Lotus Script (tęsinys)

Vartotojo apibėžiami tipai. Sąrašai ir masyvai yra sudėtiniai vienatipių duomenų apjungimo būdai. Norint apjungti skirtingų tipų reikšmes reikia operatoriumi TYPE … END TYPE, sukurti naują tipą. Pvz.

```
TYPE Address
   ID AS STRING * 6     ‘fiksuoto ilgio simbolių eilutė
   Country  AS STRING
   Town     AS STRING
   Street   AS STRING
   Building  AS STRING
   Room     AS INTEGER
END TYPE
```

Apibrėžus tipą, galima apibėžti šio tipo kintamuosius:

```
DIM ValueAddr     AS Address
DIM ArrAddr(10)  AS Address
```

Į tokio tipo laukus kreipiamasi pavartojant tašką. Kreipiamosi pvz.:

```
ValueAddr.ID = “111222”
ValueAddr.Country = “Lithuania”
ArrAddr(1).ID = “222333”
```

Vartotojo apibrėžti tipai toliau gali būti vartojami apibrėžiant naujus tipus, pvz.:

```
TYPE Student
   Name     AS STRING * 30
   Course   AS INTEGER
   StudAddresss    AS Address
END TYPE

DIM Stud AS Student
Stud.Name = “Jonaitis”
Stud.Course = 3
Stud.StudAddress.Town = “Vilnius”

DIM FirstCourse(300) AS Student
FORALL x AS FirstKurs
   x.Course = 1
END FORALL
```

Funkcijos ir paprogramės. Tai vardinė skripto dalis, atliekanti tam tikrus veiksmus ir į kurią galima kreiptis vardu iš kitos skripto dalies. Apibrėžiant funkciją ar paprogramę, reikia nurodyti jos vardą ir parametrų sąrašą, o taip pat grąžinamos reikšmės tipą. Jei grąžinamos reikšmės tipas nenurodytas, laikoma, kad jo tipas yra VARIANT. Pateiksime paprasčiausios funkcijos ir paprogramės apibrėžimus:

```
FUNCTION SimpleFunc ( x AS INTEGER, y AS INTEGER ) AS INTEGER
   SimpleFunc = x * y
END FUNCTION
SUB SimpleSub ( x AS INTEGER, y AS INTEGER)
  PRINT x*y
END SUB
```

Funkcijos iškvietimas:

```
PRINT SimpleFunc(3, 4)          ‘ Bus išvesta: 12
```

Paprogramės iškvietimas:

```
SimpleSub 5, 6                  ‘ Bus išvesta: 30
```

Pilnas formalaus parametro apibrėžimo pavidalas:

[BYVAL] <argumento vardas> [( ) | LIST] [AS <duomenų tipas>]

Nebūtinas parametras BYVAL nurodo parametro perdavimo būdą: pagal reikšmę ar pagal nuorodą. Pagal nutylėjimą, parametrai perduodami pagal nuorodą.

Jei parametru perduodamas masyvas, tai po parametro vardo turi būti nurodyta riestiniai skliaustai, o jei sąrašas – nurodoma raktinis žodis LIST. Parametro tipą galima nurodyti sufiksu arba, pagal nutylėjimą - VARIANT.

Tiek funkcija, tiek paprogramė gali ir neturėti parametrų, tuomet apibrėžime prie funkcijos vardo būtinai dedami skliaustai, o, paprogramės atveju, skliaustai gali būti ir praleisti.

