# Web-технологии в автоматизации предприятий и производств

# Ввведение в технологию разработки веб-приложений ASP.NET WebForms

# Инструментарий

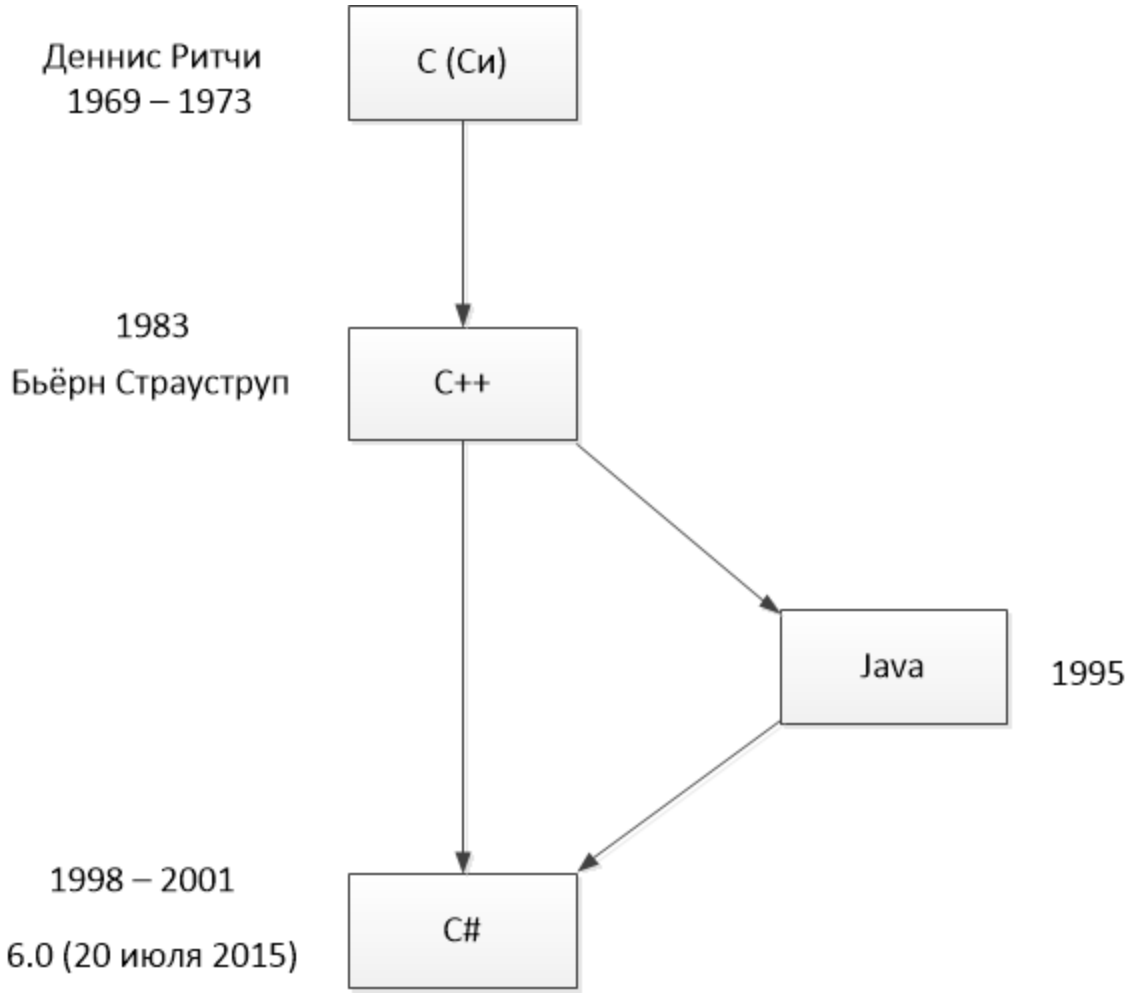
Visual Studio 2017 Community

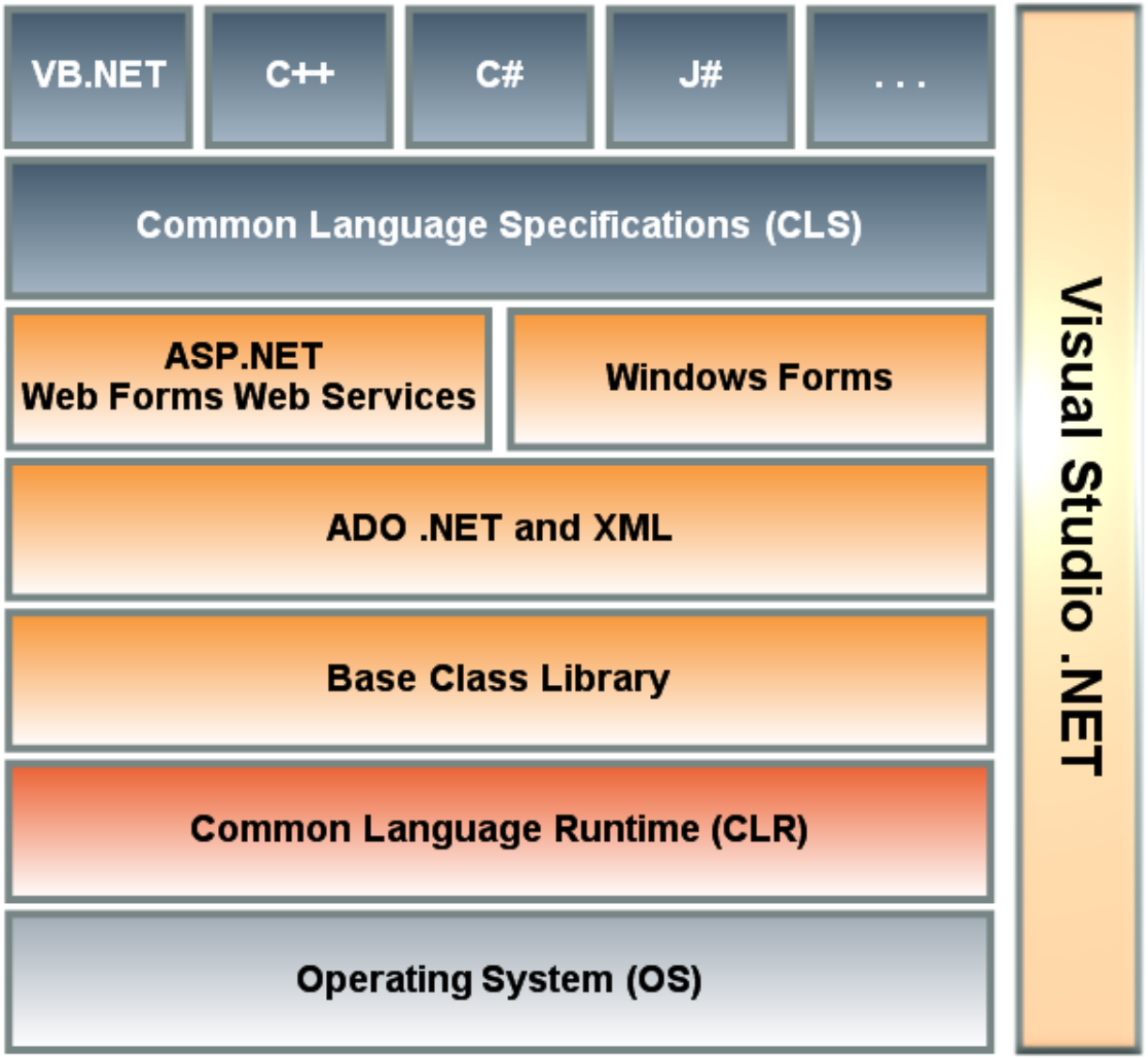
Microsoft SQL Server 2014 Express

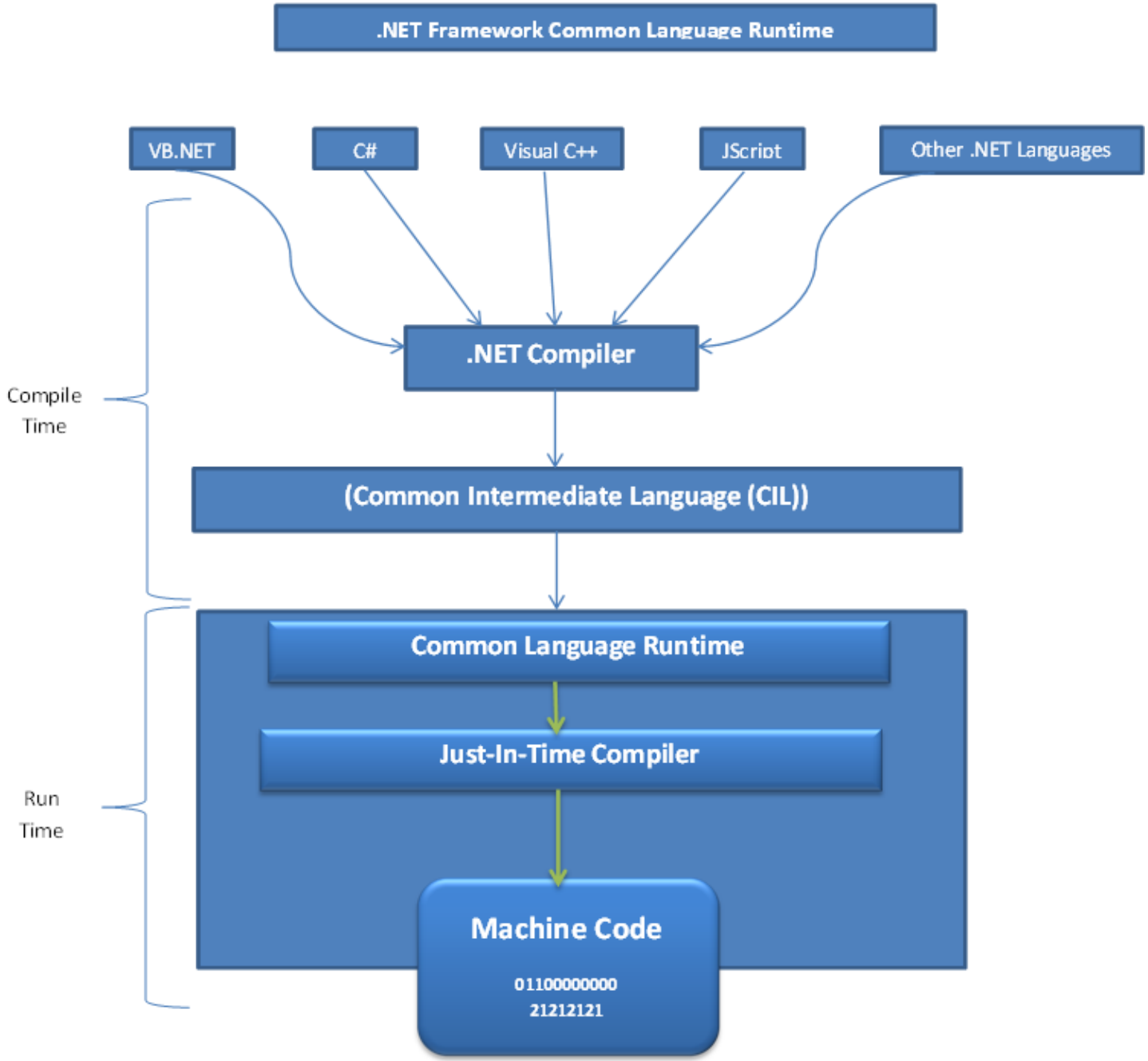
Microsoft SQL Server 2014 Management Studio

# Канал на YouTube
# «Введение в ООП на C# и ASP.NET»

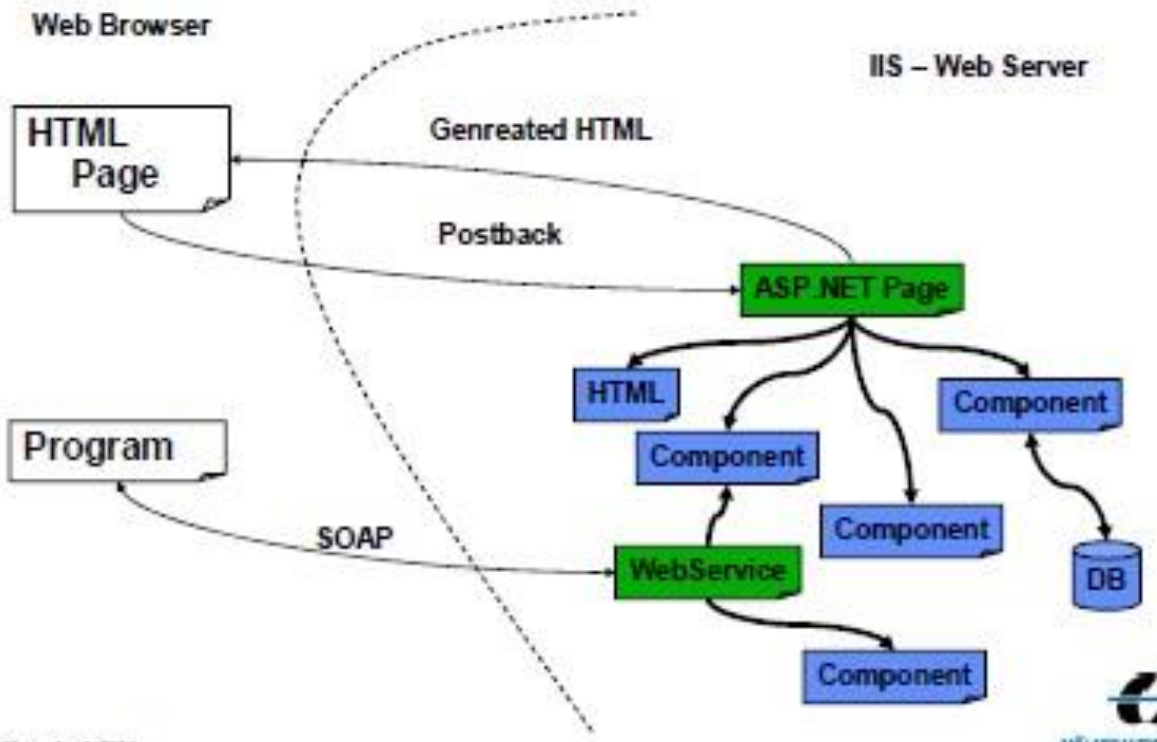
https://www.youtube.com/playlist?list=PLgpowKVDMNLWbYgDxowiHmxLSs1ncgUI8

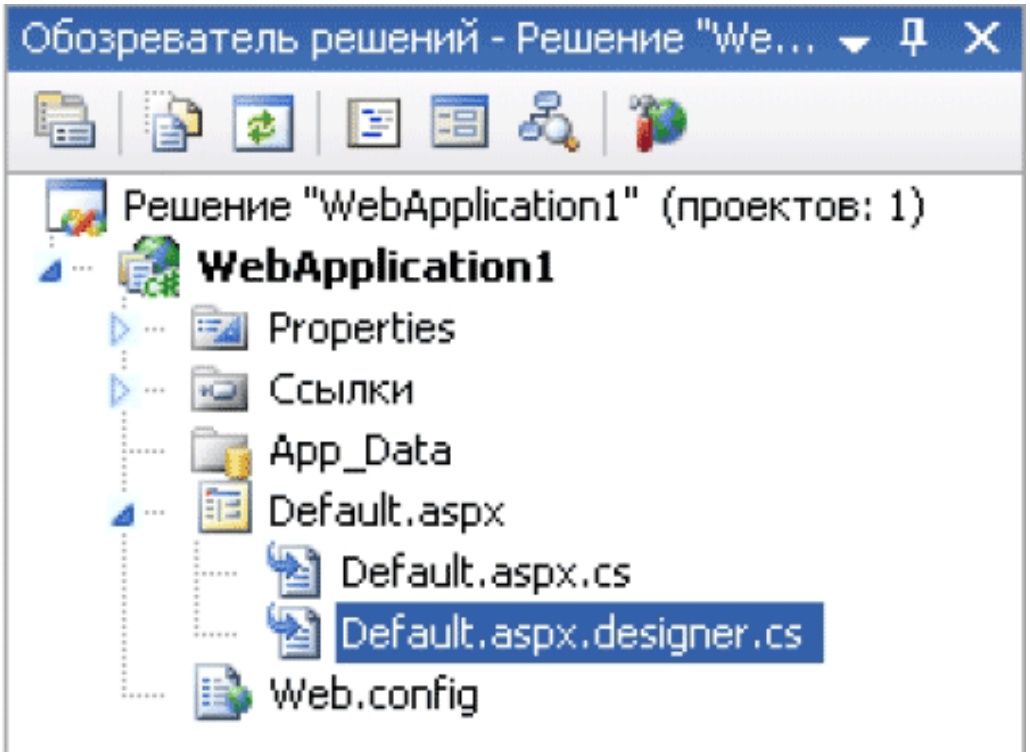

Деннис Ритчи
1969 – 1973
C (Си)
1983
Бьёрн Страуструп
C++
Java
1995
1998 – 2001
6.0 (20 июля 2015)
C#

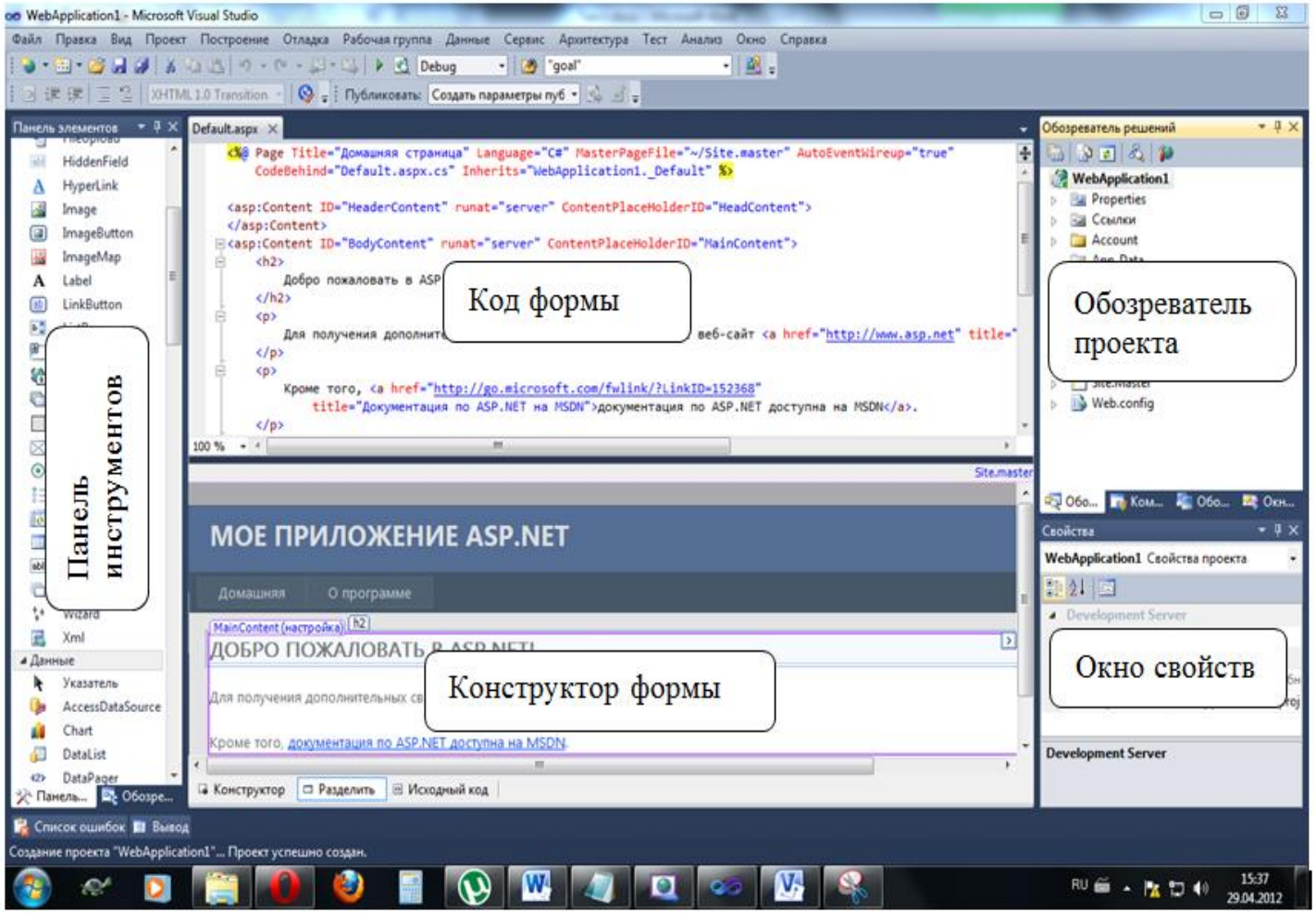

VB.NET
C++
C#
J#
. . .
Common Language Specifications (CLS)
ASP.NET
Web Forms Web Services
Windows Forms
ADO .NET and XML
Base Class Library
Common Language Runtime (CLR)
Operating System (OS)
Visual Studio .NET

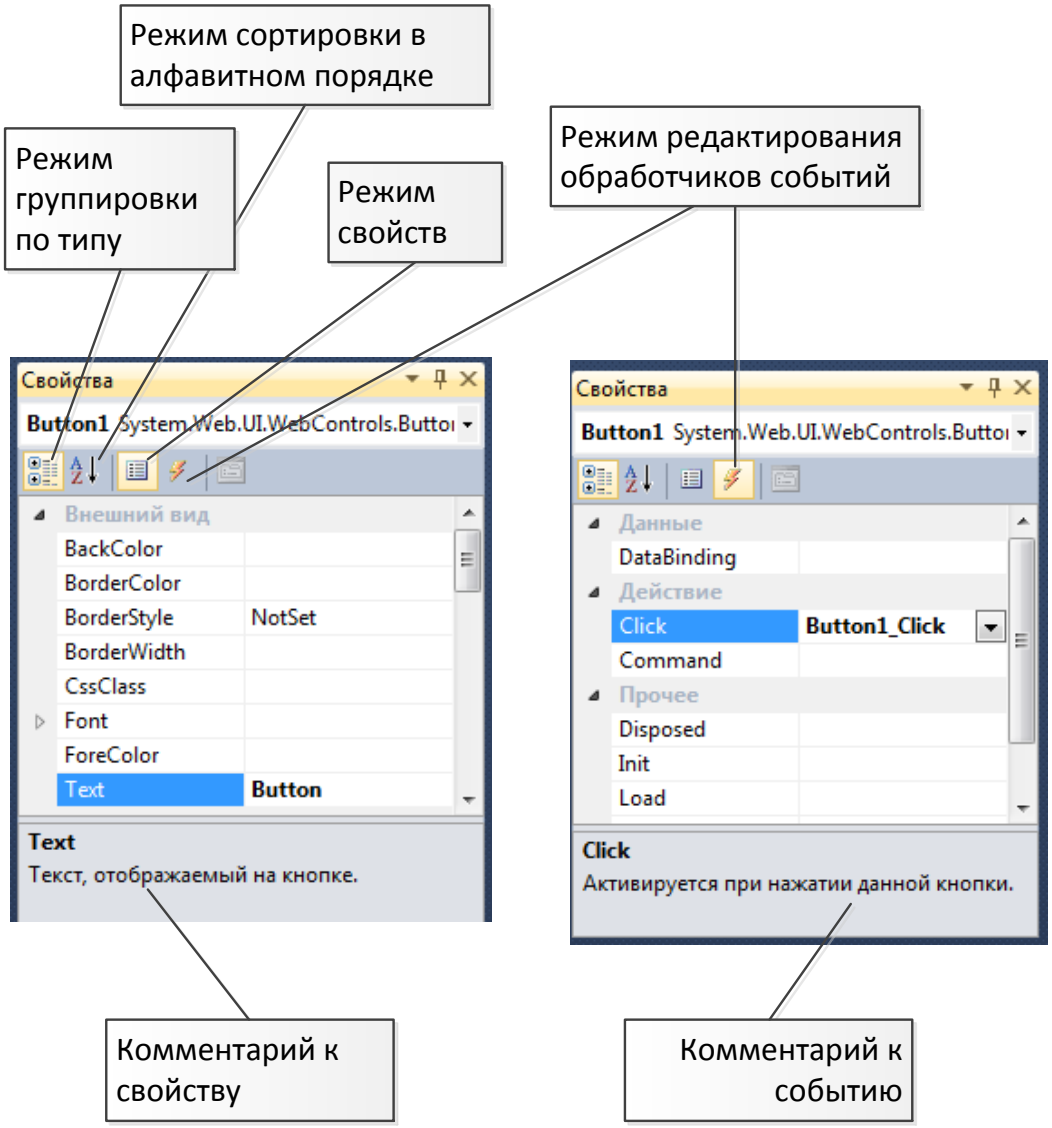

.NET Framework Common Language Runtime
VB.NET
C#
Visual C++
JScript
Other .NET Languages
.NET Compiler
Compile Time
(Common Intermediate Language (CIL))
Common Language Runtime
Just-In-Time Compiler
Run Time
Machine Code
01100000000
21212121

