КАЗАН ФЕДЕРАЛЬ УНИВЕРСИТЕТЫ
ФИЛОЛОГИЯ ҺӘМ МӘДӘНИЯТАРА БАГЛАНЫШЛАР ИНСТИТУТЫ
Билингваль һәм цифрлы белем бирү кафедрасы

Ә.Ф. ГАЛИМҖАНОВ, Ә.И. ГАЛИМҖАНОВА

DELPHI НИГЕЗЛӘРЕ

Казан – 2022

УДК 004.4

Принято на заседании учебно-методической комиссии ИФМК
Протокол № 24 от 7 апреля 2022 года

Рецензенты:
кандидат физико-математических наук,
заведующий кафедрой информационных систем **Ф.М. Гафаров;**
кандидат педагогических наук,
декан ВШНКО ИФМК **Р.Х. Мирзагитов**

Галимянов А.Ф., Галимянова А.И.
Основы DELPHI / А.Ф. Галимянов, А.И. Галимянова – Казань:
Казан. ун-т, 2022. – 134 с.

Уку әсбабы урта мәктәп укучыларына һәм югары уку йортлары студентларына тәкъдим ителә. Әсбап DELPHI мохитендә эш һәм Object Pascal нигезләре белән таныштыра. Күп санлы мисаллар китерелә.

ЭЧТӘЛЕК

КЕРЕШ

Мәктәп “Информатика” курсында программалау башлыча PASCAL, PYTHON, C++ телләренә нигезләнеп алып барыла. Уку әсбабы PASCAL теле нигезләре, визуаль программалау, объектка ориентлашкан программалауның фундаменталь нигезләре һәм мөһим принциплары белән таныштыра. ООПның төшенчәләрен һәм ысулларын аңлатучы мисаллар Delphi мохитендә Object Pascal программалау теле кулланып эшләнгән. Pascal теле урта мәктәптә программалау нигезләрен өйрәтә башлаганда еш кулланыла, Object Pascal – ул Pascal теленең объектлы киңәйтелүе. Шуңа Object Pascal нең югары уку йортларында булачак педагогларны укыту барышында кулланылуы нигезле. Бу телнең хәзер (бик корректлы булмаса да) Delphi теле дип аталуын әйтеп китик, әсбапта укыту күчүчәнлеген саклау максатыннан Object Pascal исеме калдырылды. Delphi 1 эш мохите (Borland Delphi) 1995 елда Borland компаниясе тарафыннан эшләнелә. Аның соңгы версиясе булып Delphi 11 Alexandria тора, ул 2021 елдан кулланучыларга таныш, Embarcadero Technologies компаниясе чыгарган. Бу әсбаптагы мисаллар Delphi 7 мохите өчен эшләнгән, чөнки ул, 2002 елда чыкканнан башлап, иң еш кулланылучы уңышлы продукт булып тора.

Хәзерге вакытта укыту процессында Lazarus программ тәэминат эшләү мохите дә киң кулланыла, ул Object Pascal гә нигезләнгән.

Lazarusның формалар редакторы һәм объектлар инспекторы Delphi мохитенекенә охшаш.

1. ПРОГРАММ ТӘЭМИНАТ ТӨЗҮНЕҢ КАТЛАУЛЫЛЫГЫ

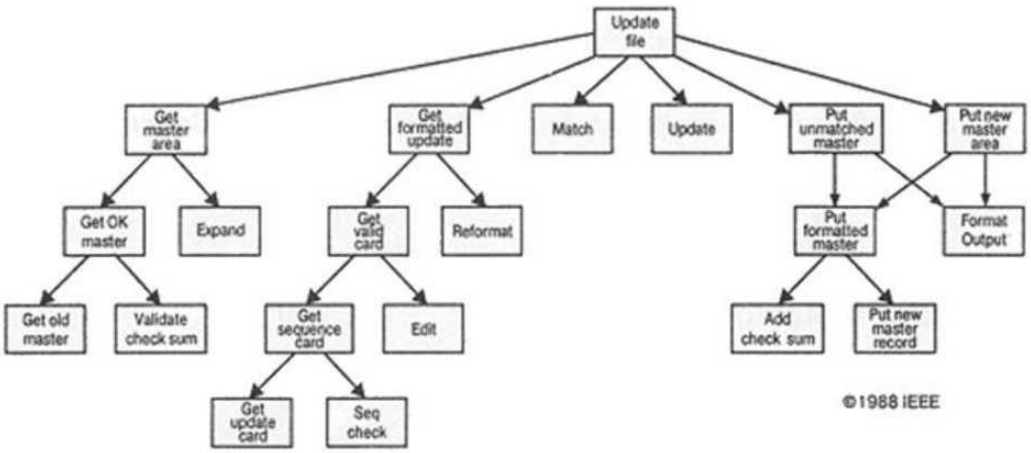
Без катлаулы системалар белән әйләндереп алынган. Мәсәлән, шәхси санак; теләсә нинди агач, чәчәк, хайван; атомнан алып йолдызлар һәм галактикаларга кадәр теләсә кайсы материя; җәмгыять институтлары – корпорацияләр, берләшмәләр – катлаулы система мисаллары булып торалар.

Күпчелек катлаулы системалар иерархияле структурага ия. Ләкин барлык программ тәэминат та катлаулы түгел. Бер үк кеше проектлый, төзи һәм куллана торган кушымталар классы бар. Тик аларны куллану өлкәсе чикләнгән. Корпоратив программ тәэминат төзегәндә, сәнәгать өчен программалауны караганда катлаулылык сораулары килеп чыга. Программ тәэминат төзү катлаулылыгының дүрт төп сәбәбе бар:

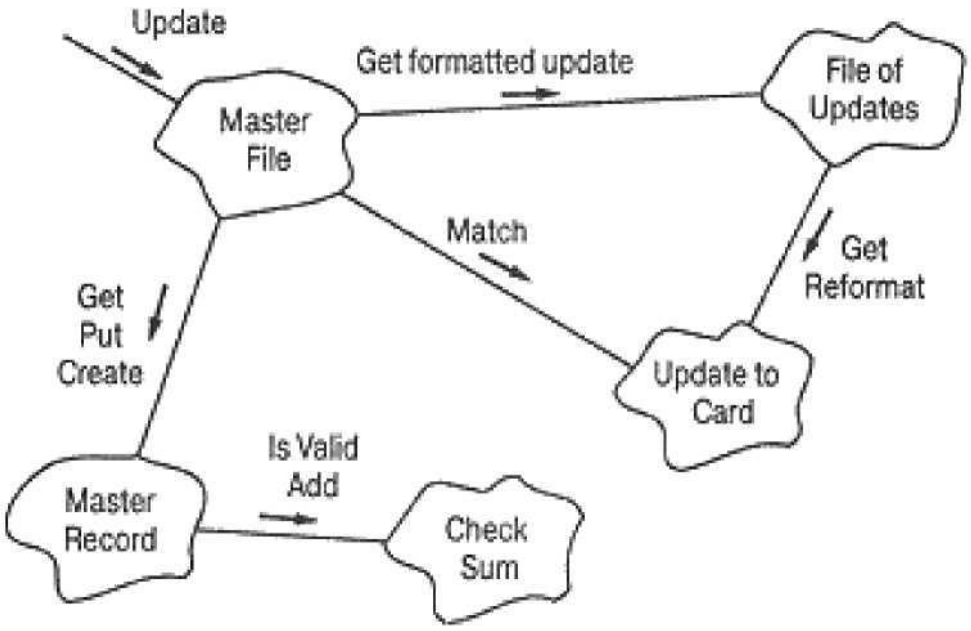
- программа төзүгә заказ чыккан предмет өлкәсенең катлаулылыгы;
- программа төзү белән идарә итүнең катлаулылыгы;
- программаның җитәрлек сыгылмалы булуы кирәклеге;
- зур дискрет системаларны тасвирлауның катлаулылыгы.

1.1. ДЕКОМПОЗИЦИЯ

Декомпозиция – катлаулылык белән көрәшнең бер ысулы.



1нче рәс. Алгоритмик декомпозиция мисалы



2 нче рәс. Объектка юнәлтелгән декомпозиция мисалы

Системаны бәйсез астсистемаларга бүлеп, һәрбersен аерым эшләү зарур. Декомпозициянең түбәндәге ысулларын билгелиләр:

- алгоритмик декомпозиция (1 нче рәсем);
- объектка юнәлтелгән декомпозиция (2 нче рәсем).

1.2. ПРОГРАММАЛАУ ТЕЛЛӘРЕНЕҢ КЫСКАЧА ТАРИХЫ

Югары дәрәҗәле программалау телләре үсешенең түбәндәге этапларын билгелиләр:

- Беренче буын телләре (1954-1958)

- FORTRAN 1 Математик формулалар
- ALGOL-58 Математик формулалар

- Икенче буын телләре (1959-1961)
  - FORTRAN II Астпрограммалар
  - ALGOL-60 Блоклы структура, мәгълүмат типлары
  - COBOL Мәгълүматны тасвирлау, файллар белән эш
  - LISP Исемлекләрне эшкәртү, күрсәткечләр, чүпне җыю

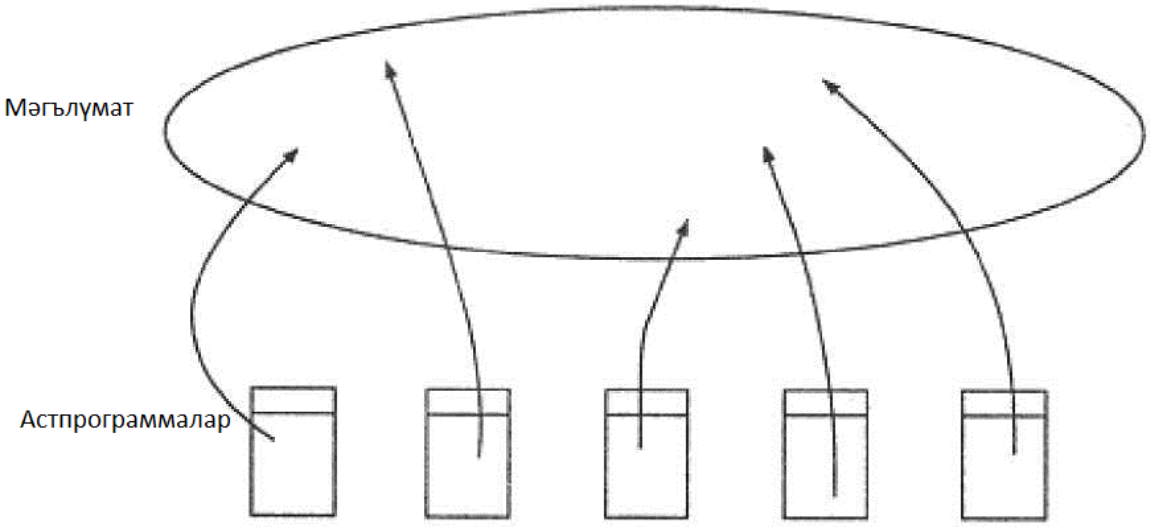
- Өченче буын телләре (1962-1970)
  - PL/I FORTRAN+ALGOL+COBOL
  - Pascal ALGOL-60 ның дәвам иттерүчесе, варисы
  - Simula Класслар, мәгълүматны абстракцияләү

- Бунынnан буынга күчүнең өзелүе (1970-1980)
  - C Эффектив, нәтиҗәле югары дәрәҗәле тел
  - FORTRAN 77 Блоклы структура, мәгълүмат типлары

- Объектка юнәлтелгән программалау үсеше (1980-1990)
  - Smalltalk 80 Чиста объектка юнәлтелгән программалау теле
  - C++ C + Simula
  - Ada83 Катгый типлаштыру; Pascalнең көчле йогынтысы

- Инфраструктуралар барлыкка килү (1990-...)
  - Java Блоклы структура, мәгълүмат типлары
  - Python Объектка юнәлтелгән сценарийлар теле
  - Visual C# Microsoft.NET мөхите өчен Java теленең конкуренты.

1 нче буын (3нче рәсем) башлыча фәнни һәм техник исәпләүләр өчен кулланыла, математик сүзлек. Телләр ассемблерның катлаулылыгыннан азат итәләр, бу санакның техник детальләренә игътибар итмәскә мөмкинлек бирә.

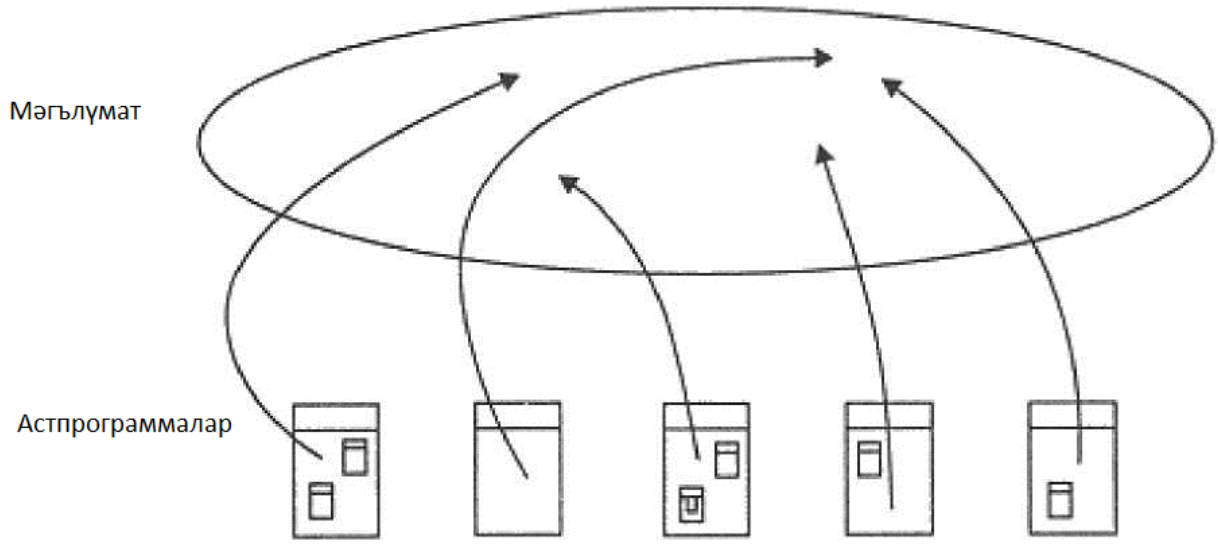


3нче рәс. Беренче буын телләре топологиясе

Беренче буын телләрендә язылган программалар мәгълүмат һәм астпрограммалардан торучы чагыштырмача гади структурага ия.

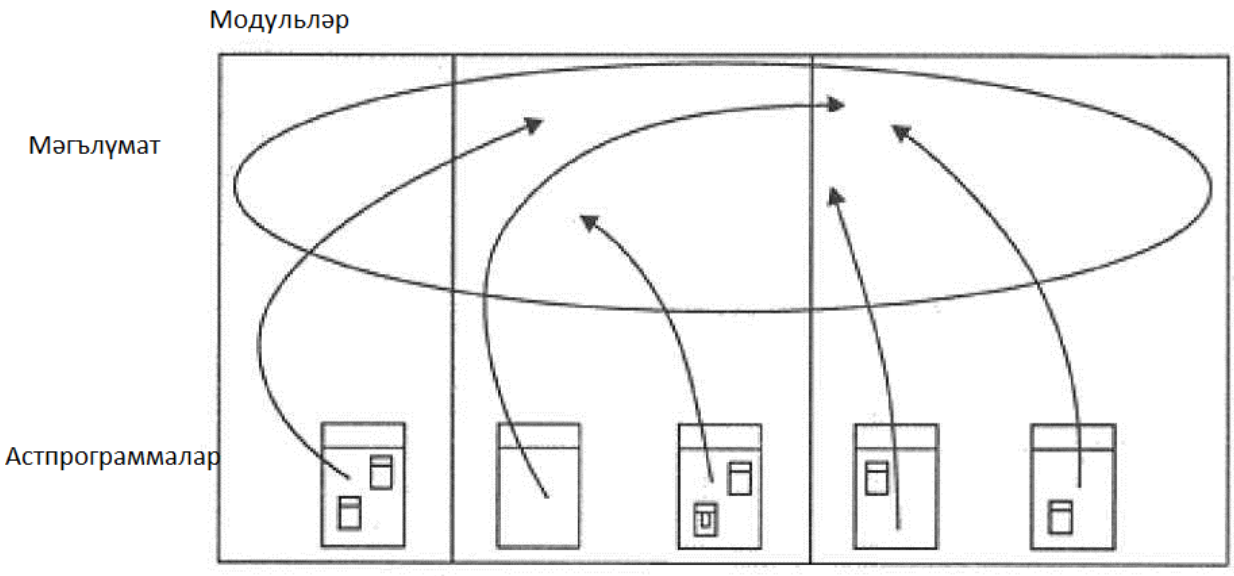
2 нче буын телләре (4 нче рәсем) алгоритмик абстракцияләргә басым ясый, бу үзлек программа эшләүчеләрне предмет өлкәсенә якынайта. Абстракт программ функцияләрне астпрограммалар рәвешендә тасвирларга мөмкинлек бирүче процедур абстракция барлыкка килә.

3 нче буын телләренә (5 нче рәсем) аппарат тәэминатның бәяләре кисәк кимүе, шул ук вакытта җитештерүчәнлеге экспоненциаль закон буенча артуы йогынты ясый. Телләр мәгълүматны абстракцияләргә мөмкинлек бирә һәм кулланучы үз мәгълүмат типларын тасвирлый ала.



4 нче рәс. Икенче буын телләре топологиясе

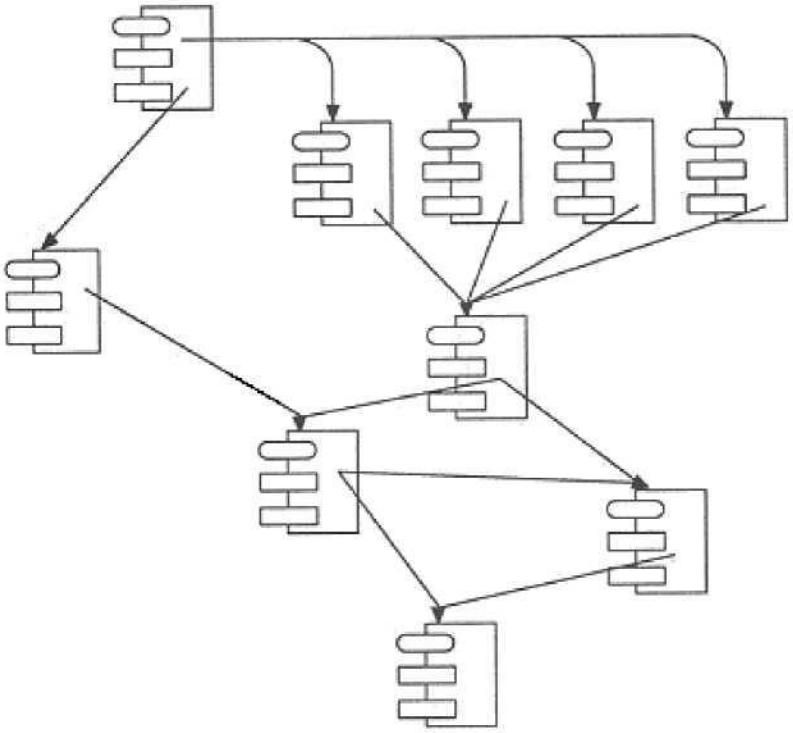
1970 нче елларда конкрет мәсьәләләрне чишү өчен берничә мең программалау теле эшләнелә, ләкин аларның барысы диярлек юкка чыга. Вакыт сынавын узган, хәзерге вакытта билгеле булган берничә тел генә кала.



5 нче рәс. Өченче буын телләре топологиясе

Модульләргә бергә үзгәртеләчәк астпрограммар җыела, ләкин аларны абстракцияләүнең яңа техникасы итеп карамыйлар.

1980 нче елларда объектка юнәлтелгән программалау көчле үсеш кичерә (6 нчы рәсем). Бу чор телләре программ тәэминатның объектка юнәлтелгән декомпозициясен башкарырга ярдәм итә. 90 нчы елларда зур күләмдә интеграллашкан сервислар тәкъдим итүче инфраструктуралар (J2EE, .NET) барлыкка килә.



6 нчы рәс. Объектка юнәлтелгән программалау топологиясе

Конструкциянең төп элементы булып астпрограмма түгел, логик яктан бәйле класслар һәм объектлардан торган модуль хезмәт итә.

1.3 . ПРОГРАММАЛАУНЫҢ ТӨП ПАРАДИГМАЛАРЫНА КЫСКАЧА КҮЗӘТҮ

Санак техникасының теоретик концепциясенә нигез салучы *фон-Нейман* концепциясе буенча, мәгълүматны эшкәртү өчен процессор мәгълүмат белән бер үк җәһәт хәтердә булган инструкцияләрне (командаларны, күрсәтмәләрне) үти.

Шул рәвешле, мәгълүматны эшкәртү процессының төп ике асылын аерып күрсәтеп була: инструкцияләр җыелмасы булган **код** һәм **мәгълүмат, бирелмәләр**. Сайлап алынган программалау технологиясенә бәйле рәвештә барлык программалар да концептуаль яктан үзләренең кодлары яки үзләренең мәгълүмат җыелмалары тирәсендә оештырылган.

Бүгенге көндәге төп программалау парадигмаларын карап үтик:

1) **Процесска юнәлтелгән парадигма**, программа бер-бер артлы үтәлүче операцияләрдән тора – фон-Нейман моделе. Бу очракта **код мәгълүматка тәэсир итә**. Бу парадигманы гамәлгә ашыручы телләр **процедур** яки **императив** дип атала. Мондый телләр мисалы булып С, Pasca1, һ.б. лар тора.

2) **Объектка юнәлтелгән парадигма**, бу очракта программа мәгълүматның аерым җыелмаларын – **объектларны** эшкәртүче код фрагментлары берлеге итеп карала. Мондый объектлар бер-берсе белән **интерфейс** аша тәэсир итешәләр. Бу очракта мәгълүмат кодның үтәлүе белән идарә итә.

Алгоритмның катлаулылыгы артканда процесска юнәлтелгән парадигма алдында җитди проблемалар туа. Программалауның объектлы принципларына күчү программаның эчке оештырылуын яхшыртырга мөмкинлек бирә, моның нәтиҗәсе булып программ комплекслар эшләү вакытында нәтиҗәлелек арту тора.

Югарыда әйтеп үтелгән төп ике парадигма белән беррәттән хәзерге вакытта тагын ике парадигма кулланыла:

3) **Аппликатив** яки **функциональ** парадигма. Бу ысулның төп идеясы булып программа үти торган функцияне формаль яктан билгеләү тора. Шул рәвешле, нәтиҗә алу өчен санак үтәргә тиешле халәтләр эзлеклелеге билгеләү урынына бирелгән мәгълүматка кулланганда кирәкле нәтиҗә

китереп чыгаручы функцияне билгеләргә кирәк:

$$\overline{y} = f(\overline{x})$$

Бу очракта программа язу бирелгән стандарт гади функцияләрдән катлаулы функция төзүгә кайтып кала:

$$\overline{y} = f_1(f_2(f_3(\dots), f_4(\dots), \dots))$$

Бу парадигманы кулланучы телләр булып, мәсәлән, LISP, Wolfram Mathematica һәм ML тора. Бу ысулны кулланганда мәгълүмат та, код та бертөрле структуралы тезмәләр белән күрсәтелә, димәк, программа, интерпретатор идарәсе астында эшләп, үз кодын мәгълүмат кебек эшкәртә ала. Бу очракта код белән мәгълүмат арасында аерма әкренләп югала. Шуңа күрә карала торган парадигманың мөһим куллану өлкәләреңең берсе булып **ясалма интеллект** системалары тора.

Искәрмә. Кодларны мәгълүмат эшкәрткән кебек эшкәртү процесска юнәлтелгән алымны кулланганда да мөмкин, ләкин алай булганда программалау түбән дәрәҗәле мөхиттә – ассемблер телендә башкарылырга тиеш.

4) **Кагыйдәләр системасын** куллануга нигезләнгән парадигма (**логик программалау парадигмасы**). Бу алымны кулланганда программаның операторлары язылган тәртиптә түгел, **рөхсәт итүче шартлар** (русча разрешающие условия – РУ) анализы нигезендә үтәләләр.

Бу парадигма очрагында программа парлар исемлегеннән тора:

$РУ_1 \rightarrow D_1$

$РУ_2 \rightarrow D_2$

……

$РУ_N \rightarrow D_N$

$РУ_1$, $РУ_2$, ...,$РУ_N$ рөхсәт итүче шартлары дөрес булганда тиешле D_1, D_2, … D_N гамәлләре үтәлә .

Программа эше рөхсәт итүче шартларны циклик рәвештә тикшерү һәм алар хак булганда тиешле гамәлләрне үтәүдән тора.

Логик программалау мисалы булып PROLOG теле тора.

