBOLANG rendszer tartalmaz egy segédrutint, mely egy tag változóit az ABC szerinti sorrendbe rendezi (SRTRM rutin). /A SYMBOLANG rutinok mindig feltételezik, hogy a változók az ABC szerinti sorrendben helyezkednek el a tagokban; maguk a rutinok ezt a sorrendet nem rontják el./

A READTERM rutin automatikusan hivja az SRTRM rendező rutint. Ha azonban a programban állitjuk elő a kifejezéseket, /a NEWTERM és PUT szubrutinnal/ egy - egy tag elkészülte után célszerü azt az SRTRM rutinnal rendezni.

3.1.3. Segédrutinok

Ide tartoznak azok a rutinok, amelyek szigoruan véve nem algebrai tipusu müveleteket végeznek, hanem különböző segédfeladatokat oldanak meg.

- Tag rendezése a változók nevének ABC - sorrendje szerint /SRTRM/
- Egy kifejezés szétválasztása adott változót tartalmazó és azt nem tartalmazó részre /BREAK/
- Egy adott kitevőjü változó együtthatójának megkeresése /GETCOE/
- Egy tagban egy adott változó kitevőjének meghatározása /POWER/
- Egy kifejezés lemásolása /LSSCPY/
- Egy tag lemásolása /CPYTRM/
- Egy kifejezés szorzása -1 -gyel /NEGATE/
- Adott együtthatóju adott változó legkisebb kitevőjének megkeresése /TRCAL/

3.2. A dinamikus adatbázis kijelölése

Egy SYMBOLANG program FORTRAN utasitások sorozatából áll. Minden programnak a futtatás kezdetén ki kell jelölnie a memória területet a kifejezések számára. A felhasználónak programjában deklarálnia kell egy egydimenziós tömböt és a polinom - manipulációkat végző rutinok hivása elött hivnia kell az INITAS szubrutint. Az INITAS rutin a tömb memóriaterületén előkésziti a továbbiakban dinamikusan kezelt adatbázist.
Tehát minden programnak a következőképpen kell kezdődnie:

```
MASTER PÉLDA
DIMENSION AREA( 5000 )
.
.
.
CALL INITAS ( AREA, 5000)
```

A továbbiakban a felhasználó közvetlenül nem hivatkozhat az AREA tömbre. /A DIMENSION utasitásban természetesen a fentitől különböző tömbméretet is megadhatunk. Az ICL 1905-ös és a CDC 3300-as gépi reprezentációban a tömb mérete csak páros szám lehet./

4. A SYMBOLANG rutinok részletes leirása

4.1. Algebrai müveleteket végző rutinok

SUBROUTINE ADD(L1, L2)

Az L1 nevü kifejezést hozzáadja az L2 nevü kifejezéshez, a lehetséges összevonásokat elvégzi. Az L1 kifejezés változatlan marad.
Az összeadás eredménye az L2 nevü kifejezés lesz.
Paraméterek: L1, L2-kifejezések nevei.

SUBROUTINE SUBTR(L1, L2)

Az L1 nevü kifejezést kivonja az L2 nevü kifejezésből, a lehetséges összevonásokat elvégzi. Az L1 kifejezés változatlan marad. A kivonás eredménye az L2 nevü kifejezés lesz.
Paraméterek: L1, L2-kifejezések nevei.

SUBROUTINE CMULT(L1, C, L2)

Az L1 nevü kifejezést megszorozza a C konstanssal, a szorzás eredménye hozzáadódik az L2 nevü kifejezéshez. Összevonás nem történik.
Az L1 nevü kifejezés változatlan marad.
Paraméterek: L1, L2-kifejezések nevei,
C-lebegőpontos szám.

SUBROUTINE SUMPY(L1, L2, L3)

Az L1 nevü kifejezést összeszorozza az L2 nevü kifejezéssel, az eredmény hozzáadódik az L3 nevü kifejezéshez. Összevonás nem történik.

Az L1 és L2 kifejezések változatlanok maradnak.

Paraméterek: L1, L2, L3-kifejezések nevei.

SUBROUTINE DVSUMF(L1, C, L2)

Az L1 nevü kifejezést elosztja a C lebegőpontos számmal, az eredménykifejezés hozzáadódik az L2 kifejezéshez. Összevonás nem történik. Az L1 kifejezés változatlan marad.

Paraméterek: L1, L2-kifejezések nevei,
C-lebegőpontos szám.

SUBROUTINE DVSUMH (L1, HVAR, L2)

Az L1 nevü kifejezést elosztja a HVAR szimbolikus változóval, az eredménykifejezés hozzáadódik az L2 kifejezéshez. Összevonás nem történik. Az L1 kifejezés változatlan marad.