Tiek funkcijas tiek ir paprogrames galima iškviesti ir operatoriumi CALL:

CALL <Funkcijos ar paprogramės vardas> ([<argumentų sąrašas>])

Lyginant šį operatorių su paprastu paprogramės ar funkcijos kvietimu, skiriasi tik faktinių parametrų užrašymas: paprogramės atveju vienu atveju dedami skliaustai, o kitu ne ir pan. Vidinės LotusScript funkcijos negali būti iškviestos CALL operatoriumi. Pvz.:

```
‘ Funkcijos apibrėžimas ir jos iškvietimas trimis būdais
FUNCTION MiniMult (x AS INTEGER, y AS INTEGER ) AS INTEGER
    MiniMult = x% ∗ y%
END FUNCTION
DIM result AS INTEGER
CALL MiniMult(3, 4)
MiniMult 3, 4
result% = MiniMult(3, 4)
PRINT result

‘Paprogramės apibrėžimas ir jos iškvietimas dviem būdais
SUB PrintProduct( a AS INTEGER, b AS INTEGER)
    PRINT a% * b%
END SUB
CALL PrintProduct(34, 5)
PrintProduct 34, 5
```

Parametrai, perduodami nuoroda, privalo tiksliai sutapti pagal tipą su nurodytu apibrėžime. Išimtį sudaro tik parametrai tipo VARIANT, kurie yra suderinami su bet kokiu tipu. Išraiškos ir konstantos visada perduodamos pagal reikšmę. Masyvai, sąrašai ir TYPE tipo kintamieji visada perduodami pagal nuorodą.

Parametras, funkcijos ar paprogramės apraše apibrėžtas perduoti nuoroda, kreipiantis, gali būti perduodamas ir reikšme – pakanka faktinį parametrą apskliausti, pvz.:

```
FUNCTION  Func ( BYVAL  x%, y % )
….
END FUNCTION
DIM a%, b%, res AS INTEGER
a% = 5 : b% = 78
res = Func(4, 5)                ‘ abu parametrai perduodami reikšme
res = Func(a%, b%)    ‘ a% perduodamas reikšme, o b% - nuoroda
res = Func(a%, (b%)) ‘ abu parametrai perduodami reikšme
```

Norint grąžinti funkcijos reikšmę, funkcijos kūne jos vardui reikia priskirti reikšmę. Grąžinamos reikšmės pavartojimas nėra būtinas. Gražinama reikšmė ignoruojama, kai kviečiama CALL operatoriumi.

Bet kuri funkcija gali iškviesti ir pačią save, t.y. galima rekursija:

```
FUNCTION Factorial (n AS INTEGER) AS LONG
   IF n <= 1 THEN
      Factorial = 1
   ELSE
      Factorial = n ∗Factorial(n - 1)
   END IF
END FUNCTION
```

LotusScript leidžia iškviesti išorines funkcijas, parašytas C programavimo kalba. Tokios funkcijos turi būti DLL (Dynamic Link Library) bibliotekose. Tai, žinoma, susiję su tam

tikromis subtilybėmis.

Savybės. Savybe LotusScript kalboje vadinama paprogramių pora, kurios gali būti vartojamos taip, lyg tai būtų paprastas kintamasis. Savybės apjungia kintamojo ir paprogramės savybes. Kaip kintamasis jos gali būti vartojamos reiškiniuose, o kaip paprogramės gali keisti bei grąžinti reikšmes. Savybė – tai sintaksinis susitarimas, t.y. vietoje skripto su savybėmis visuomet galima užrašyti skriptą be savybių. Pvz.,

```
SUB SetSubDir( todir AS STRING)
    CHDIR "c:\work\" + todir    'CHDIR - operatorius
END SUB
FUNCTION GetSubDir()
    GetSubDir=MID$(CURDIR$, 9, 255)    'MID$ – funkcija,
                                       'CURDIR$ – simbolių eilutė
END FUNCTION
```

Pavartojimas:

```
x$ = GetSubDir()
SetSubDir  "123"
SetSubDir  x$
```

Šios funkcija ir paprogramė gali būti pertvarkyti į savybę SubDir, pavartojant operatorius PROPERTY SET ir PROPERTY GET:

```
PROPERTY SET SubDir AS STRING
    CHDIR "c:\work\" + SubDir
END PROPERTY
PROPERTY GET SubDir AS STRING
    SubDir = MID$(CURDIR$, 9, 255)
END PROPERTY
```

Apibrėžtas savybes galima pavartoti taip:

```
x$ = SubDir
SubDir = "123"
SubDir = x$
```

Kaip matosi, jei savybė vartojama kairėje priskyrimo pusėje, tai iškviečiama PROPERTY SET, o jei dešinėje pusėje – PROPERTY GET.

