ЎЗБЕКИСТОН РЕСПУБЛИКАСИ ОЛИЙ ВА ЎРТА МАХСУС ТАЪЛИМ ВАЗИРЛИГИ

БУХОРО ОЗИҚ - ОВҚАТ ВА ЕНГИЛ САНОАТ ТЕХНОЛОГИЯСИ ИНСТИТУТИ

“Касбий таълим” факултети

«Информатика ва ахборот технологиялари» кафедраси

«ОБЪЕКТГА МЎЛЖАЛЛАНГАН ДАСТУРЛАШ»
ФАНИДАН

ТАЖРИБА ИШЛАРИНИ БАЖАРИШ БЎЙИЧА УСЛУБИЙ ҚЎЛЛАНМА

БУХОРО – 2010 й.

Услубий қўлланма муаллифи: «Ахборот технологиялари» кафедраси
к. ўқ. Нарзиев У.З.

Услубий қўлланма Бухоро озиқ овқат ва енгил саноат технологияси институти «Касбий таълим» факултети 5140900 Касб таълими «Информатика ва ахборот технологяилари» таълим йўналиши талабаларига «Объектга мўлжалланган дастурлаш» фанидан тажриба машғулотларни ўтказиш учун мўлжалланган.

Мундарижа

Тажриба №1.

Delphi ни ўрнатиш ва муҳит билан танишиш. Ўрнатиш усуллари.

Delphi тизими тўғри ўрнатилиши ва нормал ишлаши учун сизнинг компютерингизнинг техник ҳолати, унинг тизимли, дастурий воситалари қуйидаги талабларга жавоб бериши керак. Қуйидаги жадвалда Delphi иши учун зарур бўлган тизимли, техник ҳамда дастурий воситалар тавсифи келтирилган.

Delphi нинг техник талаблари	
Компютер	Intel® ёки унга монанд бошқаси Pentium 166 MHz ёки ундан юқори.
Оператив хотира(RAM[1])	Enterprise Edition: 16 MB minimum, 64 MB, юқорироғи тавсия қилинади. Standard Edition: минимум 64 MB Personal Edition: Windows 2000 учун минимум 64 MB, бошқа ОСлар учун минимум 32 MB . Developer Edition: минимум 64 MB Desktop Engine: Windows 2000 учун минимум 64 MB , бошқа ОС лар учун минимум 32 MB .
Қаттиқ диск ҳажми (HDD)[2]	Delphi components: 95 гача 270 MB, 250 MB typical . Analysis Services: минимум 50 MB, 130 MB typical . English Query: 80 MB Desktop Engine only: 44 MB
Монитор	VGA 800x600 дан юқориси тавсия этилади .
Бошқариш қурилмаси	Microsoft Mouse ёки шунга мувофиғи
CD-ROM drive	тавсия этилади.
Номи	**Операцион тизимга қуйиладиган талаблар**
Delphi	Microsoft Windows 98, Microsoft Windows NT, Windows 2000, Windows XP

Диққат!

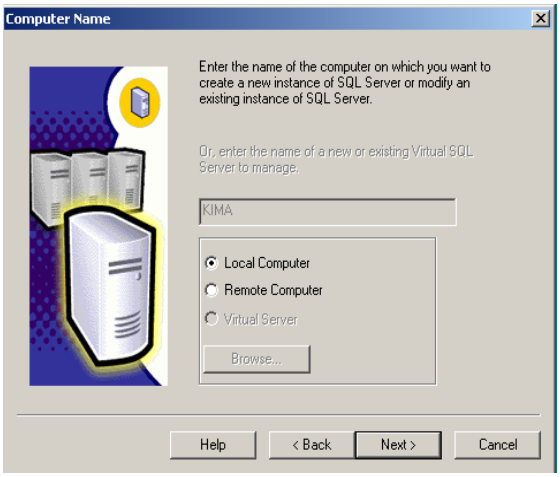
Сизнинг тизимингизда ҳар кунги **сана ва вақт** тўғри белгиланиб бориши муҳим аҳамиятга эгадир – бу маълумотларни жўнатиш ҳамда уларни қайд этишда юз бериши мумкин бўлган хатоликлар олдини олади.

Delphiни ўрнатиш

1. Borland Delphi 6.0 ни ўрнатиш учун унинг юкловчи диски (загрузочный диск) ёки юкловчи файли (загрузочный файл) керак бўлади. Ўрнатиш файлини ишга туширганингиздан кейин куйидаги ойна ҳосил бўлади.

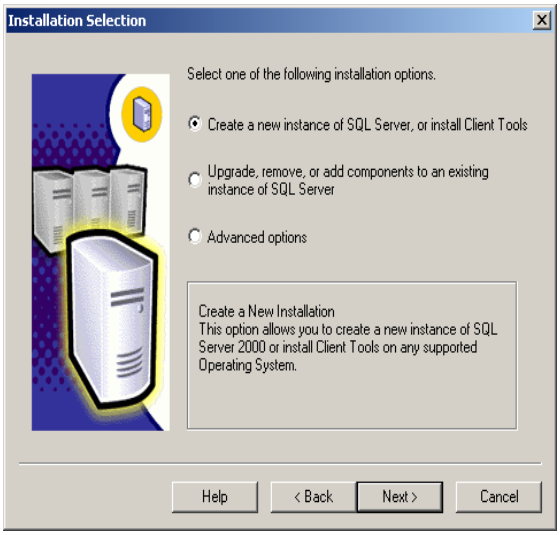
2. Delphi 6’ бандини танланг.
3. Пайдо бўлган навбатдаги ойнада **Next** тугмачасини босинг.

4. Қуйидаги ойна ҳосил бўлади:



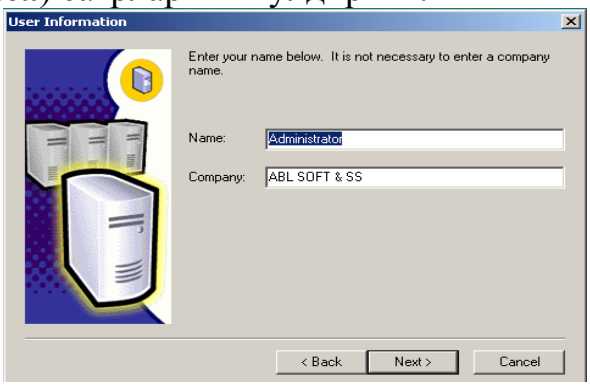
Local Computer'ни танлаб, **OK** тугмачасини босинг.

5. Навбатдаги ойна пайдо бўлади:



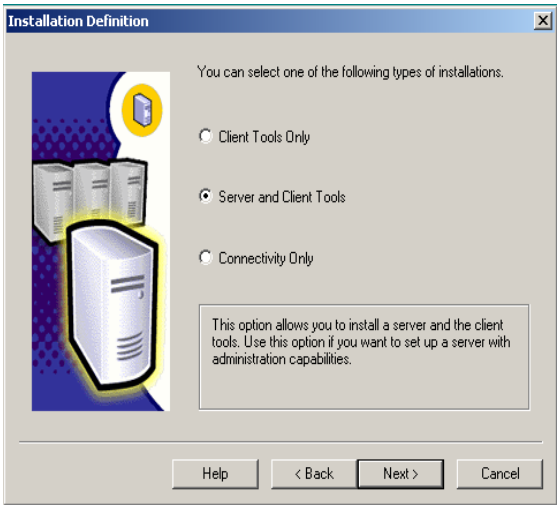
Бу ойнада **Create a new instance of SQL Server, or install Client Tools** бандини танлаб, **Next** тугмачасини босинг.

6. Ҳосил бўлган ойнада, агар лозим топсангиз, **Name (Ном)** ва **Company (Компания - муассаса)** сатрларини тўлдиринг.



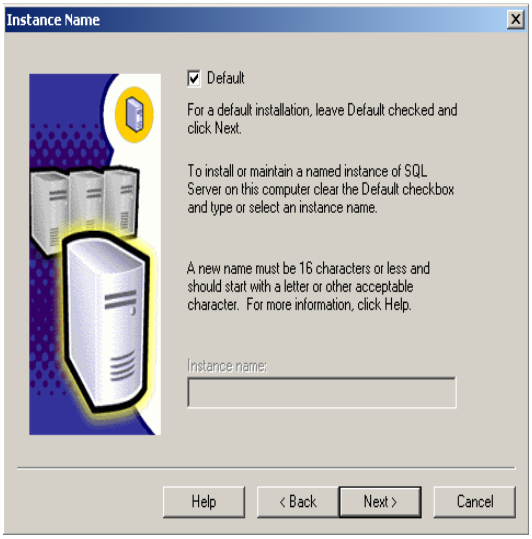
7. Навбатдаги ойнада сиз лицензия битимини тасдиклашингиз керак бўлади. Ишни давом эттириш учун **Yes** тугмачасини босинг.