Логик программалау вакытында программаның структурасы мәгълүматны эшкәртү алгоритмын алыштырып кую берлеге итеп күрсәтүче **Марков нормаль алгоритмнары** теоретик концепциясе белән концептуаль бәйләнгән:

$T_{11} \rightarrow T_{12}$

$T_{21} \rightarrow T_{22}$

……

$T_{N1} \rightarrow T_{N2}$

Рөхсәт итүче шартлар операторлары һәм алыштырып куюлар йомгаклаучы шарт табылганчы циклик рәвештә карап чыгыла.

Без бу курста объектка юнәлтелгән парадигмага бәйле сорауларны карарбыз.

1.4. ОБЪЕКТКА ЮНӘЛТЕЛГӘН ПРОГРАММАЛАУНЫҢ (ОЮП) ТӨП ПРИНЦИПЛАРЫ

ОЮПның үзәк идеясы булып **«абстракция»** төшенчәсен гамәлгә ашыру тора. Абстракциянең мәгънәсе шунда: теләсә нинди катлаулы *затны, төшенчәне* эчке төзелеше һәм эше детальләренә игътибар итмичә, бербөтен итеп карап, гамәлләрне дә аның өстендә шул рәвешле эшләп була.

Программ комплекс төзегәндә кирәкле абстракцияләрне билгеләү мөһим.

Мисал: Дәресләр расписаниесы төзү.

Кирәкле **абстракцияләр**: студент, лекцияләр курсы, укытучы, аудитория.

Операцияләр:

- Студентны группага билгеләү
- Группага аудитория билгеләү

Абстракцияләр билгеләүнең төп ысулларының берсе – **иерархияле классификация** концепциясен куллану. Аның асылы булып катлаулы системаларның гадиерәк фрагментларга (кисәкләргә) бүленүе тора.

Гомумән алганда барлык катлаулы системалар да иерархияле, аларның иерархия дәрәҗәсе абстракцияләрнең төрле дәрәҗәсен (биеклеген, күрсәткечен) чагылдыра. Һәр конкрет мәсьәләнең үз дәрәҗәсе карала. Абстракциянең иң түбән дәрәҗәсен сайлау шактый ирекле. Бер очракта иң түбән дип сайланган дәрәҗә башка проектта хәйран югары абстракция дәрәҗәсе булырга мөмкин.

Тип иерархиясе һәм **структура** иерархиясен аералар, без аларны **класслар структурасы** һәм **объектлар структурасы** дип атарбыз.

Барлык объектка юнәлтелгән программалау телләрендә түбәндәге **төп механизмнар** (постулатлар) реализацияләнгән:

- Инкапсуляция
- Мирас итеп алу
- Полиморфизм

Бу механизмнар абстракцияләрне билгеләү һәм куллану өчен мөһим.

1) **Инкапсуляция** – кодны һәм код тәэсир итә торган мәгълүматны бәйләүче, шул ук вакытта аларны бу код өчен тышкы (чит) булган код тарафыннан кирәк булмаган тәэсир итүдән саклаучы механизм. Код һәм мәгълүмат белән эш мөмкинлеге интерфейс тарафыннан катгый саклана, контрольләнә.

ОЮП кулланганда инкапсуляция нигезе булып класс тора.

Инкапсуляция механизмы классның кайбер детальләрен кулланучыдан яшерергә (капсула эченә куярга) мөмкинлек бирә, бу гамәл класс объектлары белән эшләүне җиңеләйтә.

2) **Мирас итеп алу** – бу механизм ярдәмендә бер объект (төзелгән классныкы, бала классныкы) икенче объектның (баба классныкын, база классыныкын) сыйфатларын үзенә ала. Мирас итеп алуны кулланганда яңа объектны өр яңадан тасвирлау мәҗбүри түгел, шуңа программлаучының эше күпкә җиңеләя. Мирас итеп алу объектка үзенең баба объекты (баба, база классыннан) атрибутларын (үзлекләрен, сыйфатларын) алырга, ә үзе өчен үзенә генә хас булган, уникаль характеристикаларны билгеләргә мөмкинлек бирә.

Мирас итеп алу иерархияле классификация концепциясенә туры килүче бик мөһим төшенчә.

3) **Полиморфизм** – бер үк интерфейсны гамәлләрнең гомуми классына куллану мөмкинчелеге бирә торган механизм.

Мисал: Саклау өчен 3 төрле стәк бар:

- бөтен саннар өчен
- йөзмә нокталы саннар өчен
- символлар өчен

Объектка юнәлтелгән программада өч идарә итүче астпрограмма урынына бер генә астпрограмма (бер интерфейс) кирәк булачак.

Полиморфизмның гомуми концепциясе: **бер интерфейс – күп метод, ысул.**

Конкрет очракка карата кирәкле ысулны (методны) сайлау компиляторга йөкләнә. Программалаучыга берничә урынына бер интерфейсны истә калдыру һәм куллану җитә, бу шулай ук эшне җиңеләйтә.

Статик (компиляция этабында функция һәм операцияләрне яңадан йөкләү ярдәмендә башкарыла), **динамик** (программа үтәлгән вакытта виртуаль функцияләр механизмы ярдәмендә башкарыла) һәм **параметрик** (компиляция этабында шаблоннар механизмын кулланып башкарыла) полиморфизм була.

Искәрмә. Каралган абстракция, инкапсуляция, мирас итеп алу, полиморфизм төшенчәләре ОЮП парадигмасына гына хас түгел. Бөтен саннар һәм йөзмә нокталы саннар өстендә арифметик гамәлләр процессорда төрле алгоритмнар буенча башкарыла. Бу очракта полиморфизм анык түгел, яшерен күренә.

Хәзер каралган һәм башка төшенчәләрне Delphi мөхите өчен Object Pascal теленә карата өйрәник.

2. DELPHI ДА ГАДИ КУШЫМТА ТӨЗҮ

2.1. ФОРМАЛАР БЕЛӘН ЭШ.

Формаларны проектлау – DELPHI мохитында визуаль төзүнең нигезе. Формага урнаштырылган һәр компонент яисә аңа бирелгән үзлек форманы тасвирлаучы фвйлда саклана (DFM – файл), шулай ук форма белән бәйле башлангыч текстка күпмедер тәэсир итә PAS – файл).

Яңа проектны буш форма ясап яисә инде булган формадан, төрле шаблоннар файдаланып ясап, яисә проектка яңа формалар өстәп була. Проект (кушымта) эченә теләсә ничә форма керә ала.

Форма белән эшләгәндә форманың үзенең үзлекләрен, аның компонентларының берсенең яисә компонентлар төркеменең үзлекләрен үзгәртеп була. Форма яисә компонентны сайлау өчен аңа тычкан белән чирттерергә яисә объектлар инспекторын (Object Selector) файдаланырга була, анда сайланган элементның исеме һәм тибы күрсәтелә. Берничә компонентны сайлау өчен Shift клавишасына басарга һәм компонентларга тычканның сул төймәсе белән чирттерергә кирәк.

Компонентның урынын күрсәтү өчен тычканнан тыш тагын ике ысул бар:

- Объектлар инспекторында Top һәм Left үзлекләре кыйммәтләрен кую
- Ctrl клавишасына баскан хәлдә курсор клавишаларын файдалану

Ctrl+курсор клавишасы методы элемент урынын төгәл кую өчен аерата уңайлы. Нәкъ шулай ук, Shift клавишасына баскан хәлдә курсор клавишаларына басып компонент үлчәмен көйләп була.

2.2. КОМПОНЕНТЛАР ПАЛИТРАСЫ

Агымдагы формага яңа компонент өстәү өчен аны компонентлар (Components) палитрасының битләренең берсеннән сайларга, формага чирттерегә кирәк. Сул төймәгә басып компонентны формага сөйрәп куеп та була, бу вакытта аны шундук урынына куярга мөмкин, ә компонетка һәм формага бер тапкыр чирттерү Delphi га аны килешү буенча үлчәмен куярга рөхсәт итә.

Палитраның һәр битендә пиктограмма һәм исем белән тамгаланган компонентлар бар. Бу исемнәр компонентларның рәсми исемнәре булып тора. Чынлыкта бу компонентларны тасвирлаучы класслар исемнәре (беренче T хәрефеннән тыш), мәсәлән, әгәр класс исеме Tbutton икән, компонент исеме Button булачак. Әгәр формага бер төрдәге берничә компонент урнаштырырга кирәк икән, палитрадагы компонент сайлаганда Shift клавишасына басып торырга кирәк. Аннары исә Delphi формасына һәр тапкыр чирттергәндә сайланган төрдәге яңа компонент куелачак. Бу гамәлне туктату өчен компонентлар палитрасының сул ягындагы стандарт селекторга (ук пиктограммасы) чирттерүҗитә.

2.3. ОБЪЕКТЛАР ИНСПЕКТОРЫ (OBJECT INSPECTOR)

Объектлар инспекторы форманы проектлаганда компонент (яисә форманың үзенекенең) үзлекләрен кую – билгеләү өчен файдаланыла. Аның тәрәзендә ике баганада сайланган элементның үзлекләре (яисә вакыйгалар) исемнәре һәм аларның кыйммәтләре күрсәтелә. Инспекторлар объектының өске өлешендәге тәрәз агымдагы компонент һәм аның тибын күрсәтә, бу селекторны агымдагы сайлауны үзгәртү өчен файдаланырга мөмкин.

Объектлар инспекторында компонентларның барлык үзлекләре дә түгел, ә проектлау барышында билгеләп була торганнары гына санап кителә. Объектлар инспеторының уң баганасы үзлекнең бирелгән тибына бәйле рәвештә дөрес билгеләү яисә төзәтүне тәэмин итә. Үзлеккә бәйле рәвештә юл яисә сан куырга, опцияләр исемлегеннән сайларга (бу мөмкинлекне ук күрсәтә) яисә махсус редакторны чакырырга мөмкин. Кайбер үзлекләр, мәсәлән, Color өчен, кыйммәт кертергә дә, исемлектән элемент сайларга да, махсус редактор чакырырга да мөмкин.

2.4. КОД ЯЗУ

Форманы төзегәч Delphi да вакыйгаларны эшкәртүче код язу таләп ителә. Формадагы яисә компоненттагы төймәгә баскач, Windows бу вакыйга турында кушымтага хәбәр җибәрә. Delphi реакциясе бу вакыйга турында хәбәр алудан һәм вакыйганы эшкәртүче тиешле методны чакырудан гыйбарәт. Компонентларның һәр төре өчен Delphi төрле вакыйгалар билгели. Форма яисә компонент өчен мөмкин компонентлар турында бу компонентны сайлагач объектлар инспекторының вакыйгалар (Events) битендә күрергә мөмкин.

Формага компонент – төймә өстик һәм аны аңа тычканның сул төймәсе белән басу белән бәйләнгән вакыйгага җавап бирерлек итик. Төймәгә тычкан белән чирттергәч эшләүче программада OnClick (чирттерү буенча) вакыйгасы барлыкка килә. Әлегә бу вакыйга программа тарафыннан берничек тә эшкәртелми, шуңа төймәгә басу бернинди нәтиҗәгә дә китерми. Программаны төймәгә басуга җавап бирдерү өчен Object Pascal телендә вакыйга эшкәртүче дип аталучы программа фрагменты язу зарур. Фрагмент махсус астпрограмма – процедура рәвешендә языла.

Delphi мөстәкыйль рәвештә OnClick вакыйга эшкәрткече өчен процедура әзерләмәсе язсын дисәк урнаштырылган компонентка икеләтә чирттерергә кирәк. Җавап итеп Delphi код тәрәзен активлаштыра һәм анда түбәндәгегә охшаган текст фрагменты пәйда була:

```
procedure TForm1.Button1Click(Sender: Tobject);
begin



end;
```

procedure сүзе компиляторга астпрограмма-процедура башлану турында хәбәр итә. Аннан соң процедура исеме TForm1.Button1Click бара. Бу – төзелмә исем: ул класс исеме TForm1 дән һәм прцедура исеме Button1Click тән тора.

Һәр компонент билгеле бер класска керә, ә компонентларның формага куелган барлык конкрет экземплярлары класс исеме + санлы индекстан торган исем ала. Delphi да кабул ителгән килешү буенча барлык класс исемнәре T хәрефеннән башлана. Шулай итеп, TForm1 исеме стандарт TForm классы стандарты үрнәге буенча төзелгән класс исемен белдерә. Әгәр код тәрәзендәге текст башына карасак, анда мондый юллар күренер:

```
type
    TForm1 = class(TForm)
        Button1: Tbutton;
        procedure TForm1.Button1Click(Sender: Tobject);
  private
    {private declarations}
  public
    {public declaration}
    end;
var
    Form1: TForm1;
```

TForm1 = **class**(TForm) юлы стандарт TForm классыннан тудырылган яңа TForm1 классын билгели. TForm стандарт классы буш Windows – тәрәзне билгеләсә, TForm1 классы аңа төймә компоненты куелган тәрәзне тасвирлый.

Һәр кабат вакыйга белән эшләгәндә Delphi форма белән бәйләнгән башлангыч файлны ача. Редактор башлангыч текст булган берничә файл белән берьюлы эшләү мөмкинлеген бирә, моны "ярлыклары күңел дәфтәре" образын файдаланып эшләп була. Күңел дәфтәренең һәр бите аерым файлга туры килә.

2.5. ПРОЕКТНЫ КОМПИЛЯЦИЯЛӘҮ

Проектны компиляцияләү аны эшләтеп җибәргәндә башкарыла. Ләкин Project сайлагында Build All командасы сайлана икән, үзгәрмәгән булса да һәр файл компиляцияләнәчәк.

Проектка проект өлешләре булып торучы файллар шулай ук аларга тиңдәш формалар (әгәр булсалар) керә. Башта башлангыч кодның һәр файлы Delphi ның компиляцияләнгән модуленә, ягъни исеме Паскаль телендәге файл исеме кебек, ләкин DCU өстәмәле файлга әверелдерелә. Проектны компиляцияләү һәм компоновка вакытында проектны төзүче компиляцияләнгән модульләр бер-берсе һәм VCL библиотекасы коды белән берләштерелә һәм башкарылучы файл барлыкка килә.

2.6. ИНТЕГРАЦИЯЛӘНГӘН КӨЙЛӘҮЧЕ

Delphi да мөмкинлекләре гаять зур булган эчке көйләүче бар. Delphi мохитыннан эшләтеп җибәрелгәндә программа көйләүчедә башкарыла. Тукталыш ноктaлары дип аталган ноктaлар бар. Аларга җиткәч программа тукталып тора һәм анда үзгәрешлеләрнең агымдагы кыйммәтләрен белергә һәм башка гамәлләр башкарырга була. Тукталыш ноктaларын кую өчен редакцияләү тәрәзенең сул кысасы һәм текст арасына чирттерергә, яисә Toggle Breakpoint командасын контекст сайлактан

сайларга, яисә F5 төймәсенә басарга була. Берничә тукалыш ноктасын билгеләгәннән соң Breakpoints List тәрәзен ачу өчен View сайлагының Breakpoints командасын сайларга була. Breakpoints List тәрәзенең өске өлешенең пунктларыннан берсе тукталыш ноктасына программа бары тик шул шарт үтәлгәндә генә башкарылырлык итеп шарт өстәүне күздә тота. SpeedBar линейкасындагы Step Over төймәсе операторлар үтәлешен берсен артыннан берсен карарга мөмкинлек бирә, ә Trace Into төймәсе чакырылучы методларны трассировкаларга, ягъни аспрограммалар кодын адымлап башкарырга мөмкинлек бирә.

Әгәр программа көйләүчедә тукталган икән, программаның бу ноктасында мөмкин булган теләсә кайсы идентификатор (үзгәрешле, объект, компонент һ.б.ш) кыйммәтен тикшереп була. Моның ике ысулы бар: Evaluate/Modify диалог аслыгын файдалану яисә элементныWatch List тәрәзенә өстәү. Evaluate/Modify диалог аслыгын ачуның иң гади ысулы – үзгәрешлене башлангыч текст редакторында тамгаларга, аннары редакторның контекст сайлагында Evaluate/Modify командасын сайларга. Шулай ук, редакторның контекст сайлагында Add Wath at Cursor командасын файдаланып, күзәтү элементларын куярга мөмкин.

2.7. HELP БЕЛЕШМӘ СИСТЕМАСЫНА МӨРӘҖӘГАТЬ ИТҮ

Белешмә системасын чакыру өчен Help сайлагында тиешле команданы сайларга яисә интерфейс элементын башлангыч текстта тамгаларга һәм F1 төймәсенә басарга кирәк. Help тәрәзендәге “Бүлекләр” төймәсенә баскач Windows белешмә системасының диалог тәрәзе пәйда була, ул Help төркеменең барлык файлларын карарга, ачкыч сүзне индексы буенча табарга яисә эзләү процессын яңадан башларга мөмкинлек бирә.

2.8. DELPHI САЙЛАГЫ ҺӘМ КОМАНДАЛАРЫ

Delphi мохитында команда бирү өчен өч төп ысулны файдаланырга мөмкин:

Сайлак ярдәмендә

SpeedBar тасмасы (инструменталь линейка) ярдәмендә

SpeedMenu ярдәмендә (тычканның уң төймәсенә басканда активлашучы локаль сайлакларның берсе)

File сайлагы

File сайлагы командаларын проектлар белән дә, башлангыч код файллары белән дә эш өчен файдаланырга мөмкин.

Проектлар белән эш командаларына New, New Application, Open, Reopen, Save Project As, Save All, Close All, Add to Project һәм Remove from Project командалары керә. Башлангыч код файллары белән New, New Form, New Data Module, Open, Reopen, Save As, Save, Close һәм Print командалары эшли. Төп команда булып File/New командасы тора, аны экспертлар чакыру, яңа кушымта белән эшли башлау, форманы инде булган формадан мирас итеп алу һ.б.ш. өчен файдаланырга мөмкин. Соңгы тапкыр эшләгән проект яисә башлангыч код файлын ачу өчен File/Reopen командасы файдаланыла.

Edit сайлагы

Edit сайлагының стандарт мөмкинлекләре текстка да, форма компонентларына да кулланылырга мөмкин. Редакторда теге яисә бу текстны күчермәләргә һәм өстәргә, бер формада компонентларны күчермәләргә һәм өстәргә, шулай ук компонентларны бер формадан

икенчесенә күчерергә/күчермәләргә мөмкин. Шулай ук компонентларны күчермәләргә һәм шул ук формадагы төркемле тәрәзгә, мәсәлән, панельгә яисә группа блогына куярга; компонентларны формадан редакторга һәм киресенчә куярга була. Delphi компонентларны текст тасвирламасына әверелдереп алмашу буферына урнаштыра. Бу текстны кирәкле рәвештә үзгәртергә һәм формага яңа компонент рәвешендә куярга мөмкин. Берничә компонент сайларга һәм аларны башка формага, шулай ук текст редакторына да күчермәләргә була. Бу берничә охшаш компонент белән эшләгәндә ярап куярга мөмкин. Бер компонентны редакторга күчермәләргә, аны тиешле санда күбәйтергә һәм аннары формага бөтен бер төркемне куярга була.

Search сайлагы

Әгәр Incremental Search командасы сайлана икән, эзләү үрнәге булган диалог тәрәзе чыгарыласы урынга Delphi редакторга күчә. Беренче хәрефне керткәч, редактор бу хәрефтән башланган беренче сүзгә күчә. Хәрефләр җыю дәвам ителсә, курсор эзлекле рәвештә башында кертелгән символлар торучы сүзләргә күчәчәк. Бу команда бик эффектив һәм тиз эшли. Browse Symbol командасы Object Browser ны – компиляцияләнгән программаны тикшергәндә күп кенә детальләрне каоау өчен файдаланырга мөмкин булган инструментны чакыра.

View сайлагы

View сайлагының командаларының күбесе Delphi мохитының теге яки бу тәрәзен, мәсәлән, Project Manager, Breakpoints List яки Components List тәрәзләрен күрсәтә. Бу тәрәзләр берсе белән берсе бәйләнмәгән. Toggle Form/Unit командасы агымдагы эштә бклган формадан аның башлангыч

кодына һәм киресенчә күчү өчен файдаланыла. New edit window командасы редакцияләү тәрәзе һәм аның эченең дубликатын төзи. Delphi да бу ике файлны рәттән карау өчен бердәнбер ысул, чөнки редактор берничә йөкләнгән файлны күрсәтү өчен ярлыклар куллана. Дубликациядән соң редакцияләү тәрәзләрендә төрле файллар булырга мөмкин. View сайлагының соңгы ике командасын экраннан SpeedBar тасмасын һәм компонентлар (Components) палитрасын алып ташлау өчен файдаланып була, тик бу вакытта Delphi мохиты уңайлырак була дип әйтеп булмый. Build All командасы проектның һәр башлангыч файлын, хәтта соңгы трансляциядән соң үзгәртелмәгән булсалар да, компиляцияләргә мөмкинлек бирә. Компиляцияләми генә язылган кодны тикшерү өчен Syntax Check командасын файдаланырга мөмкин. Information командасы соңгы тапкыр башкарылган компиляция турында җентеклерәк мәгълүмат алырга мөмкинлек бирә. Options командасы проект опцияләрен: компилятор һәм элемтәләр редакторы, кушымта объекты һ.б.ш. опцияләрне урнаштыру өчен файдаланыла.

Run сайлагы

Run сайлагын Debug (көйләү) дип атап булыр иде. Андагы командаларның күбесе, шул исәптән Run да көйләүгә карый. Delphi мохитында эшләтеп җибәрелгән программа, әгәр тиешле опция өзелмәгән булса, аның интеграцияләнгән көйләүчесендә башкарыла. Кушымтаны тиз эшләтеп җибәрү өчен F9 клавишасы файдаланыла. Калган командалар программаны көйләү яисә адымлап башкару, тукталыш ноктaларын кую, объект һәм үзгәрешлеләр кыйммәтләрен карау һ.б.ш өчен файдаланыла.

Component сайлагы

Component сайлагы командаларын компонентлар язу, аларны библиотекага өстәү, шулай ук библиотеканы яисә компонентлар палитрасын конфигурацияләү өчен кулланырга мөмкин.

Tools сайлагы

Tools сайлагы берничә тышкы программа һәм инструменталь чара исемен үз эченә ала. Tools командасы бу сайлакны конфигурацияләргә һәм аңа яңа тышкы чаралар өстәргә мөмкинлек бирә. Tools сайлагы шулай ук репозиторий көйләү өчен команданы һәм барлык Delphi эш мохитын конфигурацияләүче Options командасын үз эченә ала.

2.9. ТИЗ ЭШ ӨЧЕН SPEEDBAR ТӨЙМӘЛӘР ТАСМАСЫ

Delphi ның иң еш файдаланылучы командалары SpeedBar инструменталь линейкасында бар. SpeedBar үлчәмен аның белән компонентлар палитрасы арасындагы калын сызыкны сөйрәп күчереп үзгәртеп була. SpeedBar дагы башка рөхсәт ителгән гамәлләрне SpeedBar ның үз локаль сайлагының Configure командасы ярдәмендә пиктограммаларны өстәп үзгәртеп яисә бетереп үзгәртеп була. Бу команда SpeedBar Editor инструментын чакыра. SpeedBar га пиктограмма өстәү өчен аны кирәкле категориядән табарга һәм тасмага сөйрәп күчерергә кирәк. Нәкъ шулай ук пиктограмманы SpeedBar тышына чыгарып куярга яисә башка урынга күчерергә мөмкин.

2.10 ЛОКАЛЬ SPEEDMENU САЙЛАГЫ

Delphi ның сайлагында командалар бик күп булса да, төшелмә сайлакта барлык командалар да юк. Кайвакыт кайсыдыр тәрәзләр яисә тәрәз өлкәләре өчен SpeedMenu (локаль сайлак, контекст сайлак) кулланырга туры килә. SpeedMenu ны активлаштыру өчен кулланучы интерфейсының кирәкле элементына тычканның уң төймәсе белән чирттерергә яисә Alt һәм F10 төймәләренә басарга кирәк.

2.11. СИСТЕМА ТАРАФЫННАН ТӨЗЕЛҮЧЕ ФАЙЛЛАР

Яңа проект сакланганда, Delphi берничә файл төзи. Аларның иң мөһимнәре болар.

- Проектның төп проекты .DPR өстәмәле. Ул проектның башлангыч текстының төп модуле. Һәр проектның бер генә .DPR файлы була. Бу файлда шул исәптән проектны төзүче башка файллар исемнәре дә санап кителә.
- .DFM өстәмәле форма файлларары. Болар визуаль формаларны тасвирлаган икешәрле ресурс файллары. Проекта күп форма булырга мөмкин һәм һәрберсенең үз .DFM файлы була.
- .PAS өстәмәле Паскаль модульләре. Тиешле форма өчен Object Pascal кодын яисә автоном модуль кодын үз эченә ала.
- .OPT өстәмәле проект опцияләре файлы. Delphi ның төрле көйләнмәләрен үз эченә ала (текст файлы).
- .DCU тибындагы модульнең компиляцияләнгән файллары. .PAS-файлның объект кодын үз эченә ала.
- .EXE тибындагы компиляцияләнгән програм файллар. Бу инде Windows кушымтасы (программасы).
- Динамик библиотекаларның .DLL өстәмәле компиляцияләнгән файллары. Бу компиляцияләнгән

модульләр берьюлы берничә Windows программасы тарафыннан файдаланыла ала.