Paraméterek: L1, L2-kifejezések nevei,
HVAR-szimbolikus változó neve /Hollerith konstans/.

SUBROUTINE SBST (L1, HVAR, L2, L3)

Az L2 nevü kifejezésben szereplő HVAR szimbolikus változó helyére az L1 nevü kifejezést helyettesiti. A lehetséges egyszerüsitéseket elvégzi. Az L1 kifejezés nem tartalmazhat olyan tagot, melyben a HVAR változó szerepel. Az L2 kifjezésben a HVAR szimbolikus változó csak pozitiv /lebegőpontos/ egész hatványon szerepelhet. Az L1 és L2 kifejezések változatlanok maradnak. Az eredmény az L3 kifejezéshez adódik hozzá.

Paraméterek: L1, L2, L3-kifejezések nevei,
HVAR szimbolikus változó neve /Hollerith konstans/.

UBROUTINE SMPL (L)

Az L nevü kifejezést egyszerüsiti, azaz a lehetséges összevonásokat elvégzi, és a kifejezést a tagok szerint rendezi. Két tag közül az kerül előbbre, amelyben a szubrutin az összehasonlitás során az első nem egyezés esetén az ABC-ben előbb álló

változót talál; ha pedig csak azonos változókat tartalmaznak, akkor az, amelyikben előbb fordul elő magasabb kitevő. A^2BC tehát megelőzi AB^2D-t, A^4X^2Y pedig AX^3Y^2-t. A rutin feltételezi, hogy a tagok a szimbolikus változók nevei szerint rendezve vannak. Az egyszerüsitett és rendezett eredmény az L kifejezés lesz.

Paraméterek: L-kifejezés neve.

SUBROUTINE DERIV(L1, HVAR, L2)

Az L1 nevü kifejezést a HVAR szimbolikus változó szerint deriválja. A derivált kifejezés az L2 nevü kifejezéshez adódik hozzá. Az L1 kifejezés változatlan marad.

Paraméterek: L1, L2-kifejezések nevei,
HVAR-szimbolikus változó neve /Hollerith konstans/.

SUBROUTINE INTEGR(L1, DX, L2)

Az L1 nevü kifejezést a DX szimbolikus változó szerint integrálja, és az eredményt hozzáadja az L2 nevü kifejezéshez. Az L1 kifejezés egyik tagjában sem lehet a DX változó kitevője - 1.0-gyel egyenlő.

Az L1 kifejezés változatlan marad.

Paraméterek: L1, L2-kifejezések nevei,
DX-szimbolikus változó neve /Hollerith konstans/.

SUBROUTINE TRUNC(L1, VAR, EXP)

Az L1 nevü kifejezésből elhagyja azokat a tagokat, melyek a VAR változót EXP vagy annál nagyobb hatványon tartalmazzák /a kifejezést csonkitja/.

Paraméterek: L1-kifejezés neve,
VAR-szimbolikus változó neve /Hollerith konstans/,
EXP-lebegőpontos szám.

SUBROUTINE SOLVE(L1, VAR, L2)

Az L1 nevü - a VAR változóban elsőfoku - kifejezést egyenletnek tekinti és VAR-ra megoldja. Az eredmény az L2 nevü kifejezéshez adódik hozzá. L1 változatlan marad.

Paraméterek: L1, L2-kifejezések nevei,
VAR-szimbolikus változó neve /Hollerith konstans/.

FUNCTION VALUE(L1, HVAR, E, L2, IND)

Az L1 kifejezésben a HVAR változó helyébe az E értékét helyettesiti. Ha az L1 kifejezés egyváltozós /csak a HVAR változót tartalmazza/, a függvény értéke a kifejezés behelyettesitési értékével lesz egyenlő, L1 és L2 változatlan marad, IND=0 lesz. Ha az L1 kifejezés többváltozós, a behelyettesités megtörténik és az eredmény-kifejezés /amely a HVAR változót természetesen már nem tartalmazza/ hozzáadódik az L2 kifejezéshez. A függvény értéke 0, IND értéke -1 lesz. Az L1 kifejezés változatlan marad.

Paraméterek: L1, L2-kifejezések nevei,
IND-egész változó, kimenő paraméter,
HVAR-szimbolikus változó neve /Hollerith konstans/,
E-valós szám.

4.2. Kifejezések előállitása és megszüntetése

FUNCTION NEWTERM(A)

Egy tagnevet generál. A függvény értéke a tag neve lesz. /Az A paraméternek nincsen szerepe, csak azért van rá szükség, mivel a FORTRAN-ban a függvényeknek legalább egy paramétere kell, hogy legyen. /A tag nevéhez a generálás után a PUT szubrutinnal hozzá kell rendelni a megfelelő adatokat /együttható, változónév, kitevő, stb./ a 3.1.2. pontban leirt módon.

