Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Локтионова Оксана Геннадьевна
Должность: проректор по учебной работе
Дата подписания: 15.06.2023 10:11:51
Уникальный программный ключ:
0b817ca911e6668abb13a5d426d39e5f1c11eabbf73e943df4a4851fda56d089

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«25» 10 _____ 2022 г.

АНАЛИЗ РЫНКА ИНФОРМАЦИОННЫХ СИСТЕМ

Методические указания для выполнения практических заданий для студентов направления подготовки 09.03.03 Прикладная информатика

Курск 2022

УДК 004
Составитель А.С. Сизов

Рецензент
Кандидат технических наук, Киселев А.В.

Анализ рынка информационных систем: методические указания для выполнения практических заданий для студентов направления подготовки 09.03.03 Прикладная информатика/ Юго-Зап. гос. ун-т; сост А.С. Сизов, Курск, 2022. 102 с.

Содержат методические указания для выполнения практических заданий.

Методические указания по структуре, содержанию и стилю изложения материала соответствуют методическим и научным требования, предъявляемым к учебным и методическим пособиям.

Предназначены для студентов направления подготовки 09.03.03 Прикладная информатика.

Текст печатается в авторской редакции

Подписано в печать Формат 60 х 84 1/16.
Усл. печ. л. Уч.- изд. л. Тираж 50 экз. Заказ 2143 Бесплатно.
Юго-Западный государственный университет.
305040, Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

Практическая работа №1
Состав инструментальных средств информационных систем. Нормативно-правовое регулирование информационного обмена, разработки и эксплуатации информационных систем

Материалы для дискуссии:

1. Прежде чем рассматривать инструментальные средства информационных систем, (используя информационно-справочную систему «Консультант Плюс») необходимо раскрыть понятия «информационная система», «автоматизированная информационная система», «информационная технология» *с точки зрения* Федерального закона РФ от 27 июля 2006 г. № 149-ФЗ «Об информации, информационных технологиях и о защите информации».

При этом под инструментальными средствами ИС понимают совокупность аппаратных и программных средств, обеспечивающих функционирование ИС.

2. Используя материалы лекций и самостоятельной работы раскрыть состав и структуру инструментальных средств ИС.

2. Изучить виды испытаний АИС и требования к ним в соответствии с ГОСТ 34.601.

При этом ответить на вопросы:

- Какие виды испытаний АС проводят на стадии "Ввода в действие" по ГОСТ 34.601 и какова цель их проведения?
- Охарактеризуйте этапы испытания АС.
- Какой документ разрабатывают для планирования проведения всех видов испытаний?
- Какова цель проведения опытной эксплуатации АС?
- Какова цель проведения приемочных испытаний АС?
- Какие объекты подвергаются проверке при испытаниях АС?
- Оформлением какого документа сопровождается приемка АС в постоянную эксплуатацию?

Практическая работа №2
Применение инструментальных средства разработки пользовательских приложений (с использованием VBA)
Практическая работа №2 (часть 2.1)
Создание простейших элементов пользовательского интерфейса средствами VBA

Цель работы: знакомство с основными приемами создания простейших элементов пользовательского интерфейса средствами VBA

Создание простейшего окна вывода сообщений

1. Запустите программу *Excel*.
2. Откройте редактор VBA (Visual Basic for Application) (*Сервис* → *Макрос* → *Редактор Visual Basic*).
3. Добавьте в проект новый программный модуль (*Insert* → *Module*).
4. Добавьте новую процедуру, выполнив команду *Insert* → *Procedure…*, а затем введя в поле *Name* значение *MyProcedure* и выбрав в поле *Type* значение *Sub*.
5. Введите недостающий текст так, чтобы содержимое окна редактора имело следующий вид:

```
Public Sub MyProcedure()
  MsgBox ("Привет!")
End Sub
```

6. Установите текстовый курсор в пределах введенного текста и запустите созданную процедуру на выполнение с помощью команды *Run* → *Run Sub/User Form.*
7. Измените текст процедуры как указано ниже и повторите действия п.6.

```
Public Sub MyProcedure()
  s = MsgBox("Привет!", , "Новое сообщение")
     End Sub
```

8. Сравните полученные результаты.

Создание процедуры вычисления площади круга

1. Измените текст процедуры следующим образом:

```
Public Sub MyProcedure()
  Const Pi As Double = 3.1415993
  Dim Radius As Double
  Dim S_circles As Double

  Radius = 10
  S_circles = Pi * Radius ^ 2

  s = MsgBox (S_circles, , "Вычисление площади круга")
End Sub
```

2. Запустите процедуру вычисление площади круга (см. п.6). Измените значение переменной **Radius**, введя произвольное значение (например, **Radius = 12.4**), и вновь запустите процедуру. Изменилось ли вычисленное значение? В чем недостаток процедуры?

3. Замените строку

```
  Radius = ...
```

на

```
  Radius = InputBox("Введите радиус круга:")
```

4. Запустите измененную процедуру. Проверьте правильность вычислений для нескольких значений радиуса.

5. Замените строку

```
  s = MsgBox (S_circles, , "Вычисление площади круга")
```

на

```
  MsgBox ("Площадь круга = " + CStr(S_circles))
```

6. Запустите процедуру. Сравните результат с полученным ранее.

Создание процедуры сравнения двух чисел

1. Измените текст процедуры следующим образом:

```
Public Sub MyProcedure()
  Dim A, B As Double

  A = InputBox("Введите число A:")
  B = InputBox("Введите число B:")

  If A = B Then
```

```
    MsgBox ("Числа A и B равны.")
  Else
   If A > B Then
    MsgBox ("Число A больше числа B.")
   Else
    MsgBox ("Число A меньше числа B.")
   End If
  End If
End Sub
```

2. Запустите процедуру. Проверьте правильность сравнения для нескольких пар чисел A и B.

Самостоятельная работа

Создать процедуру вычисления следующей функции, обратив внимание на область определения (номер задания указывает преподаватель):

1) $y = x^3 \frac{2 + x^2}{\sqrt{2x}} - 1{,}56;$

2) $y = \frac{x - 2x^3 - 2}{4{,}3x} \sqrt{x};$

3) $y = (x - 1)(x^2 + 3{,}14) \frac{1}{\sqrt{x}};$

4) $y = (x^3 + 2x^2 + x - 1{,}89)^2 \frac{1}{x^2};$

5) $y = \frac{(2{,}3x - 1)(\sqrt{x} - \sqrt[4]{x^3})}{x};$

6) $y = x^3 - \frac{2x + 4}{\sqrt{x}} + 23{,}9;$

7) $y = \frac{x - 0{,}1x^3 + 1}{5x} + \frac{\sqrt{x}}{6};$

8) $y = (x + 1{,}2)(x^2 - 2{,}72) - \frac{1}{\sqrt{5x}};$

9) $y = (0{,}61x^3 - 2{,}4x^2 - x + 1)^2 \frac{1}{x^2 - 1};$

10) $y = 7{,}64 - \frac{(x + 8)(1 + \sqrt{x} - \sqrt{x^3})}{x};$

11) $y = \frac{7}{x^3} + \frac{0{,}2x}{\sqrt{x}} - 2{,}718$;

12) $y = 9{,}98 - \frac{x - x^4}{3x} + \frac{\sqrt{x}}{9}$;

13) $y = (2{,}2 - x)(x^2 - 1{,}13) + \frac{4}{\sqrt[4]{1{,}1x}}$;

14) $y = (9{,}2x^5 - 13x^3 - x)\frac{1}{x^3 - 4{,}3}$;

15) $y = 14{,}05x - \frac{(x-6)(11 - \sqrt{x} - \sqrt{x^3})}{x}$.

Отчет по* части 2.1 *практической работы 2 должен содержать: цель работы; краткое описание использованных в работе элементов языка VBA; текст созданных процедур и результаты их работы; выводы по результатам практической работы. При оформлении отчета в печатном виде в нижний колонтитул следует поместить фамилию, инициалы и номер группы обучаемого (8 пт., Arial, выравнивание по правому краю).

Практическая работа №2 (часть 2.2)

Применение инструментальных программных средств для создания пользовательских функций при решении научно-практических задач

Цель работы: изучение приемов создания пользовательской функции средствами VBA при решении научно-практических задач.

Создание пользовательской функции

1. Запустите программу Excel.
2. Откройте редактор VBA (Visual Basic for Application) (*Сервис* → *Макрос* → *Редактор Visual Basic*).
3. Добавьте в проект новый программный модуль (*Insert* → *Module*).
4. Добавьте новую процедуру, выполнив команду *Insert* → *Procedure…*, а затем и введя в поле *Name* значение *MyProcedure* и выбрав в поле *Type* значение *Sub*.

5. Введите недостающий текст так, чтобы содержимое окна редактора имело следующий вид:

```
Public Sub MyProcedure()
  Dim N, I As Integer
  Dim F As Long

  N = InputBox("Введите число N:")

  F = 1
  For I = 1 To N
  F = F * I
  Next

  MsgBox ("N!=" + CStr(F))
End Sub
```

6. Установите текстовый курсор в пределах введенного текста и запустите созданную процедуру на выполнение с помощью команды *Run → Run Sub/User Form*. Проверьте правильность вычисления факториала для значений числа N от 0 до 13. В чем причина возникновения ошибки?

7. Замените в строке **Dim F As Long**
Long на **Double**.

8. Снова запустите процедуру и проверьте правильность вычисления факториала для нескольких значений числа N, начиная с 13. Как можно объяснить полученный результат?

9. Для создания на основе имеющейся процедуры пользовательской функции измените введенный ранее текст следующим образом:

```
Public Function ФАКТОРИАЛ(N As Integer) As Double
  Dim I As Integer
  Dim F As Double

  F = 1
  For I = 1 To N
    F = F * I
  Next

  ФАКТОРИАЛ = F
```

End Function

Для добавления в модуль новой функции можно воспользоваться командой *Insert → Procedure…*, введя затем в поле *Name* имя создаваемой функции и выбрав в поле *Type* значение *Function*.

10. Выберите Лист 1 книги Excel и установите для всех ячеек формат *Числовой*, число десятичных знаков – 0. Введите в ячейку A1 число 0, а в ячейку B1 – формулу =ФАКТОРИАЛ(A1).

При создании формулы можно воспользоваться Мастером функций, выбрав категорию *Определенные пользователем*.

11. Введите в ячейку A2 число 1, выделите диапазон A1:A2 и с помощью автозаполнения заполните ячейки до A21 включительно. Заполните соответствующие ячейки столбца B, также используя автозаполнение.

12. В ячейку C1 введите формулу =ФАКТР(A1). С помощью автозаполнения заполните другие ячейки столбца C (до C21 включительно).

13. Убедитесь в равенстве значений, полученных с помощью созданной функции ФАКТОРИАЛ() и стандартной функции ФАКТР().

Создание процедуры заполнения ячеек листа Excel случайными числами

1. Добавьте в модуль новую процедуру, выполнив команду *Insert → Procedure…*, а затем и введя в поле *Name* значение *FillRandom* и выбрав в поле *Type* значение *Sub*.

2. Введите недостающий текст так, чтобы содержимое окна редактора имело следующий вид:

```
Public Sub FillRandom()
  Dim I As Integer
  Randomize
  For I = 1 To 10
    Cells(I, 1) = Rnd * 100
  Next
End Sub
```

3. Выберите Лист 2 книги Excel и запустите созданную процедуру с помощью команды *Сервис* → *Макрос* → *Макросы…*, выбрав затем имя макроса FillRandom и нажав кнопку *Выполнить*.

4. Запустите процедуру несколько раз, убедившись, что ячейки A1:A10 каждый раз принимают новые значения.

5. Введите в ячейку C1 формулу =СЛЧИС()*100. Используя автозаполнение, распространите ее на ячейки C2:C10.

6. Определите, в чем разница в использовании двух приведенных способов заполнения ячеек случайными значениями.

7. Измените текст процедуры следующим образом:

```
Public Sub FillRandom()
  Dim I, R, C As Integer
  Randomize
  R = ActiveCell.Row
  C = ActiveCell.Column
  For I = R To R + 10
    Cells(I, C) = Rnd * 100
  Next
End Sub
```

8. Выберите Лист 3 книги Excel и запустите процедуру несколько раз (см. п.3), выбирая на листе каждый раз активную ячейку произвольным образом. Что изменилось в работе функции?

Самостоятельная работа

Создать в MS Excel таблицу, имеющую следующую структуру:

	A	B	C	D
1	**Наименование товара**	**Цена**	**Количество**	**Стоимость**
2	Товар 1	- р.	0	- р.
3	Товар 2	- р.	0	- р.
4	Товар 3	- р.	0	- р.
5	Товар 4	- р.	0	- р.
6	Товар 5	- р.	0	- р.
7				
8			Сумма	**- р.**
9			Сумма со скидкой	**- р.**

Для расчета поля «Сумма со скидкой» создать средствами VBA пользовательскую функцию СоСкидкой(). Функция должна возвращать величину суммы со скидкой, в качестве параметра функции указывается исходная сумма (без скидки).

Расчет производится следующим образом:

- часть суммы, превышающая S_1, уменьшается на D_1 (%);
- часть полученной суммы, превышающая S_2, уменьшается на D_2 (%);
- часть полученной суммы, превышающая S_3, уменьшается на D_3 (%).

Значения S_1, S_2, S_3 и D_1, D_2, D_3. указываются преподавателем.

Отчет по части 2.2 практической работы 2 должен содержать: цель работы; краткое описание структур процедуры и функции; тексты созданных процедур и функций с их подробным описанием, а также результаты их работы; результаты выполнения самостоятельной работы с подробным описанием созданной функции; выводы по результатам практической работы. При оформлении отчета в печатном виде в нижний колонтитул следует поместить фамилию, инициалы и номер группы обучаемого (8 пт., Arial, выравнивание по правому краю).

Практическая работа №2 (часть 2.3)

Применение инструментальных программных средств для создания пользовательских форм при решении научно-практических задач

Цель работы: приобретение навыков применения инструментальных программных средств визуального и объектно-ориентированного программирования для создания пользовательских форм при решении научно-практических задач (на примере создания пользовательской формы в среде VBA)

Порядок выполнения работы

1. Запустите программу Excel.
2. Откройте редактор VBA (Visual Basic for Application) (*Сервис → Макрос → Редактор Visual Basic*).
3. Добавьте в проект новую форму (*Insert → UserForm*).
4. Установите в окне свойств (*Properties*) для созданной формы (*UserForm1*) значение свойства *Width* (ширина) равным *300*, а

свойства *Height* (высота) равным *252*. Введите заголовок формы (свойство *Caption*) “Решение квадратного уравнения” (здесь и далее вводимые строки – без кавычек!).

5. Выберите на панели инструментов (*Toolbox*) элемент управления типа *Label* (надпись). Создайте в левом верхнем углу формы элемент управления *Label1* (аналогично создания прямоугольника в графическом редакторе, имя будем присвоено по умолчанию).

6. Для созданной надписи установите значения следующих свойств: *Font* – Arial, жирный, 14 пт.; *Caption* – “a =”.

7. Скопируйте надпись *Label1*, поместив копию путем перетаскивания под оригиналом (по умолчанию копии будет присвоено имя *Label2*). Измените для *Label2* свойство *Caption* на “b =”.

8. Скопируйте надпись *Label2*, поместив копию путем перетаскивания под оригиналом (по умолчанию копии будет присвоено имя *Label3*). Измените для *Label3* свойство *Caption* на “c =”.

9. Выберите на панели инструментов (*Toolbox*) элемент управления типа *TextBox* (текстовое поле). Создайте на форме напротив надписи *Label1* элемент управления *TextBox1*.

10. Для созданного элемента управления *TextBox1* установите значение свойства *Font* – Arial, жирный, 14 пт.

11. Скопируйте элемент управления *TextBox1* два раза, поместив копии *TextBox2* и *TextBox3* напротив *Label2* и *Label3* соответственно.

12. Скопируйте надпись *Label3*, поместив копию напротив *TextBox1*. Для созданной надписи *Label4* установите значения следующих свойств: *Font* – Arial, жирный курсив, 14 пт.; *Caption* – “a*x^2 + b*x + c = 0”.

13. Выберите на панели инструментов (*Toolbox*) элемент управления типа *CommandButton* (кнопка). Создайте на форме напротив поля *TextBox3* элемент управления *CommandButton1*.

14. Для созданной кнопки установите значения следующих свойств: *Font* – Arial, жирный, 14 пт.; *Caption* – “Решить”.

15. Выберите на панели инструментов (*Toolbox*) элемент управления типа *Label* (надпись). Создайте на форме ниже ранее

созданных элементов управления надпись *Label5*, установите ширину надписи несколько меньше ширины формы.

16. Для созданной надписи установите значения следующих свойств: *Font* – Arial, жирный, 12 пт.; *ForeColor* – “Highlight”; *TextAlign* – “2 - fmTextAlignCenter”.

17. Скопируйте надпись *Label5* три раза, поместив копии *Label6*, *Label7* и *Label8* под надписью *Label5*.

18. Созданная форма должна иметь вид, приведенный на рисунке.

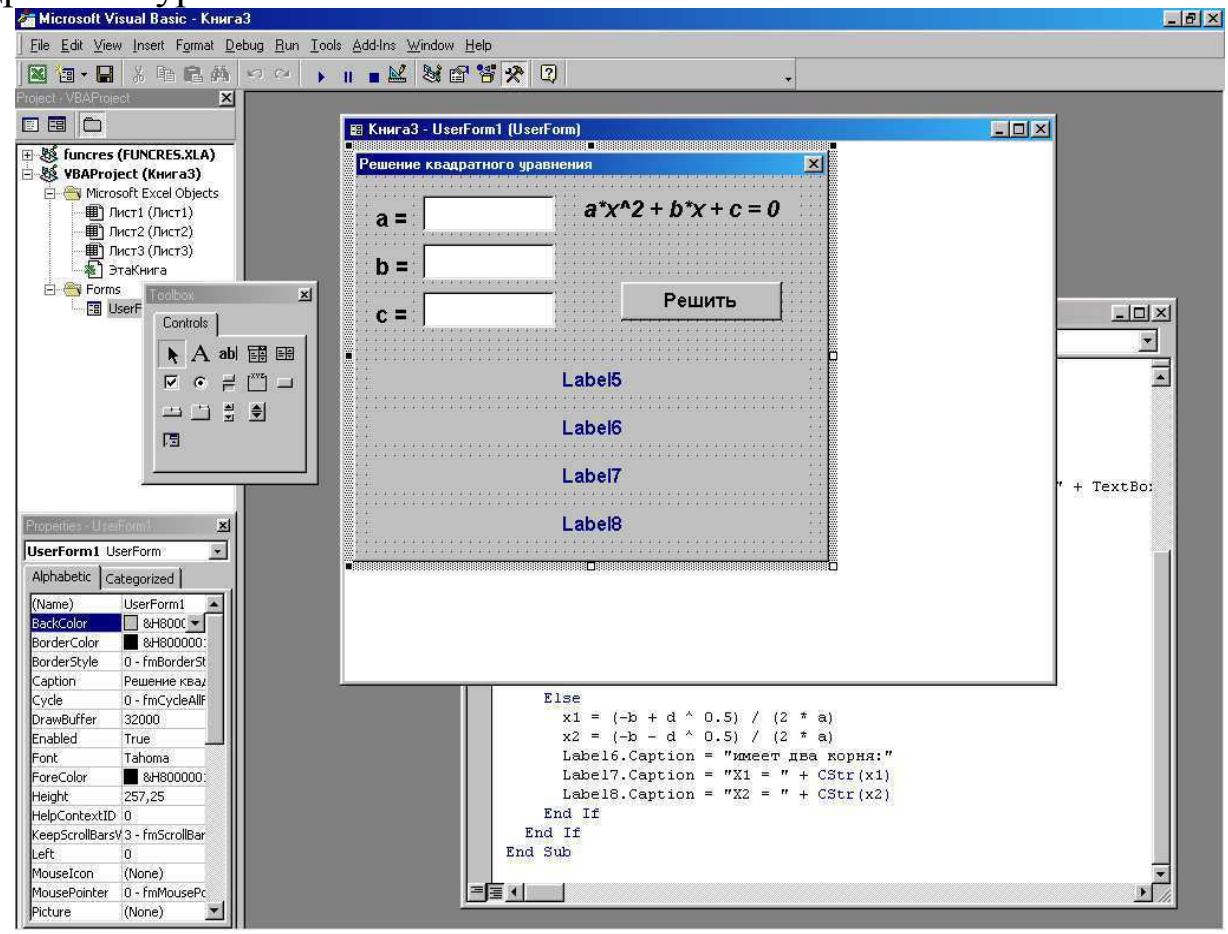
19. Выполните двойной щелчок левой кнопкой «мыши» на экранной кнопке *CommandButton1*.

20. Введите недостающий текст так, чтобы содержимое окна редактора имело следующий вид:

```
Private Sub CommandButton1_Click()
  Dim a, b, c As Double
  Dim d, x1, x2 As Double
  If TextBox1.Text = "" Then
    TextBox1.Text = "1"
  End If
  If TextBox2.Text = "" Then
    TextBox2.Text = "0"
  End If
  If TextBox3.Text = "" Then
    TextBox3.Text = "0"
  End If
  a = CDbl(TextBox1.Text)
  b = CDbl(TextBox2.Text)
  c = CDbl(TextBox3.Text)
  d = b ^ 2 - 4 * a * c
  Label5.Caption = "Уравнение " + TextBox1.Text + _
                   "*X^2 + " + TextBox2.Text + "*X + " + _
                   TextBox3.Text + " = 0"
  If d < 0 Then
    Label6.Caption = "корней не имеет!"
    Label7.Caption = ""
    Label8.Caption = ""
  Else
    If d = 0 Then
```

```
    x1 = -b / (2 * a)
    Label6.Caption = "имеет один корень:"
    Label7.Caption = "X = " + CStr(x1)
    Label8.Caption = ""
   Else
    x1 = (-b + d ^ 0.5) / (2 * a)
    x2 = (-b - d ^ 0.5) / (2 * a)
    Label6.Caption = "имеет два корня:"
    Label7.Caption = "X1 = " + CStr(x1)
    Label8.Caption = "X2 = " + CStr(x2)
  End If
     End If
End Sub
```

21. Запустите созданную форму на выполнение с помощью команды *Run → Run Sub/User Form*. Проверьте правильность работы программы при разных значениях коэффициентов *a*, *b* и *c* квадратного уравнения.



Отчет по части 2.3 практической работы 2 должен содержать: цель работы; краткое описание структур процедуры и

функции; созданные формы, а также результаты проверки правильности работы программ; выводы по результатам практической работы. При оформлении отчета в печатном виде в нижний колонтитул следует поместить фамилию, инициалы и номер группы обучаемого (8 пт., Arial, выравнивание по правому краю).

Практическая работа №3
Инструментальные средства реализации проектов малой и средней сложности

Цель работы: приобретение навыков применения инструментальных средств разработки программного обеспечения при реализации проектов малой и средней сложности (на примере интегрированной среды разработки PascalABC.NET)

Теоретические сведения

Платформа Microsoft.NET - это комплекс программ, устанавливаемый поверх операционной системы и обеспечивающий выполнение программ, написанных специально для .NET.

NET-программы компактны, пользуются единым набором типов данных и библиотек. Компания Microsoft активно развивает платформу .NET, выпуская новые версии с расширенными возможностями.

В результате компиляции .NET-программы генерируется не машинный код, а так называемый байт-код, содержащий команды виртуальной машины (в .NET он называется IL-кодом от англ. Intermediate Language - промежуточный язык). Команды байт-кода не зависят от процессора и используемой операционной системы. При запуске программа, содержащая IL-код, подается на вход виртуальной машины, которая и производит выполнение программы. Часть виртуальной машины, называемая JIT-компилятором (Just In Time - непосредственно в данный момент), сразу после запуска .NET-программы переводит ее промежуточный код в машинный (проводя при этом его оптимизацию), после чего запускает программу на исполнение. Если быть точными, то промежуточный код переводится в машинный частями по мере выполнения программы.

Такой способ двойной компиляции сложнее обычного, но имеет ряд преимуществ. Во-первых, JIT-компилятор может определить тип процессора, установленного на данном компьютере, поэтому генерирует максимально эффективный машинный код. Тесты показывают, что за счет этого некоторые программы выполняются даже быстрее обычных. Во-вторых, IL-код - гораздо более высокоуровневый, чем машинный, и содержит ряд объектно-ориентированных команд. В их числе - команда newobj вызова конструктора объекта, команда callvirt вызова виртуального метода объекта и команда throw генерации исключения.

Программа или библиотека для .NET называется *сборкой* и имеет традиционное расширение - exe или dll. Поскольку в сборках содержится IL-код, они значительно компактнее обычных программ и библиотек. Так, приложение с главным окном, меню и элементами управления занимает на диске всего несколько десятков килобайт.

Наиболее "чистым" .NET-языком является **C#**: он создавался специально для платформы .NET и включает практически все ее возможности. .NET-языки легко взаимодействуют друг с другом не только за счет высокоуровневого промежуточного кода, но и за счет общей системы типов (CTS - Common Type System - общая система типов). Все стандартные типы (строковые, символьные, числовые и логический) имеют одинаковое представление в памяти во всех .NET-языках. Это позволяет, например, создать библиотеку dll на C#, поместить в нее описание класса, а затем воспользоваться этой библиотекой из программы на PascalABC.NET, сконструировав объект данного класса. Можно также разработать библиотеку на PascalABC.NET, а потом подключить ее к проекту на Visual Basic.NET. Отметим, что традиционные библиотеки dll не позволяют хранить классы, доступные извне, и обладают рядом других ограничений.

Достоинства платформы .NET:

1. Платформа .NET поддерживает множество .NET-языков. В их числе C#, Visual Basic.NET, F#, управляемый C++, Delphi Prism, Oberon, Zonnon, Iron Python, Iron Ruby,PascalABC.NET.
2. Любой .NET-язык содержит самые современные языковые возможности: классы, свойства, полиморфизм, исключения, перегрузка операций, легкое создание библиотек.

3. .NET-языки легко сочетаются друг с другом, похожи друг на друга по синтаксическим конструкциям и системе типов.
4. Имеется обширная библиотека стандартных классов FCL.
5. .NET-приложения компактны.
6. Платформа .NET активно развивается фирмой Microsoft, добавляются как новые языковые возможности, так и новые библиотеки.
7. Компилятор .NET-языка создать значительно проще, чем компилятор обычного языка.

Недостатки платформы .NET

1. Запуск .NET-приложения выполняется в несколько раз медленнее запуска обычного приложения, поскольку требует загрузки в оперативную память компонентов виртуальной машины и внешних библиотек.
2. .NET-код в некоторых ситуациях работает медленнее обычного (однако, в большинстве задач это отставание незначительно, а в некоторых - приложения .NET могут опережать обычные программы).
3. Сборщик мусора начинает работу в момент исчерпания динамической памяти, его работа занимает несколько миллисекунд. Для приложений реального времени это непозволительно.
4. Запуск .NET-приложения обязательно требует установки на компьютере платформы .NET. Без нее приложение работать не будет (Отметим, что в Windows Vista и в Windows 7платформа .NET встроена).