8. Кейинги ойнадан



Server and Client Tools бандини танланг ва **Next** тугмачасини босинг.

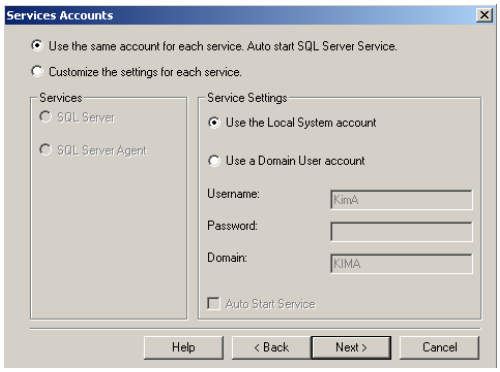
9. Навбатдаги ойнанинг кўриниши куйидагича :



Сизнинг компьютерингизда **Delphi 6** ўрнатилмаганлиги учун ушбу амалларни бажармокдасиз. Шу боис **Default** дарчасини белгиланган ҳолида қолдиринг, сўнгра **Next** тугмачасини босинг.

10. Ҳосил бўлган куйидаги ойнадан Typical бандини танлаб Next тугмачасини босинг.

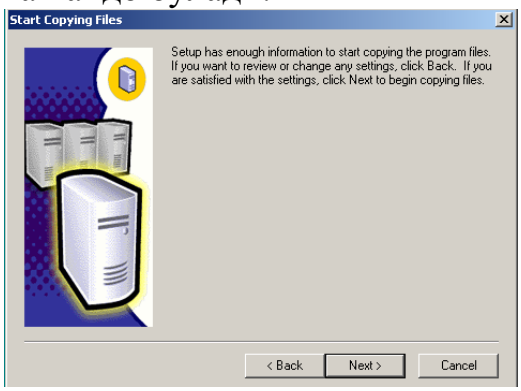
11. Пайдо бўлган навбатдаги ойнада



Use the same account for each service. банди танлаган бўлиши керак. Пастдаги **Service Settings** бўлимидан **Use the Local System account** бандини танланг.

12. Ҳосил бўлган ойнадан **Next** тугмачасини босинг.

13. Навбатдаги ойна пайдо бўлади:



Delphi 6 файлларни кўчиришни бошлаш учун **Next** тугмачасини босинг.

14. Файллар кўчириб ўтказиш ниҳоясига етгач, компютерни қайта юкланг (перезагрузка).

15. Шу ишларнинг барчасини бажариб чиккандан сўнг тизим сизнинг компютерингизга ўрнатилади.

Саволлар

1. Delphi 6 нинг тизим заҳираларига талаблари.
2. Delphi 6ни ўрнатиш усуллари.

Топшириқлар:

1. Delphi дастурини турли усуллар билан ўрнатиш.

ТАЖРИБА ИШИ №2

Дастлабки лоиҳалар. Дастур матнини кўриш.

1. Янги дастур киритиш учун таҳрирлашнинг янги ойнаси очилади.
2. Янги дастур матни терилади.
3. Дастур матни дискка езилади (янги дастур матни ишга туширилгунига қадар дискка езилиши зарур).
4. Дастур бажаришга юборилади (Ctrl + F9).
5. Агар дастурда синтаксис хатоларга йўл қўйилган бўлса, экранда тегишли ахборот пайдо бўлади, курсор эса хато жойни кўрсатади. Бундай пайтда таҳрир буйруқлари ёрдамида тузатилиши ва яна дастур бажаришга юборилиши керак.
6. Дастур натижаларини кўриш (Alt + F5).
7. Агар хато натижалар олинган бўлса, йўл қўйилган алгоритмик хатоларни тузатиш ва дастурни яна бажаришга юбориш керак.
8. 4 - 7 - босқичлар тўғри ечимлар олингунча такрорланади.
9. Тузатилган дастур дискда сақланади (F2).

Дастур матни бўлаги билан ишлаш клавишлари.

Shift + "йўналиш клавишлари" - дастур бўлагини ажратади.

Shift + Del - дастурнинг ажратилган бўлагини ўчиради ва Clipboard буферига жойлаштиради.

Ctrl + Ins - дастурнинг ажратилган бўлагини Clipboard буферига нусхалайди.

Shift + Ins - Clipboard даги ажратилган бўлакни ойнадаги курсор жойлашган хонага қўяди.

Ctrl+Del - ажратилган бўлакни Clipboard буферига узатмасдан ўчиради.

Alt + BackSpace - таҳрирлашдаги охирги ишни бекор қилади.

Компиляция ва бажаришга юбориш клавишлари.

Alt + F9 - таҳрирлашнинг фаол ойнасидаги файлни компиляция қилиш.

F9 – .exe файлини яратиш билан кўп модулли дастурни шартли компиляция қилиш. Агар охирги компиляция вақтидан баъзи бир модулларга ўзгартиришлар киритилган бўлса, фақат ўзгартирилган ва уларга боғлиқ бўлган модулларгина қайта компиляция қилинади. Оддий дастурлар учун шунга тенг кучли **Alt + F9** босилади.

Ctrl+F9 - таҳрирлашнинг актив ойнасидаги дастурни бажаришга юбориш.

Дастурларни тузатиш клавишлари.

Alt + F5 - дастур бажарилиши натижаларини кўриш.

F8 - дастурнинг қадамба-қадам бажарилиши. Процедура ва функцияларни чақириш битта оператор (битта қадам) каби бажарилади.

F7 - дастурни қадамба-қадам бажариш. Процедура еки функцияларни чақиришда унинг матнига кириш ва операторларни қадамба - қадам бажариш содир бўлади.

F4 - дастур қадамба-қадам бажарилишининг жорий сатридан курсор жойлашган сатргача дастур қисмининг бажарилиши.

Ctrl + F2 - дастурни таҳрир қилиш ишини якунлайди ва уни хотирадан бўшатади.

Ctrl + F3 - Call Stack ойнасини очади. Бу ойнада шу дақиқада бажарилаетган процедурагача чиқарилган дастур процедуралари рўйхати кетма-кетлиги кўрсатилади.

Ctrl + F4 - Evaluate and modify ойнасини очади. Бу ойнада қиймати аниқланиши талаб этилган ифодани кўрсатиш, дастур ўзгарувчилари қийматларини ва берилганлар элементларини қараб чиқиш ҳамда уларни ўзгартириш керак.

Ctrl + F7 - Add Watch мулоқот ойнани очади. Бу ойнага дастурчи тузатишни бажариш вақтида, қийматлари қизиқтирадиган ифода ёки ўзгарувчи номини кўрсатиши мумкин.

Саволлар:

1. Дастур матни нима?
2. Тезкор тугмалар тавсифи?

