

Сибирский федеральный университет  
Институт математики и фундаментальной информатики  
Базовая кафедра вычислительных и информационных технологий

Методические материалы для подготовки к экзамену  
по предмету

«Языки и технологии программирования»

3 семестр

Лектор: Олейников Б. В.

Составители: Олейников Б. В., Чередниченко О. М.

Красноярск, 2013 г.

## Оглавление

Темы, выносимые на экзамен .....	3
Теоретические вопросы .....	3
Типовая задача.....	5
Задача «Бортовой журнал».....	5
Решение .....	6
Билет .....	24
Литература.....	24

## Темы, выносимые на экзамен

1. Основы ООП. Общее понятие класса и объекта, абстракции. Инкапсуляция: поля, методы, свойства, область видимости. Конструктор и деструктор.
2. Наследуемые объекты: поля, методы, свойства. Класс TObject — родоначальник иерархии наследования.
3. Полиморфизм. Переопределяемые и перегруженные методы. События. Обработчики событий.
4. Использование готовых библиотек на примере класса TList модуля Classes в Delphi.
5. Обобщенное программирование: стандартные реализации обобщенных списков в модуле Generics.Collections. Анонимные методы.
6. Компонентная модель программирования, использование компонентов библиотеки VCL Delphi для визуального программирования.

## Теоретические вопросы

1. Объектно-ориентированная методология: основные составляющие. История ООП.
2. Основная концепция и принципы ООП. Абстракция. Класс. Объект.
3. Инкапсуляция. Поля, методы, свойства. Конструктор. Деструктор. Основной принцип работы с объектом. Взаимодействия с объектом. Параметр Self.
4. Свойства. Виды свойств. Использование свойств. Двойное назначение default. Классификация свойств.
5. Модуль. Области видимости объекта и его составляющих.
6. События. Программирование, управляемое событиями. Обработчики событий. Нотация. Понятие делегирования события.
7. Наследование. Перекрывание методов. Виды перекрываемых методов. Статические методы. Раннее связывание. Использование Inherited.
8. Наследование и видимость. Переопределение свойств при наследовании.
9. Полиморфизм. Основной механизм реализации полиморфизма. Виртуальные и динамические методы. Позднее связывание. Методы обработки сообщений.
10. Перегружаемые методы. Необходимость введения. Использование.
11. Абстрактные объекты и методы. Необходимость их введения.
12. Объект изнутри. Таблицы VMT и DMT.
13. Методы класса и указатели на класс. Операторы is и as.
14. Понятие компонента. Назначение, структура и состав библиотеки VCL Delphi.
15. Основной класс TObject. Назначение. Возможности. Важнейшие методы класса.
16. Интерфейсы в Delphi. Необходимость. Объявление. Использование.
17. Структуры-совокупности объектов. Контейнеры. Коллекции. Шаблоны (паттерны).
18. Итераторы при работе с совокупностями объектов. Назначение. Отличие от индексации. Виды итераторов.
19. Универсальный стандартный контейнер TList библиотеки VCL. Его свойства и методы.
20. Создание контейнера на основе TList. Сортировка и вывод его элементов.
21. Внешние библиотеки, включающие работу с совокупностями объектов. Обзор. Подключение внешних библиотек к Delphi (на примере DeCAL).

22. Понятие обобщенного программирования. Дженерики. Их необходимость и преимущества использования.
23. Типы и методы, подлежащие обобщению в Delphi.
24. Создание обобщенных совокупностей объектов в Delphi.
25. Анонимные функции (методы). Их соотношение с процедурным типом.
26. Понятие визуального программирования. Визуальное программирование в Delphi. Основные этапы.
27. Форма. Инспектор объектов, редактор кода, палитра компонентов – базовые конструкции для визуального программирования. Их практическое использование.
28. Графика. Растровые и векторные изображения. Графика в Delphi. Базовые классы. Свойство Canvas. Canvas и Windows GDI.
29. Основные функции Canvas. Основные инструменты и мастера. Внешние графические библиотеки.
30. Графика в Delphi. Рисование на форме. Основные этапы.
31. Необходимость наглядного представления ООМ. Язык UML. Назначение. Предыстория и история языка UML.
32. Общее представление языка UML. Структура языка UML. Пиктограммы для изображения элементов структуры.
33. Основные диаграммы языка UML. Их назначение и использование. Примеры.
34. ПО поддержки языка UML. Обзор. Использование.

## Типовая задача

### Задача «Бортовой журнал»

На своем пути космический корабль регистрирует в бортовом журнале (см. папку cosmos) данные о встречающихся космических объектах: искусственных и естественных. Об искусственном объекте записывается название, диаметр (в см), год и страна производства, фотография. О естественном — название, диаметр (в см), предположительный возраст и фотография.

В среде Delphi с использованием ООП написать программу, предоставляющую удобный интерфейс работы с данной информацией и позволяющую:

1. Организовать данные для дальнейшей удобной, оперативной работы с ними, в частности, просматривать совокупность зарегистрированных объектов с выводом на экран их общих характеристик в соответствии с файлом журнал.txt. (5 баллов)
2. Упорядочивать эту совокупность по названию и диаметру объекта. (5 баллов)
3. Искать объекты в задаваемом интервале диаметров объектов из числа зарегистрированных объектов. (5 баллов)
4. Просматривать фотографию и остальные характеристики выбираемого объекта. (10 баллов)

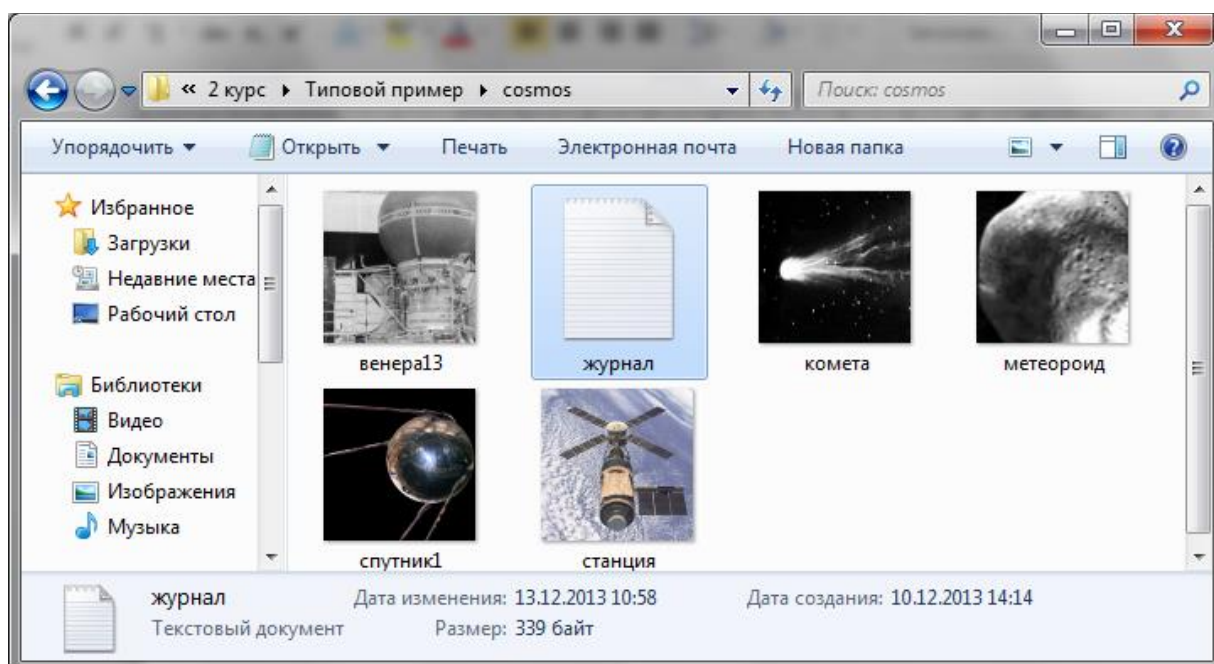
Дополнительные баллы:

Использование TList (10 баллов) или использование обобщенных классов из стандартной коллекции (15 баллов).