FUNCTION NEWEXP(A)

Egy kifejezésnevet generál. A függvény értéke a kifejezés neve lesz. /Az A paraméternek nincsen szerepe/ A kifejezés nevéhez ezután a PUT szubrutinnal hozzá kell rendelni az egyes tagok neveit /l. 3.1.2. pont./

SUBROUTINE PUT (A, L)

Az L névhez hozzárendeli az A adatot. Ha L egy tag neve, ugy A-nak minden esetben adatnak kell lennie. Ha L egy kifejezés neve, ugy A-nak egy tag nevének kell lennie.

Paraméterek: L kifejezés vagy tagnév
A vagy adat, vagy tagnév

SUBROUTINE DELEXP (L)

Az L kifejezést megszünteti. A kifejezés tárolására szolgáló memóriaterület felszabadul /hozzácsatolódik a dinamikus adatbázishoz/. L-re a továbbiakban mint kifejezésnévre nem lehet hivatkozni.

Paraméterek: L-kifejezésnév.

FUNCTION READTERM (K)

K számu változót tartalmazó tag beolvasására szolgál. A függvény generál egy tagnevet és ehhez a névhez hozzárendeli a beolvasott tagot. A függvény értéke a tag neve lesz. K értéke maximálisan 20 lehet. A függvény a tag változóit ABC szerint rendezi.

A függvény csak az ICL 1900-as gépi reprezentációban használható, mivel a beolvasásnál az itt megengedett F0.0 formátumot használja.

A tagot az adatszalagon a következő formátumban kell megadni:

- a tag együtthatója /egy, az F0.0 formátumnak megfelelő valós szám/,

- egy db * karakter,
- első változó neve /8 karakter/,
- két db * karakter,
- első változó kitevője /az F0.0 formátumnak megfelelő valós szám/,
- egy db * karakter,
- második változó neve,

•
•

- K-adik változó neve,
- két db * karakter,
- K-adik változó kitevője.

Az F0.0 formátumnak megfelelően beolvasott számoknak /együttható és kitevők/ a végét egy darab space vagy newline karakter jelöli az adatszalagon. Az utolsó /K-adik/ kitevő után ujsor /newline/ karakternek kell következnie.

Példa az $3x^2y$ tag beolvasására:

R = READTERM (2)

ahol R a tag neve lesz.

Az adatszalag képe a következő:

3.0▽*▽▽▽▽▽▽▽X * * 2.0▽*▽▽▽▽▽▽▽Y * * 1.0 /newline/

/▽ a space karakter jele az ICL 1900 gépi reprezentációban/

Paraméter: K-egész szám.

SUBROUTINE WREXP (DENT, L)

Az L nevü kifejezést a sornyomtatón kinyomtatja a következő formátum szerint:

1 sor: EXPRESSION DENT,
ahol DENT 8 karakterből álló karaktersorozat.

2 sor: TERM NUMBER 1

3 sor: az első tag. Nyomtatási képe:

- együttható: E 20.12 specifikációju lebegőpontos szám;

- * karakter;
- 8 karakter, az első változó neve /8H specifikáció/;
- 2 db * karakter;
- E 20.12 specifikációju lebegőpontos szám, az első változó kitevője;
- * karakter;
- 8 karakter, a második változó;
- 2 db * karakter;
- E 20.12 specifikációju lebegőpontos szám, a második változó kitevője;

•
•
•

/Ha a tag egy sorban nem nyomtatható ki, a szubrutin a nyomtatási képet tördeléssel megfelelően alakitja; 3 változót nyomtat soronként./

k+3 sor: TERM NUMBER 2
k=..(n+2)/3 egész része, ahol n az első tagban szereplő változók száma

k+4 sor: a második tag. Nyomtatási képe /lásd fent/

•
•

utolsó sor: * * END OF EXPRESSION DENT * *
Paraméterek: DENT Hollerith konstans
L kifejezés neve
Például az a^2+ 2b kifejezést a
CALL WREXP(8HPOLINOM▽, L)
utasitás hatására az alábbi formában nyomtatja ki a program
/L az a^2+ 2b kifejezés neve/:
EXPRESSION POLINOM

```
TERM NUMBER 1
  0.100000000000E+01 *        A* *0.200000000000E+01
TERM NUMBER 2
  0.200000000000E+01 *        B* *0.100000000000E+01
* * END OF EXPRESSION POLINOM * *
```

4.3. Segédrutinok

SUBROUTINE SRTRM (L)

Az L nevü tag változóit ABC sorrendbe rendezi, az átrendezett tag neve továbbra is L lesz. /Minden más SYMBOLANG rutin feltételezi, hogy a változók a tagokban ABC rendben helyezkednek el. A rutinok a meglévő sorrendet nem rontják el./
A READTERM szubrutin mindig rendezi a beolvasott tagot a SRTRM rutinnal. Ha azonban a tagot a programban generáltuk /a NEWTERM és PUT rutinokkal/, ezután célszerü a SRTRM rutinnal a tagot rendezni.
Paraméter: L-tag neve.

