9. LotusScript

LotusScript tai programavimo kalba, leidzianti automatizuoti darba su firmos Lotus produktais. Tai BASIC tipo kalba, tuo paciu metu esanti ir objektinio-programavimo kalba.

Skriptas (Script) - tai pradinis programos LotusScript kalba tekstas. Sukompiliuotas skriptas vadinamas moduliu. Skripta sudaro LotusScript kalbos operatoriai. Kiekvienas operatorius sudaromas iš raktiniu zodziu, operacijos zenklu, kintamuju ir funkciju vardu, literalu ir skyrybos (punktacijos) zenklu, kurie rašomi pagal kalbos sintakses taisykles.

Skripto operatoriai – tai simboliu eilutes. Tos eilutes, prieš skripto ivykdyma, nuosekliai yra apdorojamos kompiliatoriaus, kad gauti vykdoma moduli.

Be operatoriu skripte gali buti kompiliatoriaus direktyvos.

Komentaro direktyva:

%REM … %ENDREM

Kai kompiliatorius aptinka %REM direktyva, jis ja praleidzia iki pat %ENDREM.

Itraukimo direktyva:

%INCLUDE <failo vardas>

Kai kompiliatorius aptinka tokia direktyva, itraukia (iterpia) i skripta teksta, esanti nurodytame faile, ir toliau kompiliuoja skripta, pradedant nuo iterpto teksto pradzios.

Skripte gali buti pavartoti ir vienos eilutes komentarai, eilutes pradzioje patalpinus operatoriu REM – tekstas iki eilutes pabaigos bus interpretuojamas komentaru.

Komentarai gali prasideti ir eilutes viduryje - viskas kas yra uz apostrofo (‘) – laikoma komentaru.

Skripto operatorius uzsirašo kaip simboliu eilute. Jis gali prasideti eilutes pradzioje ar po bet kokio skaiciaus tarpu. Galimos ir tuščios eilutes. Paprastai vienoje eiluteje rašoma vienas operatorius, taciau vienoje eiluteje ju gali buti ir daugiau. Ir atvirkščiai, vienas operatorius gali buti uzrašomas keliose eilutese. Norint pratesti operatoriu kitoje eiluteje, einamosios eilutes gale reikia patalpinti pabraukimo zenkla (_). Jei šis zenklas yra komentaro eiluteje, - jis neinterpretuojamas, kaip pernešimo simbolis.

Du operatoriai esantys vienoje eiluteje atskiriami dvitaškiu (:).

Kompiliatoriaus direktyvos ir operatoriai su zyme butinai turi buti pirmais operatoriais eiluteje.

Kai kurie operatoriai gali ne tik uzimti kelias teksto eilutes, bet ir tureti savyje kitus operatorius, t.y. galimi operatoriu blokai. LotusScript turi visa eile raktiniu zodziu, leidzianciu apibrezti blokus. Paminesime tik keleta iš ju:

FUNCTION … END FUNCTION - funkcijos apibrezimas

WHILE … WEND - ciklo apibrezimas.

Kalbos elementai. Tai:

- literalai – skaiciai ir simboliu eilutes;
- identifikatoriai;
- operaciju zenklai;
- raktiniai zodziai;
- specialieji simboliai.

Skaiciai: dešimtainiai sveikieji (123, -10, 1234567890), dešimtainiai su trupmenine dalimi (3.14159), skaiciai mokslineje notacijoje (77.7E+02). Sveikieji skaiciai taip pat gali buti ir dvejetaineje (&B1, &B10000000, &B100101 – prasideda prefiksu &B ir sudaro nedaugiau 32 skaitmenu), aštuntaineje (&O777, &O124 – prasideda prefiksu &O, sudaro nedaugiau 11 skaitmenu) ir šešioliktaineje (&H123, &HFA9 – prasideda prefiksu &H, nedaugiau 8 skaitmenu) formoje.