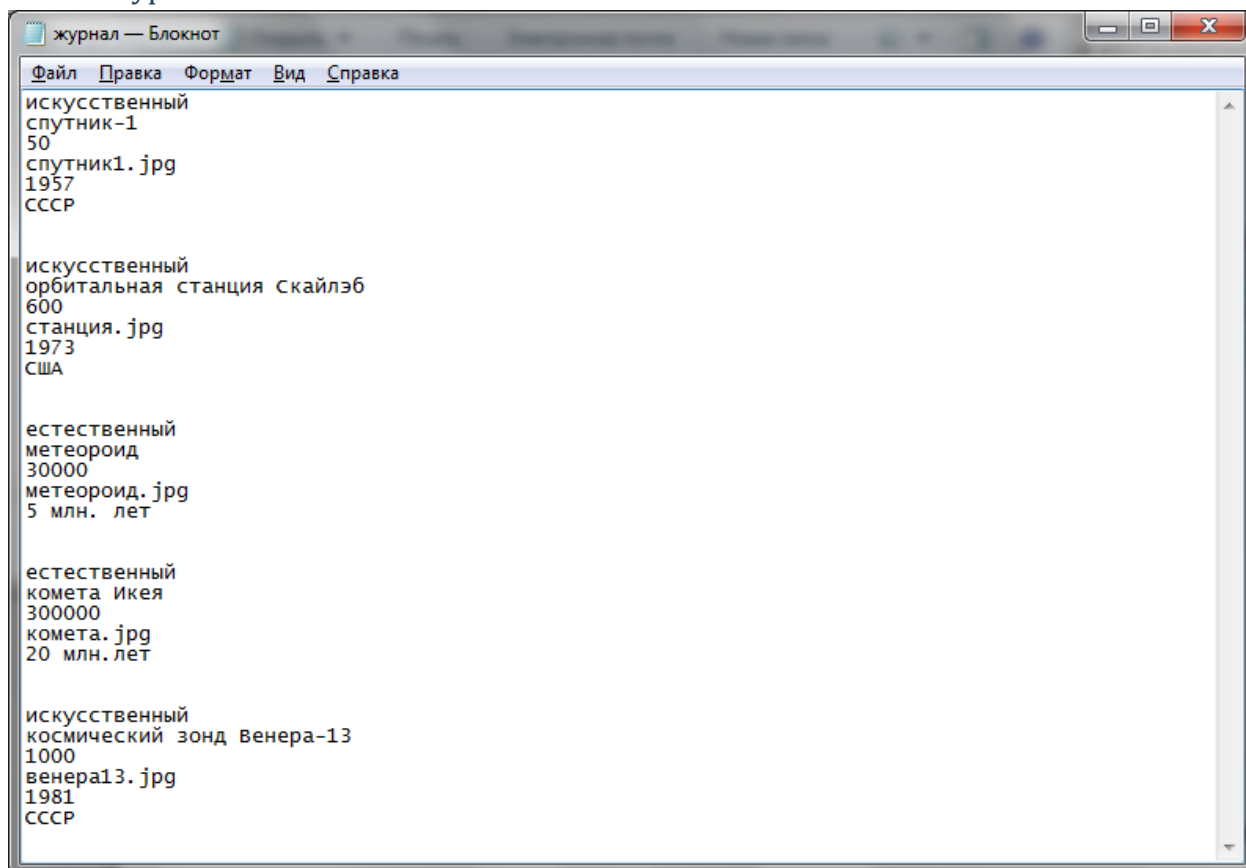
Графический интерфейс. (10 баллов)

### Содержимое папки cosmos

В папке содержатся текстовый файл «журнал.txt» и файлы с изображениями объектов в формате JPEG.



## Файл «журнал.txt»



## Решение

### Пункт 1

В соответствии с требованиями ООП выявим основные абстракции предметной области. В нашем случае — это бортовой журнал и космический объект, который может быть либо искусственным, либо естественным.

Космический объект в программе представим соответствующим абстрактным классом `TSpaceObject`, в который соберем общие характеристики космических объектов обоих типов: название (свойство `Name`), диаметр (свойство `Diameter`), фотография (свойство `Image`).

Объявим класс абстрактным потому, что не планируем создавать в программе объекты непосредственно этого класса, он будет родительским классом для двух конкретных классов космических объектов.

Свойства объявим доступными только на чтение, так как по условию задачи, не требуется изменять их значения после того, как они были загружены из файла. Собственно значения будем хранить в полях с ограничением доступа до уровня `protected` (видимость в пределах класса и в классах-наследниках).

Кроме того, предусмотрим метод загрузки из текстового файла значений соответствующих полей (ReadFromFile).

Поместим определение класса в модуль SpaceObjectUnit.

```
unit SpaceObjectUnit;

interface
type
  TSpaceObject=class abstract (TObject)
  protected
    fName:string;
    fDiameter:integer;
    fImage:string;
  public
    property Name:string read fName;
    property Diameter:integer read fDiameter;
    property Image:string read fImage;
    procedure ReadFromFile(var f:TextFile); virtual;
  end;

implementation

  procedure TSpaceObject.ReadFromFile(var f: TextFile);
  begin
    readln(f,fName);
    readln(f,fDiameter);
    readln(f,fImage);
  end;

end.
```

Далее, объявим классы-наследники, представляющие искусственный и естественный космические объекты: TArtificialObject и TNaturalObject.

Добавим в класс TArtificialObject свойства год (Year) и страна производства (Country) с аналогичными ограничениями, как в TSpaceObject, а в класс TNaturalObject — свойство возраст (Age).

Переопределим в обоих подклассах метод чтения данных из файла, так, чтобы он обращался к родительскому для загрузки полей из класса-предка.

Для целей тестирования добавим в классы метод распечатки информации об объекте в консоль (Print), виртуальный, как и метод чтения из файла.