SUBROUTINE BREAK (L1, CHAR, L2, L3)

Az L1 nevü kifejezés tagjait aszerint csoportositja, hogy tartalmazzák - e a CHAR nevü szimbolikus változót. A CHAR változót tartalmazó tagokat az L2 kifejezés, a CHAR nevü változót nem tartalmazó tagokat az L3 kifejezés fogja tartalmazni. Az L1 kifejezés változatlan marad. A BREAK rutin az SBST /behelyettesitő/ rutin segédrutinja.
Paraméterek: L1, L2, L3-kifejezésnevek,
CHAR-szimbolikus változó neve /Hollerith konstans/.

FUNCTION GETCOE (VAR, JEXP, L1, L2)

Az L1 nevü kifejezés minden tagját megvizsgálja. Ha egy tagban megtalálja a VAR nevü szimbolikus változót a JEXP hatványon, ugy a VAR * * JEXP együtthatóját hozzáadja az L2 kifejezéshez

/azaz a VAR * * JEXP-et tartalmazó tagból kiemeli a VAR * * JEXP-et/.

Az L1 kifejezés változatlan marad. A függvény értéke megadja annak a tagnak a sorszámát, melyben legutoljára szerepelt VAR * * JEXP. Ha a VAR * * JEXP nem fordul elő a kifejezésben, a függvény értéke 0.

Paraméterek: L1, L2 kifejezésnevek,
JEXP-egész szám,
VAR-szimbolikus változó /Hollerith konstans/.

FUNCTION POWER (L1, VAR)

Az L1 nevü kifejezésben megkeresi az első olyan tagot, melyben előfordul a VAR szimbolikus változó. A függvény értéke egyenlő lesz VAR kitevőjével ebben a tagban. Ha a VAR nevü változó nem fordul elő a kifejezésben, a függvény értéke 0 lesz. Az L1 kifejezés változatlan marad.

Paraméterek: L1-kifejezés neve,
VAR-szimbolikus változó neve /Hollerith konstans/.

FUNCTION LSSCPY (L)

Az L nevü kifejezést lemásolja. A függvény értéke az uj kifejezés neve lesz. Az L kifejezés változatlsn marad.

Paraméter: L-kifejezés neve.

FUNCTION CPYTRM (L1, L2)

Az L1 nevü kifejezés első tagját átmásolja az L2 kifejezésre /hozzáadja az L2 kifejezéshez /.Az L1 kifejezésből elhagyja az átmásolt tagot. A függvény értéke 1.0, ha az L1 kifejezés nem volt üres, különben 0.

Paraméterek: L1, L2-kifejezésnevek.

SUBROUTINE NEGATE (L)

Az L nevü kifejezést -1.0-gyel megszorozza.

Paraméter: L-kifejezésnév.

SUBROUTINE TRMMPY (L1, L2, L3)

Az L1 és L2 nevü tagokat összeszorozza és az igy kapott tagot hozzárendeli az L3 kifejezéshez.

A rutin eredményeként L3 egy egytagu kifejezés lesz. L1 és L2 változatlanok maradnak. A SUMPY rutin segédrutinjai.

Paraméterek: L1, L2-tagok nevei,

L3-kifejezésnév.

SUBROUTINE DVTRMF (L1, F)

Az L1 nevü tagot elosztja az F lebegőpontos számmal.

Paraméterek: L1-tagnév,

F-lebegőpontos szám.

SUBROUTINE DVTRMH (L1, HVAR)

Az L1 nevü tagot elosztja a HVAR nevü szimbolikus változóval.

Paraméterek: L1-tagnév,

HVAR-szimbolikus változó neve /Hollerith konstans/.

SUBROUTINE TRCAL (L1, V1, V2)

Megkeresi az L1 nevü kifejezésben azt a tagot, melyben a V1 és V2 szimbolikus változók szerepelnek és melyben a V1 változó kitevője a legkisebb. **/Ezt a minimális kitevőt M-mel jelöljük./ Ezután az L1 kifejezésből elhagyja azokat a tagokat, melyekben a V1 változó kitevője nagyobb mint M.**

Paraméterek: L1-kifejezésnév,

V1, V2-szimbolikus változók nevei /Hollerith konstansok/.