Отметим, что достоинства платформы .NET многократно перекрывают ее недостатки.

Рассмотрим возможность применения платформы Microsoft.NET при создании приложений малой и средней сложности с использованием языка PascalABC.NET.

PascalABC.NET включает в себя практически весь стандартный язык Паскаль, а также большинство языковых расширений языка Delphi. Однако, этих средств недостаточно для современного программирования. Именно поэтому PascalABC.NET расширен рядом конструкций, а его стандартный модуль - рядом подпрограмм, типов и классов, что позволяет создавать легко читающиеся приложения средней сложности.

Кроме этого, язык PascalABC.NET использует большинство средств, предоставляемых платформой .NET: единая система типов, классы, интерфейсы, исключения, делегаты, перегрузка операций, обобщенные типы (generics), методы расширения, лямбда-выражения.

Стандартный модуль PABCSystem, автоматически подключаемый к любой программе, содержит огромное количество стандартных типов и подпрограмм, позволяющих писать ясные и компактные программы.

В распоряжении PascalABC.NET находятся все средства .NET-библиотек классов, постоянно расширяющихся самыми современными возможностями. Это позволяет легко писать наPascalABC.NET приложения для работы с сетью, Web, XML-документами, использовать регулярные выражения и многое другое.

Язык PascalABC.NET позволяет программировать в классическом процедурном стиле, в объектно-ориентированном стиле и содержит множество элементов для программирования в функциональном стиле. Выбор стиля или комбинации этих стилей - дело вкуса программиста, а при использовании в обучении - методический подход преподавателя.

Программа содержит ключевые слова, идентификаторы, комментарии. Ключевые слова используются для выделения синтаксических конструкций и подсвечиваются жирным шрифтом в редакторе. Идентификаторы являются именами объектов программы и не могут совпадать с ключевыми словами.

Программа на языке PascalABC.NET имеет следующий вид:

program *имя программы*;
раздел **uses**
раздел описаний
begin
  операторы
end.

Первая строка называется ***заголовком программы*** и не является обязательной.

Раздел uses состоит из нескольких подряд идущих секций uses, каждая из которых начинается с ключевого слова uses, за которым

следует список имен модулей и пространств имен .NET, перечисляемых через запятую.

Раздел описаний может включать следующие подразделы:

- раздел описания переменных
- раздел описания констант
- раздел описания типов
- раздел описания меток
- раздел описания процедур и функций

Данные подразделы следуют друг за другом в произвольном порядке.

Описание переменных

Переменные могут быть описаны в разделе описаний, а также непосредственно внутри любого блока **begin/end**.

Раздел описания переменных начинается с ключевого слова **var**, после которого следуют элементы описания вида

список имен: *тип*;

или

имя: *тип* := *выражение*;

или

имя: *тип* = *выражение*; // для совместимости с Delphi

или

имя := *выражение*;

Имена в списке перечисляются через запятую. Например:

```
var
  a,b,c: integer;
  d: real := 3.7;
  s := 'PascalABC forever';
  al := new List<integer>;
  p1 := 1;
```

В последних трех случаях тип переменной автоматически определяется по типу правой части.

Переменные могут описываться непосредственно внутри блока. Такие описания называются внутриблочными и представляют собой оператор описания переменной.

Кроме того, переменные-параметры цикла могут описываться в заголовке операторов for и foreach.

Глобальные переменные инициализируются нулевыми значениями. Для локальных переменных это не гарантируется - их следует инициализировать явно.

Описание констант

Раздел описания именованных констант начинается со служебного слова **const**, после которого следуют элементы описания вида

имя константы = значение;

или

имя константы : тип = значение;

Например:

```
const
  Pi = 3.14;
  Count = 10;
  Name = 'Mike';
  DigitsSet = ['0'..'9'];
  Arr: array [1..5] of integer = (1,3,5,7,9);
  Rec: record name: string; age: integer end = (name: 'Иванов'; age: 23);
  Arr2: array [1..2,1..2] of real = ((1,2),(3,4));
```

Описание меток

Раздел описания меток начинается с зарезервированного слова **label**, после которого следует список меток, перечисляемых через запятую. В качестве меток могут быть использованы идентификаторы и положительные целые числа:

label a1,l2,777777;

Метки используются для перехода в операторе goto.

Описание типов

Раздел описания типов начинается со служебного слова **type**, после которого следуют строки вида

имя типа = тип;

Например,

```
type
  arr10 = array [1..10] of integer;
  myint = integer;
  pinteger = ^integer;
  IntFunc = function(x: integer): integer;
```

Обычно описание используется для составных типов (статические массивы, процедурные переменные, записи, классы) чтобы дать имя

сложному типу. Если для типа определенаименная эквивалентность типов, это единственный способ передать переменные этого типа в подпрограмму.

Описание типов для классов использовать обязательно:

```
type
  A = class
    i: integer;
    constructor Create(ii: integer);
    begin
      i:=ii;
    end;
  end;
```

Если описание типа используется просто для того чтобы заменить одно имя на другое, то такие типы называются *синонимами типов*:

```
type
  int = integer;
  double = real;
```

Описания типов могут быть обобщёнными, т.е. включать параметры-типы в угловых скобках после имени типа.

```
type
  Dict<K,V> = Dictionary<K,V>;
  Arr<T> = array of T;
```

Использование такого типа с конкретным параметром-типом называется *инстанцированием* типа:

```
var
  a: Arr<integer>;
  d: Dict<string,integer>;
```

При описании рекурсивных структур данных указатель на тип может фигурировать раньше описания самого типа в определении другого типа:

```
type
  PNode = ^TNode;
  TNode = record
    data: integer;
    next: PNode;
  end;
```

При этом важно, чтобы определения обоих типов находились в одном разделе **type**.

В отличие от Delphi Object Pascal следующее рекурсивное описание верно:

```
type
  TNode = record
    data: integer;
    next: ^TNode;
  end;
```

Отметим, что для ссылочных типов (классов) разрешается описание поля с типом, совпадающим с типом текущего класса:

```
type
  Node = class
    data: integer;
    next: Node;
  end;
```

Область действия идентификатора

Любой используемый в программе идентификатор должен быть предварительно описан. Идентификаторы описываются в разделе описаний. Идентификаторы для переменных могут также описываться внутри блока.

Основная программа, подпрограмма, блок, модуль, класс образуют так называемое ***пространство имен*** - область в программе, в которой имя должно иметь единственное описание. Таким образом, в одном пространстве имен не может быть описано двух одинаковых имен (исключение составляют перегруженные имена подпрограмм). Кроме того, в сборках .NET имеются явные определения пространств имен.

Область действия идентификатора (т.е. место, где он может быть использован) простирается от момента описания до конца блока, в котором он описан. Область действия глобального идентификатора, описанного в модуле, простирается на весь модуль, а также на основную программу, к которой данный модуль подключен в разделе **uses.**

Кроме этого, имеются переменные, определенные в блоке и связанные с некоторыми конструкциями (**for**, **foreach**). В этом случае действие переменной i простирается до конца соответствующей конструкции. Так, следующий код корректен:

```
var a: array of integer := (3,5,7);
for i: integer := 1 to 9 do
  write(a[i]);
foreach i: integer in a do
  write(i);
```

Идентификатор с тем же именем, определенный во вложенном пространстве имен, *скрывает* идентификатор, определенный во внешнем пространстве имен. Например, в коде

```
var i: integer;
procedure p;
var i: integer;
begin
  i := 5;
end;
```

значение 5 будет присвоено переменной i, описанной в процедуре p; внутри же процедуры p сослаться на глобальную переменную i невозможно.

Переменные, описанные внутри блока, не могут иметь те же имена, что и переменные из раздела описаний этого блока. Например, следующая программа ошибочна:

```
var i: integer;
begin
  var i: integer; // ошибка
end.
```

В производных классах, напротив, можно определять члены с теми же именами, что и в базовых классах, при этом их имена скрывают соответствующие имена в базовых классах. Для обращения к одноименному члену базового класса из метода производного класса используется ключевое слово inherited:

```
type
  A=class
    i: integer;
    procedure p;
    begin
      i := 5;
    end;
  end;
  B=class(A)
```

```
    i: integer;
    procedure p;
    begin
      i := 5;
      inherited p;
    end;
  end;
```

Алгоритм поиска имени в классе следующий: вначале имя ищется в текущем классе, затем в его базовых классах, а если не найдено, то в глобальной области видимости.

Алгоритм поиска имени в глобальной области видимости при наличии нескольких подключенных модулей следующий: вначале имя ищется в текущем модуле, затем, если не найдено, по цепочке подключенных модулей в порядке справа налево. Например, в программе

```
uses unit1,unit2;
begin
  id := 2;
end.
```

описание переменной id будет искаться вначале в основной программе, затем в модуле unit2, затем в модуле unit1. При этом в разных модулях могут быть описаны разные переменные id. Данная ситуация означает, что unit1 образует внешнее пространство имен, пространство имен unit2 в него непосредственно вложено, а пространство имен основной программы вложено в unit2.

Если в последнем примере оба модуля - unit1 и unit2 - определяют переменные id, то рекомендуется уточнять имя переменной именем модуля, используя конструкцию*ИмяМодуля*.*Имя*:

```
uses unit1,unit2;
begin
  unit1.id := 2;
end.
```

Обзор типов

Типы в **PascalABC.NET** подразделяются на простые, структурированные,типы указателей, процедурные типы, последоват ельности и классы.

К ***простым*** относятся целые и вещественные типы, логический, с имвольный, перечислимый и диапазонный тип.

Тип данных называется ***структурированным***, если в одной переменной этого типа может содержаться множество значений.
К структурированным типам
относятся массивы, строки, записи, кортежи, множества, файлы и кла ссы.

Особым типом данных является последовательность, которая хранит по-существу алгоритм получения данных последовательности один за другим.

Все простые типы, кроме вещественного, называются ***порядковыми***. Только значения этих типов могут быть индексами статических массивов и параметрами цикла **for**. Кроме того, для порядковых типов используются функции Ord, Pred и Succ, а также процедуры Inc и Dec.

Все типы, кроме типов указателей, являются производными от типа Object. Каждый тип в PascalABC.NET имеет отображение на тип .NET. Тип указателя принадлежит к неуправляемому коду и моделируется типом void*.

Все типы в PascalABC.NET подразделяются на две большие группы: *размерные* и *ссылочные*.

Выражения и операции: обзор

Выражение - это конструкция, возвращающая значение некоторого типа. Простыми выражениями являются переменные и константы. Более сложные выражения строятся из простых с помощью операций, вызовов функций и скобок. Данные, к которым применяются операции, называются *операндами*.

В PascalABC.NET имеются следующие
операции: @, **not**, ^, *, /, **div**, **mod**, **and**, **shl**, **shr**, +, -
, **or**, **xor**, =, >, <, <>, <=, >=, **as**, **is**, **in**, а также операция new и операция приведения типа. Операции @, -, +, ^, **not**, операция приведения типа и операция new являются унарными (имеют один операнд), остальные являются бинарными (имеют два операнда), операции + и -являются и бинарными и унарными.
Справка по операциям PascalABC.NET

- Арифметические операции
- Логические операции
- Операции сравнения
- Строковые операции
- Побитовые операции

- Операции с множествами
- Операция явного приведения типов
- Операции is и as
- Операция new
- Операция @ получения адреса
- Операции с указателями
- Операции typeof и sizeof

Порядок выполенения операций определяется их приоритетом. В языке PascalABC.NET имеется четыре уровня приоритетов операций, задаваемых таблицей приоритетов.

Таблица приоритетов операций

Ряд операций для определяемых пользователем типов можно перегружать.

Операторы: обзор

В PascalABC.NET определены следующие операторы.

- Операторы присваивания
- Составной оператор
- Оператор описания переменной
- Оператор цикла for
- Оператор цикла foreach
- Операторы цикла while и repeat
- Условный оператор if
- Оператор выбора варианта case
- Оператор вызова процедуры
- Оператор try except
- Оператор try finally
- Оператор raise
- Операторы break, continue и exit
- Оператор goto
- Оператор lock
- Оператор with
- Пустой оператор

Структура модуля

Модули предназначены для разбиения текста программы на несколько файлов. В модулях описываются переменные, константы, типы, классы, процедуры и функции. Для того чтобы эти объекты можно было использовать в вызывающем модуле (которым может быть и основная программа), следует указать имя файла модуля (без

расширения .pas) в разделе **uses**вызывающего модуля. Файл модуля (.pas) или откомпилированный файл модуля (.pcu) должен находиться либо в том же каталоге, что и основная программа, либо в подкаталоге **Lib** системного каталога программы PascalABC.NET.

Модуль имеет следующую структуру:

unit *имя модуля*;
interface
раздел интерфейса
implementation
раздел реализации
initialization
раздел инициализации
finalization
раздел финализации
end.

Имеется также упрощенный синтаксис модулей без разделов интерфейса и реализации.

Первая строка обязательна и называется ***заголовком модуля***. Имя модуля должно совпадать с именем файла.

Раздел интерфейса и раздел реализации модуля могут начинаться с раздела uses подключения внешних модулей и пространств имен .NET. Имена в двух разделах **uses** не должны пересекаться.

Раздел интерфейса включает объявление всех имен, которые экспортируются данным модулем в другие модули (при подключении его в разделе **uses**). Это могут быть константы, переменные, процедуры, функции, классы, интерфейсы. Реализация методов классов может быть дана прямо в разделе интерфейса, но это не рекомендуется.

Раздел реализации содержит реализацию всех процедур, функций и методов, объявленных в разделе интерфейса. Кроме этого, в разделе реализации могут быть описания внутренних имен, которые не видны вне модуля и используются лишь как вспомогательные.

Раздел инициализации и *раздел финализации* представляют собой последовательность операторов, разделяемых символом ;. Операторы из раздела инициализации модуля выполняются до начала основной программы, операторы из раздела финализации

модуля - после окончания основной программы. Порядок выполнения разделов инициализации и разделов финализации подключенных модулей непредсказуем. Как раздел инициализации, так и раздел финализации могут отсутствовать.

Вместо разделов инициализации и финализации может присутствовать только раздел инициализации в виде

```
begin
  последовательность операторов
end.
```

Например:

```
unit Lib;
interface
uses GraphABC;
const Dim = 5;
var Colors: array [1..Dim] of integer;
function RandomColor: integer;
procedure FillByRandomColor;
implementation
function RandomColor: integer;
begin
  Result := RGB(Random(255),Random(255),Random(255));
end;
procedure FillByRandomColor;
begin
  for i: integer := 1 to Dim do
    Colors[i] := RandomColor;
end;
initialization
  FillByRandomColor;
end.
```

Циклические ссылки между модулями возможны при определенных ограничениях.

Обзор классов и объектов

Описание классов

Класс представляет собой составной тип, состоящий из полей (переменных), методов (процедур и функций) и свойств. Описание класса имеет вид:

```
type
  имя класса = class
    секция1
    секция2
      ...
  end;
```

Каждая секция имеет вид:

модификатор доступа
описания полей
объявления или описания методов и описания свойств

Модификатор доступа в первой секции может отсутствовать, при этом подразумевается модификатор **internal** (видимость всюду внутри сборки).

Методы могут описываться как внутри, так и вне класса. При описании метода внутри класса его имя предваряется именем класса с последующей точкой. Например:

```
type
  Person = class
  private
    fName: string;
    fAge: integer;
  public
    constructor Create(Name: string; Age: integer);
    begin
      fName := Name;
      fAge := Age;
    end;
    procedure Print;
    property Name: string read fName;
    property Age: integer read fAge;
  end;

procedure Person.Print;
begin
  writelnFormat('Имя: {0} Возраст: {1}', Name, Age);
end;
```

После слова **class** в скобках может быть указано имя класса-предка), а также через запятую список поддерживаемых интерфейсов.

Перед словом **class** может быть указано ключевое слово **sealed** – в этом случае от класса запрещено наследовать.

Все описания и объявления внутри класса образуют ***тело класса***. Поля и методы образуют ***интерфейс*** класса..

Классы могут описываться только на глобальном уровне. Локальные определения классов (т.е. определения в разделе описания подпрограмм) запрещены.

Переменные типа класс

В языке PascalABC.NET классы являются ссылочными типами. Это значит, что переменная типа класс хранит в действительности ссылку на объект.

Переменные типа класс называются *объектами* или *экземплярами класса*. Они инициализируются вызовом конструктора класса - специального метода, выделяющего память под объект класса и инициализирующего его поля:

var p: Person := new Person('Иванов',20);

После инициализации через переменную типа класс можно обращаться к публичным членам класса (полям, методам, свойствам), используя точечную нотацию:

Print(p.Name,p.Age);
p.Print;

Вывод переменной типа класс

По умолчанию процедура write для переменной типа класс выводит содержимое её публичных полей и свойств в круглых скобках через запятую:

write(p); // Иванов 20

Чтобы изменить это поведение, в классе следует переопределить виртуальный метод ToString класса Object - в этом случае именно он будет вызываться при выводе объекта.

Например:

```
type
  Person = class
    ...
    function ToString: string; override;
```

```
    begin
      Result := string.Format('Имя: {0}  Возраст: {1}', Name, Age);
    end;
  end;
  ...
var p: Person := new Person('Иванов',20);
writeln(p); // Имя: Иванов  Возраст: 20
```

Присваивание и передача в качестве параметров подпрограмм

Переменная типа класс является ссылкой и хранит ссылку на объект, создаваемый вызовом конструктора.

Как ссылка переменная типа класс может хранить значение **nil**:

```
p := nil;
...
if p = nil then ...
```

При присваивании переменных типа класс копируется только ссылка. После присваивания обе переменные типа класс будут ссылаться на один объект и совместно модифицировать его:

```
var p1,p2: Person;
...
p1 := new Person('Петров',20);
p2 := p1;
p1.IncAge;
p2.Print; // Имя: Петров  Возраст: 21
```

Сравнение на равенство

При сравнении переменных типа класс на равенство сравниваются ссылки, а не значения.

```
var p1 := new Person('Петров',20);
var p2 := new Person('Петров',20);
writeln(p1=p2); // False
p2 := p1;
writeln(p1=p2); // True
```

Это поведение можно изменить, перегрузив операцию = для класса.

Интерфейсы: обзор

Интерфейс - это тип данных, содержащий набор заголовков методов и свойств, предназначенных для реализации некоторым классом. Интерфейсы описываются в разделе **type**следующим образом:

ИмяИнтерфейса = **interface**
  объявления методов и свойств
end;

Для метода приводится только заголовок, для свойства после возвращаемого типа указываются необходимые модификаторы доступа read и write.

Например:

```
type
  IShape = interface
    procedure Draw;
    property X: integer read;
    property Y: integer read;
  end;
  ICloneable = interface
    function Clone: Object;
  end;
```

Поля и статические методы не могут входить в интерфейс.

Класс реализует интерфейс, если он реализует все методы и свойства интерфейса в **public**-секции. Если класс не реализует хотя бы один метод или свойство интерфейса, возникает ошибка компиляции. Класс может реализовывать также несколько интерфейсов. Список реализуемых интерфейсов указывается в скобках после ключевого слова **class** (если указано имя предка, то после имени предка).

Например:

```
type
  Point = class(IShape,ICloneable)
  private
    xx,yy: integer;
  public
    constructor Create(x,y: integer);
    begin
      xx := x; yy := y;
    end;
    procedure Draw; begin end;
    property X: integer read xx;
    property Y: integer read yy;
    function Clone: Object;
```

```
    begin
      Result := new Point(xx,yy);
    end;
    procedure Print;
    begin
      write(xx,' ',yy);
    end;
  end;
```

Интерфейсы можно наследовать друг от друга:

```
type
  IPosition = interface
    property X: integer read;
    property Y: integer read;
  end;
  IDrawable = interface
    procedure Draw;
  end;
  IShape = interface(IPosition,IDrawable)
  end;
```

Интерфейс по-существу представляет собой абстрактный класс без реализации входящих в него методов. Для интерфейсов, в частности, применимы все правила приведения типов объектов: тип объекта, реализующего интерфейс, может быть неявно приведен к типу интерфейса, а обратное преобразование производится только явно и может вызвать исключение при невозможности преобразования:

```
var ip: IShape := new Point(20,30);
ip.Draw;
Point(ip).Print;
```

Все методы класса, реализующего интерфейс, являются виртуальными без использования ключевых слов **virtual** или **override**. В частности, ip.Draw вызовет метод Draw классаPoint. Однако, цепочка виртуальности таких методов обрывается. Чтобы продолжить цепочку виртуальности методов, реализующих интерфейс, в подклассах, следует использовать ключевое слово **virtual**:

```
type
  Point = class(IShape,ICloneable)
```

```
    ...
    function Clone: Object; virtual;
    begin
      Result := new Point(xx,yy);
    end;
  end;
```

Для интерфейсов, как и для классов, можно также использовать операции **is** и **as**:

```
if ip is Point then
  ...
var p: Point := ip as Point;
if p<>nil then
  writeln('Преобразование успешно');
```

Далее следует блок **begin**/**end**, внутри которого находятся операторы, отделяемые один от другого символом "точка с запятой". Среди операторов может присутствовать оператор описания переменной, который позволяет описывать переменные внутри блока.

Раздел **uses** и раздел описаний могут отсутствовать.

Например:

```
        program MyProgram;
var
  a,b: integer;
  x: real;
begin
  readln(a,b);
  x := a/b;
  writeln(x);
end.
```

или

```
        uses GraphABC;
begin
  var x := 100;
  var y := 100;
  var r := 50;
  Circle(x,y,r);
end.
```

Задания для практического выполнения

Используя основные конструкции языка PascalABC**.**NET на базе платформ Microsoft .NET Framework + PascalABC.NET реализовать следующие приложения (в отчете представить программный код и результаты работы программы)

1. Поиск минимума из двух значений

```
var
  x,y: integer;
  min: integer;
begin
  write('Введите x и y: ');
  readln(x,y);
  if x<y then
    min := x
  else min := y;
  writeln('Минимум = ',min);
end.
```

Определение четности числа

```
var x: integer;
begin
  write('Введите x: ');
  readln(x);
  if x mod 2 = 0 then
    writeln('Это четное число')
  else writeln('Это нечетное число');
end.
```

Состоит ли двузначное число из одинаковых цифр

Код на PascalABC.NET:

```
var x: integer;
begin
  write('Введите двузначное число: ');
  readln(x);
```

```
  var c1 := x div 10;
  var c2 := x mod 10;
  if c1=c2 then
    writeln('Цифры числа совпадают')
  else writeln('Цифры числа не совпадают');
end.
```

2. Упорядочение двух значений по возрастанию

```
var
  x,y: integer;
  v: integer;
begin
  write('Введите x,y: ');
  readln(x,y);
  if x>y then
  begin
    v := x;
    x := y;
    y := v
  end;
  writeln('Результат упорядочения по возрастанию: ',x,' ',y);
end.
```

Проверка числа на двузначность

```
var x: integer;
begin
  write('Введите x: ');
  readln(x);
  if (x>=10) and (x<=100) then
    writeln('Двузначное число')
  else writeln('Не двузначное число')
end.
```

3. Наименование сезона по номеру месяца

```
var
  Month: integer;
  Season: string;
begin
```

```
  write('Введите номер месяца: ');
  readln(Month);
  if (Month=1) or (Month=2) or (Month=12) then
    Season := 'Зима'
  else if (Month=3) or (Month=4) or (Month=5) then
    Season := 'Весна'
  else if (Month=6) or (Month=7) or (Month=8) then
    Season := 'Лето'
  else Season := 'Осень';
  writeln('Это ',Season)
end.
```

4. Определение четверти, в которой находится точка

```
var
  x,y: integer;    // Координаты точки
  Quater: integer; // Номер четверти
begin
  write('Введите координаты точки: ');
  readln(x,y);
  if x>0 then
    if y>0 then
      Quater := 1
    else Quater := 4
  else
    if y>0 then
      Quater := 2
    else Quater := 3;
  writeln('Номер четверти = ',Quater);
end.
```

5. Словесное наименование сезона по номеру месяца

```
var
  Month: integer;
  Season: string;
begin
  write('Введите номер месяца: ');
  readln(Month);
  case Month of
```

```
    1,2,12: Season := 'Зима';
    3..5:   Season := 'Весна';
    6..8:   Season := 'Лето';
    9..11:  Season := 'Осень';
    else    Season := 'Вы ввели неверный номер месяца';
  end;
  writeln('Это ',Season)
end.
```

6. Определение того, является ли символ цифрой или буквой

```
var
  Symbol: char;
begin
  write('Введите символ: ');
  readln(Symbol);
  case Symbol of
    'a'..'z': writeln('Это маленькая английская буква');
    'A'..'Z': writeln('Это большая английская буква');
    '0'..'9': writeln('Это цифра');
  end;
end.
```

7. Рисование мышью в графическом окне

Данная программа осуществляет рисование мышью в графическом окне:

```
uses GraphABC;

procedure MouseDown(x,y,mb: integer);
begin
  MoveTo(x,y);
end;

procedure MouseMove(x,y,mb: integer);
begin
  if mb=1 then LineTo(x,y);
end;
```

```
begin
  // Привязка обработчиков к событиям
  OnMouseDown := MouseDown;
  OnMouseMove := MouseMove
end.
```

8. Перемещение окна с помощью клавиатуры

Данная программа осуществляет перемещение графического окна с помощью клавиатуры:

```
uses GraphABC;

procedure KeyDown(Key: integer);
begin
   case Key of
  VK_Left:  Window.Left := Window.Left - 2;
  VK_Right: Window.Left := Window.Left + 2;
  VK_Up:    Window.Top := Window.Top - 2;
  VK_Down:  Window.Top := Window.Top + 2;
   end;
end;

begin
  // Привязка обработчиков к событиям
  OnKeyDown := KeyDown;
end.
```

Задание для самостоятельного выполнения

Написать программу нахождение корней квадратного уравнения

Отчет практической работе 3 должен содержать: цель работы; краткое описание использованных в работе элементов языка PascalABC.NET; код созданных программ и результаты их работы; выводы по результатам практической работы. При оформлении отчета в печатном виде в нижний колонтитул следует поместить фамилию, инициалы и номер группы обучаемого (8 пт., Arial, выравнивание по правому краю).