Simboliu eilutes, arba simboliu literalas, - tai simboliu sekos tarp dvigubu kabuciu. Maksimalus eilutes ilgis – 32 KB. Jei simboliu eilute turi uzimti kelias teksto eilutes, - tai simboliai apskliaudziami ne kabutemis, o vertikaliais brukšniais, pvz.:

| Tai labai ilga simboliu eilute,
kuri netelpa vienoje eiluteje |

Identifikatorius – tai vardas, kuris gali buti priskirtas kintamajam, konstantai, tipui, klasei,

funkcijai, paprogramei ar zymei. Vietoje termino “identifikatorius” daznai vartojamas terminas “vardas”. Tai simboliu eilute, sudaryta iš lotyniškų raidziu, skaitmenu ir pabraukimo zenklo, prasidedanti raide ir neilgesne nei 40 zenklu. Didziosios raides identifikatoriuje nesiskiria nuo mazuju. Identifikatoriai gali baigtis vienu iš simboliu-sufiksu, nurodanciu kintamojo tipa: %, &, !, #, @ ir $. Šie sufiksai neiskaiciuojami i identifikatoriaus ilgi. Identifikatoriuje gali buti pavartota ir spec. zenklas tik tuomet prieš ji dedamas simbolis “~”. Pvz., Valiuta~$UsDollar, Breaks~{123~}.

Operacijos ir reiškiniai. Reiškiniai - tai leistinos operandu ir operacijos zenklu kombinacijos. Kiekvienas reiškinys yra arba skaitinis arba simbolinis. Visos operacijos yra arba binarines arba unarines.

Operacijos su skaiciais (prioritetu mazejimo tvarka):

^ (kelimas laipsniu); - (minusas); * (daugyba); / (dalyba); \ (sveikaskaitine dalyba); MOD (sveikaskaitines dalybos liekana); + (sudetis); - (atimtis); = (lygu); <>, >< (nelygu); < (maziau); > (daugiau); <=,=< (maziau lygu); >=,=> (daugiau lygu); NOT (loginis neigimas); AND (loginis “ir”); OR (loginis “arba”); XOR (loginis “X arba“- moduliu 2); EQV (loginis ekvivalentumas); IMP (logine implikacija).

Palyginimo operaciju rezulatas: 1 (TRUE), 0 (FALSE) arba NULL (praktiškai tuomet, kai abu operandai yra NULL).

Operacijos su simboliu eilutemis (prioritetu mazejimo tvarka): +, & (konkatenacija); = (lygu); <>, >< (nelygu); < (maziau); > (daugiau); <=,=< (maziau lygu); >=,=> (daugiau lygu); LIKE (angl., pattern matching).

Raktiniai zodziai – tai operatoriu komponenciu (ON, END,..), vidiniu tipu, funkciju bei paprogramiu vardai ir vidiniu reikšmiu bei operaciju identifikatoriai - viso apie 250 raktiniu zodziu.

Specialieji simboliai: “” – simbliu eiluciu atskyrejai; | - simboliu eiluciu, uzimanciu kelias teksto eilutes, atskyrejas; : - keliu operatoriu vienoje eiluteje atskyrejas; %, &, ! – duomenu tipu sufiksai; , - saraš o elemetu atskyrejas; ‘ – veinos eilutes komentaro pozymis; _ - operatoriaus pernešimo i kita eilute pozymis; (,), *,..

Kintamieji ir konstantos - vardines atminties sritys duomenims saugoti. Kintamieji ir konstantos visuomet yra tam tikro tipo.

Skaliariniai tipai:

- INTEGER (sufiksas %) – trumpas (2 baitai) sveikas skaicius;
- LONG (sufiksas &) – ilgas (4 baitai) sveikas skaicius;
- SINGLE (sufiksas !) – trumpas (4 baitai) trupmeninis skaicius;
- DOUBLE (sufiksas #) – ilgas (8 baitai) trupmeninis skaicius;
- CURRENCY (sufiksas @) – dešimtainis su fiksuotu kableliu (4 skaitmenys po kablelio) (8 baitai);
- STRING (sufiksas #) – simboliu eilute (iki 32 KB).

Sufiksas yra pridedamas prie kintamojo vardo, kai kintamasis apibreziamas neišreikštai.

Sudetingi tipai:

- masyvas – iki 8 matavimu, kiekvieno iš kuriu diapazonas nuo –32768 iki 32767;
- saraš as - vienmatis vienatipiu duomenu apjungimas;
- struktura (TYPE …. ENDTYPE);
- klase (CLASS … ENDCLASS) – elementu-duomenu ir proceduru rinkinys;
- nuoroda i objekta (vidinio tipo objekta, klases objekta ar OLE objekta);
- VARIANT – gali buti susietas su kiekvienu aukšciau išvardintu tipu.

Vidines konstantos:

- NULL – duomenu nera (gali buti priskirta tik VARIANT tipo kintamajam);
- EMPTY – pradine VARIANT tipo reikšme;
- NOTHING – pradine nuorodos i objekta reikšme;
- PI – 3.14159…
- TRUE - logine rekšme “tiesa” (1);
- FALSE - logine rekšme “netiesa” (0).