5. A rendszer használatával kapcsolatos tudnivalók

5.1. A SYMBOLANG használata az ICL 1905-ös számológépen.

A SYMBOLANG - rendszer a FORTRAN nyelvnek egy speciális szubrutincsomaggal való kiterjesztését jelenti. Ezért elkészitettünk egy speciális könyvtárszalagot, mely a SYMBOLANG /és a SLIP/ -rendszer szubrutinjait tartalmazza.

Az ICL 1905-ös gépen a SYMBOLANG-ot használó programokat a „SLIP - LIBRARY" könyvtárszalaggal kell leforditani, amely a rendszeren kivül tartalmazza az XFAM FORTRAN fordítóprogramot és az SRF7 FORTRAN szubrutinblokkot is.

A programleiró szegmensben a rendszerben használt logikai periféria számokhoz perifériát kell rendelni:

```
INPUT 99 = TRO
OUTPUT 100 = LPO
```

A programot tilos COMPRESS INTEGER AND LOGICAL módban forditani. Segédtároló kezelésére a rendszer nem nyujt lehetőséget.

5.2. Hibajelzések. Gyakori hibaforrások.

A SYMBOLANG rutinok hibajelzései

A SYMBOLANG rendszer használata során az alábbi hibaüzeneteket kaphatjuk /az üzenetek a sornyomtatón jelennek meg/:

a/ ERROR IN INTEGRATION

A hibajelzés oka: az INTEGR szubrutinnal integrálandó kifejezés egyik tagjában az integrációs változó a -1.0 kitevőn szerepel.

b/ DATA ITEM ON EXPRESSION LIST

A hibajelzést a WREXP szubrutin adja. Oka: a kiirandó kifejezés formája hibás. A kifejezés nevéhez a generáláskor olyan FORTRAN változót rendeltünk hozzá, amely nem tag neve.

Példa:

```
Q = 0.0
NAME = NEWEXP(A)
R = READTERM( 2 * K + 1)
CALL PUT ( Q, NAME )
CALL WREXP( 5HPELDA, NAME )
```

A példában a WREXP szubrutin hibajelzést ad, mert a Q FORTRAN változó, melyet a NAME kifejezésnévhez rendeltünk, nem tag nevét tartalmazza.

A SLIP rendszer hibajelzései

A SYMBOLANG programokban a rendszer saját hibajelzésein kivül a SLIP szubrutinok alábbi hibaüzenetei is előfordulhatnak /ugyancsak sornyomtatón/:

a/ AVAILABLE SPACE EXHAUSTED

/a rendelkezésre álló hely kimerült/

A programot, ha lehetséges, ujból kell forditani, ugy, hogy az adatmező számára nagyobb helyet jelölünk ki. /Az AREA tömb deklarációjában és az INITAS szubrutin hivásában. L. 3.2. pont/

Sok esetben a program megfelelő megszervezésével elérhetjük a helyigény csökkenését: Pl. két kifejezés összeszorzása /SUMPY rutin/ után célszerü a SMPL rutinnal az eredményül kapott kifejezésben a megfelelő összevonásokat elvégezni, mielőtt a kifejezéssel további müveleteket végeznénk.

Meg kell jegyeznünk, hogy a formulák szimbolikus kezelésénél gyakran okoz problémát az, hogy az egyes müveletek /kifejezések szorzása, hatványozása/ során a tagok száma rohamosan növekszik. /L. pl. [5]/ Igy hosszabb, több müveletet igénylő átalakitások során előfordulhat, hogy a memóriában rendelkezésre álló hely kimerül,

/sokszor olyan esetekben is, amikor a végeredmény kifejezés - a megfelelő egyszerüsitések elvégzése után - már ábrázolható lenne a memóriában/.

b/ A LIST WAS REQUIRED AS AN OPERAND BUT WAS NOT FOUND

SYMBOLANG programban ez az üzenet azt jelenti, hogy egy szubrutin paramétereként nem kifejezés, ill. tag nevét tartalmazó változót adtunk meg. /Ilyen hibajelzést okozhat pl. ha egy kifejezést megszüntettünk a DELEXP rutinnal, a FORTRAN változót azonban, amelyik a kifejezés nevét tartalmazta, továbbra is kifejezésnévként használjuk./

A fenti két hiba fatális, a program futása nem folytatódik.

Egyéb hibaforrások

Gyakori hiba a SYMBOLANG-ban a szimbolikus változók nevének elirása. Emlékeztetünk arra, hogy a nevekben a szóköz /space/ karakter is szignifikáns. Tehát ALFA▼▼▼▼ és ▼ALFA▼▼▼ két különböző szimbolikus változót jelent.