Практическая работа №4
Особенности различных сред разработки программного обеспечения информационных систем (Семинар)

Темы докладов (+ презентация) к семинару:

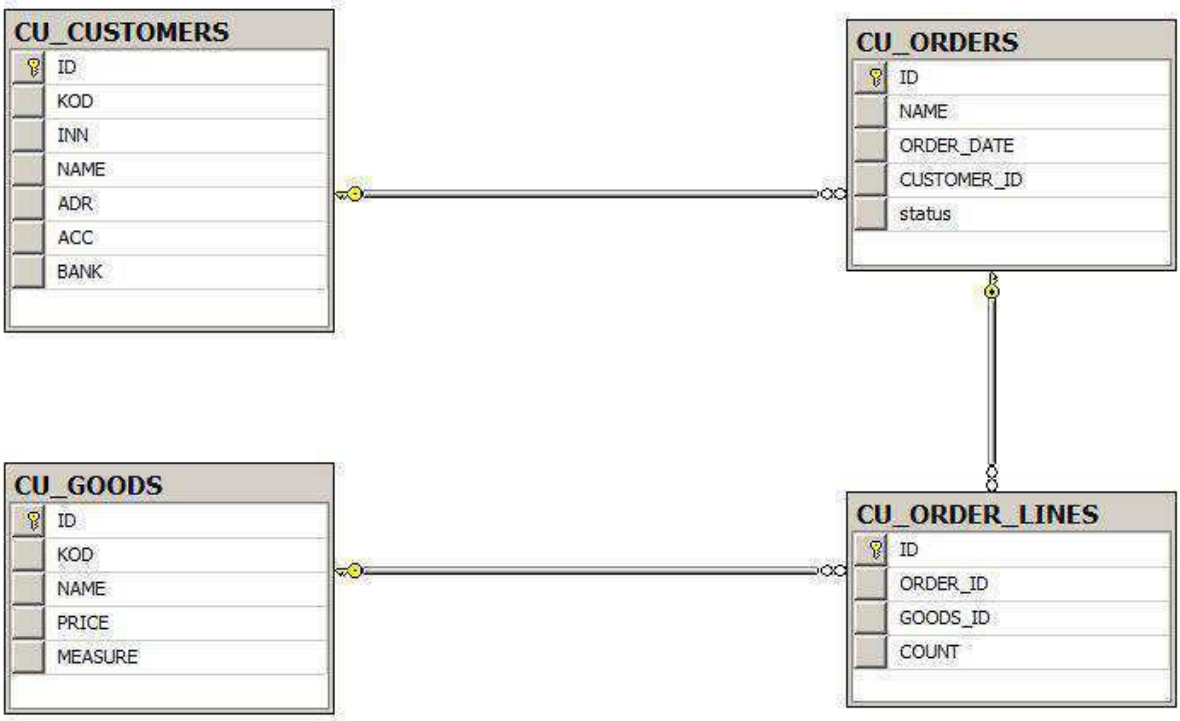
- Интегрированная среда разработки приложений NetBeans IDE
- Интегрированная среда разработки приложений Microsoft Visual Studio
- Интегрированная среда разработки приложений Eclipse
- Интегрированная среда разработки приложений SharpDevelop
- Интегрированная среда разработки приложений JetBrains Rider
- Интегрированная среда разработки приложений Monodevelop
- Интегрированная среда разработки приложений Android Studio
- Средство разработки бизнес-ориентированных приложений eXpressApp Framework (XAF) (компания Devexpress)
- Средство разработки бизнес-ориентированных приложений Xafari Framework
- Средство разработки многофункциональных интернет-приложений Microsoft Silverlight
- Средство построения клиентских приложений Windows WPF (Windows Presentation Foundation)
- Средство создания элементов пользовательского интерфейса для настольных и web-приложений Expression Design.
- Инструментальное средство веб- дизайна Expression Web

Практическая работа №5
Инструментальные средства создания программного обеспечения информационных систем управления предприятием (платформа RP-Server + Microsoft SQL Server)

Цель работы: приобретение навыков применения инструментальных средств разработки программного обеспечения информационных систем управления предприятием (на примере бесплатно распространяемой платформы RP-Server + Microsoft SQL Server)

Решаемая задача: автоматизация учета заказов компании.

При этом диаграмма схемы данных нашей базы данных должна быть следующей:

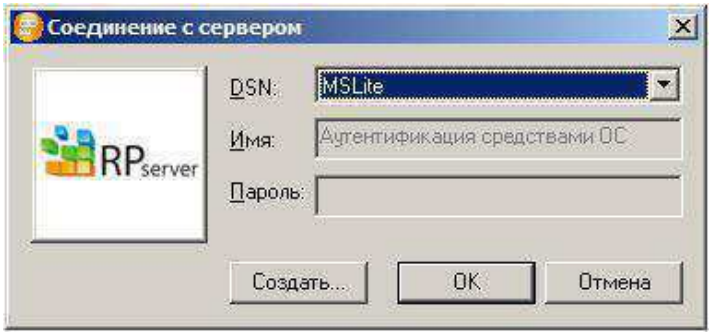


Порядок выполнения работы:

I. Создание новой базы данных.

Запускаем RP-Server (Desiner):

В появившемся диалоге нажимаем пункт «Создать»:

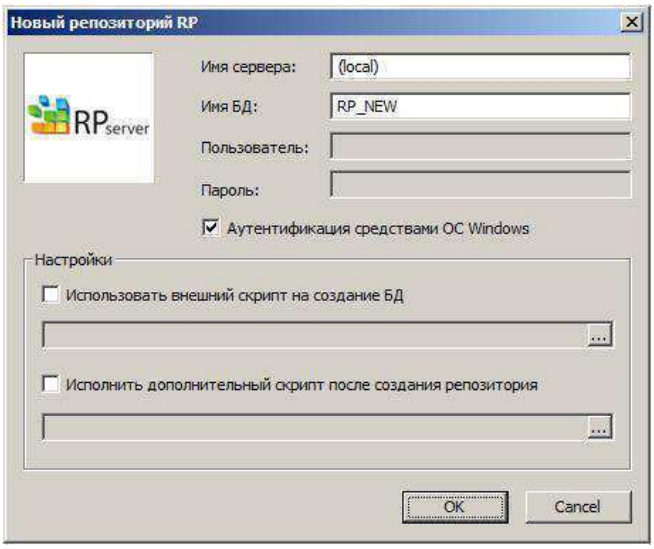


В открывшемся диалоге указываем имя нашего Microsoft SQL Server и имя базы данных, которую нужно создать. Остальные поля оставляем по умолчанию.

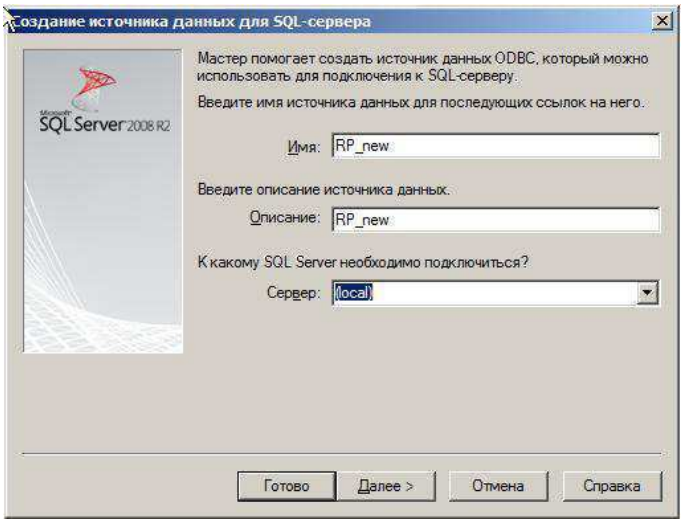
Если SQL Server установлен локально на том же компьютере, где запускается RP Designer, и только в одном экземпляре, то имя

сервера может быть указано как (local), либо должно быть прописано одно из имен доступных SQL серверов.

Нажимаем «OK».



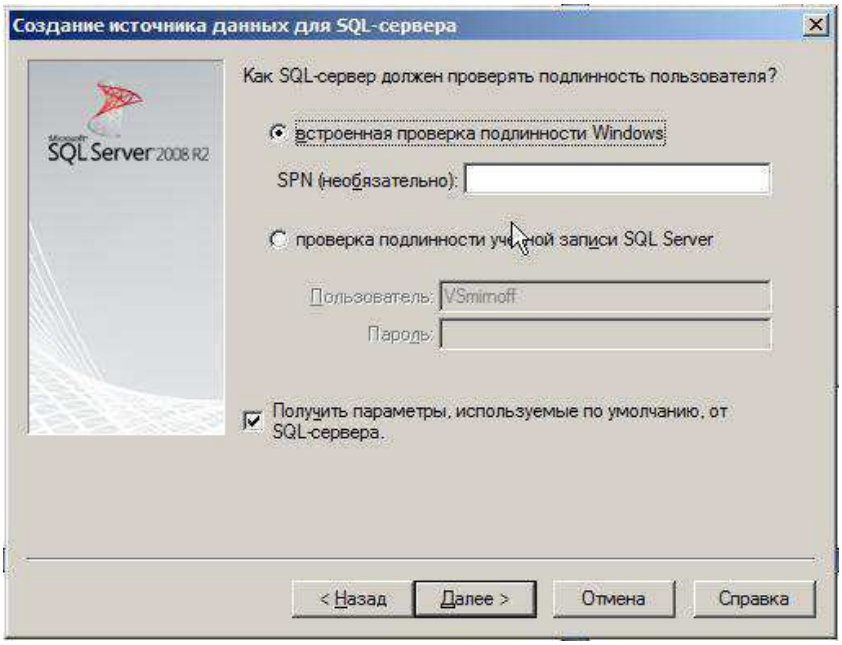
В следующем диалоге указываем название ODBC источника, через который мы будем работать с нашей базой данных.



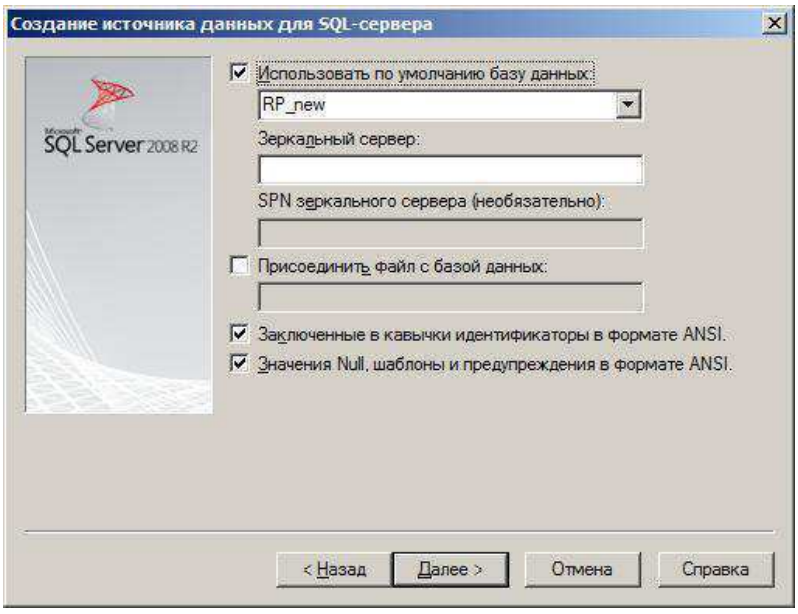
Имя ODBC источника может быть любым, в том числе может совпадать с именем базы данных.

Нажимаем «Далее».

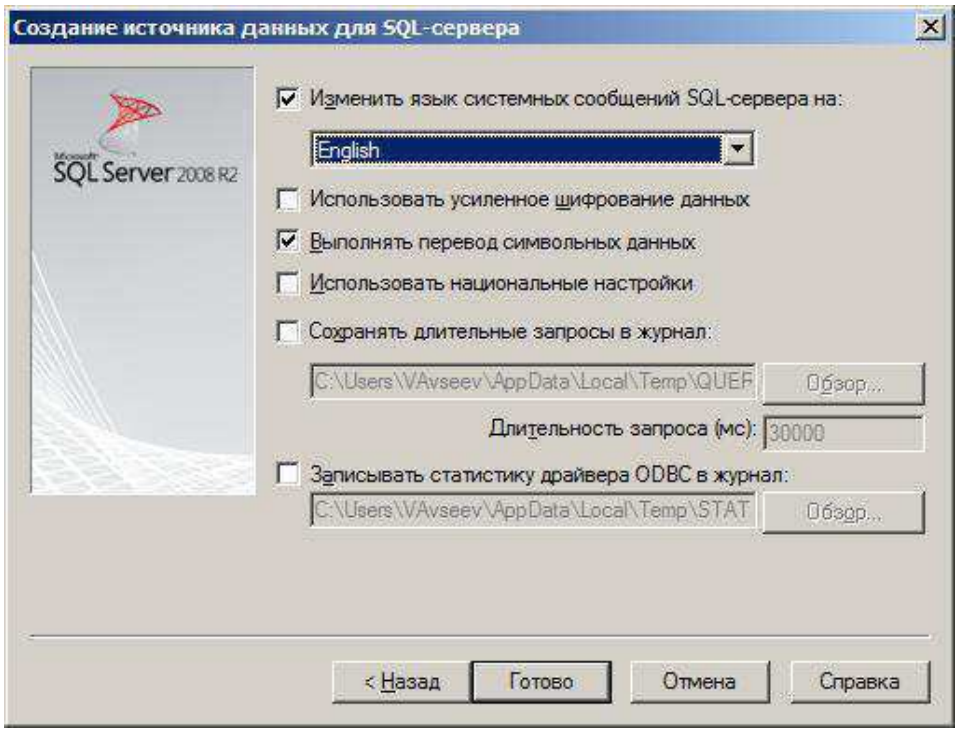
В следующем диалоге оставляем поля по умолчанию и нажимаем «Далее».



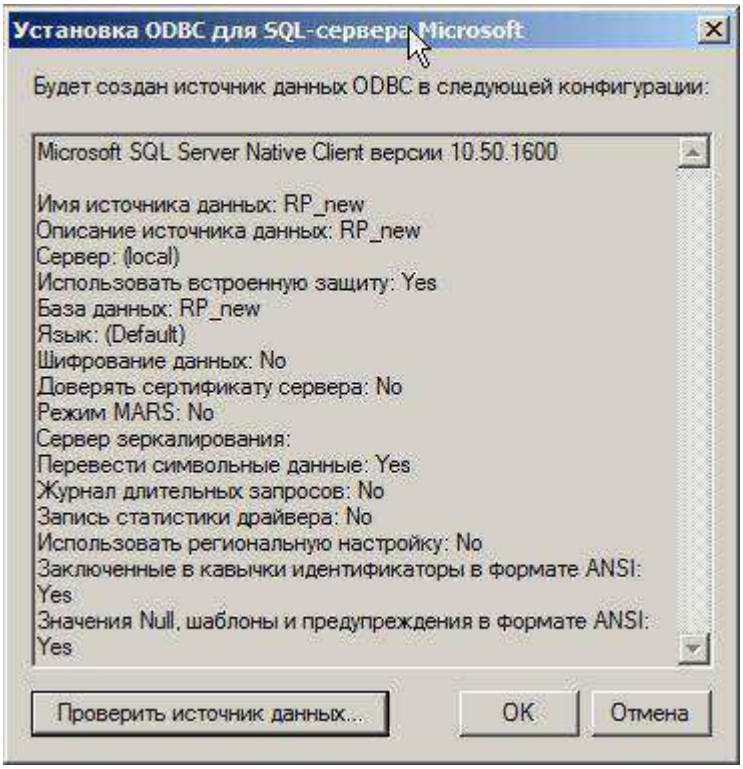
После этого выбираем нашу базу данных как базу по умолчанию и нажимаем «Далее».



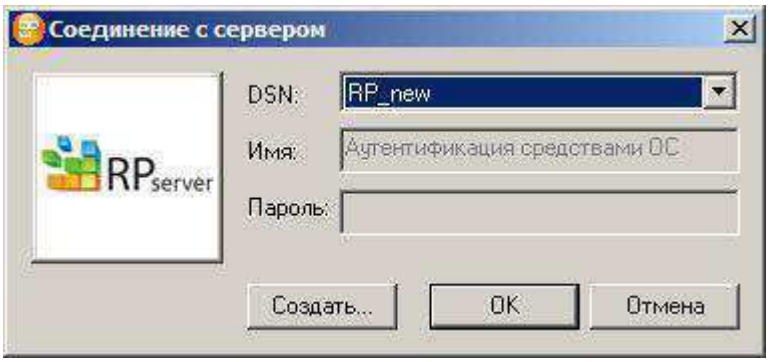
В этом диалоге можно указать, язык системных сообщений сервера. Лучше поднять флаг «Изменить язык…» и указать English. Нажимаем «Готово».



И, наконец, в последнем диалоге мастер настройки показывает нам все установленные параметры. Можно сразу нажать на «OK», можно проверить источник данных.



После этого программа снова предлагает диалог соединения с сервером, в котором теперь можно выбрать только что созданное нами ODBC соединение и нажать ОК:

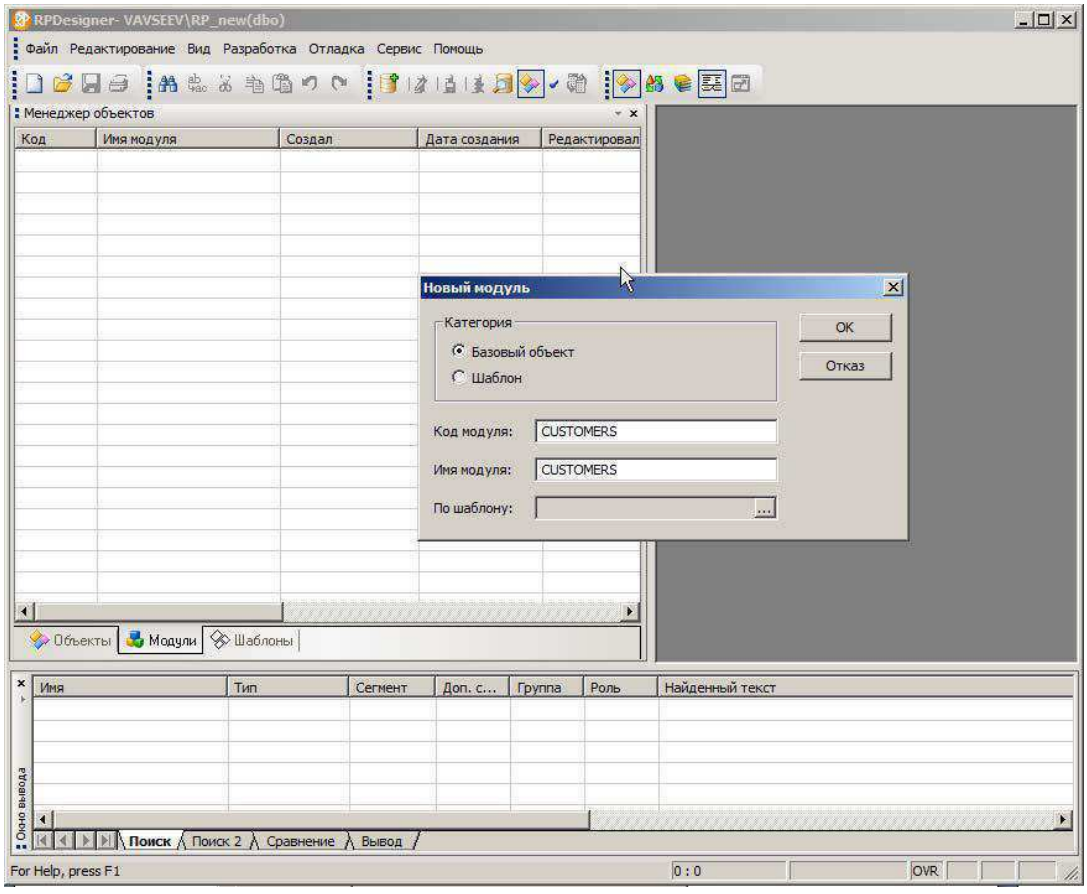


Откроется среда разработки RP Designer.

Создание конструктора модуля

Модулем называется любое приложение на RP платформе. Модуль – это совокупность объектов (процедур, списков, диалогов, отчетов и т.п.), предназначенных для выполнения какой- либо единой задачи создаваемой прикладной системы. Как правило, все эти объекты объединены одним общим меню пользователя.

В среде RP Designer выбираем закладку «Модули» и нажимаем <Ins> на клавиатуре или через правую кнопку меню. Раз мы собираемся автоматизировать заказы, то назовем модуль «CUSTOMERS».

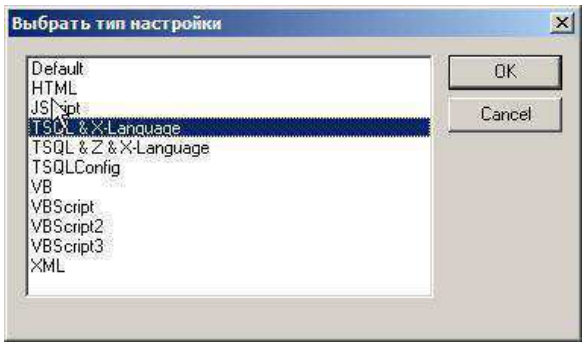


Теперь вызываем созданный модуль на редактирование и в секции «Конструктор» вписываем строку:

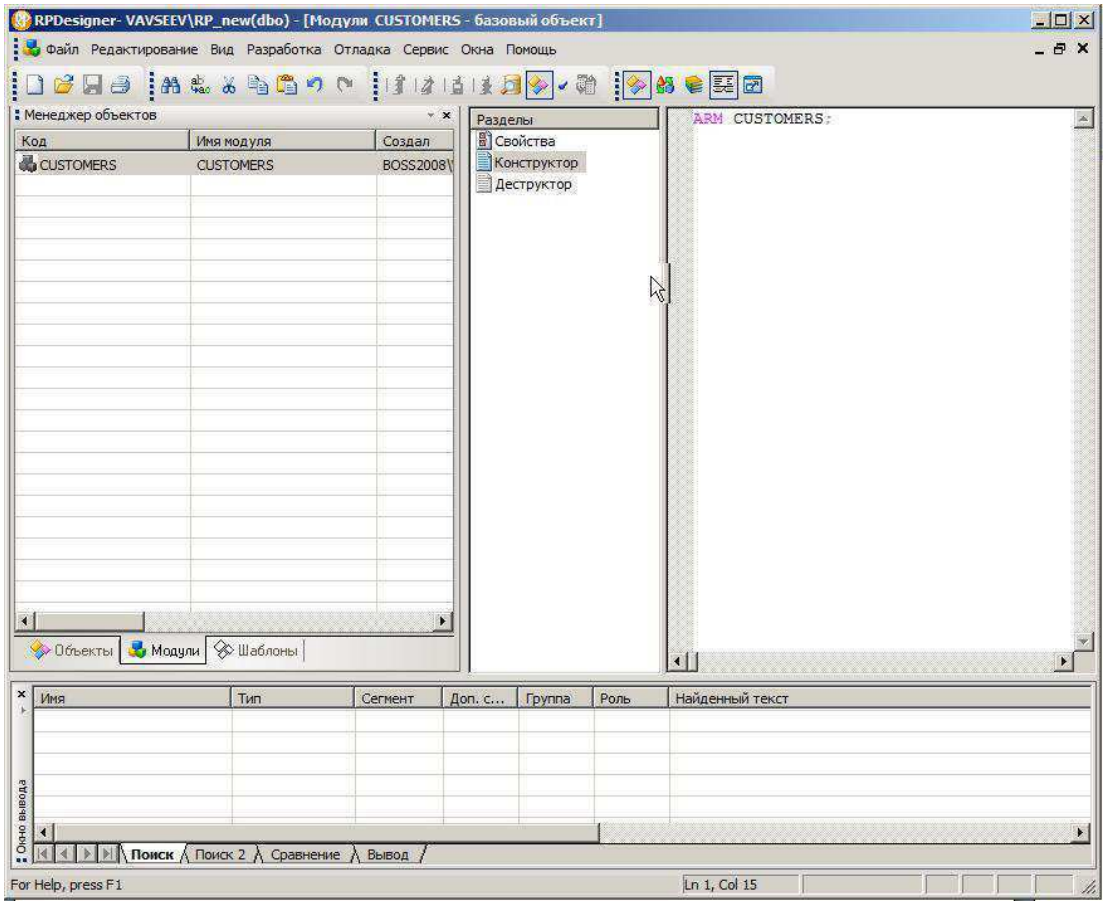
ARM CUSTOMERS;

На самом деле конструктор модуля – это специальная процедура, которая выполняется при запуске модуля на исполнение и она может содержать код любой степени сложности на внутреннем языке RP платформы (X- языке). Но в нашем случае мы всего лишь указали, что хотим вызывать меню модуля, которое называется «CUSTOMERS».

При входе в редактор конструктора модуля может появиться вот такой диалог с вопросом о том, какой тип настройки редактора нам требуется:



Здесь и далее можно смело выбирать вариант «TSQL & X- Language».

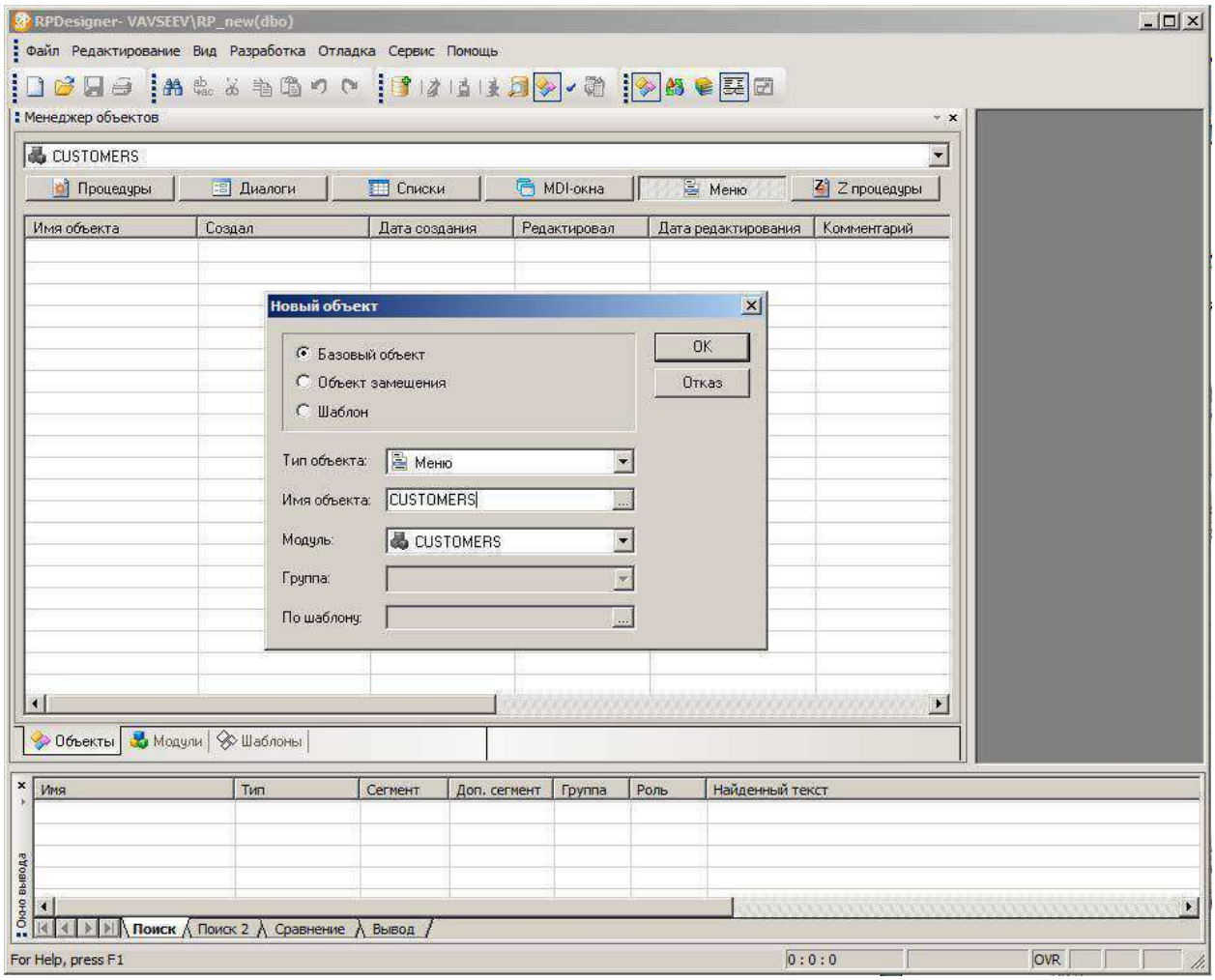


Сохраняем созданный модуль, нажав на иконку  .