В виду того, что классы тесно связаны логически и небольшие по объему кода, поместим определения обоих классов в тот же модуль SpaceObjectUnit.

```
unit SpaceObjectUnit;

interface
type
  TSpaceObject=class abstract (TObject)
  protected
    fName:string;
    fDiameter:integer;
```

```

    flmage:string;

    public
    property Name:string read fName;
    property Diameter:integer read fDiameter;
    property Image:string read flmage;
    procedure ReadFromFile(var f:TextFile); virtual;
    procedure Print; virtual;
end;

TArtificialObject=class(TSpaceObject)
protected
    fYear:integer;
    fCountry:string;
public
    property Country:string read fCountry;
    property Year:integer read fYear;
    procedure ReadFromFile(var f:TextFile); override;
    procedure Print; override;
end;

TNaturalObject=class(TSpaceObject)
protected
    fAge:string;
public
    property Age:string read fAge;
    procedure ReadFromFile(var f:TextFile); override;
    procedure Print; override;
end;

implementation

    procedure TSpaceObject.ReadFromFile(var f: TextFile);
    begin
        readln(f,fName);
        readln(f,fDiameter);
        readln(f,flmage);
    end;

    procedure TSpaceObject.Print;
    begin
        writeln('Название: ',Name);
        writeln('Диаметр: ', Diameter);
        writeln('Изображение: ', Image);
    end;

    procedure TArtificialObject.ReadFromFile(var f: TextFile);
    begin
        inherited ReadFromFile(f);
        readln(f,fYear);
        readln(f,fCountry);
    end;

    procedure TArtificialObject.Print;
    begin
        writeln('Искусственный объект');
        inherited Print;
        writeln('Год: ',Year);
        writeln('Страна: ',Country);
    end;

    procedure TNaturalObject.ReadFromFile(var f: TextFile);
    begin

```



```

    inherited readFromFile(f);
    readln(f,fAge);
end;

procedure TNaturalObject.Print;
begin
    writeln('Естественный объект');
    inherited Print;
    writeln('Возраст: ',Age);
end;

end.

```

Перейдем к бортовому журналу. Бортвой журнал представляет собой список космических объектов. Представим его в программе классом TLogBook, содержащим поле — список объектов (List). Для хранения списка выберем обобщенный класс TList<T> из стандартной коллекции обобщенных списков Delphi (модуль Generics.Collections).

В классе TLogBook предусмотрим конструктор, инициализирующий список, и деструктор, уничтожающий его.

Для загрузки фактического списка объектов из текстового файла в журнал, объявим метод LoadFromFile, принимающий в качестве параметра имя файла. В этом методе будем построчно считывать файл. После обнаружения строки «искусственный» или «естественный», с помощью метода ReadFromFile будем создавать соответствующий объект и стандартным для Tlist методом Add добавлять его в бортовой журнал.

В целях тестирования добавим метод распечатки журнала в консоль (Print).

Поместим определение класса в отдельный модуль LogBookUnit.

```

unit LogBookUnit;

interface
uses Generics.Collections, SpaceObjectUnit;
type
    TLogBook=class(TObject)
    List:TList<TSpaceObject>;
    constructor Create;
    procedure LoadFromFile(FileName:string);
    procedure Print;
    destructor Destroy;
end;

implementation

constructor TLogBook.Create;
begin
    List:=TList<TSpaceObject>.Create;
end;

procedure TLogBook.LoadFromFile(FileName: string);
var
    f:TextFile;
    objType:string;
    obj:TSpaceObject;
begin

```

```

AssignFile(f,FileName);
Reset(f);
while not eof(f) do
begin
  readln(f,objType);
  if objType=' ' then Continue;
  if objType='искусственный' then
    obj:=TArtificialObject.Create
  else
    obj:=TNaturalObject.Create;
  obj.ReadFromFile(f);
  List.Add(obj);
end;
CloseFile(f);
end;

procedure TLogBook.Print;
var
  i: Integer;
begin
  writeln('Бортовой журнал:');
  for i := 0 to List.Count - 1 do
  begin
    writeln;
    List[i].Print;
  end;
end;

destructor TLogBook.Destroy;
var i:integer;
begin
  for i := 0 to List.Count - 1 do
    List[i].Free;
  List.Free;
end;

end.

```

Протестируем работоспособность кода, создав консольное приложение и подключив в него написанные модули: создадим бортовой журнал, загрузим данные из файла, выведем информацию на экран и удалим журнал. Если возникли ошибки, в режиме отладки (пункты меню Run-> Step Over и/или Run-> Trace Into) выполним приложение, найдем и исправим ошибки.

```

program LogBookTestProject;

{$APPTYPE CONSOLE}

uses
  SysUtils, SpaceObjectUnit, LogBookUnit, windows;

var LogBook:TLogBook;
begin
  SetConsoleOutputCP(1251);
  LogBook:=TLogBook.Create;
  LogBook.LoadFromFile('cosmos/журнал.txt');
  LogBook.Print;
  LogBook.Free;
  readln;
end.

```

## Пункт 2

Для упорядочения списка объектов воспользуемся методом Sort класса TList<T>, который реализует обобщенный алгоритм сортировки. Вспомним, что метод Sort требует в качестве параметра объект класса TComparer<T> с методом Compare, умеющим сравнивать два объекта из списка. Чтобы не объявлять два дополнительных класса для реализации двух видов сравнения (по имени и по диаметру), воспользуемся методом Construct класса TComparer, который конструирует требуемый объект на основе анонимной функции сравнения двух элементов списка. Добавим в класс TLogBook методы SortByName и SortByDiameter, которые внутри вызывают метод Sort списка с соответствующей анонимной функцией.

```
procedure TLogBook.SortByName;
begin
  if List.Count>0 then
    List.Sort(TComparer<TSpaceObject>.Construct(
      function (const Left,Right:TSpaceObject):integer
      begin
        result:=1;
        if(Left.Name<Right.Name) then result:=-1
        else if (Left.Name=Right.Name) then result:=0;
      end
    ));
  end;

procedure TLogBook.SortByDiameter;
begin
  if List.Count>0 then
    List.Sort(TComparer<TSpaceObject>.Construct(
      function (const Left,Right:TSpaceObject):integer
      begin
        result:=Left.Diameter-Right.Diameter;
      end
    ));
  end;
```

Вызовем новые методы в тексте основной программы, чтобы убедиться в их работоспособности, если необходимо, выполним программу в режиме отладки и устраним ошибки.

```
program LogBookTestProject;

{$APPTYPE CONSOLE}

uses
  SysUtils, SpaceObjectUnit, LogBookUnit, windows;

var LogBook:TLogBook;
begin
  SetConsoleOutputCP(1251);
  LogBook:=TLogBook.Create;
  LogBook.LoadFromFile('cosmos/журнал.txt');
  LogBook.Print;
```

```

LogBook.SortByName;
writeln;
writeln('Упорядоченные по названию');
writeln('-----');
writeln('нажмите любую клавишу, чтобы продолжить...');
readln;
LogBook.Print;

LogBook.SortByDiameter;
writeln;
writeln('Упорядоченные по диаметру');
writeln('-----');
writeln('нажмите любую клавишу, чтобы продолжить...');
readln;
LogBook.Print;

LogBook.Free;
readln;
end.