A SYMBOLANG függvények tipusa a FORTRAN implicit tipusdeklarációinak felel meg: az I, J, K, L, M, N betükkel kezdődő azonositók egész tipusuak. A felhasználónak figyelemmel kell lennie a függvények tipusára, amikor hivja őket.

A tagnév, ill. kifejezésnév tárolása akár egész, akár valós FORTRAN változóban történhet. Azok a rutinok, amelyekben paraméterként szerepel tag - ill. kifejezésnév, nem vizsgálják a nevet tartalmazó változó tipusát.

Ha azonban a tag - ill. kifejezésnév függvényutasitásban kap értéket, ügyelnünk kell, hogy ne történjék felesleges és nem kivánatos konverzió.

Például ha egy tagnevet a NEWTERM függvénnyel akarunk generálni, a tagnévnek egész tipusu változóban kell lennie. Ha azonban a READTERM függvénnyel generálunk tagnevet, a baloldalon valós tipusu változónak kell állnia:

```
NTAG = NEWTERM(A)
```

ill.

```
TAG = READTERM(NUM)
```

Ha nem megfelelő tipusu változó áll a baloldalon, a program a névre való legközelebbi hivatkozáskor hibajelzéssel megáll.

Szubrutinok paramétereként szereplő tag - ill. kifejezésnév akár valós, akár egész tipusu változó lehet.

```
CALL PUT (TAG, NAME)
```

v.

```
CALL PUT (NTAG, NAME)
```

helyes utasitások, feltéve, hogy TAG - ill. NTAG a fenti módon generált tagnevet tartalmazza.

6. Példák.

6.1. Kifejezés hatványozása

A következő programrészlet az $(A+B+C)^3$ kifejezés kifejtett alakját állitja elő. /Feltesszük, hogy az A+B+C kifejezést előzőleg már előállitottuk, és nevét az LSUM FORTRAN változó tartalmazza./

```
......
C       GENERALJUK A LEXP SEGEDKIFEJEZEST
C       LEXP = X**3
    1   LEXP=NEWEXP(A)
        LTRM=NEWTERM(A)
        CALL PUT(1.0,LTRM)
        CALL PUT(1HX,LTRM)
        CALL PUT(3.0,LTRM)
C       AZ LTRM TAGOT HOZZARENDELJUK AZ LEXP
C       KIFEJEZESHEZ
        CALL PUT(LTRM,LEXP)
C       GENERALJUK AZ EREDMENYKIFEJEZES NEVET
        LRES=NEWEXP(A)
C       AZ LSUM KIFEJEZEST X HELYEBE
C       BEHELYETTESITJUK LEXP-BEN
C       A SBST RUTIN ELVEGZI A KIFEJTEST ES
C       A LEHETSEGES OSSZEVONASOKAT
C       AZ EREDMENY LRES-BE KERUL
        CALL SBST(LSUM,1HX,LEXP,LRES)
C       AZ LEXP SEGEDKIFEJEZES ALTAL ELFOGLALT
C       MEMORIATERULETET FELSZABADITJUK
        CALL DELEXP(LEXP)
......
```

6.2. Behelyettesités és csonkitás

Az LPOLX kifejezés az X változó egy P(X) polinomját tartalmazza, melyet a program a READTERM szubrutin ismételt hivásával olvas be. A program az $e^{P(X)}$ kifejezés Taylor-sorának azt a részletösszegét állitja elő, amely X-ben legfeljebb 7-edfoku tagokat tartalmaz.

```
.....
C       P(X) NEVENEK GENERALASA
        LPOLX=NEWEXP(A)
C       P(X) BEOLVASASA
        READ(99,1000)K
1000    FORMAT(I8)
C       K = P(X) TAGJAINAK SZAMA
        DO 10 J=1,K
        TERM=READTERM(1)
  10    CALL PUT(TERM,LPOLX)
C       P(X) RENDEZESE
        CALL SMPL(LPOLX)
C       AZ EXP(Z) FUGGVENY TAYLOR-POLINOMJANAK
C       GENERALASA. LEXPZ A TAYLOR-POLINOM NEVE
        LEXPZ=NEWEXP(A)
C       A KONSTANS TAG GENERALASA
        LT=NEWTERM(A)
        CALL PUT(1.0,LT)
        CALL PUT(LT,LEXPZ)
C       A J-EDIK TAG =(Z**J)/J!
        FACT=1.0
        DO 20 J=1,7
        LT=NEWTERM(A)
        EXP=FLOAT(J)
        FACT=FACT*EXP
        COEF=1.0/FACT
```

```
      CALL PUT(COEF,LT)
      CALL PUT(1HZ,LT)
      CALL PUT(EXP,LT)
C     A J-EDIK TAGOT HOZZARENDELJUK A
C     LEXPZ KIFEJEZESHEZ
   20 CALL PUT(LT,LEXPZ)
C     LNEW AZ EXP(P(X)) KIFEJEZES NEVE
      LNEW=NEWEXP(A)
C     BEHELYETTESITES,AZ EREDMENY LNEW-BEN
      CALL SBST(LPOLX,1HZ,LEXPZ,LNEW)
C     CSONKITAS: LNEW-BOL TOROLJUK A  7-NEL
C     NAGYOBB KITEVOJU TAGOKAT. A VALOS
C     KONVERZIO ESETLEGES PONTATLANSAGA
C     MIATT 7.1-ET ADUNK MEG KUSZOBKENT
      CALL TRUNC(LNEW,1HX,7.1)
......
```