Savybės yra labai patogios klasėse, kai reikia atlikti veiksmus susijusius su atributų reikšmių pakeitimu.

Kiekviena funkcija ar paprogramė turi būti apibrėžtos iki pirmojo jos pavartojimo. Jei jų aprašai yra žemiau tekste arba kitame modulyje, galima pavartoti išankstinį funkcijos paskelbimą be funkcijos kūno:

```
DECLARE [ STATIC | PUBLIC | PRIVATE ] <FUNCTION | SUB>
            <Vardas> [([Argumentų sąrašas])]  [AS <return Tipas>]
```

Išankstinis paskelbimas galimas tik modulyje arba klasėje, bet ne funkcijos ar paprogramės viduje.

Darbas su failais. LotusScript turi gana išvystytas darbo su failais priemones. Vartojami trys failų tipai:

- nuoseklus (Sequential) – tai praktiškai tekstinio failo ekvivalentas. Tokį failą galima papildyti į galą, rašyti ir skaityti nuo failo pradžios ir pan. Gali būti apdorojamas ir paprastu teksto redaktoriumi;
- tiesioginio kreipimosi (Random) – dažniausiai vartojamas darbui su struktūriniais duomenų tipais – gali būti apdorojamas tik LotusScript priemonėmis. Į tokį failą galima kreiptis įrašo numeriu;
- dvejetainis (Binary) – dirbama su baitais, galima pozicionuoti nuo bet kurio baito ir perskaityti norimą kiekį baitų.

Prieš dirbant su failu, jį reikia atidaryti. Atidarant suteikiamas numeris (nuo 1 iki 255), kuris naudojamas vėlesnėse operacijose. Numerį suteikia vartotojas. Atitinkamybė išsaugojama iki failo uždarymo.

Dialoginių langų vartojimas. LotusScript turi keletą funkcijų dialogui su vartotoju

organizuoti. Tai visų, pirma, INPUTBOX ir MESSAGEBOX

Objektinės LotusScript priemonės. LotusScript turi objektiškai orientuoto programavimo bruožų. LotusScript būdingos visos pagrindinės objektinio programavimo sąvokos:

- objektai (duomenų ir jais manipuliuojančių funkcijų apjungimas);
- paveldimumas;
- polimorfizmas;
- duomenų tipų abstrakcija.

Klasės. LotusScript turi sudėtinį tipą CLASS (klasė), kuris, kaip ir TYPE, leidžia apjungti skirtingų tipų duomenis. Klasėje apjungiama ne tik duomenys (nariai-duomenys) bet ir paprogramės bei funkcijos (nariai-metodai). Pvz.:

```
CLASS Customer
  PUBLIC Name         AS STRING
  PUBLIC Address      AS STRING
  PUBLIC Balance      AS CURRENCY
  SUB CheckOverdue
    IF Balance > 0 THEN
       PRINT "Overdue balance"
    END IF
  END SUB
END CLASS
```

Klasės nariai-duomenys apibrėžiami taip pat kaip ir paprasti kintamieji, tik be raktinio žodžio DIM. Nariai gali būti apibrėžiami raktiniu žodžiu PUBLIC (vieši) arba PRIVATE (privatūs, šis ir pagal nutylėjimą). PUBLIC nariai gali būti pasiekiami ir išorėje. Apibrėžimai su raktiniu žodžiu STATIC - negalimi. Metodai apibrėžiami taip pat, kaip paprogramės. Metodai, pagal nutylėjimą, apibrėžiami su raktiniu žodžiu PUBLIC, bet metodams apibrėžti galima naudoti ir PRIVATE. Vienu operatoriumi apibrėžiamas tik vienas narys.