Taciau LotusScript turi ir daugiau apibreztu konstantu, kurios yra faile LSCONST.LSS ir kuris gali buti itrauktas direktyva %INCLUDE.

Kiekvienas kintamasis gali buti matomas modulyje, paprogrameje ar funkcijoje, tipo ar klases apibrezime.

Kintamieji apibreziami operatoriumi DIM arba vienu iš operatoriu:

- STATIC – kintamuju reikšmes išsaugojamos tarp keliu funkcijos ar paprogrames kvietiniu;
- PUBLIC - kintamuju reikšmes pasiekiamos ir modulio, funkcijos ar paprogrames išoreje;
- PRIVATE - kintamuju reikšmes "nematomos" modulio, funkcijos ar paprogrames, kur jie apibrezti, išoreje.

Operatoriumi DIM apibreziamu kintamuju "matomumo" sritis nusakoma pagal nutylejima tam tikromis taisyklemis. Bendras operatoriaus pavidalas:

DIM | STATIC | PUBLIC | PRIVATE <kint. apibreztis> {,<kint. apibreztis>}

Kur <kint. apibreztis> :

- <kintamojo vardas>[sufiksas] [AS <kintamojo tipas>] – skaliarinio tipo kintamajam;
- <kintamojo vardas>[sufiksas] ([reziai]) [AS <kintamojo tipas>] – masyvui;
- <kintamojo vardas>[sufiksas] LIST [AS <kintamojo tipas>] – sarašui;
- <kintamojo vardas> AS [NEW] <kintamojo tipas> [argumentu sarašas] – nuorodai.

Pvz.:

```
DIM x AS INTEGER
DIM x AS SINGLE, str AS STRING
PUBLIC pblobj AS DOUBLE
STATIC stob AS INTEGER
```

Jei kintamasis yra vartojamas prieš jo apibrezima, LotusScript apibrezia ji neišreikštiniu budu. Ši apibrezimo buda galima atšaukti operatoriumi OPTION DECLARE.

Konstantu apibrezimo operatorius:

[PUBLIC | PRIVATE] CONST <konstantos vardas> = <reiškinys>
{,<konstantos vardas> = <reiškinys>}

Jei galiojimo sritis praleista, vadinasi, PRIVATE. Pvz.:

```
CONST W = 123.45
CONST ManoAdresas$ = "Naugarduko 24"   '-tai simboline konstanta
```

Pradedant vykdyti skripta, kiekvienas, išreikštiniu budu apibreztas kintamasis yra inicializuojamas pradine reikšme, kuri priklauso nuo kintamojo tipo:

- INTEGER, LONG, SINGLE, DOUBLE, CURRENCY – nuliu;
- kintamo ilgio simboliu eilute – "", o fiksuoto ilgio – uzpildoma nuliais;
- VARIANT – EMPTY;
- nuoroda i klase – NOTHING;
- TYPE ir CLASS tipo elementu inicializacija priklauso nuo ju tipo;
- kiekvienas masyvo elementas – priklausomai nuo tipo;
- sarašas - be elementu (tuščias sarašas).

Zymes – vardai neilgesni nei 40 simboliu, rašomos prieš operatoriu ir atskiriamos nuo jo dvitaškiu. Eiluteje gali buti tik viena zyme. Zymes vartojamos valdymo perdavimui. Kiekvienas operatorius gali buti pazymetas keletu zymiu - tuomet jos turi buti skirtingose eilutese.

Masyvai. Tai bene dazniausiai vartojama duomenu struktura. Masyvu aprašu pavyzdziai:

DIM arr (1 TO 100) AS INTEGER

– 100 sveiku skaiciu masyvas. Kreipiantis i 2-a elementa rašoma arr(2);

PUBLIC matrix (1 TO 7, 1 TO 8)

- masyvo elementu tipas – VARIANT, kreipimosi pvz.: matrix(1, 2);

Masyvo indeksai gali buti ir neigiami.

Apibrezimas DIM array (2, 3) yra analogiškas vienam iš tokiu:

DIM array (0 TO 2, 0 TO 3) arba

DIM array (1 TO 2, 1 TO 3) – pradine indekso reikšme valdoma operatoriumi OPTION BASE:

```
OPTION BASE 0
OPTION BASE 1
```

Pagal nutylejima: OPTION BASE 0.

Draudziama apibrezti masyva uzimanti daugiau negu 64KB.