2.12. ОБЪЕКТЛАР РЕПОЗИТОРИЯСЕ БИТЛӘРЕ

Delphi да яңа форма, яңа кушымта, яңа мәгълүмат модуле, яңа компонент һ.б.ш. төзергә мөмкинлек бирүче берничә сайлак командасы бар. Бу командалар File сайлагында, шулай ук башка төшүче сайлакларда да бар. Ләкин алар урынына File/New командасы сайланса, Delphi Object Repository тәрәзен ачачак.

Репозиторий теләсә кайсы төрдәге: форма, кушымта, мәгълүмат модуле, библиотека, компонент һ.б.ш. яңа элемент төзү өчен файдаланыла. Object Repository диалог тәрәзе берничә биттән тора

- New бите күп төрле типтагы яңа элементлар төзергә мөмкинлек бирә.
- Агымдагы проект бите (чынлыкта бу бит ярлыгында проект исеме, мәсәлән, Project1 күренә) агымдагы проектка кертелгән аналогик объекттан форма яисә мәгълүмат модулен мирас итеп алырга мөмкинлек бирә.
- Forms, Dialogs , Data Modules битләре бу типтагы экспертлар яисә булган объектларны файдаланып яңа формалар, диалог панельләре һәм мәгълүмат модульләре төзергә мөмкинлек бирә
- Project бите репозиторийда сакланган проекттан файлларны күчермәләргә мөмкинлек бирә.

Object Repository диалог тәрәзенең аскы өлешендә өч радиотөймә бар, алар ярдәмендә булган элементны күчермәләүне, аны мирас итеп алуны яисә күчермәләмичә турыдан-туры куллануны күрсәтергә мөмкин.

New бите

Бу бит ярдәмендә төзеп булган элементлар исемлеге:

Application яңа буш проект төзи.

Data Module яңа буш мәгълүмат модуле төзи.

DLL яңа DLL библиотекасы төзи.

Form яңа буш форма төзи.

Text редакторда яңа текст файлы ача.

Unit форма белән бәйләнмәгән яңа буш модуль төзи.

Forms бите

Эш өчен кирәкле алдан билгеләнгән формалар исемлеге мондый:

About Box – “Программа турында” гади панеле.

Duil List Box – ике төрле исемлекне үз эченә алган форма: кулланучыга бер исемлектән элементларны сайлап алырга һәм аларны икенчесенә күчерергә мөмкинлек бирә. компонентлардан тыш бу форма Паскаль телендәге бик үк гади булмаган шактый күләмдәге кодка да ия.

Dialogs бите

Бу бит алдагысына охшаш, ләкин түбәндәге элементларны үз эченә ала:

Dialog Expert – гади эксперт, бер яки берничә битле төрле диалог панельләре генерацияли ала

Dialog with help – белешмә информация чакыручы төймәсе булган диалог панеленең ике варианты.

Password dialog – гади редакцияләү тәрәзе булган диалог панеле, серсүз кертү өчен опцияләре бар, код юк.

Standart Dialog Box – стандарт диалог панеле, ике вариантта төймәләрнең төрлечә урнашуларыбелән үзенчәлекле.

Data Modules бите

Мәгълүматлар модуле – форманың аерым рәвеше, ул башкарылу вакытында экранда күренми һәм визуаль булмаган компонентлар саклау өчен файдаланыла ала. Ул еш кына мәгълүматлар базасы белән эшләү өчен файдаланыла.

Projects бите

Бу биттә кушымта төзү өчен файдаланырга мөмкин булган проектлар схемалары бар.

Application Expert – гади эксперт, анда файл структурасын һәм кушымтаның кайбер башка элементларын сайларга мөмкин.

MDI Application Multiple Document Interface (MDI) интерфейслы программаның төп элементларын бирә. Бу кушымтада MDI-фреймның сайлак, халәт структурасы, инструменталь линейкаларны үз эченә алучы тәрәзе өчен төп форма билгеләнгән. Моннан тыш башкару вакытында бала тәрәзләр төзү өчен файдаланырга мөмкин булган икенче форма да бар.

SDI Application кулланучының заманча интерфейсының стандарт атрибутлары булган төп форманы билгели, анда инструменталь линейка, халәт юлы һәм типик About панеле бар.

2.13. DELPHI ЭКСПЕРТЛАРЫ

Delphi булган кодны күчермәләргә һәм файдаланырга гына түгел, экспертны кулланып яңа формалар, кушымталар яисә кодны үз эченә алган башка файллар төзергә дә мөмкинлек бирә. Эксперт бер төркем опцияләр

кертергә мөмкинлек бирә һәм ниндидер эчке схема нигезендә сезнең заказга туры килүче код төзи.

Application Expert

Аны Object Repository тәрәзенең Project битеннән активлаштырып була. Бу экспертның беренче бите программага File, Edit, Window һәм Help кебек төшүче стандарт сайлакларны өстәргә мөмкинлек бирә. Экспертның икенче битендә алар белән программа эшләргә мөмкин булган файлларның өстәмәләрен бирергә мөмкин. Бу файлларның тасвирламаларын кертергә кирәк булачак, мәсәлән, текст файлы (*.txt) һәм аның өстәмәсе txt. Бу кыйммәтләр Application Expert программага өстәгән File Open һәм File Save диалог тәрәзләре тарафыннан килешү буенча кыйммәтләр сыйфатында файдаланылачак.

Application Expert шулай ук экранга инструменталь линейка төзү өчен файдалаырга мөмкин булган визуаль чара чыгара. Анда төшүче сайлактан берсен сайларга мөмкин, һәм аның типик элементларына тиңдәш стандарт төймәләр пәйда була (моның өчен бу сайлакны экспертның беренче битендә сайлау зарур).

Инструменталь линейка белән эшне тәмамлагач, экспертның соңгы битенә күчәргә була. Биредә күп кенә өстәмә опцияләр куярга мөмкин, мәсәлән, MDI интерфейсын тәэмин итүне, халәт юлын өстәргә яисә калкучы ярдәмне рөхсәт итәргә мөмкин. Шулай ук яңа кушымта исемен бирергә һәм аның башлангыч файллары өчен каталогны күрсәтергә мөмкин. Кушымталар өчен каталог бу вакытта инде булырга тиеш. Әгәр дә проект файлларын яңа каталогка урнаштырырга кирәк икән, Browse төймәсенә басарга һәм пәда булган диалог тәрәзендә яңа юлны кертергә кирәк.

Dialog Box Expert

Бу демонстраөион мисал сыйфатында башлангыч тексты белән бергә китерелгән гади эксперт. Бу экспертны ике төрле диалог панельләре: гади һәм күп битле диалог панельләре төзү өчен файдаланырга була. Әгәр гади диалог панеле сайланса, эксперт төймәләр компоновкасын бирергә мөмкин булган өченче биткә күчәчәк. Әгәр күп битле панель сайланса, төрле ярлыклар өчен текстлар кертергә мөмкинлек бирүче арадаш бит пәйда була.

3. DELPHI ДА КУШЫМТА ТӨЗҮ. ВАКЫЙГАЛАР

Delphi да кушымта төзү ике этаптан тора.

Кушымта интерфейсын төзү

Кушымтаның функциональлеген билгеләү (код язу)

Кушымта интерфейсы кулланучы һәм кушымтаның үзара тәэсир итешү ысулын, ягъни кушымта башкарылганда форманың тышкы кыяфәтен һәм кулланучының кушымта белән ничек идарә итәчәген билгели. Интерфейс формада компонентлар урнаштырып төзелә. Кушымта фуенкөиональлеге исә билгеле бер вакыйгалар барлыкка килгәндә башкарылучы процедуралар белән билгеләнә.

Иң гади кушымта кушымта өчен барлык кирәкле нәрсәне тәэмин итүче каркас (әзерләмә) һәм Windows ның тулы канлы кушымтасы булып тора.

Windows операцион системасы анда барлыкка килгән вакыйгаларны эшкәртү принцибы буенча эшли, бу вакыйгалар, мәсәлән, төймәгә тычкан белән чирттерү, клавишага басу, тәрәз үлчәмнәрен үзгәртү һ.б.ш. булырга мөмкин. Гадәттә программа вакыйгалар турында хәбәр көтә һәм аны тиешле рәвештә эшкәртә. Хәбәрләр программа тарафыннан берьюлы түгел, ә эзлекле рәвештә эшкәртелә. Delphi мохитындагы

программа актив кушымта өчен нинди булса да вакыйга килеп чыкканда (форманы ачу, компонентка тычкан белән чирттерү, клавиатурадагы клавишага басу һ.б.ш.) башкарылырга тиешле алгоритмнарны тасвирлау кебек төзелә. Аларның иң киң кулланучылары мондый:

onClick – басу вакыйгасы, компонентка тычканның сул төймәсе белән чирттергәндә, яисә идарә элементына башка ысул белән басканда килеп чыгучы вакыйга;

onMouseDown – тычканның теләсә кайсы төймәсенә басканда килеп чыгучы вакыйга;

onMouseUp – тычкан төймәсен җибәргәндә килеп чыгучы вакыйга;

onDblClick – тычканның (сул) төймәсе белән ике тапкыр чирттергәндә (икеле чирттерү) килеп чыгучы вакыйга;

onKeyDown – клавиатурадагы клавишаны басканда килеп чыгучы вакыйга;

onKeyUp – клавиатурадагы клавишаны җибәргәндә килеп чыгучы вакыйга;

onCreate – форманы төзегәндә бер тапкыр килеп чыгучы вакыйга;

onClose – форманы япканда килеп чыгучы вакыйга

Вакыйга эшкәрткече объектлар инспекторының (Object Inspector) Вакыйгалар (Events) битендә яисә тиешле компонентка икеле чирттерү юлы белән сайлана.

Delphi да шулай ук консоль кушымтасы да төзергә мөмкин.

3.1. ГАДИ МИСАЛЛАР

Хәзер исә шушы мәгълүматны файдаланып, гади калькулятор төзик. Биредә китерелгән гамәлләр тәртибе барлык төзелүче кушымталар өчен стандарт, шуңа аларны истә калдыру зарур.

3.2. ЭШ ТӘРТИБЕ

1. Стандарт ысул (пиктограммага икеләтә чирттереп яисә төп сайлак аша) белән Delphi мохитына керергә.

2. Төп сайлакта File/New Application пунктын сайларга, бу вакытта яңа кушымта проекты төзелә.

3. File/Save Project As... командасын сайларга. Диалог тәрәзе пәйда була. Бу тәрәз ярдәмендә эш өстәлендәге (яисә дисктагы башка урындагы) алдан төзелгән папканы сайларга. Һәр проект өчен аерым папка төзү мәгъкуль.

4. Инспекторлар объектында Form1 компоненты исемен “Form1” дән “Гади кушымта” га үзгәртергә.

5. F9 клавишасына басып проектны эшләтеп җибәрергә.

6. Стандарт ябу төймәсенә басып кушымтаны ябарга.

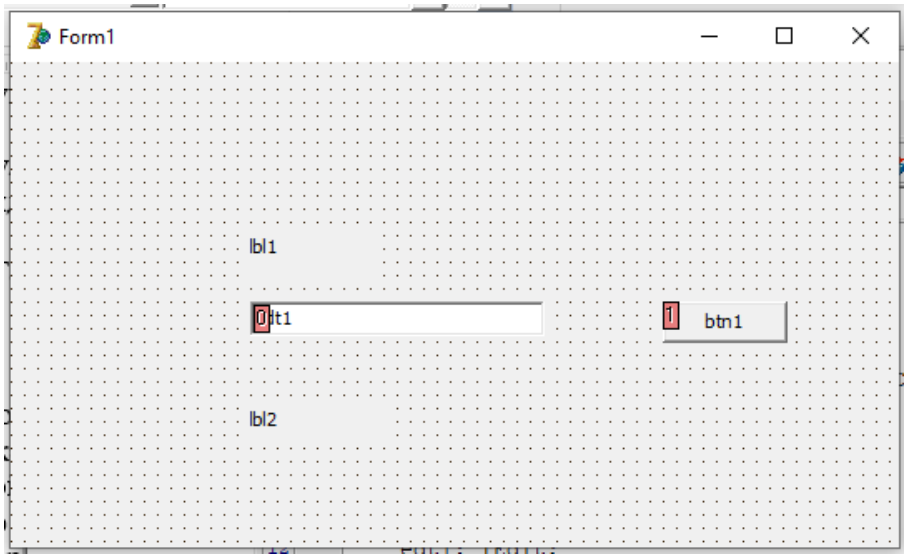
Бу кушымтада без форманың исемен генә алыштырдык. Фактта буш форма да Windows операцион системасының иң гади кушымтасы формасы икән. Аның кушымта өчен кирәкле барлык атрибутлары – җәю, җыю, ябу төймәләре бар, аны экранның төрле урынына күчереп була h.б. Тик ул берни дә эшләми. Хәзер исә нинди дә булса гамәл башкаручы гади кушымта төзеп карарбыз.

2.2. Теләсә нинди радиустагы түгәрәк мәйданын табу программасын төзергә.

2.2.1 Эш тәртибе

2.2. нче пункттагы 1-3 гамәлләрне кабатларга.

4.Формага Lbl1, Lbl2, Edit1 и Btn1 компонентларын куярга (1.1 нче рәс).. (Игътибар! DELPHI бу компонентларга башка исем дә бирергә мөмкин, ләкин алар шуларга охшаш булачак, бу көйләнмәләргә бәйле. DELPHI 2009 га кадәр версияләрдә UNICODE кодировкасын файдалану кыенлыкларга очрый, шуңа күрә без татар әлифбасына гына хас хәрефләрне файдаланмабыз.)



7 нче рәс.

5. Компонентларны формага куйганда шундук аларның үлчәмнәрен дә билгеләргә мөмкин. Моны компонентлар палитрасында компонентны сайлаганнан соң шундук формада эшләп була. Үлчәмне үзгәртү өчен куелган компонентның аскы уң почмагына чирттерү һәм төймәне баскон килеш кирәкле урынга тарттыру зарур. Шулай ук компонентка чирттереп сул төймәне баскан килеш компонентны тулаем башка урынга күчереп куеп була.Шуны ук форма белән дә башкарырга мөмкин. Компонентларның

исемнәрен компонентка чирттереп (тамгалап) һәм объектлар инспекторына Caption үзлегенә тиешле юл кертеп үзгәртеп була. Компонент үлчәмнәрен исә объектлар инспекторында Height (Биеклек) и Width (Киңлек) үзлекләренә кирәкле кыйммәтләр биреп үзгәртергә мөмкин. Left һәм Top үзлекләре компонент куелган контейнерның сул өске почмагыннан компонентның сул өске почмагын кадәр сулдан һәм өстән араны билгелиләр (пикселләрдә). Аларны, мәсәлән, болай куярга була:

	Lbl1	Lbl2
Height	57	Теләсә нинди сан
Width	129	Теләсә нинди сан
Left	131	100
Top	34	218

Бу кыйммәтләрне объектлар инспекторы ярдәмендә дә, тычкан ярдәмендә дә урнаштырырга мөмкин.

Lbl1 тамгасының Caption үзлегенә “Радиусны керегез, Санау га басыгыз” кыйммәтен кертәбез, ә Lbl2 тамгасының Caption үзлеге программа башкарылу вакытында билгеләнәчәк.

AutoSize үзлеге тамга үлчәме аңа урнаштырылган символ юлы күләме белән билгеләнәчәкме (true) яисә юкмы (false) икәнен билгели. WordWrap (Сүзләрне күчерү) исә юл тамгага сыймаса (AutoSize = False булганда) сүзләрне күчерергәме (true) яисә юкмы (false) икәнен билгели. Aligment (тигезләү) үзлеге исә текстның тамга эчендә сулгамы, урта буенчамы яисә уң як буенчамы икәнен билгел. Бу үзлекләрне, мәсәлән, болай куярга була:

	Lbl1	Lbl2

AutoSize	False	True
WordWrap	True	False
Aligment	taCenter	taLeftJustify

Шрифт характеристикаларын билгеләү өчен исә Объектлар инспекторында Font үзлеген сайларга һәм уң баганада пәйда булган өч ноктага чирттерергә кирәк. Экранда стандарт “Шрифт сайлау” тәрәзе күренәчәк. Бу тәрәз ярдәмендә теләсә нинди редактордагы кебек (мәсәлән, Word) шрифт характеристикаларын сайларга була. Бу тәрәз ярдәмендә, мәсәлән, тамгалар өчен мондый характеристикалар куярга була (төрле кыйммәтләр белән тәҗрибә ясап карагыз)

	Lbl1	Lbl2
Шрифт	Times New Roman	Arial
Сызылыш	Ярым калын	Курсив
Үлчәм	10	11

6. Объектлар инспекторында Edt1 һәм Btn1 компонентлары өчен, мәсәлән, мондый кыйммәтләр куярга була:

	Edt1	Btn1
Height	21	25
Width	193	75
Left	31	285
Top	146	146

Text үзлеге Edt1 компоненты өчен төп үзлек булып тора һәм символ юлларын кертү (яиә чыгару) өчен билгеләнә. Кушымта эшли башлаганда ул

буш булырга тиеш. Шуңа объектлар инспекторында бу үзлек кыйммәте сыйфатында буш юл билгелибез.

Btn1 компонентының Caption үзлеге кыйммәте сыйфатында “Санау” символ юлын кертәбез.

7. Кушымтада (программада) файдаланылучы барлык компонентларның исемнәре бар, алар Name (исем) үзлегендә бирелә. Компонентлар исемнәре автомат рәвештә DELHI тарафыннан компонент төзелгәндә, мәсәлән, компонентны формага урнаштыргана билгеләнә. Компонент исеме сыйфатында класс исеме (T хәрефеннән башка) файдаланыла, исем ахырында компонентның тәртип номеры куела (шушы класстагы компонент төзелү тәртип номеры).

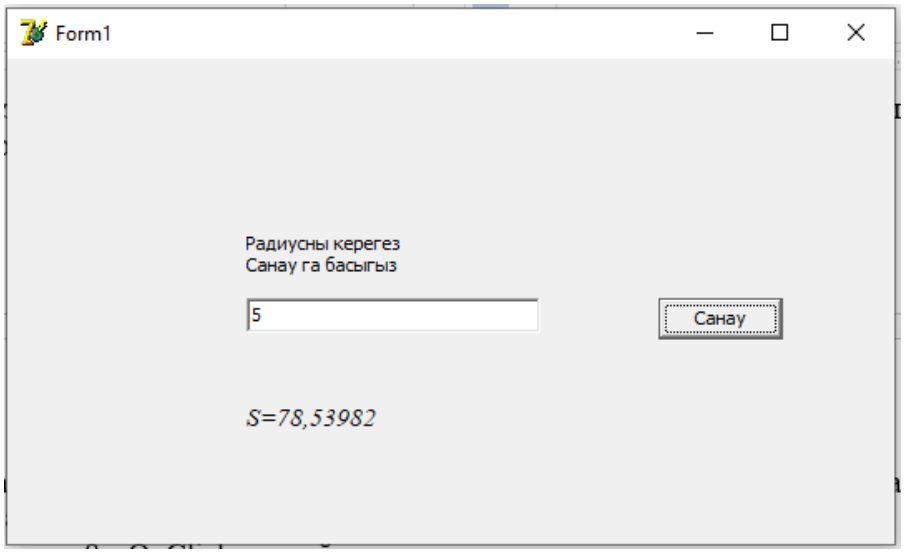
8. OnClick вакыйгасы эшкәрткече әзерләмәсен төзү өчен Btn1 төймәсенә ике тапкыр чирттерергә кирәк (икеле чирттерү). Нәтиҗәдә код редакторы тәрәзе активлашачак һәм анда вакыйга эшкәрткече әзерләмәсе булачак.

Әзерләмәдә түгәрәк мәйданы исәпләү өчен кодны язып бетерәбез:

```
procedure TForm1.btn1Click(Sender: TObject);
 var r,s:real;
begin
 r:=StrToFloat(Edt1.Text);
s:=pi*sqr(r);
Lbl2.Caption:='S=' + FloatToStrF(s,ffGeneral,7,2);
end;
```

9. Гадәттәгечә, программаны сакылыйбыз (төп сайлактан File–Save All).

10. Яшел өчпочмакка басып, программаны эшләтеп җибәрәбез. Экранда (хата булмаса) кушымта тәрәзе пәйда булачак:



Код редакторында без бары вакыйганы эшкәртүче процедураны гына төзедек. Редактор ярдәмендә өскә менеп һәм аска төшеп карасак, без анда инде система тарафыннан язылган программа өлешлшрен, төгәлрәк әйткәндә, форма белән бәйләнгән модуль текстын күрербез.Кыскача гына модуль төзелешенә күзәтү ясап китик.

4. DELPHI ДА МОДУЛЬ СТРУКТУРАСЫ

Модуль структурасы мондый:

unit Модуль; {*модуль исеме*}

{**********}**

interface {*Интерфейс өлешен тасвирлау*}

{**Бүлекләр бу модульне кулланучы модульләргә дә күренә:**}

uses {*модульләр бүлеге – программада файдаланылучы модульләр* }

Мод1, МодN; {*Файдаланылучы модульләр исемнәре: «Мод1», «МодN»*}

Const {*Константалар бүлеге*}

Конст1 = Кыймм1; { *«Кыймм1» кыйммәтен «Конст1» константасына үзләштерү*}

Type {*Типлар бүлеге – файдаланылучы типлар* }

Тип1 = тасв1; { *«Тип1» исемле типны тасвирлау*}

var {*Үзгәрешлеләр бүлеге – файдаланылучы үзгәрешлеләр* }

Үзг1 : Тип1; { *«Тип1» типлы «Үзг1» үзгәрешлесен тасвирлау*}

{**********}**

implementation {*модульне тасвирлау бүлеге* }

{**Бүлекләр модуль эчендә глобаль күренә:**}

Uses {*Модульләр бүлеге – программада файдаланылучы модульләр* }

Мод2; {*Модуль эчендә «Мод2» модуле файдаланылда*}

Const {*Константалар бүлеге*}

Конст2 = Кыймм2; { *«Конст2 константасына «Кыймм2» кыйммәтен үзләштерү»*}

Label {*Тамгаларны тасвирлау бүлеге* }

Тамга11; {*«Тамга1» тамгасын тасвирлау*}

Type {*Типлар бүлеге – файдаланылучы типлар* }

Тип2 = Тасвир2; { *«Тип2» исемле типны тасвирлау*}

var {*Үзгәрешлеләр бүлеге – файдаланылучы үзгәрешлеләр* }

Үзг2 : тип2; { *«Тип2» типлы «Үзг2» үзгәрешлесе*}

{**********}**

procedure *Проц1*; { *«Проц1» исемле процедураны тасвирлау*}

{**Бүлекләр процедура эчендә локаль күренә:**}

Type {*Типлар бүлеге: файдаланылучы типлар* }

Тип3 = Тасвир3;{ *«Тип3» исемле типны тасвирлау*}

var { *Үзгәрешлеләр бүлеге – файдаланылучы үзгәрешлеләр* }

Үзг3 : Тип3; {*«Тип3» типлы «Үзг3» үзгәрешлесе*}

Begin {*«Проц1» процедурасы башы* }

Гамәл1; {« *Гамәл1» гамәлен башкаручы «Проц1», җисеме*}

end; {*«Проц1» процедурсы ахыры*}

end. { *«Модуль» тасвирламасы ахыры*}

Шулай итеп, модульдә өч бүлек бар:

1. Бу модульне файдаланучы башка модульләр файдалана ала торган (ул модульләргә күренгән) типлар, константалар, процедуралар тасвирламасы;

2. Бу модульдә генә файдаланылучы типлар, константалар, процедуралар тасвирламасы (Глобаль үзгәрешлеләр бүлеге);

3. Үз типлары, константалары, үзгәрешлеләре (локаль тасвирламалар) белән процедураларның үзләрен тасвирлау.

Һәр оператор яисә операторлар төркеме өчен комментарий язып була. Комментарий фигуралы җәяләр эченә урнаша. Тулаем бер юлны комментарий итү өчен «икеле слэш» символларын кулланырга була. Әгәр инде программаның инде комментарийлары булган өлешен комментарий рәвешендә күрсәтергә кирәк икән, йолдыөзчык белән түгәрәк җәяләрне куллану зарур:

//бер юлны тулаем комментарий итү

{** текст өлешен комментарий итү**}

(Инде комментарий булган **{**текст өлешен**}** комментарий итү ***)***

Модульне язганда уку уңайлы булсын өчен конкрет бүлекләрне тасвирлаганда чигенешләр файдалану зарур., бу тиешле бүлекнең башын һәм ахырын күререгә мөмкинлек бирә.