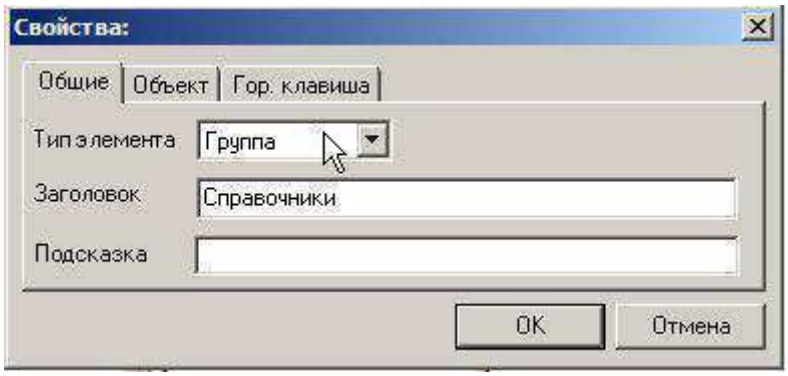
Создание меню

Переключаемся на закладку «Объекты» и выбираем тип объекта «Меню». В верхней части окна «Менеджер объектов» указываем, что текущий модуль – CUSTOMERS. Нажимаем <Ins>.

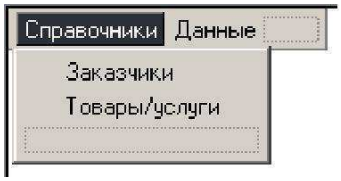
!!! Обратите внимание: Новый объект автоматически привязывается к тому модулю, который указан как текущий. Для удобства работы все создаваемые нами объекты лучше сразу привязывать к модулю CUSTOMERS. Хотя, это можно сделать и позже, например, по правой кнопке мыши в списке объектов выбрать пункт «Включить в модуль».



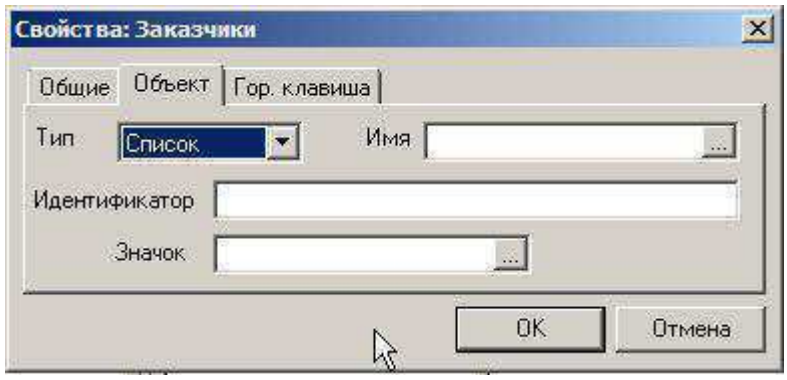
Создаем на верхнем уровне меню 2 пункта – «Справочники» и «Данные»



Теперь добавим выпадающие пункты меню, как показано на рисунках.

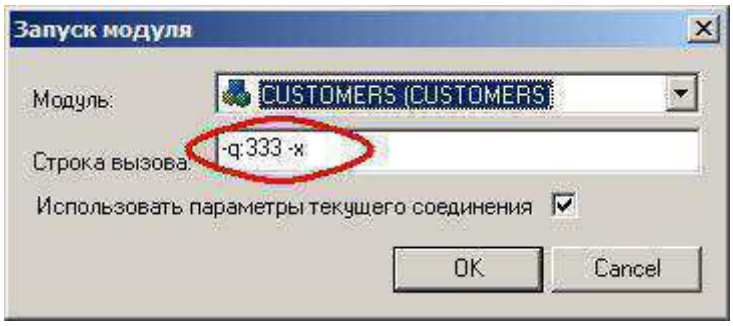


При создании выпадающего меню программа попросит указать тип элементов. Пока можно указать, что тип – «Список»:



Сохраняем созданное таким образом двухуровневое меню, нажав на иконку  .

С этого момента созданный модуль доступен для запуска из среды RP Designer с целью отладки. Нажимаем <F7> и говорим «ОК» в появившемся диалоге:



Обратите внимание на параметры в строке вызова. Их следует указать как отмечено красным кружком.

В запустившемся приложении мы увидим сформированное нами меню. Можно даже понажимать на его пункты. Но, как и следует ожидать, они пока пустые.

Выходим из запущенного приложения.

Создаем справочник «Заказчики»

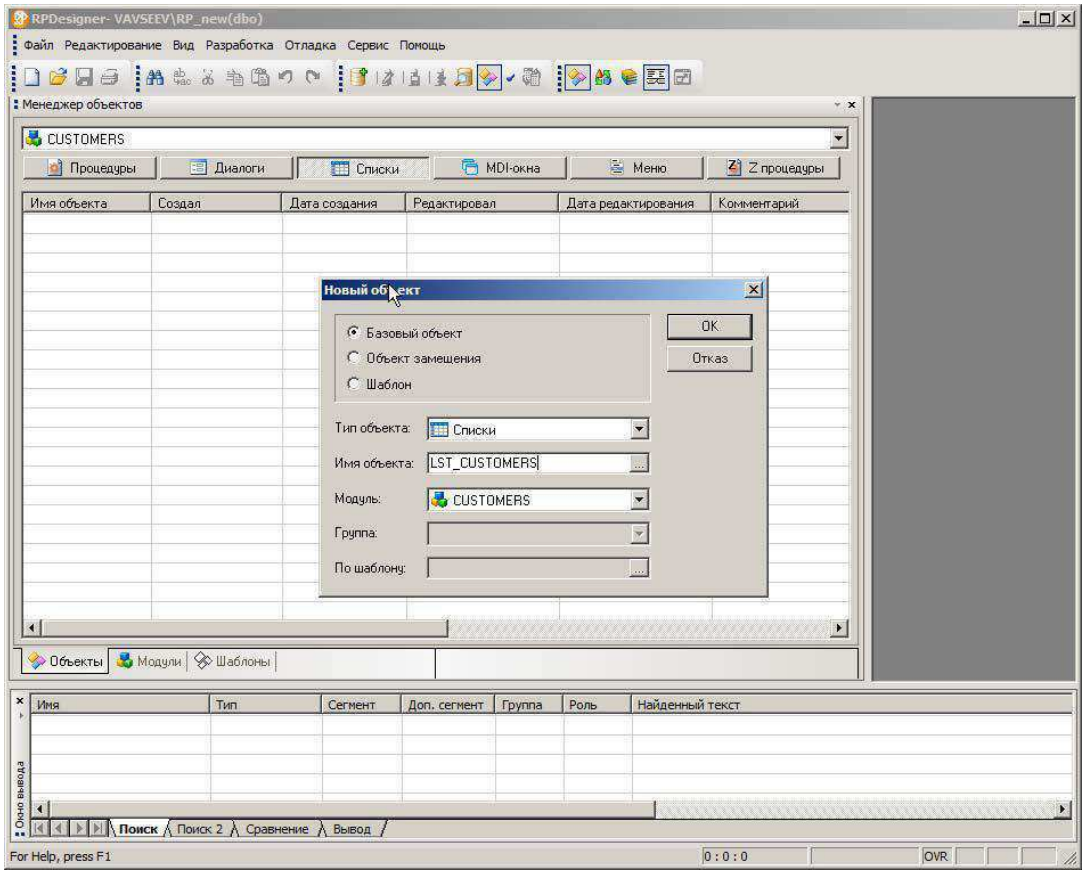
Для начала надо создать таблицу базы данных:

```
CREATE TABLE CU_CUSTOMERS
(
PRIMARY KEY (ID),                  - - первичный ключ

ID INTEGER IDENTITY(1,1),
KOD VARCHAR(25),                   - - код
INN VARCHAR(25),                   - - ИНН
NAME VARCHAR(255),                 - - наименование
ADR VARCHAR(255),                  - - адрес
ACC VARCHAR(25),                   - - счет в банке
BANK VARCHAR(255)                  - - банк

)
GO
GRANT ALL ON CU_CUSTOMERS TO public
GO
```

!!! Обратите внимание: Таблицу в БД можно создать несколькими способами, в том числе используя соответствующие интерфейсы SQL Server Management Studio (в составе утилит Microsoft SQL Server). Так же создавать таблицы можно и не выходя из RP Designer, например, в режиме отладки приложения. Для этого нужно запустить любое приложение, а пока оно у нас единственное (клавиша <F7>, см. предыдущий раздел) и нажать комбинацию клавиш SHIFT- F1. При этом появляется дополнительное окно отладки, предназначенное в том числе для выполнения любого скрипта. В это окно просто переносим (вставляем) указанный выше код на создание таблицы CU_CUSTOMERS, выделяем его целиком и выполняем по клавише <F8>. Теперь можем закрыть и это окно отладки, и запущенное приложение.

В БД появилась соответствующая таблица.

Переключимся в менеджере объектов среды RP Designer на закладку «Списки» и добавим по клавише <Ins> список, который назовем, например, «LST_CUSTOMERS».



В секции «SQL-запрос» создаваемого списка напишем вот такой несложный оператор SELECT, позволяющий извлечь данные из созданной только что таблицы. Лучше сразу дать этому запросу имя, например, @LST_CUSTOMERS, чтобы потом можно было свободно обращаться к этому «поименованному» запросу (строго говоря, конечно, это имя не запроса, как такового, а того буфера, куда попадут данные, возвращаемые этим запросом).

```
@LST_CUSTOMERS

SELECT id, kod[]"Код", inn[]"ИНН", name[]"Наименование",
adr[]"Адрес", acc[]"Счет", bank[]"Банк"
 FROM CU_CUSTOMERS
ORDER BY kod;
```

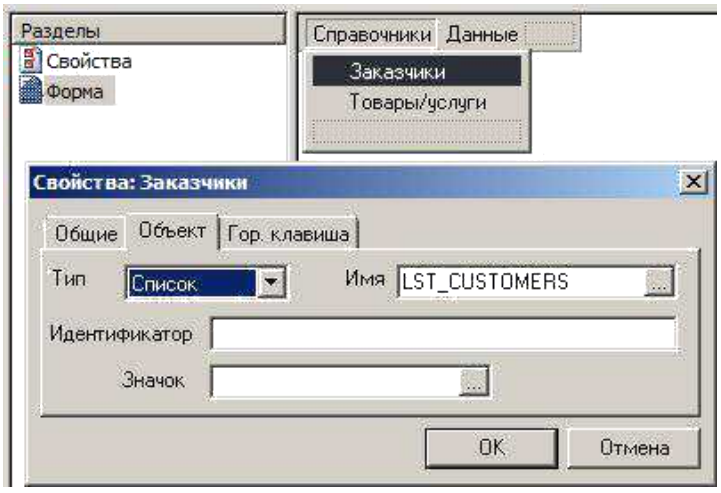
Сохраним созданный список, нажав на иконку  .

!!! Обратите внимание: Если в данный момент запустить наше приложение (по клавише <F7>), то непосредственно в среде RP Designer автоматически становятся доступны средства отладки. Например, выделив введенный SQL запрос (непосредственно в секции «SQL- запрос» списка) и нажав на клавишу F5 мы получим автоматически сформированный список с содержимым таблицы CU_CUSTOMERS или сообщение об ошибке, если запрос написан не правильно.

Теперь нужно связать созданный список с пунктом меню.

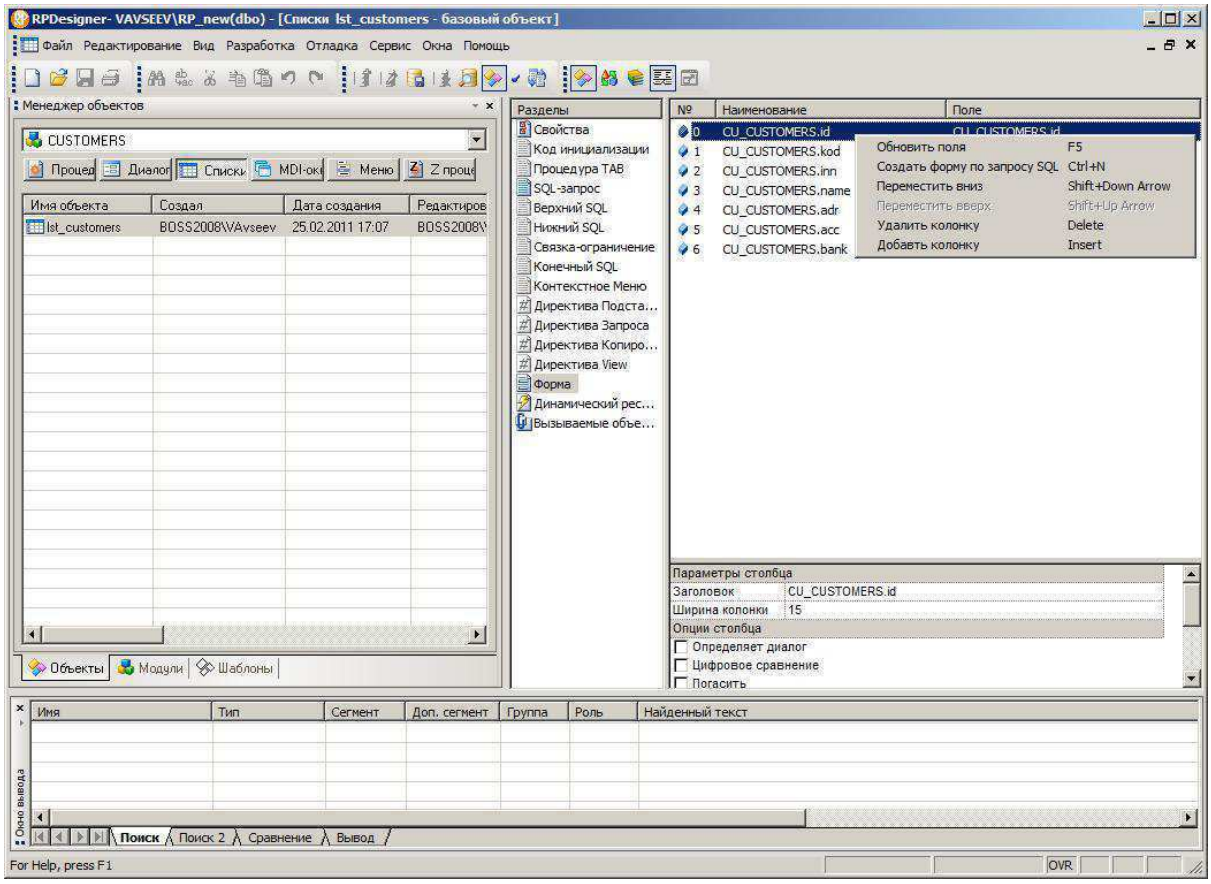
На закладке «Меню» выбираем наше меню и вызываем его на редактирование.

В секции диалога редактирования меню «Форма» выбираем пункт меню «Заказчики» и привязываем объект список LST_CUSTOMERS:



Сохраняем изменения в меню, нажав на иконку  .

По клавише <F7> проверим работу созданного объекта. Он должен вызываться при нажатии на пункт меню «Заказчики». Обратите внимание, что все столбцы (кроме ID) имеют русские названия. Это произошло потому, что мы их прописали непосредственно в SQL запросе списка. А вот столбец ID следует просто погасить. Сделать это можно так: в диалоге редактирования списка заходим в секцию «Форма» (приложение при этом запущено), встаем на список полей и нажимаем Ctrl- N. Встав на первое поле ID внизу в его свойствах, поднимаем флажок «Погасить».



Сохраняем изменения в меню, нажав на иконку  .

Сразу же после этого в окне нашего приложения (даже не закрывая его) можно заново запустить список и убедиться, что в нем отразилось внесенное изменение. Поле ID погашено.

Создаем справочник «Товары/услуги»

Создадим таблицу товаров:

```
CREATE TABLE CU_GOODS
(
PRIMARY KEY (ID),                    - - первичный ключ
ID INTEGER IDENTITY(1,1),
KOD VARCHAR(25),                     - - код
NAME VARCHAR(255),                   - - наименование
PRICE NUMERIC(19,2),                 - - цена
MEASURE VARCHAR(25)                  - - единица измерения
)
GO

GRANT ALL ON CU_GOODS TO public
GO
```

В менеджере объектов среды RP Designer выбираем закладку «Списки» и добавим список
«LST_GOODS».

В секции «SQL-запрос» этого списка напишем вот такой поименованный оператор SELECT, позволяющий извлечь данные из таблицы Товары/услуги:

```
@LST_GOODS

SELECT id, kod[]"Код", name[]"Наименование", PRICE[]"Цена", MEASURE[]"Е/и"
 FROM CU_GOODS
ORDER BY kod;
```

Аналогично предыдущему пункту подключим созданный список к пункту меню «Товары/услуги», а также создадим список полей этого списка и для первого поля ID отметим «Погасить».

Проверим его работу в приложении.

Создаем список «Заказы»

Создадим таблицу товаров:

```
CREATE TABLE CU_ORDERS
(
PRIMARY KEY (ID),                    - - первичный ключ

ID INTEGER IDENTITY(1,1),
NAME VARCHAR(255),
ORDER_DATE datetime,                 - - наименование
CUSTOMER_ID                          - - дата заказа
INTEGER,                             - - ссылка на покупателя
status INTEGER
                                     - - статус заказа

)
GO
```

```
GRANT ALL ON CU_ORDERS TO public
GO
```

На закладке «Списки» добавим список «LST_ORDERS».

Для секции TAB списка напишем такой код:

```
MSG 2041, @LST_ORDER_LINES;
```

Эта команда в будущем станет перерисовывать строки заказа при перемещении по списку заказов, когда мы реализуем список «Строки заказа».

В секции «SQL- запрос» списка напишем вот такой оператор SELECT:

```
@LST_ORDERS

SELECT CU_ORDERS.id,
CU_ORDERS.name[]"Наименование заказа",
ORDER_DATE[%x] "Дата",
CU_CUSTOMERS.name[]"Заказчик"
 FROM CU_ORDERS _HINTBROWSER JOIN
 CU_CUSTOMERS ON CU_CUSTOMERS.id =
CU_ORDERS.CUSTOMER_ID
 ORDER BY CU_CUSTOMERS.name, ORDER_DATE;
```

Подключим полученный список к пункту меню «Заказы». При этом не забудем создать перечень полей списка и для первого поля ID отметим «Погасить». Перезапустим приложение и проверим его работу.

Создаем список «Строки заказа», связанный со списком «Заказы»

Создадим таблицу строк заказа:

```
CREATE TABLE CU_ORDER_LINES
(
PRIMARY KEY (ID),                    - - первичный ключ

ID INTEGER IDENTITY(1,1),
ORDER_ID INTEGER,
GOODS_ID INTEGER,                    - - ссылка на заказ
COUNT NUMERIC(19,2)                  - - ссылка на
                                     товар/услугу

                                     - - количество

)
GO
GRANT ALL ON CU_ORDER_LINES TO public
GO
```

На закладке «Списки» добавим список «LST_ORDER_LINES».

В секции «SQL- запрос» напишем:

```
@LST_ORDER_LINES

SELECT CU_ORDER_LINES.ID,
            ORDER_ID,
            CU_GOODS.NAME[]"Товар",
            CU_GOODS.PRICE[%12.2f] "Цена",
            CU_GOODS.MEASURE[]"Ед. Изм. ",
            COUNT[%12.2f] "Количество",
            CU_GOODS.PRICE * COUNT AS total[%12.2f]
      "ВСЕГО"
 FROM CU_ORDER_LINES, CU_GOODS
WHERE CU_ORDER_LINES.GOODS_ID = CU_GOODS.id
 AND ORDER_ID = @LST_ORDERS:ID
ORDER BY ID;
```

В квадратных скобках указаны маски полей ввода. В данном случае это ввод дробного числа с двумя знаками после запятой.

Не забудем создать перечень полей списка и для полей ID и ORDER_ID отметим признак «Погасить».

Мы предполагаем, что список строк заказа должен работать совместно со списком заказов и быть подчиненным по отношению к нему. Для этого в запросе строк заказа имеется ссылка на текущую строку списка заказов ORDER_ID = @LST_ORDERS:ID. Поэтому чтобы наш запрос списка срок заказа отработал, а соответственно, чтобы получить корректный список его полей в секции «Форма», запустим наше приложение и откроем в нем уже имеющийся список «Заказы». Тогда RP Designer корректно обработает ссылку на текущую строку этого списка @LST_ORDERS:ID.

Создаем MDI окно «Заказы+строки».

В менеджере объектов среды RP Designer выбираем закладку «MDI- окна» и добавляем объект – MDI окно под названием «ORDERS_MDI».

В секции «Список объектов» вписываем вот такой код:

```
BROWSER LST_ORDERS;
BROWSER LST_ORDER_LINES;
{- 1, 2, 1, 1}
```

Что означает этот код?

Мы перечислили объекты, которые должны быть отображены в MDI окне (в данном случае – два уже известных нам списка – заказы и их строки). Параметры в фигурных скобках определяют лишь как списки должны быть взаимно ориентированы относительно друг друга.

Сохраняем объект.

Подключим созданное MDI окно к пункту меню «Заказы+строки»

Через клавишу <F7> традиционно можно проверить работоспособность созданного объекта, т.е. посмотреть как запускается созданное MDI окно.

То, что мы сделали, это пока еще не законченное приложение. Мы создали списки просмотра. И если бы в базе данных была информация, то мы могли бы ее видеть через эти списки. Причем у конечного пользователя автоматически доступны все сервисы поиска, фильтрации данных, индивидуальной настройки опций списков, выгрузки данных в популярные форматы Word, Excell и многое другое. Но нам необходима функциональность и для редактирования (добавления/удаления/изменения) данных. Для этого мы должны к спискам подключить диалоги редактирования.

Диалог «Заказчики»

В менеджере объектов среды RP Designer выбираем закладку «Диалоги» и создаем диалог

«DLG_CUSTOMERS».

В секции «SQL-запрос» прописываем запрос для диалога, также присвоив ему индивидуальное имя:

```
@DLG_CUSTOMERS
```

```
SELECT id, kod, inn, name, adr, acc, bank
 FROM CU_CUSTOMERS;
```

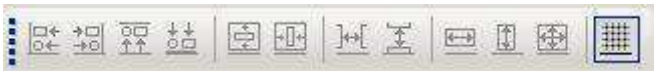
Далее переключаемся на секцию «Форма».

С помощью команды «Добавить поле редактирования» 

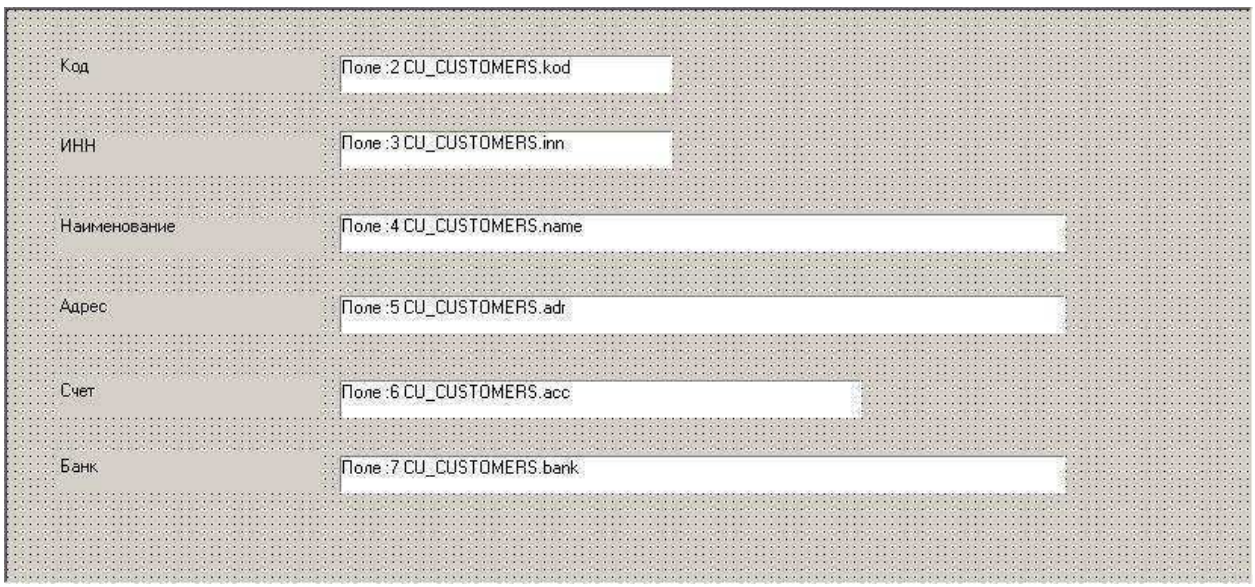
создаем на диалоге 6 полей.

!!! Обратите внимание: В этот момент в другом окне наше созданное приложение должно быть запущено по клавише <F7>.