Kiekvienas masyvas gali buti fiksuoto dydzio arba kintamo. Visi ankstesni pvz. buvo

fiksuoto dydzio masyvai. Kintamo dydzio masyvu dydi galima keisti dinamiškai. Apibreziant toki masyva nereikia nurodyti reziu, pvz.:

DIM DynArr() AS INTEGER

Tokio apibrezimo rezultate neivyksta atminties išskyrimo. Bet prieš vartojima butina nurodyti rezius, pvz.:

REDIM DynArr (2, 3, 1 TO 3) – dabar galima kreiptis i jo elementus.

Veliau reziai gali buti keiciami, pvz.: REDIM DynArr (20, 30, 10), bet matavimu skaiciaus ir elementu tipo keisti negalima. Tokiu masyvu uzimama atminti galima atlaisvinti išreikštiniu budu, pavartojant operatoriu ERASE. Pavartojus ši operatoriu, galima pakeisti ir masyvo matavimus. Ši operatoriu galima pavartoti ir fiksuoto dydzio masyvams, taciau atmintis neatlaisvinama – ivyksta tik masyvo reinicializacija.

Sarašai. Sarašas – tai vienmatis duomenu apjungimas. Jis panašus i vienmati masyva, tik kiekvienas elementas pasiekiamas pagal unikalu varda. Elementai dinamiškai gali buti itraukiami i saraša ir šalinami iš jo. Pvz.

DIM Address LIST AS STRING – apibreziamas sarašas, kurio elementai – simboliu eilutes. Iš pradziu sarašas tuščias, bet po to i ji galima itraukti elementus, pvz.,

Address (“Jonas”) = “Kalvariju 20 - 30”

Address (“Petras”) = “Vilniaus 30 - 20”

Pavartojimo pvz.:

IF ISELEMENT(Address (“Jonas”)) THEN PRINT Address(“Jonas”)

ERASE Address (“Petras”) – pašalinti viena elementa;

ERASE Address – pašalinti visus sarašo elementus.

Kreipiantis i neegzistuojanti elementa, ivyksta klaida. Elemento vardas gali buti jautrus raidziu registrui arba ne – tai valdoma operatoriumi OPTION COMPARE.

LotusScript valdantieji operatoriai . Jiems priskirima:

- salyginiai operatoriai;
- ciklo operatorius;
- daugelio variantu išrinkimo operatoriai;
- salyginio ir besalyginio valdymo perdavimo operatoriai.

Salyginiai operatoriai:

IF <salyga> THEN [<operatoriai>] [ELSE [<operatoriai>]]

Jei rekia operatoriaus dali perkelti i kita eilute, eilutes gali reikia pavartoti pabraukimo zenkla. Pvz.,

IF a>b THEN max = a : min=b ELSE max = b : min = a

Galima ir kita forma, kuri leidzia patogiai išdestyti operatoriu keliose eilutese:

```
IF <salyga> THEN
    <operatoriai>
[ELSE
    <operatoriai>]
END IF
```

Pvz.:

```
IF a>b THEN
   max = a
   min = b
ELSE
   max = b
   min = a
END IF
```

Galimas variantas ir su raktiniu zodziu ELSE IF kuriu kiekis neribojamas.

Daugelio variantu išrinkimo operatorius.

```
SELECT CASE  <reiškinys>
        [ CASE <salyga>
                [<operatoriai>] ]
        [ CASE <salyga>
                [<operatoriai>] ]
```

….
[CASE ELSE
[<operatoriai>]]
END SELECT

Apskaiciavus reiškini, paeiliui tikrinamos salygos. Kai tik randama patenkinta salyga,- pradedama vykdyti atitinkama operatoriu seka, po to operatoriaus vykdymas nutraukiamas. šiame operatoriuje

<salyga> :: = <loginis reiškinys> | <reiškinys> TO <reiškinys> |
IS <palyginimo operacija> <reiškinys>

Pvz.:

```
DIM SelectValue AS DOUBLE
‘ Daugelio variantu išrinkimas…….
SELECT CASE SelectValue
    CASE 0          : Print “0”
    CASE 1          : Print “1”
    CASE 3 TO 5     : Print “3 to 5”
    CASE IS >= 6    : Print “>=6”
    CASE ELSE       : Print “Else”
END SELECT
```

Ciklo operatoriai: LotusSript turi keleta ciklo operatoriu:

FOR <kintamasis> = <reiškinys> TO <reškinys> [STEP <reiškinys>]
[<operatoriai>]
NEXT [<kintamasis>]

Zingsnis pagal nutylejima yra lygus 1, bet gali buti tiek teigiamas tiek ir neigiamas. Salygos tikrinimas priklauso nuo zingsnio zenklo, jei teigiamas – tikrinama salyga “maziau lygu” (<=), o jei neigiama - “daugiau lygu” (>=). Išejus iš ciklo, kintamojo reikšme išlieka. Iš ciklo galima išeiti ir operatoriumi GOTO, kai zyme yra ne ciklo kune. Cikla galima uzbaigti ir operatoriumi EXIT DO, perduodant valdyma operatoiui einanciam iš karto po operatoriaus NEXT.