```

### Пункт 3

Поиск объектов по заданному диапазону диаметров можно интерпретировать как наложение фильтра (критерия) при просмотре списка. Одно из возможных решений следующее: создадим копию списка List в виде скрытого поля (SourceList) и метод Filter, который будет «отфильтровывать» подходящие объекты и добавлять их в открытый список List.

Созданные ранее методы сортировки и вывода на экран, при работе с открытым списком не изменятся. Потребуется только изменить конструктор, деструктор и метод загрузки информации из файла: теперь поступающие из файла объекты будут формировать скрытый список, после чего будет применен пустой фильтр, который добавит все объекты в открытый список тоже.

Метод Filter будет иметь два параметра — минимальный и максимальный диаметр, если значение отрицательное или равно нулю, то будем считать это отсутствием ограничения (фильтра).

```

unit LogBookUnit;

interface
uses Generics.Collections, Generics.Defaults, SpaceObjectUnit;
type
TLogBook=class(TObject)
private
SourceList: TList<TSpaceObject>;
public
List:TList<TSpaceObject>;
constructor Create;
procedure LoadFromFile(fileName:string);
procedure Print;
procedure SortByName;
procedure SortByDiameter;
procedure Filter(MinDiameter:integer=0;MaxDiameter:integer=0);
destructor Destroy;
end;

```

implementation

```
constructor TLogBook.Create;
begin
  SourceList:=TList<TSpaceObject>.Create;
  List:=TList<TSpaceObject>.Create;
end;

procedure TLogBook.LoadFromFile(FileName: string);
var
  f:TextFile;
  objType:string;
  obj:TSpaceObject;
begin
  AssignFile(f,FileName);
  Reset(f);
  while not eof(f) do
  begin
    readln(f,objType);
    if objType=" then Continue;
    if objType='искусственный' then
      obj:=TArtificialObject.Create
    else
      obj:=TNaturalObject.Create;
    obj.ReadFromFile(f);
    SourceList.Add(obj);
  end;
  CloseFile(f);
  Filter;
end;

procedure TLogBook.Print;
var
  i: Integer;
begin
  writeln('Бортовой журнал:');
  for i := 0 to List.Count - 1 do
  begin
    writeln;
    list[i].Print;
  end;
end;

destructor TLogBook.Destroy;
var i:integer;
begin
  for i := 0 to SourceList.Count - 1 do
    SourceList[i].Free;
  SourceList.Free;
  List.Free;
end;

procedure TLogBook.SortByName;
begin
  if List.Count>0 then
    List.Sort(TComparer<TSpaceObject>.Construct(
      function (const Left,Right:TSpaceObject):integer
      begin
        result:=1;
        if(Left.Name<Right.Name) then result:=-1
        else if (Left.Name=Right.Name) then result:=0;
      end
```

```

));
end;

procedure TLogBook.SortByDiameter;
begin
  if List.Count>0 then
    List.Sort(TComparer<TSpaceObject>.Construct(
      function (const Left,Right:TSpaceObject):integer
      begin
        result:=Left.Diameter-Right.Diameter;
      end
    ));
  end;

procedure TLogBook.Filter(MinDiameter: Integer = 0; MaxDiameter: Integer = 0);
var i:integer;
begin
  List.Clear; //очищаем список
  for i := 0 to SourceList.Count - 1 do
    if (MinDiameter<=0) or (SourceList[i].Diameter>=MinDiameter) then
      if (MaxDiameter<=0) or (SourceList[i].Diameter<=MaxDiameter) then
        List.Add(sourceList[i]);
  end;
end.

```

Тестируем и отлаживаем код из основной программы.

```

program LogBookTestProject;

{$APPTYPE CONSOLE}

uses
  SysUtils, SpaceObjectUnit, LogBookUnit, windows;

var LogBook:TLogBook;
begin
  SetConsoleOutputCP(1251);
  LogBook:=TLogBook.Create;
  LogBook.LoadFromFile('..\cosmos/журнал.txt');
  LogBook.Print;

  LogBook.SortByName;
  writeln;
  writeln('Упорядоченные по названию');
  writeln('-----');
  writeln('нажмите любую клавишу, чтобы продолжить...');
  readln;
  LogBook.Print;

  LogBook.SortByDiameter;
  writeln;
  writeln('Упорядоченные по диаметру');
  writeln('-----');
  writeln('нажмите любую клавишу, чтобы продолжить...');
  readln;
  LogBook.Print;

  LogBook.Filter(1000,0);
  writeln;
  writeln('Список объектов с диаметром из диапазона [1000,...]:');
  writeln('-----');

```

```
writeln('нажмите любую клавишу, чтобы продолжить...');
readln;
LogBook.Print;

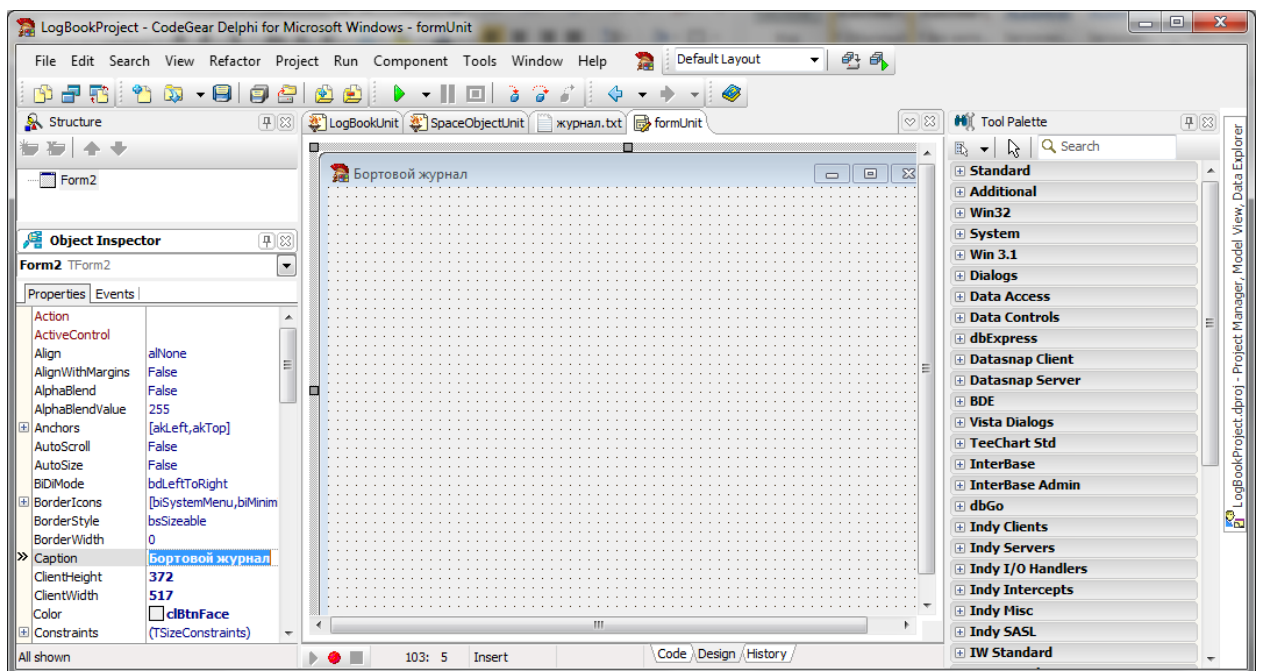
LogBook.Free;
writeln;
writeln('нажмите любую клавишу, чтобы завершить программу...');
readln;
end.
```

## Пункт 4

Так как показать изображение возможно только в графическом режиме, то этот пункт будет реализован в процессе написания VCL-приложения с использованием разработанных модулей.