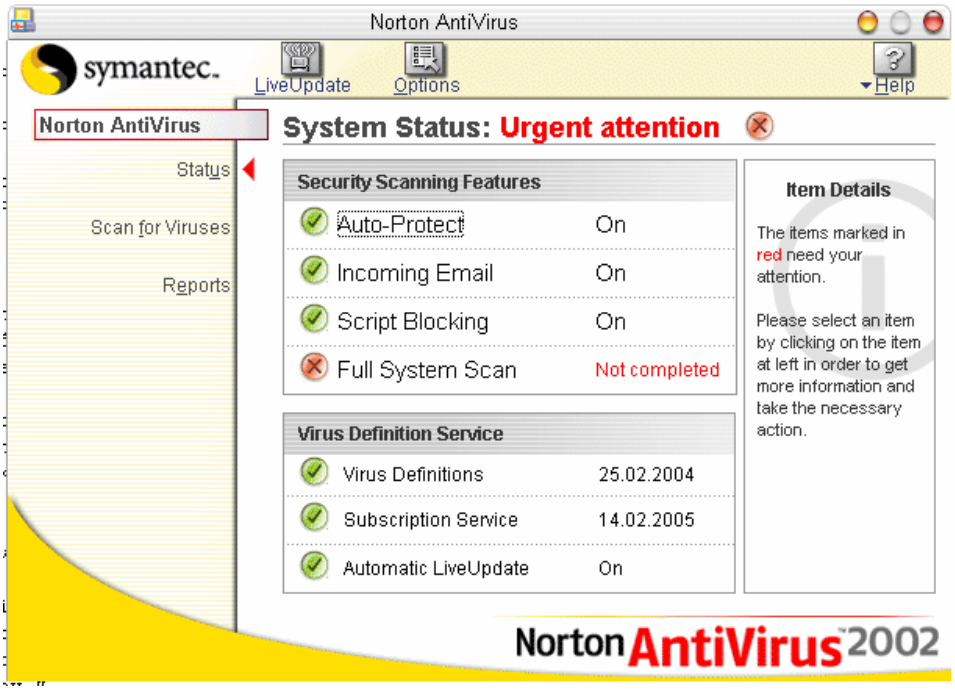
Топшириқлар:

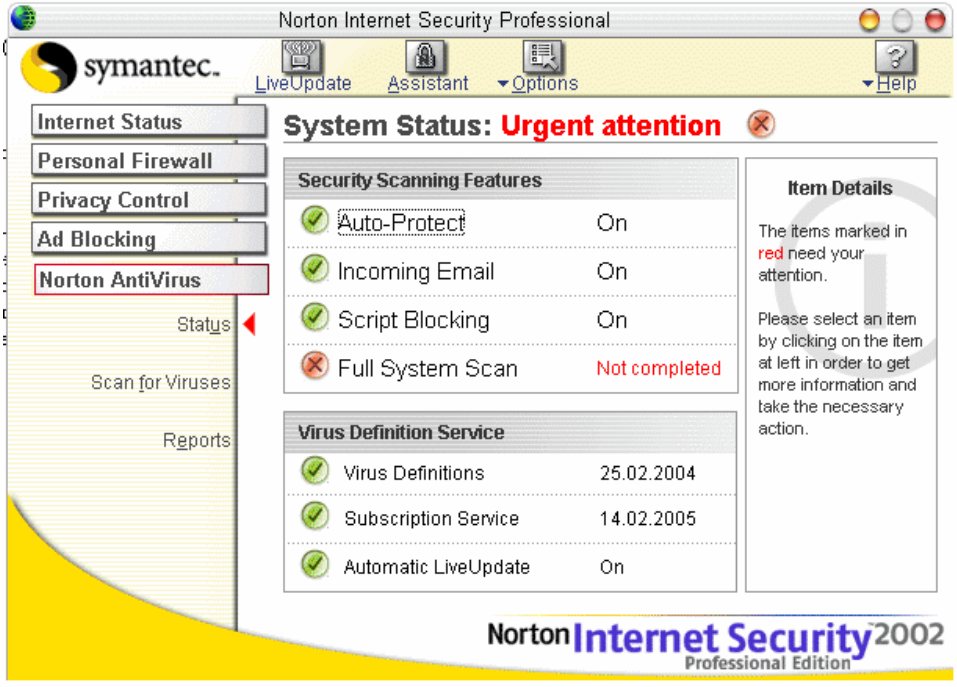
2. Delphi да янги дастур яратинг.
3. Дастурни синтактик ва бошқа хатоликларини текширинг.

ТАЖРИБА ИШИ №3

Форма ва объектларнинг хусусиятлари билан ишлаш.

Баъзида формалар интерфейсини турли кўринишга олиб келишга ҳаракат қилиниши мумкин. Масалан қуйидаги ойнада дастур формасида антивирус дастурининг ҳолати чиқарилган.





Delphiнинг ёрдам файлларида компонентни кўчириш учун унга Parent (асосий) қийматини бериш кифоя деб ёзилган. Агар лоиҳада иккита форма бўлса, панелни қуйидагича олиб ташлаш мумкин:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Form2.Panel2.Parent := Form1;
end;
```

ва панел керагидай биринчи формага ўтади, қайтариш учун эса:

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  Form2.Panel2.Parent := Form2;
end;
```

кодини ёзиш кифоя.

Форма кўринишини DLLдан олиш.

Аниқланишича оддий .Parent қиймати билан иш битмас экан. Асосий дастурда процедураларни аниқлаймиз:

```
const
  DLLName = 'paneldll.dll';

function InitLib(App: Integer): Boolean; external DLLName;
function CreateFM(): Boolean; external DLLName;
function ShowPN(pn: HWND): Pointer; external DLLName;
function HidePN(): Boolean; external DLLName;
function ReleaseFM(): Boolean; external DLLName;
function UnloadLib(): Boolean; external DLLName;
```

Фаоллаштириш DLL файли дастурнинг бир қисми сифатида ишлаши учун керак. Сўнгра формани яратамиз. Энди панелни DLL дан формага кўчириш учун қуйидаги кодни ёзамиз:

```
procedure TfmHostApp.btShowClick(Sender: TObject);
```

```
begin
  TPanel(ShowPN(fmHostApp.Handle)).Parent := fmHostApp;
end;
```

DLLнинг ўзида эса:

```
function ShowPN(pn: HWND): Pointer; register;
exports ShowPN;
begin
  //ShowMessage(fmDLL.pnDLL.Parent.Name);
  fmDLL.pnDLL.Parent := nil;
  fmDLL.pnDLL.ParentWindow := pn;
  //ShowMessage(fmDLL.pnDLL.Parent.Name);
  Result := fmDLL.pnDLL;
end;
```

Юқоридаги даастуни бироз соддалаштирамиз.

```
 fmDLL.pnDLL.Parent:=nil;
 fmDLL.pnDLL.ParentWindow:=pn;
```

Энди панелни қайтариш учун қуйидаги процедарани чақирамиз:

```
procedure TfmHostApp.btHideClick(Sender: TObject);
begin
  HidePN();
end;
```

DLLда эса олдинги код:

```
function HidePN(): Boolean;
begin
  fmDLL.pnDLL.Parent := nil;
  fmDLL.pnDLL.ParentWindow := fmDLL.Handle;
  fmDLL.pnDLL.Parent := fmDLL;
  Result := True;
end;
```

Биз хохлаган мақсадга эришишимиз учун шу кодлар етарли.

Формани тушириш ва қайта тиклашда анимациядан фойдаланиш

```
{
In Win9X or NT4, there's a 'zooming effect' when an application is minimized
to the taskbar or restored from the taskbar.
Delphi applications don't have this zooming effect.
You can switch the effect on or off with the following piece of code:
}
 Info: TAnimationInfo;
 begin
   ZeroMemory(@Info,SizeOf(Info));
   Info.cbSize := SizeOf(TAnimationInfo);
   BOOL(Info.iMinAnimate) := Value;
   SystemParametersInfo(SPI_SETANIMATION, SizeOf(Info), @Info, 0);
 end;

 procedure TForm1.Button1Click(Sender: TObject);
 begin
   SetAnimation(True);
 end;
```

Саволлар:

3. Формалар дизайнери хусусиятлари?
4. Формани анимациялаш?
5. Форма хусусиятлари?

Топшириқлар:

4. Анимацияли ишга тушувчи форма яратинг.
5. Формага фон сифатида турли расмларни ўрнатинг.

ТАЖРИБА ИШИ №4

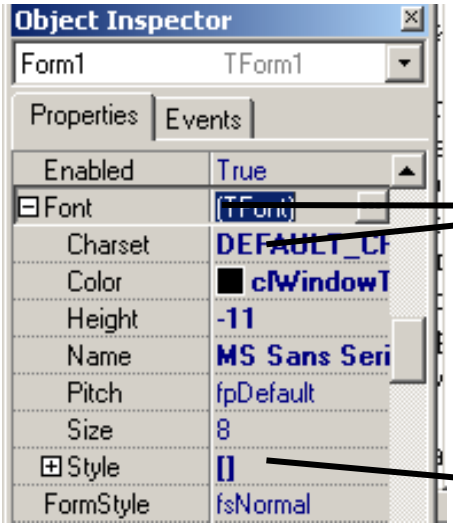
Ҳодисалар ва услублардан фойдаланиш.