# ASP.NET and Web Forms



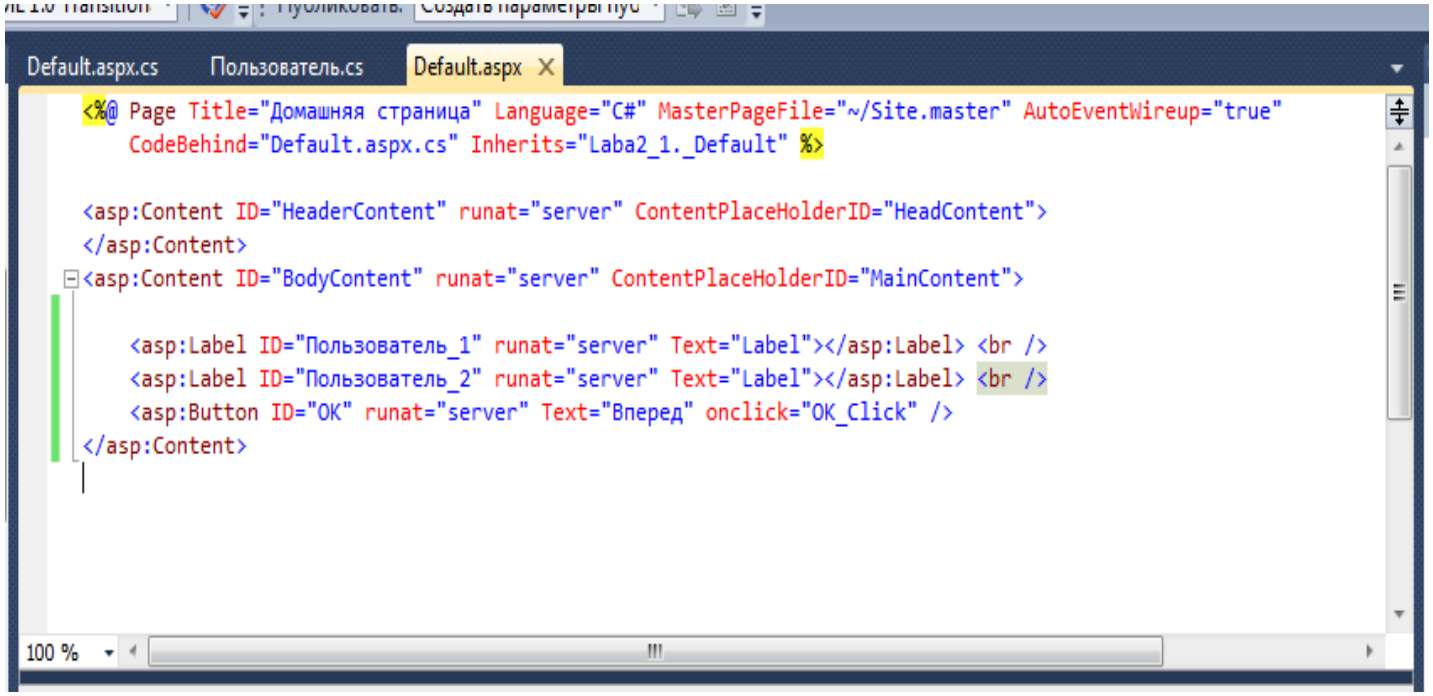

Обозреватель решений - Решение "We...
Решение "WebApplication1" (проектов: 1)
WebApplication1
Properties
Ссылки
App_Data
Default.aspx
Default.aspx.cs
Default.aspx.designer.cs
Web.config

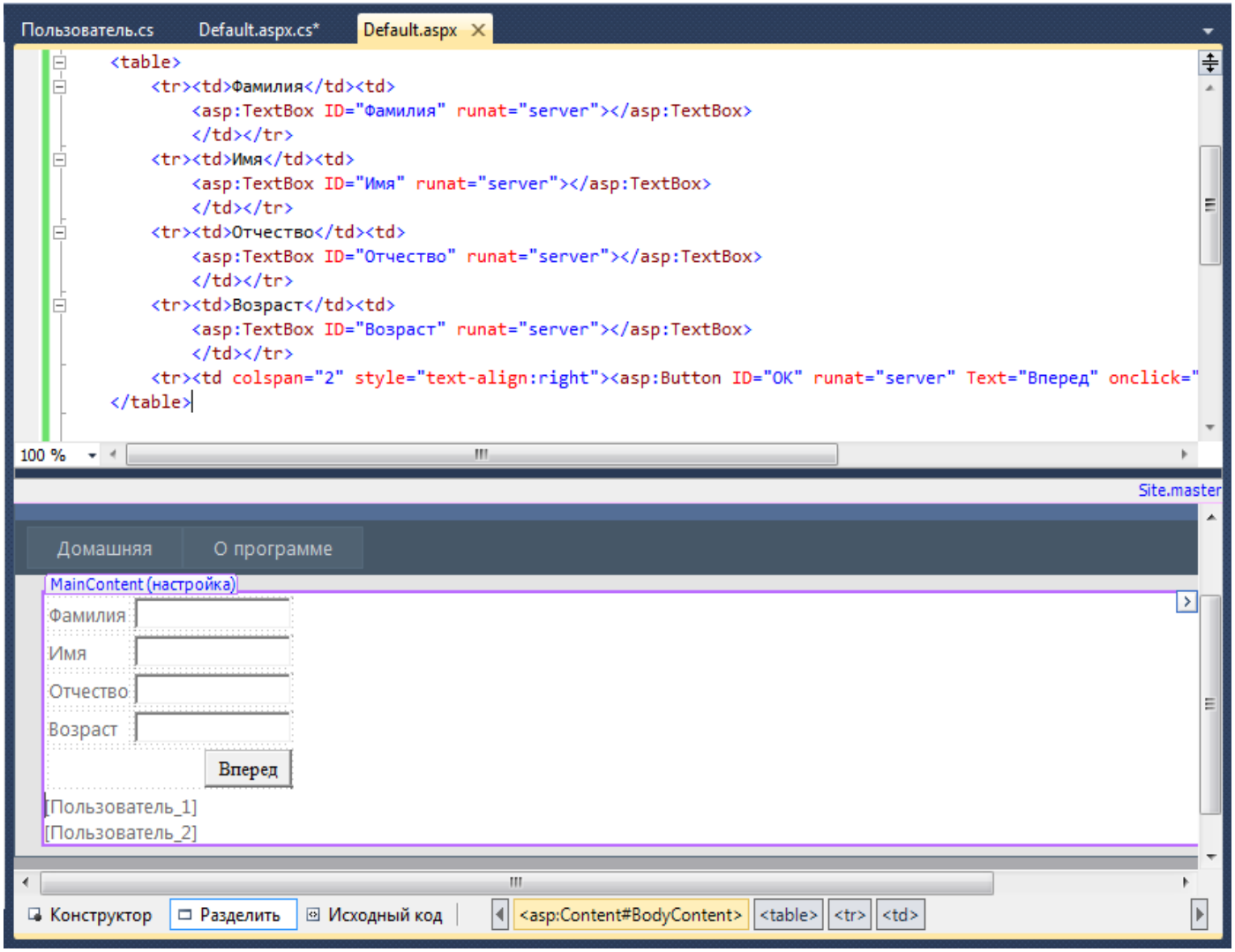

Панель инструментов
Код формы
Обозреватель проекта
Конструктор формы
Окно свойств

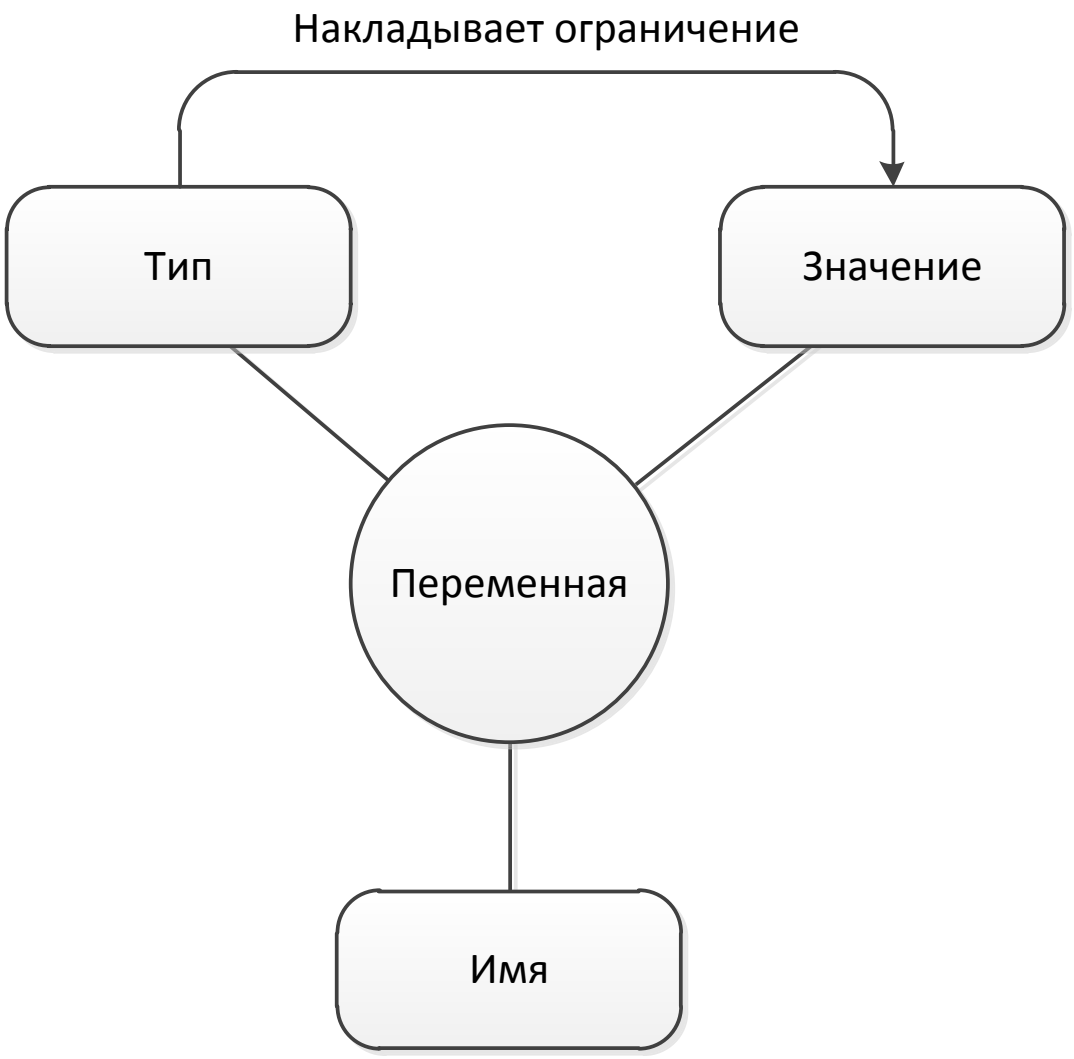



РЕЖИМ РЕДАКТИРОВАНИЯ СВОЙСТВ

РЕЖИМ РЕДАКТИРОВАНИЯ СОБЫТИЙ

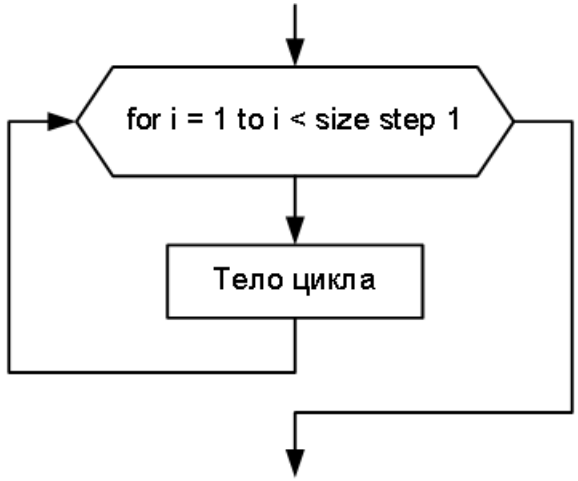

Default.aspx.cs
Пользователь.cs
Default.aspx
<%@ Page Title="Домашняя страница" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="Laba2_1._Default" %>
<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
<asp:Label ID="Пользователь_1" runat="server" Text="Label"></asp:Label> <br />
<asp:Label ID="Пользователь_2" runat="server" Text="Label"></asp:Label> <br />
<asp:Button ID="OK" runat="server" Text="Вперед" onclick="OK_Click" />
</asp:Content>
100 %


Пользователь.cs
Default.aspx.cs*
Default.aspx
<table>
<tr><td>Фамилия</td><td>
<asp:TextBox ID="Фамилия" runat="server"></asp:TextBox>
</td></tr>
<tr><td>Имя</td><td>
<asp:TextBox ID="Имя" runat="server"></asp:TextBox>
</td></tr>
<tr><td>Отчество</td><td>
<asp:TextBox ID="Отчество" runat="server"></asp:TextBox>
</td></tr>
<tr><td>Возраст</td><td>
<asp:TextBox ID="Возраст" runat="server"></asp:TextBox>
</td></tr>
<tr><td colspan="2" style="text-align:right"><asp:Button ID="OK" runat="server" Text="Вперед" onclick="
</table>
100 %
Site.master
Домашняя
О программе
MainContent (настройка)
Фамилия
Имя
Отчество
Возраст
Вперед
[Пользователь_1]
[Пользователь_2]
Конструктор
Разделить
Исходный код
<asp:Content#BodyContent>
<table>
<tr>
<td>

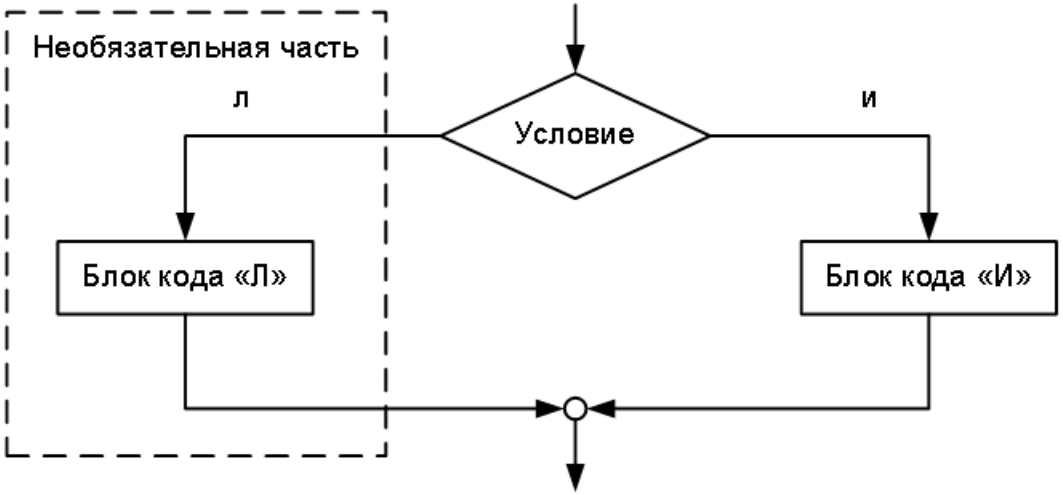

Накладывает ограничение
Тип
Значение
Переменная
Имя



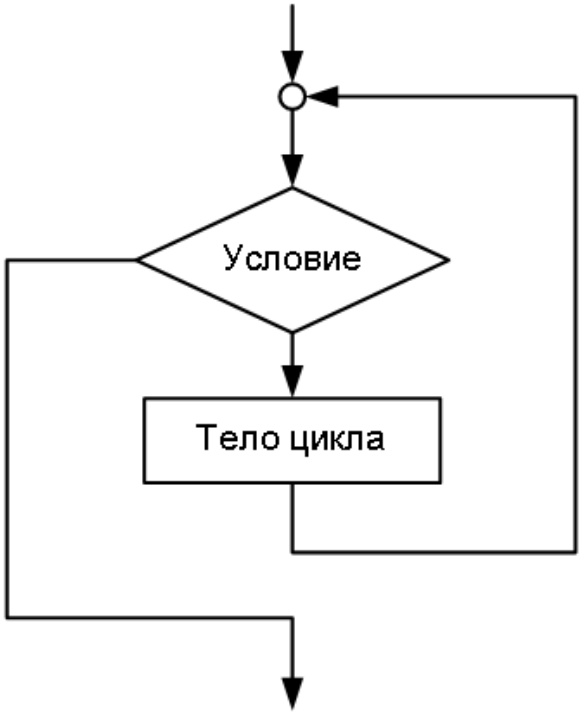
Последовательное выполнение алгоритма



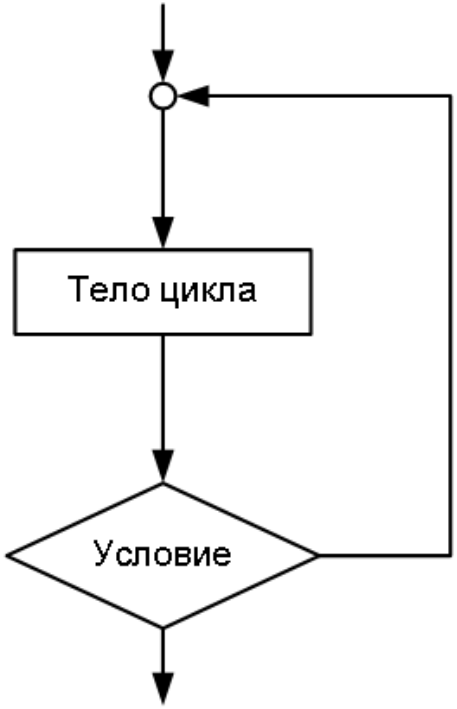

Цикл со счетчиком



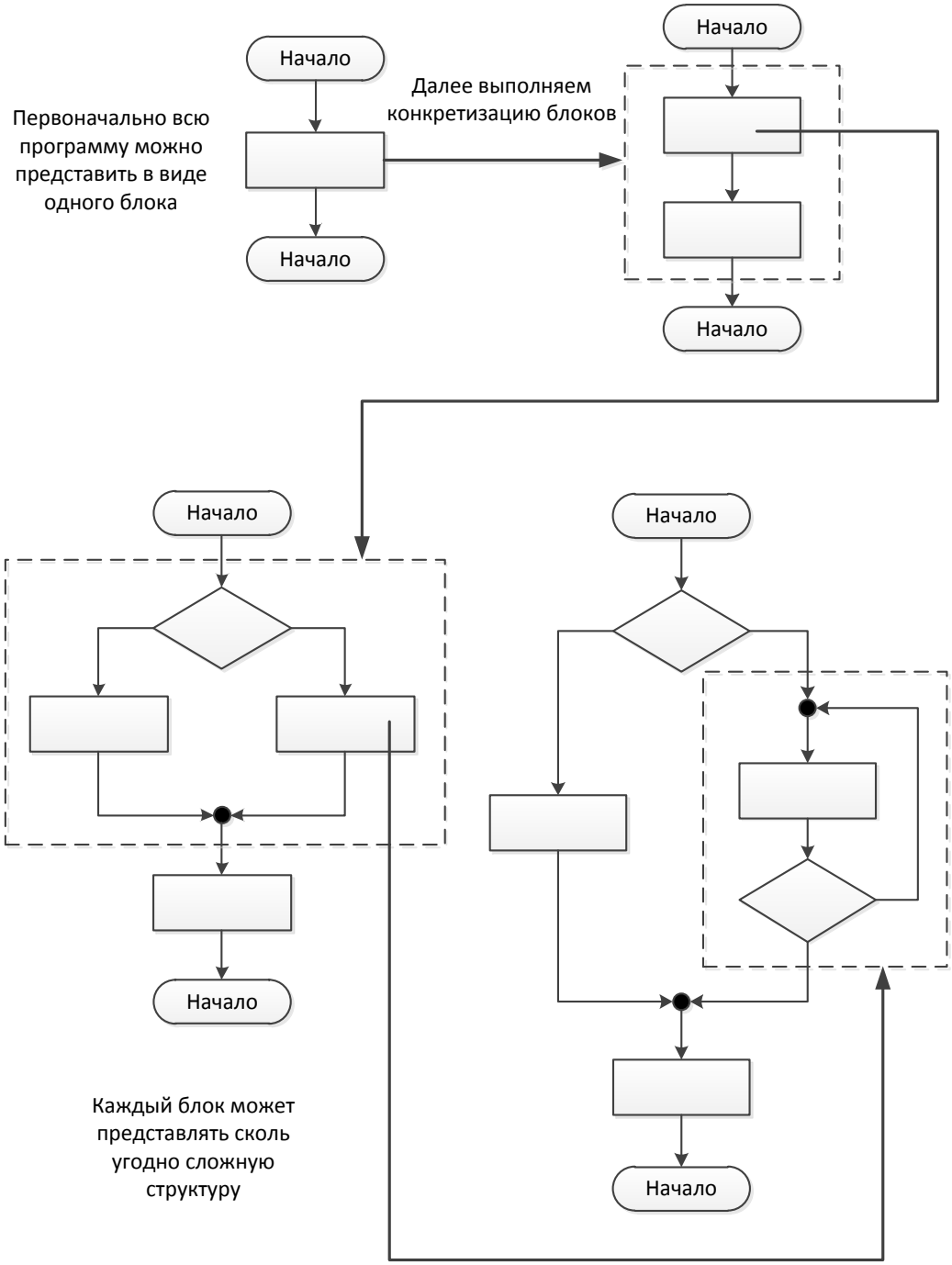

Ветвящаяся структура



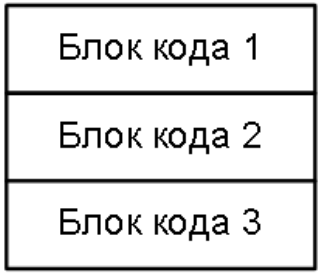

Цикл с предусловием



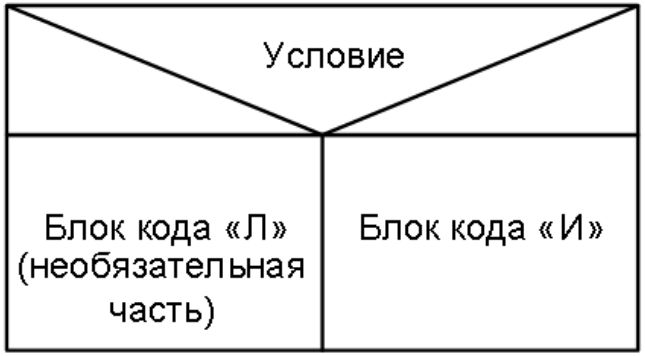

Цикл с предусловием


Начало
Начало
Первоначально всю программу можно представить в виде одного блока
Далее выполняем конкретизацию блоков
Начало
Начало
Начало
Начало
Начало
Начало
Каждый блок может представлять сколь угодно сложную структуру