Apibrėžus klasę, operatoriumi DIM galima apibrėžti kintamuosius, nurodančius į klasės objektus – pats objektas tuomet dar neegzistuoja ir kintamasis įgyja reikšmę NOTHING. Galimas ir nuorodų masyvas.

```
DIM x              AS Customer    ` paprastas kintamasis
DIM ArrX (23)      AS Customer    ` nuorodų masyvas
DIM ListX LIST     AS Customer    ` nuorodų sąrašas
```

Patys objektai kuriami vienu iš dviejų būdų:

- apibrėžiant nuorodos kintamąjį ir priskiriant jam nuorodą į objektą, naujai sukuriamą, pavartojant raktinį žodį NEW:
  DIM x AS NEW Customer
  Šiuo būdu negali būti sukurti objektų masyvo ar sąrašo.
- pavartojant raktinį žodį SET kartu su NEW, priskiriant nuorodą į naują objektą anksčiau apibrėžtam kintamajam – tokiu būdu gali būti kuriami ir objektų masyvai ir sąrašai (kiekvienas elementas kuriamas atskirai):
  SET x = NEW Customer
  SET ArrX(20) = NEW Customer
  SET ListX("Objektas") = NEW Customer

Tai vienintelis operatoriaus SET vartojimo variantas. Juo galima priskirti nuorodą ir į anksčiau sukurtą objektą (šiuo atveju operatoriuje SET nevartojame NEW). Sukūrus objektą kiekvienas narys inicializuojamas priklausomai nuo jo tipo.

Dinamiškai sukurti objektai, dinamiškai gali būti ir naikinami. Tai atliekama operatoriumi DELETE. Kai objektas pašalinamas visi kintamieji, nurodantys į jį įgyja reikšmę NOTHING. Objektas pašalinamas automatiškai jei nuorodų į jį skaičius tampa lygus 0.

Kai objektas yra sukurtas, galima kreiptis į jo PUBLIC elementus. Tai atliekama operacija ".". Pvz.:

```
DIM x AS NEW Customer
x.Name = "MaF"
x.Address = "Vilnius"
```

```
x.Balance = 14.92

DIM b AS CURRENCY
b = x.Balance
PRINT b; x.Balance
```

Panašiai ir su metodų vartojimu:

```
DIM Y AS Customer
SET Y = x
Y.CheckOverdue
PRINT x.Balance; Y.Balance
```

LotusScript yra raktinis žodis ME, kuris klasės apibrėžime leidžia vienareikšmiškai traktuoti klasės elementus, pvz. ME.Balance.

Objektinėse kalbose paprastai egzistuoja konstruktoriai ir destruktoriai, t.y. metodai kurie automatiškai iškviečiami sukuriant objektą ir sunaikinant jį. LotusSript - tai paprogramės (ne funkcijos) NEW ir DELETE. Neišreikštiniu būdu visi klasės kintamieji inicializuojami pagal jų tipą, bet galima tai atlikti išreikštiniu būdu pavartojant konstruktorių NEW. Pvz., klasės Customer apibrėžimas gali būti papildytas tokiais dviem metodais:

```
SUB NEW (argN AS STRING, argA AS STRING, argB AS CURRENCY)
    ME.Name = argN
    ME.Address = argA
    ME.Balance = argB
END SUB
SUB DELETE
    PRINT “Šalinamas objektas:”; ME.Name
END SUB
```

Vartojimo pvz.:

```
DIM x AS NEW Customer(“ChF”, “Vilnius”, 0.92)  `Kviečiamas konstruktorius
DIM y AS Customer
SET y = NEW Customer(“MaF”, “Vilnius”, 14.92)  `Kviečiamas konstruktorius
…
DELETE x                        `Kviečiamas destruktorius
SET y = NOTHING   `Kviečiamas destruktorius
```

Konstruktorius gali turėti argumentus, destruktorius – ne.