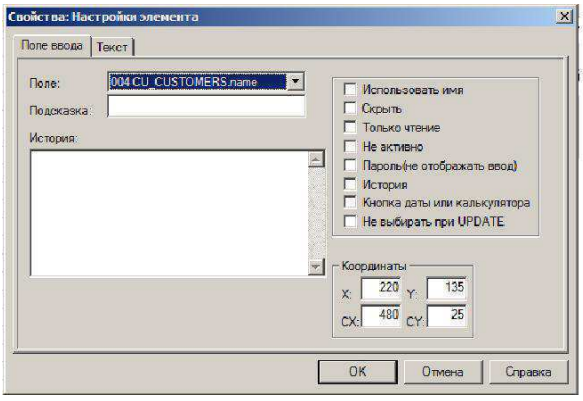
Поля можно разместить на экране, выровнять и определить их размер с помощью управляющих элементов в линейке инструментов «Layout»:

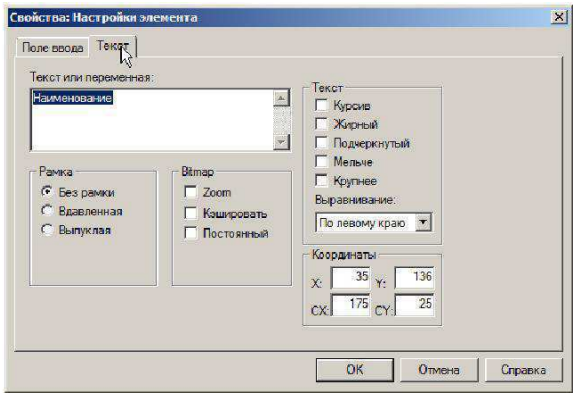


Поля диалога нужно привязать к полям запроса диалога и прописать для них соответствующие заголовки:



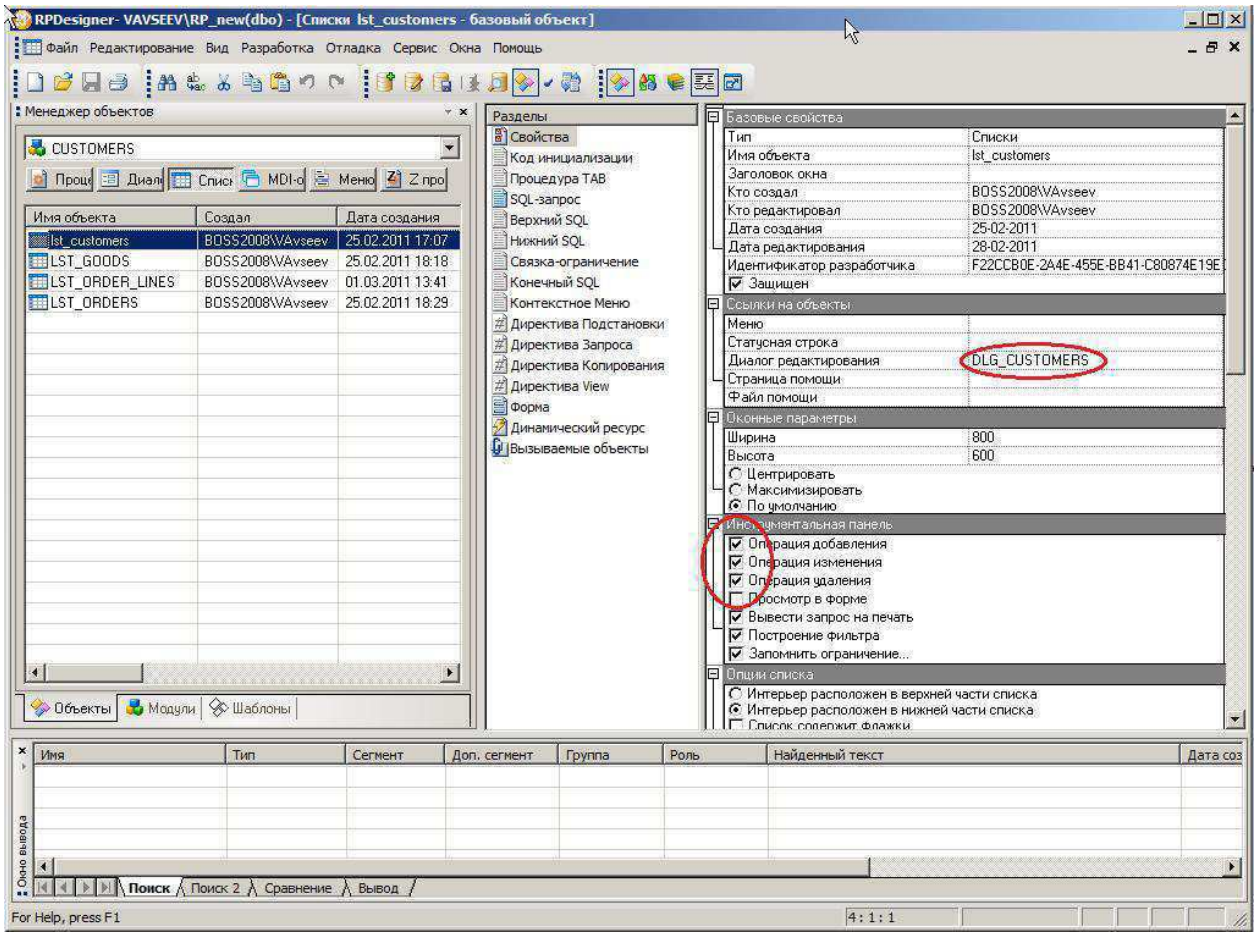
Для привязки полей диалога и изменения их свойств нужно дважды щелкнуть мышкой на поле:





Сохраняем диалог в базе данных -  .

Следующим шагом привязываем диалог к списку «LST_CUSTOMERS». Для этого в режиме редактирования списка «LST_CUSTOMERS» указываем в его свойствах, что диалогом редактирования для этого списка является диалог «DLG_CUSTOMERS» и что в списке разрешены операции добавления/изменения/удаления информации:



По кнопке  сохраняем изменения в списке.

Теперь мы можем проверить работу нашего приложения уже и в части редактирования данных о заказчиках. Данные в этом списке теперь можно добавлять и удалять через диалог редактирования. Заведем и отредактируем несколько записей в качестве теста.

!!! Обратите внимание: Мы получили полностью работающий функционал по редактированию справочника заказчиков, не написав ни одной строки программного кода, описывающего как эти операции должны исполняться на сервере. Единственное что потребовалось, это знать структуру данных в базе и корректно написать запрос SELECT для извлечения данных.

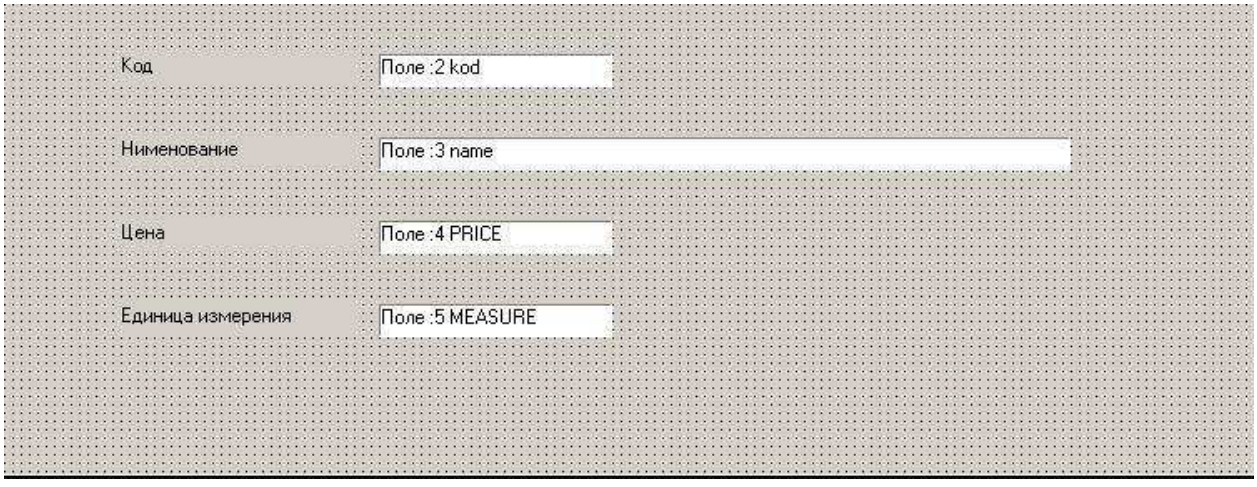
Диалог «Товары/услуги»

Аналогично предыдущему диалогу создаем новый диалог под именем «DLG_GOODS».

В секции «SQL- запрос» прописываем запрос для диалога:

```
@DLG_GOODS

SELECT id, kod, name, PRICE, MEASURE
 FROM CU_GOODS;
```

Заполняем форму диалога полями редактирования:



Подключим этот диалог к списку «LST_GOODS» и так же укажем, что в списке разрешены операции добавления/изменения/удаления информации.

Диалог «Заказы»

Аналогично предыдущему диалогу создаем еще один новый диалог под именем «DLG_ORDERS».

В секции «SQL- запрос» прописываем запрос для диалога:

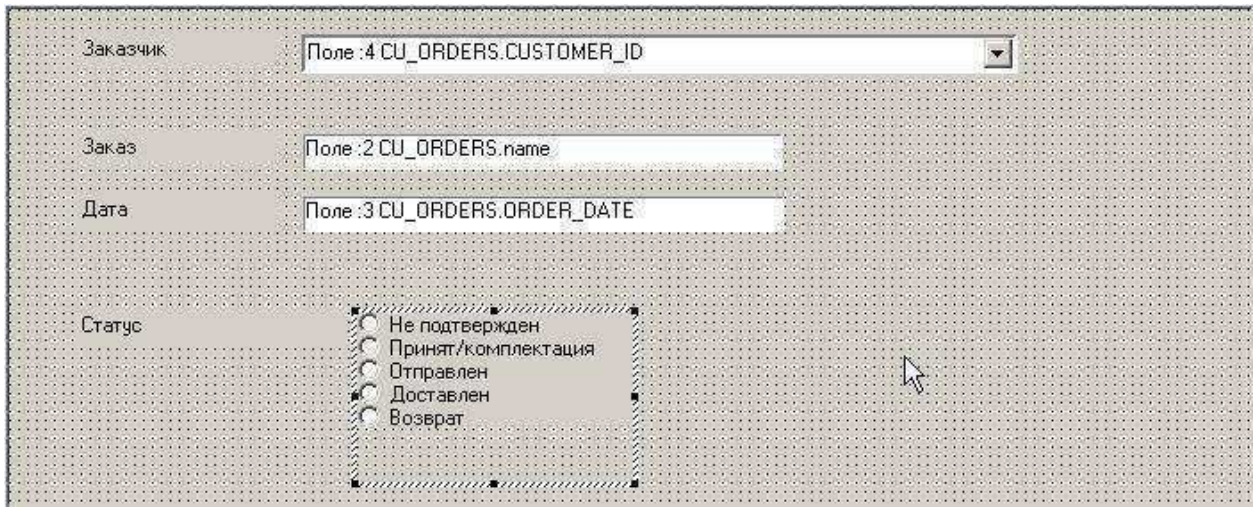
```
@DLG_ORDERS
SELECT id, name, ORDER_DATE, CUSTOMER_ID, status
 FROM CU_ORDERS;
```

В дополнение в секции «LET инициализация» пропишем вот такой код:

```
ORDER_DATE = getdate();
STATUS = 0;
```

В этом коде мы просто ссылаемся на поля диалога и определяем их значение по умолчанию, определяя поведение диалога при его открытии на добавление новой записи в список.

Сам диалог должен выглядеть вот таким образом:



Поля «Заказ» и «Дата» являются простыми полями редактирования, как это было в предыдущих диалогах. В вот поля «Заказчик» и «Статус» имеют другой тип. Поле

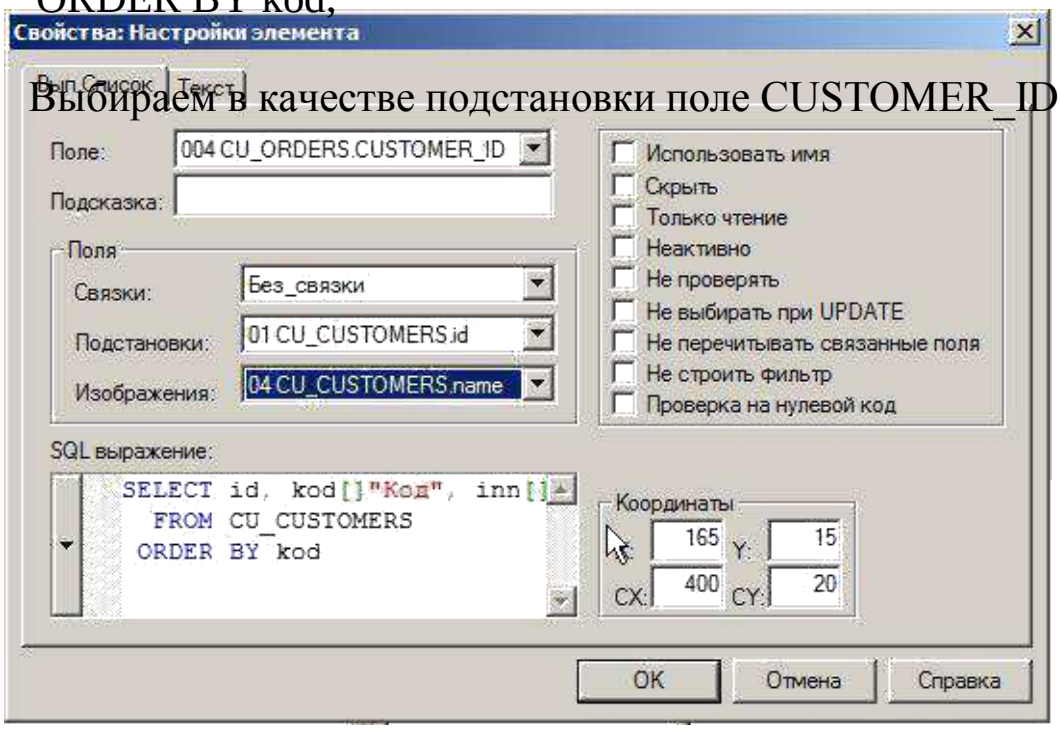
«Заказчик» имеет тип «Выпадающий список», а поле «Статус» - имеет тип «Переключатели».

Для поля «Заказчик» делаем вот такую настройку его свойств:

В поле «SQL выражение» (свойства элемента «Заказчик») вписываем запрос:

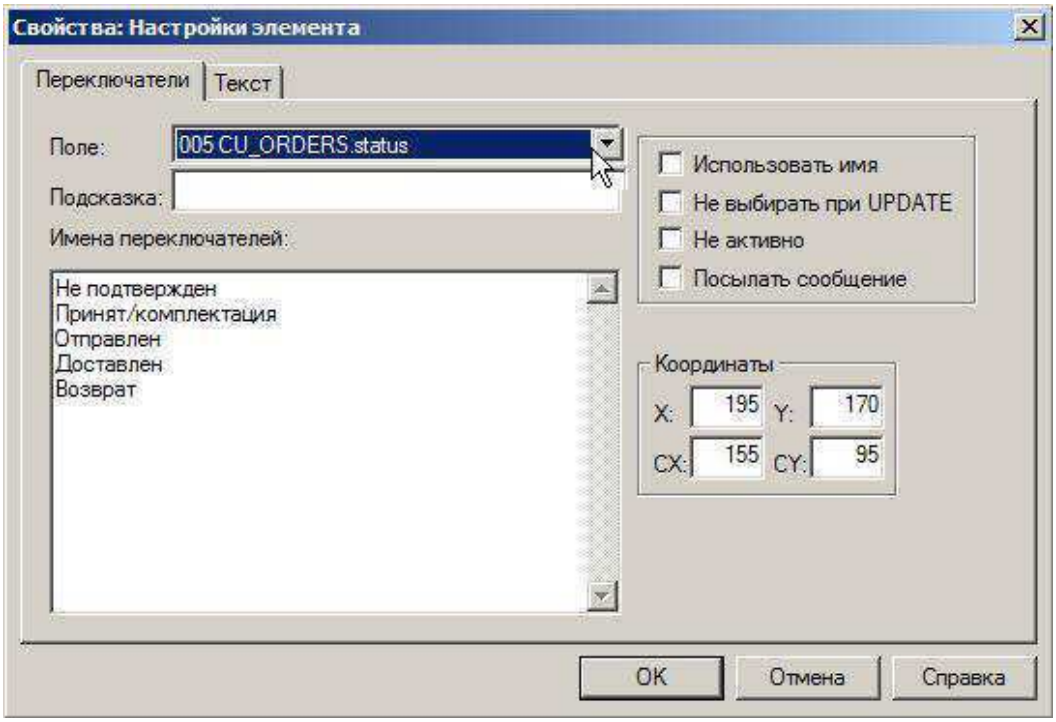
```
SELECT id, kod[]"Код", inn[]"ИНН", name[]"Наименование",
       adr[]"Адрес", acc[]"Счет", bank[]"Банк"

 FROM CU_CUSTOMERS
ORDER BY kod;
```

Выбираем в качестве подстановки поле CUSTOMER_ID.



При такой настройке указанное SQL выражение позволяет развернуть на экране для выбора подстановки список заказчиков и выбрать из них нужного. При этом в базе данных будет сохранено поле ID (ссылка на заказчика), а в диалоге будет отображаться название заказчика.

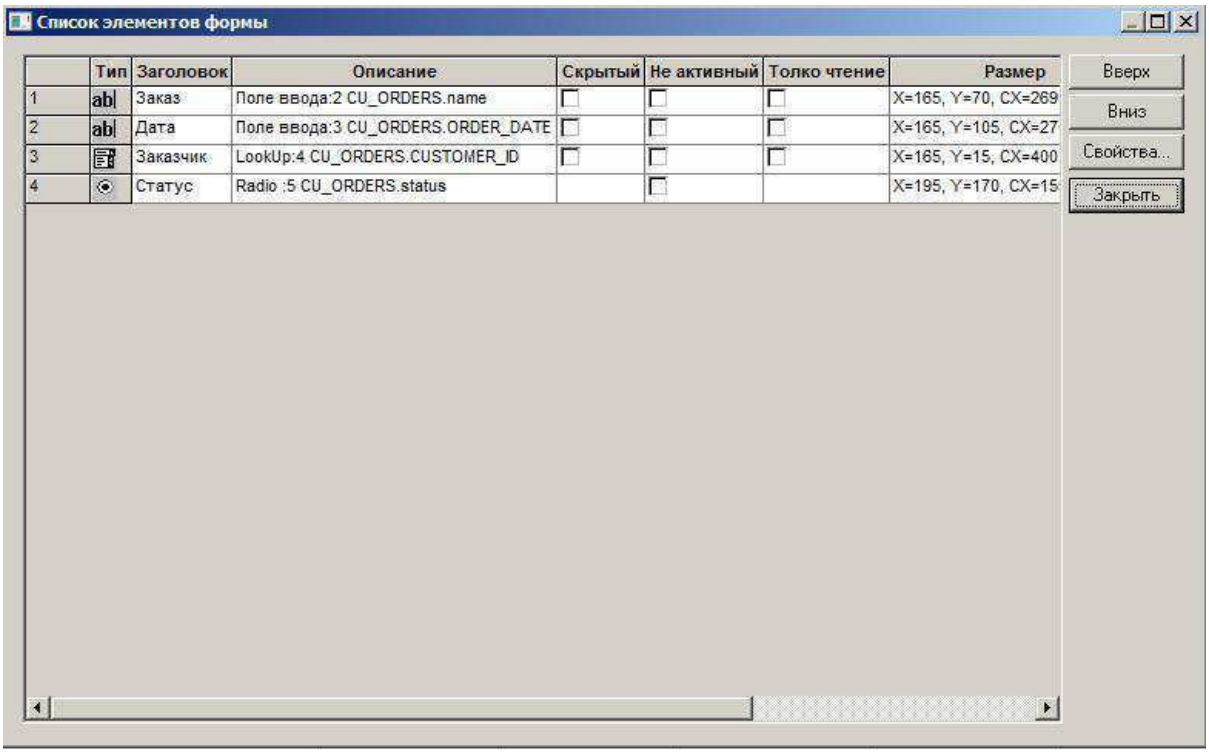
Для поля «Статус» делаем следующую настройку:



Подключаем созданный диалог к списку «LST_ORDERS», не забыв указать, что в нем разрешены операции добавления/изменения/удаления информации.

Если щелкнуть на поле редактирования диалога правой кнопкой мыши, то можно перейти в список элементов. Можем проверить, соответствует ли получившийся у нас порядок полей правильному порядку их обхода сверху вниз.

Если это необходимо, изменим порядок обхода клавишами «Вверх» или «Вниз»:



Диалог «Заказы+строки»

Аналогично предыдущему диалогу создаем новый диалог под именем «DLG_ORDER_LINES».

В секции «SQL-запрос» прописываем запрос для диалога:

```
@DLG_ORDER_LINES

SELECT ID, ORDER_ID, GOODS_ID, "COUNT"
 FROM CU_ORDER_LINES;
```

В секции «LET инициализация» пропишем вот такой код:

```
- -          Присваиваем
ссылку на текущий заказ.
ORDER_ID =
@LST_ORDERS:ID;
```

Это означает, что при открытии диалога на вставку в поле «заказ» будет подставлена ссылка на текущий заказ из списка заказов.

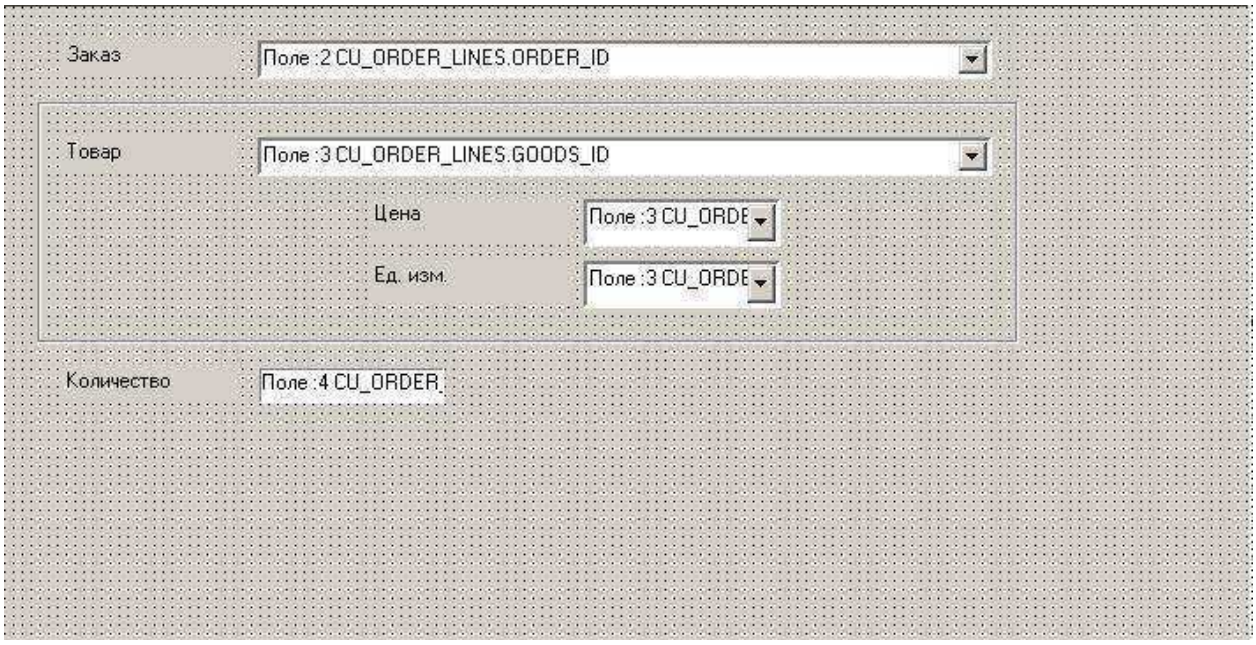
А в секции «Конечный SQL» напишем:

- - При изменении информации о строке заказа перечитать список заказов

MSG 2041, @LST_ORDERS;

Это означает, что при изменении информации о строке заказа обязательно нужно перечитать список заказов (команда перечитки, посылаемая соответствующему списку).

Сам диалог должен выглядеть вот таким образом:



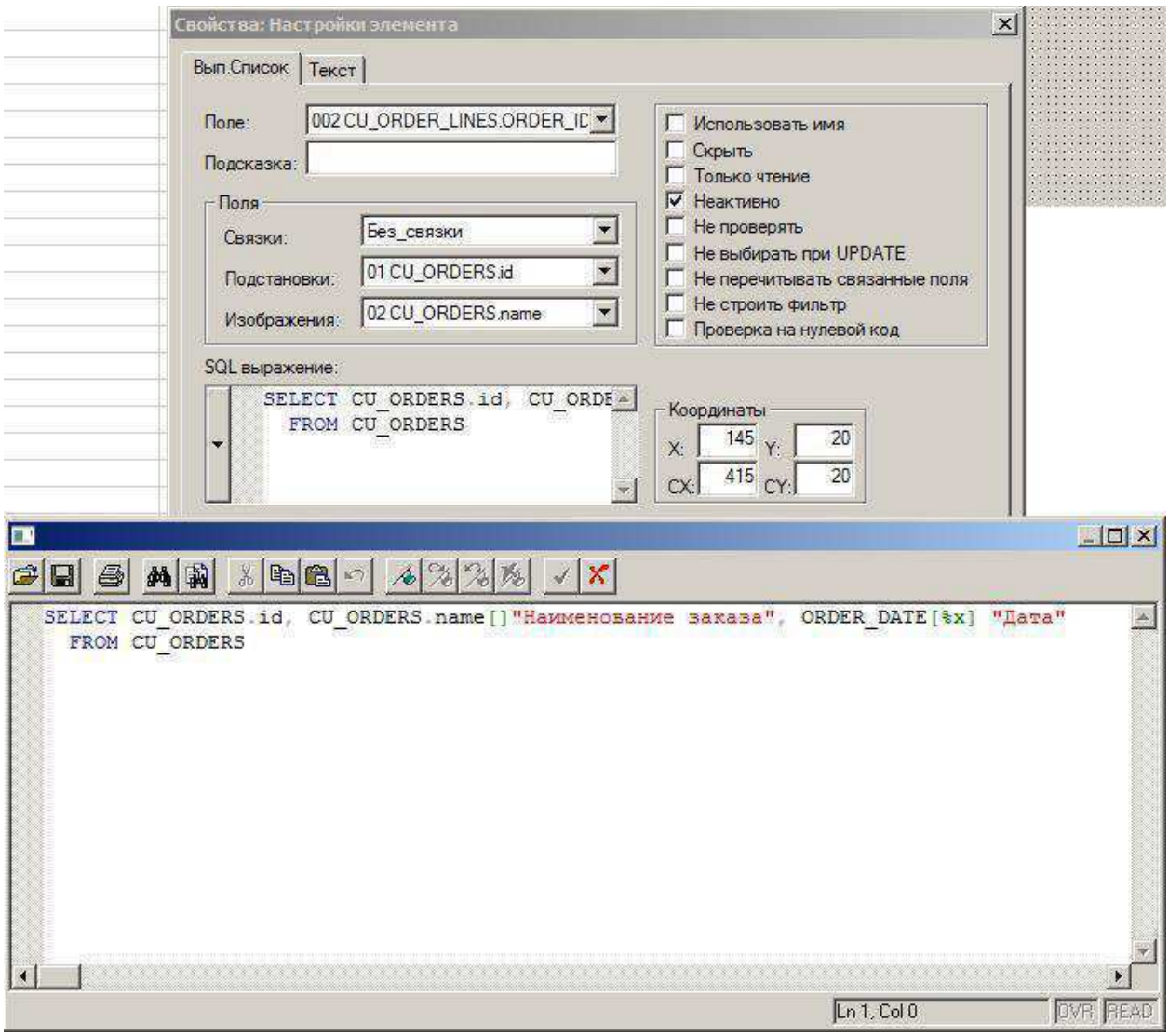
Поле «Количество» - простое поле редактирования.