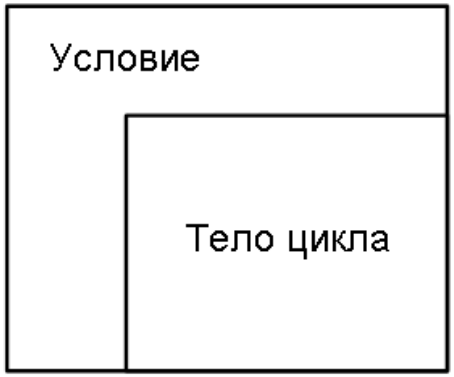
# Диаграммы Насси-Шнайдермана



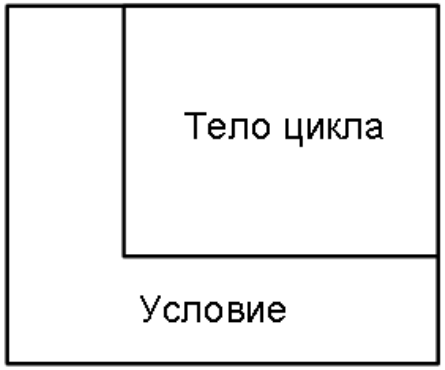

Последовательное выполнение алгоритма



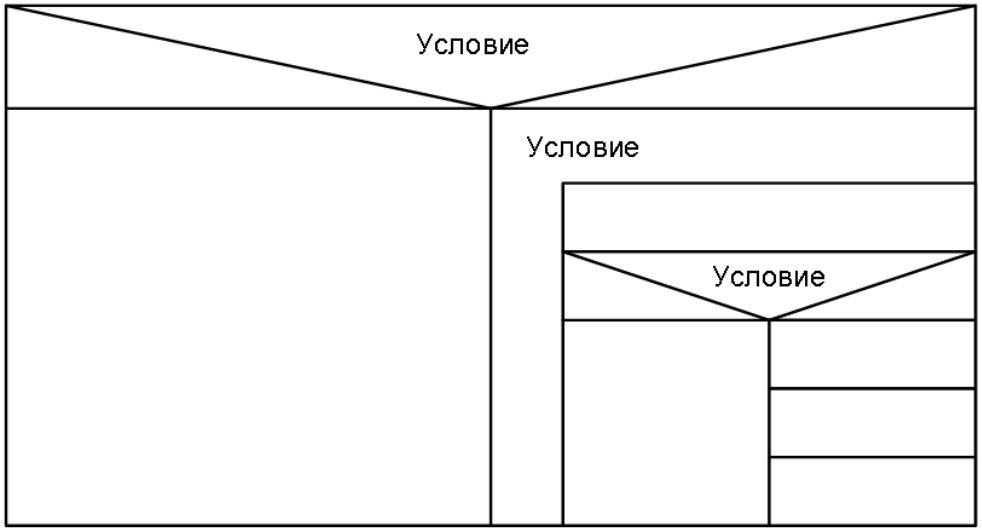

Ветвящаяся структура



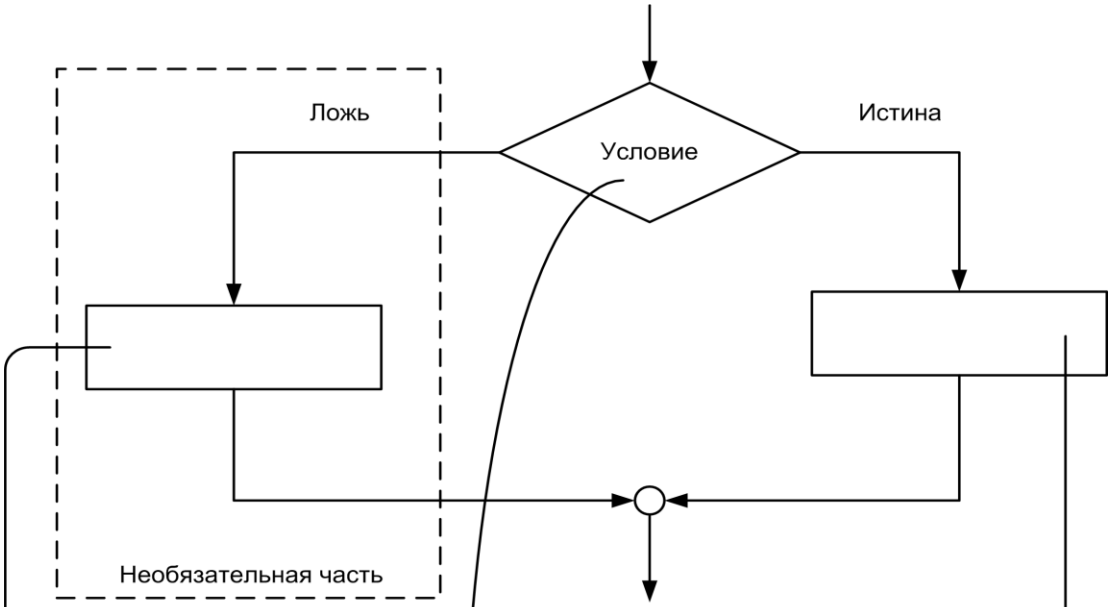

Цикл с предусловием




Цикл с постусловием




Пример вложенных структур диаграмм Насси-Шнайдермана

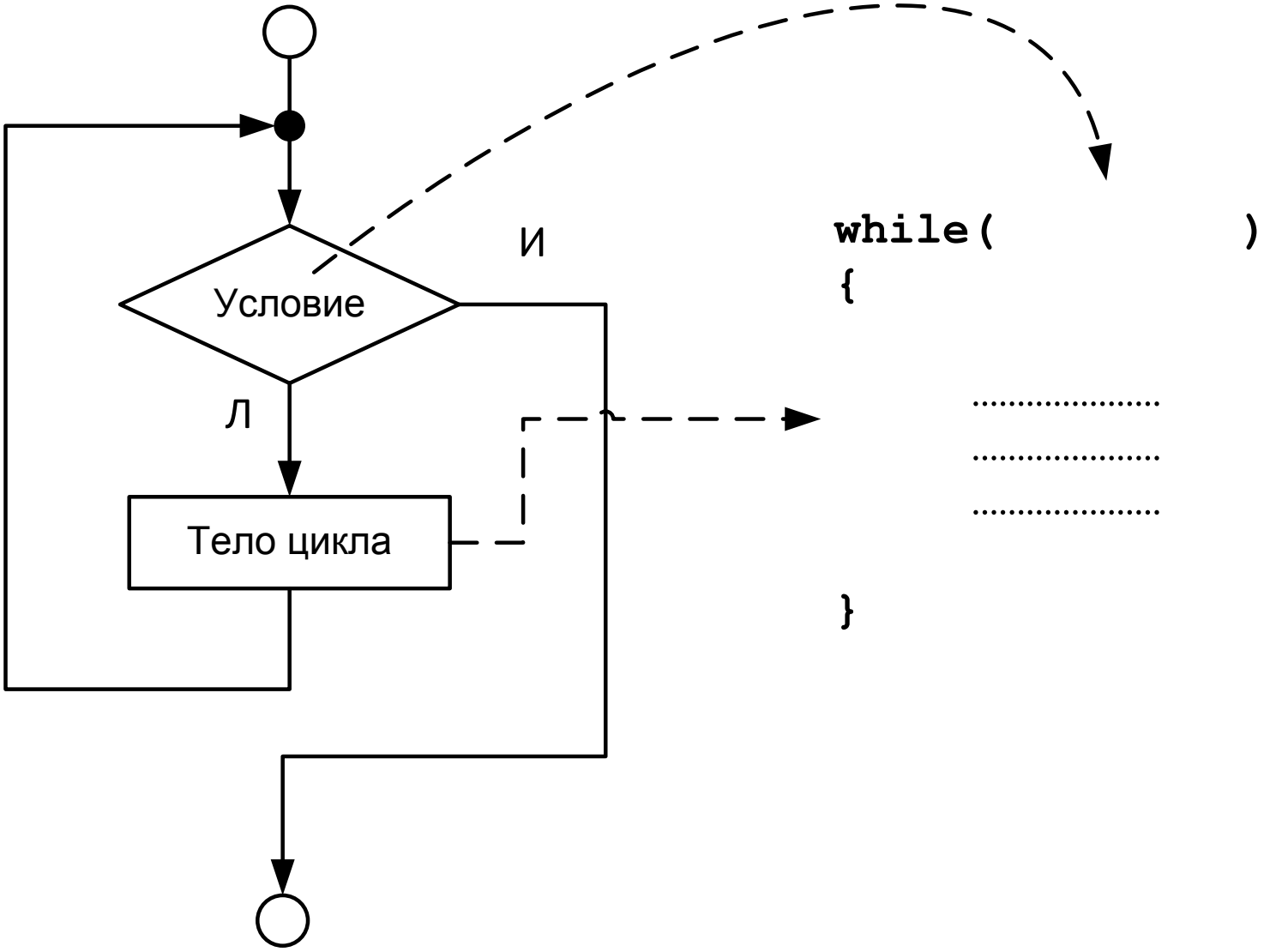


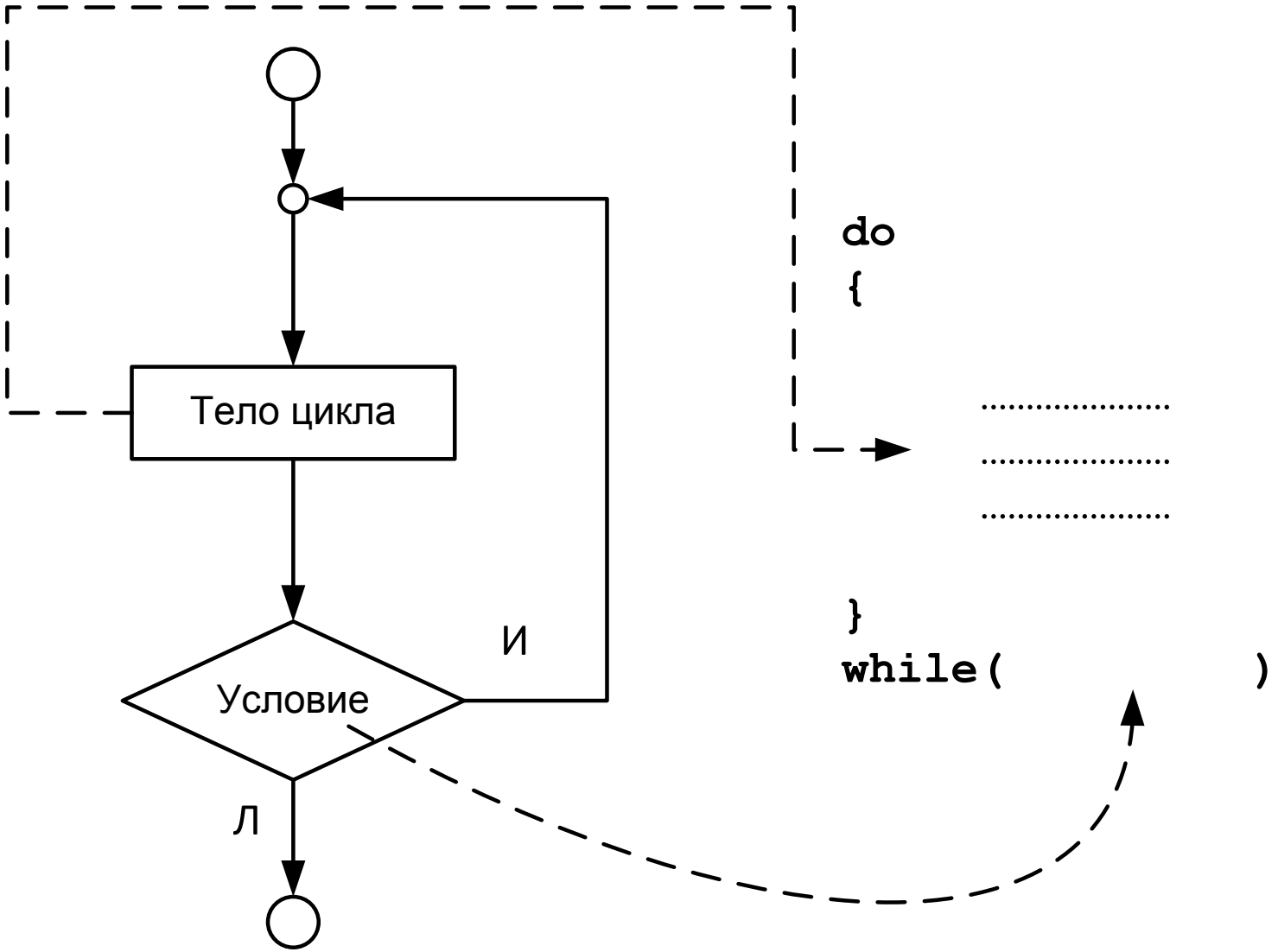

а)

```
if(    )
{
     .....
     .....
}
else
{
     .....
     .....
}
```

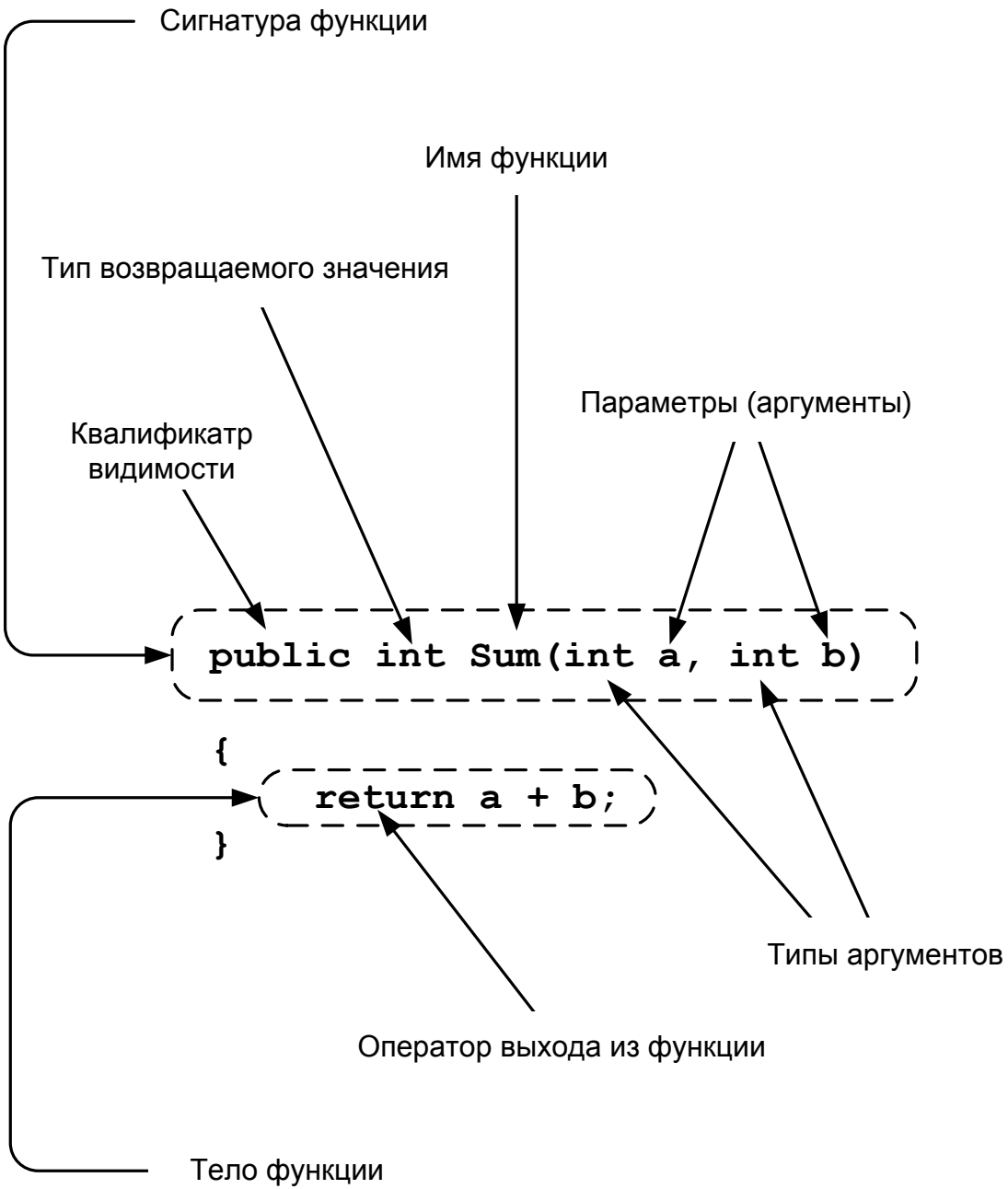
Необязательная часть

б)


Условие
И
Л
Тело цикла
while( )
{
.....
.....
.....
}

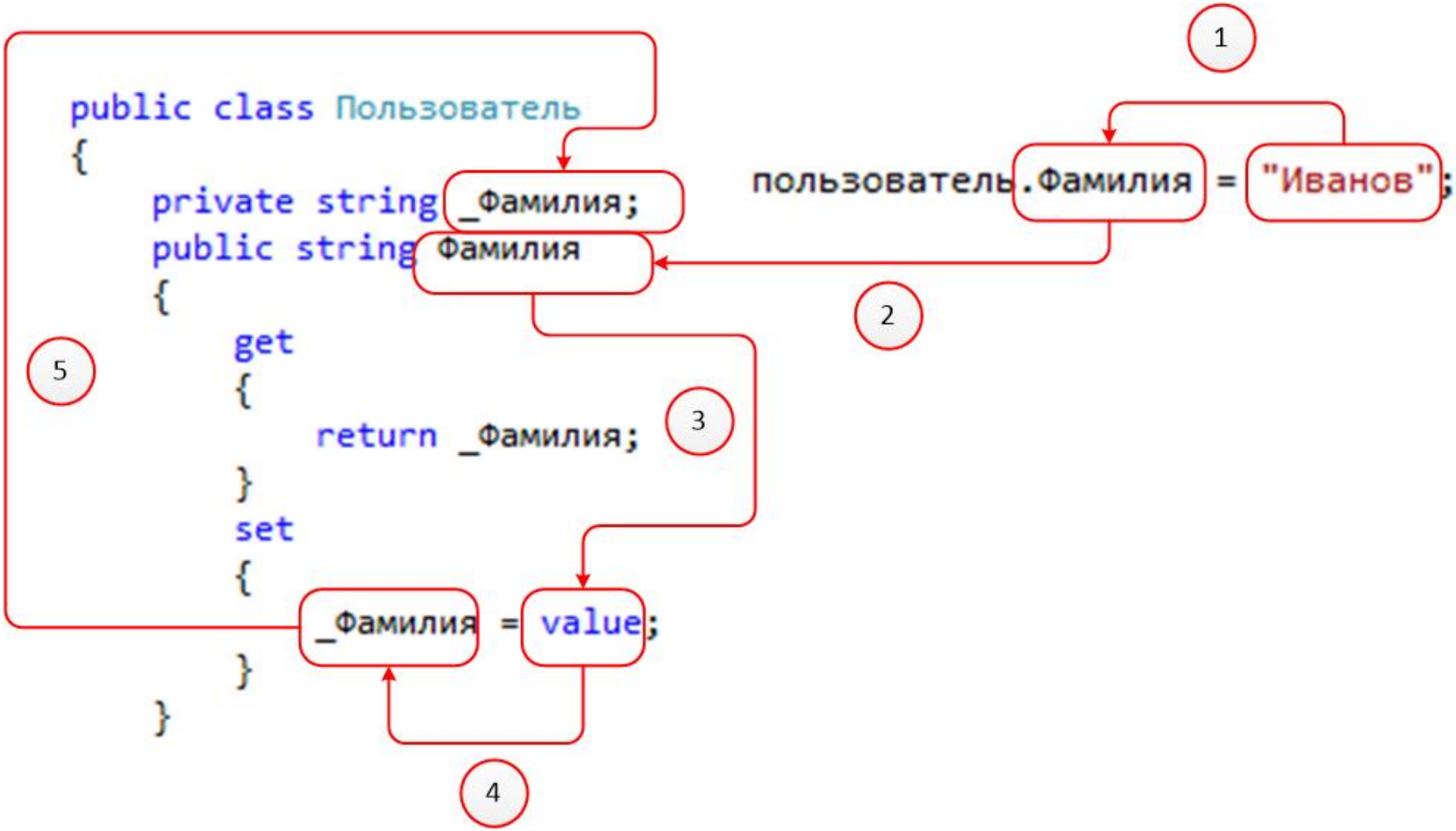

Тело цикла
Условие
И
Л
do
{
.....
.....
.....
}
while(
)

```
public class Пользователь
{
    public string Фамилия;
    public string Имя;
    public string Отчество;

    public string Получить_ФИО()
    {
        return Фамилия + " " + Имя + " " + Отчество;
    }
}
```


Сигнатура функции
Имя функции
Тип возвращаемого значения
Параметры (аргументы)
Квалификатр видимости
public int Sum(int a, int b)
{
return a + b;
}
Типы аргументов
Оператор выхода из функции
Тело функции

```
Пользователь пользователь = new Пользователь();
пользователь.Фамилия = "Иванов";
пользователь.Имя = "Иван";
пользователь.Отчество = "Иванович";

Label1.Text = пользователь.Получить_ФИО();
```

```
public class Пользователь
{
    private string _Фамилия;
    public string Фамилия
    {
        get
        {
            return _Фамилия;
        }
        set
        {
            _Фамилия = value;
        }
    }
```

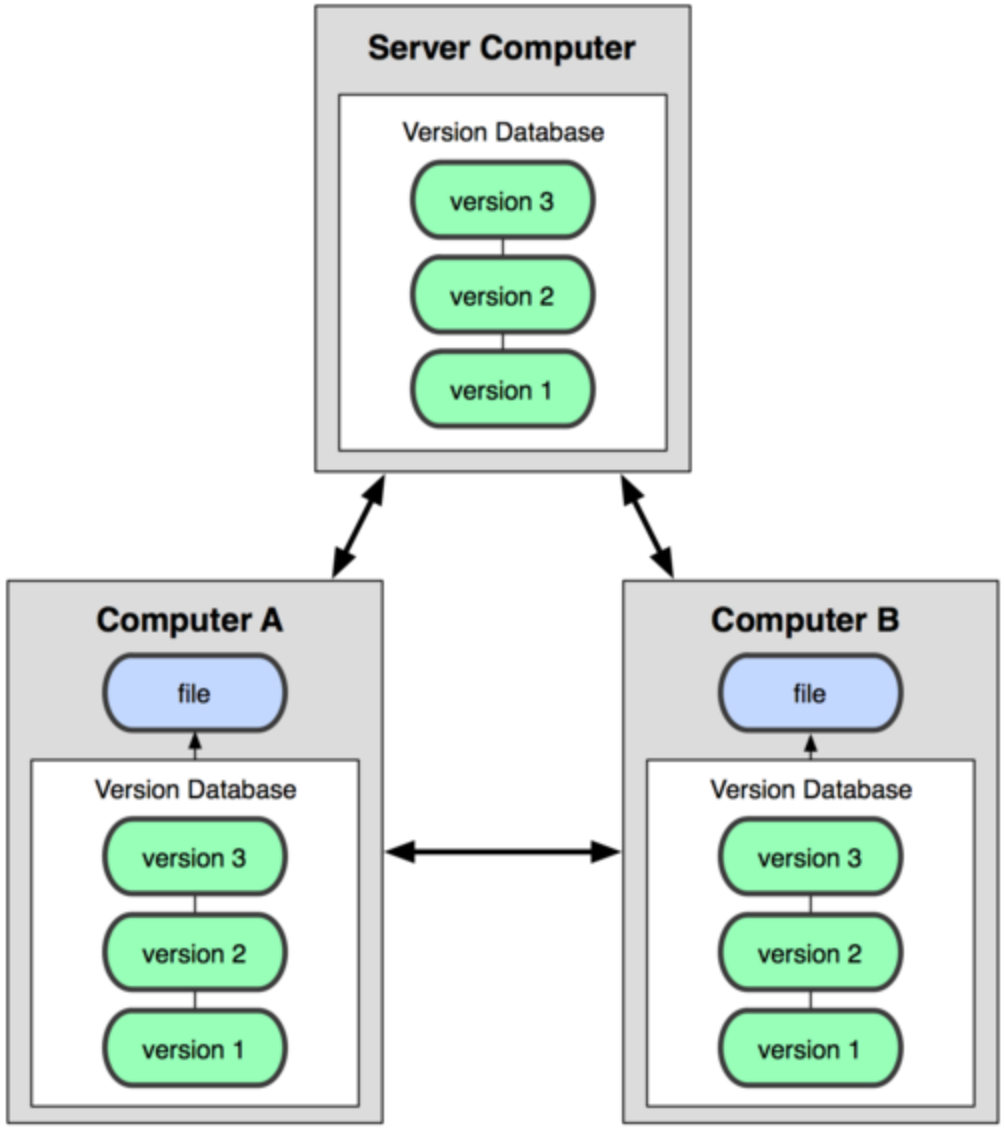

public class Пользователь
{
private string _Фамилия;
public string Фамилия
{
get
{
return _Фамилия;
}
set
{
_Фамилия = value;
}
}
пользователь.Фамилия = "Иванов";
1
2
3
4
5