4.1. DELPHI РЕДАКТОРЫНЫҢ КАЙБЕР КОМАНДАЛАРЫ

Белешмә алу:

<F1> – белешмә бирә (әгәр курсор стандарт операторда икән бу оператор буенча белешмә биреләчәк);

Тәрәзләр арасында күчеп йөрү:

<F12> – **Визуаль проект** һәм **Программа редакторы** тәрәзләре арасында күчеп йөрергә мөмкинлек бирә

<F11> – **Объектлар инспекторы** тәрәзен активлаштыра**;**

<F10> – **Delphi ның төп тәрәзе** н активлаштыра**.**

Текст буенча күчеп йөрү:

<↑> <↓> <→> <←> – курсорны тиешле юнәлештзә күчерү;

<PgUp>, <PgDn> – битне өскә; аска күчерү;

<Ctrl><PgUp>,

<Ctrl><PgDn> – текст башына, текст ахырына күчү.

Текст кертү һәм бетерү:

<Insert> – «өстәү»/»алыштыру» режимы, агымдагы режим «**Программа редакторы**» тәрәзенең астында күрсәтелгән – «*Insert*» (өстәү) яки «*Overwrite*» (алыштыру);

<Enter> – яңа юл өстәү;

<Delete>, <BackSpace> – курсор позициясендәге (Delete) яки курсордан сулдагы (BackSpace) бер символны бетерү;

<Alt><BackSpace> – текстны тергезү (*Мисал*, бетерелгән фрагмент).

Текст фрагменты белән эш:

<Shift><«уклар»>– текстны тамгалау (текст фоны төсе инвертлана);

<Ctrl><Insert> – фрагментны алмашу буферына урнаштыру;

<Shift><Delete> – текстны алмашу буферына күчереп кую (экранда бетерелә);

<Shift><Insert> – алмашу буферындагы текстны өстәү;

<Ctrl><K><H> – фрагментны тамгалауны бетерү;

<Ctrl><K><U> – фрагментны бер символ сулга күчерү;

<Ctrl><K><I> – фрагментны бер символ уңга күчерү.

Кыстыргычлар кую:

Текст зур күләмдә булса, текст буенча күчеп йөрүне җиңеләйтү өчен кыстыргычлар куярга була.

<Ctrl><K><«N»> - «N» нчы номерлы кыстыргыч кую (N – кыстыргыч номеры, юлның сул позициясендә бу номер пәйда була, (мәсәлән, <Ctrl><K><1> клавишлары комбинациясе))

<Ctrl><K><«N»> - «N» нчы номерлы кыстыргычка күчү.

Программаны компиляцияләү һәм эшләтеп җибәрү:

<Ctrl><F9> - программаны компитляцияләү («exe» өстәмәле файл төзү, ләкин аны эшләтеп җибәрмәү);

<F9> - программаны эшләтеп җибәрү.

5. DELPHI НЫҢ КОНСОЛЬ РЕЖИМЫ

DELPHI консоль режимында да эшли ала. Бу вакытта программа Паскальнең DOS режимнарына охшаш эшли. Кртү-чыгару, элекке версияләрдәге кебек үк read, readln, write, writeln процедуралары ярдәмендә башкарыла. Консоль кушымтасы төзү өчен түбәндәгеләрне башкару мәгъкуль:

1. Delphi ны Windows мохитында эшләтеп җибәрергә

2. Delphi төп сайлагында File–New командасын башкарырга.

3. Ачылган тәрәздә Console Application пунктын сайларга. Нәтиҗәдә бер файлдан торган яңа проект төзелә. Бу файл консоль кушымтасы булып тора да инде. Башта ул мондый рәвешле була:

```
program Project2;

{$APPTYPE CONSOLE}

uses SysUtils;

begin

// Insert user code here

end.
```

{$APPTYPE CONSOLE} директивасы компиляторга Delphi ның консоль режимында эшләвен белдерә. Программа коды begin ... end бүлегендә, ә тасвирлаулар тасвирлаулар бүлегендә языла.

6. OBJECT PASCAL ТУРЫНДА КЫСКАЧА БЕЛЕШМӘЛӘР

Мәгълүмат типлары. Object Pascal телендә берничә стандарт тип бар. Алар Паскаль стандартының стандарт типларын нигезендә төзелгән.: **Integer** – бөтен тип, бу типтагы үзгрешлелеләр *–32768* до *32767* диапазоныннан уңай һәм тискәре саннар булырга мөмкин; **Real** – реаль тип, бу типтагы үзгәрешлелеәр гади һәм экспоненциаль формада язылырга мөмкин булган уңай һәм тискәре кыйммәтләр ала ала: 10.12, 5.6 – гади форма, 2.123E5 (=$2.123*10^5$), 0.854E-3 (=$0.854*10^{-4}$) – экспоненциональ форма. Бу типтагы саннар диапазоны 2.9E-39 .. 1.7E38; **Char** –символ тибы, бу типтагы үзгәрешлеләр символлар кыйммәтләрен язу өчен файдаланыла, алар гадәттә туры апострофларга алына: ‘А’, ‘ю’, ‘7’ – үзгәрешле, мәсәлән, А, ю, 7 кыйммәтләре ала (аерым-аерым); **Boolean** – логик тип, бу типтагы үзгәрешлеләр бары ике кыйммәт: *true* (чын) һәм *false* (ялган) кыйммәтләре ала алалар; **Саналышлы тип** – үзгәрешленең мөмкин кыйммәтләрен күрсәтә, *Мисал*: 1, 5, 45, 56 – бу типтагы үзгәрешле санап кителгән

цифрларның берсен генә үзләштерә ала. Үзгәрешлеләрнең стандарт типлары нигезендә Object Pascal нең калган типлары төзелә. Object Pascal типлары түбәндәге таблицада китерелгән (стандарт типлар аерып күрсәтелгән).

Таблица 1

Бөтен саннар			
Мәгълүмат тибы		*Диапазон*	*Формат*
ShortInt		-128…127	Тамгалы, 8 бит
SmalInt		-32768…32767	Тамгалы, 16 бит
LongInt		-2147483648…2147483647	Тамгалы, 32 бит
Byte		0…255	Тамгасыз, 8 бит
Word		0…65535	Тамгасыз, 16 бит
Integer	*16 разряд*	–32768…32767	Тамгалы, 16 бит
Cardinal		0..65535	Тамгасыз, 16 бит
Integer	*32 разряд*	– 2147483648…2147483647	Тамгалы, 32 бит
Cardinal		0… 2147483647	Тамгасыз, 32 бит
Comp		$-9.2*10^{18}$ …$9.2*10^{18}$	Бөтен, абсолют төгәл
Реаль саннар			
Мәгълүмат тибы	*Диапазон*	*Цифрлар саны*	*Байтлар саны*

Real	$2.9*10^{-39}…1.7*10^{+38}$	11–12	6
Single	$1.5*10^{-45}…3.4*10^{+38}$	7–8	4
Double	$5.0*10^{-324}…1.7*10^{+308}$	15–16	8
Extended	$3.4*10^{-4932}…1.1*10^{+4932}$	19–20	10

Символ

Мәгълүмат тибы	*Символ*	*Кодлар*
AnsiChar	ANSI	0…255
WideChar	Unicode	0…65535
Char	ANSI	0...255

Юллар

Тип	*Юл тибы*	
String	0 до 255 символга кадәр статик озынлык	
ShortString		
LongString	Динамик	Озынлык хәтер күләменә бәйле
WideString		Шулай ук, ләкин һәрбер символ Unicode-символ (16 бит)

Логик

Тип	*Кыйммәт*
Boolean	True (чын), False (ялган)

Үзгәрешлеләр һәм константалар. Үзгәрешлеләрне тасвирлау өчен Var ачкыч сүзе белән башланучы үзгәрешлеләрне тасвирлау бүлеге хезмәт итә. Үзгәрешлеләр глобаль (программаның теләсә кайсы урыныннан күренүче) һәм локаль (үзләре тасвирланган процедура эчендә генә күренүче) була ала. Үзгәрешле болай игълан ителә:

исем : *тип*;

биредә *исем* – игълан ителүче үзгәрешле исеме, *тип* –Object Pascal типларының берсе исеме (стандарт яисә кулланучы төзегән тип) . *Мисал:*

var

a,b,c: real;

S : String;

Мисалда өч *real* типлы һәм бер *string* типлы үзгәрешлеләр игълан ителгән. Бер типлы үзгәрешлеләрне өтер аша санап китәргә мөмкин.

Искәрмә: *үзгәрешле исемендә бары тик латин хәрефләре, цифрлар һәм махсус символлар гына була ала, исем символлары регистрга бәйле түгел, ягъни s һәм S – бер үк үзгәрешле. Төрле типтагы үзгәрешлеләрне төрле юлларда тасвирлау уңайлы.*

Үзгәрешлегә кыйммәт биргәндә үзләштерү операторы – «:=» файдаланыла.

Гомуми очракта үзләштерү мондый рәвешле була:

Исем := Аңлатма;

Мисал: k := 5; –*k* үзгәрешәлесенә 5 кыйммәте бирү; c := 5 + t; – *c* үзгәрешлесенә *5 + t* кыйммәте бирү .

Паскальдә **константаларның** ике төре бар**:** *гади һәм исемле*. **Гади константа** – бөтен яисә вакланма сан, символлар юлы яисә аерым символ,

логик кыйммәт. **Исемле константалар,** үзгәрешлеләр кебек үк, файдаланылыр алдыннан константаларны тасвирлау (игълан итү) бүлегендә игълан ителергә тиешләр. Исемле константа гомуми рәвештә болай тасвирлана:

Константа = кыйммәт;

биредә *константа* – игълан ителүче константа исеме; *кыйммәт* – исемлек онстанта кыйммәте.

Мисал:

Const

Ch = 'S'; // Ch константасында S символы

V = 3; // V константасында санлы кыйммәт 3

Константа тибы аның кыйиммәте белән билгеләнә. Константа игълан иткәннән соң программа инструкцияләрендә константа кыйммәте урынына аның исемен файдаланырга мөмкин.

Үзгәрешлеләр белән гамәлләр. Аңлатма операторлар һәм операндлардан тора. Операндлар булып константалар, үзгәрешлеләр, функцияләр һәм башка аңлатмалар хезмәт итә ала. Операцияләр – гамәлләр операндлар белән башкарылучы гамәлләрне билгели. Түбәндәге таблицада операторлар, аларның җгамәлләре һәм табылган аңлатма тибы күрсәтелгән.

Таблица 2

Оператор	**Гамәл**	**Операндлар тибы**	**Аңлатма тибы**
+	Кушу	Бер операнд *real* барысы *integer*	*Real* *Integer*

-	Алу	Бер операнд *real* барысы *integer*	*Real* *Integer*
*	Тапкырлау	Бер операнд *real* барысы *integer*	*Real* *Integer*
/	Бүлү	*real* яки *integer*	*Real*
div	Бөтен бүлү	Һәрвакыт *integer*	*Integer*
mod	Бүлүдән калдык	Һәрвакыт *integer*	*Integer*

Логик операторлар

Оператор	**Гамәл**	**Операнд1**	**Операнд2**	**Результат**
and	Дизъюнкция («ҺӘМ»)	*False*	*false*	*false*
		False	*true*	*false*
		True	*false*	*false*
		True	*true*	*true*
or	Конъюнкция («ЯКИ»)	*False*	*false*	*false*
		True	*false*	*true*
		False	*true*	*true*
		True	*true*	*true*
Not	Инверсия (Инкарь)	*false*		*true*
		true		*false*

Оператор	**Мәгънәсе**	**Аңлатма кыйммәте**
>	Зуррак	*True*, әгәр беренче операнд икенчесеннән зуррак булса, булмаса *False*
<	Кечерәк	*True*, әгәр беренче операнд икенчесеннән кечерәк булса, булмаса *False*
=	Тигез	*True*, әгәр беренче операнд икенчесенә тигез булса, булмаса *False*
<>	Тигез түгел	*True*, әгәр беренче операнд икенчесенә тигез булмаса, тигез булса *False*
>=	Зуррак яки тигез	*True*, әгәр беренче операнд икенчесеннән зуррак яки тигез булса, булмаса *False*
<=	Кечерәк яки тигез	*True*, әгәр беренче операнд икенчесеннән кечерәк яки тигез булса, булмаса *False*

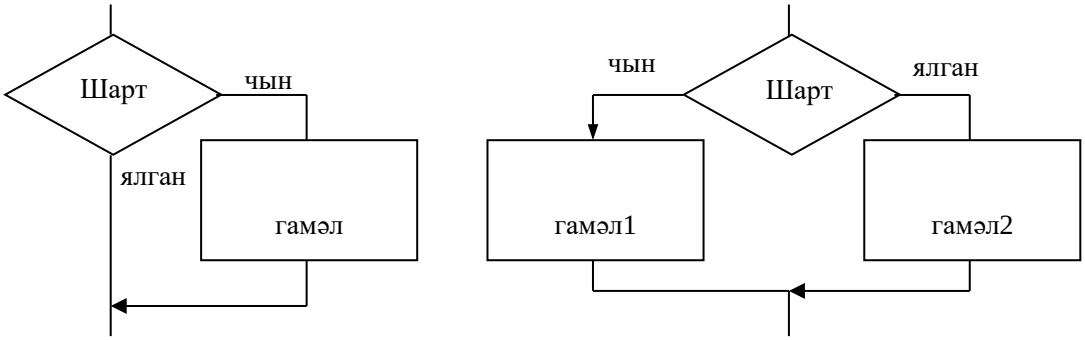
6.1. ШАРТ ОПЕРАТОРЛАРЫ.

Шарт операторы ике гамәлдән берсен сайларга мөмкинлек бирә, сайлау программа башкарылу вакытында бара. Шарт операторларының ике төре бар:

1. **If** шарт **then** гамәл;
2. **If** шарт **then** гамәл1 **else** гамәл2.

биредә *шарт* – логик типлы аңлатма; *гамәл1*, *гамәл2* – аерым операторлар яисә **begin** һәм **end** оператор җәяләре ярдәмендә берләштерелгән операторлар. Андый оператор төзелмә дип атала.

Блок-схема ярдәмендә шарт операторларын болай сурәтләп була:



Беренче тип оператор Икенче тпп оператор

8 нче рәс. Шарт операторының структур схемасы

Беренче тип операторында, әгәр шарт аңлатмасы **true** кыйммәте ала икән, **then** ачкыч сүзеннән соң килгән гамәл башкарыла. Әгәр дә шарт **false** булса, гамәл башкарылмый, шарт операторыннан соң килгән оператор эшли башлый. *Мисал:* **If** X > Y **then** X := 5; - X үзгәрешлесе бары тик X > Y булганда гына 5 кыйммәте алачак, булмаса X кыйммәте элеккечә калачак..

Икенче тип операторында, әгәр шарт **true** кыйммәте ала икән, **then** нән соң килгән *гамәл1* операторы үтәлә, шарт **false** кыйммәте алса, **else** сүзеннән соң килгән *гамәл2* операторы үтәлә.

Мисал: **If** X > Y **then** X := 5 **else** X := 3; - X үгәрешлесе кыйммәте X > Y булса 5 кыйммәте алачак, шарт үтәлмәсә X кыйммәте 3 кә тигез булачак.

Искәрмә 1: ***else*** *сүзе алдыннан нокталы өтер куелмый!*

Искәрмә 2: инде әйткәнебезчә, *шарт операторында* ***then*** *яки* ***else*** *сүзләреннән соң берничә оператор башкарырга кирәк икән, бу операторларны* ***begin*** *һәм* ***end*** *операторлар җәяләре арасына урнаштырырга кирәк.*

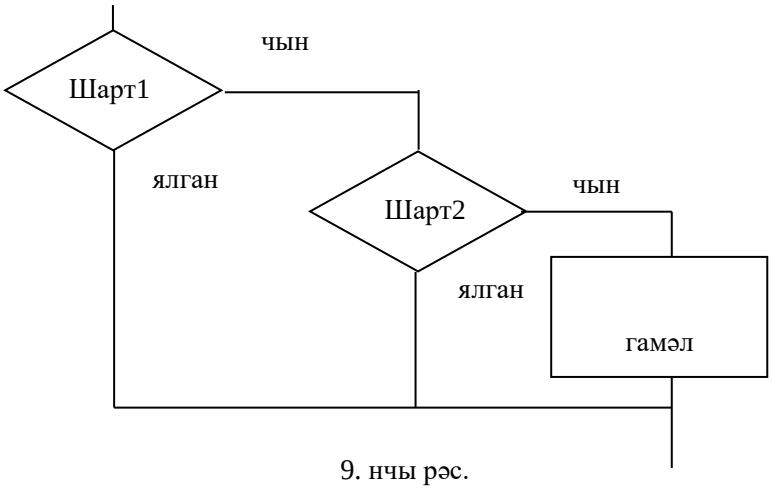
Мәсәлән, X һәм Y үзгәрешлеләре A>0 шарты үтәлгәндә тиңдәшле рәвештә 5 һәм 23 кыйммәтләре алырга һәм әгәр A<0 булса 0 кыйммәте алырга тиеш булсыннар. Шарт операторы мондый булачак:

```
If A > 0  then
begin
    X := 5;
    Y := 23;
end
else
begin
    X := 0;
    Y := 0;
end;
```

then яисә **else** сүзләреннән соң, үз чиратында шулай ук шарт операторлары торырга мөмкин.

1. Беренче тип шарт операторы - **If** *шарт* **then** *гамәл*

- *гамәл* беренче тип шарт операторы булырга мөмкин. (2.2 нче рәс.).



9. нчы рәс.

Ул вакытта болай булачак:

If *Шарт1* **then**

If *Шарт2* **then** *гамәл*

Бу очракта *гамәл* операторы бердәй билгеләнә.

- гамәл операторы икенче тип шарт операторы булырга мөмкин, ул вакытта мондый конструкция булачак:

If *Шарт1* **then**

If *Шарт2* **then** *гамәл1*

else *гамәл2*.

Кайсы **then** сүзенә **else** туры килә соң? (2.3 нче рәс.). Бер кыйммәтлекне саклау өчен Паскальдә һәр **else** сүзенә аннан алда торган ирекле **then** сүзе тора дигән килешү кабул ителгән**.**

2. Икенче тип шарт операторы – **If** Шарт **then** гамәл1 **else** гамәл2

- *гамәл1* беренче тип шарт операторы, *гамәл2* шарт операторы түгел (2.4 нче рәс.). Мондый конструкция табабыз:

If *Шарт1* **then**

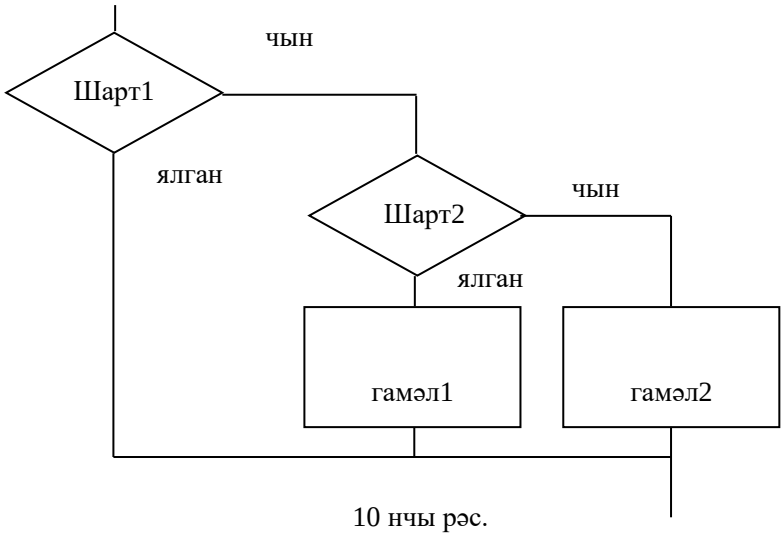
begin

If *Шарт2* **then** *гамәл1*

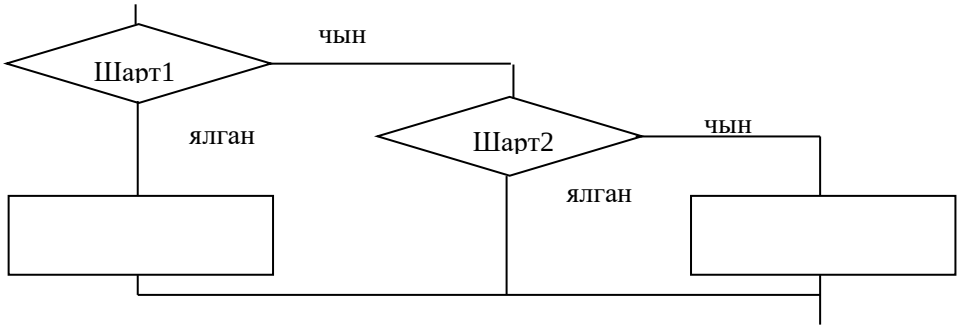
end

else *гамәл2*.

гамәл1 не оператор җәяләренә алырга кирәклеге ачык, юкса 10 нчы



10 нчы рәс.



11 нче рәс.

схема чыгачак.

Тәкъдимнәр:

1. *Программа текстын укуны җиңеләйтү өчен шарт операторларын язганда һәр бүлекне аерып күрсәтү өчен буш аралар куллану мәгъкуль.*
2. *Әгәр шарт операторында гамәлләр башкару тәртибе турында сорау туарга мөмкин икән, оператор җәяләрен куллану зарур.*

6.2. САЙЛАУ ОПЕРАТОРЫ.

Әгәр шарт операторын төзегәндә шартның мөмкин кыйммәтләренең күбесе билгеле икән, сайлау операторын файдалану зарур. Гомуми рәвештә аның структурасы мондый:

Case *үзгәрешле* **of**

Кыймм1 :*Гамәл1;*

…

КыймmN:*ГамәлN;*

else

ГамәлN1;

end;

биредә

Үзгәрешле – тәртип типлы үзгәрешле (*мәсәлән*: integer, char);

*Кыймм1, ... , КыймmN– Үзгәрешле*нең мөмкин кыйммәтләре;

*Гамәл1, ... , ГамәлN– Үзгәрешле*нең кыйммәтләре *Кыймм1, ... , КыймmN* кыйммәтләренең берсенә тигез булганда башкарылучы гамәлләр*;*

ГамәлN1 – Үзгәрешле санап кителгән (*Кыймм1, ... , КыймmN*) кыйммәтләрнең берсен дә алмаганда башкарылучы гамәл.

Операторның структур схемасы 12 нче рәсемдә күрсәтелгән.

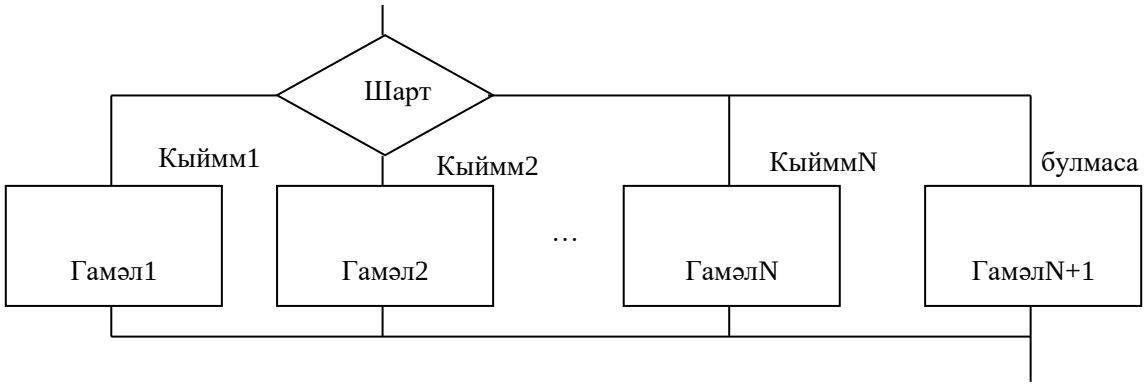
Нәкъ шарт операторында кебек үк, әгәр *Гамәл* берничә оператордан тора икән, аларны **begin** һәм **end** оператор җәяләренә алырга кирәк. Үзгәрешле санап кителгән кыйммәтләрне алмаганада башкарылучы гамәл булмаска да мөмкин, ул вакытта сайлау операторы схемасы мондый булачак:

Case *үзгәрешле* **of**

Кыймм1 : *Гамәл1;*

…

КыймmN : *ГамәлN;*



12 нче рәс. Сайлау операторы

end;

Искәрмә: ***Case*** *операторы* *<u>һәрвакыт</u>* ***end*** *белән тәмамлана.*

Мәсәлән, сайлау операторын клавиатура клавишларын басуны анализлаганда файдалану уңайлы.

*Мисал***:** Үзгәрешлеләрне тасвирлау бүлегендә Char типлы Ch үзгәрешлесе тасвирланган булсын.

Case Ch **of**

‘W’ : Y :=Y + 1;

‘X’ : Y :=Y – 1;

end;

Мисалда *Ch* үзгәрешлесе анализлана. Әгәр *Ch* та «W» символы булса, Y үзгәрешлесе 1 гә арттырыла; Әгәр *Ch* үзгәрешлесендә «X» символы булса, Y үзгәрешлесе 1 гәз кимителә.

6.3. СТАНДАРТ һӘМ МАТЕМАТИК ФУНКЦИЯЛӘР

Бу бүлек алдыннан еш кулланылучы стандарт функцияләрне карап китик.

Стандарт функцияләр. Еш башкарылучы исәпләүләр һәм үзгәртүләрне башкару өчен Object Pascal дә стандарт функцияләр бар.

Кайбер стандарт функцияләр 3 таблицада китерелгән.

Таблица 3

Стандарт функцияләр

Функция	**Гамәл**
Abs(n)	**n** ның абсолют кыйммәте
Sqr(n)	**n** санының квадраты
Sqrt(n)	**n** нан квадрат тамыр
Sin(n)	**n** ның синусы
Cos(n)	**n** ның косинусы
ArcTan(n)	**n** ның арктангенсы
Exp(n)	**n** ның экспонентасы (**e** санының **n** нчы дәрәҗәсе)
Ln(n)	**n** нан натураль логарифм

Катлаулырак исәпләүләр өчен программага **Math** модулен тоташтыру зарур. . Моның өчен модульне тасвирлау бүлегендә (мәсәлән, **implementation** сүзеннән соң) **Uses Math** юлын өстәргә була**;**

Бу модульнең кайбер функцияләре 4 нче таблицада китерелгән.

Таблица 4

Математик функцияләр

Функция	**Гамәл**
Арифметик	
Lnxp1(x)	(**x + 1)** санының натураль логарифмы (**x** саны 0 гә якын булганда кулланыла)
Ceil(x)	**x** ка аннан зуррак иң якын бөтен сан
Floor(x)	**x** ка аннан кечерәк иң якын бөтен сан
IntPower(a,x)	**a** ның x нчы бөтен дәрәҗәсе – $\mathbf{a^x}$
LdExp(x,p)	2 нең бөтен p нчы дәрәҗәсенә тапырланган **x** – $\mathbf{x*2^p}$
Log10(x)	**x** ның унарлы логарифмы **(Lg(x))**
Log2(x)	**x** ның нигезе 2 булган логарифмы **(Lb(x))**
LogN(n,x)	**x** ның нигезе n булган логарифмы $\mathbf{(Log_n x)}$
Max(a,b)	**a** һәм **b** саннарының иң зурысы
Min(a,b)	**a** һәм **b** саннарының иң кечкенәсе
Poly(x,C)	**C** (массив) коэффициентлары белән рәт– $\mathbf{c_0 + c_1x + c_2x^2 + \ldots + c_nx^n}$
Power(a,x)	**a** санының **x** нчы дәрәҗәсе $\mathbf{(a^x)}$
Тригонометрик	
PI	π саны($\mathbf{\pi \approx 3{,}1415926535897932385}$)

ArcCos(x)	**x** ның арккосинусы
ArcCosH(x)	**x** ның гиперболик арккосинусы
ArcSin(x)	**x** ның арксинусы
ArcSinH(x)	**x** ның гиперболик арксинусы
ArcTanH(x)	**x** ның гиперболик арктангенсы
CosH(x)	**x** ның гиперболик косинусы
Cotan(x)	**x** ның котангенсы **(Cos(x) / Sin(x))**
Hypon(a,b)	Турыпочмаклы өчпочмак ипотенузасы - $\sqrt{a^2+b^2}$
SinH(x)	**x** ның гиперболик синусы
Tan(x)	**x** ның тангенсы **(Sin(x) / Cos(x))**
TanH(x)	**x** ның гиперболик тангенсы
Почмакларны бер системадан икенчесенә күчерү	
CycleToRad(x)	**x** периодларын радианнарга күчерү **(1 период 2π га туры килә)**
DegToRad(x)	**x** градусны радианнарга күчерү **(180° π радианга туры ктилә)**
RadToCycle(x)	Радианнарда күрсәтелгән **x** зурлыгын периодларга күчерү **(2π периодка туры килә)**
RadToDeg(x)	Ралианнарда күрсәтелгән **x** зурлыгын градусларга күчерү **(π радиан 180° ка туры килә)**
Статистик исәпләүләр	

RandG(m,sd)	Очраклы санны Гаусс таралышы буенча генерацияләү, **m** – математик көтелеш, **sd**-стандарт тайпылыш бирелә
MaxIntValue(D)	**D** бөтен саннар массивыннан максималь сан
MaxValue(D)	**D** саннар массивыннан максималь сан
Mean(D)	**D** массивының уртачасы
MeanAndStdDev(D, m,sd)	**D** массивының уртача саны **m** һәм квадратик тайпылышы **sd**
MinIntValue(D)	**D** бөтен саннар массивыннан минималь сан
MinValue(D)	**D** саннар массивыннан минималь сан
StdDev(D)	**D** массивының квадратик тайпылышы
Sum(D)	**D** массивындагы саннар суммасы
SumInt(D)	**D** бөтен массивындагы саннар суммасы

Типларны үзгәртү функцияләре. Типларны үзгәртү функцияләре бер типтагы үзгәрешлеләрне башка типка күчерү өчен хезмәт итәләр. Барыннан да ешрак андый функцияләр мәгълүмат керткәндә һәм чыгарганда файдаланылалар. Мәсәлән, санлы типтагы үзгәрешле кыйммәтен кертү/чыгару кырына чыгару өчен кыйммәтне юл тибына әверелдерергә кирәк, чөнки мондый кырга юлларны гына чыгарырга була. Чираттагы таблицада типларны үзгәртү өчен төп функцияләр китерелгән.

Таблица 5

Типларны үзгәртү функцияләре

Функция	**Кыйммәт**

Chr(n)	ANSI стандартындагы **n** нчы код номерлы символ
IntToStr(n)	Бөтен **n** санын юл тибына үзгәртү
FloatToStr(n)	Реаль **n** санын юл тибына үзгәртү
FloatToStrF(n,f,l,m)	Реаль **n** санын **f** форматлы юл тибына үзгәртү, юл озынлыгы **l** һәм унарлы ноктадан соң цифрлар саны **m**
StrToInt(s)	**s** юлын бөтен санга үзгәртү
StrToFloat(s)	**s** юлын реаль санга үзгәртү
Round(n)	**n** га иң якын бөтен сан
Trunc(n)	**n** ның бөтен өлеше
Frac(n)	Реаль n санының вакланма өлеше
Int(n)	Реаль n санының бөтен өлеше

6.4. ЦИКЛ ОПЕРАТОРЛАРЫ.

Күп кенә мәсьәләләр чишү өчен алгоритмнар төзегәндә еш кына билгеле бер адымнар төркемен кабатларга кирәк була. Кабатлауларны цикларны) эшкә ашыру өчен өч төрле цикллар файдаланылырга мөмкин: параметрлы, алшартлы, артшартлы цикллар. Билгеле, боларның һәркайсын эшкә ашыруны шарт операторы һәм шартсыз күчү операторын кулланып та башкарырга мөмкин.

Әгәр цикл җисемен кабатлаулар саны билгеле икән, **параметрлы**

13 нче рәс. Параметрлы цикл

цикл операторы кулланыла**.** Аның структур схемасы мондый.

Гомуми рәвеше:

For *үзгәрешле* := *баш_кыйммәт* **to** *ахыр_кыйммәт* **do** *җисем* яки

For *үзгәрешле* := *ахыр_кыйммәт* **downto** *баш_кыйммәт* **do** *җисем*

биредә

үзгәрешле – цикл параметры, параметр сыйфатында тәртпи типлы теләсә нинди үзгәрешле файдаланылырга мөмкин (*мәсәлән,* integer);

баш_кыйммәт һәм

ахыр_кыйммәт – тиңдәшле рәвештә цикл параметрының баш һәм ахыр кыйммәтләрен билгеләүче аңлатмалар;

җисем – гади яисә төзелмә оператор.

Циклның башлангыч һәм ахыр кыйммәтләре тибы цикл параметры тибы белән тәңгәл булырга тиеш. Цикл операторы башкарылу үзгәрешле <= ахыр_кыйммәт шартын тикшерүдән башлана. Әгәр ул үтәлмәсә, идарә циклдан соң килгән операторга тапшырыла. Әгәр шарт чын булса (үтәлсә) җисем башкарыла, аннары цикл параметрына чираттагы (**to** циклы) яисә алдагы кыйммәт (**downto** циклы) бирелә, аннары процесс кабатлана. Әгәр

дә цикл параметры бөтен сан икән (барыннан да ешрак ул шулай була да) бу цикл җисеме башкарылганда бергә арттыру яисә кимүне аңата.

Искәрмә: *параметрның үзгәрү адымы һәрвакытта 1 яки –1 гә тигез.*

Мисаллар:

(үзгәрешлеләр болай тасвирланган дип исәпләнелә: **var** S,I: Integer; Sim : Char; L : Boolean;)

1) S:= 0; **For** I := 5 **to 7 do** Inc(S);

Inc(S,K) гамәле S := S + K, ягъни S параметрын K кыйммәтенә арттыруны белдерә. Әгәр K бирелмәгән икән, K = 1 дип санала. (Кире функция **Dec**(S,K) –S үзгәрешлесе кыйммәтен K кыйммәтенә киметү). Мисалда цикл башкарылганнан соң S үзгәрешлесе 3 кыйммәте алачак.

2) S:=10; **For** I:= 10 **downto** 6 **do** Dec(S,2); S үзгәрешлесе 0 кыйммәте ала.

3) S:=0; **For** Sim := ‘A’ **to** ‘D’ **do** S:= S + 1; S 4 кыйммәте ала.

4) S:=0; **For** L:=false **to** true **do** Inc(S,3); S 6 кыйммәте ала.

Циклда берничә оператор башкарырга кирәк булса, төзелмә оператор файдаланыла.

Мисал:

```
S:=0;
    For I:= 1 to 3 do
    begin
        S:= S + Sqr(I);
P := S + 1;
```

end;

Цикл башкарылганнан соң S үзгәрешлесе 14 кә, ә P үзгәрешлесе 15 кә тигез булачак.

Искәрмәләр:

1. *Цикл параметры, цикл параметрының башлангыч һәм ахыр кыйммәтләрен цикл эчендә (җисемендә) үзгәртү тәкъдим ителми;*
2. *Циклдан мәҗбүри (тәмамламыйча)чыгу өчен Object Pascal дә* **Break** операторы файдаланыла*;*
3. *Цикл җисемен цикл параметрының чираттагы кыйммәте белән исәпләү өчен (бәлки, агымдагысын тәмамламыйча) Object Pascal дә* **Continue** операторын кулланалар*;*
4. *Циклдан чыккач цикл параметры кыйммәте билгесез була;*

Параметрлы цикл операторы цикл ничә тапкыр башкарылачагы билгеле булса файдаланыла. Ләкин еш кына цикл җисеменең ничә тапкыр башкарылачагы билгеле булмый. Бу вакытта циклның башка типларын: **алшартлы** һәм **артшартлы** цикларны файдаланалар.

Алшартлы цикл операторы мондый рәвешле була:

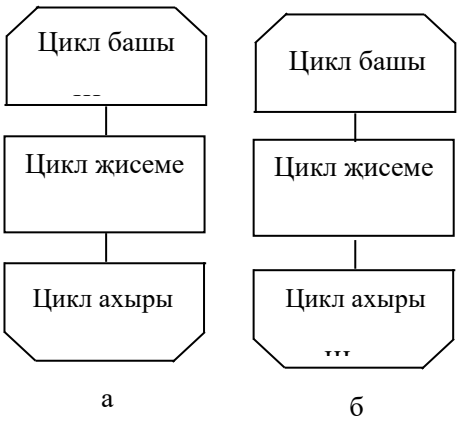
while *шарт* **do** *җисем**;***

биредә

шарт – логик типлы аңлатма;

җисем- гади яки төзелмә оператор.

Циклның структур схемасы мондый:



14 нче рәс. Алшартлы (а) һәм артшартлы (б) цикллар

Операторны башкару шарт аңлатмасы кыйммәтен исәпләүдән башлана. Әгәр ул кыйммәт **True** икән, *җисем* башкарыла. Аннары тагын шарт тикшерелә һәм процесс кабатлана. Бу шарт аңлатмасы кыйммәте **False** булганчы дәвам итә. Аннан соң инде идарә цикл операторыннан соңгы операторга тапршырыла. Әгәр дә шарт аңлатмасы кыйммәте беренче кергәндә үк **false** икән, цикл җисеме бер тапкыр да башкарылмый. **Искәрмә:** *цикл эчендәге операторларның берсе шарт аңлатмасы кыйммәтен үзгәртергә тиеш, башкача цикл чиксез дәвам итәчәк.*

Мисал:

```
k := 0;
while k <6 do
begin
    Inc(k,2);
    Y := Sqr(k);
end;
```

Мисалда *k* үзгәрешлесе кыйммәтен 2 гә арттыручы *Inc(k,2)* процедурасы файдаланылган.

Цикл башкарылу процессында Y үзгәрешлесе 4, 16, 36 кыйммәтләре алачак.

Артшартлы цикл операторы 14 нче рәсемдә күрсәтелгән структур схемага ия һәм алшартлы цикл операторына охшаш, ләкин шарт цикл җисемен тәшкил итүче операторлардан соң тикшерелә. Операторның гомуми рәвеше:

Repeat

җисем;

until *шарт;*

биредә

шарт – логик типлы аңлатма;

җисем - оператор яки төзелмә оператор;

Искәрмәләр:

1. *Цикл җисеме циклның ачкыч сүзләре арасына урнашкан булганга, цикл җисемендә төзелмә оператор файдаланылганда аны* ***begin*** *һәм* ***end*** *оператор җәяләре арасына урнаштырырга кирәкми;*
2. *Цикл шарты цикл ахырында тикшерелгәнгә цикл җисеме бер тапкыр булса да үтәлә.*

Артшартлы цикл операторы цикл җисеме башкарылудан башлана. Аннары шарт аңлатмасы исәпләнелә. Әгәр кыйммәт **true** икән, Циклдан чыгу башкарыла, ә **false** булса, җисем кабат башкарыла.

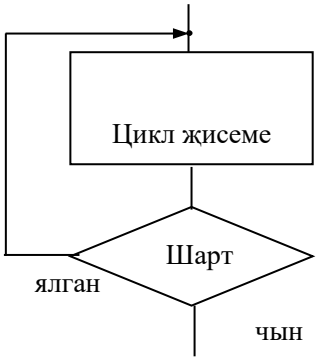
Искәрмә: ***алшартлы*** *циклдан чыгу шарт* ***ялган*** *булса, ә* ***артшартлы*** *циклдан чвыгу шарт* ***чын кыйммәте алса*** *башкарыла.*

Мисал: алшартлы циклдагы кебек үк мисалны карыйк:

```
k:=0;
repeat
    Inc(k,2);
    Y := Sqr(k);
until(k>6);
```

нәтиҗә шул ук булачак.

Теләсә кайсы тип цикл операторы **шарт операторы** ярдәмендә төзелергә мөмкин. Мәсәлән, шарт операторы ярдәмендә төзелгән



15 нче рәс. Шарт операторы ярдәмендә төзелгән артшартлы цикл

артшартлы цикл операторы схемасы мондый була:

Шарт операторы ярдәмендә цикл төзү өчен тамгалар һәм шартсыз күчү операторы – **goto** файдалану зарур. Үзгәрешлеләр кебек үк, тамгалар игълан ителергә тиеш. Тамгалар **тамгалар игълан итү бүлегендә** (**Label**) игълан ителә. Тамганы файдалану өчен тамга исеме һәм ике нокта файдаланыла. Тамгага күчү **goto** *тамга(лар)* операторы белән башкарыла.

Хәзер исә шарт операторы кулланып төзелгән шул ук мисалны карыйк. Тамгаларны игълан итү бүлегендә тамганы тасвирлау зарур. Тамга исеме **beg** булсын**.**

Мисал:

```
Label
  beg;
begin
                {башка операторлар }
                k:=0;
   beg   :Inc(k,2);
                Y := Sqr(k);
                If k < 6 then goto beg;
                {башка операторлар }
end;
```

Тәкъдим: *Шартсыз күчү операторы* ***goto*** *ны файдалану программа текстын укуны авырайта, шуңа шартсыз күчүне файдалану тәкъдим ителми.*

6.5. ПРОГРАММАНЫ КӨЙЛӘҮ.

Программа төзегәндә хаталар килеп чыгарга мөмкин. Хаталарны өч төркемгә бүлү кабул ителгән:

1. Синтаксик
2. Башкару вакыты хаталары;
3. Алгоритмик хаталар.

Аларның иң гадиләре **синтаксик** – ягъни операторларны дөрес язмау хаталары. Алар программаны компиляцияләү процессында төзәтелә.. **2. Башкару вакыты хаталары** – мәсәлән, исәпләү хаталары (мәсәлән, 0 гә бүлү), типлар хаталары шулай ук чагыштырмача тиз табыла. Программа Delphi тышчасыннан эшләтеп җибәрелә дип фараз итик. Андый хата

чыккач, хата турындагы хәбәр тәрәзендәге «Ok» га басарга, аннары программа башкарылуын сайлактагы «**Run/Program reset**» яисә <**Ctrl**><**F2**> клавишалар комбинациясенә басып тәмамларга кирәк. Андый хата чыккач курсор хатаны чыгарган юлга күрсәтәчәк. Иң катлаулы хаталар исә **алгоритмик** хаталар. Программа хаталарсыз компиляцияләнә, эшләтеп җибәреп караганда да хаталар чыкмый, ләкин нәтиҗәләрне анализлаганда аның дөрес түгел икәнлеге ачыклана. Алгоритмны кулдан «әйләндерү» , ягъни **программаны көйләү** таләп ителә.

Программаны **трассировкалау** (ягъни «алгоритм логикасын» тикшерү) өчен түбәндәге гамәлләрне башкару зарур. Сайлакта «**Run/Step over**» яисә «**Run/Trace in to**» ны сайларга, аларга тиңдәшле рәвештә <**F8**> һәм <**F7**> туры килә, ә «**Run/Trace in to**» командасы җентеклерәк трассировка белән аерылып тора. Трассировканы туктату һәм программа автомат рәвештә башкарылуны дәвам итү өчен «**Run/Run**» пунктын сайларга яки <**F9**> га басарга кирәк. Программаны «**Run/Program reset**» командасы яисә <**Ctrl**><**F2**> клавишлар комбинациясе белән туктатып була. Кайвакыт трассировканы программаның билгеле бер юлыннан башкарырга кирәк була. Бу максат өчен курсорны кирәкле юлга китерү һәм «**Run/Run to cursor**» командасын башкару яки <**F4**> кә басу зарур. Еш кына алгоритм хатасының ихтималлыгы иң зур булган урын билгеле була, бу вакытта **тукталу ноктасын** файдаланалар**.** Программист кирәкле юлга курсорны урнаштыра һәм анда «**Run/Add Breakpoint**» сайлак командасы яки <**Ctrl**><**F8**> клавишлар комбинациясе ярдәмендә тукталу ноктасын урнаштыра. Сайланган юл тамгаланачак. Тукталу нокталарын бетерү өчен бу юлда кабат <**Ctrl**><**F8**> гә басарга кирәк. Програма башкарылганда программа тукталу ноктасына кадәр башкарылачак, аннары программист программаны <**F7**> һәм <**F8**> ярдәмендә трассировкалый алачак. Кирәк булса тукталу ноктасында тукталу башкарылачак **шартны** күрсәтергә була

(көйләү «**Run/Add breakpoints**» сайлак пунктының «**Breakpoints**» тәрәзендә башкарыла).

Программаны адымлап башкарганда еш кына «алгоритм логикасын» гына түгел, ә кайбер үзгәрешлелеәрнең кыйммәтләрен дә тикшерергә туры килә. Моның өчен «**View/Watch/Add watch**» сайлак командасын башкаралар һәм тиешле үзгәрешле исемен кертәләр. Бу команданы болай да башкарып була: курсорны кыйммәтен карарга кирәкле үзгәрешлегә китерергә һәм <**Ctrl**><**F7**> гә басарга. Программаны трассировкалаганда бу очракта «**Watch list**» тәрәзендә кызыксындырган үзгәрешлеләрнең кыйммәтләрен күреп була.

6.6. МАССИВ МӘГЪЛҮМАТ ТИБЫ

Яңа мәгълүмат тибы. Яңа мәгълүмат тибы игълан итәргә кирәк булганда аны тасвирлауны типларны тасвирлау бүлегенә урнаштырырга кирәк була. Гомуми очракта типларны тасвирлау болай башкарыла:

Исем = Типны тасвирлау;

биредә

Исем – яңа тип исеме;

Типны тасвирлау – төзелгән типлы үзгәрешлеләрнең мөмкин кыйммәтләрен тасвирлау.

Искәрмә: *яңа тип төзегәндә тип исеменнән соң «тигез» билгесе куела, аннары тип тасвирламасы китә.*

Мисаллар:

DayOfWeek = (Monday, Wednesday, Friday);

Day = 1..31;

Мондый тип саналышлы тип дип атала, бу тип үзгәрешлеләре санап кителгән кыйммәтләр генә ала ала. Мисалда бу атна көннәренең исемнәреннән берсе (**DayOfWeek** тибы) яки 1 дән 31 гә кадәр саннарның берсе (**Day** тибы). Саналышлы тип үзгәрешлеләренә **Pred**(үзгәрешле) һәм **Succ**(үзгәрешле) функцияләрен кулланырга була, алар мөмкин кыйммәтләрнең элеккесен (**Pred**) һәм чираттагысын (**Succ**) кайтаралар.

Мисаллар:

Үзгәрешлеләр игълан ителгән булсын W : DayOfWeek и D : Day ул вакытта:

W := Wednesday;

Succ(W); {оператор ‘Monday’ кыйммәтен кайтарачак}

D := 5;

Pred(D); {оператор ‘4’ кыйммәтен кайтарачак}

Искәрмәләр:

1. Саналышлы тип кыйммәтләрендә кириллица кулланылырга тиеш түгел;
2. Әгәр **Succ** яки **Pred** функцияләре ярдәмендә **Succ** өчен соңгы яисә **Pred** өчен беренче элементка мөрәҗәгать ителсә, хата чыгачак.

Массивлар. **Массив –** уртак исемле, бер типлы һәм тәртипкә салынган мәгълүмат җыелмасы. Массивларны бериш информация (вектор, матрица һ.б.ш.) саклау өчен файдалану уңайлы.

Массивны игълан итү. Теләсә кайсы үзгәрешле кебек үк, массив типлы үзгәрешле файдалану алдыннан үзгәрешлеләрне игълан итү бүлегендә игълан ителергә тиеш. Үзгәрешлеләрне игълан итү бүлегендә бер үлчәмле массивны игълан итү гомуми очракта мондый рәвешле булачак:

Исем : **array** [аскы_чик_*индекс*..*өске_чик_индекс*] **of** *тип*

биредә

Исем – массив-үзгәрешле исеме;

array – ачкыч сүз, массив игълан ителгәнен белдерә;

аскы_чик_индекс,

өске_чик_индекс – массив элементлары индекс ы үзгәрү диапазонын, димәк, массивтагы элементлар санын да билгеләүче тәртип типлы аңлатмалар (гомуми очракта);

тип – массив элементлары тибы.

Мисаллар:

day = **array** [1..30] **of** integer; {*30 бөтен саннан торган массив* }

r : **array** [5..7] **of** boolean; {*3 логик элементтан торган массив* }

Әгәр дә бертөрле массив типлы берничә үзгәрешле игълан итәргә кирәк икән, типларны тасвирлау бүлегендә яңа массив тибы төзү һәм үзгәрешлеләрне тасвирлау бүлегендә тип сыйфатында төзелгән типны күрсәтү мәгъкуль. Бу очракта массив тасвирламасы мондый булачак:

type *{типларны тасвирлау бүлеге}*

ТипИсеме = **array** [*аскы_чик_индекс..өске_чик_индекс*] **of** *тип;*

var *{үзгәрешлеләрне тасвирлау бүлеге }*

Исем : ТипИсеме;

Массив игълан иткәндә шулай ук исемле константаларны файдалану уңайлы.

Мисал:

Const

N = 1;

E = 5;

Type

Ar = **array** [N..E] **of** Char;

Массивлар шулай ук күп үлчәмле дә була ала. Мондый массивны тасвирлау өчен һәр үлчәмнең индексләр диапазонын өтер аша күрсәтү зарур.

Гомуми очракта, мәсәлән, ике үлчәмле массив тасвирламасы мондый булачак:

Исем : **array** [*a_u1..ө_u1, a_u2..ө_u2*] **of** *тип*

биредә

a_u1, ө_u1, a_u2, ө_u2 – беренче һәм икенче үлчәмнең өске һәм аскы чикләрен (индексларын) билгелүче тәртип типлы аңлатмалар, мәсәлән, бөтен константалар.

Мисал:

Coord : **array** [1..5, 1..3] **of** integer;

{ Coord массив типлы үзгәрешле ике үлчәмле массивны тасвирлый}

Массив элементын файдалану өчен массив иснемен һәм элементның номерын – индексын күрсәтү зарур. Индекс константа яисә тәртип типлы аңлатма булырга һәм квадрат җәяләргә алынырга тиеш.

Мисаллар:

P := Coord[1,3];

Coord[5,2] := Coord[3,1];

Массив белән типик гамәлләр. Боларга массивны кертү, чыгару, тутыру, массивтагы максималь яисә минималь элементны эзләү, бирелгән элементны эзләү, массивны сортлау кебек гамәлләрне кертергә мөмкин.

Псевдоочраклы саннар генераторы. Әгәр очраклы кыйммәтләр алырга кирәк икән, псевдоочраклы саннар генераторын файдаланырга була. Эш шунда ки, чиста очраклы саннар закончалыкка буйсынырга тиеш түгел, ә генератор тапкан саннар барыбер ниндидер алгоритм нигезендә табыла һәм шуңа күрә, бәлки бик озын эзлеклелектән соң, кабатлана башлаячаклар. Шуңа күрә мондый саннар очраклы дип түгел, ә псевдоочраклы дип атала. Хәер, безнең мәсьәләләр өчен алар барыбер яраклы. Ике процедура бар:

1. **Randomize –** саннар генераторын «эшләтеп җибәрә» (башкача генераөияләнүче саннар эзлеклелеге бер үк булачак);
2. **Random[(R)]** – 0 дән R га кадәр (R ны кертеп) очраклы сан генерацияләү өчен хезмәт итә. Биредә һәм арырак, операторлар, процедураларны тасвирлаганда файдаланылучы квадрат җәяләр квадрат җәя эчендәге элементның булмаска да мөмкин икәнен күрсәтәчәк; бу вакытта стандарт – килешү буенча көйләнешләр файдаланыла. Мәсәлән, әгәр Random операторында R чиге гүрсәтелмәгән икән, саннар 0 дән 1 гә кадәр (1 не кертеп) диапазонында генерацияләнәчәкләр.

Нормаль таралыш буенча очраклы сан генерацияләү өчен **Math** модуленнән **RandG**(m,sd) функциясе файдаланыла (математик көтелеш **m** һәм стандарт тайпылыш **sd** бирелә).

Минималь (максималь) элементны эзләү. Мисал сыйфатында бер үлчәмле бөтен саннар массивын файдаланырбыз.

Минималь (максималь) элементны эзләү алгоритмы җитәрлек дәрәҗәдә гади: башта массивның беренче элементы минималь (максималь) дип игълан ителә, аннары массивның калган элементлары бу элемент белән

чагыштырыла. Әгәр чираттагы чагыштыру вакытында тикшерелүче элемент минималь (максималь) дип кабул ителгән элементтан кечерәк (зуррак) икән, бу элемент минималь (максималь) дип кабул ителә һәм калган элементларны тикшерү дәвам ителә.

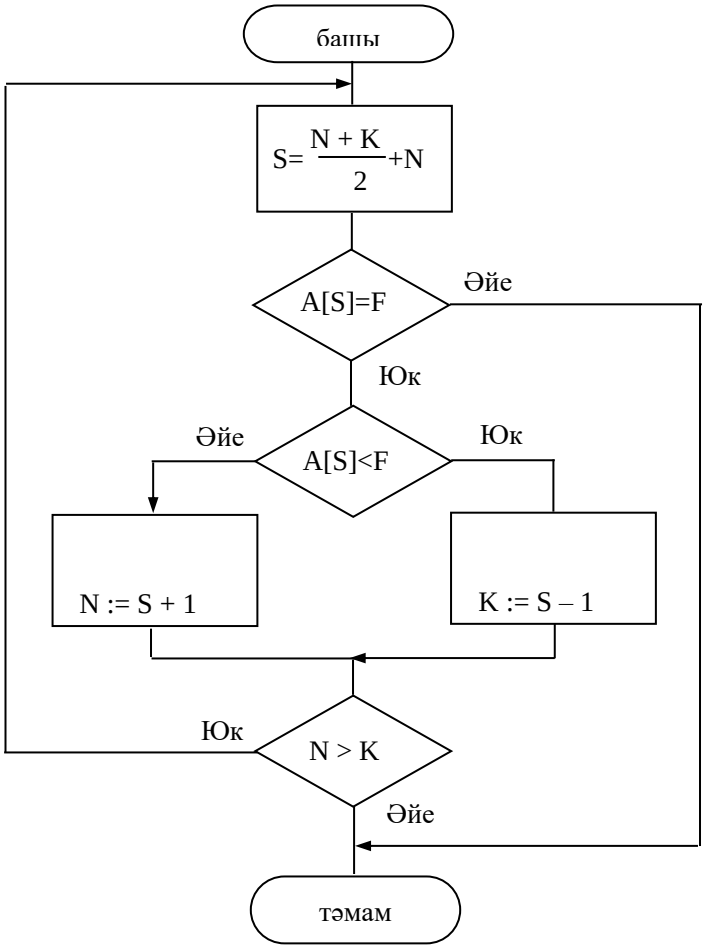
Мисал: **Size** –**a** массивы үлчәме, **min –** минимумны саклау өчен үзгәрешле, **max –** максимумны саклау өчен үзгәрешле, **I** – цикл параметры булсын.

Ул вакытта минималь һәм максималь кыйммәтләрне эзләү мондый булачак:

```
min := a[1];
max := a[1];
For I := 1 to Size do
Begin
    If a[I] < min then min := a[I];
    If a[I] > max then max := a[I];
end;
```

Цикл тәмамланганнан соң **min** үзгәрешлесендә массивтагы минималь, ә **max** үзгәрешлесендә максималь кыйммәт булачак.

Бирелгән элементны эзләү. Иң гади ысул – аралау – массив кыйммәтләрен эзлекле рәвештә карап чыгу һәм эзләнүче кыйммәт белән чагыштыру. Ләкин массив үлчәме зур булса, моңа вакыт бик күп китәчәк. Оптималь ысул булып **бинар эзләү** ысулы тора. Ләкин бу алгоритм массивның инде сортланган икәнен күздә тота. Алгоритмның структур схемасы мондый:



16 нчы рәс. Бинар эзләү

Биредәге тамгаланышлар: A – кыйммәтләр массивы; S – массивның урта индексы; N – массивның башлангыч индексы; K – массивның ахыргы индексы; F – эзләнүче

Программаны күрсәтелгән структур схема буенча төзү авыр түгел.

6.7. ЯЗМАЛАР

Төрле типтагы мәгълүмат төркеме белән эшләү өчен «язма» тибы кулланыа

Язма тибын машина хуҗалары исемлеге мисалында карыйк :

№	Хуҗасы	марка
1	Бибигалиев А.Б.	ВАЗ - 2102
3	Мтоян И.Ә.	BMW
3	Курбануглы Т.М.	ОКА

Мондагы һәр юл аерым элементлар – төрле типтагы мәгълүматттан тора:

а) тәртип номеры – бөтен сан;

б) Фамилия И. А.И.. – символ юлы;

в) машина маркасы – символ юлы

Бу мәгълүматны бер төркемгә берләштерергә һәм язма дип карарга мөмкин. Тулаем язма һәм аның аерым элементлары (кырлары) исемнәр белән тамгалана.

Мәсәлән, түбәндәге тамгалар кертик:

Xuza- барлык язма исеме;

N - тәртип номеры;

NAME - фамилия И.А.и..;

Ident – машина маркасы.

Язма элементына *(кырына) мөрәҗәгать программада аныкланган (төзелмә) исем ярдәмендә башкарыла. Төзелмә исемгә язма исеме һәм кыр исеме керә һәм болай языла:

<язма исеме>.<кыр исеме>

6.8. ЯЗМАНЫ ИГЪЛАН ИТҮ (ДЕКЛАРАЦИЯ)

Язманы үзгәрешлеләрне игълан итү бүлеге VAR да яисә типлар игълан итү бүлеге TYPE бүлегендә игълан ителә.

VAR <язма исеме>: RECORD

<кыр исеме 1>: тип;

<кыр исеме 2>: тип;

. . .

<кыр исеме n>: тип

END;

Яки

TYPE

<язма тибы исеме>= RECORD

<кыр исеме 1>: тип;

<кыр исеме 2>: тип;

. . .

<кыр исеме n>: тип

END;

VAR

<язма исеме>: <тип исеме>;

Мисаллар:

Язманы игълан итү

```
TYPE xuza= record
N: Integer;
NAME: String[25];
RB: String[15];
  END;
  VAR
  R1, R2: xuza;
```

Яки

```
VAR  xuza: record
N: Integer;
NAME: String[25];
IDENT: String[15];
END;
```

Язма элементлары (кырлар) программада гадәти үзгәрешлеләр кебек кулланыла ала.

Язма элементы (кыры) белән аның тибындагы мәгълүмат белән эшләргә мөмкин булган гамәлләрне башкарырга ярый.

Мисал:

R1.N := 2;

R2.Name := ‘ Бибигалиев А.Б.’;

Тулаем язмага мөрәҗәгать итү бары тик үзләштерү операторында гына мөмкин. Бу вакытта бер үк типтагы язмалар гына кулланыла ала.

R1 := R2;

6.9. ЯЛГАУ ОПЕРАТОРЫ

Язма кырларына мөрәҗәгать итү аныкланган исем ярдәмендә башкарыла. Ялгау операторы язма кырына мөрәҗәгатьне гадиләштерергә мөмкинлек бирә. Исем башламга чыгарыла, ә блокта бары элементлар исемнәре генә файдаланыла.

Ялгау операторының гомуми рәвеше

WITH <язма исеме> DO

Begin

{язма элементлары исеме генә булган операторлар}

End;

Мәсәлән:

WITH R_EX1 DO

```
Begin
Write('Язма номерын кертегез  '); Readln(N);
Write('Фамилия, и, а.и. кертегез  ') Readln(Name);
End;
```

Вариантлы язмалар

Язмаларның аерым тибы булып «вариантлы язмалар» тора, алар махсус сүз ярдәмендә игълан ителә:

Болай итеп зур гына уртак өлешләре һәм төрле структураларда төрлечә булган кечкенәрәк өлешләре булган язмаларны сакларга мөмкин.

Мисал. Геометрик фигуралар параметрларын саклаучы язма. Бу, мәсәлән,

Квадратның бер ягы,

Өчпочмакның ике ягы һәм алар арасындагы почмак,

Әйләнәнең радиусы.

Уртак өлешләре булып агымдагы нокта координаталары x,y тора.

Игълан мондый булачак:

```
VAR
MS: Record        {язма типлы үзгәрешле}
x: real;    {агымдагы координата x}
y: real;    {агымдагы координата y}
Case Fig: (Square, Triangle, Circle) of
{вариант өлеше}
Square: (side: real);          {квадрат}
```

```
Triangle: (s1, s2, angle: real);    {өчпочмак}
Circle: (Radius: real);             {әйләнә}
END;
```

Вариант өлеше бер генә була ала һәм язма ахырында урнаша. Ул case сүзеннән башлана, аннан соң вариант сайлау үзгәрешлесе килә.

яки

```
VAR
MS: Record
x: real;
y: real;
Case Fig: (Square, Triangle, Circle) of
Square: (side: real);          {квадрат}
Triangle: (s1, s2, angle: real); {өчпочмак}
Circle: (Radius: real);        {әйләнә}
END;
```

Бу үзгәрешленең һәр мөмкин кыйммәтеннән соң түгәрәк җәяләр эчендә типларын күрсәтеп бу вариант кырлары языла. Кырлар булмаса да түгәрәк җәяләр мәҗбүри.

7. СОРТЛАУ АЛГОРИТМНАРЫ

Сортлау ул – массивтагы элементларны билгеле бер тәртиптә, мәсәлән, үсә бару тәртибендә тезеп чыгу.

7.1. САЙЛАУ ЫСУЛЫ БЕЛӘН СОРТЛАУ

Башлангыч массивта иң кечкенә элемент табыла һәм ул массивтагы беренче элемент белән алмаштырыла. Аннан соң бу гамәл массивның калган өлеше белән башкарыла, ягъни икенче элемент калган өлештәге иң кечкеә элемент белән алыштырыла h.б. Шулай итеп, массивны сортлау өчен N+(N-1)+(N-2)+...+1 яисә N*N карап чыгу кирәк булачак..

Листинг 1. Сайлау ысулы белән сортлау

```
procedure SellectionSort( var a: array of integer; min,
max: Integer);
var
i, j, best_value, best_j: longint;
begin
for i:=min to max do begin
best_value:=a[i];
best_j:=i;
for j:=i+1 to max do begin
if a[j]<best_value then begin
best_value:=a[j];
best_j:=j;
end;
end;
a[best_j]:=a[i];
a[i]:=best_value;
end;
```

```
end;
```

min һәм max элементлары белән массивның сортау башкарыла торган өлкәсен чикләргә мөмкин. Барлык массиваны сортлау өчен түбәндәге оператор языла

Листинг 2. Код Delphi/Pascal

```
SellectionSort(a, 0, high(a));
```

7.2. ӨСТӘҮ ЫСУЛЫ БЕЛӘН СОРТЛАУ

Яңа массивка (массивның “яңа” өлешенә) төзелүче массив тәртипкә салынган булырлык итеп “иске” массивтан (массивның “иске” өлешеннән) элементлар өстәлеп бара.

Листинг 3. Өстәү ысулы белән сортлау

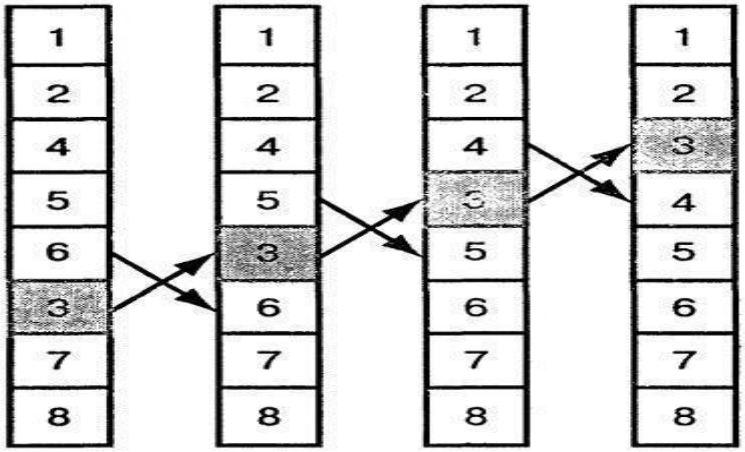
```
procedure InsertionSort( var a: array of integer; N: integer);
var
B: array [0..10000] of integer;
i, j: integer;
begin
for i:=0 to N do begin
j:=i;
while (j>1) and (B[j-1]>A[i]) do begin
B[j]:=B[j-1];
j:=j-1;
end;
B[j]:=A[i];
end;
for i:=0 to N do
```

```
A[i]:=b[i];
end;
```

Алгоритм эше өчен N*N нән артык үтеш кирәк.

7.3. КҮБЕК ЫСУЛЫ БЕЛӘН СОРТЛАУ

Бу алгоритм кирәкле тәртиптә тормаган парны эзли һәм ул пардагы элементларның урыннарын алыштыра. Процесс сортлау тәмамланганчы дәвам итә. Рәсемдә бу ысул белән сортлау мисалы күрсәтелгән.



Рәсемдәге элемент өскә таба судагы һава күбеге кебек менә, шуңа күрә бу күбек ысулы белән сортлау дип атала.

Листинг 4. Күбек ысулы белән сортлау

```
procedure BubbleSort( var a: array of integer; min, max: Integer);
var
i, j, tmp: integer;
begin
for i:=min to max do
for j:=min to max-i do
if A[j]>A[j+1] then
```

```
begin {Элементларның урыннарын алмаштыру}
tmp:=A[j];
A[j]:=A[j+1];
A[j+1]:=tmp;
end;
end;
```

7.4. ҖӘҺӘТ СОРТЛАУ.

Массив ике өлешкә бүленә, аннан соң сортлау өчен үзен-үзе рекурсив чакыра. Бу вакытта беренче өлеш элементлары икенче өлеш элементларыннан кечерәк. Мисал: әгәр алгоритм 0 яисә 1 элементтан торган массив өчен чакырыла икән, процедура тәмамлана, алай булмаса, аңа карата исемлек ике өлешкә бүленә торган элемент сайлана, беренче өлешкә сайланган элементтан кечкен элементлар, икенчесенә сайланган элементтан зур элементлар урнаштырыла. Аннан соң ул ике астмассивны да сортлар өчен үзен-үзе рекурсив чакыра.

Листинг 5. Җәһәт сортлау

```
procedure QuickSort( var a: array of integer; min, max: Integer);
Var
i,j,mid, tmp : integer;
Begin
if min<max then begin
mid:=A[min];
i:=min-1;
j:=max+1;
while i<j do begin
repeat
i:=i+1;
```

```
until A[i]>=mid;
repeat
j:=j-1;
until A[j]<=mid;
if i<j then begin
tmp:=A[i];
A[i]:=A[j];
A[j]:=tmp;
end;
end;
QuickSort(a, min,j);
QuickSort(a, j+1,max);
end;
end;
```

7.5. ШЕЛЛ ЫСУЛЫ БЕЛӘН СОРТЛАУ.

Төп идея: бер-берсеннән ераста урнашкан элементларны чагыштырып, массивтагы “тәртипсезлек”не бетерү. Чагыштырылучы элементлар арасындагы интервал акрынлап бергә кадәр кими. Соңгы стадиядә сортлау күрше эементлар урыннарын алыштыруга кайтып кала.

Листинг 6. Шелл ысулы белән сортлау

```
procedure TForm1.SortShell( var a: array of real; N: Integer);
var
h:Variant;
c:Boolean;
g:Integer;
i:Integer;
j:Integer;
```

```
tmp:Real;
begin
h:=1;
g:=0;
repeat
h:=3*h+1
until (h>=n);
if (h>n) then begin
h:= h/3;
g:=h;
end;
n:=n-1;
repeat
i:=g;
repeat
j:=i-g;
c:=True;
repeat
if a[j]<=a[j+g] then begin
c:=False;
end
else begin
Tmp:=a[j];
a[j]:=a[j+g];
a[j+g]:=Tmp;
end;
j:=j-1
until not((j>=0)and(C));
```

```
i:=i+1
until not(i<=n);
h:=g;
h:=h/3;
g:=h;
until not(g>0);
end;
```

8. ТЕКСТ ИНФОРМАЦИЯ БЕЛӘН ЭШ

Символлар һәм юллар. Программада символ үзгәрешлеләрне файдалану өчен Паскальнең стандарт тибы – **Char** ны кулланырга мөмкин. **Char** типлы үзәрешле кыйммәте булып коды булган визуаль (экранда күренүче) яисә визуаль булмаган (күренмәүче, мәсәлән <Enter> клавишасы символы) символ тора ала. Үзгәрешле үзгәрешлелеәрне тасвирлау бүлегшендә игълан ителгән булырга тиеш. Үзгәрешле кыйммәте, мәсәлән, үзләштергәндә, апострофларга алына, мәсәлән: ch := ‘Л’ – ch үзгәрешлесе Л хәрефе символын үзләштерә. **Char** типлы кыйммәтләр т әртип типлы, ягъни аларны бер-берсе белән чагыштырып була: ‘0’<’1’<…<’A’<’B’<…<’a’<’b’<…< ’A’<’Б’<…<’a’ <…<’я’. Әйткәнебезчә, символларның бер өлеше экранда чагылмый, алар өчен клавиатурада да тамга юк. Бу символларны, мәсәлән, **Chr(код)** функциясе ярдәмендә файдаланып була, биредә код – кирәкле символның номеры – коды. Барлык символлар кодка ия, алар өчен махсус код таблицалары бар. **Chr** функциясе урынына # операторын кулланырга була. Кире гамәл – күрсәтелгән символ кодын табу өчен **Ord(символ)** функциясе файдаланыла.

Мисал:

Ch := Chr(13) – символ типлы Ch үзгәрешлесенә «яңа юлга күчү» кыйммәтен бирү;

S := #10 – символ типлы S үзгәрешлесенә «юл ахыры» символын бирү.

V := Ord(’5’) – бөтен типлы үзгәрешле ’5’ символы кодын үзләштерә –’5’ символы коды 53 кә тигез.

Char стандарт тибы нигезендә «юл» тибы - **String** һәм бу тип нигезендә төзелгән барлык типлар төзелә. **String** стандарт тибын карап китик**.** Юл символлар массивын тәшкил итә, ягъни аны **array [0..255] of Char** дип тасвирлап була. Шулай итеп, юл элементына номеры буенча мөрәҗәгать итәргә мөмкин. Мәсәлән, **S** үзгәрешлесе юл типлы икән, **S[2]** – юлның икенче символы, **S[5]** – юлның бишенче символы, **S[0]** – юл озынлыгын чагылдырган символны күрсәтәчәк. Мәсәлән, әгәр **S** юлының озынлыгы 7 символ икән, **S[0]** кыйммәте **’7’** символы булачак. Килешү буенча, юлның максималь озынлыгы 255 символ. Әгәр юл озынлыгын чикләргә кирәк икән, үзгәрешле тибын сурәтләгәндә **String** сүзеннән соң юл озынлыгын куялар.

Мисаллар:

S : String[15]; – S үзгәрешлесе максималь озынлыгы 15 булган юл булып тора;

Әгәр S := ’абвгд’ икән, S[2] кыйммәте ’б’ – юлдагы икенче символ булачак..

Юлларны шулай ук бер-берсе белән чагыштырырга була. Юллар символлап чагыштырыла – әгәр юлларның символлары бер үк икән, андый

юллар тигез, әгәр бер үк позицияләрдә төрле символлар урнаша икән, коды зуррак символлы юл зуррак дип санала.

Мисаллар:

’Аб’ > ’АБ’ – икенче позициядәге Б символының коды б кодыннан зуррак;

’Аб’ < ’Аб ’ – икенче юлның озынлыгы зуррак;

’Аб’ = ’Аб’ – юллар озынлык буенча да, символлар составлары буенча да тигез.

Юллар белән мондый гамәлләр башкарып була:

Length(*X***)** – *X* үзгәрешлесе белән билгеләнгән юл озынлыгын кайтара. *Үзгәрешле* [0] дән аермалы буларак **Length** функциясе юлның санлы озынлыгын кайтара.

Delete(*X, P, K***) –** *X юлыннан P позициясеннән башлап K симолны бетерә.*

Pos(*PX, X***)** – PX астъюлыгның X юлына кергән позициясен кайтара. Әгәр юлда мондый астъюл юк икән, **Pos** функциясе 0 кыйммәтен кайтарачак.

Copy(*X, P, K***)** – X юлыннан P позициясеннән башлап K үзгәрешлесендә күрсәтелгән кадәр символы күчермәли.

Insert(*X, PX, P***)** – X юлына P позициясеннән башлап PX юлын өстәп куя.

Мисаллар:

Юл тибындагы **S** үзгәрешлесе, аның кыйммәте ’*Түбән Кама*’ һәм бөтен типлы **V** үзгәрешлесе булсын:

V := **Length**(S) – V үзгәрешлесе 10 кыйммәте алачак;

Delete(S,1,6) - S үзгәрешлесе ’Кама’ кыйммәте алачак;

V := **Pos**(’ма’,S) – V үзгәрешлесе 3 кыйммәте алачак;

S:= ’Түбән Кама’;

S := **Copy**(S,7,4) - S үзгәрешлесе ’Кама’ кыйммәте алачак;

S:= ’Түбән Кама’;

Insert(S, ’-’,6) - S юлы получит значение ’*Түбән-Кама*’ кыйммәте алачак.

8.1. ФАЙЛЛАР ҺӘМ ТЕКСТ БЕЛӘН ЭШ.

Еш кына мәгълүматны экранга түгел, ә файлга чыгарырга һәм файлдан укырга туры килә. Файллар белән эш өчен **file** типлы үзгәрешлеләр кулланыла. Текст типлы, типлаштырылган файллар һәм типсыз файллар үзгәрешлеләре була. Типлы файллар **file of тип** тасвирлана. Текст файллары, ягъни аларда текст булган файллар **TextFile** дип тасвирлана.

Мисаллар

Res : file of char; Res үзгәрешлесе символлардан торган файлга күрсәтә;

T : file of integer; T үзгәрешлесе бөтен саннардан торган файлга күрсәтә;

Str : File of String; Str үзгәрешлесе юллардан торган файлга күрсәтә;

Text : TextFile; Text үзгәрешлесе текст файлына күрсәтә.

Искәрмә: ***юллар файлы*** *һәм* ***текст файл*** *бу бер үк нәрсә түгел! т файлы юл ахыры һәм яңа юлга күчү тамгаларын үзе эшкәртә.*

Файл белән эш биш баскычтан тора. Беренчесе – файл үзгәрешлесен игълан итү, икенчесе – файл үзгәрешлесен физик файл – дисктагы файл белән бәйләү, өченчесе – файлны уку яисә язу өчен ачу, дүртенчесе - файл белән эш – файлдан уку, файлга язу, файлны үзгәртү, бишенчесе – файлны ябу. Файл типлы үзгәрешле үзгәрешлеләрне игълан итү бүлегендә игълан ителә. Файл үзгәрешлесен дисктагы конкрет физик файл белән бәйләү өчен **AssignFile**(*файл_тип_үзг*,файлга_юл) процедурасы файдаланыла.

Мисаллар:

AssignFile(text,'document.txt'); – **text** файл үзгәрешлесе һәм **'document.txt'** физик файлы бәйләнә;

AssignFile(res,'a:\chars.dat'); - **res** файл типлы үзгәрешле һәм **a:** дискының тамыр папкасында урнашкан**'chars.dat'** физик файлы бәйләнә.

Искәрмә: *Әгәр дә кирәк булмаса, файл агымдагы папкада, ягъни кушымта урнашкан ук папкада дип санала, ягъни аңа тулы юл күрсәтелми. Ә инде файл конкрет дискның конкрет папкасында икән, билгеле, тулы юлны күрсәтү зарур.*

Файллар белән эшнең өченче баскычы – файлны ачу. Файлны аннан уку, аңа язу яисә мәгълүмат өстәү (текст файлары өчен генә) ачарга мөмкин.Файл түбәндәге процедуралар ярдәмендә ачыла (процедураларга мөрәҗәгатьнең гадиләштерелгән формасы бирелә).

Reset(*файл_үзгәрешлесе*) - файлны аннан уку өчен ачу;

Rewrite(*файл_үзгәрешлесе*) – файлны аңа язу өчен ачу;

Append(*файл_үзгәрешлесе*) – текст файлын аңа яхма өстәү өчен ачу.

Бу процедураларны куллануның кайбер үзенчәлекләрен әйтеп китик. **Reset** файлны уку өчен ача, әгәр ачканда файл булмый икән, хата барлыкка килә. Шул ук чикләмә **Append** процедурасына да карый. Язу өчен ачканда исә (**Rewrite** процедурасы) әгәр файл бар икән, андагы мәгълүмат бетереләчәк, әгәр файл юк икән, ул төзеләчәк.

Искәрмә. Андый хата чыкканда DELPI хата турындагы хәбшрне белән стандларт тәрәз чыгара. Файллар белән эшләгәндә чыгарга мөмкин хаталарны үзбаш исәпкә алу һәм эшкәртү өчен түбәгндәгеләрне эшләргә кирәк:

1. Файллар белән эшне автомат идарә итүне өзәргә - **{$R-}** директивасы компиляторга гадәттән тыш ситуацияләрне автомат эшкәртү өзелгәнен белдерә. (Компиляторның барлык директивалары ике тамга эзлеклелегеннән – “аяылучы фигуралы җәя”+”доллар тамгасы” тамгаларыннан тора һәм ябылучы фигуралы җәя белән тәмамлана;
2. Критик гамәл башкарыла (ачу, уку һ.б.ш.);
3. Файллар белән эш гамәлләренең корректлылыгын автомат тикшерүне ялгау – **{$R+} директивасы**.
4. Гамәл нәтиҗәсен тикшерү. Гамәл нәтиҗәсен **IOResult** функциясе кайтара. Әгәр бу функция **0** кайтарсв, гамәл уңышлы, башкача ул хата кодын кайтара;
5. **IOResult** функциясе хәбәрен анализлау һәм кирәкле гамәлләрне башкару (мәсәлән, уку өчен файл юк икән, аны төзү).

Файл белән эшнең дүртенче баскычы – файл белән эшнең үзе, ягъни файлга мәгълүмат язу яисә аннан мәгълүмат уку.

Мәгълүматны файлга язу өчен **write**(*файл_үзгәрешлесе*, *мәгълүмат*)

яки **writeln**(*файл_үзгәрешлесе, мәгълүмат*) (текст файллары өчен) проөедуралары файдаланыла, биредә

файл_үзгәрешлесе – мәгълүмат чыгарыла торган файлга күрсәтүче үзгәрешле;

мәгълүмат – файлга язылучы мәгълүмат.

writeln процедурасы, әйткәнебезчә, текст файллары өчен генә файдаланыла, аның үзенчәлеге шунда ки, күрсәтелгән мәгълүмат язылганнан соң файлга юл ахыры тамгасы (коды) языла. Ә **writeln**(*файл_үзгәрешлесе)* – файлга юл ахыры кодын гына язачак, визуализацияләгәндә бу буш юл өстәү булып күренәчәк.

Мисал*:*

Текст файлы типлы **F** үзгәрешлесе игълан ителгән булсын (F : TextFile) һәм чыгарылучы мәгълүмат **X1** һәм **Y1** үзгәрешлеләрендә саклансын. Бу үзгәрешлеләр кыйммәтләрен юл ахыры символын өстәп файлга чыгару болай башкарылырга мөмкин: **writeln(F,X1); writeln(F,Y1);**

Файлдан мәгълүмат уку өчен **read**(*файл_үзгәрешлесе*, *үзгәрешлеләр_исемлеге*) яисә или **readln**(*файл_үзгәрешлесе, үзгәрешлеләр_исемлеге*) (текст файллары өчен генә) файдаланыла. **Readln** процедурасы мәгълүматны юллап укый, ягъни, күрсәтелгән үзгәрешлеләргә “сыймаган” мәгълүмат төшереп калдырыла.

Нинаять, файл белән эшне тәмамлагач, файлны **CloseFile**(*файл_үзгәрешлесе*) процедурасы ярдәмендә ябу зарур. Әгәр файлдан уку гамәле генә башкарылган икән, бу процедура мәҗбүри түгел, чөнки программа эше тәмамлангач барлык ачык файллар автомат рәвештә ябылачак. Ә инде файлга язу гамәле башкарылганда файл ябылмаса мәгълүмат файлга язылып бетмәскә мөмкин. Эш шунда ки, файл белән эш

алмашу буферы аша башкарыла һәм ул алмашу буферы үлчәмен файлны уку яисә язу өчен ачканда анык билгеләргә мөмкин (без биредә стандарт көйләнмәләрне файдаландык). Ягъни, файлга язарга хасиятләнгән мәгълүмат язу процедурасы башкарылгач турыдан-туры файлга язылмый, ә башта алмашу буферына тапшырыла. Алмашу буферы тулгач, андагы мәгълүмат тулаем файлга языла. Бу файл белән эшне тизләтү өчен шулай эшләнгән. Файлны япканда алмашу буферындагы мәгълүмат, күпме булуга карамастан, файлга языла һәм файл белән эш тәмамлана. Әгәр дә файлны ябу процедурасы юк икән, программа эше тәмамлангач файл барыбер ябыла, ләкин алмашу буферындагы мәгълүмат файлга күчерелмичә кала. Шулай итеп, язылырга тиешле барлык мәгълүмат файлга язылып бетмәскә мөмкин. Файлда укыганда исә мондый ситуация күзәтелми. Ләкин, корректлы эш өчен, һәр файлны ачу гамәленә файлны ябу гамәле туры килүе таләп ителә.

Файллар белән эш вакытында мондый функцияләр файдалы булырга мөмкин. **EOF**(*файл_үзгәрешлесе)* (***End Of File*** – файл ахыры сүзләреннән) функциясе файл курсоры файл ахыры тамгасында торганда **true** , алай булмаса **false** кыйммәтен ала. Шуңа бу функцияне мәгълүматны файл ахырына кадәр уку өчен куллану уңайлы. Текст файллары белән эш вакытында исә **EOLn**(*файл_үзгәрешлесе*), (***End Of Line*** – юл ахыры дип укыла) файдалы булырга мөмкин, ул файл курсоры юл ахыры тамгасында торганда **true,** алай булмаса **false** кыйммәтен ала.

Файл белән эш мисалы итеп **‘document.txt’** файлыннан символлап уку һәм файлдагы символлар санын санаучы программа фрагментын карыйк. Үзгәрешлеләрне тасвирлау бүлегендә **F**:TextFile; **C**:Char {(укылучы символ)};; **I**:Integer; дип игълан ителгән булсын.

Мисал:

```
Var   //үзгәрешлеләрне тасвирлау бүлеге
F : TextFile;
C : Char;
I : Integer;
begin
AssignFile(F, 'document.txt');      // файл үзгәрешлесе белән физик файлны бәйләү
{$R-}     //Автомат тикшерүне өзү
reset(F);    //Файлны уку өчен ачу
{$R+}    //Автомат тикшерүне ялгау
        If IOResult = 0 then       //Гәмәл уңышлы булды
        begin
        I := 0;//Символлар санагычын 0 дип кую
    While Not EOF(F) do    //Файл азагына җиткәнче
        begin
            While Not EOLn(F) do  //Юл ахырына җиткәнче
            begin
                Read(F,C); //Чираттагы символны уку
                Inc(I); //Символлар санагычын 1 гә арттыру
            end;
            Readln(F); //Яңа юлга күчү
        end
```

```
        end
        else
        begin
            MessageDlg('Файлны ачу хатасы', mtError, [mbOk], 0);
            //Хата чыкканда хәбәр бирелә
        end;
    end;
```

8.2. СТАНДАРТ ДИАЛОГ ТӘРӘЗЛӘРЕ.

Мисалда хата турында хәбәр чыгару өчен стандарт диалог тәрәзе файдаланылды. Алар турында җентекләбрәк сөйләшик.

Төрле хәбәрләр чыгару өчен стандарт диалог тәрәзләрен файдаланып була. Диалог тәрәзләрен эшләтеп җибәрү командасы:

MessageDlg(*хәбәр, Тип, Төймәләр, КонтекстБелешмә*)

биредә

Хәбәр – чыгарылучы хәбәр;

Тип – хәбәр тибы, Хәбәр информацион, кисәтүче яки хата турындагы хәбәр булырга мөмкин;

Кнопки – төймәләрнең санын һәм рәвешен билгеләүче исемлек;

КонтекстБелешмә – экран белешмәсенең номерын күрсәтә, экран белешмәсе файдаланылмаганга, 0 кыйммәтен куярга кирәк.

Хәбәр тәрәзе тибын һәм төймәләрнең рәвешен билгеләүче константалар түбәндәге таблицада китерелгән.

Таблица 6

Хәбәр типлары һәм төймәләре

Хәбәр тибы		
Константа	**Хәбәр**	**Тамга**
MtWarning	Игътибар	
MtError	Хата	
MtInformation	Информация	
mtConfirmation	Раслау таләбе	
MtCustom	Кулланучы сайлагы	-
Төймәләр рәвеше		
Константа	**Тамга**	**Кайтарылу коды**
MbYes	Әйе	mrYes
MbNo	Юк	mrNo
MbOk	Ok	mrOk
MbCancel	Баш тарту	mrCancel
MbHelp	Ярдәм	mrHelp
MbAbort	Тәмамлау	mrAbort
MbRetry	Кабатлау	mrRetry
MbIgnore	Калдырып үтү	mrIgnory

MbAll	Барысы	mrAll

Төймәләр җыелмасы булган диалог тәрәзен чакырганда төймәләр константалары өтер аша квадрат җәяләр эчендә санап кителә. Нинди төймә активлашканын белү өчен диалог тәрәзе тарафыннан кайтарылган кыйммәтне **word** типлы үзгәрешлегә үзләштерергә кирәк.

Мисал:

«Yes», «No» һәм «Cancel» төймәләре булган диалог тәрәзен чакырырга һәм кайсы төймә басылганын анализларга кирәк булсын.

```
W := MessageDlg('Кайсы төймәгә басарга?', mtConfirmation,
        [mbYes, mbNo, mbCancel]);
Case W of
  mrYes: { Yes төймәсен басу белән бәйле гамәл};
  mrNo : {No төймәсен басу белән бәйле гамәл };
  mrcancel: { Cancel төймәсен басу белән бәйле гамәл };
end;
```

9. КЛАССЛАР

Класслар дип Object Pascal дә кырлары, методлары (ысуллары) һәм үзлекләре булган махсус типлар атала. Теләсә нинди башка тип кебек үк, класс реализациянең (гамәлгә ашыруның) объект дип аталучы конкрет (анык) нөхсәләре төзү өчен үрнәк булып кына тора. Аныклап китик, Object Pascal дән элгәр Turbo Pascal дә объектлар дип Object Pascal класслары белән күп уртак сыйфатлары булган типлар атала. Тик Object Pascal нең объект моделенә кертелгән җитди камилләштерүләр телне төзүчеләрне объектларны билгеләү өчен махсус «класс» термины кертергә мәҗбүр

итәләр. Сүз уңаенда, бу термин Си++ тән алынган. Элегрәк Turbo Pascal with Objects 7.0 системасында эшләнгән программалар белән ярашу өчен Object Pascal дә object тип-объекты сакланган, ул «иске» объектлы модельгә туры килә. Бу модельнең барлык мөмкинчелекләре классларга хас булганга, без аларны аерым карамабыз, «бушаган» *объект* терминын классның конкрет нөхсәсе реализациясен билгеләү өчен кулланачакбыз.

Классның башка типлардан мөһим аермасы: класс объектлары һәрвакыт бергә бирелә. Шуңа күрә объект-үзгәрешле асылда хәтернең динамик өлкәсенә күрсәткеч кенә булып тора.Тик башка күрсәткечләрдән аермалы буларак, объект билгеләгән әйберләргә сылтамада объект исеменнән соң «^» символын куллану тыела:

```
type
TMyClass = class(TObject) Field: Integer;
end;
var
MyClass: TMyClass;
begin
MyClass^.Field := 0; // Хата! Болай язарга кирәк:
MyClass.Field := 0;
end;
```

9.1 OBJECT PASCAL ДӘ ОЮПНЫҢ ТӨП ТӨШЕНЧӘЛӘРЕ

Класслар – программалаучылар катлаулы программалар төзүне гадиләштерү һәм сыйфатларын яхшырту өчен уйлап тапкан үзенчәлекле «әйбер». Югарыда әйтелгәнчә, класслар нигезендә инкапсуляция, мирас итеп алу һәм полиморфизм дип аталган өч нигез принцибы ята. Object

Pascal дә класс дип составында үзгәрешлеләр, функцияләр һәм процедуралар булган тел структурасы атала. Үзгәрешлеләр билгеләнешләренә карап **кырлар** яки **үзлекләр** дип атала. Классның процедуралары һәм функцияләре – **методлар, ысуллар**. **Ысуллар** – класс эчендә бирелгән, тасвирланган, кырлар белән операцияләр башкарырга каралган процедуралар һәм функцияләр**.** Класс составында ысулларны чакыру өчен кирәк булган мәгълүмат урнашкан махсус таблицага күрсәткеч бар. Ысуллар чакырылганда аларга, гадәти процедуралар һәм функцияләрдән аермалы буларак, ысулны чакырган объектка күрсәткеч тапшырыла. Шуңа күрә нәкъ ысулны чакырган объектның кырлары эшкәртелә. Ысул эчендә ысулны чакырган объектка күрсәткеч махсус self исеме астында була.

Терминология. Объектка юнәлтелгән программалауда кырлар дип күренү өлкәсе public булган элементларны атау каралган. Элементлар шулай ук атрибут дип атала. Ысулларны операцияләр дип тә атыйлар**.**

9.2. ИНКАПСУЛЯЦИЯ

Класс өч төшенчә – кырлар, ысуллар һәм үзлекләр берлеген гәүдәләндерә. Бу затларны бер бөтенгә берләштерү инкапсуляция дип атала. Инкапсуляция классны программаның башка өлешләреннән аерырга мөмкинлек бирә, аны конкрет мәсьәлә чишү ихтыяҗларын канәгатьләндererлек итә. Нәтиҗә буларак, класс һәрвакыт үзендә бертөрле функциональлек йөртә. Мәсәлән, tForm классы үз составында Windows-тәрәзәләр төзү өчен кирәкле бар әйберне йөртә (инкапсуляцияли). TMemo классы – тулы функцияле текст редакторы, TTimer классы таймер белән эшне тәэмин итә, һ.б.

Объектка юнәлтелгән программалауның классик кагыйдәсе түбәндәгене раслый: ышанычлылыкны тәэмин итү өчен объект кырларына

турыдан туры керү, мөрәҗәгать итү кирәкми, аларны уку һәм яңарту кирәкле ысулларны чакыру ярдәмендә башкарылырга тиеш.

Бер типта кырларны һәм ул кырлар белән эшләү өчен ысулларны (функцияләрне) берләштерү инкапсуляция булачак.

Моннан тыш, инкапсуляция кырларга мөрәҗәгать итү турыдан-туры түгел (язмалар очрагында кебек), ә ысуллар ярдәмендә башкарылуын таләп итә.

Инкапсуляция – әзер программа өлешләре белән алмашу мөмкинчелеге бирә торган көчле чара. Delphi класслары тупламы – ул асылда Borland программалаучылары төзегән, сезнең программаларда куллану өчен әзер «кирпечләр» җыелмасы.

9.3. МИРАС ИТЕП АЛУ

Теләсә кайсы класс башка класстан төзелергә мөмкин. Моның өчен аны тасвирлаганда баба классның исеме күрсәтелә:

TChildClass = class (TParentClass)

Төзелгән класс автомат рәвештә үзенең баба классының кырларын, ысулларын, үзлекләрен мирас итеп ала һәм яңалары белән тулыландыра ала. Шуңа күрә мирас итеп алу принцибы катлаулы классларны бер-бер артлы төзүне һәм яңа класс тупламалары эшләүне тәэмин итә.

Object Pascal нең барлык класслары бердәнбер TObject баба классыннан, нәсел башыннан төзелгән. Бу классның кырлары һәм үзлекләре юк, ләкин аның барлык объектларның төзүдән башлап бетерүгә кадәр яшәү циклын тәэмин итүче гомуми ысуллары бар. Программалаучы TObject классының бала классы булмаган класс төзи алмый. Түбәндәге ике белдерү бердәй:

TaClass = class(TObject)

TaClass = class

Мисал. Программада ике класс төзелә. TStudent классы TPerson классының дәвамчысы, варисы, мирас итеп алучысы. TStudent классында curs дигән яңа атрибут бар.

program Project2;

{$APPTYPE CONSOLE}

Uses SysUtils;

Type

TPerson=class

name:String;

fam:String;

constructor Create(NewName:String;NewFam:String); Destructor Destroy;

function GetName:String; function GetFam:String;

end;

TStudent=class(TPerson)

curs:Integer;

constructor

Create(NewName:String;NewFam:String;NewCurs:Integer); function GetCurs:Integer; end;

constructor TPerson.Create (NewName:String;NewFam:String); begin

name:=NewName;

fam:=NewFam;

```
end;
function TPerson.GetName:String; begin
GetName:=name;
end;
function TPerson.GetFam:String; begin
GetFam:=fam;
end;
Destructor TPerson.Destroy ;
begin
end;
constructor
TStudent.Create(NewName:String;NewFam:String;NewCurs:Integer);
begin
name:=NewName;
fam:=NewFam;
curs:=NewCurs;
end;
function TStudent.GetCurs:Integer; begin
GetCurs:=curs;
end;
Var L:TPerson; S:String; Lena:TStudent; begin
L:=TPerson.Create('Leonard','Euler');
```

Lena:=TStudent.Create('Elena','Petrova',3);

S:=L.GetName;

Writeln(S,' ',L.GetFam);

Writeln(Lena.GetName, ' ', Lena.GetFam);

Readln;

L.Destroy; end.

1 нче кисәтү. Object Pascal дә класс нөсхәләре бары тик динамик кына була ала. Бу L үзгәрешлесе объект адресын үз эченә алучы күрсәткеч булып тора дигән сүз. Object Pascal дә классның берничә конструкторы була ала. Гадәттә конструкторга Create() исеме бирелә (Turbo Pascal дә конструкторның исеме Init(),C++тә конструктор исеме класс исеме белән бер). Детсрукторның гадәттәге исеме Destroy().

Объект аны инициаллаштыручы махсус ысул – конструкторны чакыру нәтиҗәсендә төзелә.

L:=TPerson.Create('Leonard','Euler');

Төзелгән нөхсәне башка ысул – деструктор ярдәмендә бетереп була:

L. Destroy;

2 нче кисәтү. Инкапсуляция кырларга мөрәҗәгать итү турыдан-туры түгел (язмалар очрагында кебек), ә ысуллар ярдәмендә башкарылуын таләп итә.

S:=L.GetName;

дип язу урынына

S:=L.Name;

дип язып, ягъни кырга турыдан-туры мөрәҗәгать итеп була. Без инкапсуляция шартларына игътибар итмәдек, гәрчә бу очракта хата юк.

3 нче кисәтү. Бала классының конструкторын түбәндәгечә күчереп язып була:

```
Constructor
TStudent.Create(NewName:String;NewFam:String;NewCurs:Integer);

begin

inherited Create(NewName,NewFam);

curs:=NewCurs;

end;
```

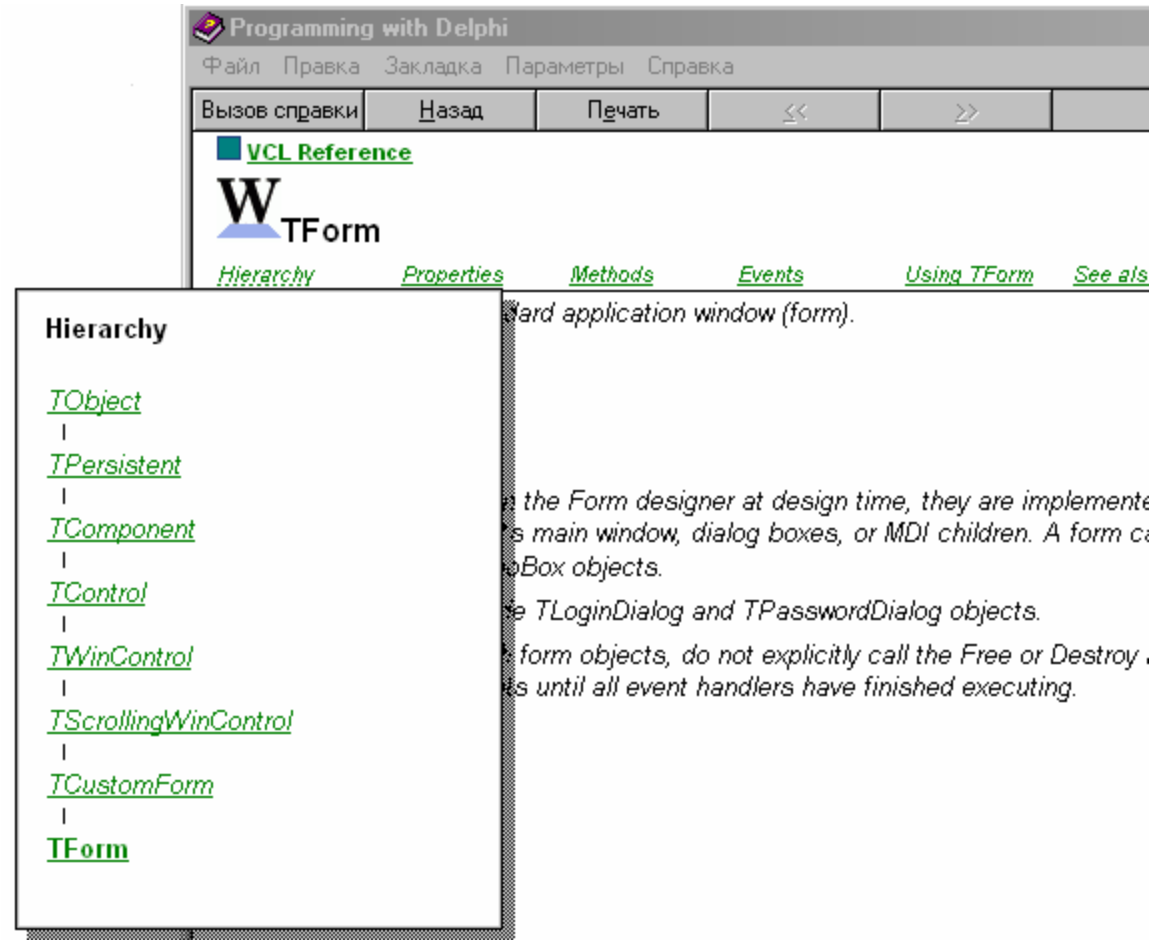
9.4. КЛАССЛАР ДИАГРАММАСЫ

Мирас итеп алу принцибы классларның тармакланган агачын төзүгә китерә. Ул нәсел башы TObject тан токымнарына күчкәндә эзлекле рәвештә үсә. Һәр токым үзенең бабасының мөмкинлекләрен яңалары белән тулыландыра һәм үзенең токымнарына тапшыра. Мисал өчен рәсемдә Delphi класслары агачының бер өлеше күрсәтелгән (7 нче рәсем). Tpersistent классы үзенең бабасы TObject ның мөмкинлекләрен баета: ул мәгълүматны файлда саклый һәм файлдан ала «белә», нәтиҗәдә аның токымнары, балалары да бу операцияләрне эшли ала. Үз чиратында TComponent классы төзүче мөхите белән үзара тәэсир итешә ала һәм бу «сәләтен» токымнарына, балаларына тапшыра. TControl файллар һәм төзүче мөхите белән генә эшләп калмый, ул экранда күренә торган сурәтләрне дә булдыра һәм аларга хезмәт күрсәтә ала, ә аның токымы TWinControl Windows-тәрәзәләр төзи ала һ.б.

Бала класста баба класстан мирас итеп алган кырлар һәм ысуллар белән эш мөмкин; әгәр ысул исемнәре тәңгәл килсә, алар бер-берсен каплый дип әйтәләр.

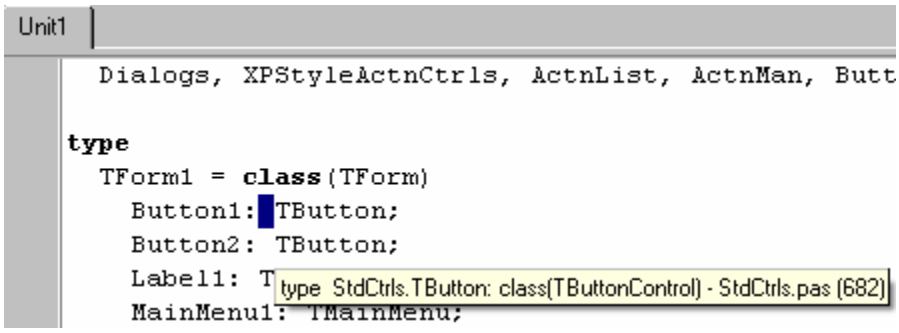
Чакырганда нинди гамәлләр үтәлүенә карап, ысуллар өч төркемгә бүленә. Беренче төркемгә статик ысуллар, икенчегә – виртуаль һәм динамик, өченчегә – Delphi 4 тән башлап кертелгән яңадан йөкләнә ала торган (overload) ысуллар керә.

Беренче төркем ысуллары бала классларда яңадан билгеләнгәндә тулысынча капланалар. Бу очракта ысулны игълан итүне тулысынча үзгәртеп була. Икенче төркем ысуллары мирас итеп алынганда атамаларын һәм типларын сакларга тиеш. Яңадан йөкләнә торган ысуллар мирас итеп алу механизмына куллану шартларына карап ысулның кирәкле вариантын (үзенеке яки бабаныкы) сайлау мөмкинлеген өстиләр.

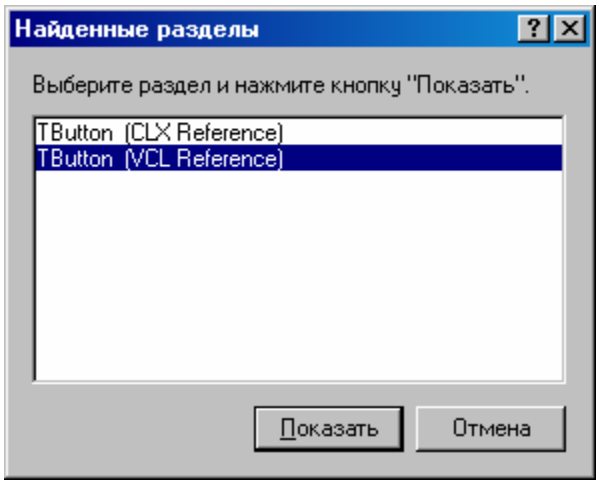


17 нче рәс. Мирас итеп алу диаграммасы мисалы. Delphi белешмә хезмәтеннән.

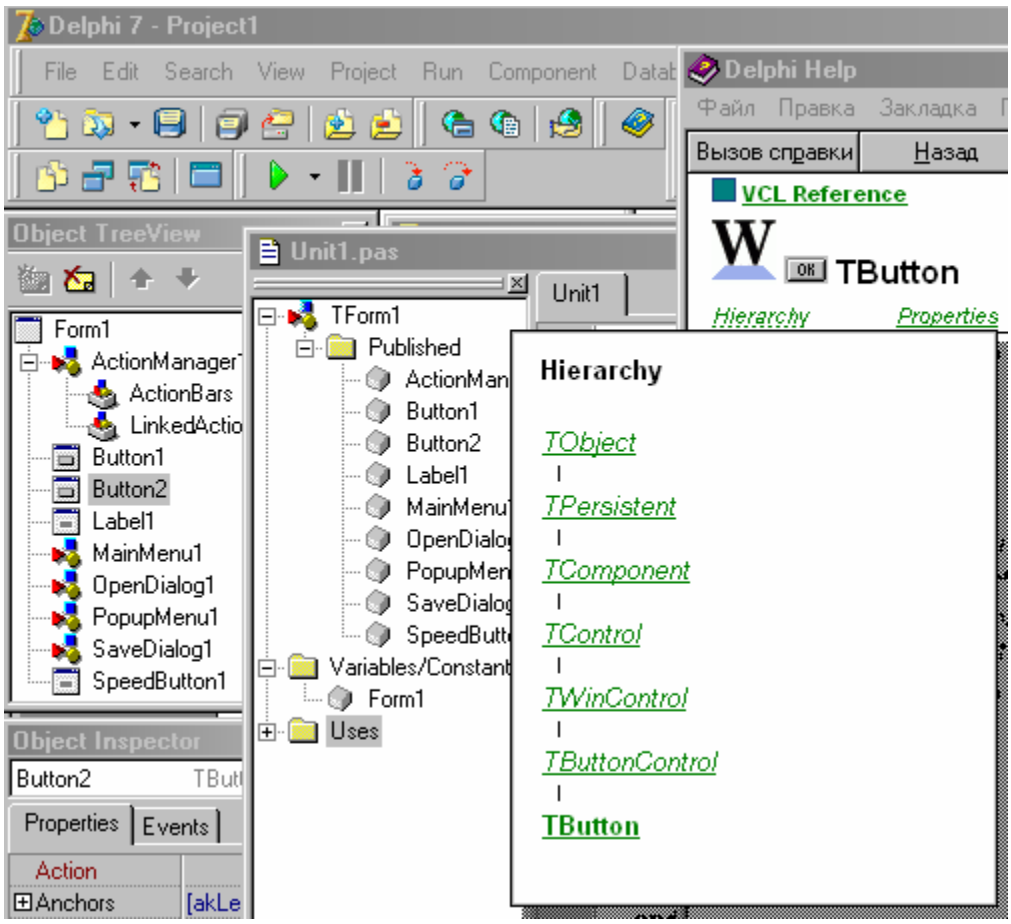
Класс турында белешмә, мәгълүмат алу өчен курсәрне класс исеменә куярга кирәк.



<F1> төймәсенә басарга



VCL Reference ны сайларга һәм «Күрсәтергә» төймәсенә чиртергә



9.5. КЛАССНЫ ИГЪЛАН ИТҮ

Теләсә нинди яңа төзелә торган классның махсус сүзләр: published (басылган, дөнья күргән), private (ябык), protected (сакланган), public (ачык) һәм automated (автоматлаштырылган) белән билгеләнгән бүлекләре булырга мөмкин. Һәр бүлекнең эчендә башта кырларны, аннары ысуллар һәм үзлекләрне билгелиләр.

Бүлекләр класс тасвирлау элементларының күренүчәнлек өлкәләрен билгелиләр. Public бүлеге анда санап үтелгән кырлар, ысуллар һәм үзлекләрнең күренүчәнлек өлкәсенә чикләүләр куймый, аларны программаның теләсә нинди башка модулендә чакырып була. Published бүлеге шулай ук күренүчәнлек өлкәсенә чикләү куймый, тик анда башкару этабында гына түгел, төзү этабында да (ягъни Объект инспекторы тәрәзәсендә) мөрәҗәгать итеп була торган үзлекләр санап үтелә. Published бүлеге стандарт булмаган компонентлар эшләгәндә генә кулланыла. Delphi

мөхите формага куелган компонентларның тасвирламаларын класс бүлеге башламасыннан соң урнашкан һәм беренче игълан ителгән бүлеккә кадәр дәвам иткән исемсез махсус бүлеккә урнаштыра. Бу – published бүлеге. Программалаучыга ул бүлеккә үзенең класс тасвирлау элементларын урнаштырырга яки ул бүлектән мохит тарафыннан куелган элементларны алып атарга кирәкми. Private бүлеге күренүчәнлек өлкәсен минимумга кадәр кыса: тасвирлауның ябык элементларына бирелгән класс ысуллары эчендә һәм класс тасвирланган модульдәге астпрограммаларда гына мөрәҗәгать итеп була. Private бүлегендә игълан ителгән элемент башка модульдә урнашкан классның иң якын токымнары өчен дә мөрәҗәгать итеп булмый торганга әйләнә. Protected бүлеге классның үз ысулларына һәм кайсы модульдә урнашканга бәйсез рәвештә барлык токымнарына ачык. Ниһаять, automated бүлеге автоматлаштыруның OLE-объектлар интерфейсына өстәләчәк үзлекләр һәм ысулларны игълан итү өчен генә кулланыла; бу бүлек әгъзаларының күренүчәнлек өлкәсе чикләнмәгән.

В Object Pascal дә теләсә кайсы бүлекне күпме кирәк, шуның кадәр тапкыр игълан итеп була, бүлекләрнең бер-бер артлы урнашу тәртибе мөһим түгел. Теләсә кайсы бүлек буш була ала.

Түбәндәге код фрагменты күренүчәнлек өлкәләрен аңлата.

```
Unit Unit1;

Interface

Uses Controls, Forms;

type

TForm1 = class(TForm)

Button1: TButton; // Бу бүлекне Delphi эшли

// Аның элементлары һәркемгә ачык
```

// Бу бүлек Unit1 модулендә ачык

private

FIntField: Integers

Procedure SetValue(Value: Integer);

Function GetValue: Integer;

published

// Бу бүлек теләсә кайсы модульдә ачык

Property IntField: read GetValue write SetValue;

protected // Бу бүлек бала классларга ачык

Procedure Proc1;

public // Бу бүлек теләсә кайсы модульдә ачык

Procedure Proc2;

end;

var Formi: TForm1;

Implementation Procedure TFormI.Proc1 ;

Button1.Color := clBtnFace;

// Болай ярый

FIntField := 0;

// Болай ярый

IntField := 0;

// Болай ярый *Proc1;*

// Болай ярый *Proc2;*

// Болай ярый

end;

begin

Form1.Button1.Color := clBtnFace; // Болай ярый

Form1.FIntField := 0; // Болай ярый

Form1.IntField := 0; // Болай ярый

Form1.Proc1; // Болай ярамый!

Form1.Proc2; // Болай ярый

end.

Unit Unit2;

Interface

Uses Controls, Unit1;

type

TForm2 = class(TForm1) Button2: TButton;

Procedure Button2Click(Sender: TObject);

end;

var Form2: TForm2;

Implementation

Procedure TForm2.Button2Click(Sender: TObject);

begin

Button1.Color := clBtnFace; // Болай ярый

FIntField := 0; // Болай ярамый!

IntField := 0; // Болай ярый

Proc1; // Болай ярый

Proc2; // Болай ярый

end;

begin

Form1.Button1.Color := clBtnFace; // Болай ярый

Form1.FIntField := 0; // Болай ярамый!

Form1.IntField := 0; // Болай ярый

Form1.Proc1; // Болай ярамый!

Form1.Proc2; // Болай ярый

end.

Бала классны игълан иткәндә класс элементларын бер күренүчәнлек өлкәсеннән икенчесенә күчерергә рөхсәт ителә. Югарыдагы мисал өчен мондый игълан итү мөмкин:

type

TForm2 = class(Tform1)

Public

Procedure Proc1;

end;

Моннан соң unit2 модулендә мондый мөрәҗәгать мөмкин:

Form2.Proc1;

Private бүлегенә күчерелгәннән соң тасвирлау элементы токымнарга күренми (әгәр токым, күп очрактагыча, башка модульдә игълан ителсә). Шуңа аны башка бүлеккә күчереп булмый.

Класс бары тик модульнең интерфейс өлкәсендә яки башкару бүлеге башында ук игълан ителә ала. Классны астпрограммалар тасвирлау бүлегендә билгеләргә ярамый.

Искәрмә. Өч күренүчәнлек өлкәсе – private, protected, public – ысуллар күренүчәнлеге үсү тәртибендә урнашкан. Токым классларда ысуллар һәм үзлекләрнең күренүчәнлеген арттырып була, киметеп булмый. Бала классны тасвирлаганда ысуллар һәм үзлекләрне бер күренүчәнлек өлкәсеннән икенчесенә күчерү өчен аларны күчереп язу, тасвирлау кирәк түгел, башка урында искә алу җитә.

9.6. ВИРТУАЛЬ ЫСУЛЛАР ҺӘМ ПОЛИМОРФИЗМ

Югарыда әйтелгәнчә, Delphi ның хәзерге вакытта кулланылучы версияләрендәге «яңа» объект моделендә һәр класс күрсәткеч тасвирламасы булып тора һәм барлык объектлар да динамик. Объектларны хәтердә урнаштыру белән класс конструкторы шөгыльләнә. Баба конструкторны чакыру өчен **inherited** инструкциясе кулланыла.

9.7. СТАТИК ҺӘМ ВИРТУАЛЬ ЫСУЛЛАР

Ысуллар статик һәм виртуаль була ала. Тышкы аерма шунда: виртуаль ысулның тасвирламасы башламыннан соң virtual ачкыч сүзе күрсәтелә, мәсәлән,

Procedure PushPet;virtual;

Виртуаль ысулның статик ысулдан аермасы: объект тибы нөхсәсе белән статик ысул арасында бәйләнеш компиляция вакытында урнаша, нөхсә белән виртуаль ысул арасында бәйләнеш программа башкарылу вакытында урнаша.

Статик ысулны чакыруны компиляция вакытында рөхсәт итү процессы иртә бәйләү дип атала. Соң бәйләү виртуаль ысуллар өчен кулланыла.

Әгәр баба класста ысул виртуаль дип игълан ителсә, теләсә кайсы бала класстагы шул ук исемдәге ысуллар виртуаль була. Һәр классның виртуаль ысуллар таблицасы бар (таблица виртуальных методов – ВМТ яки VMT). Бу таблицада бирелгән типка (класска) бүленгән үлчәм һәм һәр виртуаль ысул өчен бу ысулны реализацияләүче код күрсәткече бар. Конструктор нөхсә (объект) белән виртуаль ысуллар таблицасы арасында бәйләнеш урнаштыра. Һәр класс өчен виртуаль ысуллар таблицасы бер генә.

1 нче мисал:

```
program Pets2;
{$APPTYPE CONSOLE}
Type
TPet=class
Name:String;
constructor Create(NameI:String);
procedure Speak;virtual;
procedure PushPet;virtual;
end;
Type
TDog=class(TPet)
procedure Speak;override;
```

```
end;
Type
TCat=class(TPet)
procedure Speak;override;
end;
Type
TBird=class(TPet)
procedure Speak;override;
end;
constructor TPet.Create(NameI:String);
begin
Name := NameI;
end;
procedure TPet.Speak;
begin
writeln('!!!!!!!');
end;
procedure TDog.Speak;
begin
writeln('Gav!');
end;
procedure TCat.Speak;
```

```
begin
writeln('May!');
end;
procedure TBird.Speak;
begin
writeln('Kar!!');
end;
procedure TPet.PushPet;
begin
{ .... .... .... }
writeln('Pet:',Name);
Speak;
end;
Var
Bob : TDog; Boss : TCat; Vorona : TBird;
Begin
Bob:= TDog.Create('Bobik');
Boss:= TCat.Create('Bossik');
Vorona:= TBird.Create('Karkusha');
Boss.PushPet;
Vorona.PushPet;
readln;
```

End.

Искәрмә. Яңа модельдә ысулларны яңабаштан билгеләү override директивасы ярдәмендә бирелә (virtual түгел).

Исемнәре бер виртуаль ысуллар туган классларда төрле гамәлләр башкара. ОЮП да мондый мөмкинлек полиморфизм дип атала.

Полиморфизм – ул классларның мәгънәләре охшаш проблемаларны төрле ысуллар белән чишү үзлеге. Object Pascal дә классның үз-үзен тотышы үзлекләре аңа керүче ысуллар белән билгеләнә. Токым класста теге яки бу ысулның алгоритмын үзгәртеп, программалаучы бу токымнарга баба класста булмаган аерым үзлекләр бирергә мөмкин. Ысулны үзгәртү өчен токымда аны капларга, ягъни токымда шул ук исемдәге ысулны игълан итәргә һәм анда зарур гамәлләрне башкарырга кирәк. Нәтиҗәдә баба объектта һәм бала объектта ике бер исемдәге ысул эшләячәк, аларның алгоритмик нигезе төрле булачак, алар объектларга төрле үзлекләр бирәчәкләр. Бу объектлар полиморфизмы дип атала.

Object Pascal дә полиморфизм югарыда тасвирланган баба объект ысулларын мирас итеп алу һәм каплау механизмы ярдәмендә генә түгел, ә бәлки, күрсәтелгәнчә, баба ысулларына бала ысулларын чакыру мөмкинлеге бирүче виртуализация кулланып та үтәлә. 1 нче мисалны аңлатучы рәсемне карыйк:

Бер классның барлык объектлары өчен виртуаль ысуллар таблицасы бер.

2 нче мисал.

```
program Project1;
{$APPTYPE CONSOLE}
uses
SysUtils;
Type
    Transport=class
Name:String;
constructor Create(NameI:String);
procedure PlaySignal; virtual;
procedure Info; virtual;
end;
Type
    TVelo=class(Transport)
wheels:Integer;{Число колес}
constructor Create(NameI:String;Wh:Integer);
procedure PlaySignal;override;
end;
Type
    TMoto=class(Transport)
```

```
Cylinders:Integer;
constructor Create(NameI:String;Cyl:Integer);
procedure PlaySignal;override;
end;
Type
    TAuto=class(Transport)
doors : Integer;
constructor Create(NameI:String;d:Integer);
procedure PlaySignal;override;
end;
constructor Transport.Create(NameI:String);
begin
Name := NameI;
end;
constructor TVelo.Create(NameI:String;Wh:Integer);
begin
Name := NameI;
wheels := Wh;
end;
constructor TMoto.Create(NameI:String;Cyl:Integer);
begin
Name := NameI;
```

```
Cylinders:=Cyl;
end;
constructor TAuto.Create(NameI:String;d:Integer);
begin
Name := NameI;
doors := d;
end;
procedure Transport.PlaySignal;
begin
writeln('!!!!!!!');
end;
procedure TVelo.PlaySignal;
begin
writeln('Din Din');
end;
procedure TMoto.PlaySignal;
begin
writeln('Bap Bap!');
end;
procedure TAuto.PlaySignal;
begin
writeln('Bee Bip ');
```

```
end;
procedure Transport.Info;
begin
writeln('Transport: ',Name);
PlaySignal;
end;
Var p : Array [1..4] of Transport;
i : Integer;
begin
p[1]:= TVelo.Create('Sputnik',2);
p[2]:= TMoto.Create('Ural',2);
p[3]:= TAuto.Create('Lada',4);
p[4]:= TAuto.Create('Niva',2);
for i := 1 to 4 do
p[i].Info;
readln;
end.
```

9.8. БЕРНИЧӘ КОНСТРУКТОРЛЫ КЛАССЛАР

Берничә конструкторлы класс үрнәге итеп сызыкча булмаган тигезләмәләрне урталай бүлү ысулы, итерацияләр ысулы һәм Ньютон ысулы кулланып чишүне карыйк.

```
program Project4_4_1;
{$APPTYPE CONSOLE}
```

{Тигезләмәләр чишү өчен класс}

Const

MaxN=1000; {Ысул адымнарының ның максималь саны}

Type TFunct=function(x:double):double;

Type

TEquation=class

f: TFunct; {f(x)=0}

g: TFunct;

{x-g(x)=0 һәм f(x)=0 эквивалент}

Df: Tfunct; { f(x) функциясенең чыгарылмасы}

a,b: double; {кисемтә чикләре}

x0: double; {тамырның якынча кыйммәте}

eps: double; {исәпләү төгәллеге}

N: Integer; {Якынайту ысулы адымнары саны}

{барлык ысуллар өчен бер уртак конструктор}

constructor Create(f_,g_,Df_:TFunct;

a_,b_,x0_,eps_:double);overload;

{урталай бүлү ысулы өчен конструктор }

constructor Create(f_:TFunct;

a_,b_,eps_:double);overload;

{итерацияләр ысулы өчен конструктор}

constructor Create(f_,g_:TFunct;

```
x0_,eps_:double);overload;
{Ньютон ысулы өчен конструктор}
constructor Create(x0_,eps_:double;
f_,Df_:TFunct);overload;
destructor Destroy;
function Bisect:double;
function Iterat:double;
function Newton:double;
function GetN:Integer;
end;
constructor TEquation.Create(f_,g_,Df_:TFunct;
a_,b_,x0_,eps_:double);
{уртак конструктор}
begin
f:=f_; g:=g_; Df:=Df_;
a:=a_; b:=b_; x0:=x0_; eps:=eps_;
end;
constructor TEquation.Create(f_:TFunct;
a_,b_,eps_:double);
{урталай бүлү ысулы өчен конструктор }
begin
f:=f_;
```

```
a:=a_; b:=b_; eps:=eps_;
end;
constructor TEquation.Create(f_,g_:TFunct;
x0_,eps_:double);
{итерацияләр ысулы өчен конструктор}
begin
f:=f_; g:=g_;
x0:=x0_; eps:=eps_;
end;
constructor TEquation.Create(x0_,eps_:double;
f_,Df_:TFunct);
{Ньютон ысулы өчен конструктор}
begin
begin
f:=f_; Df:=Df_;
x0:=x0_; eps:=eps_;
end;
destructor TEquation.Destroy;
begin
end;
function TEquation.Bisect:double;
{урталай бүлү ысулы}
```

```
Var u, v, f1, f2 : double;
begin
N := 1;
u := a; v := b;
f1 := f(u);
repeat
x0 := (u + v) * 0.5;
f2 := f(x0);
if f2 = 0 then break;
if f1*f2 >0 { . . f1 f2 }
then
begin
u := x0; f1 := f2;
end
else v := x0;
Inc(N);
until (v - u) < eps;
Result := x0;
end; {Bisect}
function TEquation.Iterat:double;
{итерацияләр ысулы}
Var x,t:double;
```

```
begin
N := 1;
repeat
x := g(x0);
t := abs(x-x0);
Inc(N);
x0 := x;
until (t<eps) Or (N>MaxN);
Result := x;
end;{Iterat}
function TEquation.Newton:double;
{Ньютон ысулы}
Var t,x:double;
begin
N := 1;
repeat
x := x0-f(x0)/df(x0);
t := abs(x-x0);
Inc(N);
x0 := x;
until (t<eps) Or (N>MaxN);
Result := x;
```

```
end; {Newton}

function TEquation.GetN:Integer;

begin

Result := N;

end;

{Тигезләмә мисалы}

function f(x:double):double;

{тигезләмәнең сул ягы}

begin

Result := x*x*x - 2*x - 5;

end;

function g(x:double):double;

{x=g(x) һәм f(x)=0 бердәй, бертөрле көчле}

{итерацияләр ысулында кулланыла}

begin

Result := -x*x*x*0.04 + 1.08*x + 0.2;

end;

function df(x:double):double;

{f(x) функциясе чыгарылмасы}

{Ньютон ысулында кулланыла}

begin

Result := 3*x*x - 2;
```

```
end;

Var Eq: TEquation;

Begin

Eq := TEquation.Create(f,2,3,0.000001);

Writeln(' Bisect ',

Eq.Bisect,' ',Eq.GetN,' ');

Eq := TEquation.Create(f,g,2.5,0.000001);

Writeln(' ',

Eq.Iterat, ' ',Eq.GetN,' ');

Eq:= TEquation.Create(2.5,0.000001,f,df);

Writeln(' ',

Eq.Newton,' ',Eq.GetN,' ');

Readln; {тоткарлау}

End.
```

Бу мисалда берничә конструктор кулланыла, өстәвенә шул мөһим, аларның исемнәре бертөрле. Delphi да бертөрле исемдә бердән артык функция яки процедура игълан итәргә ярый. Мондый функцияләр яңадан йөкләнгән (overloading) дип атала. Бу очракта функцияләрне overload директивасы белән тәэмин итәргә кирәк, ул функция башламының иң ахырында урнаштырыла, нокталы өтер белән аерылып алына. Функцияләрнең сигнатуралары төрле булырга тиеш. Бу функцияләрнең параметрлары исемлегендә параметрлар саны яки параметр типлары исемлеге төрле булырга тиеш дигән сүз. Гадәти функцияләр һәм процедуралар да, класс ысуллары да яңадан йөкләнгән була ала.

ӘДӘБИЯТ

1. Немцова Т.И., Голова С.Ю., Абратов И.В. Программирование на языке высокого уровня. Программирование на языке Object Pascal: учебеное пособие / под ред. Л.Г.Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М. 2015. – 496 с.

2. Фаронов В. В. Delphi : программирование на языке высокого уровня : учеб. для студ. / Санкт-Петербург: Питер, 2005 . – 640 с.

3. Бобровский С. И. Delphi 7 : учеб. курс/ Санкт-Петербург: ПИТЕР, 2005 . – 736 с.

4. Иванова Г. С. Объектно-ориентированное программирование : учеб. для студ. / Москва: Изд-во МГТУ им. Н. Э. Баумана, 2003 . – 368 с.

5. Программирование на языке Delphi [Электронный ресурс] URL: http://rsdn.ru/?article/Delphi/Delphi_7_07.xml (дата обращения 12.05.2016)

6. Галимҗангов Ә.Ф., Миңнегалиева Ч.Б. Объектка ориентлашкан программалауга кереш: уку әсбабы /А.Ф. Галимянов, Ч.Б. Миннегалиева – Казань: Казан. ун-т, 2016.- 141 с.