Поле «Заказ» – это выпадающий список, запрос которого выглядит вот таким образом:

SELECT CU_ORDERS.id, CU_ORDERS.name[]"Наименование заказа", ORDER_DATE[%x] "Дата"

FROM CU_ORDERS

Настройки поля «Заказ», соответственно, такие:

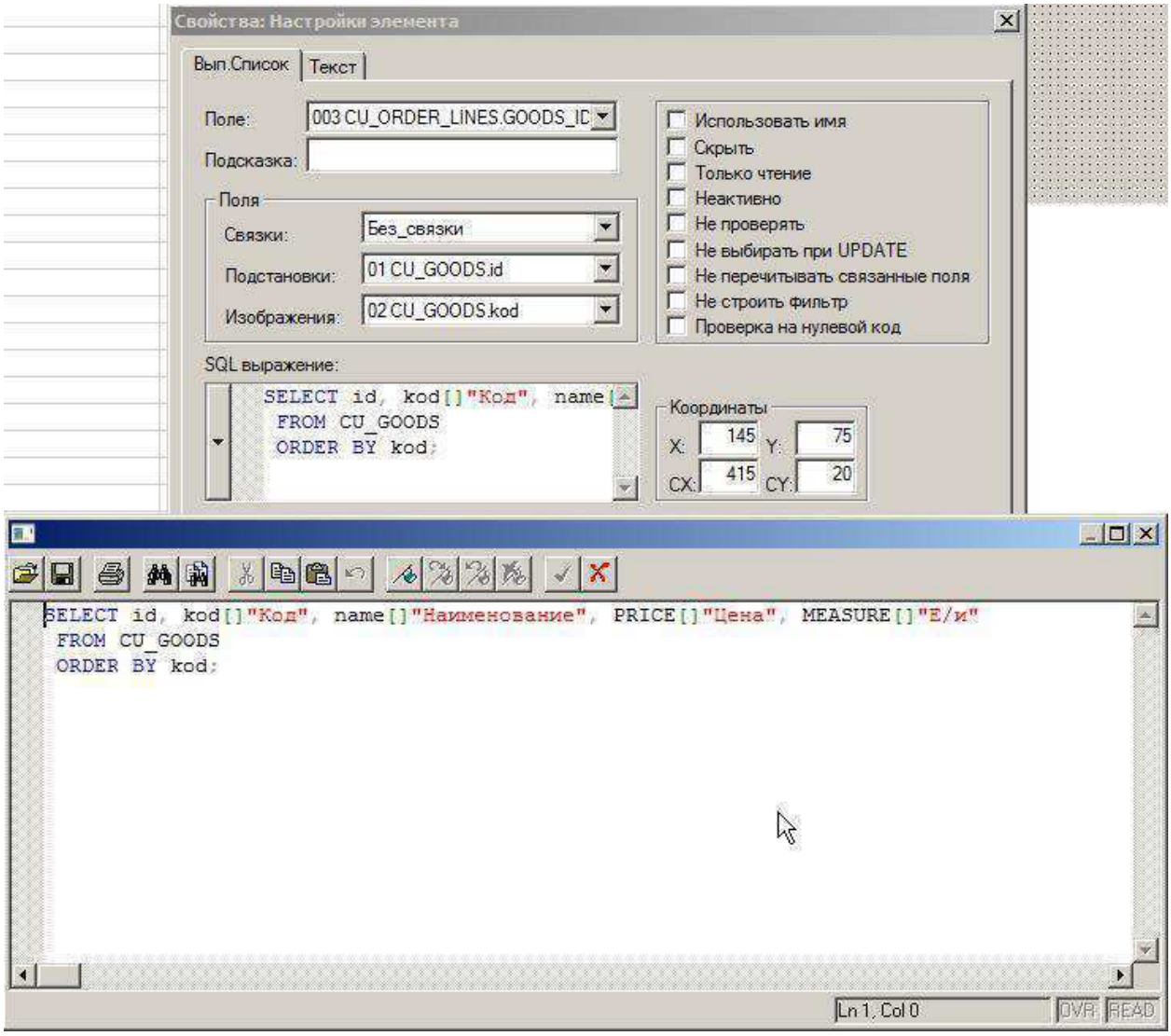


Поле «Товар» - тоже выпадающий список. Вот его запрос:

SELECT id, kod[]"Код", name[]"Наименование", PRICE[]"Цена", MEASURE[]"Е/и"

FROM CU_GOODS
ORDER BY kod;

Настройки поля «Товар», соответственно, такие:



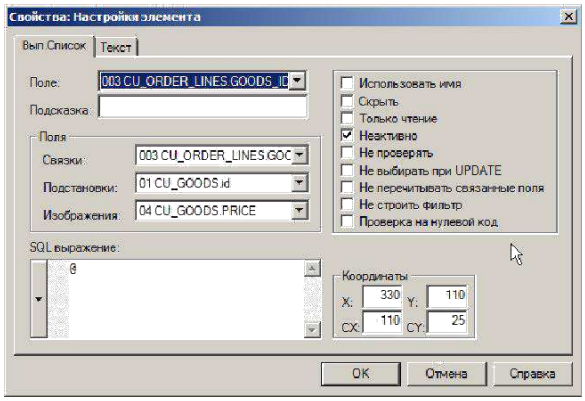
Немного дополним диалог.

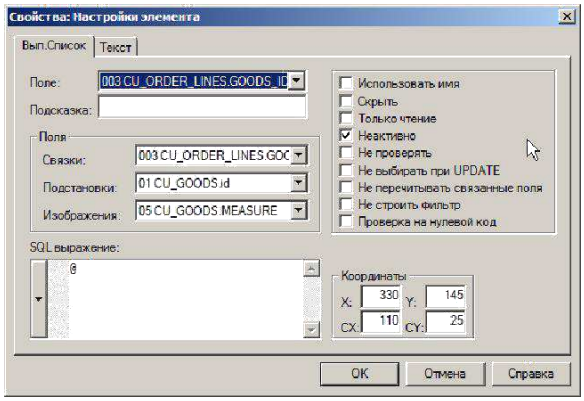
Мы выбираем товар из выпадающего списка. В поле «Товар» мы видим название товара/услуги.

Но у товара/услуги есть еще ряд полей, которые было бы интересно видеть в диалоге. Это поля

«Цена» и «Единица измерения». Будем выводить их в диалоге под названием выбранного товара, но без возможности их редактирования.

Для этого под полем «Товар» добавляем 2 поля типа «Выпадающий список», указав для них поле связки «GOODS.ID». Вот пример настройки свойств этих полей:





Подключаем созданный диалог к списку «LST_ORDER_LINES», не забыв указать, что в списке разрешены операции добавления/изменения/удаления информации.

Теперь выполним «оформление»

1. Пропишем заголовки окон.

У каждого окна в системе может быть заголовок. Это относится и к спискам, и к диалогам, и к MDI окнам. Пройдитесь по свойствам объектов и определите их заголовки. У списков и MDI окон это свойство называется «Заголовок окна». У диалогов это свойство называется «Добавить к заголовку».

!!! Обратите внимание: Заголовки диалогов следует указывать в винительном падеже, поскольку к введенному нами заголовку система сама автоматически будет добавлять слова «Добавить»/«Переписать»/ «Удалить» в зависимости от выполняемого пользователем действия.

2. Введем отображение статусов заказов

Для этого отредактируем запрос списка заказов (дополнив его тем, что выделено красным):

```
@LST_ORDERS
SELECT CU_ORDERS.id,
CU_ORDERS.name[]"Наименование заказа",
ORDER_DATE[%x] "Дата",
CU_CUSTOMERS.name[]"Заказчик",
(CASE status WHEN 0 THEN 'Не подтвержден'
WHEN 1 THEN 'Принят/комплектация'
WHEN 2 THEN 'Отправлен'
WHEN 3 THEN 'Доставлен'
```

```
                    WHEN 4 THEN 'Возврат' END)
                []"Статус",
    (CASE status WHEN 0 THEN 239
                    WHEN 1 THEN 251
                    WHEN 2 THEN 229
                    WHEN 3 THEN 228
                    WHEN 4 THEN 227 END)
    FROM CU_ORDERS _HINTBROWSER JOIN
CU_CUSTOMERS ON CU_CUSTOMERS.id =
    CU_ORDERS.CUSTOMER_ID
    ORDER BY CU_CUSTOMERS.name, ORDER_DATE;
```

Далее зайдем в секцию «Форма» этого списка и 2 раза нажмем <Ins> (“Добавить колонку”) и <F5>. Для последнего из 2- х появившихся новых полей поднимем флажки «Погасить» и «Определяет цвет» (т.е. само не светится в списке, но определяет цвет всей строки).

Если у нас в списке заказов есть строки, то, изменяя их статус, мы теперь будете получать разное выделение строк цветом (цвет определяется значением числа поля, определяющего цвет).

3. Выведем итоговые показатели в заказах.

Займемся дальнейшей редакцией запроса списка заказов. Еще дополним запрос (выделено красным):

```
    @LST_ORDERS
    SELECT                          CU_ORDERS.id,
CU_ORDERS.name[]"Наименование              заказа",
ORDER_DATE[%x]                          "Дата",
CU_CUSTOMERS.name[]"Заказчик",
        (CASE status WHEN 0 THEN 'Не подтвержден'

                    WHEN 1 THEN
                'Принят/комплектация'
                    WHEN 2 THEN 'Отправлен'
                    WHEN 3 THEN 'Доставлен'
                    WHEN 4 THEN 'Возврат' END)
                []"Статус",
    (CASE status WHEN 0 THEN 239
```

```
                                    WHEN 1 THEN 251
                                    WHEN 2 THEN 229
                                    WHEN 3 THEN 228
                                    WHEN 4 THEN 227 END),
            (SELECT SUM(PRICE * COUNT)
                  FROM CU_ORDER_LINES, CU_GOODS
                  WHERE      CU_ORDER_LINES.GOODS_ID      =
            CU_GOODS.ID
             AND ORDER_ID = CU_ORDERS.ID)
      AS    TOTAL[%19.2f]"ИТОГО",    (SELECT
      COUNT(*)
                  FROM CU_ORDER_LINES
            WHERE  ORDER_ID  =  CU_ORDERS.ID)
 AS  STROK[%19.0f]"строк"  FROM  CU_ORDERS
 _HINTBROWSER  JOIN  CU_CUSTOMERS  ON
 CU_CUSTOMERS.id =
      CU_ORDERS.CUSTOMER_ID
       ORDER BY CU_CUSTOMERS.name, ORDER_DATE;
```

Опять зайдем в секцию «Форма» этого списка и 2 раза нажмем <Ins> (“Добавить колонку”) и <F5>. Теперь при изменении состава заказа его общая стоимость и число строк заказа пересчитываются автоматически.

ИТАК, МЫ СОЗДАЛИ ПРИЛОЖЕНИЕ, АВТОМАТИЗИРУЮЩЕЕ УЧЕТ ЗАКАЗОВ НАШЕГО ПРЕДПРИЯТИЯ!

Отчет по практической работе 5 должен содержать: цель работы; краткое описание использованных в работе элементов языка рассмотренной платформы; текст созданных процедур и результаты их работы; выводы по результатам практической работы. При оформлении отчета в печатном виде в нижний колонтитул следует поместить фамилию, инициалы и номер группы обучаемого (8 пт., Arial, выравнивание по правому краю).

Практическая работа № 6
Инструментальные средства технологической платформы «1С:Предприятие»
Практическая работа 6 (Часть 6.1)
Особенности платформы «1С:Предприятие 8». Использование инструментальных средств платформы «1С:Предприятие 8» для создания создание информационной базы.

Цель работы: Изучение приемов создания информационных баз данных с использованием инструментальных средств платформы «1С:Предприятие 8», знакомство со структурой интерфейса.

Задание 1.1 Создание и регистрация информационной базы данных

Создать информационную базу данных Учебная группа_№ подгруппы.

Решение

1. Вызвать на экран окно запуска программы «1С:Предприятие 8» двойным щелчком на ярлыке 1С:Предприятие 8 либо командой Пуск => Программы => 1С:Предприятие 8 => 1С:Предприятие.
2. Щелкнуть на кнопке Добавить в открывшемся окне.
3. В форме Добавление информационной базы / группы установить переключатель в положение Создание новой информационной базы и щелкнуть на кнопке Далее.
4. На следующем экране установить переключатель в положение Создать информационную базу из шаблона, открыть группу шаблонов Бухгалтерия предприятия, выбрать шаблон, выделенный для соответствующей учебной группы, и щелкнуть на кнопке Далее.
5. На следующем экране указать наименование информационной базы Учебная группа_№ одгруппы и щелкнуть на кнопке Далее.
6. На последнем экране указать расположение информационной базы на компьютере и щелкнуть на кнопке Готово.

Задание 1.2 Общее знакомство с платформой

1. Запустить программу «1С:Предприятие 8» для работы с информационной базой Бухгалтерия предприятия (демо).
2. Ознакомиться с элементами главного окна программы: заголовком окна, строкой главного меню, списком команд каждого пункта главного меню, панелями инструментов, пиктограммами, строкой состояния.
3. Закрыть программу «1С:Предприятие 8».

Практическая работа 6 (Часть 6.2)
Использование инструментальных средств платформы «1С:Предприятие 8» для первоначальной настройки модели предметной области

Цель работы: Изучение основных приемов первоначальной настройки модели предметной области с использованием инструментальных средств платформы «1С:Предприятие 8».

Задание 2.1 Ввод сведений об организации

Необходимо подготовить информационную базу для ведения учета организации ЗАО «ФАБРИКА ЭКСКЛЮЗИВ», учредителями которой являются юридические лица — организации ОАО «ИНЭК» и АКБ «Русь-Инвест» а также физические лица Русанов Е.Л., Власов В.А. и Петров Е.Д.

Регистрация ЗАО «ФАБРИКА ЭКСКЛЮЗИВ» произведена 12 января 2009 г.

Ввести в справочник Организации сведения о ЗАО «ФАБРИКА ЭКСКЛЮЗИВ».

Решение

1. Командой меню Предприятие Организации открыть справочник Организации и дважды щелкнуть мышью на строке с наименованием Наша организация. Указанная строка была введена автоматически при начальном заполнении информационной базы.
2. В верхней части формы ввести сведения об организации из примечания 2.1
3. Для заполнения реквизита Осн. банковский счет кнопкой «…» в правой части поля ввода открыть справочник Банковские

счета, щелчком на пиктограмме **(**Добавить**)** открыть форму Элемент. Банковские счета и заполнить ее данными из табл. Сведения о счете сохранить в справочнике (сохранение происходит при щелчке на кнопке ОК**),** установить ссылку на этот счет как основной счет организации и продолжить заполнение формы со сведениями об организации.

4. На закладке Основные ввести основные сведения об организации из примечания 2.1

5. На закладке Контактная информация ввести сведения о юридическом, фактическом и почтовом адресах, а также телефоне ЗАО «ФАБРИКА ЭКСКЛЮЗИВ» из примечания 2.1

Для ввода адреса нужно дважды щелкнуть мышью в колонке Представление соответствующей строки, кнопкой «…» открыть форму, и заполнить ее.

6. На закладке Коды ввести присвоенные организации коды из примечания 2.1

7. На закладке Фонды ввести присвоенные организации коды в Пенсионном фонде России и Фонде социального страхования Российской Федерации из примечания 2.1

8. Закрыть форму Организации: ЗАО ФАБРИКА ЭКСКЛЮЗИВ щелчком на кнопке ОК. Закладка Доступ к объектам в учебном примере не заполняется.

Примечание 2.1

Сведения об организации

Наименование	ЗАО «ФАБРИКА ЭКСКЛЮЗИВ»
Полное наименование	Закрытое акционерное общество «ФАБРИКА ЭКСКЛЮЗИВ»
Наименование плательщика в платежных документах на перечисление налогов	Закрытое акционерное общество «ФАБРИКА ЭКСКЛЮЗИВ»

Банковские реквизиты

Наименование счета (рабочее)	Основной
Наименование банка	ЗАО АКБ «Альтаир»
Корр. счет	30101810800000000272

БИК	044585272
Город	г. Воронеж
Адрес банка	129228, Воронеж, ул. Ленина, д. 7
Телефон	(4714) 224-16-81
Расчетный счет	40702810600006132001
Вид счета	Расчетный
Дата открытия	12.01.2009
Валюта счета	руб.

Основные сведения об организации

ИНН	7705200107
КПП	770501001
ОГРН	1023142218109
Дата государственной регистрации	
Код ИФНС	7705
Наименование ИФНС	Инспекция ФНС № 05 по г. Воронеж
Дата выдачи свидетельства о постановке на налоговый учет	12.01.2009
Серия и номер свидетельства	77 № 1012341234
Код налогового органа, выдавшего свидетельство	7705
Наименование налогового органа, выдавшего свиде-тельство	Инспекция ФНС № 05 по г. Воронеж

Контактная информация

Почтовый адрес	121151, Воронеж, а/я 151
Юридический адрес	121151, Воронеж, ул. Гагарина, д. 23
Фактический адрес	121151, Воронеж, ул. Гагарина, д. 23
Телефон	(4714) 924-75-18

Коды

ОКАТО	45286560000

ОКПО	52707832
Код организационно-правовой формы по ОКОПФ	67
Наименование организационно-правовой формы	Закрытое акционерное общество
Код формы собственности по ОКФС	17
Наименование формы собственности	Смешанная
Код вида деятельности по ОКВЭД	36
Наименование вида деятельности	Производство мебели

Фонды

Регистрационный номер в ПФР	087-105-071284
Регистрационный номер в ФСС	770810116

Задание 2.2 Ввод сведений об учетной политике организации

Ввести сведения об учетной политике бухгалтерского учета организации ЗАО «ФАБРИКА ЭКСКЛЮЗИВ».

Решение

1. Командой меню Предприятие => Учетная политика =>Учетная политика **(**бухгалтерский учет) вывести на экран форму регистра, а затем щелчком на пиктограмме панели инструментов открыть форму для ввода сведений об учетной политике.

2. Указать сведения об учетной политике ЗАО «ФАБРИКА ЭКСКЛЮЗИВ» на 2009 г. и сохранить их, щелкнув на кнопке ОК.

В результате в регистре сведений Учетная политика (бухгалтерский учет) появится соответствующая запись.

Сведения об учетной политике, определяющие поведение системы для целей налогового учета, хранятся в другом регистре сведений — Учетная политика (налоговый учет). Для описания

учетной политики для целей налогообложения необходимо открыть регистр командой меню Предприятие => Учетная политика =>Учетная политика (налоговый учет)**,** щелчком на пиктограмме открыть форму новой записи, заполнить закладки Основная, НДС и Налог на прибыль и сохранить сведения, щелкнув на кнопке ОК.

Примечание 2.2

Из приказа об учетной политике бухгалтерского учета ЗАО «ФАБРИКА ЭКСКЛЮЗИВ» на 2009 г.:

1. Оценка материально-производственных запасов производится по средней себестоимости.

2. Косвенные расходы, собираемые по дебету счетов 25 Общепроизводственные расходы и 26 Общехозяйственные расходы, распределяются между видами номенклатуры — объектами калькулирования пропорционально заработной плате основных производственных рабочих.

3. Учет выпуска готовой продукции организуется с применением счета 40 Выпуск продукции (работ, услуг).

4. Организация применяет Положение по бухгалтерскому учету Учет расчетов по налогу на прибыль (ПБУ 18/02).

Задание 2.3 Ввод сведений об учетной политике организации для целей налогообложения

Ввести сведения об учетной политике для целей налогообложения организации ЗАО «ФАБРИКА ЭКСКЛЮЗИВ».

Решение

1. Командой меню Предприятие => Учетная политика => Учетная политика (налоговый учет) вывести на экран форму регистра, а затем щелчком на пиктограмме панели инструментов открыть форму для ввода сведений об учетной политике.

2. На закладке Основная указать, что учетная политика для организации ЗАО «ФАБРИКА ЭКСКЛЮЗИВ» вводится с 1 января 2009 г., остальные реквизиты оставить без изменения.

3. На закладке НДС проверить, что налоговым периодом является Месяц, остальные реквизиты оставить без изменения.

4. На закладке Налог на прибыль проверить, что установлен порядок учета расходов по налогам с ФОТ на счетах расходов на оплату труда, остальные реквизиты оставить без изменения.

5. На закладке НДФЛ оставить в поле Особенности исчисления значение по умолчанию Стандартные вычеты предоставляются нарастающим итогом с начала налогового периода и сохранить сведения об учетной политике, щелкнув на кнопке ОК.

В результате в регистре сведений Учетная политика (налоговый учет) появится соответствующая запись.

В отдельном регистре Учетная политика организаций по персоналу хранятся сведения, определяющие поведение системы для целей учета расчетов с персоналом:

- поддерживать или не поддерживать внутреннее совместительство;
- при начислении НДФЛ принимать или не принимать исчисленный налог к учету как удержанный.

Для ввода соответствующих сведений необходимо открыть регистр командой меню Предприятие => Учетная политика => Учетная политика (по персоналу), щелчком на пиктограмме открыть форму новой записи, установить флажки в соответствующих параметрах учетной политики и сохранить сведения щелчком на кнопке ОК.

Примечание 2.3

Из приказа об учетной политике для целей налогообложения ЗАО «ФАБРИКА ЭКСКЛЮЗИВ» на 2009 г.:

1. Оценка материально-производственных запасов производится по средней стоимости.

2. Расходы в виде налогов, начисляемых на вознаграждения и выплаты в пользу работников с ФОТ, для целей налога на прибыль учитываются на тех же счетах, что и расходы на вознаграждения и иные выплаты, формирующие налоговую базу.

Задание 3.1 Заполнение справочника «Подразделения организаций»

Ввести в справочник Подразделения организаций подразделения ЗАО «ФАБРИКА ЭКСКЛЮЗИВ».

Решение

1. Командой меню Предприятие => Подразделения организаций вывести на экран форму Список подразделений организации ЗАО ФАБРИКА ЭКСКЛЮЗИВ.

2. Командой меню Действия => Добавить (либо командой контекстного меню Добавить, либо щелчком на пиктограмме панели инструментов, либо нажатием клавиши Insert) вывести на экран форму Подразделения организаций**.**

3. Ввести в поле Наименование наименование первой группы подразделений — Административные, код оставить «по умолчанию» и щелкнуть на кнопке ОК.

4. Повторить действия и ввести в справочник наименование второй группы подразделений — Производственные**.**

5. Открыть форму для ввода сведений о новом подразделении, в реквизите Группа указать ссылку на группу подразделений Административные, в реквизите Наименование указать Администрация и сохранить данные щелчком на кнопке ОК.

6. По аналогии ввести остальные подразделения ЗАО «ФАБРИКА ЭКСКЛЮЗИВ» в соответствии с примечанием 3.1. При правильном заполнении структура ЗАО «ФАБРИКА ЭКСКЛЮЗИВ» должна быть описана в справочнике Подразделения организаций.

Примечание 3.1

Справочник Подразделения организаций используется для ведения аналитического учета в разрезе подразделений на счетах 20, 23, 25, 26, 28, 29 и др. Справочник позволяет вести учет на местах производственных работ (столярный цех, красильный цех, полировочный цех) и по местам концентрации хозяйственных функций (администрация, бухгалтерия).

Организационная структура ЗАО «ФАБРИКА ЭКСКЛЮЗИВ» включает следующие подразделения.

Группа подразделений	**Подразделения**
Административные	Администрация
	Бухгалтерия
Производственны	Столярный цех

e	

Задание 3.2 Заполнение справочника «Номенклатурные группы»

Заполнить справочник Номенклатурные группы видами продукции, выпускаемой ЗАО «ФАБРИКА ЭКСКЛЮЗИВ».

Решение

1. Командой меню Основная деятельность => Товары (материалы, продукция, услуги) => Номенклатурные группы вывести на экран форму справочника Номенклатурные группы.

2. Двойным щелчком на строке с наименованием Основная номенклатурная группа (была введена автоматически при начальном заполнении информационной базы) открыть форму Номенклатурные группы.

3. В поле Наименование заменить значение Основная номенклатурная группа на Столы письменные и нажать на клавишу Enter.

4. В поле Код заменить значение «по умолчанию» на 3611110 и щелкнуть на кнопке ОК.

5. Командой меню Действия => Добавить вывести на экран форму для ввода следующей номенклатурной группы и ввести в поле Наименование — Столы обеденные, в поле Код — 3611100, после чего щелкнуть на кнопке ОК.

6. Повторить процедуру для вида продукции Столы кухонные.

Примечание 3.2

Справочник Номенклатурные группы применяется для ведения аналитического учета на счетах 20, 23, 28, 29, 40 и 90.

Производственной программой ЗАО «ФАБРИКА ЭКСКЛЮЗИВ» предусмотрен выпуск следующих видов продукции:

Код по ОК 004-93	Наименование вида продукции
3611110	Столы письменные
3611100	Столы обеденные
3611140	Столы кухонные

Практическая работа 6 (Часть6.3)

Использование инструментальных средств платформы «1С:Предприятие 8» для настройки прав доступа к системе

Цель работы: знакомство с приемами конфигурирования и администрирования с использованием инструментальных средств платформы «1С:Предприятие 8»

Задание для самостоятельного выполнения

Зарегистрировать себя в справочнике Пользователи. Установить значения для подстановки в формах справочников и документов: Основная валюта взаиморасчетов — руб.; Основная единица по классификатору, - штука; Основная ставка НДС — 18%; Основной ответственный — Ваша фамилия; Основной тип цен продажи — Отпускная цена.

Примечание 4.1 С программой «1С:Бухгалтерия 8.1» одновременно могут работать несколько пользователей. Список пользователей хранится в справочнике Пользователи.

Для того чтобы добавить в список нового пользователя, нужно завершить работу с информационной базой в режиме 1С:Предприятие, вновь запустить программу «1С:Предприятие 8» и выбрать режим работы Конфигуратор.

В меню Администрирование рабочего окна выбрать пункт Пользователи. Затем в форме Список пользователей на закладке Основные в поле Имя указать фамилию и инициалы пользователя, а в поле Полное имя — его фамилию, имя и отчество полностью.

Перейти на закладку Прочие, флажком отметить доступные роли — Полные права, указать интерфейс «по умолчанию» — Полный и используемый язык - Русский.

Сохранить информацию нажатием на кнопке ОК и завершить работу в режиме Конфигуратор.

Вновь запустить программу «1С: Предприятие 8.1» для работы с информационной базой в режиме 1С: Предприятие.

Для настройки параметров конфигурации конкретного пользователя нужно открыть справочник Пользователи и дважды щелкнуть на строке с именем пользователя. При этом откроется форма Настройки пользователя.

Для каждого пользователя на закладке Настройки формы сведений можно указать отдельные настройки и значения по умолчанию.