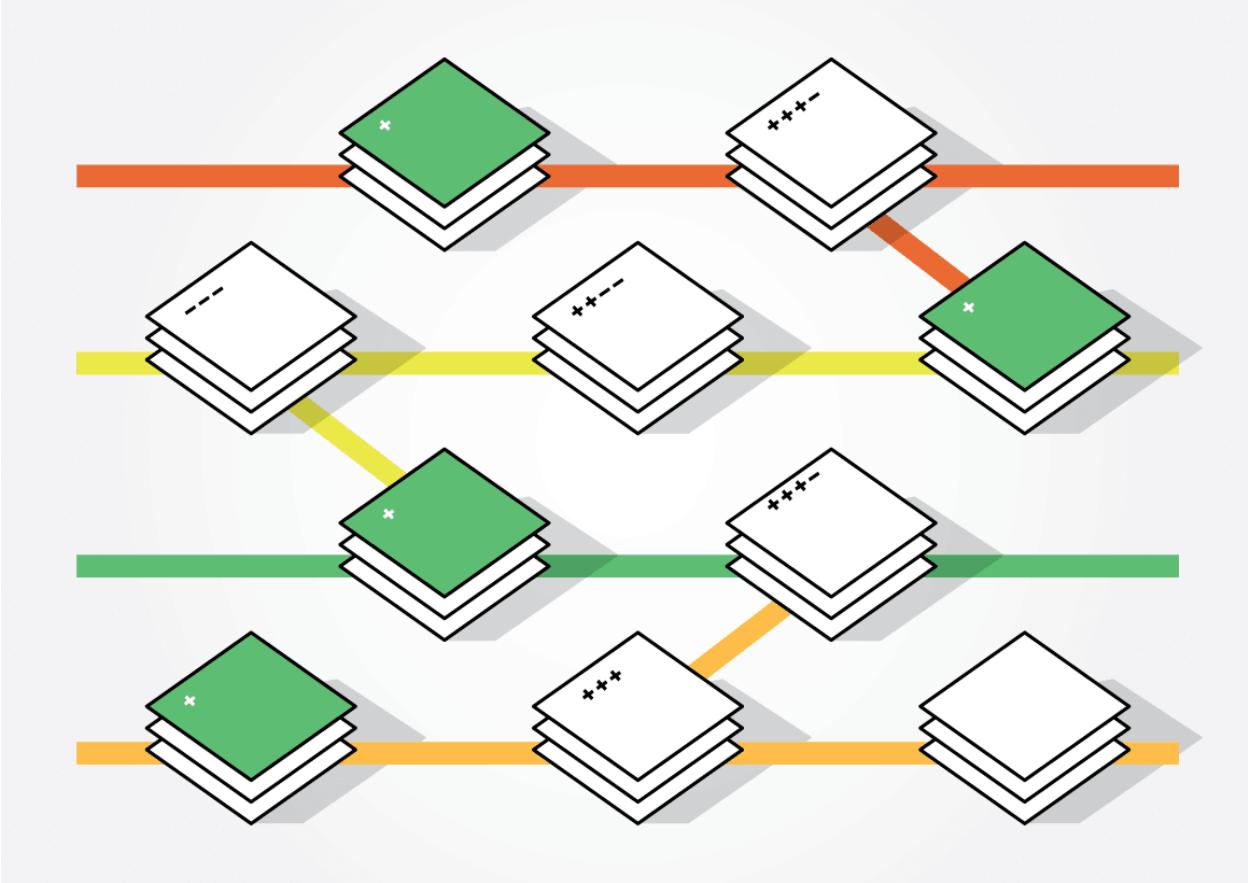
```
public class Пользователь
  {
     private string _Фамилия;
     public string Фамилия
     {
        get
        {
           return _Фамилия;
        }
        set
        {
           _Фамилия = value;
        }
     }

     private string _Имя;
     public string Имя
     {
        get
        {
           return _Имя;
        }
        set
        {
           _Имя = value;
        }
     }
     private string _Отчество;
     public string Отчество
     {
        get
        {
           return _Отчество;
        }
        set
        {
           _Отчество = value;
        }
     }

     public Пользователь()
     {
        _Фамилия = "";
        _Имя = "";
        _Отчество = "";
     }

     public Пользователь(string фамилия, string имя, string отчество)
     {
        _Фамилия = фамилия;
        _Имя = имя;
        _Отчество = отчество;
     }

     public string Получить_ФИО()
     {
        return _Фамилия + " " + _Имя + " " + _Отчество;
     }
  }
```

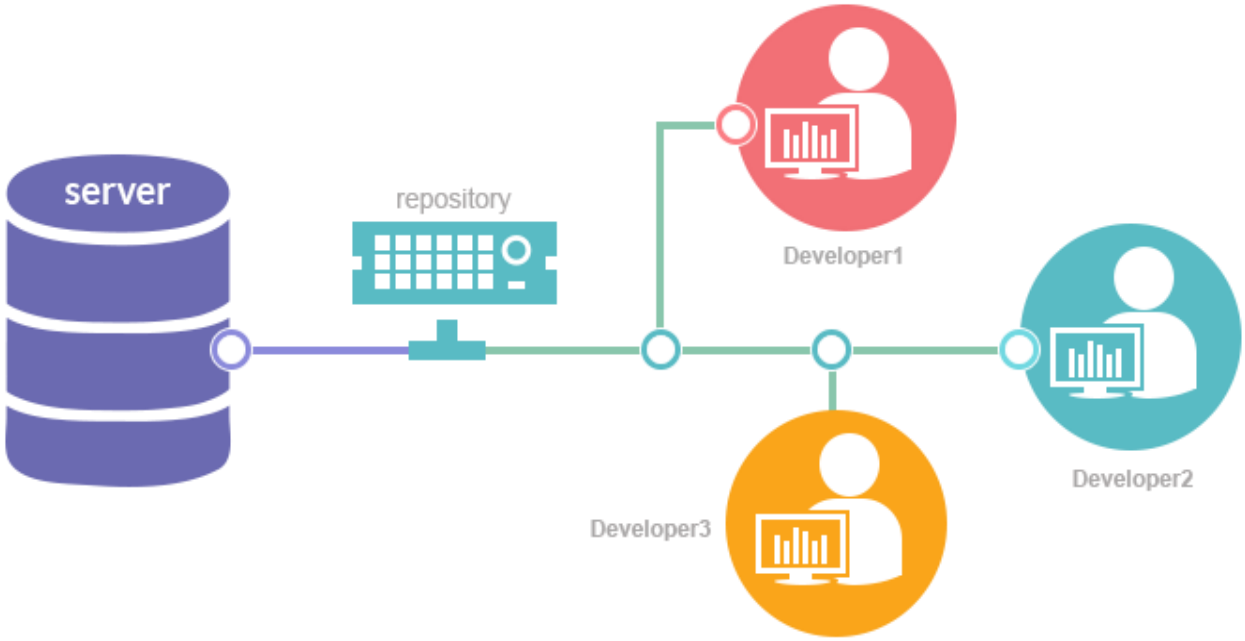
```
public Пользователь()
{
    _Фамилия = "";
    _Имя = "";
    _Отчество = "";
}

public Пользователь(string фамилия, string имя, string отчество)
{
    _Фамилия = фамилия;
    _Имя = имя;
    _Отчество = отчество;
}
```

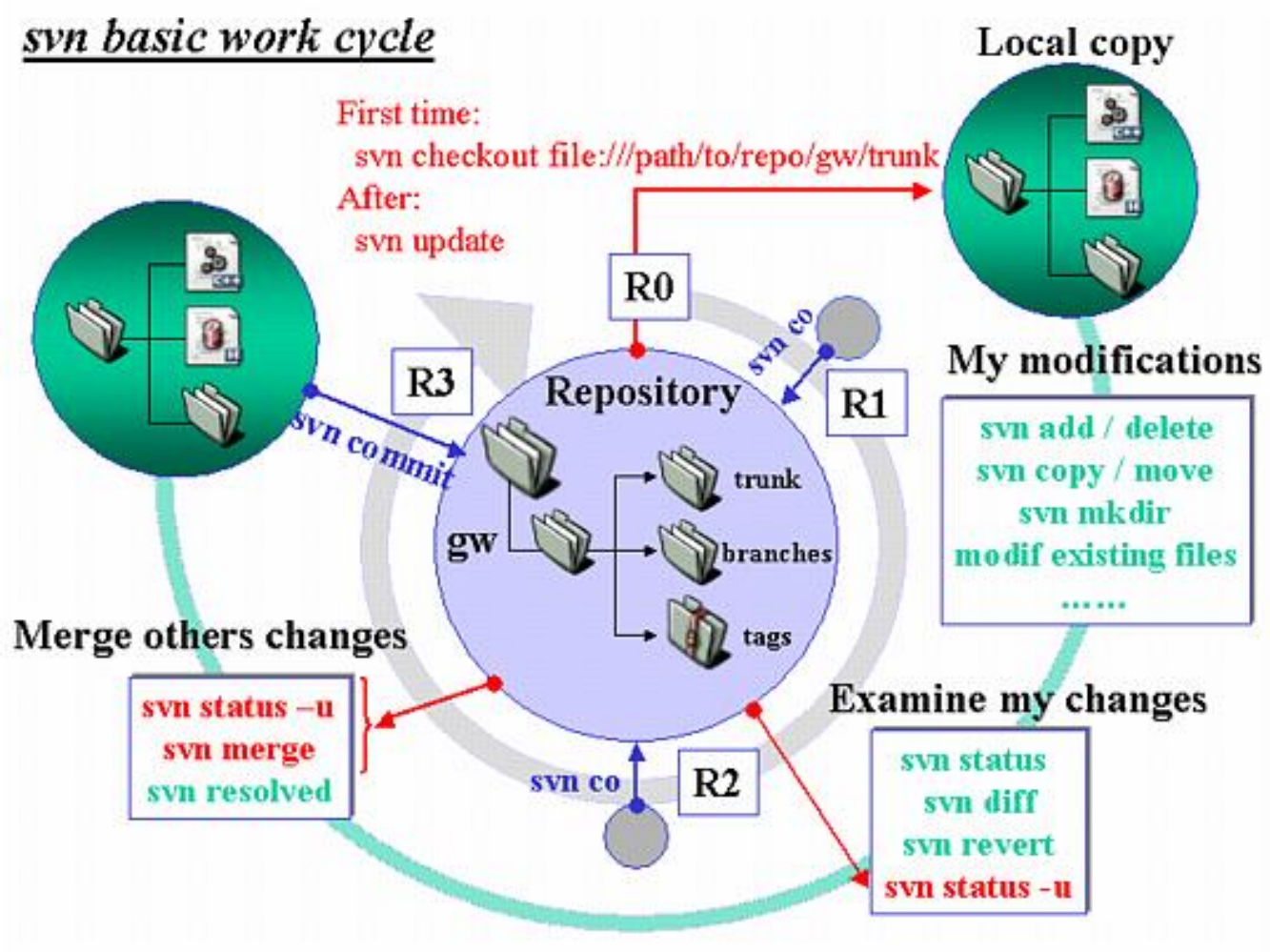
```
Пользователь пользователь = new Пользователь();
пользователь.Фамилия = "Иванов";
пользователь.Имя = "Иван";
пользователь.Отчество = "Иванович";
Пользователь пользователь2 = new Пользователь("Петров", "Петр", "Петрович");
Label1.Text = пользователь.Получить_ФИО();
Label2.Text = пользователь2.Получить_ФИО();
Label3.Text = пользователь2.Имя;
пользователь.Имя = "Александ";
Label4.Text = пользователь.Получить_ФИО();
Label5.Text = пользователь2.Получить_ФИО();
```

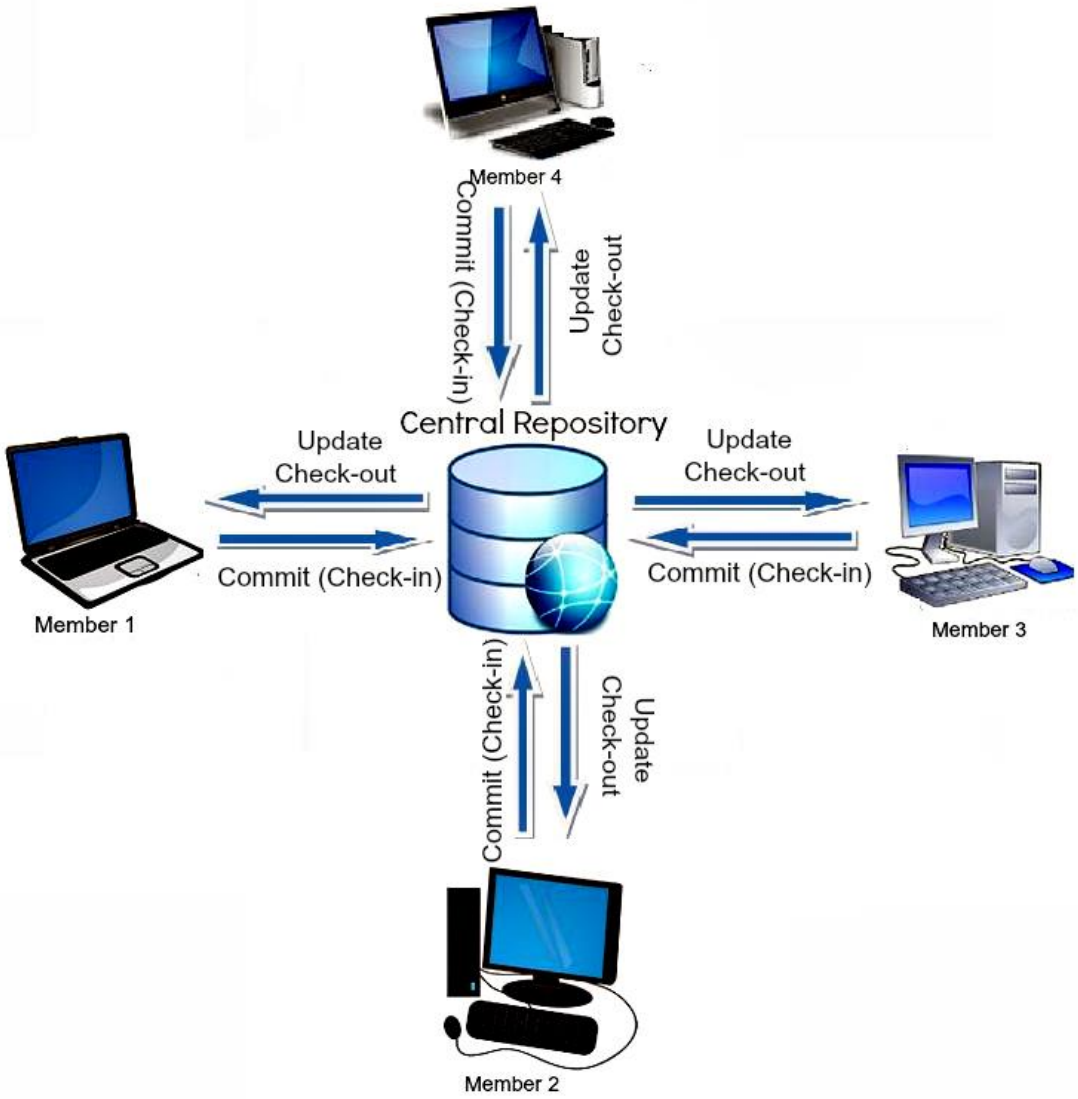
# Системы управления версиями

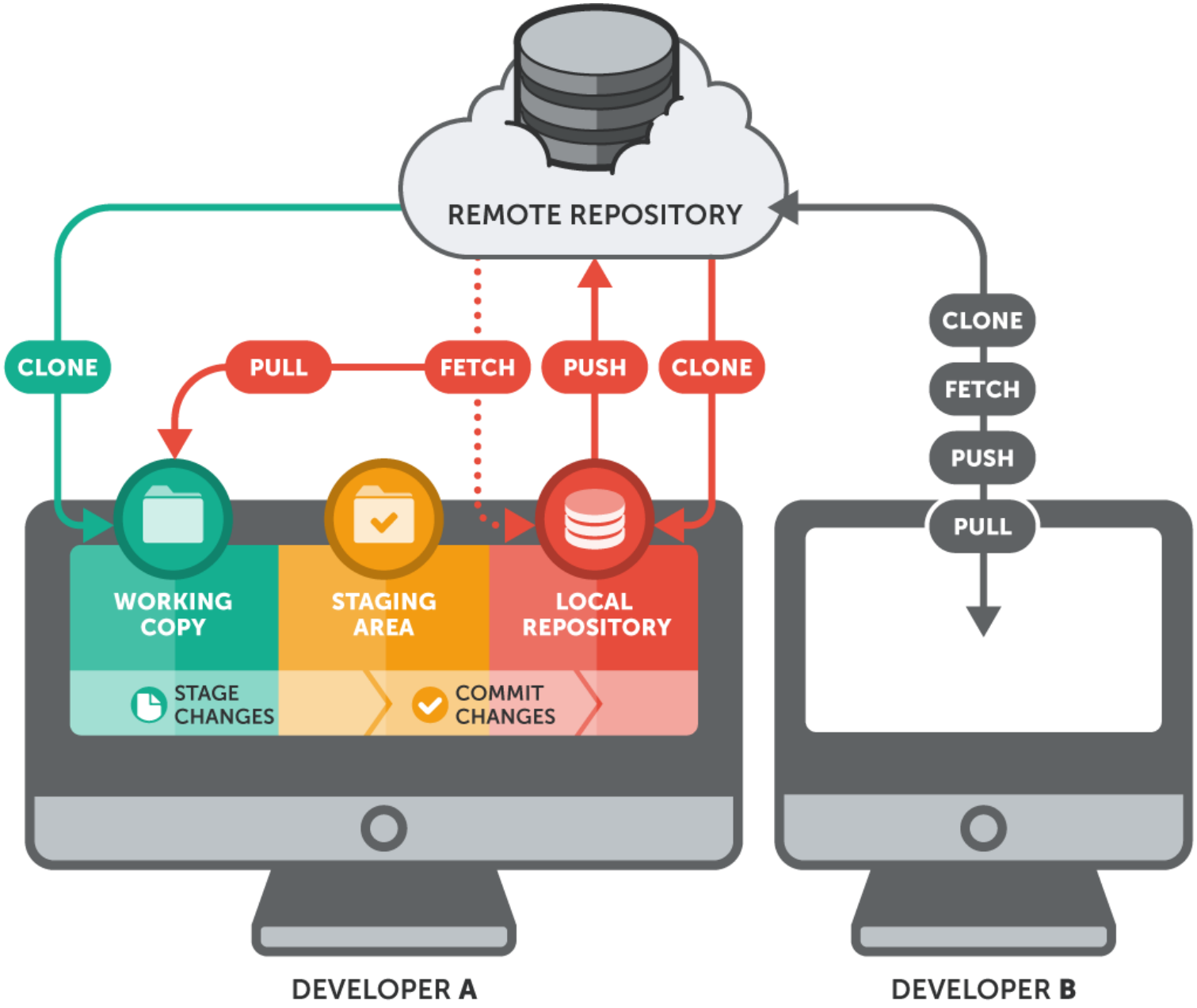

Server Computer
Version Database
version 3
version 2
version 1
Computer A
file
Version Database
version 3
version 2
version 1
Computer B
file
Version Database
version 3
version 2
version 1




server
repository
Developer1
Developer2
Developer3

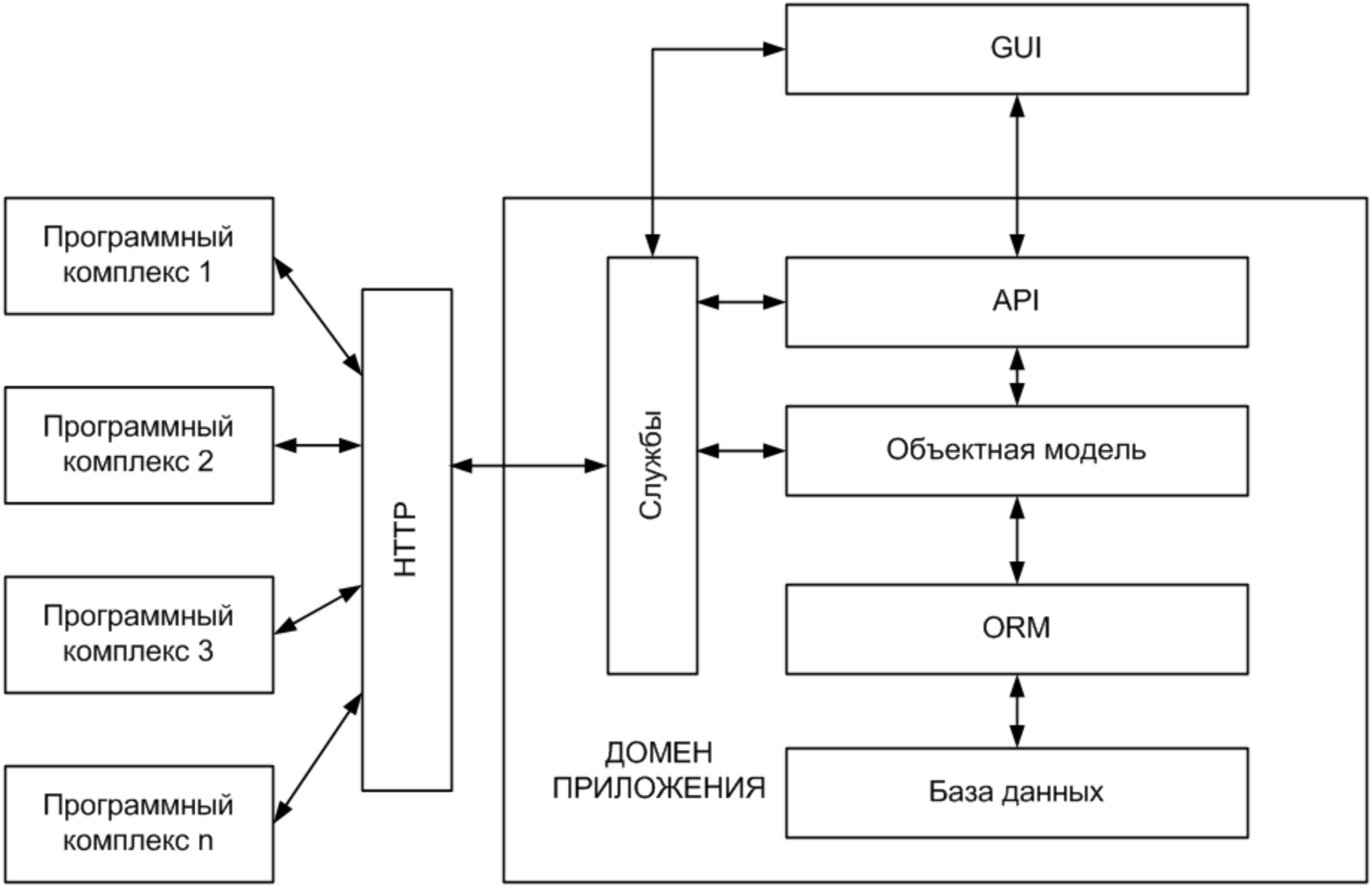

svn basic work cycle
Local copy
First time:
svn checkout file:///path/to/repo/gw/trunk
After:
svn update
R0
R1
R2
R3
svn co
svn co
svn commit
Repository
gw
trunk
branches
tags
My modifications
svn add / delete
svn copy / move
svn mkdir
modif existing files
……
Merge others changes
svn status –u
svn merge
svn resolved
Examine my changes
svn status
svn diff
svn revert
svn status -u


Member 4
Commit (Check-in)
Update
Check-out
Central Repository
Update
Check-out
Commit (Check-in)
Update
Check-out
Commit (Check-in)
Member 1
Member 3
Commit (Check-in)
Update
Check-out
Member 2


REMOTE REPOSITORY
CLONE
PULL
FETCH
PUSH
CLONE
CLONE
FETCH
PUSH
PULL
WORKING COPY
STAGING AREA
LOCAL REPOSITORY
STAGE CHANGES
COMMIT CHANGES
DEVELOPER A
DEVELOPER B