Хусусиятлар объектнинг муҳим атрибутаси эканлигига диққатингизни қаратгандирсиз. Фойдаланувчи (дастурчи) учун хусусият бу бирор тузилишнинг қиймат сақловчи майдонидир. Лекин, бу оддий майдон қиймати ўзгариши билан компонентнинг ташқи кўриниши бутунлай ўзгаради, чунки, хусусиятлар ўзида объектнинг шу майдонга ёзиш ва ўқиш ҳолатлари инкапсуляциясига боғлиқ бўлади. Хусусиятлар икки мақсадда хизмат қилади. Биринчидан, улар форма ёки компонентнинг ташқи кўринишини аниқлайди. Иккинчидан эса хусусиятлар форма ёки компонентнинг ҳаракатини белгилайди.

Хусусиятлар бир нечта типларга бўлиш мумкин.

- Оддий хусусиятлар – буларнниг қийматлари сонлар ёки сатлар бўлади. Масалан, Left ва Top хусусиятлари форма ёки компонентнинг юқориги чапки бурчаги координаталарини аниқловчи бутун сонларни сақлайди. Caption ва Name хусусиятлари форма ёки компонентнинг сарлавҳаси ва номини билдирувчи сатрларни сақлайди.

Delphiда компонентлар хусусиятларини лоиҳалаш вақтида (design time) ҳам, дастур бажарилаш вақтида (run time) ҳам ўзгартириш оддий.



19-расм ички маълумотли хусусиятларнинг Объектлар Инспекторида кўриниши
а) Бирлашган қийматлар.
б) Кўпликлар

Лоиҳалаш вақтида хусусиятларни ўзгартириш Формалар Дизайнери (Forms Designer) ёки, сизга маълум Объектлар Инспекторининг “Properties” саҳифаси ёрдамида бажарилади. Масалан, тугманинг Height (баландлиги) ва Width (кенглиги) хусуиятларини ўзгартириш учун, унинг формадаги тасвирининг бир бурчагидан сичқонча ёрдамида тортиб, керакли ўлчамларга эришиш мумкин. Худди шу натижага Object Inspectorнинг Height ва Width хусусиятларни қийматларини ўзгартириб ҳам эришиш мумкин.

```
unit OurComponent;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

type
  TOurComponent = class(TComponent)
  private
    { Private declarations }
    FMyInteger: Integer;
    FMyChar: Char;
    FMyString: string;
    procedure SetMyInteger(const Value: Integer);
    procedure SetMyChar(const Value: Char);
    procedure SetMyString(const Value: string);
  protected
    { Protected declarations }
  public
    { Public declarations }
```

```
  published
    { Published declarations }
    property MyInteger: Integer read FMyInteger write SetMyInteger;
    property MyChar: Char read FMyChar write SetMyChar;
    property MyString: string read FMyString write SetMyString;
end;

procedure register;

implementation

procedure register;
begin
  RegisterComponents('Samples', [TOurComponent]);
end;

{ TOurComponent }

procedure TOurComponent.SetMyChar(const Value: Char);
begin
  FMyChar := Value;
end;

procedure TOurComponent.SetMyInteger(const Value: Integer);
begin
  FMyInteger := Value;
end;

procedure TOurComponent.SetMyString(const Value: string);
```

```
begin
  FMyString := Value;
end;

end.
```

Санаб ўтиладиган хусусиятлар – булар мавжуд рўйхатдаги қийматлардан бирини қабул қила оладиган хусусиятларир. Оддий мисол – **Boolean** типидаги хусусият, *True* ёки *False* қийматини қабул қила олади

```
unit OurComponent;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics,
  Controls, Forms, Dialogs;

type
  TMyEnumerate = (meFirst, meSecond, meThird);

  TOurComponent = class(TComponent)
  private
    { Private declarations }
    FMyBoolean: Boolean;
    FMyCursor: TCursor;
    FMyEnumerate: TMyEnumerate;
    procedure SetMyCursor(const Value: TCursor);
    procedure SetMyBoolean(const Value: Boolean);
```

```
    procedure SetMyEnumerate(const Value: TMyEnumerate);
  protected
    { Protected declarations }
  public
    { Public declarations }
  published
    { Published declarations }
    property MyCursor: TCursor read FMyCursor write SetMyCursor;
    property MyBoolean: Boolean read FMyBoolean write SetMyBoolean;
    property MyEnumerate: TMyEnumerate read FMyEnumerate write
    SetMyEnumerate;
end;

procedure register;

implementation

procedure register;
begin
  RegisterComponents('Samples', [TOurComponent]);
end;

{ TOurComponent }

procedure TOurComponent.SetMyCursor(const Value: TCursor);
begin
  FMyCursor := Value;
end;
```

```
procedure TOurComponent.SetMyEnumerate(const Value: TMyEnumerate);
begin
  FMyEnumerate := Value;
end;

procedure TOurComponent.SetMyBoolean(const Value: Boolean);
begin
  FMyBoolean := Value;
end;

end.
```

Ички маълумотли хусусиятлар – булар ички қийматларга (ёки объектларга) эга булган хусусиятлардир. Бундай хусусиятлар Object Inspectorда чап томонидан “+” белгиси билан ажралиб туради. Бундай хусусиятлар ҳам икки хилда бўлади: *кўпликлар* ва *бирлашган қийматлар*. Object Inspectorда кўпликлар квадрат қавслар ёрдамида ифодаланади. Агар кўплик бўш бўлса, у [] сифатида тасвирланади. Кўплик кўринишидаги ички маълумотли хусусиятлар кўпинча мантиқий **Boolean** типида бўлади. Бирлашган қийматлар Объектлар Инспекторида баъзи катталиклар тўплами сифатида тасвирланади. Баъзи хусусиятлар масалан, Font ўз қийматини ўзгартириш учун мулоқот ойналарини чақира олади. Бунинг учун шу хусусият майдонининг ўнг томонидаги уч нуқтали кичик тугмачани босиш кифоя.

```
unit OurComponent;

interface

uses
```

```
  Windows, Messages, SysUtils, Classes, Graphics,
  Controls, Forms, Dialogs;

type
  TMySet = (msOne, msTwo, msThee, msFour, msFive);
  TMySets = set of TMySet;

  TOurComponent = class(TComponent)
  private
    { Private declarations }
    FMySet: TMySets;
    procedure SetMySet(const Value: TMySets);
  protected
    { Protected declarations }
  public
    { Public declarations }
  published
    { Published declarations }
    property MySet: TMySets read FMySet write SetMySet;
end;

procedure register;

implementation

procedure register;
begin
  RegisterComponents('Samples', [TOurComponent]);
end;
```

```
{ TOurComponent }

procedure TOurComponent.SetMySet(const Value: TMySets);
begin
  FMySet := Value;
end;

end.
```

Объектли хусусиятлар

```
unit OurComponent;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs;

type

  TOurComponent = class(TComponent)
  private
    { Private declarations }
    FMyFont: TFont;
    procedure SetMyFont(const Value: TFont);
  protected
    { Protected declarations }
  public
```

```
    { Public declarations }
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
  published
    { Published declarations }
    property MyFont: TFont read FMyFont write SetMyFont;
end;

procedure register;

implementation

procedure register;
begin
  RegisterComponents('Samples', [TOurComponent]);
end;

{ TOurComponent }

constructor TOurComponent.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FMyFont.Create;
end;

destructor TOurComponent.Destroy;
begin
  FMyFont.Free;
```

```
  inherited Destroy;
end;

procedure TOurComponent.SetMyFont(const Value: TFont);
begin

  FMyFont.Assign(Value);
end;

end.
```

Массив типидаги хусусиятлар

```
type
  TOurComponent = class(TComponent)
  private
    { Private declarations }
    FArrayProp: array[0..9] of integer;
    function GetArrayProp(aIndex: integer): integer;
    procedure SetArrayProp(aIndex: integer; const Value: integer);
  protected
    { Protected declarations }
  public
    { Public declarations }
    property ArrayProp[aIndex: integer]: integer read GetArrayProp
    write SetArrayProp;
  published
    { Published declarations }
end;
```

Хусусиятлар тахрирлагичи регистрацияси

Бунинг учун яратилаётган компонентнинг Register хусусиятига қуйидаги кодни киритиш керак:

```
RegisterPropertyEditor( TypeInfo( TContainedClass ),
TContainer, 'Contained', TClassProperty ) ;
```

Саволлар:

1. Хусусият тушунчасини изоҳланг.
2. Қандай хусусиятлар мавжуд?
3. Объектлар хусусиятларини қайси усулларда ўзгартириш мумкин?