7. Záró megjegyzések.

Befejezésül röviden áttekintünk néhány fontosabb problémát a SYMBOLANG rendszer implementációjával és használatával kapcsolatban. Röviden utalunk más formulakezelő rendszerekkel való egybevetésre és a továbbfejlesztés néhány lehetőségére.

7.1. Az implementáció és alkalmazás problémái.

7.1.1. Az implementáció egyszerüsége.

A SYMBOLANG rendszer a SLIP rendszerre és ezen keresztül a FORTRAN nyelvre épül. Implementációjához tehát nem szükséges fordító - v. értelmező program. Minden olyan gépen használható, amely FORTRAN fordítóprogrammal /és megfelelő méretü memóriával/ rendelkezik. /A SLIP rendszer implementációjához néhány alaprutint assembler nyelven meg kell irni. A SYMBOLANG rendszer csak FORTRAN nyelvü rutinokat tartalmaz./

7.1.2. Numerikus lehetőségek.

Mivel a rendszer FORTRAN-ra épül, a felhasználónak rendelkezésére állanak a FORTRAN nyelv numerikus lehetőségei.

7.1.3. Jelölésmód.

Mivel a rendszer az egyes polinom-müveleteket FORTRAN szubrutinok hivásával valósitja meg, a program e miatt nehezen áttekinthető, a programozás aránylag nehézkes feladat. A jelölésmód a matematikában megszokott jelölésmódtól távol áll. /Más rendszerek, pl. ALTRAN, FORMAC, a matematikaihoz közelebb álló jelölést használnak./

7.1.4. Valós aritmetika.

A rendszer a kifejezések együtthatóit és kitevőit mindig valós számként tárolja. Hosszadalmas átalakításoknál ez a kerekitési hibák halmozódásához vezet. /Ebből következőleg pl. előfordulhat, hogy egyszerüsitésnél egy tag, amelynek ki kellene esnie, 0-tól különböző kis együtthatóval a kifejezésben marad./

Egyes rendszerek az ilyen hibák elkerülése céljából többszörös hosszuságu egész, valamint racionális aritmetikát is használnak.

7.1.5. Memória és időigény.

Mivel a SLIP rutinok legnagyobb része és a SYMBOLANG rutinok mind FORTRAN nyelven vannak megirva, a SYMBOLANG programok igen hely - és időigényesek /egy hasonló teljesitőképességü, de assembler nyelvü rutinokra alapozott rendszerhez képest/. A nagy helyfoglaláshoz hozzájárul a SLIP rendszer adatstrukturája is, /a szimmetrikus listastruktura/ amely önmagában is helyigényes.

7.2. A továbbfejlesztés néhány lehetősége.

7.2.1. Közelités a matematikai jelölésmódhoz.

Jelenleg folyamatban van a SYMBOL nyelv kidolgozása. Ezen a nyelven a polinom-müveleteket a FORTRAN nyelv aritmetikai utasitásaihoz hasonló, tehát a matematikai jelölésmódhoz közelebb álló formában irhatjuk le. A SYMBOL nyelv egyszersmind a FORTRAN nyelv utasitásait is tartalmazza, tehát a szó szükebb értelmében a FORTRAN nyelv kiterjesztése.

A SYMBOL nyelven irt programot először egy preprocesszor program dolgozza fel. Ez a polinomokra vonatkozó szimbolikus utasitások helyett a megfelelő SYMBOLANG rutinok hivását generálja. A feldolgozás eredménye egy FORTRAN program, amelyet már a szokásos módon a FORTRAN forditóprogrammal leforditathatunk.