# Технология разработки современных веб-приложений

# Технология DDD (Domain-driven design – разработка приложений на основе модели предметной области)


GUI
Программный комплекс 1
Программный комплекс 2
Программный комплекс 3
Программный комплекс n
HTTP
Службы
API
Объектная модель
ORM
База данных
ДОМЕН ПРИЛОЖЕНИЯ

# Классы коллекций

```
List<int> числа = new List<int>();
for (int i = 1; i < 11; i++)
{
    числа.Add(i);
}

Label1.Text = числа.Count.ToString();
числа.RemoveAt(4);
Label2.Text = числа.Count.ToString();
Метка.Text = числа[4].ToString();

ListBox1.DataSource = числа;
ListBox1.DataBind();
```

```
public string ФИО
{
    get
    {
        return _Фамилия + " " + _Имя + " " + _Отчество;
    }
}
```

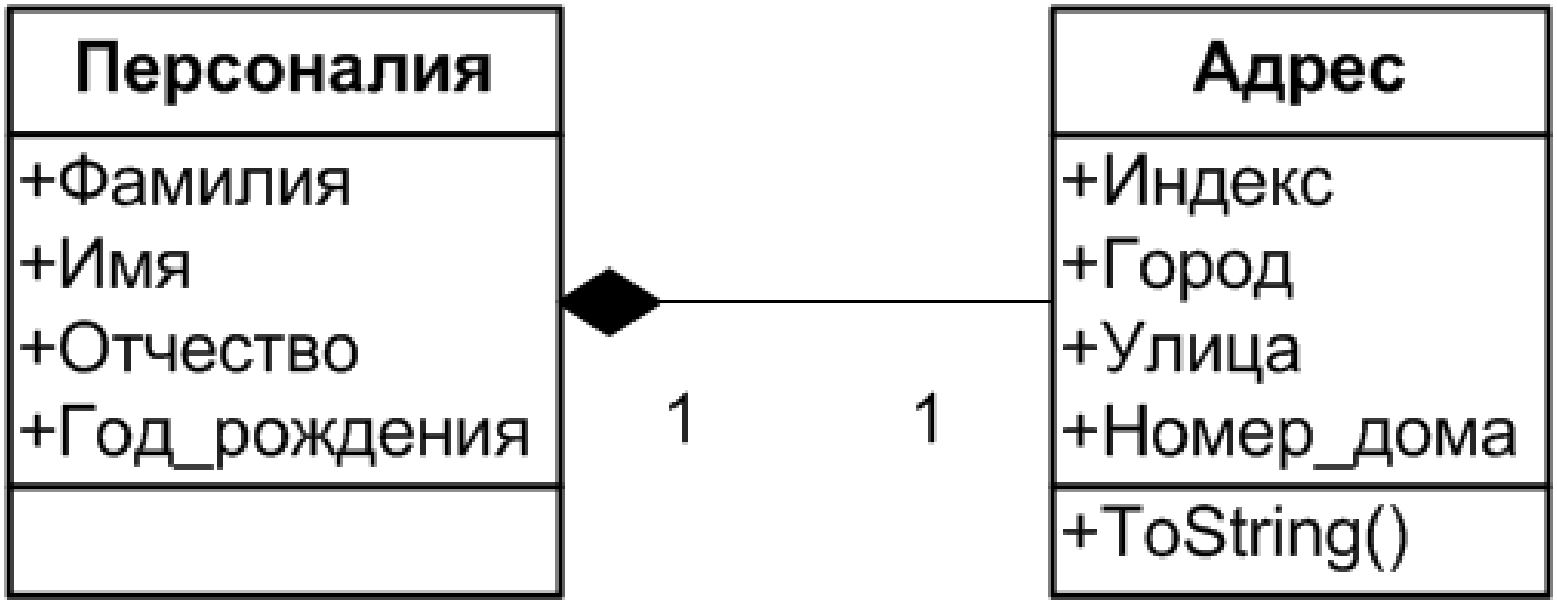
```
List<Персоналия> персоналии = new List<Персоналия>();

персоналии.Add(new Персоналия("Иванов", "Иван", "Иванович"));
персоналии.Add(new Персоналия("Петров", "Петр", "Петрович"));
персоналии.Add(new Персоналия("Степанов", "Степан", "Степанович"));

GridView1.DataSource = персоналии;
GridView1.DataBind();
```

```
if (!Page.IsPostBack)
{
    if (Session["Persons"] == null)
    {
        Session["Persons"] = new List<Персоналия>();
    }
    else
    {
        GridView1.DataSource = (List<Персоналия>)Session["Persons"];
        Page.DataBind();
    }
}
```

```
protected void AddPerson_Click(object sender, EventArgs e)
{
    Персоналия персоналия = new Персоналия();
    персоналия.Фамилия = Фамилия.Text.Trim();
    персоналия.Имя = Имя.Text.Trim();
    персоналия.Отчество = Отчество.Text.Trim();

    List<Персоналия> persons = (List<Персоналия>)Session["Persons"];
    persons.Add(персоналия);
    Page.Response.Redirect("/WebForm2.aspx");
}
```

```
protected void RemoveAll_Click(object sender, EventArgs e)
{
    Session["Persons"] = null;
    Page.Response.Redirect("/WebForm2.aspx");
}
```

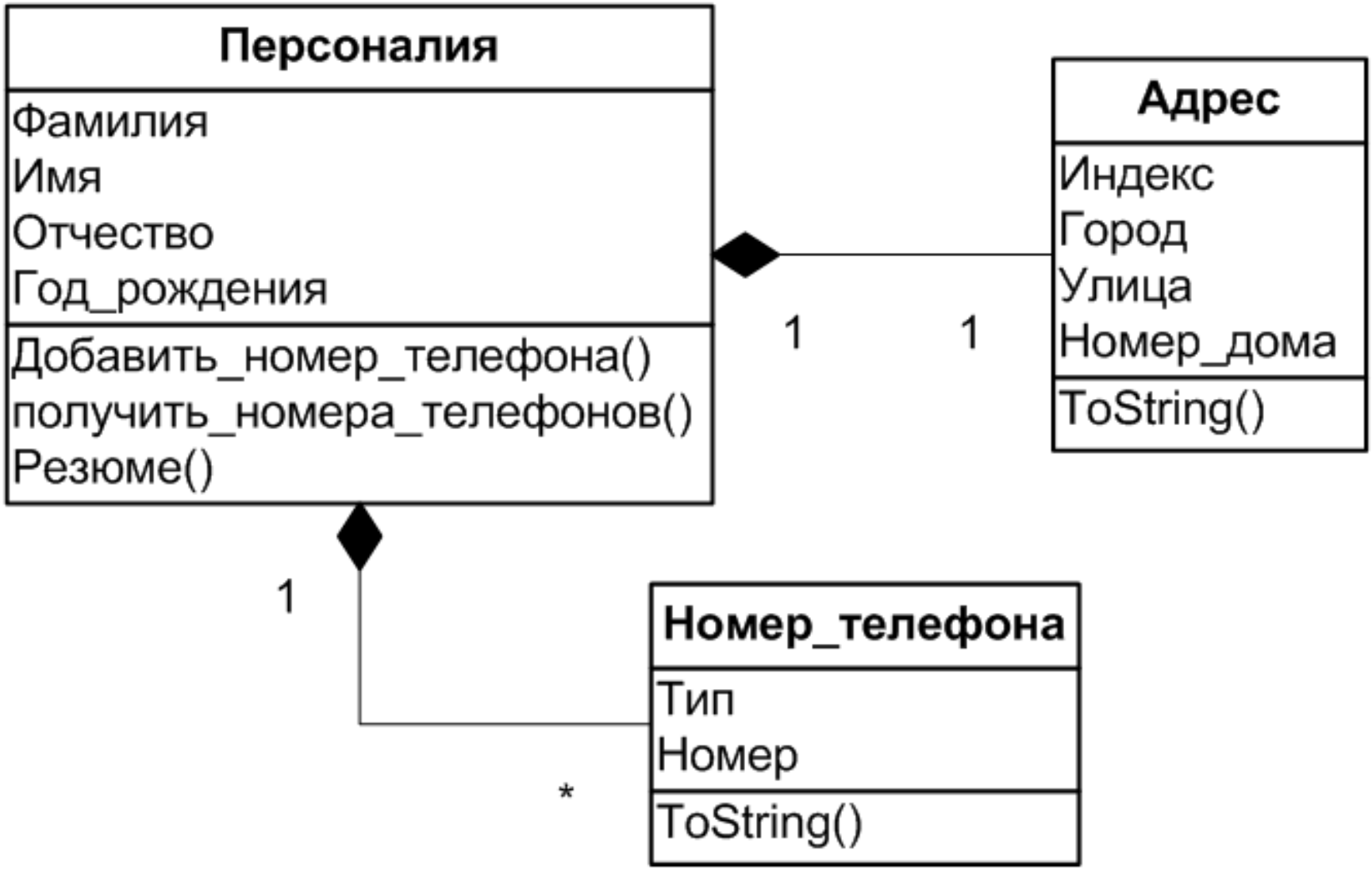
# Ассоциации


Персоналия
+Фамилия
+Имя
+Отчество
+Год_рождения
1
1
Адрес
+Индекс
+Город
+Улица
+Номер_дома
+ToString()

```
public class Адрес
{
    public string Индекс { get; set; }
    public string Город { get; set; }
    public string Улица { get; set; }
    public int Номер_дома { get; set; }
    public override string ToString()
    {
        return Индекс + ", " + Город + ", " + Улица + ", "
            + Номер_дома.ToString();
    }
}
```

```
private Адрес _Адрес = new Адрес();
public Адрес Домашний_адрес
{
    get
    {
        return _Адрес;
    }
}

public string Резюме
{
    get
    {
        return ФИО + ". " + Домашний_адрес.ToString();
    }
}
```

```
Персоналия персоналия = new Персоналия("Иванов", "Иван", "Иванович");
персоналия.Домашний_адрес.Индекс = "198500";
персоналия.Домашний_адрес.Город = "Санкт-Петербург";
персоналия.Домашний_адрес.Улица = "Невский проспект";
персоналия.Домашний_адрес.Номер_дома = 56;

Информация_о_пользователе.Text = персоналия.Резюме;
```

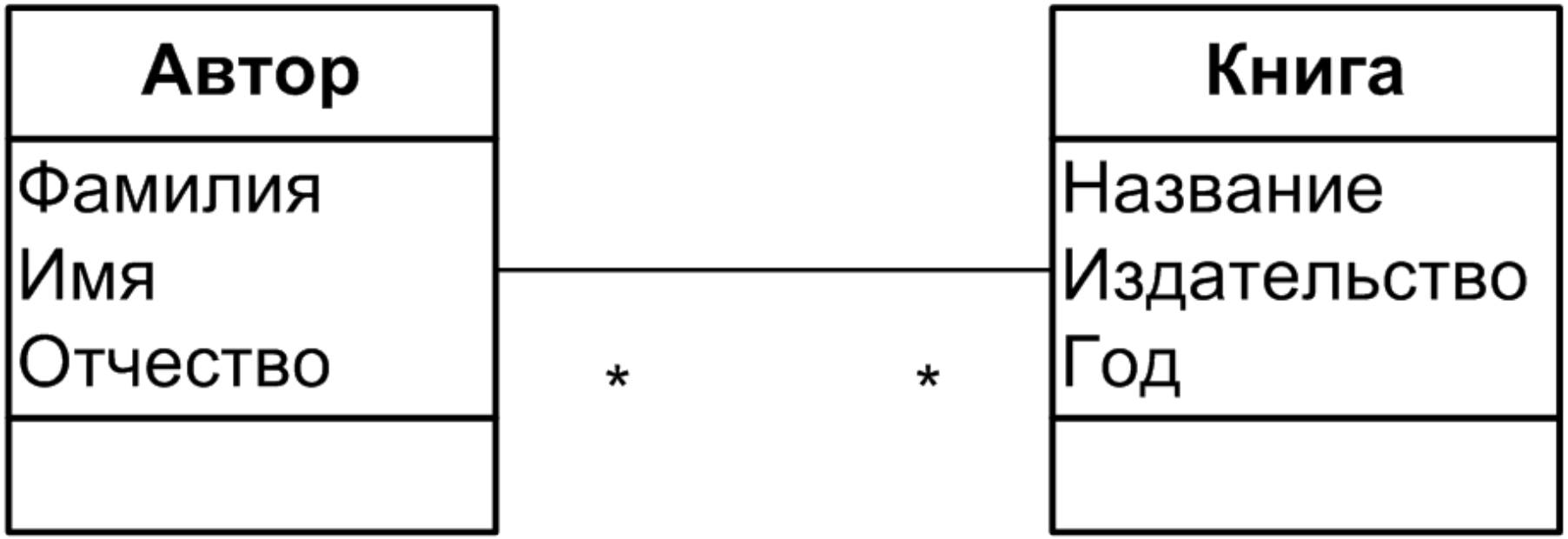

Персоналия
Фамилия
Имя
Отчество
Год_рождения
Добавить_номер_телефона()
получить_номера_телефонов()
Резюме()
Адрес
Индекс
Город
Улица
Номер_дома
ToString()
1
1
1
*
Номер_телефона
Тип
Номер
ToString()

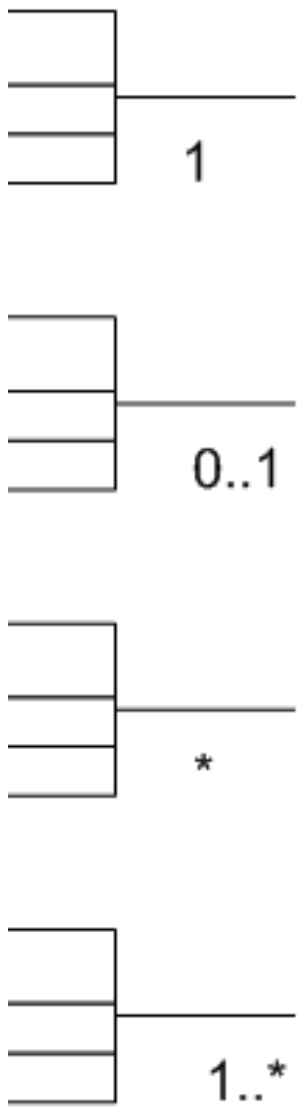
```
public class Номер_телефона
{
    public string Тип { get; set; }
    public string Номер { get; set; }

    public Номер_телефона(string тип, string номер)
    {
        Тип = тип;
        Номер = номер;
    }

    public override string ToString()
    {
        return Номер + " (" + Тип + ")";
    }
}
```

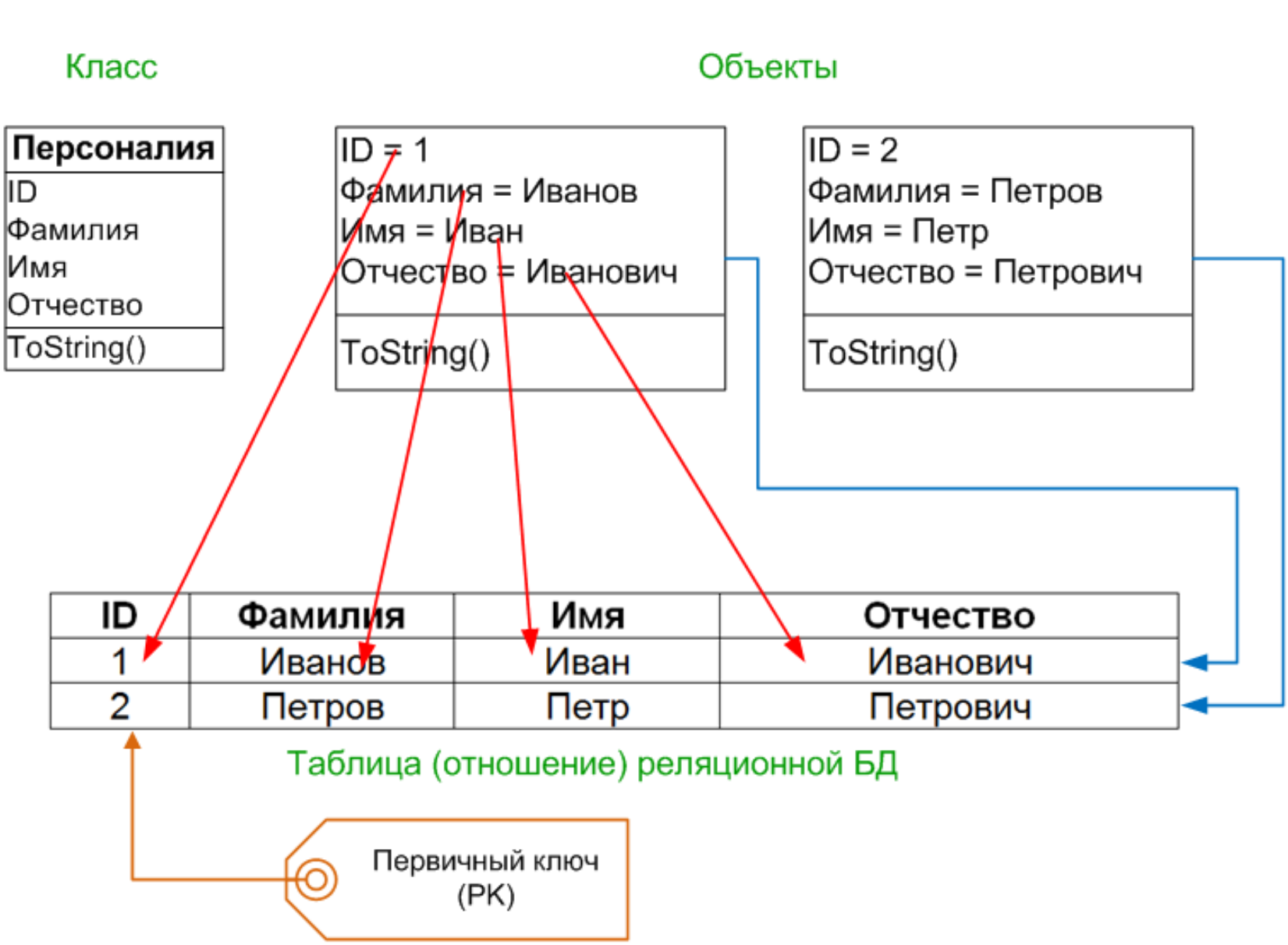
```
private List<Номер_телефона> _Номера_телефонов = new List<Номер_телефона>();
public List<Номер_телефона> Номера_телефонов
{
    get
    {
        return _Номера_телефонов;
    }
}

public void Добавить_номер_телефона(Номер_телефона номер)
{
    _Номера_телефонов.Add(номер);
}
```

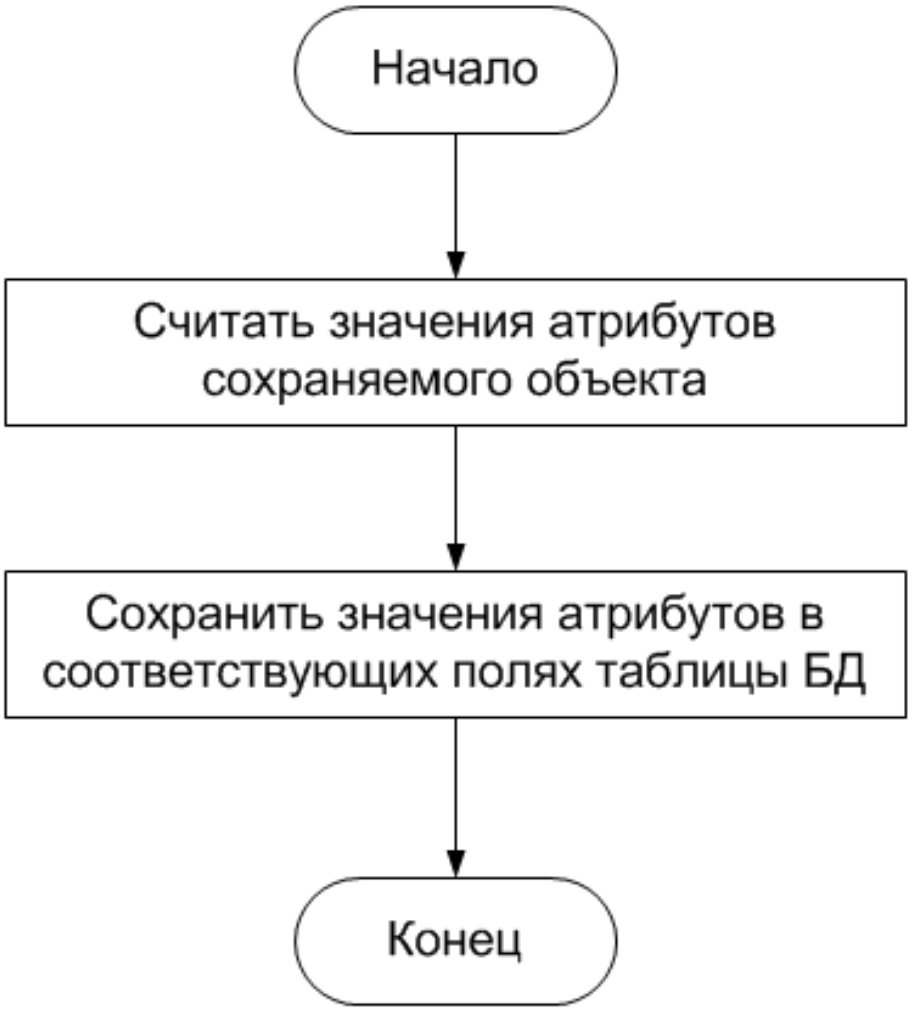
```
private string получить_номера_телефонов()
{
    string номера = "";
    foreach (Номер_телефона номер_телефона in _Номера_телефонов)
    {
        номера = номера + ", " + номер_телефона.ToString();
    }

    return номера.Remove(0, 2);
}
```

```
public string Резюме
{
    get
    {
        return ФИО + ". " + Домашний_адрес.ToString()
            + ". Телефоны: " + получить_номера_телефонов();
    }
}
```