Pvz.:

```
DIM i AS INTEGER, j AS INTEGER
FOR i = 1 TO 3
FOR j = 1 TO 2
   PRINT i
NEXT   ‘ next j
NEXT   ‘ next i
‘ Rezultatas: 1 1 2 2 3 3
```

Po raktinio zodzio NEXT išreikštiniu budu gali buti nurodytas ciklo skaitliuko vardas. Dalyje NEXT galima nurodyti ir keleta kintamuju, pvz.:

```
DIM i AS INTEGER, j AS INTEGER
FOR i = 1 TO 3
FOR j = 1 TO 2
    PRINT i
NEXT  j, i  ‘ next j, i
‘ Rezultatas: 1 1 2 2 3 3
```

Ciklas su nezinomu pasikartojimu skaiciumi:

DO [WHILE | UNTIL <salyga>]
[<operatoriai>]
LOOP

Šiame sakinyje salyga visuomet tikrinama pradzioje ciklo. Reiškinio reikšme 0 atitinka FALSE, o ne 0 - TRUE. Yra ir kitokia ciklo operatoriaus forma:

DO
[<operatoriai>]
LOOP [WHILE | UNTIL <salyga>]

Šiame operatoriuje salyga tikrinama ivykdzius sakinius.
Dar vienas ciklo operatorius:

```
WHILE <salyga>
   [ <operatoriai> ]
WEND
```

Iš tokio ciklo negalima “išeiti” su operatoriumi EXIT.

Norint atlikti operatorius su kiekvienu sudetinio tipo elementu patogu vartoti cikla:

```
FORALL <kintamasis> IN <masyvas arba sarašas>
   [ <operatoriai> ]
END FORALL
```

kur <kintamasis> – masyvo arba sarašo tipo kintamasis (papildomai jo apibrezti nereikia), kuris ciklo kune igis elemento reikšme. Šio kintamojo tipas toks pat kaip masyvo ar sarašo elemento. Pvz.:

```
DIM Array(100) AS DOUBLE
...
FORALL x IN Array
    IF x > 5 THEN x = 0
     ELSE  x = 10
     END IF
END FORALL

DIM ListObj LIST AS STRING
...
ListObj(“1”) = “Hello”
ListObj(“2”) = “,”
ListObj(“3”) = “ World”
ListObj(“4”) = “!”
FORALL y IN ListObj
   PRINT y
   y = “Hello, World !”
END FORALL
```

Ciklo FORALL vykdyma galima nutraukti ir operatoriais GOTO bei EXIT FORALL. Tokiame cikle negalima keisti masyvo reziu, t.y. negalima vartoti operatoriiaus REDIM.

Valdymo perdavimo operatoriai. Paprasciausias operatorius:

```
GOTO <zyme>
```

Tai lokalaus valdymo perdavimo operatorius, t.y. zyme turi buti toje pat funkcijoje ar paprograme je. Yra keletas salyginio valdymo perdavimo operatoriaus variantu:

```
IF <salyga> GOTO <zyme> [ ELSE <operatoriai> ]
```

Šis operatorius rašomas vienoje eiluteje. Jei reikia pernešti dali operatoriaus i kita eilute, reikia vartoti pabraukimo zenkla. Dazniausiai vartojami šio operatoriaus variantai:

```
IF <salyga> GOTO <zyme>
IF <salyga> GOTO <zyme> ELSE GOTO <zyme>
```

Gana patpgus gali buti dar vienas salyginio valdymo perdavimo operatorius:

```
ON <reiškinys> GOTO <zyme> {, <zyme> }
```

Šiame operatoriuje, reiškinys turi igyti reikšme, nedidesne nei 255. Jei šio reiškinio reikšme yra 1 – nukreipiama pirmaja zyme, jei 2 – antraja ir t.t.. Jei reikšme yra trupmenine – ji apvalinama, o jei lygi 0 ar didesne nei yra zymiu – operatorius neatlieka jokio veiksmo.

Visi valdymo perdavimo operatoriai yra lokalus - valdyma galima perduoti tik vienoje funkcijoje ar paprograme je.