Klasių paveldimumas. Klasių paveldimumas leidžia kurti klasių hierarchiją. Tai realizuojama leidžiant išvesti naują klasę iš jau egzistuojančios. Apibrėžimas:

```
CLASS Base
…..
END CLASS
CLASS Derived AS Base
….
END CLASS
```

Čia, raktinis žodis AS nurodo, kad klasė išvedama iš jau egzistuojančios klasės. Kreipimasis į bazinės klasės narius vyksta taip lyg tai būtų išvestos kalsės metodai. Suprantama, naujoji klasė gali turėti naujus narius. Pvz.:

```
CLASS Point
        PUBLIC x AS INTEGER
        PUBLIC y AS INTEGER
END CLASS
CLASS ColorPoint AS Point
        PUBLIC color AS INTEGER
END CLASS
DIM objP AS NEW Point
objP.x = 2
objP.y = 22
DIM objCP AS NEW ColorPoint
```

```
objCP.x = 9
objCP.y = 22
objCP.color = 3
```

Klasių išvedimo gylis kalboje nėra apribotas.

Tiek bazinė tiek ir išvestinė klasės gali turėti savus konstruktorius ir destruktorius. Visi jie kviečiami griežta eilės tvarka. Paskutinės išvestos klasės konstruktorius NEW yra kviečiamas paskutinis. Destruktorių kvietimo tvarka yra atvirkšia konstruktoriams. Kadangi konstruktoriai gali turėti parametrus, tai bendras konstruktoriaus su parametrais pavidalas yra:

```
SUB NEW (arg1 AS Type1, arg2 AS Type2,…),  BaseClass( arg1, arg2,…)
```

Pvz.:

```
CLASS Base
        PUBLIC NameB AS STRING
        SUB NEW (nm AS STRING)
            NameB = nm
        END SUB
END CLASS
CLASS Derive AS Base
        PUBLIC NameD AS STRING
        SUB NEW (nm AS STRING, nm1 AS STRING), Base (nm)
            NameB = nm
        END SUB
END CLASS
DIM obj AS NEW Derive (“First Name”, “Second Name”)
PRINT obj.NameB     ‘ Išvedama: First Name
PRINT obj.NameD     ‘ Išvedama: Second Name
```

Išvestinėje klasėje gali būti perrašomi bazinės klasės metodai. Išvestinėje klasėje gali būti iškviečiamas perrašomas bazinės klasės metodas, tam vartojama dvitaškis:

```
CLASS Derive AS Base
        SUB Show
        …
        END SUB
        ….
        CALL Base..Show
END CLASS
```

LotusScript, kaip įprasta, leidžia nuorodai į bazinės klasės objektą priskirti nuorodą į išvestinės klasės objektą. Priklausomai nuot to, kokio tipo nuoroda, gali būti kviečiamas vienos ar kitos klasės metodas.

Objektas gali būti perduotas kaip parametras paprogramei ar funkcijai ar net kviečiant kokios nors klasės metodą. Objektas visada perduodamas nuoroda.

```
CLASS Base
    SUB Show
        PRINT “Show from Base class”
    END SUB
END CLASS
CLASS Derive AS Base
    SUB Show
        PRINT “Show from Derive class”
    END SUB
    SUBB ShowBase
        PRINT “Show  Base call from from Derive class”
        CALL Base..Show   ‘Bazinės klasės metodo iškvietimas
    END SUB
END CLASS
SUB Display (ArgBase AS Base)
        CALL ArgBase.Show
```

```
END SUB
DIM B AS NEW Base
DIM D AS NEW Derive
CALL Display(B) ‘ Kviečiamas bazinės klasės metodas
CALL Display(D) ‘ Kviečiamas išvestos klasės metodas
SET B = D
CALL Display(D) ‘ Kviečiamas išvestos klasės metodas
CALL Display(B) ‘ Kviečiamas išvestos klasės metodas
```

Darbo su objekto nariais supaprastinimui yra operatorius WITH:

```
WITH ObjectRef
   [<operatoriai>]       `operatoriai be taško
END WITH
```

Vidinės klasės. LotusNote v.4.1 yra 18 vidinių klasių, tai vartotojo interfeiso klasės ir “antrojo plano” klasės. Yra dvi interfeiso klasės: NotesUIWorkSpace ir NotesUIDocument (išrinktas dokumentas). “Antrojo plano” klasės pagrinde skirtos darbui su LotusNotes objektais (db, view, folder ir pan.). Klasių pavadinimai: NotesDatabase, NotesView, NoteViewColumn, NoteDocument ir pan.