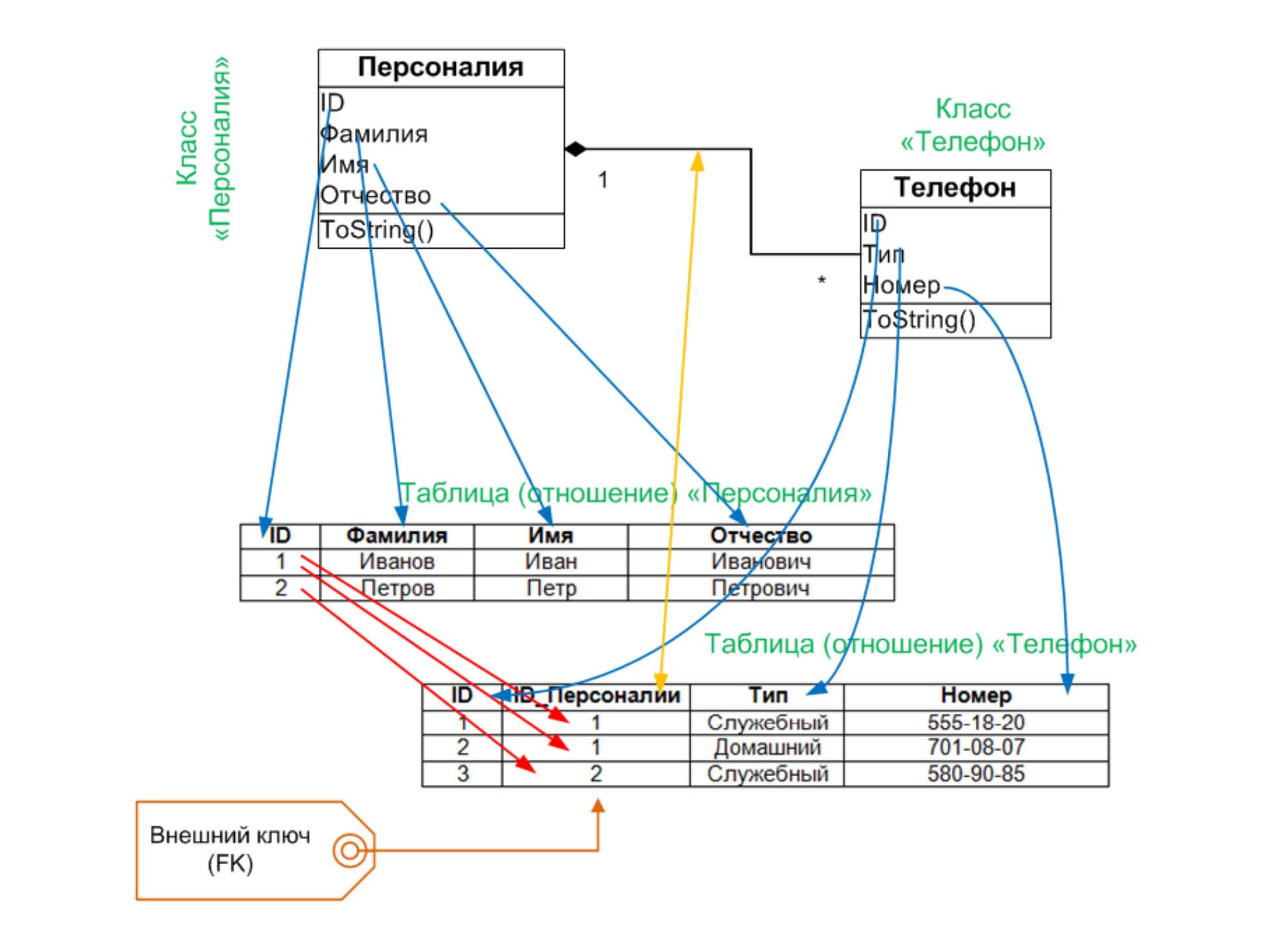

Автор
Фамилия
Имя
Отчество
Книга
Название
Издательство
Год
*
*

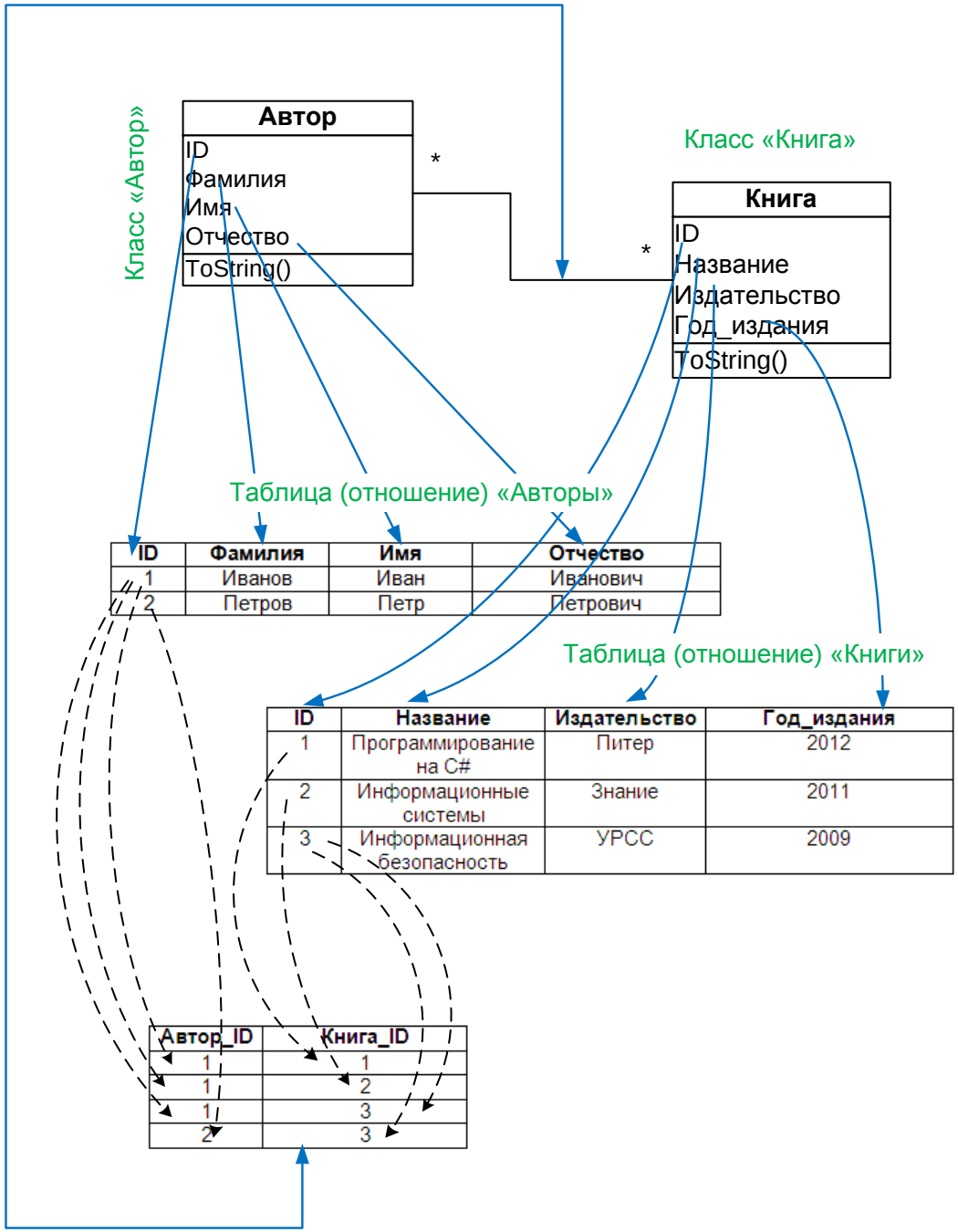

1
0..1
*
1..*


Класс
Объекты
Персоналия
ID
Фамилия
Имя
Отчество
ToString()
ID = 1
Фамилия = Иванов
Имя = Иван
Отчество = Иванович
ToString()
ID = 2
Фамилия = Петров
Имя = Петр
Отчество = Петрович
ToString()
ID
Фамилия
Имя
Отчество
1
Иванов
Иван
Иванович
2
Петров
Петр
Петрович
Таблица (отношение) реляционной БД
Первичный ключ
(PK)


Начало
Считать значения атрибутов сохраняемого объекта
Сохранить значения атрибутов в соответствующих полях таблицы БД
Конец


Начало
Создать объект
Извлечь запись БД по значению идентификатора восстанавливаемого объекта, в которой хранятся значения атрибутов
Считать значения полей, в которых хранятся значения атрибутов восстанавливаемого объекта
Присвоить атрибутам созданного объекта считанные значения
Конец


Класс «Персоналия»
Персоналия
ID
Фамилия
Имя
Отчество
ToString()
1
*
Класс «Телефон»
Телефон
ID
Тип
Номер
ToString()
Таблица (отношение) «Персоналия»
ID | Фамилия | Имя | Отчество
1 | Иванов | Иван | Иванович
2 | Петров | Петр | Петрович
Таблица (отношение) «Телефон»
ID | ID_Персоналии | Тип | Номер
1 | 1 | Служебный | 555-18-20
2 | 1 | Домашний | 701-08-07
3 | 2 | Служебный | 580-90-85
Внешний ключ (FK)

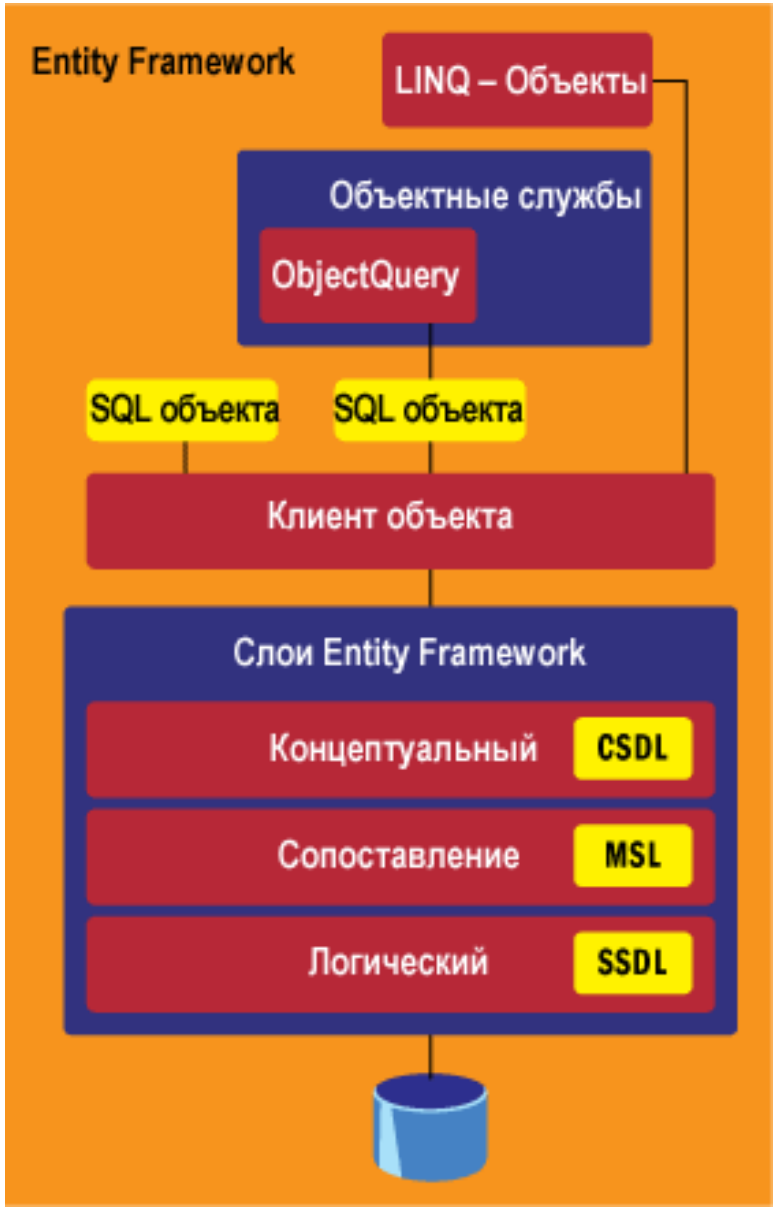

Класс «Автор»
Автор
ID
Фамилия
Имя
Отчество
ToString()
*
*
Класс «Книга»
Книга
ID
Название
Издательство
Год_издания
ToString()
Таблица (отношение) «Авторы»
ID Фамилия Имя Отчество
1 Иванов Иван Иванович
2 Петров Петр Петрович
Таблица (отношение) «Книги»
ID Название Издательство Год_издания
1 Программирование на C# Питер 2012
2 Информационные системы Знание 2011
3 Информационная безопасность УРСС 2009
Автор_ID Книга_ID
1 1
1 2
1 3
2 3

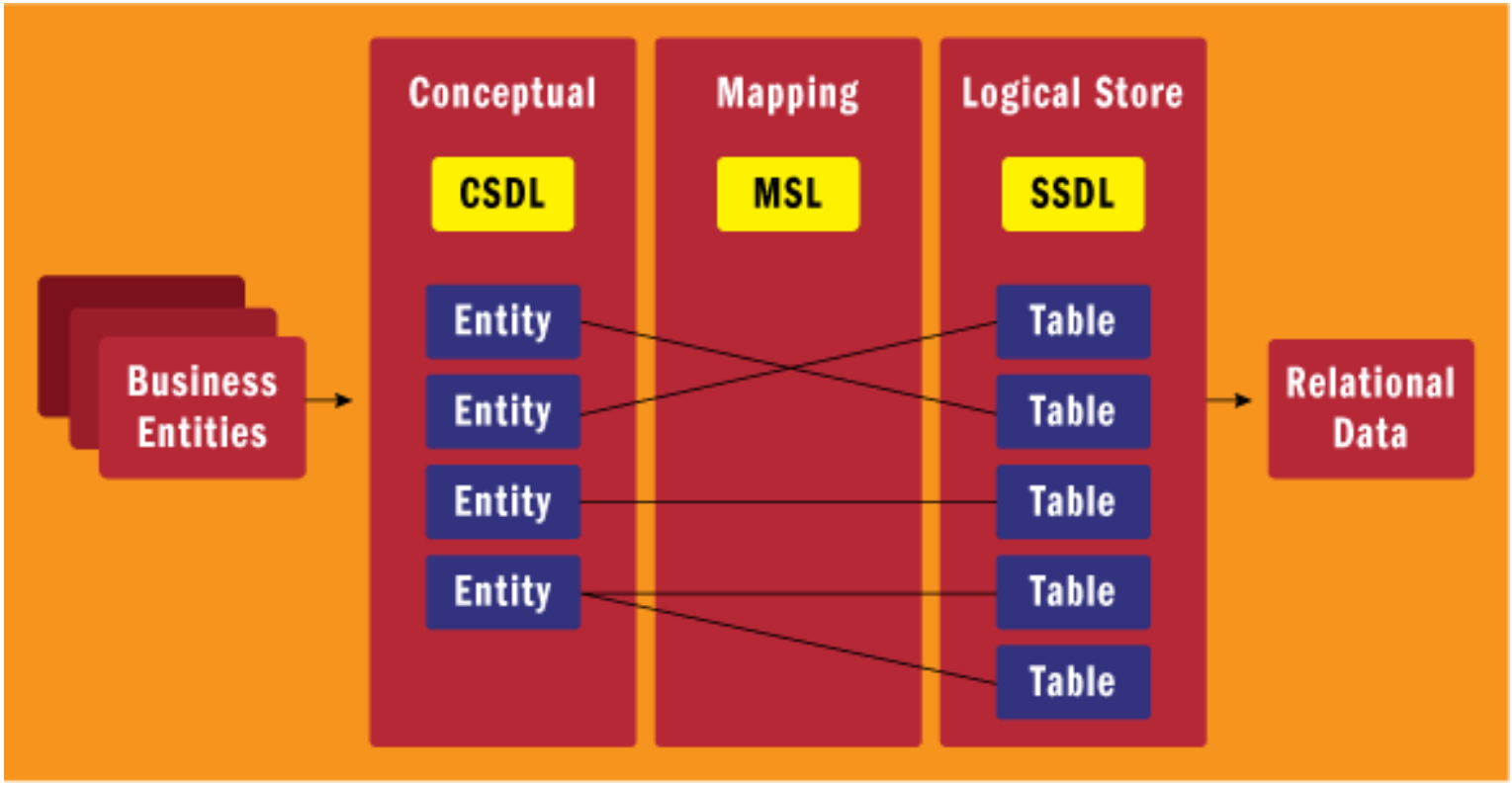
# Entity Framework

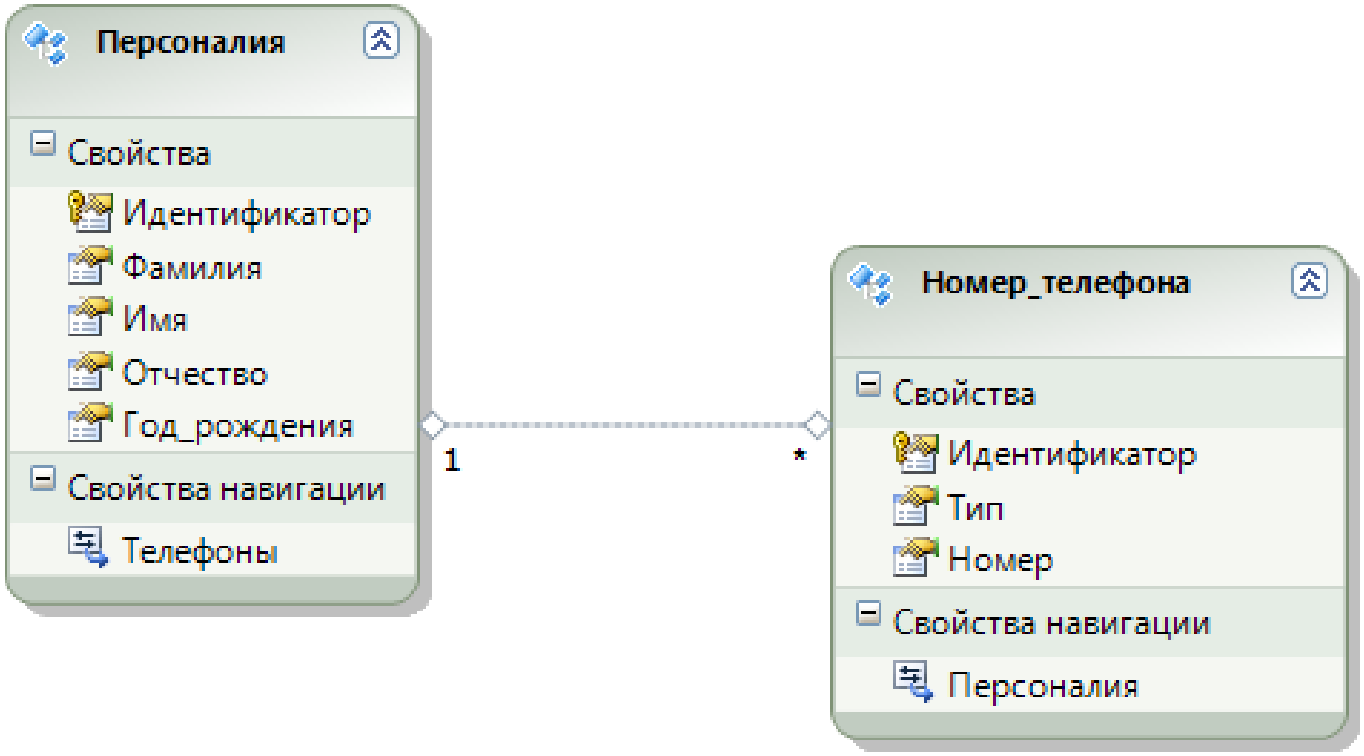
1. Создать базу данных
    a. Запустить Sql Server Management Studio
    b. Выполнить подключение
    c. Выделить на дереве проектов «Базы данных», нажать правую кнопку мыши и выбрать «создать базу данных»
    d. Ввести имя базы данных (латинницей)
2. Создать новое приложение в Visual Studio или WebDeveloper
3. В проводнике проекта открыть вкладку «Обозреватель баз данных» и подключить вновь созданную базу данных

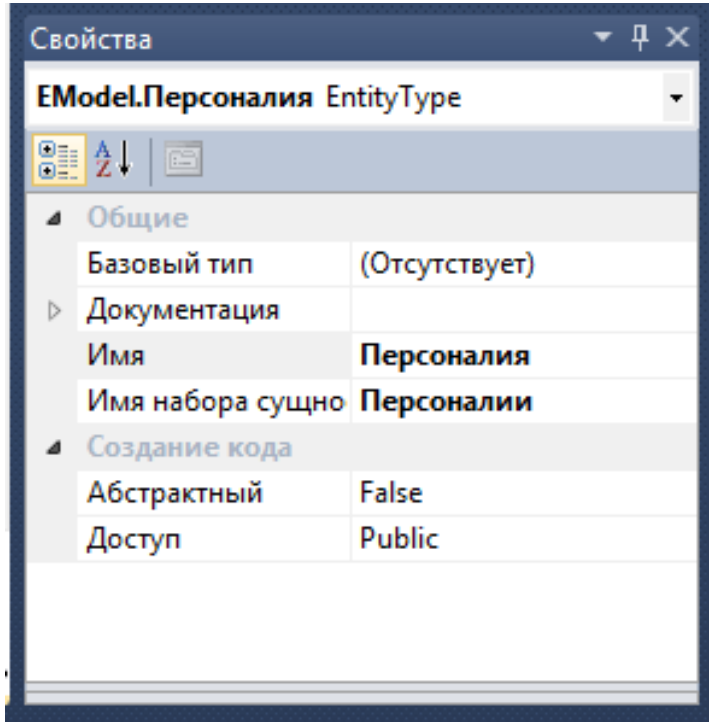
4. Добавить новую Entity Model
    a. В меню «Проект» выбрать «Добавить новый элемент»
    b. Выбрать «Модель ADO.NET EDM»
    c. Назвать ее EModel
    d. В мастере создания сущностных моделей выбрать «Пустая модель»

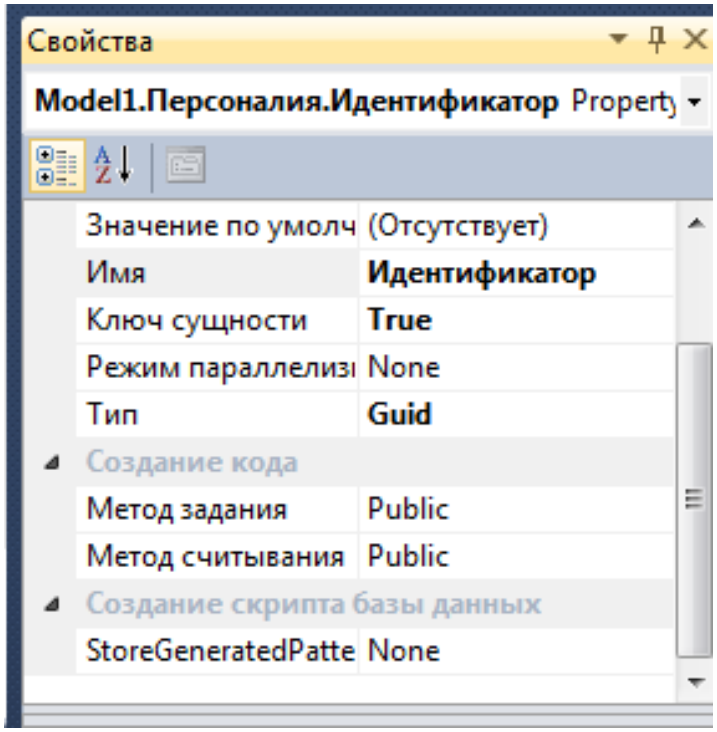
5. В конструкторе моделей создать модель предметной области (Entity Model)
    a. Создать новый сущностный класс (Сущность)
    b. Выбрать имя сущности
    c. Выбрать имя набора сущностей
    d. В качестве типа идентификатора использовать Guid и не забывать устанавливать поле Store Generated Pattern в None.
    e. Создать другие сущностные классы
    f. Создать ассоциации и указать их кратность


Entity Framework
LINQ – Объекты
Объектные службы
ObjectQuery
SQL объекта
SQL объекта
Клиент объекта
Слои Entity Framework
Концептуальный
CSDL
Сопоставление
MSL
Логический
SSDL


Conceptual
Mapping
Logical Store
CSDL
MSL
SSDL
Business Entities
Entity
Entity
Entity
Entity
Table
Table
Table
Table
Table
Relational Data


Персоналия
Свойства
Идентификатор
Фамилия
Имя
Отчество
Год_рождения
Свойства навигации
Телефоны
1
*
Номер_телефона
Свойства
Идентификатор
Тип
Номер
Свойства навигации
Персоналия


Свойства
EModel.Персоналия EntityType
Общие
Базовый тип
(Отсутствует)
Документация
Имя
Персоналия
Имя набора сущно
Персоналии
Создание кода
Абстрактный
False
Доступ
Public


Свойства
Model1.Персоналия.Идентификатор Property
Значение по умолч (Отсутствует)
Имя Идентификатор
Ключ сущности True
Режим параллелизи None
Тип Guid
Создание кода
Метод задания Public
Метод считывания Public
Создание скрипта базы данных
StoreGeneratedPatte None

# LINQ - Language Integrated Query (интегрированный язык запросов)

```
protected void GetPersons_Click(object sender, EventArgs e)
{
    EModelContainer model = new EModelContainer();
    Таблица_персоналий.DataSource = from p in model.Персоналии
                                    select p;
    Таблица_персоналий.DataBind();
}
```

```
Таблица_персоналий.DataSource = from p in model.Персоналии
                                where p.Фамилия == "Иванов"
                                select p;
```

```
= from p in model.Персоналии
  where p.Фамилия == "Иванов" && p.Имя == "Петр"
  select p;
```

```
IQueryable<string> s = from p in model.Персоналии
        where p.Фамилия == "Иванов"
        select p.Имя;
```

```
Таблица_персоналий.DataSource = (from p in model.Персоналии
                                 select p).OrderBy( s => s.Фамилия );
```

```
= (from p in model.Персоналии
   select p).OrderBy( s => s.Фамилия )
   .ThenBy( s => s.Имя)
   .ThenBy( s => s.Отчество);
```

## Назначение репозитория — обеспечение приложению стабильного API методов CRUD и реализация данных методов.

**CRUD** (сокр. от англ. **create**, **read**, **update**, **delete** — «создать, прочесть, обновить, удалить») — акроним, обозначающий четыре базовые функции, используемые при работе с персистентными хранилищами данных.