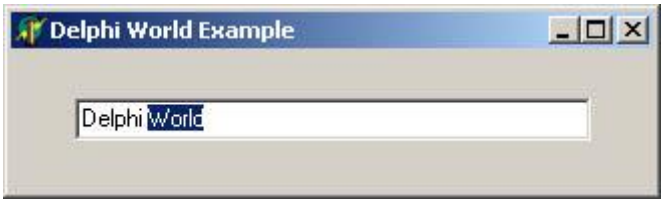
Топшириқлар:

1. Хусусиятларни дастур бажарилаш вақтида ўзгартирувчи дастур тузинг.
2. Мос хусусиятлар ва кўпликларни қўлловчи дастур тузинг.

ТАЖРИБА ИШИ №5

Турли типдаги ўзгарувчилардан фойдаланиш, типлар орасида ўтиш масалаларини ҳал қилиш.

Автоматик танловли Edit



Бу ерда кўриниши оддий матн киритиш майдончаси эга бўлган компонент келтирилган. Бу компонент ThintEdit деб аталади, у TcustomEdit дан яратилган. Бунинг учун олдиндан TStrings хусусиятига эга бўлган HintList хусусиятига киритиб қўилади.

```
unit HintEdit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  THintEdit = class(TCustomEdit)
  private
    { Private declarations }
    FHintList: TStrings;
    Searching,
      CanSearch: boolean;
    CurSPos: integer;
  protected
```

```
  { Protected declarations }
  procedure Change; override;
  procedure KeyDown(var Key: Word; Shift: TShiftState); override;
public
  { Public declarations }
  constructor Create(AOwner: TComponent); override;
  property HintList: TStrings read FHintList write FHintList;
  destructor Destroy; override;
published
  { Published declarations }
  property Anchors;
  property AutoSelect;
  property AutoSize;
  property BiDiMode;
  property BorderStyle;
  property CharCase;
  property Color;
  property Constraints;
  property Ctl3D;
  property DragCursor;
  property DragKind;
  property DragMode;
  property Enabled;
  property Font;
  property HideSelection;
  property ImeMode;
  property ImeName;
  property MaxLength;
  property OEMConvert;
```

```
property ParentBiDiMode;
property ParentColor;
property ParentCtl3D;
property ParentFont;
property ParentShowHint;
property PasswordChar;
property PopupMenu;
property ReadOnly;
property ShowHint;
property TabOrder;
property TabStop;
property Text;
property Visible;
property OnChange;
property OnClick;
property OnDblClick;
property OnDragDrop;
property OnDragOver;
property OnEndDock;
property OnEndDrag;
property OnEnter;
property OnExit;
property OnKeyDown;
property OnKeyPress;
property OnKeyUp;
property OnMouseDown;
property OnMouseMove;
property OnMouseUp;
property OnStartDock;
```

```
    property OnStartDrag;
  end;

procedure Register;

implementation

{$R *.DCR}

procedure Register;
begin
  RegisterComponents('Netscape', [THintEdit]);
end;

constructor THintEdit.Create;
begin
  inherited;
  FHintList := TStringList.Create;
  Searching := false;
  CanSearch := true;
  CurSPos := -1;
end;

procedure THintEdit.Change;
var
  i, l: integer;
begin
  if Searching then
    Exit;
```

```
  if not CanSearch then
    Exit;
  if Text = '' then
    exit;
  l := Length(Text);
  for i := 0 to FHintList.Count - 1 do
    if Copy(FHintList[i], 1, l) = Text then
    begin
      Searching := true;
      CurSPos := i;
      Text := FHintList[i];
      Searching := false;
      SelStart := Length(Text);
      SelLength := -(Length(Text) - l);
      break;
    end;
  inherited;
end;

procedure THintEdit.KeyDown;
var
  l: integer;
begin
  if Chr(Key) in ['A'..'z', 'A'..'Я', 'a'..'я'] then
    CanSearch := true
  else
    CanSearch := false;
  case Key of
    VK_DOWN:
```

```
  begin
    if (CurSPos < HintList.Count - 1) and (SelLength > 0) then
      if Copy(FHintList[CurSPos + 1], 1, SelStart) = Copy(Text, 1, SelStart)
        then
      begin
        l := SelStart;
        Inc(CurSPos);
        Text := FHintList[CurSPos];
        SelStart := Length(Text);
        SelLength := -(Length(Text) - l);
      end;
    Key := VK_RETURN;
  end;
VK_UP:
  begin
    if (CurSPos > 0) and (SelLength > 0) then
      if Copy(FHintList[CurSPos - 1], 1, SelStart) = Copy(Text, 1, SelStart)
        then
      begin
        l := SelStart;
        Dec(CurSPos);
        Text := FHintList[CurSPos];
        SelStart := Length(Text);
        SelLength := -(Length(Text) - l);
      end;
    Key := VK_RETURN;
  end;
VK_RETURN:
  begin
```

```
      SelStart := 0;
      SelLength := Length(Text);
    end;
  end;
  inherited;
end;

destructor THintEdit.Destroy;
begin
  FHintList.Free;
  inherited;
end;
```

Саволлар:

1. Қандай типлар мавжуд.
2. Типлар орасида ўтиш операторларини санаб ўтинг.

Топшириқлар:

1. Турли типларга мос келувчи Variables дастурини тузинг.
2. Variant типидаги ўзгарувчилардан фойдаланиб типларни алмаштирувчи дастур тузинг.

ТАЖРИБА ИШИ №6

Цикл ва шарт операторлари ёрдамида дастурлар тузиш.

Циклли мурожаатлар муоммоси.

```
procedure TForm1.FormCreate(Sender: TObject);
var
  I: integer;
begin
  for I:= 0 to ComponentCount -1 do
    if (Components[I] IS TEdit) then
      (Components[I] AS TEdit).
end;
```

Агар сизга edit-компонентлар тўплами керак бўлса, уларни формага жойлаштирамиз ва қуйидаги кодни ёзамиз:

```
procedure TForm1.FormCreate(Sender: TObject);
var
  I: integer;
begin
  with MyPanel do
    for I:= 0 to ControlCount -1 do
      if (Controls[I] IS TEdit) then
        (Controls[I] AS TEdit).
end;
```

Мисол Edit1, Edit2 худди Edit[1], Edit[2] кабу мурожжат учун тайр бўлади. Агар сиз компонентларга массив элементлари каби мурожаат қилмоқчи бўлсангиз, уларни TList га жойлаштиринг.

```
MyArr := TList.Create;
MyArr.Add(Edit1);
MyArr.Add(Edit2);

...

For i := 0 To MyArr.count - 1 Do
  (MyArr.items[i] As TEdit).Enabled := False;

MyArr.Free;



procedure TForm1.FormCreate(Sender: TObject);
var
  I: Integer;
begin
  for I := 0 to ComponentCount -1 do
    if Components[I] is TEdit then
      TEdit(Components[I]).Whatever := 10;
end;
```

Мурожаат олиш учун:

```
TButton(mylist.items[I]).property := sumpin;
```

ёки

```
TButton(mylist.items[I]).method;
```

```
Procedure TMyForm.MyButtonHandler(Sender: TObject);
Begin
  Case (Sender As TComponent).Tag Of
  End;
```

```
End;
```

Узоқ цикл бажарилганда бошқа дастурлаш осилиб қолмаслиги (зависание) учун қуйидаги кодни ёзамиз

Циклга эса:

```
Application.ProcessMessages;
```

Бу кодни киритганимиздан сўнг бошқа дастурлар умуман осилиб қолмайди.:

```
procedure TForm1.Button2Click(Sender: TObject);
var
  i: integer;
begin
  randomize;
  for i:=0 to 50000000 do
  begin
    Form1.Caption := IntToStr(Random(5000));
    Application.ProcessMessages;
  end;
end;
```

Саволлар:

1. Қандай цикл операторлари мавжуд.
2. Тугамайдиган цикл нима ва уни олдини олиш.

Топшириқлар:

1. Циклни қўлловчи процедураларни чақириш дастурини тузинг.
2. Циклни қўлловчи функцияларни чақириш дастурини тузинг.

ТАЖРИБА ИШИ №7

Массивлар билан ишлаш дастурларини тузиш.

Delphi да массивлар билан ишлаш.