Pvz.

- Mygtukui prikabintas skriptas:

```
Sub Click(Source AS Button) `Skriptas mygtukui
   Dim AS NotesDatabase                `Nuoroda į vidinės klasės objektą
   Set db = New NotesDatabase(“”,”leaning.nsf”) `Kuriamas vidinės klasės objektas
   Messagebox(db.title)          `Parodoma DB antraštė
End Sub
```

Paspaudus mygtuką, bus parodomas langelis su nurodytos duomenų bazės pavadinimu. Skriptas turi būti apibrėžtas įvykiui (event) Click.

- pašalinama duomenų bazė ByeBye.NSF, jei ji pastarasias 100 dienų nesikeitė

```
DIM db AS New NotesDatabase (“”, ”byebye.nsf”)
IF((TODAY - db.LastModify) > 100) THEN
   CALL db.Remove
END IF
```

- view dokumentų kiekio suradimas. Skriptas paleidžiamas iš agento, apskaičiuoja duomenų bazės, esančios faile ”leaning.nsf”, dokumentų, priklausančių view “By Author”, kiekį ir parodo jį pranešimų lange:

```
SUB Initialize
  DIM db     AS NotesDatabase
  DIM view  AS NotesView
  DIM doc    AS NotesDocument
  DIM count AS Integer
  SET db = NEW NotesDatabase(“”,”leaning.nsf”)
  SET view = db.GetView(“By Author”)
  SET doc = view.GetFirstDocument
  count = 0
  DO UNTIL doc IS NOTHING
    count = count + 1
    SET doc = view.GetNextDocument(doc)
  LOOP
  MESSAGEBOX(cstr(count))
END SUB
```

Klaidų apdorojimas. LotusScript skiria dviejų tipų klaidas: klaidos, aptinkamos kompiliavimo metu, ir klaidos, aptinkamos skripto vykdymo metu (Run-time error). Pastarosioms aptikti yra spec. priemonės.

Vykdymo metu įvykus klaidai, vykdymas pristabdomas, kol nebus atliktas klaidos

apdorojimas.

Klaidos apdorojimui yra eilė funkcijų ir kalbos operatorių:

Funkcija/operatorius	Pavadinimas
Funkcija ERL	Grąžina eilutės, kurioje įvyko klaida, numerį
Funkcija ERR	Grąžina klaidos numerį
Operatorius ERR	Nustato klaidos numerį
Funkcija ERROR	Grąžina atitinkamą klaidos pranešimo tekstą
Operatorius ERROR	Nustato atitinkamą klaidos pranešimo tekstą
Operatoriai ON ERROR ir RESUME	Grąžina valdymą nurodytam operatoriui

Sistemos faile LSERR.ERR yra visa eilė dažniausiai vartojamų konstantų ir pranešimų tekstų. Šį failą galima įtraukti į skriptą (direktyva %INCLUDE).

ON ERROR [ErrNumber] { GOTO Label | RESUME NEXT | GOTO 0 }

Šis operatorius leidžia perduoti valdymą nurodyta žyme, pereiti prie kito operatoriaus (laikoma, kad klaida apdorota), arba nurodyti, kad klaidos neapdoroti. Jei klaidos numeris nenurodytas – operatorius skirtas visams klaidoms.

Operatorių seka, apdorojanti klaidą turi pasibaigti operatoriais END SUB, END FUNCTION ar RESUME. Operatorius RESUME:

RESUME [0 | NEXT | label]

0 – grąžinti valdymą operatoriui, kur įvyko klaida (pakartotinai įvykdyti)

NEXT – tęsti vykdymą nuo kito operatoriaus (klaidingą operatorių - praleisti)

label – nukreipti vykdymą nurodyta žyme.