# Репозиторий
# со статическими методами

```
public static List<LabWork> GetLabWorks()
{
    VLabModelContainer model = new VLabModelContainer();
    return (from lab in model.LabWorkSet
            where lab.State == 0
            select lab).ToList<LabWork>();
}
```

```
LabTable.DataSource = LabWorkRepository.GetLabWorks();
LabTable.DataBind();
```

```
public static LabWork GetLabWork(Guid id, VLabModelContainer model)
{
    return (from lab in model.LabWorkSet
            where lab.Id == id
            select lab).First();
}

public static LabWork GetLabWork(Guid id)
{
    VLabModelContainer model = new VLabModelContainer();
    return GetLabWork(id, model);
}
```

```
public static void LabWorkUpdate(Guid id, string title,
    string goal, string mainInstructions,
    string reportRequirements, int labworkHours,
    string variantsCount, string labworkSkills,
    string labworkLearningObject)
{
    VLabModelContainer model = new VLabModelContainer();
    LabWork lab = GetLabWork(id, model);
    lab.Title = title;
    lab.Goal = goal;
    lab.LastUpdate = DateTime.Now;
    lab.MethodicalInstructions = mainInstructions;
    lab.ContentReportRequirement = reportRequirements;
    lab.WorkingHours = labworkHours;
    lab.VariantCount = variantsCount;
    lab.Skills = labworkSkills;
    lab.LearningObject = labworkLearningObject;
    model.SaveChanges();
}
```

```
public static Lectures DeleteLecture(Guid lectureID)
{
    VLabModelContainer model = new VLabModelContainer();
    Lectures lec = GetLecture(lectureID, model);
    lec.State = -1;
    model.SaveChanges();
    return null;
}

public static List<Lectures> GetDeletedLectures()
{
    VLabModelContainer model = new VLabModelContainer();
    return (from lec in model.LecturesSet
            where lec.State == -1
            select lec).ToList();
}
```

```
public static List<Lectures> GetAllLectures()
{
    VLabModelContainer model = new VLabModelContainer();
    return (from lec in model.LecturesSet
            where lec.State == 0
            select lec).ToList();
}


public static List<Lectures> GetAllLecturesForSubject(Guid subjectID)
{
    VLabModelContainer model = new VLabModelContainer();
    return (from lec in model.LecturesSet
            where lec.State == 0
            && lec.SubjectID == subjectID
            select lec).ToList();
}
```

# Репозиторий
# с агрегируемой моделью (контекстом)

```
public class Repository
{
    protected TestModelContainer model;
    public Repository()
    {
        model = new TestModelContainer();
    }

    public Repository(TestModelContainer model)
    {
        this.model = model;
    }

    public TestModelContainer Model
    {
        get { return model; }
    }
}
```

```
public class TestRepository : Repository
{

    public List<Test> GetAllTests(Guid userID)
    {
        return (from t in model.TestSet
                where t.State != -1
                && t.DeveloperID == userID
                select t).ToList();
    }
```

```
TestRepository repository = new TestRepository();
Guid userID = Security.GetCurrentPersonID();
List<Test> tests = repository.GetAllTests(userID);
```

```
public Test GetTest(Guid id)
{
    return (from t in model.TestSet
            where t.Id == id
            select t).First();
}
```

```
public void Edit(
    Guid testID,
    string name,
    string description,
    string keywords,
    int minPercentToExcellent,
    int minCorrectPercentToGood,
    int minCorrectPercentToPass,
    int attemptCount,
    bool attemptBeforePass,
    int questionOnActivity,
    int timeForActivity,
    bool showDetailedResultToStudent)
{
    Test test = GetTest(testID);
    test.Name = name;
    test.Desctiption = description;
    test.KeyWords = keywords;
    test.MinPercentToExcellent = minPercentToExcellent;
    test.MinPercentToGood = minCorrectPercentToGood;
    test.MinCorrectPercentToPass = minCorrectPercentToPass;
    test.AttempCount = attemptCount;
    test.AttemptBeforePass = attemptBeforePass;
    test.QuestionOnActivity = questionOnActivity;
    test.ShowDetailedResultToStudent = showDetailedResultToStudent;
    model.SaveChanges();
}
```

# Параметризованные (обобщенные) классы

# Bingo Cage (пример)

```
public class BingoCage<T>
{
    private Random _Rnd = new Random();

    private List<T> _Cage = new List<T>();
    public List<T> Cage
    {
        get { return _Cage; }
    }

    public void AddItem(T item)
    {
        Cage.Add(item);
    }

    public void AddItems(List<T> items)
    {
        Cage.AddRange(items);
    }
```

```
public T GetRundomBoll()
{
    int count = Cage.Count;
    int bollIndex;
    if (count > 1)
    {
        bollIndex = _Rnd.Next(0, count);
    }
    else
    {
        bollIndex = 0;
    }
    T boll = Cage[bollIndex];
    Cage.RemoveAt(bollIndex);
    return boll;
}
```

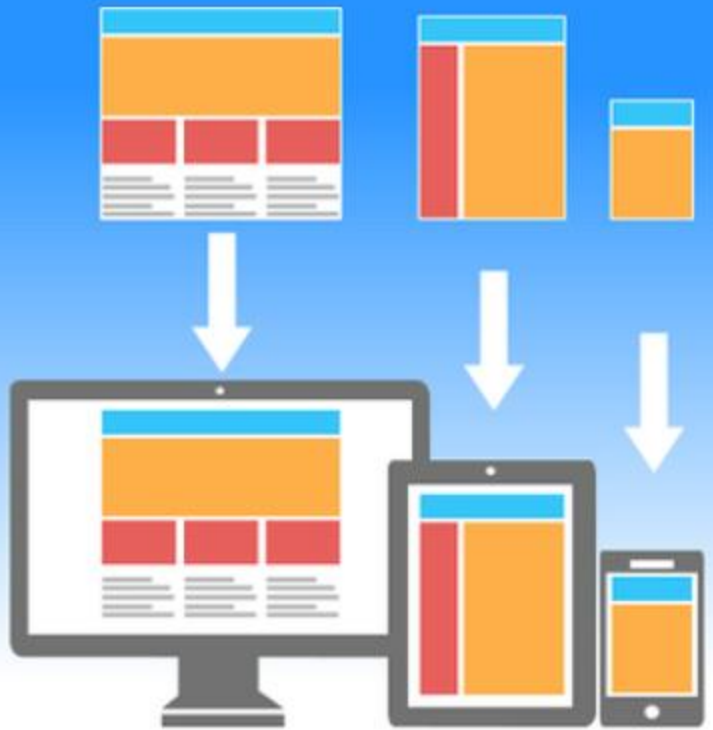
```
    public List<T> GetAllBolls()
    {
        List<T> bolls = new List<T>();
        while(Cage.Count > 0)
        {
            bolls.Add(GetRundomBoll());
        }
        return bolls;
    }

    public List<T> GetBolls(int n)
    {
        List<T> bolls = new List<T>();
        int i = 0;
        while (Cage.Count > 0 && i < n)
        {
            bolls.Add(GetRundomBoll());
            i++;
        }
        return bolls;
    }
}
```

# Технологии адаптивного дизайна пользовательского интерфейса


fotodizart
fotodizart
fotodizart

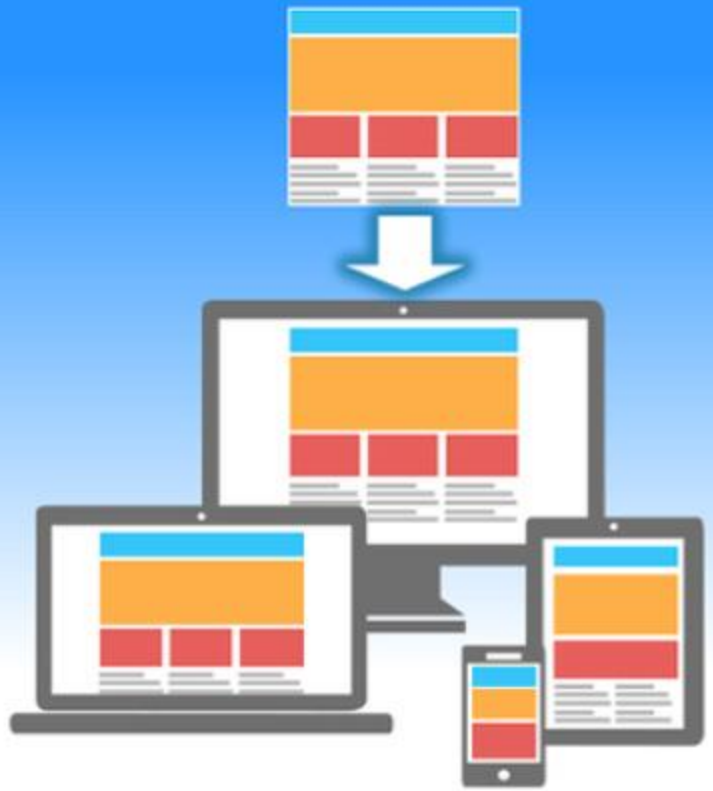
## АДАПТИВНЫЙ

Оптимизированные и уникальные шаблоны для каждого типа устройств

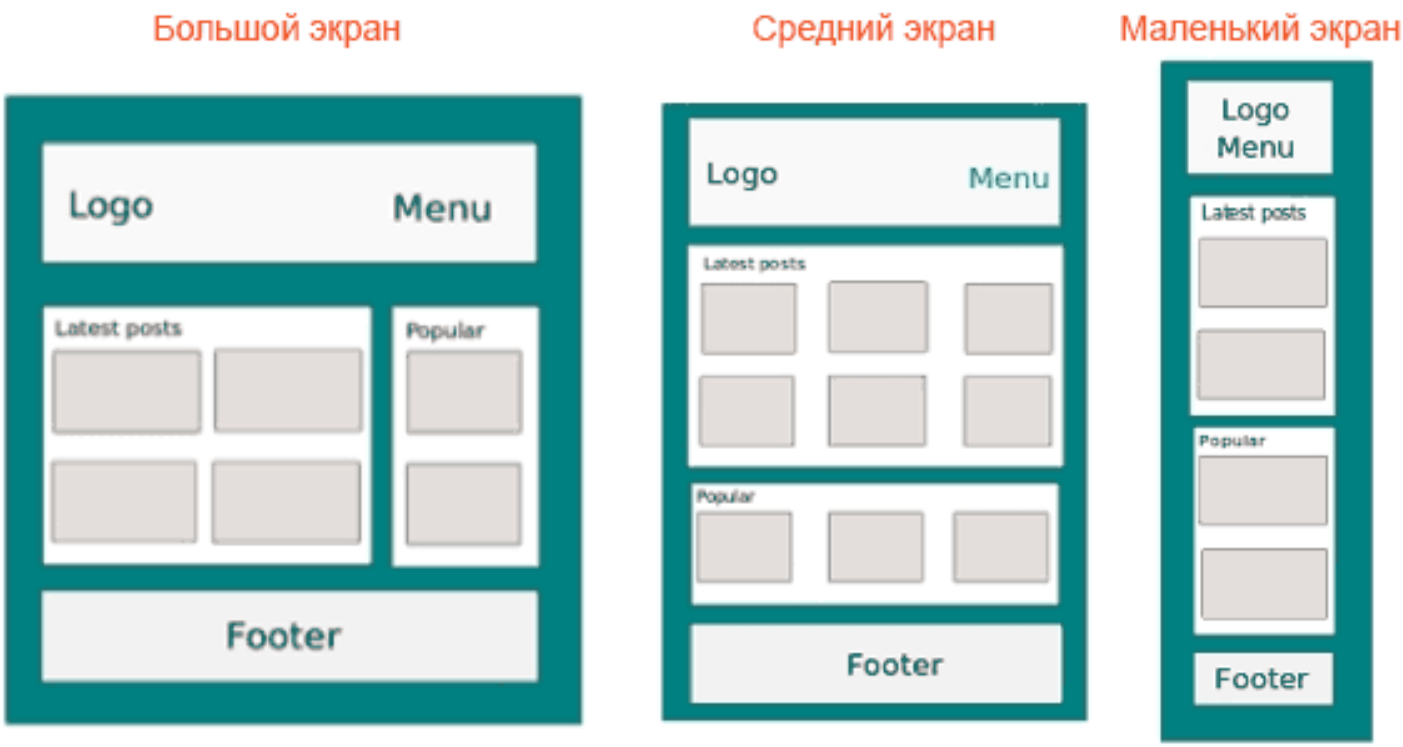


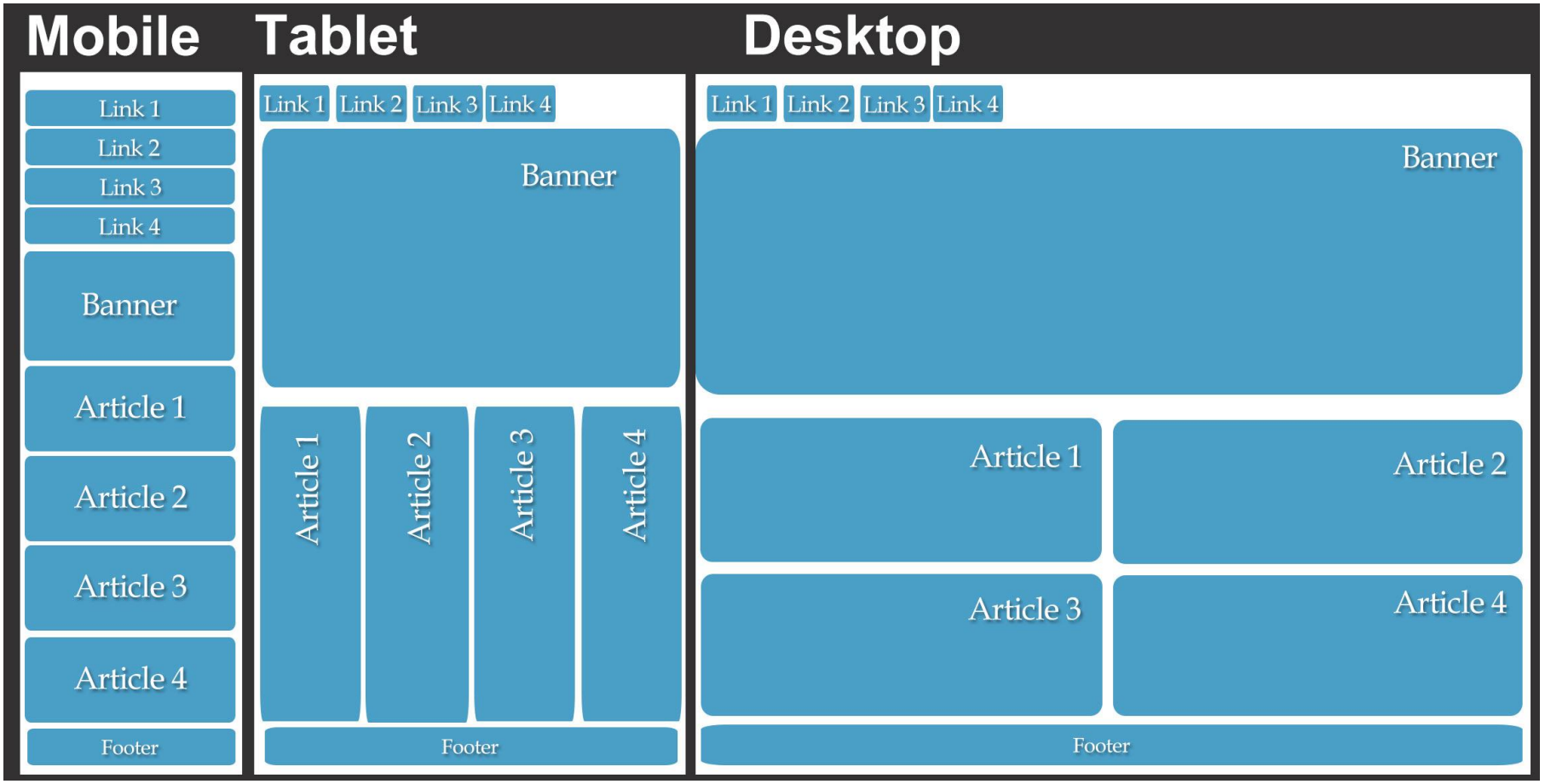
## ОТЗЫВЧИВЫЙ

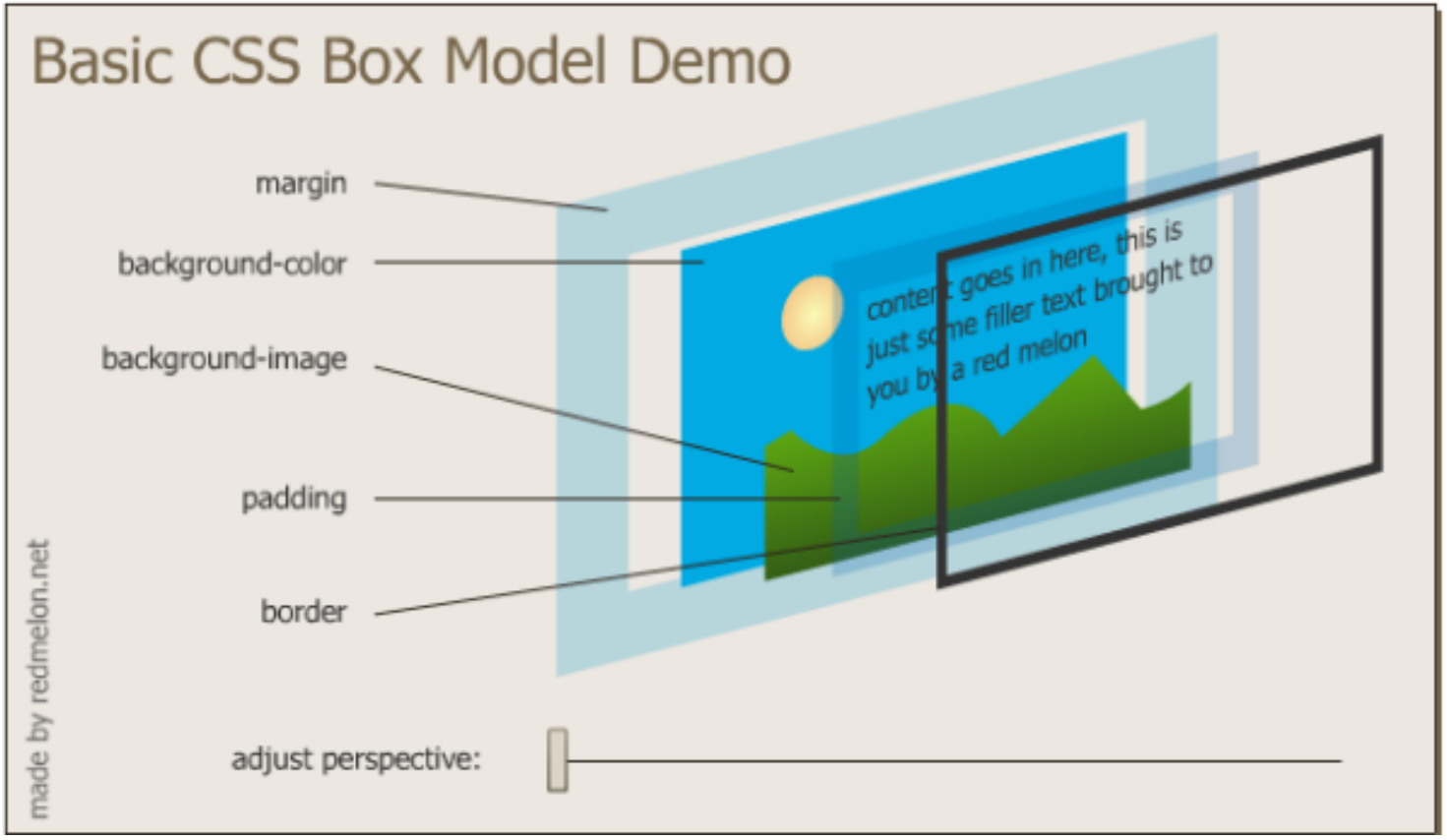
Универсальный дизайн, растягивающий или сужающий контент в зависимости от экрана

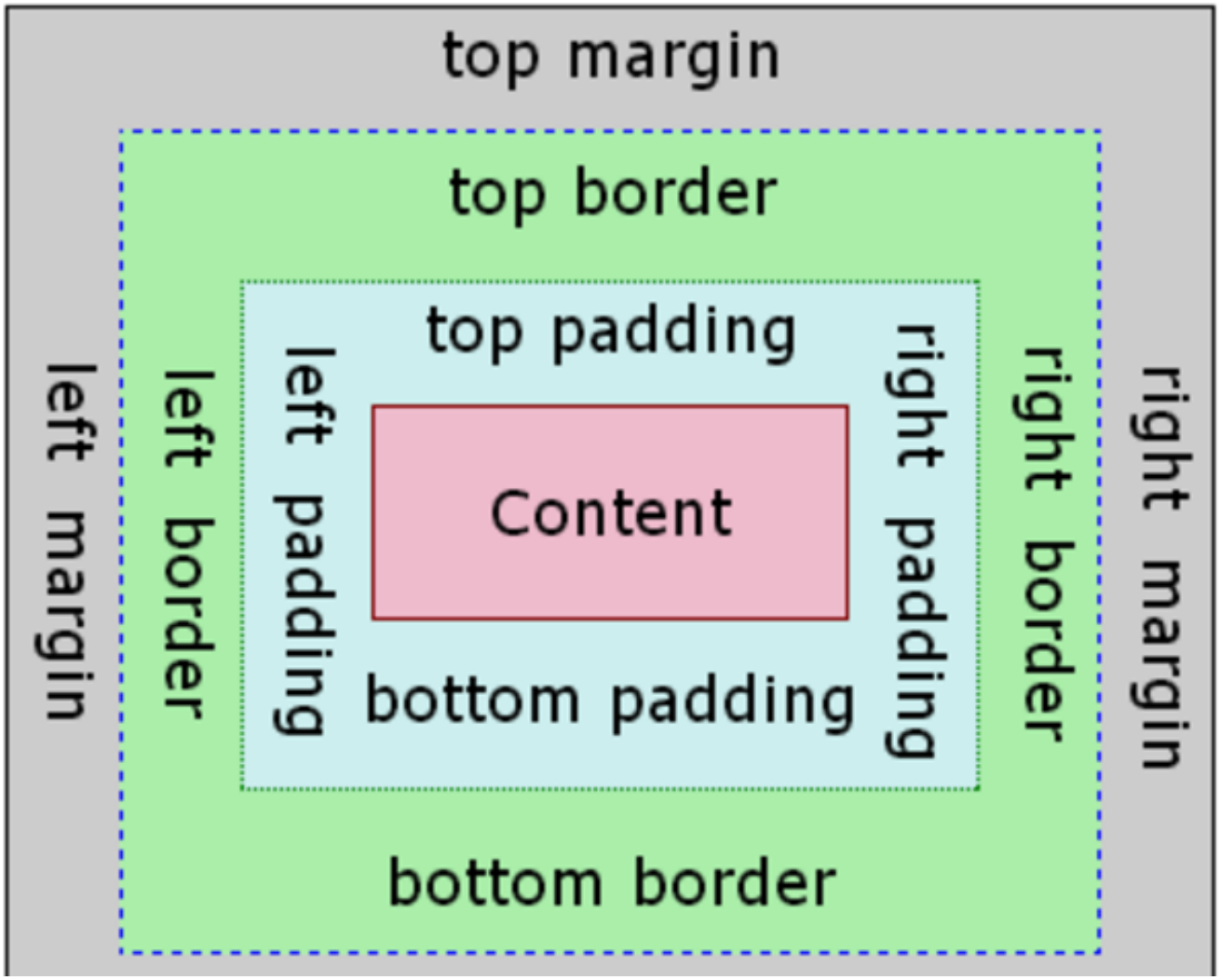


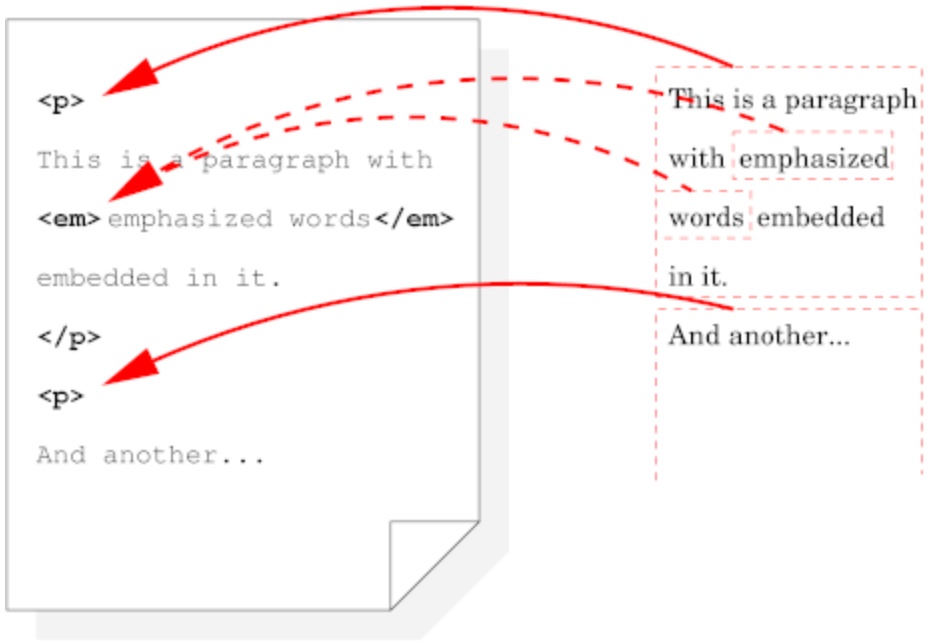

HTML
5


Большой экран
Logo
Menu
Latest posts
Popular
Footer
Средний экран
Logo
Menu
Latest posts
Popular
Footer
Маленький экран
Logo
Menu
Latest posts
Popular
Footer