## Реализация графического интерфейса

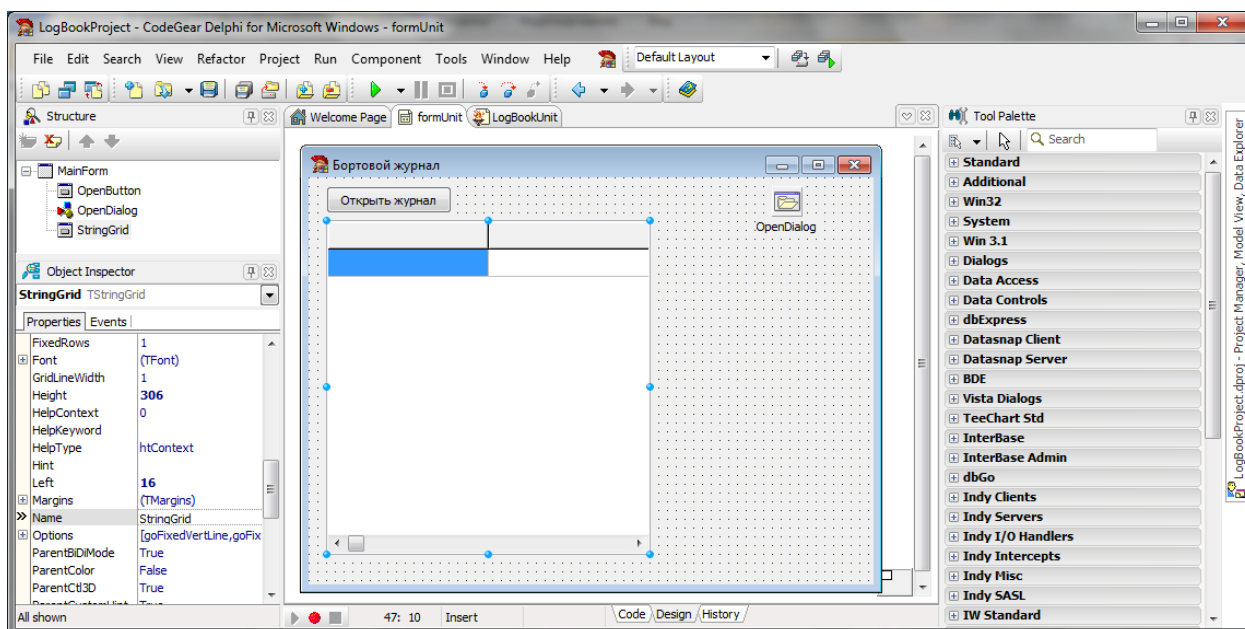
Создадим графическое приложение (VCL Forms Application), в заготовке приложения создается главная форма приложения, изменим в инспекторе объекта (Object Inspector) заголовок (Caption) формы на «Бортовой журнал», а имя (Name) формы на MainForm.



Для открытия текстового файла нам потребуется компонент TOpenDialog, выберем его из коллекции Dialogs в палитре инструментов (Tool Palette) и добавим на форму, изменим его имя на OpenFileDialog.

Выберем из коллекции Standard компонент-кнопку (TButton) и тоже добавим его на форму, изменим ее заголовок на «Открыть журнал», а имя на OpenButton. При нажатии на эту кнопку пользователю будет предложено выбрать текстовый файл, в котором хранится информация бортового журнала.

Для отображения списка объектов можно использовать компонент-таблицу (TStringGrid из коллекции Additional), добавим его на форму и настроим внешний вид. В инспекторе объекта изменим свойства: количество колонок (ColCount = 2), фиксированных строк (FixedRows = 1), фиксированных колонок (FixedCols = 0). Одна фиксированная строка будет служить шапкой таблицы, в ней будут содержаться заголовки колонок.



Для кнопки «Открыть журнал» реализуем обработчик события OnClick — процедуру, которая будет выполняться при нажатии кнопки. Для этого в инспекторе объекта для кнопки переключимся на вкладку Events и дважды щелкнем мышью в строке OnClick: будет создана заготовка — пустой метод с именем OpenButtonClick. Добавим в него код, который предлагает пользователю выбрать текстовый файл (вызовем метод Execute объекта OpenFileDialog), имя файла при успешном выборе будет сохранено в свойстве FileName объекта OpenFileDialog. После чего из этого файла загрузим журнал, а затем, используя журнал, заполним таблицу.

Переключимся на вкладку код (Code) для формы: перед нами модуль, содержащий код формы, которую мы редактировали визуально. Экземпляр формы хранится в переменной MainForm. Объявим тут же переменную типа TLogBook для хранения журнала, подключив ранее написанные модули. Создание объекта поместим в обработчик события OnCreate формы (код выполнится, как только форма будет создана).