Практическая работа 6 (Часть6.4)

Использование инструментальных средств платформы «1С:Предприятие 8» для создания новой конфигурации

Цель работы: знакомство с приемами создания новой конфигурации с использованием инструментальных средств платформы «1С:Предприятие 8»

Постановка задачи

Платформа «1С: Предприятие 8» поставляется со средством разработки, с помощью которого создаются новые или изменяются существующие прикладные решения. Это средство разработки называется «**конфигуратор**». Так как он включен в стандартную поставку 1С: Предприятия, то пользователь может самостоятельно разработать или модифицировать прикладное решение (адаптировать его под себя), возможно, с привлечением сторонних специалистов.

Конфигуратор включает следующие удобные инструменты:

- дерево конфигурации,
- окно свойств,
- различные редакторы (форм, интерфейсов, программных модулей и т.д.),
- конструкторы,
- отладчик,
- синтакс-помощник
- и другие инструменты.

Структура прикладного решения в конфигураторе создается **визуальными средствами**. Разработчик создает различные объекты и настраивает их взаимосвязи друг с другом. С помощью визуальных редакторов создаются таблицы, экранные формы, макеты отчетов и печатных форм документов. Встроенный язык используется для описания различных алгоритмов, например,

расчета налогов, исчисления себестоимости, алгоритмов проведения документов и формирования отчетов.

В практической работе выполним **разработку прикладного решения**. Создание объектов и настройка взаимосвязей между ними производится визуальными средствами. Ниже приводится общая схема данного прикладного решения (конфигурации) (рис.1).

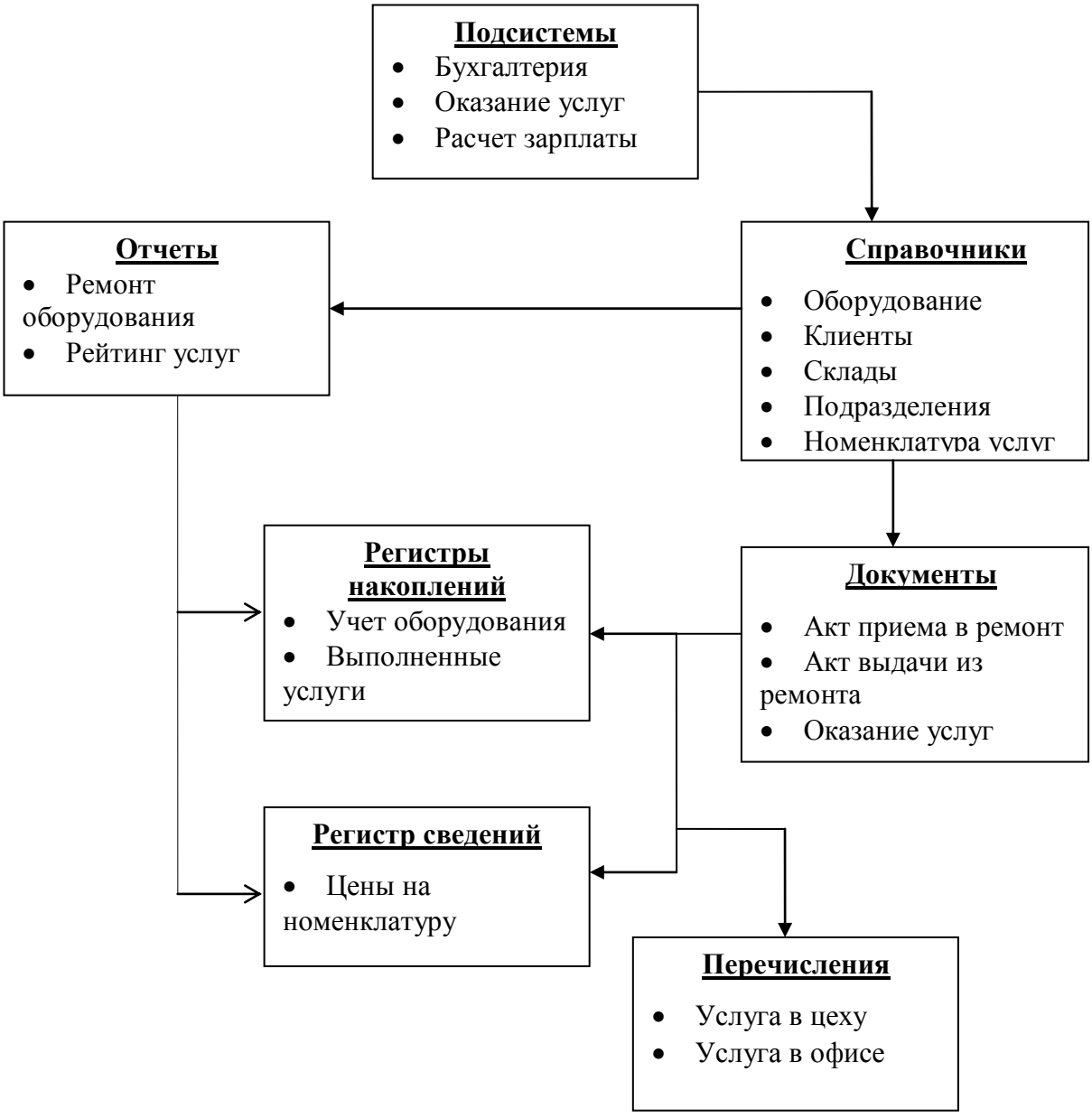


Рис.1. Общая схема будущей программы

Все эти объекты создаются с помощью визуальных средств.

Порядок выполнения работы

Прежде чем начинать работу необходимо создать новую информационную базу.

1. Создать на рабочем столе новую папку «ФИО студента».

2. Запустите программу 1С: Предприятие 8 (Учебная версия). На экране появится окно Запуск 1С: Предприятие (рис. 2).

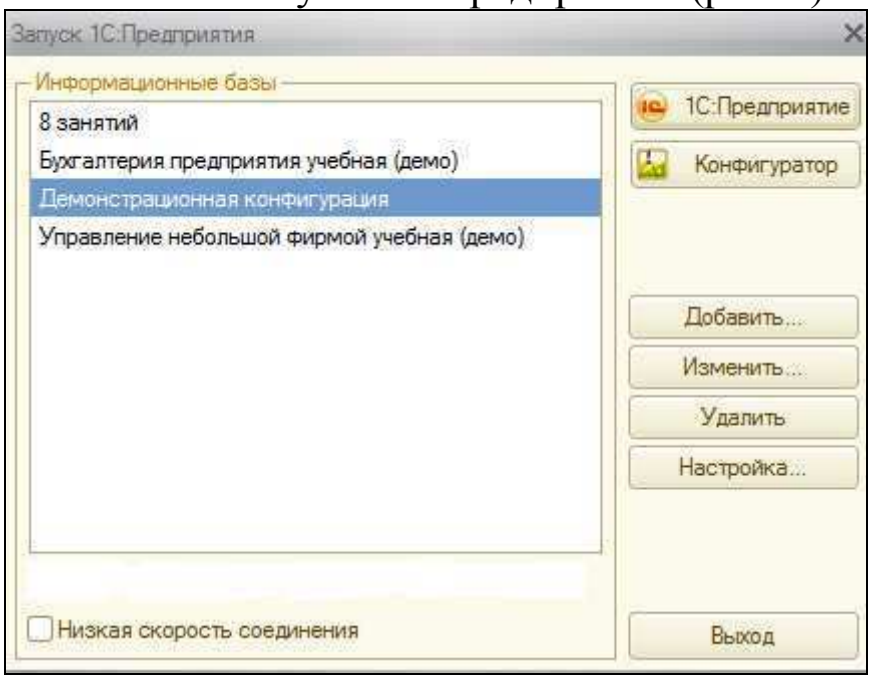


Рис.2. Запуск 1С: Предприятие

3. Нажмите на кнопку **Добавить** новую базу.

На формах (рис. 3-5а) введите следующие данные:

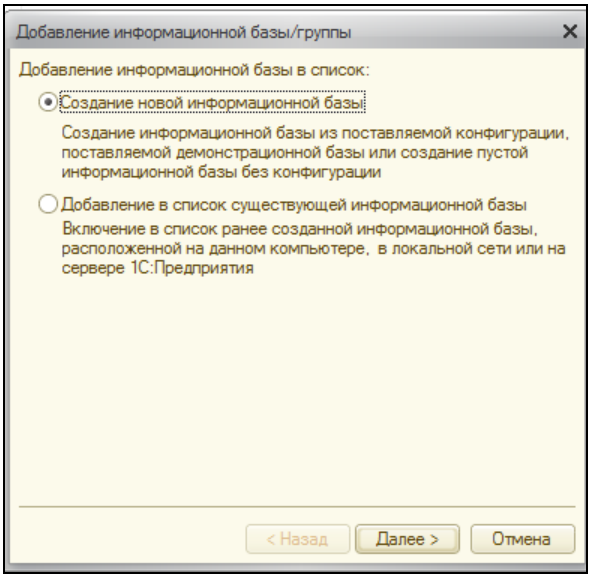


Рис. 3.Добавление новой информационной базы

Создаем информационную базу без конфигурации (рис. 4).

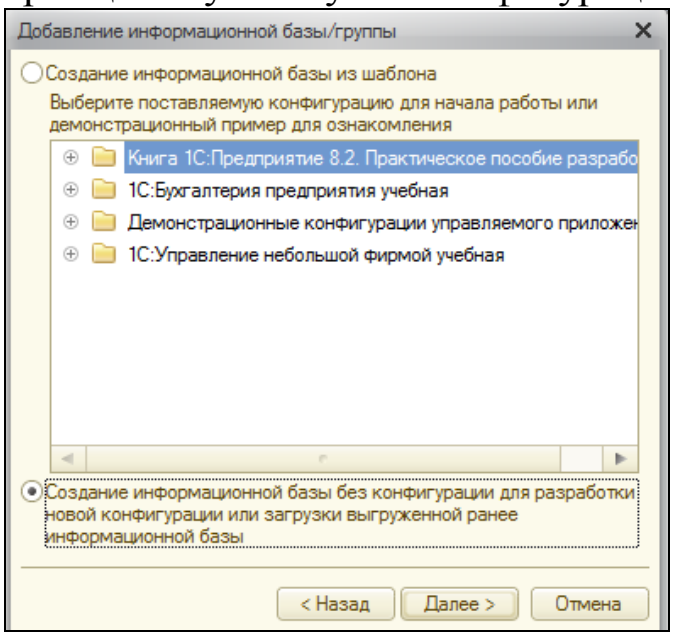


Рис. 4. Добавление новой информационной базы

В поле Укажите наименование информационной базы назовите имя новой базы Ремонт насосов (рис. 4а).

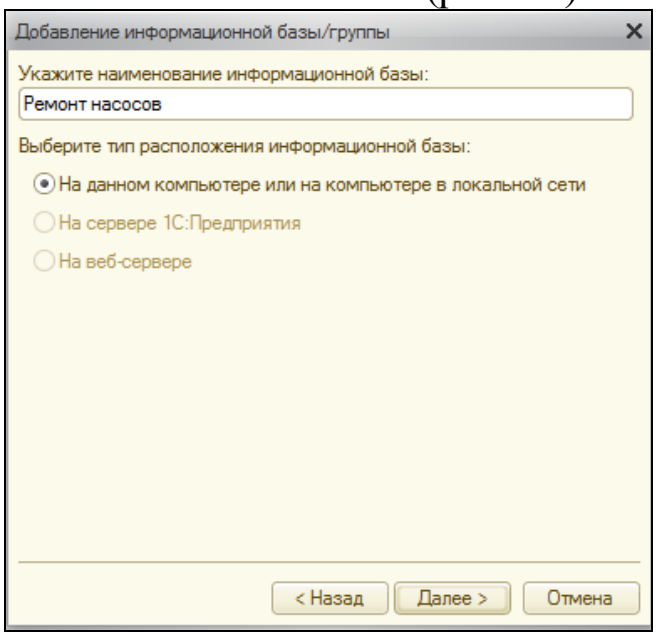


Рис. 4а. Добавление наименование новой информационной базы

В поле Каталог информационной базы укажите ту папку, которая была создана в п.1.

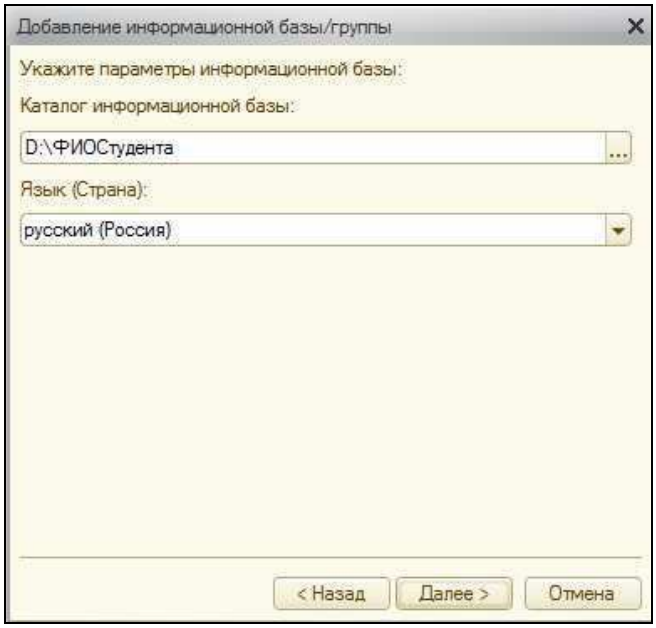


Рис. 5. Добавление новой информационной базы

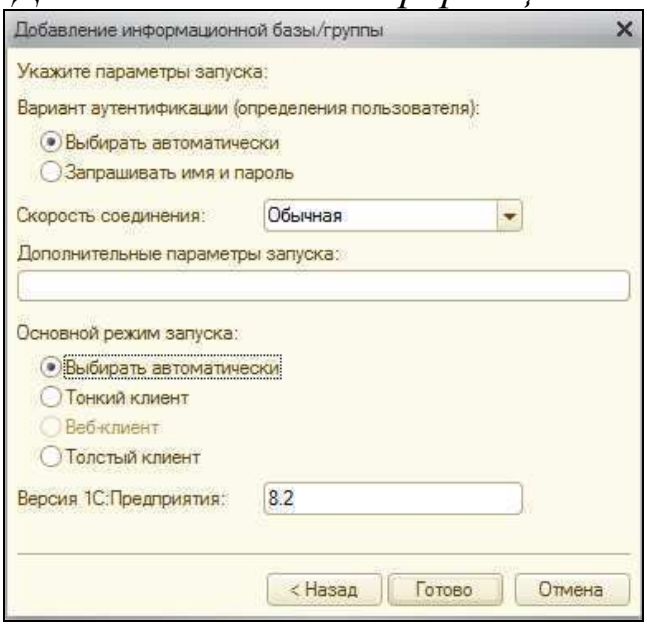


Рис. 5а. Добавление новой информационной базы

4. Выберите только что созданную информационную базу и нажмите на кнопку Конфигуратор (рис. 6).

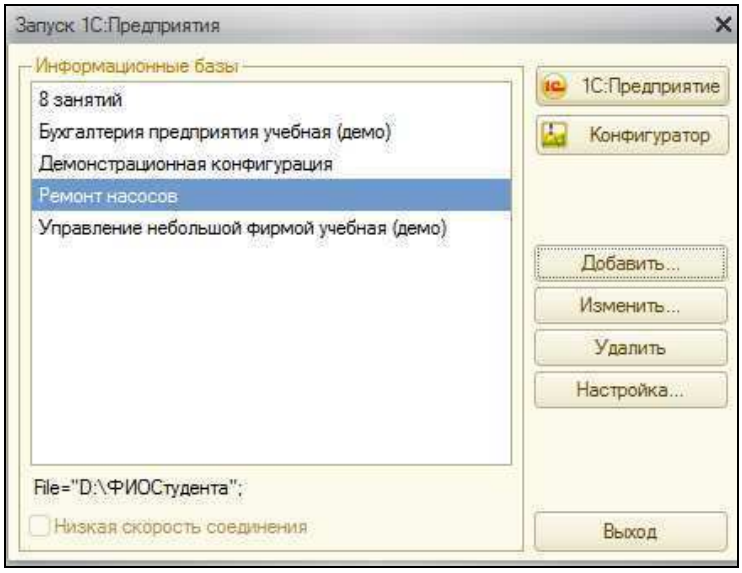


Рис. 6. Запуск 1С: Предприятие

Таким образом, была создана новая пустая информационная база, наполнением которой произведем далее.

После запуска Конфигуратора и создания новой информационной базы нужно открыть дерево конфигурации при помощи кнопки на панели инструментов (рис.7). Можно начать создание объектов конфигурации.

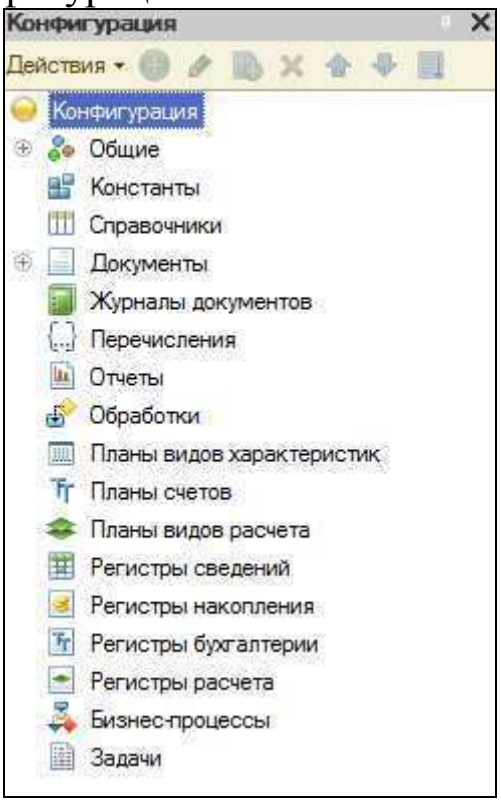


Рис. 7. Окно «Конфигурация»

Конфигурация представляет собой дерево метаданных. Метаданные описывают все аспекты работы приложения. Они представляют собой структурированное декларативное его описание. Метаданные образуют иерархию объектов, из которых формируются все составные части конфигурации и которые определяют все аспекты ее поведения. Фактически, при работе бизнес-приложения платформа «проигрывает» (интерпретирует) метаданные, обеспечивая всю необходимую функциональность.

Метаданными описываются структуры данных, состав типов, связи между объектами, особенности их поведения и визуального представления, система разграничения прав доступа, пользовательский интерфейс и т.д. В метаданных, фактически, сосредоточены сведения не только о том, «что хранить в базе данных», но и о том, «зачем» хранится та или иная информация, какова ее роль в системе и как связаны между собой информационные массивы. Использование языка программирования ограничено в основном решением тех задач, которые действительно требуют алгоритмического описания, например, расчета налогов, проверки корректности введенных данных.

Дадим конфигурации Имя. На корне Конфигурации (корень дерева конфигурации) щелкаем левой кнопкой мыши. Появляется контекстное меню (рис. 8), выбираем Свойства и задаем название Конфигурации «**Ремонт оборудования**» (рис. 9).

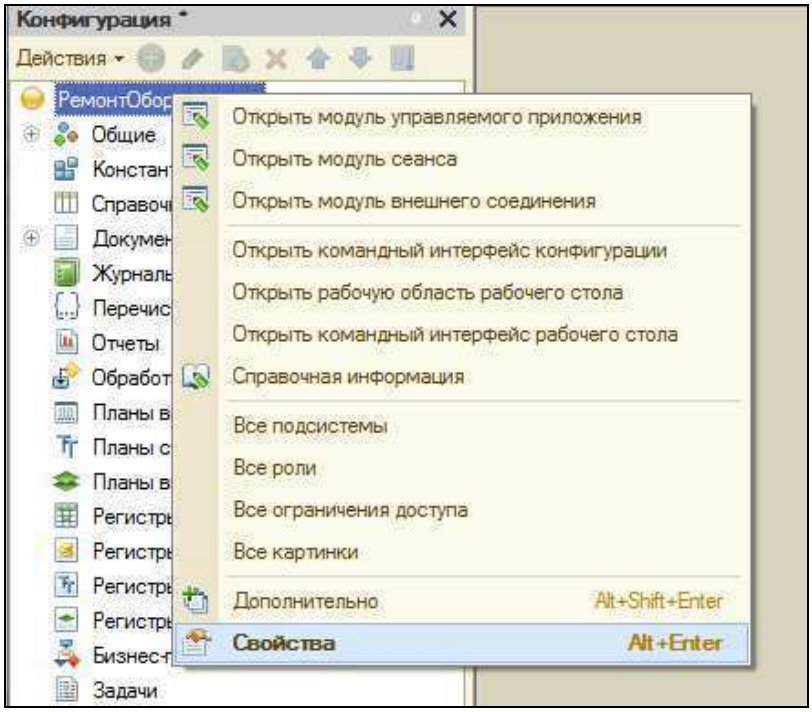


Рис. 8. Контекстное меню корня конфигурации

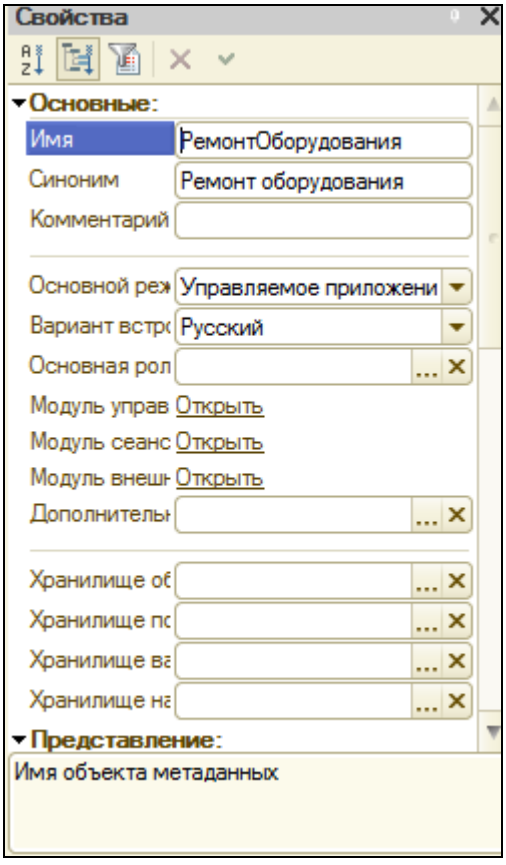


Рис. 9. Окно Свойства корня конфигурации

Задайте название новой конфигурации **РемонтОборудования**.

Автоматически сформировался синоним названия конфигурации **Ремонт оборудования**, который будет показан в заголовке основного окна в пользовательском режиме.

Проверим в работе выполненные выше действия, запустим приложение в режиме 1С: Предприятие. Выполним команду Сервис – 1С: Предприятие. Сохраним изменения. Запустится 1С: Предприятие.

Приступим к созданию объектов метаданных типа Справочник.

СОЗДАНИЕ СПРАВОЧНИКОВ

Любой справочник в системе «1С:Предприятие» имеет встроенные реквизиты «Код» и «Наименование». Встроенный реквизит «Код» будет присваиваться автоматически системой.

Создание справочника «Оборудование»

Данный справочник будет хранить информацию по оборудованию, которое сдается в ремонт клиентами.

1. Правой кнопкой мыши щелкнем по **Справочники** в конфигураторе (рис. 10) и выберем **Добавить.**

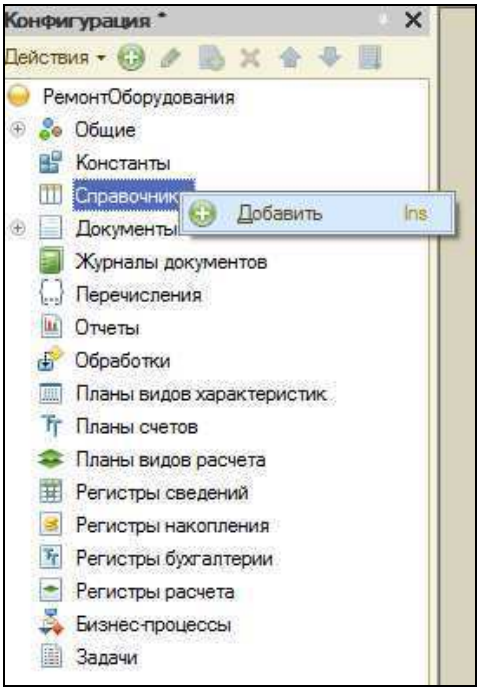


Рис. 10. Добавление нового Справочника

2. На экране появится форма **Конструктор справочника** и **Свойства**. Заполните их следующими данными (рис. 11):

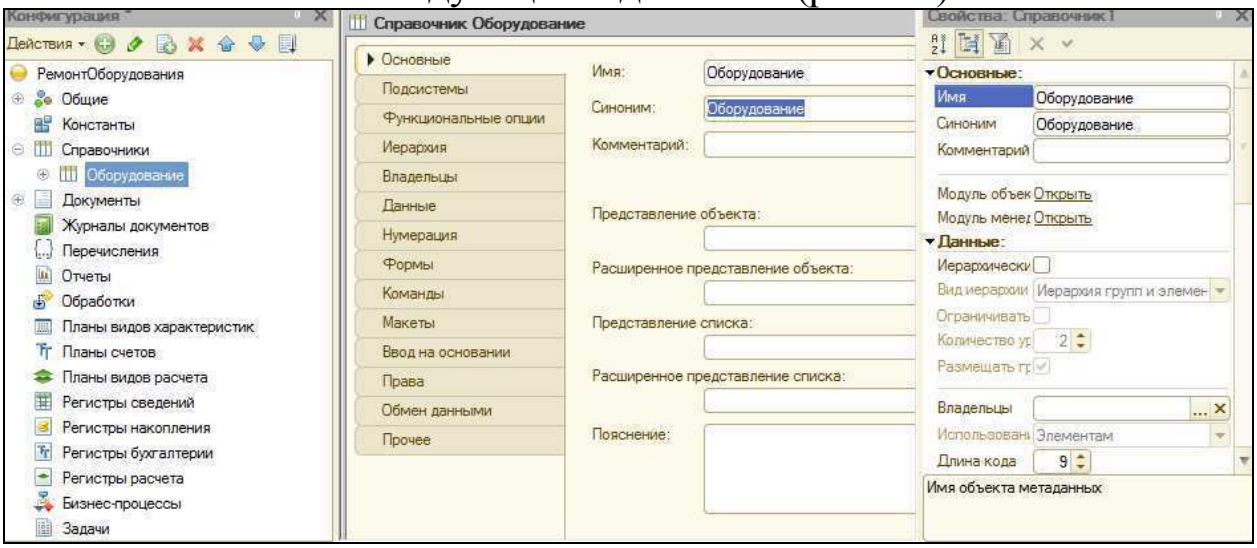


Рис. 11. Конструктор Справочника и окно Свойства

Заполните Имя – **Оборудование.**

3. На вкладке **Данные** зададим **Длину наименования** – **100** символов (рис. 12).