Példaképen közöljük a 6.2. pontban szereplő feladatnak megfelelő SYMBOL nyelvü programot. A polinomokra vonatkozó szimbolikus utasitások első oszlopában # jel áll. A nem jelzett utasitások FORTRAN utasitások, ezeket a preprocesszor változatlan formában lemásolja.

```
.....
#     SVARIABLE X,Z
#     SEXPRESSION EXPZ,POLX,NEW
.....
#     LET POLX = READEXP
#     LET EXPZ = 1.0
      DO 20 J=1,7
#     LET EXPZ = EXPZ + ( 1.0/FACT(J))*Z**J
   20 CONTINUE
#     LET NEW = SUBST(POLX,Z,EXPZ)
#     TRUNCATE(NEW,X,7.1 )
.....
```

7.2.2. A rendszer kiterjesztése racionális és elemi függvények kezelésére.

Kivánatos volna a rendszert ugy továbbfejleszteni, hogy polinomokon kivül racionális törtfüggvények és elemi függvények /sin, cos, log/ kezelése is lehetséges legyen. Ez az utóbbi feladat azonban az adatstruktura megváltoztatását is igényelné.

Irodalom jegyzék

[1] J.Weizenbaum: Symmetric List Processor.
Communications of ACM, 6. 524-544 (1963)

[2] Krammer G. - Lovas Istvánné: SLIP - szimmetrikus listakezelő rendszer az MTA CDC 3300-as és az ICT 1905-ös számológépén.
KFKI-72-32

[3] A.Lapidus - M.Goldstein: Some Experiments in Algebraic Manipulation by Computer.
Communications of ACM 8. 5o1-5o8 (1965)

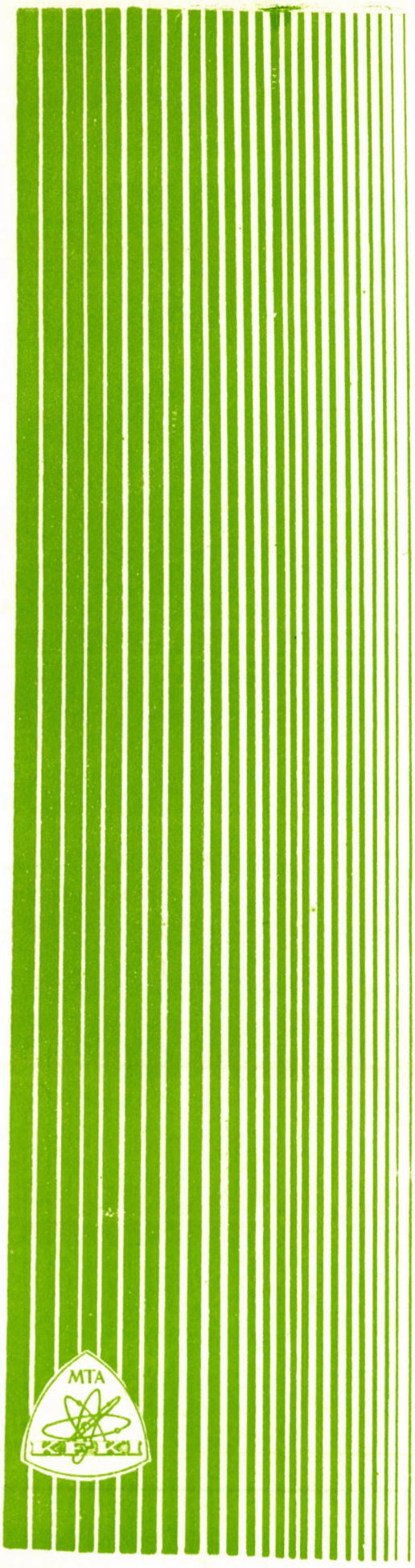
[4] CLAM, A Compatible List Processor and Algebraic Manipulator.
Courant Institute of Mathematical Sciences, 1965 (kézikönyv)

[5] D.Barton - J.P.Fitch: Review of Algebraic Manipulative Programs.
The Computer Journal 15. 362-381 (1972)

[6] A.D.Hall: The ALTRAN System for Rational Function Manipulation - A Survey.
Communications of ACM 14. 517-521 (1971)

[7] J.E.Sammet - E.R.Bond: Introduction to FORMAC.IEEE Transactions Electron. Computers EC-13 386-394 (1964)

[8] G.E.Collins: PM, A System for Polynomial Manipulation.
Communications of ACM, 9. 578-589 (1966)



Kiadja a Központi Fizikai Kutató Intézet
Felelős kiadó: Varga László, a KFKI Mérés- és Számitástechnikai Tudományos Tanácsának szekcióelnöke
Szakmai lektor: Zimányi Józsefné
Példányszám: 195 Törzsszám: 73-8995
Készült a KFKI sokszorositó üzemében
Budapest, 1973. szeptember hó