Вручную создадим метод формы ShowLogBook выводящий в таблицу информацию бортового журнала (названия и диаметры объектов). Добавим вызов этого метода после того, как журнал будет загружен из файла.

```
unit formUnit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
```



```

Dialogs, Grids, StdCtrls, LogBookUnit;

type
  TMainForm = class(TForm)
    OpenFileDialog: TOpenDialog;
    OpenButton: TButton;
    StringGrid: TStringGrid;
    procedure OpenButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure ShowLogBook;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  MainForm: TMainForm;
  LogBook: TLogBook;
implementation

{$R *.dfm}

procedure TMainForm.FormCreate(Sender: TObject);
begin
  LogBook := TLogBook.Create;
end;

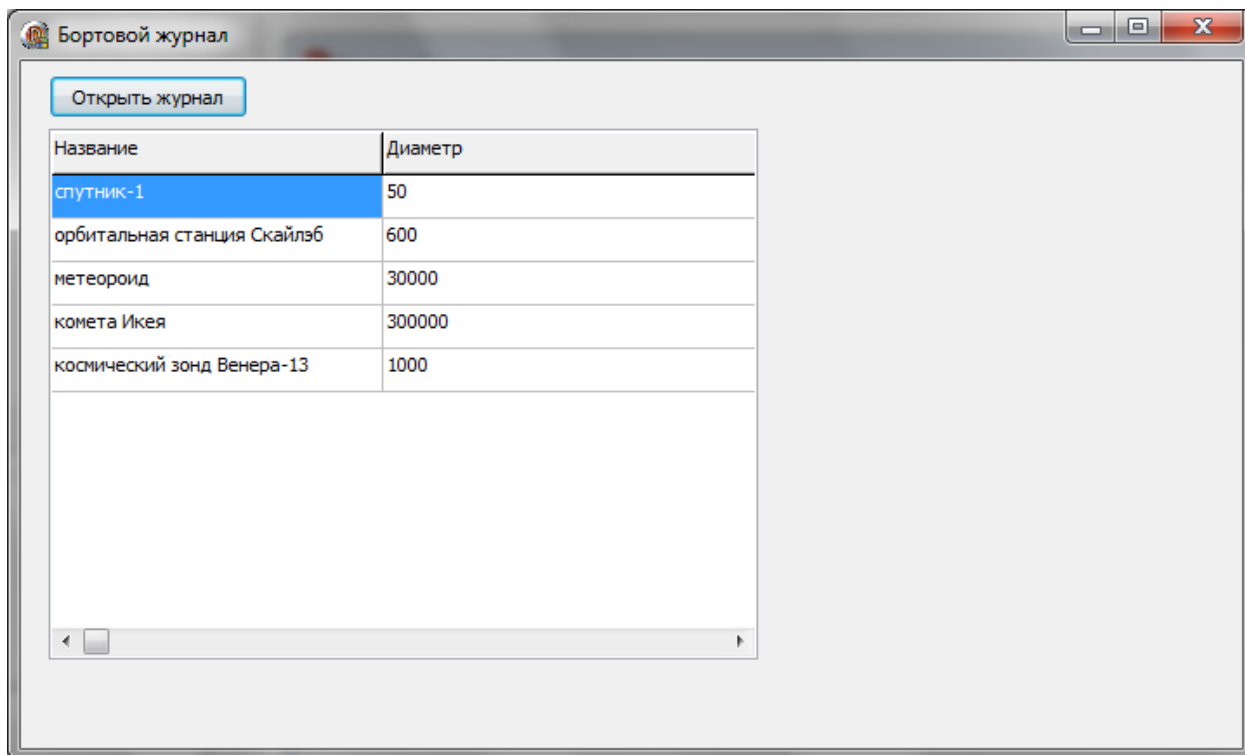
procedure TMainForm.OpenButtonClick(Sender: TObject);
begin
  if OpenFileDialog.Execute then
  begin
    LogBook.LoadFromFile(OpenDialog.FileName);
    ShowLogBook;
  end;
end;

procedure TMainForm.ShowLogBook;
var i: integer;
begin
  StringGrid.RowCount := LogBook.list.Count + 1;
  StringGrid.Cells[0, 0] := 'Название';
  StringGrid.Cells[1, 0] := 'Диаметр';
  for i := 0 to LogBook.List.Count - 1 do
  begin
    StringGrid.Cells[0, i + 1] := LogBook.List[i].Name;
    StringGrid.Cells[1, i + 1] := IntToStr(LogBook.List[i].Diameter);
  end;
end;

end.

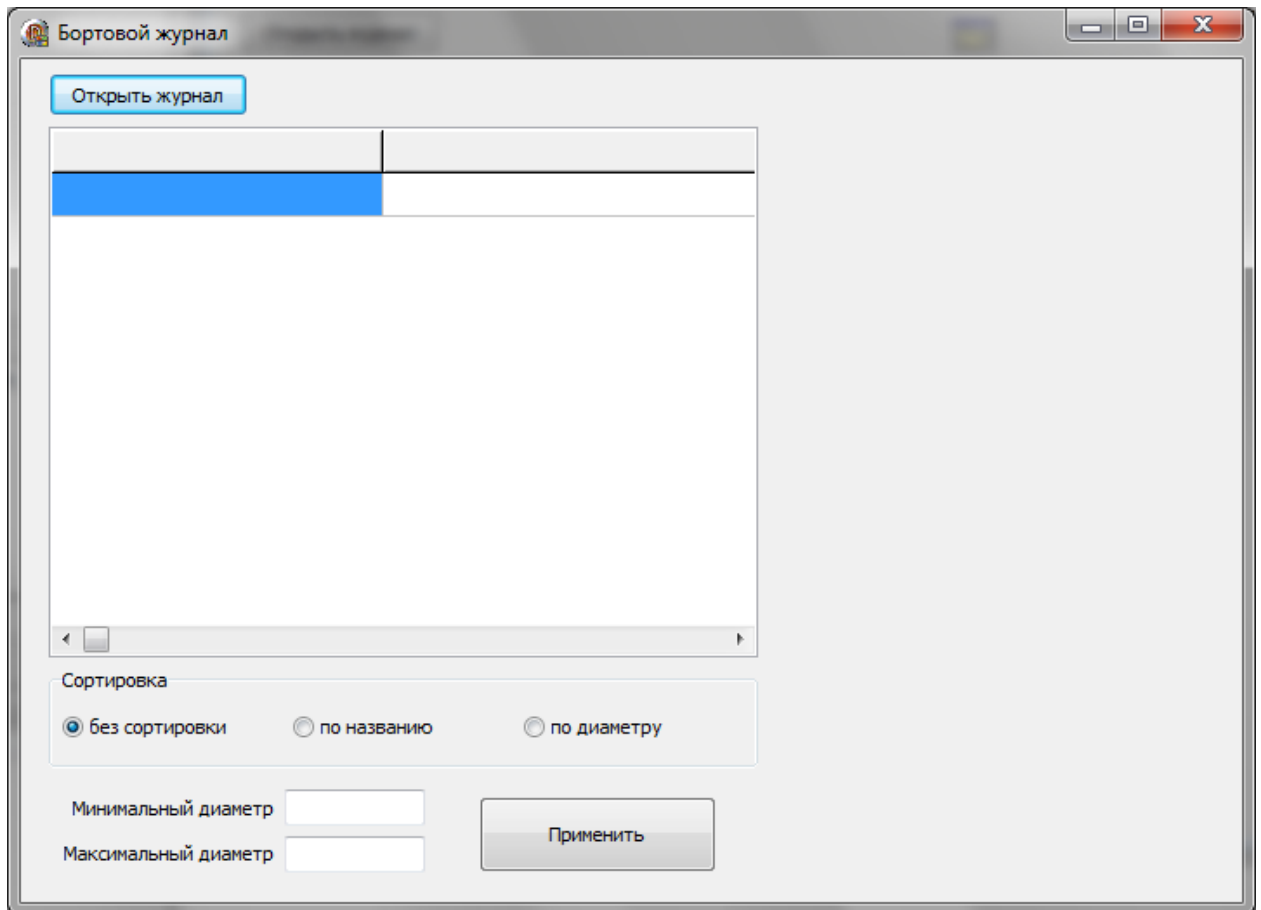
```

Запустим проект, проверим работоспособность, исправим ошибки, если требуется.



Для выбора параметров сортировки воспользуемся компонентом `TRadioGroup` из стандартной коллекции компонентов. Добавим объект на форму и настроим его свойства в инспекторе объекта: заголовок (`Caption`) — «Сортировка», число колонок (`Columns`) — 3. В свойство элементы (`Items`) добавим 3 варианта: без сортировки, по названию, по диаметру. Выбранным элементом по умолчанию укажем «без сортировки» (`ItemIndex=0`).

Для выбора параметров фильтрации добавим два текстовых поля ввода (`TEdit`) из стандартной коллекции компонентов, переименуем их в `MinDiameter` и `MaxDiameter` соответственно. Установим у них свойство `NumbersOnly=true` (чтобы пользователь мог ввести только числовые символы). Добавим рядом с этими полями пояснительные подписи — компоненты `TLabel` и кнопку «Применить». Создадим обработчик события нажатия этой кнопки, в котором просто вызовем имеющийся метод вывода журнала `ShowLogBook`.



Логике метода ShowLogBook доработаем с учетом условий сортировки и фильтрации.

```

procedure TMainForm.ShowLogBook;
var i:integer;
    min,max:integer;
    s:string;
begin
    s:=Trim(MinDiameter.Text);
    if s="" then s:='0';
    min:=StrToInt(s);
    s:=Trim(MaxDiameter.Text);
    if s="" then s:='0';
    max:=StrToInt(s);
    LogBook.Filter(min,max);
    if SortRadioGroup.ItemIndex=1 then
        LogBook.SortByName;
    if SortRadioGroup.ItemIndex=2 then
        LogBook.SortByDiameter;
    StringGrid.RowCount:=LogBook.list.Count+1;
    StringGrid.Cells[0,0]:='Название';
    StringGrid.Cells[1,0]:='Диаметр';
    for i := 0 to LogBook.List.Count - 1 do
    begin
        StringGrid.Cells[0,i+1]:=LogBook.List[i].Name;
        StringGrid.Cells[1,i+1]:=IntToStr(LogBook.List[i].Diameter);
    end;
end;

```

Запустим, проверим работоспособность, исправим ошибки.

Название	Диаметр
комета Икея	300000
космический зонд Венера-13	1000
метеороид	30000

Сортировка

☐ без сортировки
 ☒ по названию
 ☐ по диаметру

Минимальный диаметр: 1000  
 Максимальный диаметр:   
 Применить

Осталось вывести подробную информацию о выбранном объекте.

Для таблицы назначим обработчик события `OnSelectCell`, этот метод будет выполняться при выборе новой ячейки таблицы мышью, либо перемещением кнопками клавиатуры. В параметрах метода, можно увидеть,

```
procedure TMainForm.StringGridSelectCell(Sender: TObject; ACol, ARow: Integer;
var CanSelect: Boolean);
```

передаются номер колонки и строки выбранной ячейки (`ACol` и `ARow`).

В свойстве `Options` таблицы для красоты можно указать `goRowSelect=true` (при выборе ячейки будет подсвечиваться строка целиком).

Добавим на форму компоненты (например, `TLabel`) для вывода года, страны производства, возраста, типа объекта, можно продублировать название и диаметр. Доработаем код обработчика, зная, что номер объекта в бортовом журнале равен `ARow-1`.

```
procedure TMainForm.StringGridSelectCell(Sender: TObject; ACol, ARow: Integer;
var CanSelect: Boolean);
var i: integer;
    spaceObj: TSpaceObject;
    artObj: TArtificialObject;
    natObj: TNaturalObject;
begin
    //очищаем все поля
```

```

TypeLabel.Caption:="";
NameLabel.Caption:="";
DiameterLabel.Caption:="";
YearLabel.Caption:="";
CountryLabel.Caption:="";
AgeLabel.Caption:="";

//если есть выделенный объект, то выводим его информацию
i:=ARow-1;
if (i>=0) and (LogBook.list.Count>i) then
begin
  spaceObj:=LogBook.list.Items[i];
  NameLabel.Caption:=spaceObj.Name;
  DiameterLabel.Caption:=IntToStr(spaceObj.Diameter) + ' см.';
  if spaceObj.ClassType=TArtificialObject then
  begin
    artObj:=TArtificialObject(spaceObj);
    TypeLabel.Caption:='искусственный';
    YearLabel.Caption:=IntToStr(artObj.Year);
    CountryLabel.Caption:=artObj.Country;
  end
  else
  begin
    natObj:=TNaturalObject(spaceObj);
    TypeLabel.Caption:='естественный';
    AgeLabel.Caption:=natObj.Age;
  end;
end
end;

```

Для вывода изображения добавим компонент TImage из коллекции компонентов Additional. У этого компонента есть свойство Picture, которое, в свою очередь, имеет метод загрузки картинки из файла (LoadFromFile).

```
Image.Picture.LoadFromFile('имя файла');
```

В бортовом журнале объекты хранят имена файлов с изображениями, сами файлы находятся в том же каталоге, где и текстовый файл. Т.е. при выборе текстового файла пользователем, нужно запоминать каталог, где он лежит. Добавим свойство Catalog (каталог) в класс TLogBook и будем его запоминать при загрузке информации из файла (воспользуемся функцией ExtractFileDir модуля SysUtils).

```

procedure TLogBook.loadFromFile(fileName: string);
var
  f:TextFile;
  objType:string;
  obj:TSpaceObject;
begin
  Catalog:=ExtractFileDir(fileName);
  sourceList.Clear;
  AssignFile(f,fileName);
  ...

```

При выборе объекта в таблице добавим загрузку файла с изображением. Для отображения файлов в формате JPEG требуется подключить модуль Jpeg.

```

unit formUnit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, StdCtrls, LogBookUnit, SpaceObjectUnit, ExtCtrls, Jpeg;

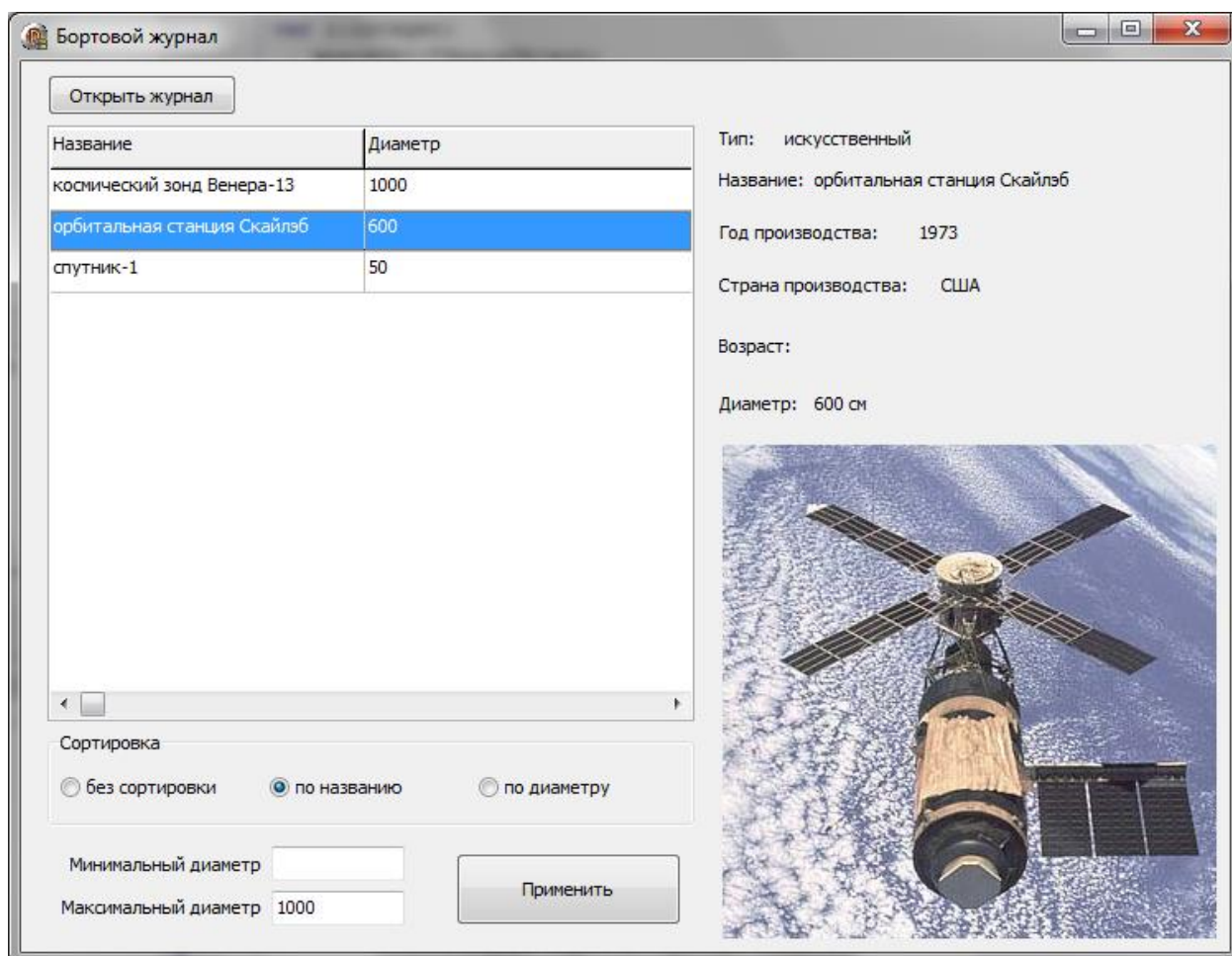
...

procedure TMainForm.StringGridSelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
var i:integer;
    spaceObj:TSpaceObject;
    artObj:TArtificialObject;
    natObj:TNaturalObject;
    imageFileName:string;
begin
  //очищаем все поля
  TypeLabel.Caption:="";
  NameLabel.Caption:="";
  DiameterLabel.Caption:="";
  YearLabel.Caption:="";
  CountryLabel.Caption:="";
  AgeLabel.Caption:="";
  Image.Picture:=nil;

  //если есть выделенный объект, то выводим его информацию
  i:=ARow-1;
  if (i>=0) and (LogBook.list.Count>i) then
  begin
    spaceObj:=LogBook.list.Items[i];
    NameLabel.Caption:=spaceObj.Name;
    DiameterLabel.Caption:=IntToStr(spaceObj.Diameter) + ' см.';
    imageFileName:=LogBook.Catalog + '\' + spaceObj.Image;
    Image.Picture.LoadFromFile(imageFileName);
    if spaceObj.ClassType=TArtificialObject then
      ...

```

Запускаем, проверяем работоспособность, исправляем ошибки, если требуется.



При разработке реального приложения, нужно понимать, что могут возникнуть разные ситуации, приводящие к ошибкам вышенаписанного кода: текстовый файл может оказаться не того формата, данные в файле могут не соответствовать требуемому содержанию, файла с изображением может не быть и т. д. Для полной, корректной работы приложения все эти ситуации нужно будет предусмотреть и обрабатывать в коде программы.

Обработка исключительных ситуаций (конструкция try ... except) не вошла в список тем данного курса, тем не менее, небольшие доработки в самой общей форме были внесены в конечный программный код. Кроме того, тестирование работы приложения выявило появление несогласованности выделенной строки таблицы и отображаемой детализированной информации об объекте после применения условий сортировки или фильтрации. Необходимые изменения также были внесены и доступны для ознакомления в конечном коде программы.

Код приложения и каталог с тестовыми данными бортового журнала доступны на сайте курса <http://study.sfu-kras.ru/course/view.php?id=121> в разделе «Экзамен».

## Билет

Экзаменационный билет содержит практическое задание (см. в качестве примера типовую задачу) и один из теоретических вопросов. Общее время на выполнение работы — четыре академических часа. Пример экзаменационного билета приводится на следующей странице.

## Литература

1. Буч Г. и др. Объектно-ориентированный анализ и проектирование с примерами приложений. / Д. Ключин. Вильямс, 2010. 720 с.
2. Гамма Э. и др. Приемы объектно-ориентированного проектирования. / А. Слинкин. СПб.: Питер, 2003. 368 с.
3. Фленов М. Е. Библия Delphi. СПб.: БХВ-Петербург, 2011. 688 с.
4. Осипов Д. Delphi XE2. СПб.: БХВ-Петербург, 2012. 912 с.



**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО  
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ  
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Институт математики и фундаментальной информатики**

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 1**

По дисциплине: «Языки и технологии программирования»

Специальность / направление: 010200.62 «Математика. Компьютерные науки»

**ПРАКТИЧЕСКОЕ ЗАДАНИЕ**

На своем пути космический корабль регистрирует в бортовом журнале (см. папку cosmos) данные о встречающихся космических объектах: искусственных и естественных. Об искусственном объекте записывается название, диаметр (в см), год и страна производства, фотография. О естественном — название, диаметр (в см), предположительный возраст и фотография. В среде Delphi с использованием ООП написать программу, предоставляющую удобный интерфейс работы с данной информацией и позволяющую:

1. Организовать данные для дальнейшей удобной, оперативной работы с ними, в частности, просматривать совокупность зарегистрированных объектов с выводом на экран их общих характеристик в соответствии с файлом журнал.txt. (5 баллов)
2. Упорядочивать эту совокупность по названию и диаметру объекта. (5 баллов)
3. Искать объекты в задаваемом интервале диаметров объектов из числа зарегистрированных объектов. (5 баллов)
4. Просматривать фотографию и остальные характеристики выбираемого объекта. (10 баллов)

Дополнительные баллы:

Использование TList (10 баллов) или использование обобщенных классов из стандартной коллекции (15 баллов).

Графический интерфейс. (10 баллов)

В каталоге cosmos (получить у экзаменатора) содержатся текстовый файл «журнал.txt» и файлы с изображениями.

Структура файла «журнал.txt»:

```
искусственный
спутник-1
50
спутник1.jpg
1957
СССР

естественный
метеороид
30000
метеороид.jpg
5 млн. лет

естественный
комета Икея
300000
комета.jpg
20 млн.лет

...
```

**ТЕОРЕТИЧЕСКИЙ ВОПРОС**

Основная концепция и принципы ООП. Абстракция. Класс. Объект.

Составил \_\_\_\_\_/ФИО/

Утверждаю

« » \_\_\_\_\_ г.

Зав.кафедрой \_\_\_\_\_/ФИО/