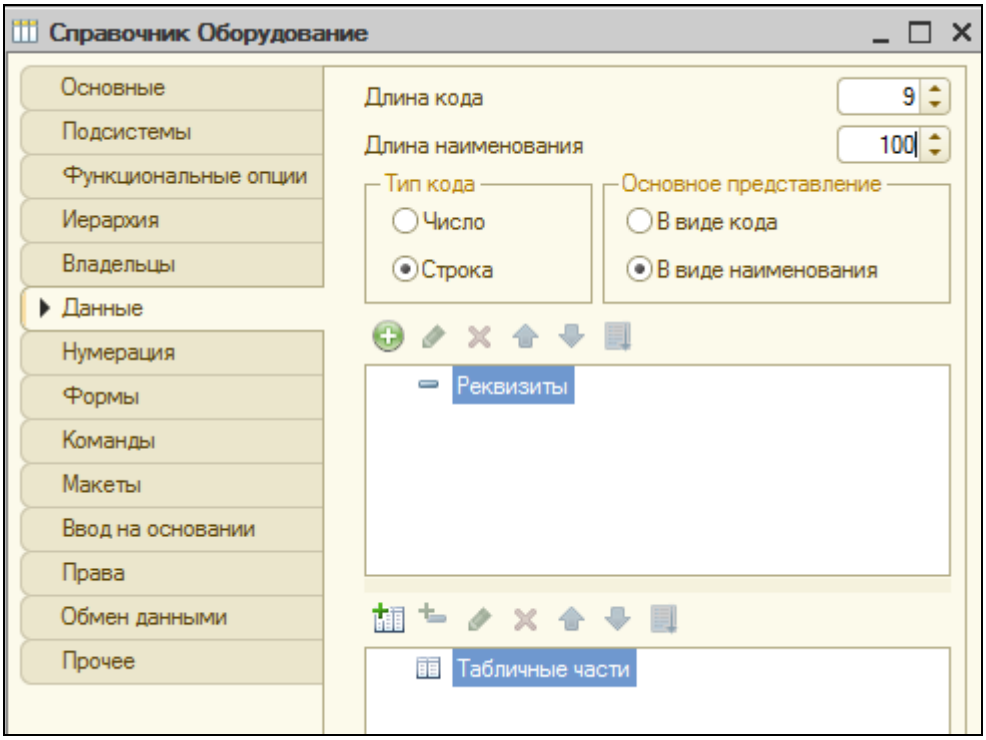


Рис. 12. Конструктор Справочника и вкладка Данные

Остальная стандартная реализация этого объекта полностью удовлетворяет заданию, поэтому все остальные параметры оставим без изменения. (Внимательно изучите данные формы).

Закройте форму (рис. 12).

Создание справочника «Клиенты»

Создадим новый справочник (рис. 13): имя – Клиенты.

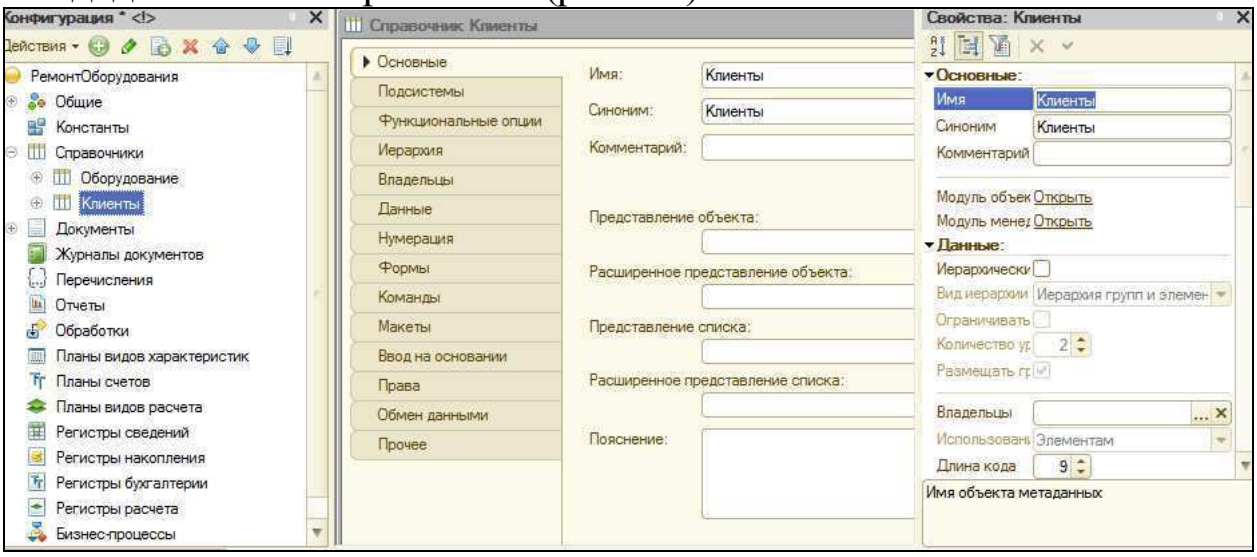


Рис. 13. Справочник Клиенты

Стандартная реализация этого объекта полностью удовлетворяет заданию, поэтому все остальные параметры оставим без изменения.

Закройте форму (рис. 13).

Создание справочника «Склады»

Ремонтная фирма, кроме центрального офиса, имеет еще филиал, расположенный в регионе. Поэтому необходимо знать, на какой склад приняли насос в ремонт и с какого склада клиент будет забирать оборудование после ремонта. Для этого создадим еще справочник Склады (рис. 14).

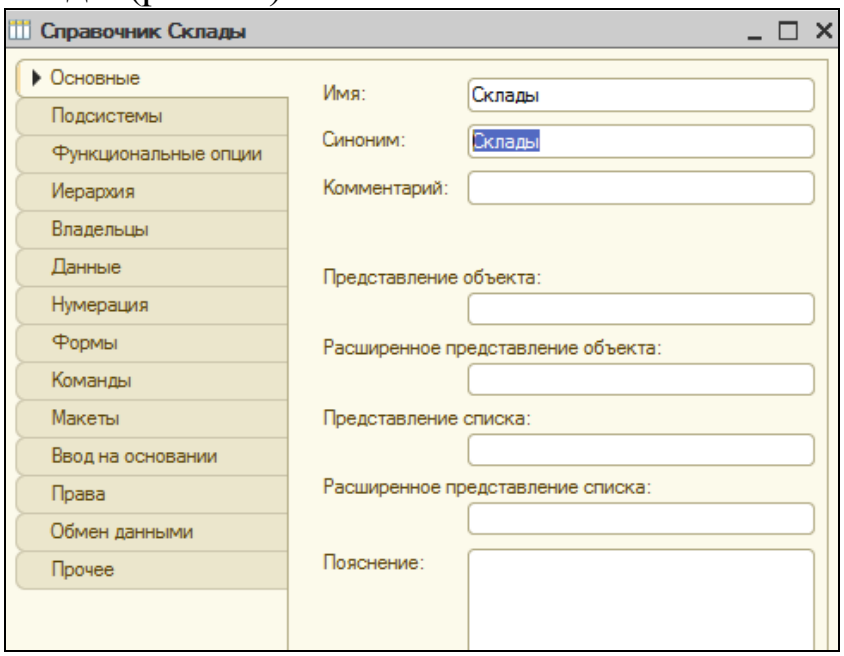


Рис. 14. Справочник Склады

Совершенно точно известно, что центральный офис существовал и будет далее существовать, потому что без него вся работа теряет смысл. А вот филиалы могут открываться и закрываться.

1. Добавим элемент **Главный склад** в справочник **Склады.** Элементы, добавляемые в конфигураторе, называются предопределенные.

Если обычные элементы пользователь может добавлять и удалять по своему желанию, то предопределенные элементы пользователь удалить не может. Эти элементы существуют всегда.

Благодаря этому на них могут быть завязаны какие-либо программные алгоритмы. (Эти возможности рассмотрим позже).

2. Переходим на закладку **Прочее** (рис. 15) и нажмем кнопку **Предопределенные.**

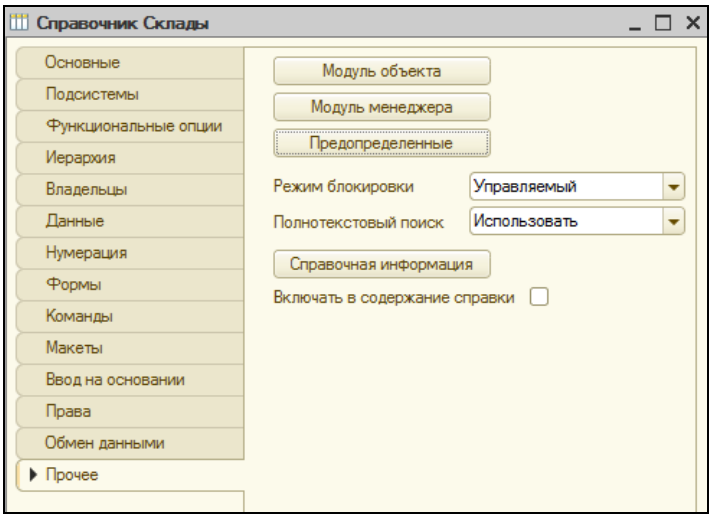


Рис. 15. Справочник Склады закладка Прочее

3. Добавим новый предопределенный элемент **ОсновнойСклад** (рис. 16, 16а).

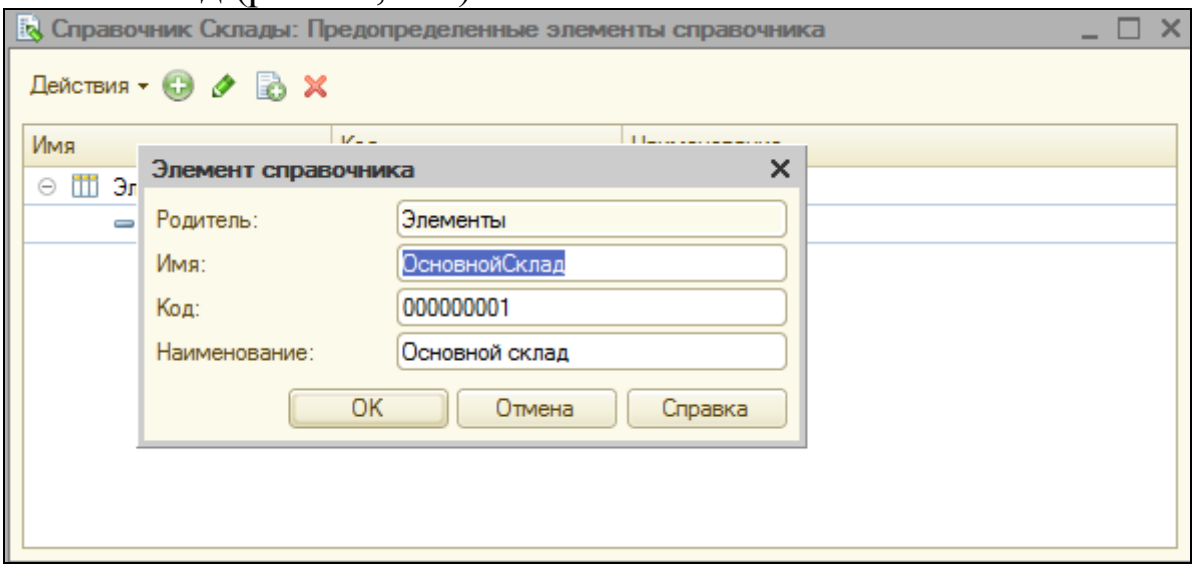


Рис. 16. Предопределенные элементы справочника

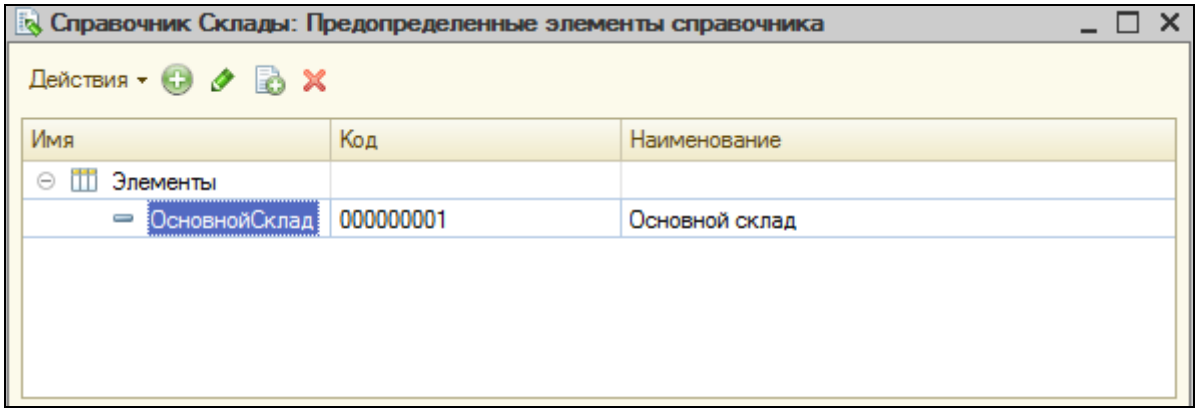


Рис. 16а. Предопределенные элементы справочника

СОЗДАНИЕ ДОКУМЕНТОВ

Нужно как-то фиксировать выполняемые менеджером действия:

1. принять насос в ремонт,
2. выдать клиенту отремонтированный насос.

Создадим два документа:

1. Акт приема в ремонт.
2. Акт выдачи из ремонта.

1. На закладке Документы в Дереве конфигуратора добавим документ **АктПриемаВРемонт** (рис. 17).

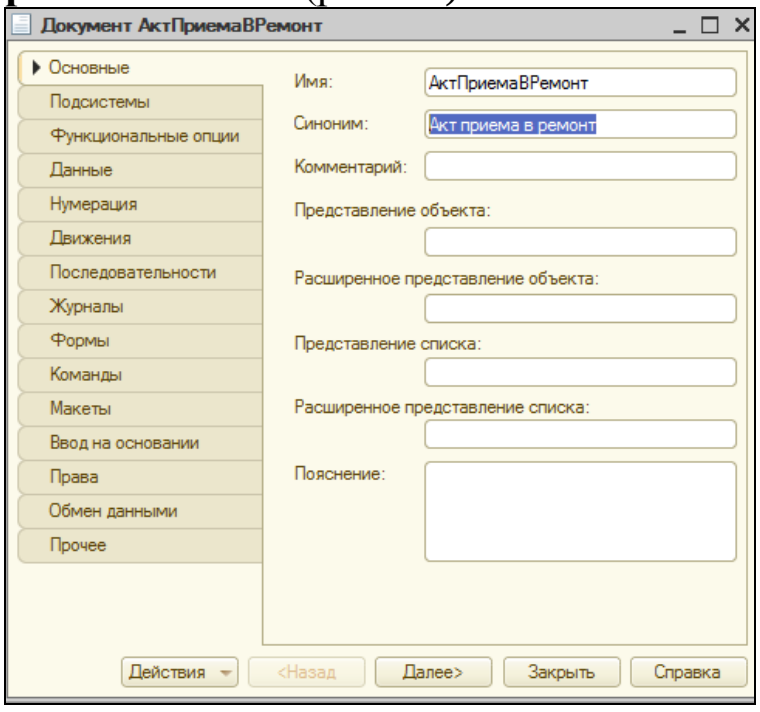


Рис. 17. Документ Акт приема в ремонт

2. Что необходимо знать, когда оборудование принимается в ремонт:

1. кто клиент,
2. какое оборудование,
3. на какой склад,
4. в чем неисправность.

Для этого переходим на закладку **Данные**. Добавляем новый реквизит – **Клиент**, тип данных – **Справочник.Ссылка.Клиенты**. Для этого нажмем на кнопку Добавить и введите необходимые данные(рис. 18, 18а).

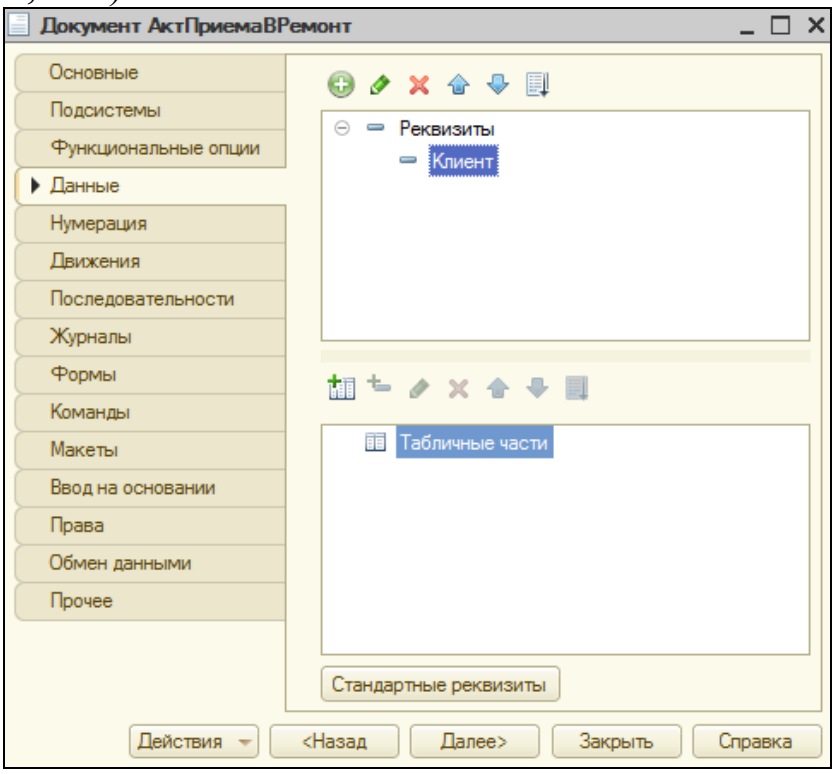


Рис. 18. Документ Акт приема в ремонт закладка Данные

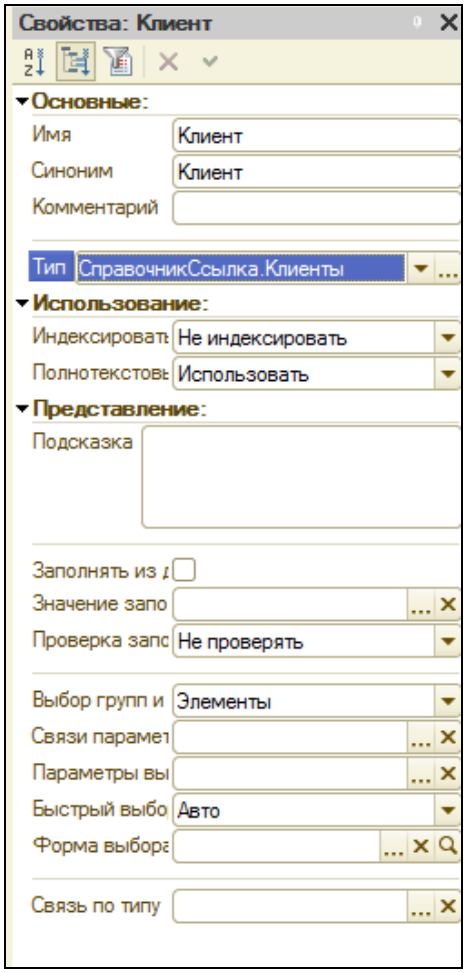


Рис. 18а. Документ Акт приема в ремонт закладка Данные Свойства реквизита Клиент

3. Аналогично создайте еще три новых **Реквизита документа**:

– Имя – **Оборудование**, Тип данных – **Справочник.Ссылка.Оборудование**

– Имя – **Склад**, Тип данных – **Справочник.Ссылка.Склады**

– Имя – **ОписаниеНеисправности**, Тип данных – **Строка**, Длина – 250. Чтобы удобнее было записывать, укажем **Многострочный режим**, **Расширенное редактирование**. (Это значит, что в форме этот реквизит будет отображаться как поле из нескольких строк. При нажатии кнопки Ввод будет происходить переход к следующей строке этого поля) (рис. 19а). Этот реквизит необходим, чтобы со слов клиента зафиксировать неисправность.

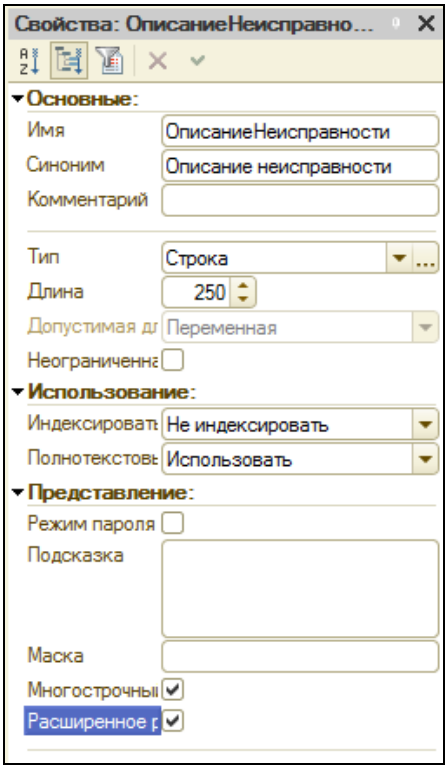


Рис. 19а. Документ Акт приема в ремонт закладка Данные Свойства реквизита ОписаниеНеисправности

На рис. 19б представлены созданные реквизиты Документа АктПриемаВРемонт.

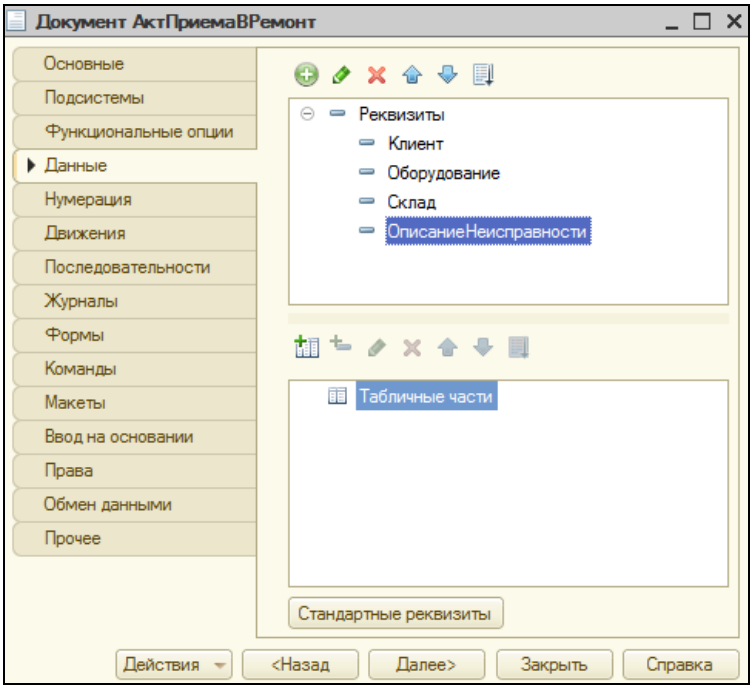


Рис. 19б. Документ Акт приема в ремонт закладка Данные

4. Создадим второй документ **Акт выдачи из ремонта**. В основном он будет аналогичен предыдущему. Необходимо указать клиента, оборудование, склад и какие ремонтные работы были выполнены. Т.е. те же реквизиты, что в Акте приема в ремонт. Для быстрого получения нового объекта воспользуемся копированием. Для копирования и вставки документа в конфигурацию используем стандартные клавиатурные команды.

Присвоим новому скопированному документу имя – **АктВыдачиИзРемонта.**

Реквизиты Клиент, Оборудование, Склад – остаются без изменений, реквизит Описание неисправностей – переименуем в **ВыполненныеРаботы.**

Проверим в работе созданные объекты, запустим приложение в режиме 1С: Предприятие. Выполним команду Сервис – 1С: Предприятие. Сохраним изменения. Запустится 1С: Предприятие.

Отчет по практической работе 6 должен содержать: цель работы; краткое описание использованных в работе элементов рассмотренной платформы; текст созданных процедур и результаты их работы; выводы по результатам практической работы. При оформлении отчета в печатном виде в нижний колонтитул следует поместить фамилию, инициалы и номер группы обучаемого (8 пт., Arial, выравнивание по правому краю).

Практическая работа №7
Особенности применения инструментальных средств информационных систем при решении научно-практических задач различного класса
(Семинар)

Темы докладов (+презентация):

- Инструментальные средства информационных систем управления предприятием (Oracle E-Business Suite)
- Система управления базами данных (Oracle MySQL Enterprise Edition)

- Инструментальные средства разработки PLM – систем (компании Siemens PLM Software)
- Инструментальные средства разработки PLM –систем (компании MSC Software Corporation)
- Инструментальные программные средства информационных систем управления предприятием (компания Baan Corporation)
- Инструментальные программные средства информационных систем управления предприятием (Microsoft Dynamics NAV)
- Мультимедийные вычислительные среды Wolfram Research (WolframAlpha)
- Система аналитических вычислений Maple
- Инструментальные средства организации, анализа и моделирования данных Statistics Toolbox (пакет расширения MATLAB)
- Инструментальные средства создания экспертных систем на основе нечеткой логики Fuzzy Logic Toolbox (пакет расширения MATLAB)
- Инструментальные средства проектирования, моделирования, разработки и визуализации нейронных сетей Neural Network Toolbox (пакет расширения MATLAB)
- Инструментальные средства для решения задач оптимизации Optimization Toolbox (пакет расширения MATLAB)
- Инструментальные средства моделирования и анализа финансовых и экономических систем статистическими методами Econometrics Toolbox (пакет расширения MATLAB)
- Инструментальные средства моделирования и проектирования систем массового обслуживания SimEvents (пакет расширения MATLAB)
- Инструментальные средства визуализации 3D – движения Simulink®3D Animation (пакет расширения MATLAB)

Список литературы

1. Советов, Б.Я. Информационные технологии [Текст]: учебник для прикладного бакалавриата / Б.Я. Советов, В.В. Цехановский. – 6-е изд., перераб. и доп. – М.: Издательство Юрайт, 2015. – 263 с.

2. Исаев, Г.Н., Информационные системы в экономике [Текст]/ Г.Н. Исаев. – М.: Омега-Л, 2011. – 462 с.

3. Информационные технологии [Текст]: учебник / Под ред. В.В. Трофимова. – М.: Юрайт, 2011. – 624 с.

4. Кислицин, Д.И. Инструментальные средства информационных систем. Внутреннее устройство ЭВМ [Текст] : учебное пособие/ Д.И. Кислицин. – Нижний Новгород: Изд-во ННГАСУ, 2011 – 143с.

5. Максимов, Н.В. Технические средства информатизации [Текст] : учебник/ Н. В. Максимов, Т. Л. Партыка, И. И. Попов. - 3-е изд., перераб. и доп. - М. : Форум, 2010. - 606 с.

6. Балдин, К.В Информационные системы в экономике [Текст]/ К.В. Балдин, В.Б. Уткин. - М.: Издательско-торговая корпорация “Дашков и К^{0}”, 2010. – 395 с.

7. Баин, А.М. Современные информационные технологии систем поддержки принятия решений [Текст]/ А.М Баин. – М.: ИД «ФОРУМ», 2009. – 240 с.

8. Информационный менеджмент [Текст]: учебник / Под ред. Н.М. Абдикеева. – М.: ИНФРА-М, 2010. – 400 с.