Қуйида икки ўлчовли массивлар билан ишлаш учун бир нечта функциялар келтирилган. SetV ва GetV процедуралари массив элементларини ўқиш ва ёзиш амалларини бажаради. Масалан:

```
type

  VArray: array[1..1] of double;
var

  X: ^VArray;
  NR, NC: Longint;

begin

  NR := 10000;
  NC := 100;
  if AllocArray(pointer(X), N * Sizeof(VArray)) then
    exit;
  SetV(X^, NC, 2000, 5, 3.27); { X[2000,5] := 3.27 }
end;

function AllocArray(var V: pointer; const N: longint): Boolean;
begin

  try
    GetMem(V, N);
  except
    ShowMessage('ОШИБКА выделения памяти. Размер:' + IntToStr(N));
```

```
    Result := True;
    exit;
  end;
  FillChar(V^, N, 0);
  Result := False;
end;

procedure SetV(var X: Varray; const N, ir, ic: LongInt; const value:
  double);
begin

  X[N * (ir - 1) + ic] := value;
end;

function GetV(const X: Varray; const N, ir, ic: Longint): double;
begin

  Result := X[N * (ir - 1) + ic];
end;
```

Энг оддий усул бу – динамик массив яратиш

```
Myarray := GetMem(rows * cols * sizeof(byte,word,single,double и пр.)
```

fetch_num функциясини яратамиз

```
function fetch_num(r,c:integer) : single;
```

```
result := pointer + row + col*rows
```

ва myarray[2,3] ўрнига қуйидагича ёзамиз:

```
myarray.fetch_num(2,3);
```

Бу процедура ва функциялар массивлар билан ишлашни енгиллаштиради.

Қуйида бир ва икки ўлчовли массивларни динамик яратишни кўриб ўтамиз:

```
(*
```

```
--    TDynaArray  :  бир ўлчовли массив
--    TDynaMatrix  :  икки ўлчовли динамик массив
*)
unit DynArray;
interface
uses
  SysUtils;
type
  TDynArrayBaseType = double;
const
  vMaxElements = (High(Cardinal) - $F) div sizeof(TDynArrayBaseType);
type
  TDynArrayNDX = 1..vMaxElements;
  TArrayElements = array[TDynArrayNDX] of TDynArrayBaseType;
  {=  биз  эълон  қилишимиз  мумкин  бўлган  энг  катта  массив
TDynArrayBaseType =}
  PArrayElements = ^TArrayElements;
  {= массивга кўрсаткич =}
  EDynArrayRangeError = class(ERangeError);
  TDynArray = class
  private
    fDimension: TDynArrayNDX;
    fMemAllocated: word;
    function GetElement(N: TDynArrayNDX): TDynArrayBaseType;
    procedure  SetElement(N:  TDynArrayNDX;  const  NewValue:
TDynArrayBaseType);
  protected
    Elements: PArrayElements;
  public
```

```
  constructor Create(NumElements: TDynArrayNDX);
  destructor Destroy; override;
  procedure Resize(NewDimension: TDynArrayNDX); virtual;
  property dimension: TDynArrayNDX
    read fDimension;
  property Element[N: TDynArrayNDX]: TDynArrayBaseType
  read GetElement
    write SetElement;
  default;
 end;

const
 vMaxMatrixColumns = 65520 div sizeof(TDynArray);
 {= TDynArray объектлар массиви ёрдамида синф матрицасини қуриш =}
type
 TMatrixNDX = 1..vMaxMatrixColumns;
 TMatrixElements = array[TMatrixNDX] of TDynArray;
 PMatrixElements = ^TMatrixElements;
 TDynaMatrix = class
 private
  fRows: TDynArrayNDX;
  fColumns: TMatrixNDX;
  fMemAllocated: longint;
  function GetElement(row: TDynArrayNDX;
    column: TMatrixNDX): TDynArrayBaseType;
  procedure SetElement(row: TDynArrayNDX;
    column: TMatrixNDX;
    const NewValue: TDynArrayBaseType);
 protected
```

```
    mtxElements: PMatrixElements;
  public
    constructor Create(NumRows: TDynArrayNDX; NumColumns:
TMatrixNDX);
    destructor Destroy; override;
    property rows: TDynArrayNDX
      read fRows;
    property columns: TMatrixNDX
      read fColumns;
    property Element[row: TDynArrayNDX; column: TMatrixNDX]:
TDynArrayBaseType
    read GetElement
      write SetElement;
    default;
  end;

implementation
(*
--  TDynArray услублари
*)
constructor TDynArray.Create(NumElements: TDynArrayNDX);
begin {==TDynArray.Create==}
  inherited Create;
  fDimension := NumElements;
  GetMem(Elements, fDimension * sizeof(TDynArrayBaseType));
  fMemAllocated := fDimension * sizeof(TDynArrayBaseType);
  FillChar(Elements^, fMemAllocated, 0);
end; {==TDynArray.Create==}
```

```
destructor TDynArray.Destroy;
begin {==TDynArray.Destroy==}
 FreeMem(Elements, fMemAllocated);
 inherited Destroy;
end; {==TDynArray.Destroy==}
procedure TDynArray.Resize(NewDimension: TDynArrayNDX);
begin {TDynArray.Resize==}
 if (NewDimension < 1) then
  raise EDynArrayRangeError.CreateFMT('Индекс вышел за границы
диапазона : %d',
    [NewDimension]);
 Elements := ReAllocMem(Elements, fMemAllocated, NewDimension *
  sizeof(TDynArrayBaseType));
 fDimension := NewDimension;
 fMemAllocated := fDimension * sizeof(TDynArrayBaseType);
end; {TDynArray.Resize==}
function TDynArray.GetElement(N: TDynArrayNDX): TDynArrayBaseType;
begin {==TDynArray.GetElement==}
 if (N < 1) or (N > fDimension) then
  raise EDynArrayRangeError.CreateFMT('Индекс вышел за границы
диапазона : %d',
    [N]);
 result := Elements^[N];
end; {==TDynArray.GetElement==}
procedure TDynArray.SetElement(N: TDynArrayNDX; const NewValue:
 TDynArrayBaseType);
begin {==TDynArray.SetElement==}
 if (N < 1) or (N > fDimension) then
  raise EDynArrayRangeError.CreateFMT('Индекс вышел за границы
```

```
диапазона : %d',
    [N]);
  Elements^[N] := NewValue;
end; {==TDynArray.SetElement==}
constructor TDynaMatrix.Create(NumRows: TDynArrayNDX; NumColumns:
TMatrixNDX);
var
  col: TMatrixNDX;
begin {==TDynaMatrix.Create==}
  inherited Create;
  fRows := NumRows;
  fColumns := NumColumns;
  GetMem(mtxElements, fColumns * sizeof(TDynArray));
  fMemAllocated := fColumns * sizeof(TDynArray);
  for col := 1 to fColumns do
  begin
    mtxElements^[col] := TDynArray.Create(fRows);
    inc(fMemAllocated, mtxElements^[col].fMemAllocated);
  end;
end; {==TDynaMatrix.Create==}

destructor TDynaMatrix.Destroy;

var
  col: TMatrixNDX;
begin {==TDynaMatrix.Destroy;==}
  for col := fColumns downto 1 do
  begin
    dec(fMemAllocated, mtxElements^[col].fMemAllocated);
```

```
    mtxElements^[col].Free;
  end;
  FreeMem(mtxElements, fMemAllocated);
  inherited Destroy;
end; {==TDynaMatrix.Destroy;==}
function TDynaMatrix.GetElement(row: TDynArrayNDX;
  column: TMatrixNDX): TDynArrayBaseType;
begin {==TDynaMatrix.GetElement==}
  if (row < 1) or (row > fRows) then
    raise
      EDynArrayRangeError.CreateFMT('Индекс строки вышел за границы
диапазона : %d', [row]);
  if (column < 1) or (column > fColumns) then
    raise
      EDynArrayRangeError.CreateFMT('Индекс столбца вышел за границы
диапазона : %d', [column]);
  result := mtxElements^[column].Elements^[row];
end; {==TDynaMatrix.GetElement==}
procedure TDynaMatrix.SetElement(row: TDynArrayNDX;
  column: TMatrixNDX;
  const NewValue: TDynArrayBaseType);
begin {==TDynaMatrix.SetElement==}
  if (row < 1) or (row > fRows) then
    raise
      EDynArrayRangeError.CreateFMT('Индекс строки вышел за границы
диапазона : %d', [row]);
  if (column < 1) or (column > fColumns) then
    raise
      EDynArrayRangeError.CreateFMT('Индекс столбца вышел за границы
```

```
диапазона : %d', [column]);
  mtxElements^[column].Elements^[row] := NewValue;
end; {==TDynaMatrix.SetElement==}
end.
```

DynArray модули учун синов дастури

```
uses DynArray, WinCRT;
const
  NumRows: integer = 7;
  NumCols: integer = 5;
var
  M: TDynaMatrix;
  row, col: integer;
begin
  M := TDynaMatrix.Create(NumRows, NumCols);
  for row := 1 to M.Rows do
    for col := 1 to M.Columns do
      M[row, col] := row + col / 10;
  writeln('Матрица');
  for row := 1 to M.Rows do
  begin
    for col := 1 to M.Columns do
      write(M[row, col]: 5: 1);
    writeln;
  end;
  writeln;
  writeln('Перемещение');
  for col := 1 to M.Columns do
  begin
    for row := 1 to M.Rows do
```

```
    write(M[row, col]: 5: 1);
  writeln;
end;
M.Free;
end.
```

Катта ўлчамли массивларни ҳисоблаш учун модул

Бу модулда массивлар устида амаллар бажарувчи бир нечта функцияларни келтирдик. Divide – тўғридан тўғри бўлиш алгоритми, MSqrt – квадрат илдиз, MAbs – абсолют қиймат. Бу модулимиз оддий мисоллардан тортиб массивларга қўлланилиши учун ишлаб чиқиш қийин бўлмайди:

Масалан

```
var
  N : Integer;
  A : Matrix;
  b, x : Vector;
begin
  N := . . .;
  A.Init( N, N );
  b.Init( N );
  x.Init( N ); // ёки x.Init( B ); ёки x.InitRow( A );
  . . .
  { A ва bустида амаллар }
  . . .
  x.Divide( b, A );
  x.Print;
  . . .
end.
```

Баъзи алгоритмларни изоҳлаш керак:

```
Matrix.E( i, j : LongWord ) или Vector.E( i : Integer ) : RealPtr
```

(RealPtr = ^Real) массив элементи манзилини аниқловчи функция.

Matrix.Invert(A : Matrix)

– агар A[N,M], ва N <> M у ҳолда – [M,N] ўлчамли матрица – = A+га тескари бўлади.

Matrix.Addition(A : Matrix; B : Real)

– асосий диагоналга сон қўшиш.

Matrix.Diag(r : Real)

– асосий диагонал қийматини ўзлаштириш.

Саволлар:

1. Қандай массивлар мавжуд.
2. Кўп ўлчовли массивларни қайта ишлаш.

Топшириқлар:

3. Бир ўлчовли массивларни киртиш ва уларни қайта ишлаш дастурини тузинг.
4. Массивларни саралаш дастурини тузинг.

ТАЖРИБА ИШИ №8

Янги процедура ва функциялар яратиш

Тизим функциялари

function ExecuteWait(const AppPath, CmdLine, DfltDirectory: String; Show: DWORD; TimeOut: DWORD; ProcID: PDWORD): Boolean; Дастурни юклаб унинг бажарилишини кутади. TimeOut – дастур қанча вақт бажарилишини билдиради. Агар шу вақтда дастур ишини тугатмаса, функция true қиймат қайтаради ва ProcID нинг қиймати nilга ўзлаштирилади. Агар TimeOut = INFINITE бўлса, дастур иши якунланиши аниқланмаган бўлади. AppPath, CmdLine, DfltDirectory – дастур номи, юкланувчи параметрлар ва ишчи каталог. ProcID – юкланган дастур учун хендл

function GetStartDir: String; дастургача бўлган йўлни қайтаради

function GetTempDir: string; Вақтинчалик каталог (C:\WINDOWS\TEMP) йўлини қайтаради.

function GetWindowsDir: string; Windows сақланувчи каталогни қайтаради.

function WinVer: TWindowsVersion; Windowsнинг жорий версиясини қайтаради. TWindowsVersion қуйидаги қийматларни қабул қилиши мумкин: wv31, wv95, wv98, wvNT, wvY2K

Типларни алмаштириш функциялари

function Int2Hex(Value: DWord; Digits: Integer): String; Integer типидаги Valueни ўн олтилик саноқ системасидаги сатрга айлантиради, Digits эса рақамларнинг энг кам сонини билиради.

function Int2Str(Value: Integer): String; Integer типидаги Valueни сатрга айлантиради.

function Int2Ths(I: Integer): String; Integer типидаги Valueни сатрга айлантиради Ҳар учта рақамдан кейин ',' белгиси қўйилади (бу мингдан ортиқ сонлар билан ишлашда қулайлик туғдиради).

function Int2Digs(Value, Digits: Integer): String; integer ни stringга айлантириш. Digits – сатрда энг ками нечта рақам борлигини кўрсатади. Агар

Digits Valuedаги рақамлардан катта бўлса, этишмайдиган хоналар ноллар билан тўлдирилади.

function Str2Int(const Value: String): Integer; string дан integer га.

Қуйидаги функциялар юқорида тасвирланганларига мос фақат типлари билан фарқ қилади:

function Str2Double(const S: String): Double;

function Double2Str(D: Double): String;

function Int64_2Str(X: Int64): String;

function Str2Int64(const S: String): Int64;

function Int64_2Double(const X: Int64): Double;

function Double2Int64(D: Double): Int64;

Сатрдан ортиқча рамзларни олиб ташлаш

```
{ **** UBPFD ********** by delphibase.endimus.com ****

Кириш қийматлари:
Input – ортиқча рамзлари олиниши керак бўлган сатр
EArray - сатр, Actionга боғлиқ равишда рухсат этилган ва тақиқланган рамзларни сақлайди
Action – EArray массиви маъносини билдиради.
Action қуйидаги қийматларни олиши мумкин:
1 - EArray массивидаги рамзлар рухсат этилган
2 - EArray - массив рамзлари таъқиқланган.
********************************************************** }
function strtst(var Input: string; EArray: string; Action: integer): string;
begin
  case Action of
    1:
      begin
        while length(Input) <> 0 do
```

```
      begin
        if pos(Input[1], EArray) = 0 then
          delete(Input, 1, 1)
        else
        begin
          result := result + Input[1];
          delete(Input, 1, 1);
        end;
      end;
    end;
   2:
    begin
      while length(Input) <> 0 do
      begin
        if pos(Input[1], EArray) <> 0 then
          delete(Input, 1, 1)
        else
        begin
          result := result + Input[1];
          delete(Input, 1, 1);
        end;
      end;
    end;
  else
    messagebox(0, 'Не корректный вызов функции.', '', mb_ok);
  end;
end;
```

Пример использования:

```
// 1.
s := edit1.text;
```

```
s := strtst(s, '0123456789.', 1);
edit1.text := s;

// масалан киришда: 0.16+
// чиқишдаги кўриниш: 0.16

// 2.
s := edit1.text;
s := strtst(s, '/*-+,', 2);
edit1.text := s;
```

Саволлар:

1. Янги процедуралар яратиш.
2. Процедура ва функциянинг фарқи.

Топшириқ

1. Оддий процедура ва функциялар яратиш.
2. Процедура ва функцияларда реккурсияни қўллаш.

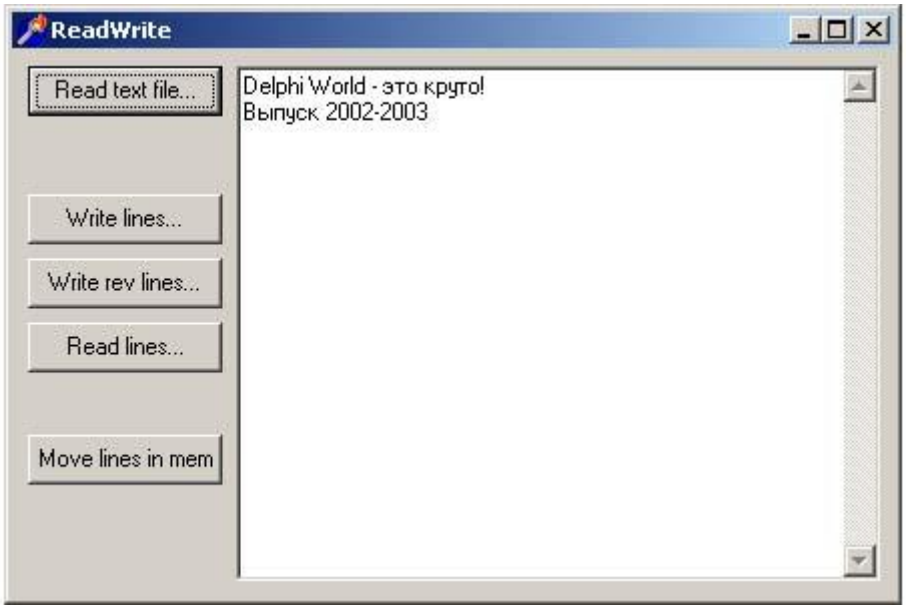
ТАЖРИБА ИШИ № 9

Файллар билан ишлаш дастурлари.

Керакли каталогдаги файллар сони

```
function GetFileCount(Dir: string): integer;
var
  fs: TSearchRec;
begin
  Result := 0;
  if FindFirst(Dir + '\*.htm', faAnyFile - faDirectory - faVolumeID, fs) = 0
    then
    repeat
      inc(Result);
    until
      FindNext(fs) <> 0;
  FindClose(fs);
end;
```

Файлдан ўқиймиз ва унга ёзамиз TWriter ва TReader




```
unit MsForm;
interface
uses
```

```
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm1 = class(TForm)
    ButtonReadText: TButton;
    ButtonWriteLines: TButton;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    Memo1: TMemo;
    ButtonReadLines: TButton;
    ButtonWriteRev: TButton;
    ButtonMoveLines: TButton;
    procedure ButtonReadTextClick(Sender: TObject);
    procedure ButtonWriteLinesClick(Sender: TObject);
    procedure ButtonReadLinesClick(Sender: TObject);
    procedure ButtonWriteRevClick(Sender: TObject);
    procedure ButtonMoveLinesClick(Sender: TObject);
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.ButtonReadTextClick(Sender: TObject);
var
  Str1: TFileStream;
  I: Integer;
begin
  OpenDialog1.Filter :=
    'Text File (*.txt)|*.txt|Pascal File (*.pas)|*.pas|' +
```

```
  'ReadWrite File (*.rwt)|*.rwt|Any file (*.*)|*.*';
 OpenDialog1.DefaultExt := 'txt';
 if OpenDialog1.Execute then
 begin
  Str1 := TFileStream.Create (
   OpenDialog1.Filename, fmOpenRead);
  try
   Memo1.Lines.LoadFromStream (Str1);
  finally
   Str1.Free;
  end;
  // enable all buttons
  for I := 0 to ControlCount - 1 do
   if Controls [I] is TButton then
    TButton (Controls [I]).Enabled := True;
 end;
end;
procedure TForm1.ButtonWriteLinesClick(Sender: TObject);
var
 Str1: TFileStream;
 Writer1: TWriter;
 I: Integer;
begin
 SaveDialog1.Filter :=
  'Read Write file (*.rwf)|*.rwf|Any file (*.*)|*.*';
 SaveDialog1.FilterIndex := 1;
 SaveDialog1.DefaultExt := 'rwt';
 if SaveDialog1.Execute then
 begin
```

```
    Str1 := TFileStream.Create (
      SaveDialog1.FileName, fmCreate or fmOpenWrite);
    Writer1 := TWriter.Create (Str1, 1024);
    try
      Writer1.WriteListBegin;
      for I := 0 to Memo1.Lines.Count - 1 do
        Writer1.WriteString (Memo1.Lines [I]);
      Writer1.WriteListEnd;
    finally
      Writer1.Free;
      Str1.Free;
    end;
  end;
end;
procedure TForm1.ButtonReadLinesClick(Sender: TObject);
var
  Str1: TFileStream;
  Reader1: TReader;
  I: Integer;
begin
  OpenDialog1.Filter :=
    'Read Write file (*.rwf)|*.rwf|Any file (*.*)|*.*';
  OpenDialog1.FilterIndex := 1;
  OpenDialog1.DefaultExt := 'rwt';
  if OpenDialog1.Execute then
  begin
    Str1 := TFileStream.Create (
      OpenDialog1.Filename, fmOpenRead);
    Reader1 := TReader.Create (Str1, 1024);
```

```
    try
      Reader1.ReadListBegin;
      Memo1.Lines.Clear;
      while not Reader1.EndOfList do
      begin
        Memo1.Lines.Add (Reader1.ReadString);
        Application.ProcessMessages;
      end;
      Reader1.ReadListEnd;
    finally
      Reader1.Free;
      Str1.Free;
    end;
    // enable all buttons
    for I := 0 to ControlCount - 1 do
      if Controls [I] is TButton then
        TButton (Controls [I]).Enabled := True;
   end;
end;
procedure TForm1.ButtonWriteRevClick(Sender: TObject);
var
  Str1: TFileStream;
  Writer1: TWriter;
  I: Integer;
begin
  SaveDialog1.Filter :=
    'Read Write Test file (*.rwt)|*.rwt|Any file (*.*)';
  SaveDialog1.DefaultExt := 'rwt';
  if SaveDialog1.Execute then
```

```
  begin
    Str1 := TFileStream.Create (
      SaveDialog1.FileName, fmCreate or fmOpenWrite);
    Writer1 := TWriter.Create (Str1, 1024);
    try
      Writer1.WriteListBegin;
      for I := Memo1.Lines.Count - 1 downto 0 do
        Writer1.WriteString (Memo1.Lines [I]);
      Writer1.WriteListEnd;
    finally
      Writer1.Free;
      Str1.Free;
    end;
  end;
end;

procedure TForm1.ButtonMoveLinesClick(Sender: TObject);
var
  MemStr1: TMemoryStream;
  Writer1: TWriter;
  Reader1: TReader;
  I: Integer;
begin
  MemStr1 := TMemoryStream.Create;
  try
    Writer1 := TWriter.Create (MemStr1, 1024);
    try
      Writer1.WriteListBegin;
      Randomize;
```

```
    while Memo1.Lines.Count > 0 do
    begin
      I := Random (Memo1.Lines.Count);
      Writer1.WriteString (Memo1.Lines [I]);
      Memo1.Lines.Delete (I);
      Application.ProcessMessages;
    end;
    Writer1.WriteListEnd;
  finally
    Writer1.Free;
  end;
  // reset, rewind, move back
  // to the beginning of the stream
  MemStr1.Seek (0, soFromBeginning);
  Reader1 := TReader.Create (MemStr1, 1024);
  try
    Reader1.ReadListBegin;
    while not Reader1.EndOfList do
    begin
      Memo1.Lines.Add (Reader1.ReadString);
      Application.ProcessMessages;
    end;
    Reader1.ReadListEnd;
    finally
      Reader1.Free;
    end;
  finally
    memStr1.Free;
end;
```

```
end;
end.
```

Файлга мурожаатни таъқиқлаш.

...действительно, когда вы запрашиваете о блокировке файла и прерывания DOS, это относится к блокировке записи. Если вы хотите иметь полностью монопольный доступ к файлу, то в этом случае вы должны воспользоваться переменной FileMode. Вот пример кода моей программы, где я использую эту переменную:

```
type
 FileShareType = (DenyCompatibility, DenyAll, DenyWrite, DenyRead, DenyNone);
 FileAccessType = (ReadOnly, WriteOnly, ReadWrite);

procedure SetFileAccess(AccessMode: FileAccessType; ShareMode: FileShareType);
{ Устанавливаем режим доступа к файлу для следующего вызова открытия файла }
begin
 FileMode := ord(AccessMode) or (ord(ShareMode) shl 4)
end;
```

Саволлар:

1. Файллар қандай очилади?
2. Файлни ўқиш ва ёзиш учун очиш.

Топшириқ

1. Процедура ва функцияларда реккурсияни қўллаш.

Адабиётлар

1. Компьютердаги ёрдам файллари.
2. Бобровский «Delphi 5», «Питер» Москва 1997г.
3. Шумаков «Delphi 4 разработка баз данных», «Питер» Москва 1996г.
4. Пачеко, Тейксера «Delphi 5 пособие программиста», «Питер» Москва 1999 г.
5. Фаронов «Delphi 4 учебное пособие», «Питер» Москва 1995 г.