Mobile
Tablet
Desktop
Link 1
Link 2
Link 3
Link 4
Banner
Article 1
Article 2
Article 3
Article 4
Footer
Link 1
Link 2
Link 3
Link 4
Banner
Article 1
Article 2
Article 3
Article 4
Footer
Link 1
Link 2
Link 3
Link 4
Banner
Article 1
Article 2
Article 3
Article 4
Footer

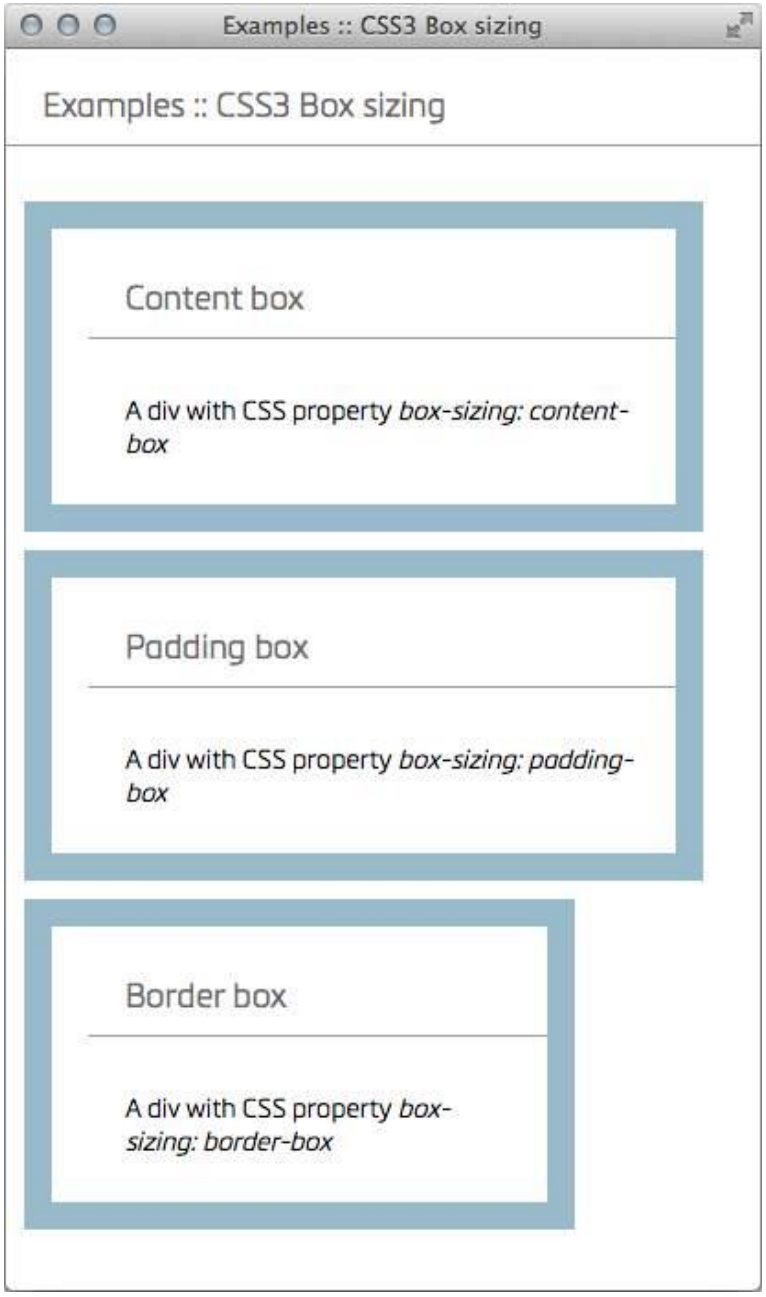

Basic CSS Box Model Demo
margin
background-color
background-image
padding
border
content goes in here, this is just some filler text brought to you by a red melon
adjust perspective:
made by redmelon.net


top margin
top border
top padding
left margin
left border
left padding
Content
right padding
right border
right margin
bottom padding
bottom border



This is a paragraph with
emphasized words
embedded in it.

And another...
This is a paragraph
with emphasized
words embedded
in it.
And another...


Examples :: CSS3 Box sizing
Examples :: CSS3 Box sizing
Content box
A div with CSS property box-sizing: content-box
Padding box
A div with CSS property box-sizing: padding-box
Border box
A div with CSS property box-sizing: border-box

# Разработка интерактивных учебно-методических комплексов

# Цели создания системы электронного обучения

Упростить преподавателям процесс проведения и проверки лабораторных и практических занятий

Предоставить студентам учебные материалы, сгруппированные по модулям, содержащие как теоретические материалы, так и практические задания

Обеспечить формирование отчетов

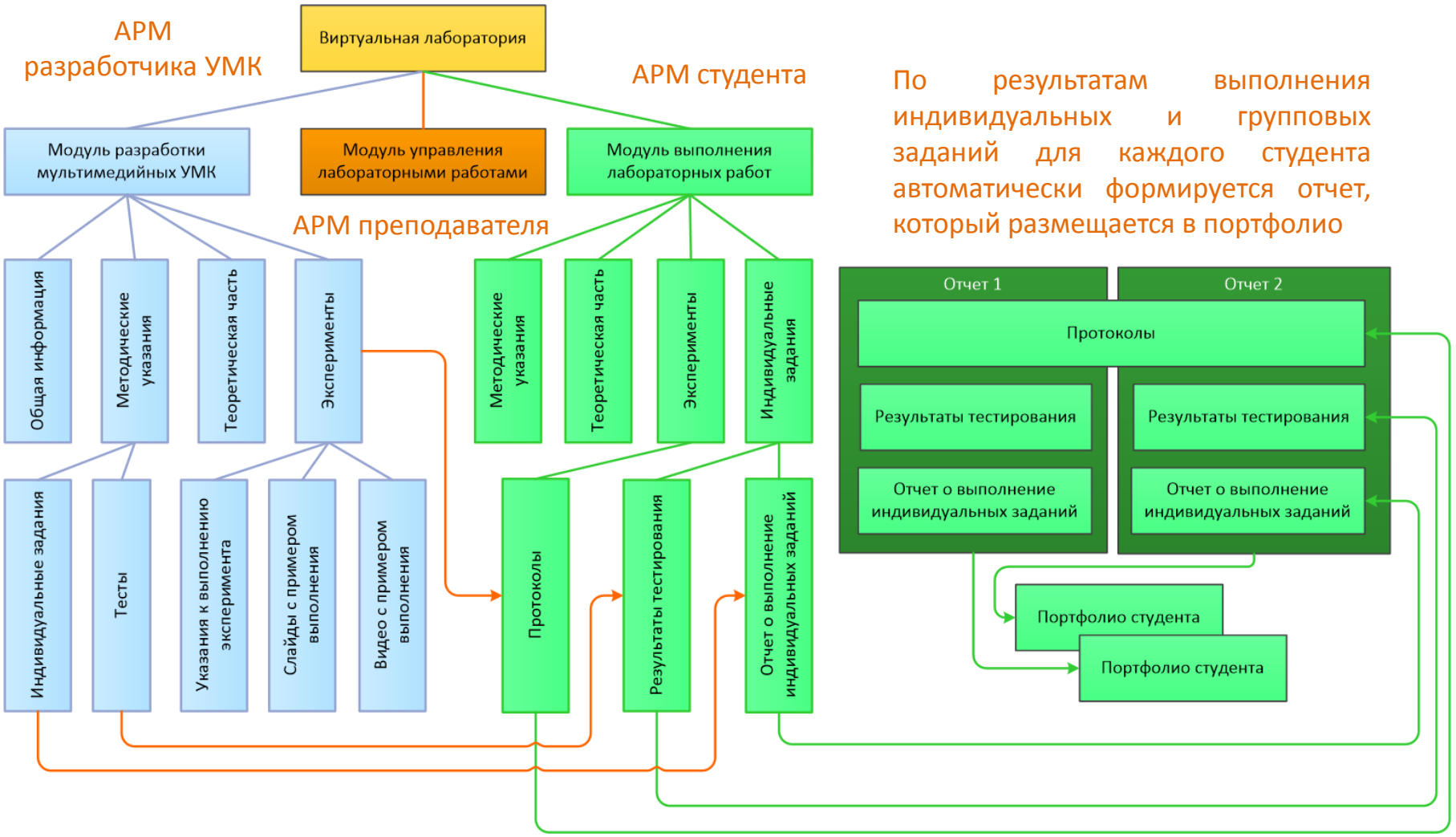
## Интерактивный мультимедийный учебно-методический комплекс дисциплины

Подготовка учебно-методических материалов

Выполнение заданий

Проведение занятий

# Структура интерактивного мультимедийного УМК



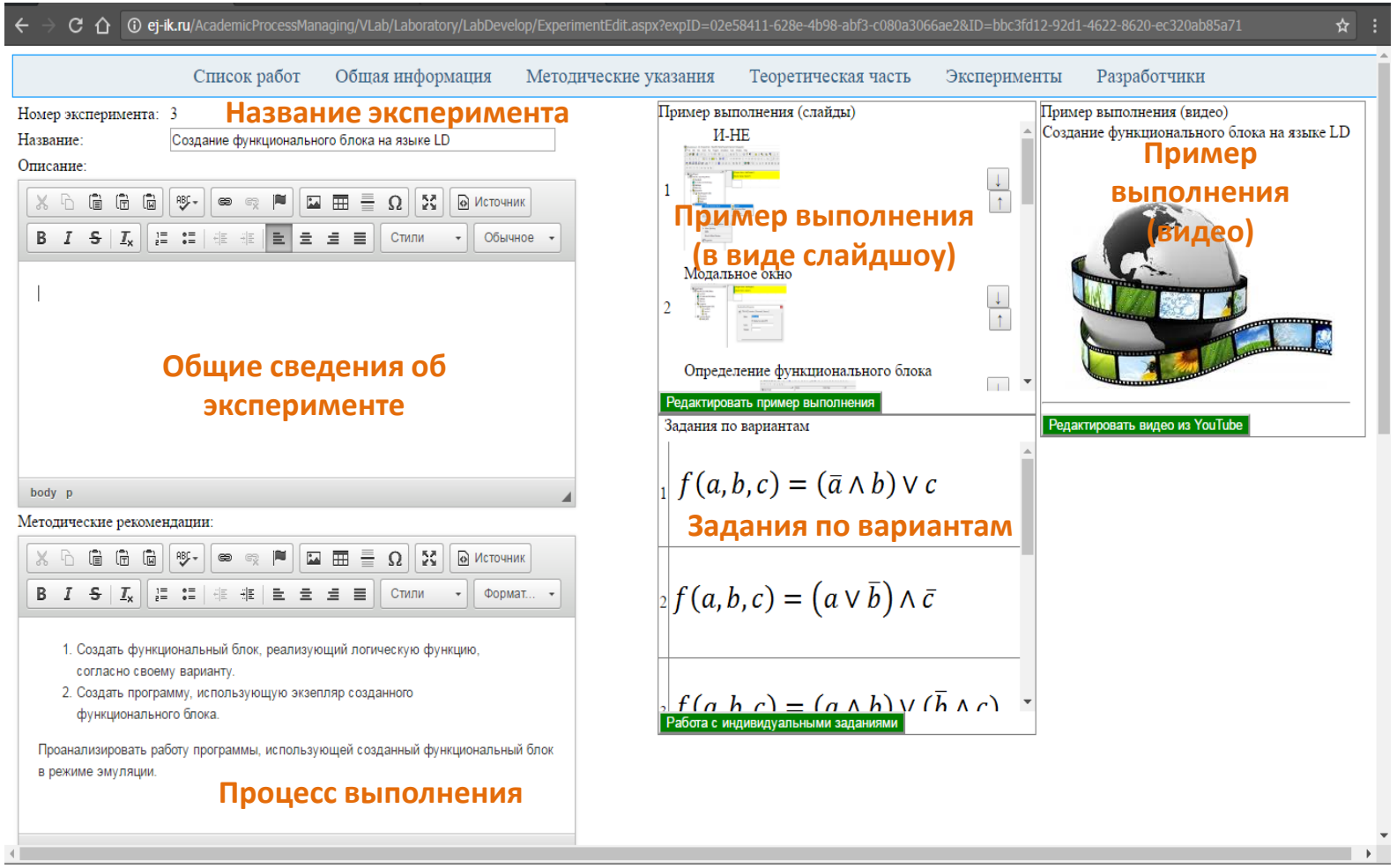

По результатам выполнения индивидуальных и групповых заданий для каждого студента автоматически формируется отчет, который размещается в портфолио

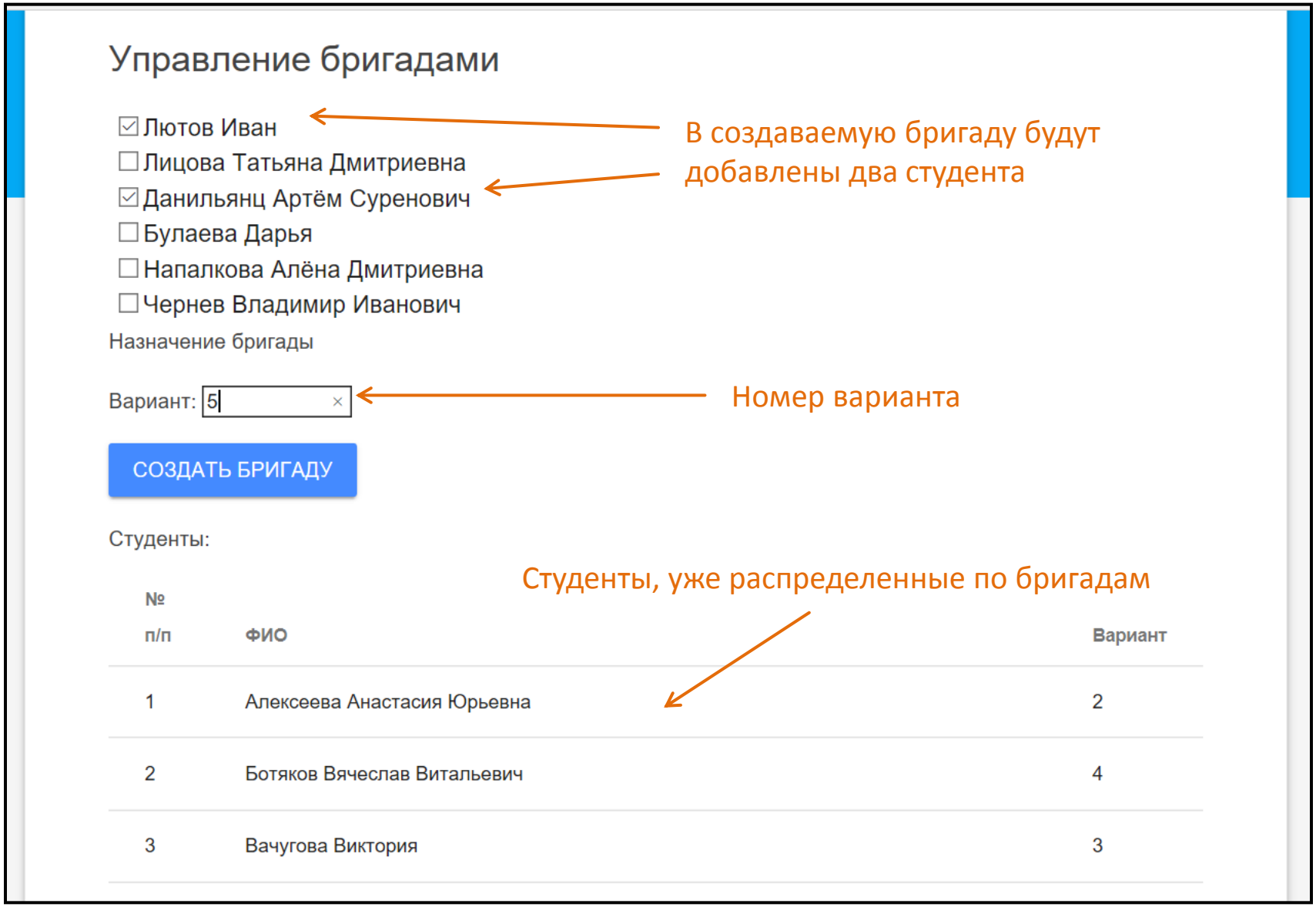
# Для преподавателя система предоставляет

Управление индивидуальной и групповой работой студентов

Возможность контроля процесса выполнения студентами индивидуальных и групповых заданий

# Экранная форма эксперимента лабораторной работы




Управление бригадами
Лютов Иван
Лицова Татьяна Дмитриевна
Данильянц Артём Суренович
Булаева Дарья
Напалкова Алёна Дмитриевна
Чернев Владимир Иванович
В создаваемую бригаду будут добавлены два студента
Назначение бригады
Вариант: 5
Номер варианта
СОЗДАТЬ БРИГАДУ
Студенты:
Студенты, уже распределенные по бригадам
№ п/п
ФИО
Вариант
1
Алексеева Анастасия Юрьевна
2
2
Ботяков Вячеслав Витальевич
4
3
Вачугова Виктория
3

# Оглавление раздела с теоретической информацией по учебному модулю

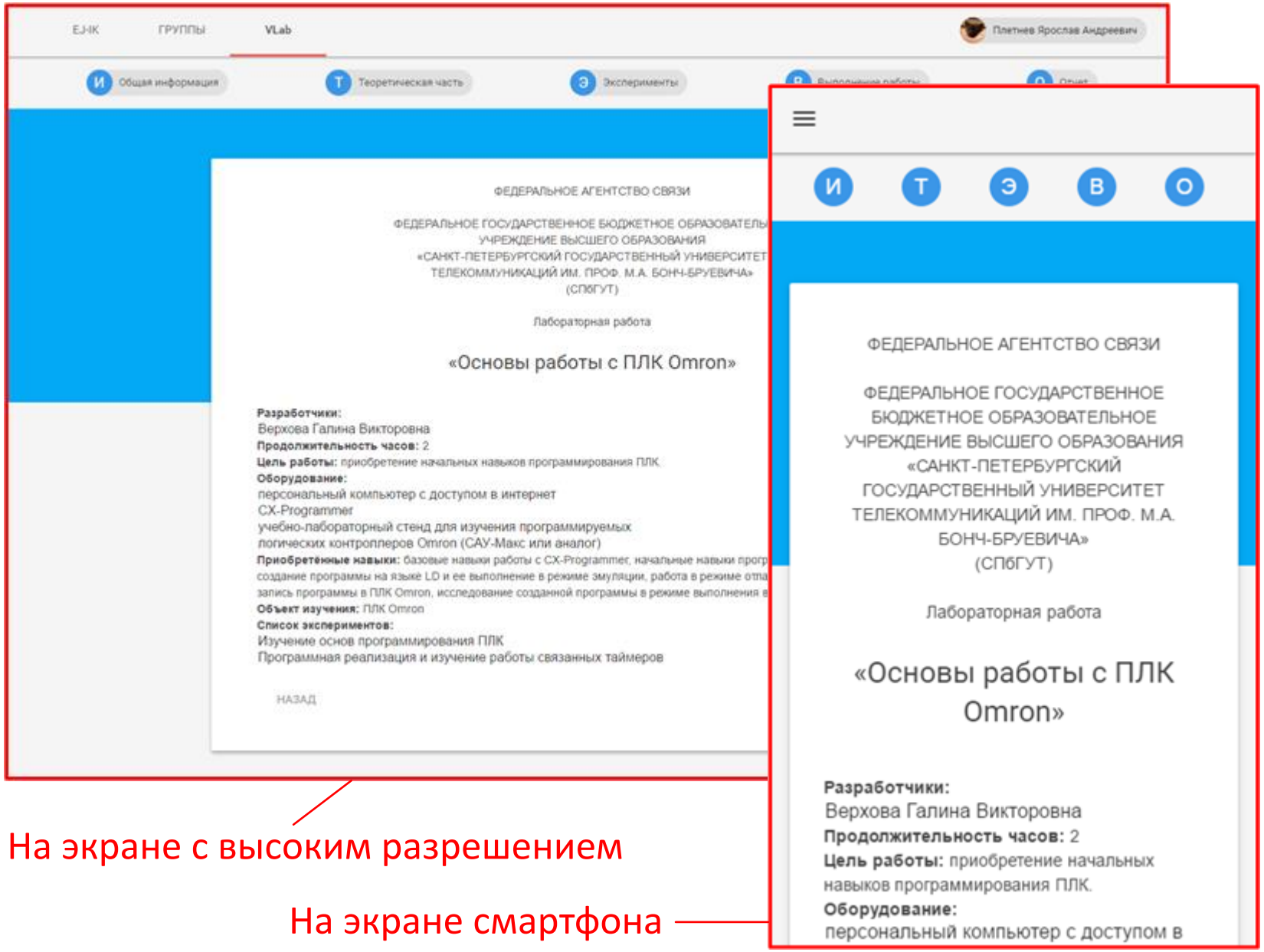
## Основы работы с ПЛК Omron. Теоретическая часть

| № | Название | |
|---|---|---|
| 1 | Обзор языков программирования IEC 61131-3 | ВЫБРАТЬ |
| 2 | Применение ПЛК в автоматизированных и автоматических системах управления | ВЫБРАТЬ |
| 3 | Принцип работы ПЛК и основы языка LD | ВЫБРАТЬ |
| 4 | Обзор ПЛК Omron (назначение, устройство, аппаратный интерфейс) | ВЫБРАТЬ |
| 5 | Среда разработки CX-Programmer | ВЫБРАТЬ |

НАЗАД ВПЕРЕД

# Список доступных для студента работ

| НАЗВАНИЕ ЛАБОРАТОРНОЙ РАБОТЫ | ДЕЙСТВИЕ |
| --- | --- |
| Основы работы с ПЛК Omron | ВЫБРАТЬ |
| Создание системы автоматического управления на основе комбинационных схем | ВЫБРАТЬ |
| Изучение базовых приемов программирования ПЛК на языке LD | ВЫБРАТЬ |




На экране с высоким разрешением

На экране смартфона

НАЗАД

ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

Кафедра Автоматизации предприятий связи

Лабораторная работа

# «Основы работы с ПЛК Omron»

**По результатам выполнения лабораторной работы формируется отчет, содержащий протоколы выполнения экспериментов, результаты тестирования и выполнения индивидуальных заданий**

Работу выполнил студент группы ИСТ-441:
Алексеева Анастасия Юрьевна

Работу проверил:
Белоус Константин Владимирович

Статус: