УДК 004.42
ББК 32.973
Т 38

Авторы-составители: Е. В. Коробейникова, ст. преподаватель;
С. М. Мовшович, канд. техн. наук, доцент;
О. А. Кравченко, канд. физ.-мат. наук, доцент

Рецензенты: В. И. Токочаков, канд. техн. наук, доцент кафедры информационных технологий Гомельского государственного технического университета им. П. О. Сухого;
Т. А. Заяц, ст. преподаватель кафедры информационно-вычислительных систем Белорусского торгово-экономического университета потребительской кооперации

Рекомендован к изданию научно-методическим советом учреждения образования «Белорусский торгово-экономический университет потребительской кооперации». Протокол № 4 от 12 апреля 2011 г.

Т 38 **Технологии** программирования : практикум для студентов специальности 1-26 03 01 «Управление информационными ресурсами» / авт.-сост. : Е. В. Коробейникова, С. М. Мовшович, О. А. Кравченко – Гомель : учреждение образования «Белорусский торгово-экономический университет потребительской кооперации», 2012. – 100 с.
ISBN 978-985-461-935-4

УДК 004.42
ББК 32.973

ISBN 978-985-461-935-4

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Курс «Технологии программирования» продолжает подготовку студентов специальности 1-26 03 01 «Управление информационными ресурсами» в области информационных технологий. Он призван систематизировать знания, полученные при изучении курса «Алгоритмизация и программирование», и привести их в соответствие с требованиями времени и действующими стандартами информационных систем и технологий. Объем часов, выделяемых для изучения курса, устанавливается в соответствии с типовым учебным планом специальности.

Целью изучения данной дисциплины является подготовка студентов к использованию современных информационных технологий в качестве инструмента для решения практических задач в социально-экономической сфере на высоком профессиональном уровне. При этом выпускники специальности ориентируются на участие в процессах разработки и внедрении изученных технологий в корпоративных информационных системах на уровне постановки задач, реализации ее решения и тестирования полученного программного продукта.

Задачами дисциплины являются приобретение студентами знаний, умений и навыков в области современных технологий программирования и программных инструментах для их практической реализации.

В результате выполнения лабораторных работ на базе системы программирования СП Delphi студенты должны приобрести навыки создания и применения следующих объектов и технологий:

- пользовательских классов в соответствии с основными принципами объектно-ориентированного программирования;
- библиотек динамической компоновки данных;
- технологии BDE;
- глобальной и локальной технологий обработки исключительных ситуаций;
- технологии ADO;
- технологии COM.

Лабораторный практикум базируется на умениях и навыках, полученных студентами при изучении дисциплин «Автоматизация экономических расчетов» и «Алгоритмизация и программирование».

ЗАДАНИЯ ЛАБОРАТОРНЫХ РАБОТ

Лабораторная работа 1
СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ КЛАССОВ

***Цель работы*:** получение навыков создания и использования собственного класса объектов.

Пример выполнения работы

Задание. Требуется разработать проект в СП Delphi, в котором должен быть создан класс *Прямоугольник*, состоящий из двух полей (ширина, высота), двух свойств, обеспечивающих доступ к этим полям, конструктора и двух методов: вычисление площади и периметра прямоугольника. На основе разработанного класса для заданных ширины и высоты каждого из двух прямоугольников проект должен определить, у какого прямоугольника большая площадь, а у какого – больший периметр.

Порядок выполнения работы

1. *Создание интерфейса проекта*. Заполним окно формы визуальными компонентами так, как это показано на рисунке 1.

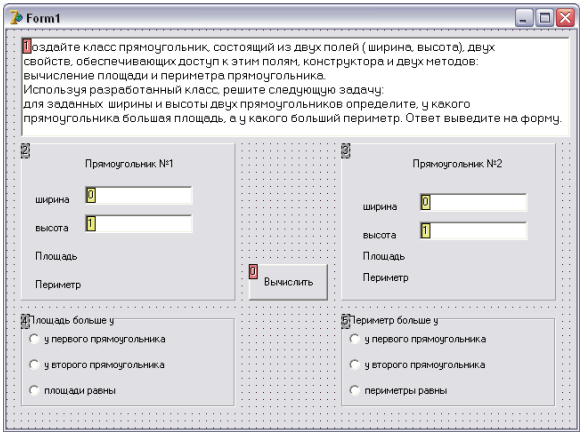

Рисунок 1 – **Интерфейс проекта создания и использования класса**

2. *Описание пользовательского класса*. В конец интерфейсной части проекта необходимо записать конструкции, описывающие создаваемый класс. На рисунке 2 приведено описание класса

Pramougolnik, включающее описание полей свойств и объявление методов класса. Кроме того, описаны переменные *a* и *b*, предназначенные для работы с двумя объектами – прямоугольниками.

```
  Pramougolnik=class //Описание класса
    private
      shir,vys:real; //Поля класса: ширина и высота
    published //Свойства класса
      property Tshir:Real read shir write shir;
      property TVys:real read vys write vys;
    public //Объявление методов класса
     Constructor Create (AOwner:TComponent);
     function Ploshad:real;
     function Perimetr:real;
  end;
var
  Form1: TForm1;
  a,b:Pramougolnik; //Описание объектов класса
```

Рисунок 2 – **Описание класса *Pramougolnik***

3. *Описание методов класса*. В начало исполняемой части проекта поместим описание трех методов проекта (рисунок 3).

```
implementation
 //Описание методов класса
constructor Pramougolnik.Create(Aowner:TComponent);
//Конструктор класса
 begin
   shir:=1;
   vys:=1;
 end;
function Pramougolnik.Ploshad:real;
//Метод: функция вычисления площади
  begin
    result:=shir*vys;
  end;
function Pramougolnik.Perimetr:real;
//Метод: функция вычисления периметра
  begin
    result:=2*(shir+vys);
  end;
{$R *.dfm}
```

Рисунок 3 – **Описание методов класса *Pramougolnik***

4. *Использование пользовательского класса*. Использование объектов созданного класса происходит в процедуре обработки щелчка по кнопке *Вычислить* (рисунок 4). Создаются объекты методом *Create*. Затем вводятся значения их полей из объектов *Edit1*, *Edit2*, *Edit3* и *Edit4*. С помощью функций вычисляются значения площади и периметра для каждого прямоугольника. В результате сравнения включается соответствующий переключатель. Обратите внимание, что два вычисленных значения принимаются равными, если абсолютная величина их разницы меньше 0,001.

```
procedure TForm1.Button1Click(Sender: TObject);
//Обработка щелчка по кнопке "Вычислить"
begin
  a:=Pramougolnik.Create(Self);
  b:=Pramougolnik.Create(Self);
  a.Tshir:=StrToFloat(Edit1.Text);
  a.Tvys:=StrToFloat(Edit2.Text);
  b.Tshir:=StrToFloat(Edit3.Text);
  b.Tvys:=StrToFloat(Edit4.Text);
  if Abs(a.Ploshad-b.Ploshad)<0.001
    then RadioGroup1.ItemIndex:=2
    else
      if a.Ploshad>b.Ploshad
        then RadioGroup1.ItemIndex:=0
        else RadioGroup1.ItemIndex:=1;
  if Abs(a.Perimetr-b.Perimetr)<0.001
    then RadioGroup2.ItemIndex:=2
    else
      if a.Perimetr>b.Perimetr
        then RadioGroup2.ItemIndex:=0
        else RadioGroup2.ItemIndex:=1;
  Edit5.Text:=FloatToStr(a.Ploshad);
  Edit6.Text:=FloatToStr(a.Perimetr);
  Edit7.Text:=FloatToStr(b.Ploshad);
  Edit8.Text:=FloatToStr(b.Perimetr);
end;
```

Рисунок 4 – **Процедура обработки щелчка по кнопке *Вычислить***

Индивидуальные задания

Вариант 1

Разработайте класс *Окружность*. Свойство: радиус. Методы: длина окружности и площадь круга, ограниченного окружностью.

На основе разработанного класса решите следующую задачу: для заданных радиусов двух окружностей определите, у какого круга большая площадь и насколько длина одной окружности отличается от другой. Ответ выведите на форму.

Формулы для расчета:

$$C = 2 \cdot \pi \cdot r; \quad S = \pi \cdot r^2,$$

где r – радиус окружности;

C – длина окружности;

S – площадь круга.

Вариант 2

Разработайте класс *Прямоугольный_Треугольник*. Свойства: длины двух катетов. Методы: длина гипотенузы и площадь треугольника.

На основе разработанного класса решите следующую задачу: для заданных длин катетов двух треугольников определите, у какого треугольника большая гипотенуза, а у какого – большая площадь. Ответ выведите на форму.

Формулы для расчета:

$$c = \sqrt{a^2 + b^2}; \quad S = \frac{a \cdot b}{2},$$

где a и b – длины катетов;

c – длина гипотенузы;

S – площадь прямоугольного треугольника.

Вариант 3

Разработайте класс *Куб*. Свойство: длина стороны. Методы: площадь поверхности и объем.

На основе разработанного класса решите следующую задачу: для заданных длин сторон двух кубов определите, у какого куба большая поверхность, и насколько объем одного куба больше другого. Ответ выведите на форму.

Формулы для расчета:

$$S = 6 \cdot a^2; \quad V = a^3,$$

где a – длина стороны куба;

S – площадь поверхности куба;

V – объем куба.

Вариант 4

Разработайте класс *Треугольник*. Свойства: длина одной стороны, величины двух прилежащих к заданной стороне углов. Методы: площадь треугольника, величина третьего угла.

На основе разработанного класса решите следующую задачу: для заданных длин сторон и величин углов двух треугольников определите, у какого треугольника большая площадь, а у какого – самый большой угол. Ответ выведите на форму.

Формулы для расчета:

$$\gamma = 180^\circ - \alpha - \beta; \quad S = \frac{a^2 \cdot \sin\alpha \cdot \sin\beta}{2 \cdot \sin(\alpha + \beta)},$$

где a – длина стороны треугольника;

α и β – углы, прилежащие к стороне a, град.;

γ – третий угол треугольника, град.;

S – площадь треугольника.

Вариант 5

Разработайте класс *Отрезок*. Свойство: координаты концов отрезка. Методы: длина отрезка, проверка параллельности оси OX.

На основе разработанного класса решите следующую задачу: для заданных координат концов двух отрезков определите, у какого отрезка большая длина, а какой из отрезков параллелен оси OX. Ответ выведите на форму.

Формула для расчета:

$$r = \sqrt{(x1 - x2)^2 + (y1 - y2)^2},$$

где ($x1$, $y1$) и ($x2$, $y2$) – координаты концов отрезка;

r – длина отрезка.

Условие параллельности отрезка оси OX: $y1 = y2$.

Вариант 6

Разработайте класс *Треугольник*. Свойства: координаты вершин треугольника. Методы: площадь треугольника, длина большей стороны.

На основе разработанного класса решите следующую задачу: для заданных координат вершин двух треугольников определите, у какого треугольника большая площадь, а у какого – самая большая длина стороны. Ответ выведите на форму.

Формулы для расчета:

$$a = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}; \quad b = \sqrt{(x2 - x3)^2 + (y2 - y3)^2};$$
$$c = \sqrt{(x1 - x3)^2 + (y1 - y3)^2}; \quad p = \frac{a + b + c}{2}; \quad S = \sqrt{p \cdot (p - a) \cdot (p - b) \cdot (p - c)},$$

где ($x1$, $y1$), ($x2$, $y2$) и ($x3$, $y3$) – координаты вершин треугольника;

a, b и c – длины сторон треугольника;

p – полупериметр треугольника;

S – площадь треугольника.

Вариант 7

Разработайте класс *Окружность*. Свойство: диаметр. Методы: длина окружности и площадь круга, ограниченного окружностью.

На основе разработанного класса решите следующую задачу: для заданных диаметров двух окружностей определите, у какого круга большая площадь и насколько длина одной окружности отличается от другой. Ответ выведите на форму.

Формулы для расчета:

$$C = \pi \cdot d; \quad S = \pi \cdot \frac{d^2}{4},$$

где d – диаметр окружности;

C – длина окружности;

S – площадь круга.

Вариант 8

Разработайте класс *Прямоугольник*. Свойства: координаты трех вершин прямоугольника. Методы: площадь прямоугольника и длина диагонали.

На основе разработанного класса решите следующую задачу: для заданных координат вершин двух прямоугольников определите, у какого прямоугольника большая диагональ, а у какого – большая площадь. Ответ выведите на форму.

Формулы для расчета:

$$a = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}; \quad b = \sqrt{(x2 - x3)^2 + (y2 - y3)^2};$$
$$d = \sqrt{(x1 - x3)^2 + (y1 - y3)^2}; \quad S = a \cdot b,$$

где ($x1$, $y1$), ($x2$, $y2$) и ($x3$, $y3$) – координаты вершин прямоугольника;

a и b – длины сторон прямоугольника;

d – длина диагонали прямоугольника;

S – площадь прямоугольника.

Лабораторная работа 2
ПРИНЦИП НАСЛЕДОВАНИЯ КЛАССОВ

***Цель работы*:** получение навыков использования принципа наследования при работе с собственным классом объектов.

Пример выполнения работы

Задание. На базе класса *Прямоугольник*, созданного в лабораторной работе 1, требуется создать класс *Новый_Прямоугольник* с добавлением в него метода вычисления длины диагонали прямоугольника. На основе разработанного класса для заданных ширины и высоты каждого из двух прямоугольников проект в СП Delphi должен определить, у какого прямоугольника большая площадь, у какого – больший периметр, а у какого – длинней диагональ.

Порядок выполнения работы

1. *Создание интерфейса проекта*. Скопируем папку с проектом предыдущей лабораторной работы, переименуем ее и добавим объекты в окно формы. Интерфейс разрабатываемого проекта приведен на рисунке 5.

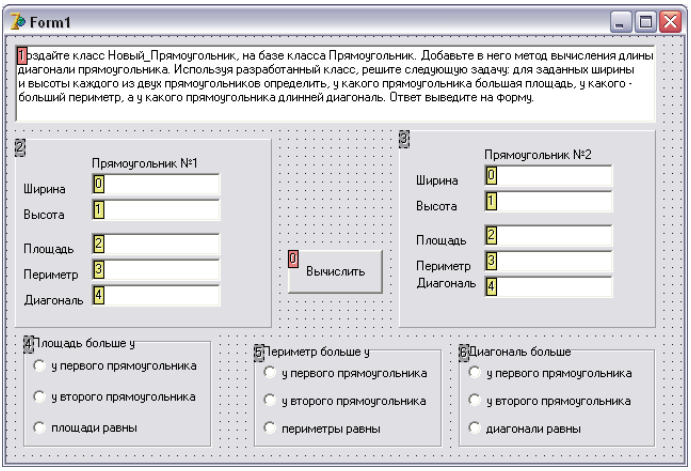

Рисунок 5 – **Интерфейс проекта наследования класса**

2. *Описание пользовательского класса*. В конец интерфейсной части проекта допишем конструкции, определяющие создаваемый класс. На рисунке 6 приведено описание класса *Pramougolnik_new*, объявляющее новый класс дочерним по отношению к классу *Pramougolnik*. В новом классе присутствует метод *Diagonal*, которого не было в родительском классе. Все остальные методы, равно как свойства и поля, новый метод унаследовал от класса *Pramougolnik*. Переменные *a* и *b*, предназначенные для работы с двумя прямоугольниками описаны как объекты класса *Pramougolnik_new*.

```
Pramougolnik_new=class(Pramougolnik)
{Определение нового класса как дочернего классу Pramougolnik
 с добавлением нового метода Diagonal}
   private
   public
   function Diagonal:real;
  end;
var
  Form1: TForm1;
  a,b:Pramougolnik_new;
```

Рисунок 6 – **Описание класса *Pramougolnik_new***

3. *Описание нового метода*. В исполняемую часть проекта добавим описание нового метода – функцию вычисления длины диагонали прямоугольника. Текст функции приведен на рисунке 7.

```
function Pramougolnik_new.Diagonal:real;
 begin
   result:=sqrt(sqr(shir)+sqr(vys));
 end;
```

Рисунок 7 – **Описание функции для вычисления длины диагонали прямоугольника**

4. *Использование созданного класса*. Добавим в процедуру обработки щелчка по кнопке *Вычислить* необходимые операторы. Полученный текст приведен на рисунке 8.

```
procedure TForm1.Button1Click(Sender: TObject);
//Обработка щелчка по кнопке "Вычислить"
begin
  a:=Pramougolnik_new.Create(Self);
  b:=Pramougolnik_new.Create(Self);
  a.Tshir:=StrToFloat(Edit1.Text);
  a.Tvys:=StrToFloat(Edit2.Text);
  b.Tshir:=StrToFloat(Edit3.Text);
  b.Tvys:=StrToFloat(Edit4.Text);
  if Abs(a.Ploshad-b.Ploshad)<0.001
    then RadioGroup1.ItemIndex:=2
    else
      if a.Ploshad>b.Ploshad
        then RadioGroup1.ItemIndex:=0
        else RadioGroup1.ItemIndex:=1;
  if Abs(a.Perimetr-b.Perimetr)<0.001
    then RadioGroup2.ItemIndex:=2
    else
      if a.Perimetr>b.Perimetr
        then RadioGroup2.ItemIndex:=0
        else RadioGroup2.ItemIndex:=1;
  if Abs(a.Diagonal-b.Diagonal)<0.001
    then RadioGroup3.ItemIndex:=2
    else
      if a.Diagonal>b.Diagonal
        then RadioGroup3.ItemIndex:=0
        else RadioGroup3.ItemIndex:=1;
  Edit5.Text:= FloatToStr(a.Ploshad);
  Edit6.Text:= FloatToStr(a.Perimetr);
  Edit7.Text:= FloatToStr(a.Diagonal);
  Edit8.Text:= FloatToStr(b.Ploshad);
  Edit9.Text:= FloatToStr(b.Perimetr);
  Edit10.Text:= FloatToStr(b.Diagonal);
end;
```

Рисунок 8 – **Процедура обработки щелчка по кнопке *Вычислить***

Индивидуальные задания

Вариант 1

Разработайте класс *Новая_Окружность* на базе класса *Окружность*. Добавьте в него два свойства, обеспечивающие доступ к полям, которые определяют координаты центра окружности, и метод вычисления принадлежности заданной точки кругу, ограниченному окружностью.

На основе разработанного класса решите следующую задачу: для заданных радиусов двух окружностей, координат их центров и координат произвольной точки определите, у какого круга большая площадь, насколько длина одной окружности отличается от другой и какому из кругов принадлежит заданная точка. Ответ выведите на форму.

Формула для расчета:

$$z = \sqrt{(x0 - x1)^2 + (y0 - y1)^2},$$

где z – расстояние от центра окружности с координатами ($x0$, $y0$) до точки ($x1$, $y1$).

Условие принадлежности точки ($x1$, $y1$) кругу:

$$z \le r,$$

где r – радиус круга.

Вариант 2

Разработайте класс *Новый_Прямоугольный_Треугольник* на базе класса *Прямоугольный_Треугольник*. Добавьте в него метод вычисления периметра прямоугольного треугольника.

На основе разработанного класса решите следующую задачу: для заданных длин катетов двух треугольников определите, у какого треугольника большая гипотенуза, у какого – большая площадь, а у какого треугольника больший периметр. Ответ выведите на форму.

Формула для расчета:

$$p = a + b + c,$$

где a и b – длины катетов;

c – длина гипотенузы;

p – периметр прямоугольного треугольника.

Вариант 3

Разработайте класс *Новый_Куб* на базе класса *Куб*. Добавьте в него три свойства, обеспечивающие доступ к полям, которые определяют координаты центра куба в пространстве, и метод вычисления расстояния от начала координат до центра куба.

На основе разработанного класса решите следующую задачу: для заданных длин сторон и координат центров двух кубов определите, у какого куба большая поверхность, насколько объем одного куба больше другого и какой куб расположен дальше от начала координат. Ответ выведите на форму.

Формула для расчета:

$$r = \sqrt{x^2 + y^2 + z^2},$$

где x, y и z – координаты центра куба;

r – расстояние от начала координат до центра куба.

Вариант 4

Разработайте класс *Новый_Треугольник* на базе класса *Треугольник*. Добавьте в него метод, который определяет, можно ли построить треугольник с заданной стороной и двумя прилежащими углами.

На основе разработанного класса решите следующую задачу: для заданных длин сторон и величин углов двух треугольников определите, существуют ли треугольники с заданными параметрами. Если такие треугольники существуют, то определите, у какого треугольника большая площадь, а у какого – самый большой угол. Ответ выведите на форму.

Условие существования треугольника:

$$\alpha + \beta < 180^\circ,$$

где α и β – заданные углы, град.

Вариант 5

Разработайте класс *Новый_Отрезок* на базе класса *Отрезок*. Добавьте в него метод, определяющий, в какой четверти координатной плоскости лежит точка с заданными координатами.

На основе разработанного класса решите следующую задачу: для заданных координат концов двух отрезков определите, у какого от-

резка большая длина, а какой из отрезков параллелен оси OX. Для каждого из концов отрезка определите номер четверти, в которой он лежит. Ответ выведите на форму.

Условия принадлежности точки с координатами (x, y) четверти координатной плоскости:

$$
\begin{aligned}
&x > 0 \text{ è } y > 0 \quad \rightarrow \quad 1 \div \text{åòâåðòü};\\
&x < 0 \text{ è } y > 0 \quad \rightarrow \quad 2 \div \text{åòâåðòü};\\
&x < 0 \text{ è } y < 0 \quad \rightarrow \quad 3 \div \text{åòâåðòü};\\
&x > 0 \text{ è } y < 0 \quad \rightarrow \quad 4 \div \text{åòâåðòü};\\
&x = 0 \quad \rightarrow \quad \text{îñü } OY;\\
&y = 0 \quad \rightarrow \quad \text{îñü } OX.
\end{aligned}
$$

Вариант 6

Разработайте класс *Новый_Треугольник* на базе класса *Треугольник*. Добавьте в него метод вычисления периметра треугольника.

На основе разработанного класса решите следующую задачу: для заданных координат вершин двух треугольников определите, у какого треугольника большая площадь, у какого – самая большая длина стороны, а у какого – самый большой периметр. Ответ выведите на форму.

Вариант 7

Разработайте класс *Новая_Окружность* на базе класса *Окружность*. Добавьте в него два свойства, обеспечивающие доступ к полям, которые определяют координаты центра окружности, и метод вычисления расстояния от центра окружности до начала координат.

На основе разработанного класса решите следующую задачу: для заданных диаметров двух окружностей определите, у какого круга большая площадь, насколько длина одной окружности отличается от другой и центр какой окружности ближе к началу координат. Ответ выведите на форму.

Формулы для расчета:

$$z = \sqrt{x^2 + y^2},$$

где z – расстояние от центра окружности с координатами (x, y) до начала координат.

Вариант 8

Разработайте класс *Новый_Прямоугольник* на базе класса *Прямоугольник*. Добавьте в него метод вычисления периметра прямоугольника.

На основе разработанного класса решите следующую задачу: для заданных координат вершин двух прямоугольников определите, у какого прямоугольника большая диагональ, у какого – большая площадь, а у какого – больший периметр. Ответ выведите на форму.

Формула для расчета:

$$P = 2 \cdot (a + b),$$

где a и b – длины сторон прямоугольника;

P – периметр прямоугольника.

Лабораторная работа 3
СОЗДАНИЕ И ВЫЗОВ ДИНАМИЧЕСКИХ БИБЛИОТЕК

***Цель работы*:** получение навыков создания и статического вызова подпрограмм динамических библиотек при разработке проектов.

Пример выполнения работы

***Задание*.** Требуется разработать проект в СП Delphi, который вводит два целочисленных массива, находит и выводит на форму суммы их элементов и их максимальные элементы. Проект должен использовать динамическую библиотеку, содержащую две функции. Первая функция предназначена для нахождения суммы элементов целочисленного массива. Вторая – для нахождения максимального элемента целочисленного массива.

Порядок выполнения работы

1. *Создание шаблона динамической библиотеки*. Выберем из меню СП Delphi команду *File / New/ Other*… . В появившемся окне *New*

Items на вкладке *New* выберем *DLL Wizard* и щелкнем по кнопке *OK*. В результате будет создан шаблон проекта новой динамической библиотеки (DLL) в окне редактора, который приведен на рисунке 9. Обратим внимание на то, что проект начинается зарезервированным словом *library*.

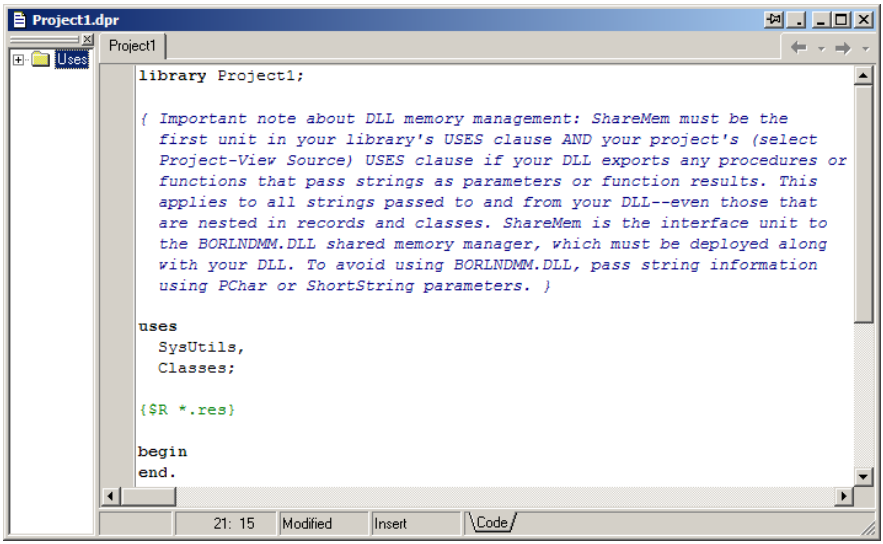

Рисунок 9 – **Шаблон новой динамической библиотеки**

Сохраним проект *Project1* в папке данной лабораторной работы.

2. *Заполнение шаблона описаниями подпрограмм*. Вставим в шаблон описание типа *mas* для целочисленного массива и описания подпрограмм-функций нахождения суммы элементов массива (*SumArray*) и максимального элемента массива (*MaxArray*). На рисунке 10 представлено описание библиотеки *Project1*.

В конце текста после описания функций находится раздел *exports*, содержащий перечень имен подпрограмм, которые будут экспортированы из DLL и могут вызываться другими приложениями. В нашем примере в DLL описаны две подпрограммы – *SumArray* и *MaxArray*, которые могут использоваться не только в разрабатываемом проекте, но и другими приложениями.

```
library Project1;
uses
  SysUtils,
  Classes;
{$R *.res}
type
  //Описание типа для целочисленного массива
  mas=array[1..20] of integer;
  //Описание функции нахождения суммы элементов массива
  function SumArray(a:mas;n:integer):integer;
    var
      s,i:integer;
    begin
      s:=0;
      for i:=1 to n do
        s:=s+a[i];
      SumArray:=s;
    end;
  //Описание функции нахождения максимального элемента массива
  function MaxArray(a:mas;n:integer):integer;
    var
      max,i:integer;
    begin
      max:=a[1];
      for i:=2 to n do
        if max<a[i]
          then max:=a[i];
      MaxArray:=max;
    end;
  exports
     SumArray, MaxArray;
begin
end.
```

Рисунок 10 – **Описание динамической библиотеки *Project1***

3. *Компиляция динамической библиотеки*. Чтобы использовать функции, описанные в DLL, мы должны скомпилировать проект нажатием комбинации клавиш *Ctrl + F9*. После сохранения в папке проекта будет создана динамическая библиотека под именем *Project1.dll*.

4. *Использование динамической библиотеки*. Для создания приложения, в котором будут использоваться подпрограммы разработанной DLL, выполним команду СП Delphi *File / New / Application*. Со-

храним файлы создаваемого приложения в той же папке, в которой была сохранена DLL. Интерфейс разрабатываемого приложения приведен на рисунке 11.

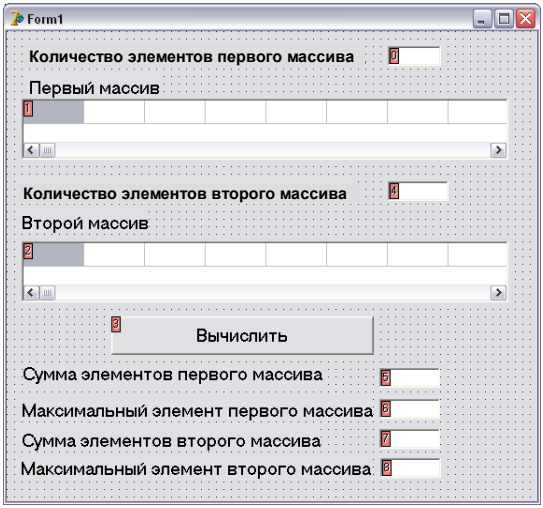

Рисунок 11 – **Интерфейс проекта создания и использования DLL**

В раздел *type* интерфейсной части проекта необходимо включить описание типа *mas* для целочисленного массива, а в раздел *var* – описание используемых функций созданной библиотеки в качестве внешних функций. Указанные описания приведены на рисунке 12.

```
type
  mas=array[1..20] of integer;
  TForm1 = class(TForm)
    Edit1: TEdit;

var
  Form1: TForm1;
  function SumArray(a:mas;n:integer):integer; external 'project1.dll';
  function MaxArray(a:mas;n:integer):integer; external 'project1.dll';
implementation
```

Рисунок 12 – **Описания в интерфейсной части проекта, использующего DLL**

Текст процедуры обработки щелчка по кнопке *Button1*, содержащей обращения к подпрограммам-функциям динамической библиотеки, приведен на рисунке 13.

```
procedure TForm1.Button1Click(Sender: TObject);
  var
     a,b:mas;
     i,n1,n2,s1,s2:integer;
     max1, max2:integer;
begin
  n1:=StrToInt(Edit1.Text);
  n2:=StrToInt(Edit2.Text);
  for i:=1 to n1 do
    a[i]:=StrToInt(StringGrid1.Cells[i-1,0]);
  for i:=1 to n2 do
    b[i]:=StrToInt(StringGrid2.Cells[i-1,0]);
  s1:=SumArray(a,n1);
  s2:=SumArray(b,n2);
  max1:= MaxArray(a,n1);
  max2:= MaxArray(b,n2);
  Edit3.Text:=IntToStr(s1);
  Edit4.Text:=IntToStr(max1);
  Edit5.Text:=IntToStr(s2);
  Edit6.Text:=IntToStr(max2);
end;
```

Рисунок 13 – **Текст процедуры, использующей DLL**

Индивидуальные задания

Вариант 1

Создать динамическую библиотеку, содержащую функции вычисления суммы элементов массива и минимального элемента массива.

Написать основную программу, которая вводит два массива и использует функции динамической библиотеки для нахождения суммы элементов в каждом массиве и для определения, в каком из двух массивов минимальный элемент больше.

Вариант 2

Создать динамическую библиотеку, содержащую функцию вычисления количества положительных элементов массива и функцию нахождения максимального элемента массива.

Написать основную программу, которая вводит два массива и использует функции динамической библиотеки для нахождения максимального элемента в каждом массиве и для определения, в каком из двух массивов количество положительных элементов больше.

Вариант 3

Создать динамическую библиотеку, содержащую функцию вычисления произведения элементов массива и функцию нахождения максимального элемента второй половины массива.

Написать основную программу, которая вводит два массива и использует функции динамической библиотеки для нахождения максимального элемента во второй половине каждого массива и для определения, в каком из двух массивов произведение элементов больше.

Вариант 4

Создать динамическую библиотеку, содержащую функцию определения количества нулевых элементов массива и функцию нахождения максимального элемента первой половины массива.

Написать основную программу, которая вводит два массива и использует функции динамической библиотеки для нахождения максимального элемента в первой половине каждого массива и для определения, у какого массива количество нулевых элементов больше.

Вариант 5

Создать динамическую библиотеку, содержащую функцию вычисления среднего арифметического элементов массива и функцию нахождения минимального элемента второй половины массива.

Написать основную программу, которая вводит два массива и использует функции динамической библиотеки для нахождения минимального элемента во второй половине каждого массива и для определения, у какого массива среднее арифметическое элементов больше.

Вариант 6

Создать динамическую библиотеку, содержащую функцию вычисления произведения отрицательных элементов массива и функцию нахождения минимального элемента первой половины массива.

Написать основную программу, которая вводит два массива и использует функции динамической библиотеки для нахождения минимального элемента в первой половине каждого массива и для определения, у какого массива произведение отрицательных элементов больше.

Вариант 7

Создать динамическую библиотеку, содержащую функцию вычисления произведения четных элементов массива и функцию нахождения положения первого максимального элемента массива.

Написать основную программу, которая вводит два массива и использует функции динамической библиотеки для нахождения положения первого максимального элемента каждого массива и для определения, у какого массива произведение четных элементов больше.

Вариант 8

Создать динамическую библиотеку, содержащую функцию вычисления суммы элементов массива, расположенных на четных местах, и функцию нахождения положения первого минимального элемента массива.

Написать основную программу, которая вводит два массива и использует функции динамической библиотеки для нахождения положения первого минимального элемента каждого массива и для определения, у какого массива сумма элементов, расположенных на четных местах, больше.

Лабораторная работа 4
ПОСЛЕДОВАТЕЛЬНЫЙ ДОСТУП К ДАННЫМ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ BDE

***Цель работы*:** получение навыков использования технологии BDE для обеспечения доступа к данным из приложений БД на примере СУБД Paradox.

Задание. Требуется разработать проект в СП Delphi, который позволяет просматривать таблицы БД Paradox, редактировать их содержимое и выполнять поиск данных путем последовательного просмотра.

Пример выполнения работы

1. *Описание структуры таблицы в системе Paradox*. Вызовем систему Paradox по команде *Пуск / Программы / Borland Delphi 7 / Database Desctop / File / New / Table*. Создадим в системе Paradox таблицу *Ученики* со следующими полями: номер ученика, фамилия, класс, адрес, год рождения, средний балл (рисунок 14).

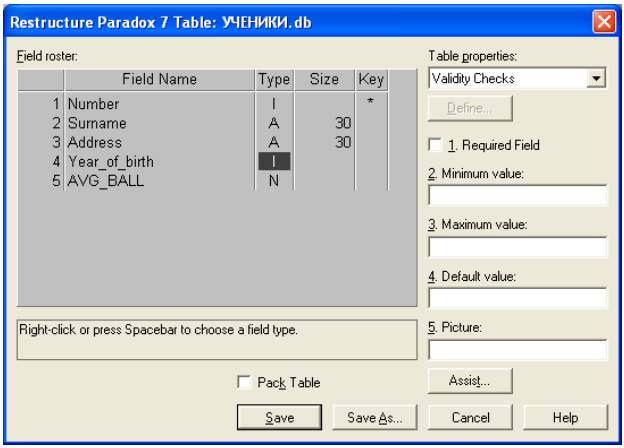

Рисунок 14 – **Структура таблицы *Ученики* в системе Paradox**

2. *Заполнение таблицы БД информацией.* Заполним таблицу БД информацией так, как это показано на рисунке 15, и закроем БД.

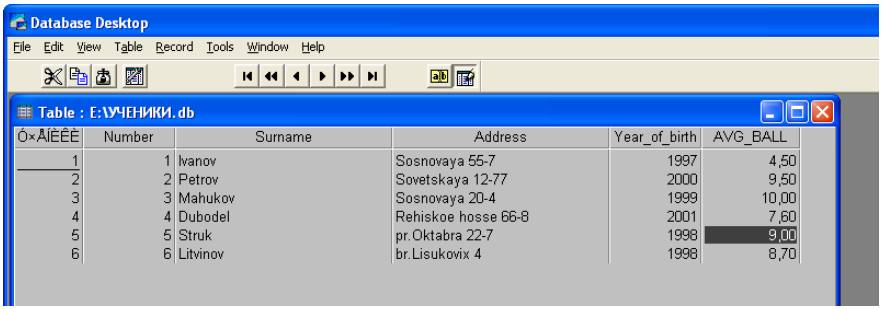

Рисунок 15 – **Содержимое таблицы *Ученики* в системе Paradox**

3. *Создание интерфейса проекта*. Вызовем СП Delphi и поместим на форму компонент *TTable* из палитры компонентов BDE, который обеспечивает соединение с таблицей *Ученики*.

Установим следующие значения для свойств этого компонента:

1. Значение *True* для свойства *Active*.

2. Укажем путь к БД Paradox для свойства *DatabaseName*.

3. Выберем из выпадающего списка имя таблицы *Ученики.db* для свойства *TableName*.

Поместим на форму компонент *DataSource1* (страница *Data Access* палитры компонентов). Установим значение *Table1* для свойства *DataSet* этого компонента.

Поместим на форму компонент *DBGrid* (страница *Data Controls* палитры компонентов), который представляет данные в виде таблицы. В столбцах таблицы размещаются поля набора данных, а в строках – записи. Связь с набором данных устанавливается значением *DataSource1* свойства *DataSource*.

Компонент *DBNavigator* (страница *Data Controls* палитры компонентов) предназначен для перемещения по записям набора данных. Связь с набором данных устанавливается значением *DataSource1* свойства *DataSource*.

Окно формы приведено на рисунке 16.

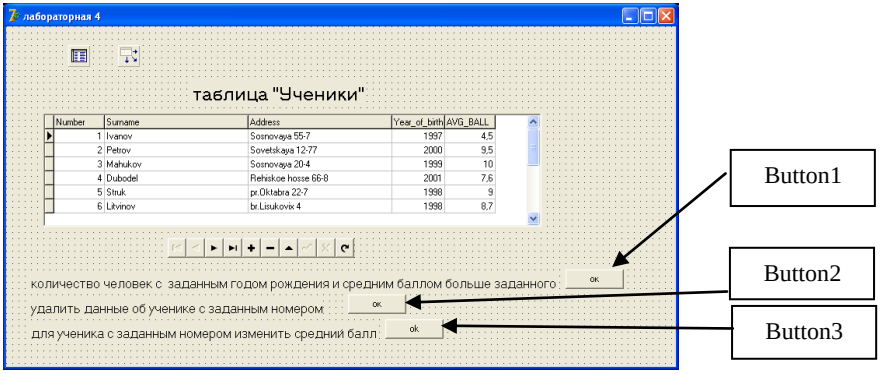

Рисунок 16 – **Интерфейс проекта**

4. *Создание процедуры на выборку.* Создадим процедуру обработки щелчка по кнопке *Button1* для подсчета количества человек с заданным годом рождения и средним баллом выше заданного значения. Текст процедуры приведен на рисунке 17.

```
procedure TForm1.Button1Click(Sender: TObject);
//Запрос на выборку
var
  s1,s2:string;
  kol:integer;
begin
   s1:=InputBox('Ввод','Введите год рождения','');
   s2:=InputBox('Ввод','Введите средний балл','');
   Table1.First;
   kol:=0;
   while not Table1.Eof do
     begin
       if (Table1.FieldByName('AVG_ball').AsFloat>StrToFloat(s2))
           and (Table1.FieldByName('Year_of_birth').AsInteger=StrToInt(s1))
         then
           kol:=kol+1;
      Table1.Next;
     end;
   ShowMessage(Количество учеников с годом рождения ' +s1+
               'и средним баллом большим ' + s2 +' равно '+IntToStr(kol));
end;
```

Рисунок 17 – **Текст процедуры, реализующей запрос на выборку**

5. *Создание процедуры удаления данных*. Создадим процедуру обработки щелчка по кнопке *Button2* для удаления данных об ученике с заданным номером. Текст процедуры приведен на рисунке 18.

```
procedure TForm1.Button2Click(Sender: TObject);
//Запрос на удаление
 var
  s1:string;
begin
   s1:=InpuBbox('Ввод','Введите номер ученика','');
   Table1.First;
   while not Table1.Eof do
     begin
       if Table1.FieldByName('Number').AsInteger=StrToInt(s1)
         then
           Table1.Delete
         else
           Table1.Next;
     end;
   Table1.Refresh;
end;
```

Рисунок 18 – **Текст процедуры, реализующей запрос на удаление**

6. *Создание процедуры обновления данных*. Создадим процедуру обработки щелчка по кнопке *Button3* для изменения среднего балла ученику с заданным номером. Текст процедуры приведен на рисунке 19.

```
procedure TForm1.Button3Click(Sender: TObject);
//Запрос на обновление
var
  s1,s2:string;
begin
  s1:=InputBox('Ввод','Введите номер ученика','');
  s2:=InputBox('Ввод','Введите его новый средний балл','');
  Table1.First;
  while not Table1.Eof do
    begin
     if Table1.FieldByName('Number').AsInteger=StrToInt(s1)
       then
          begin
            Table1.Edit;
            Table1.FieldByName('AVG_ball').AsFloat:=StrToFloat(s2);
            Table1.Post;
          end;
      Table1.Next;
    end;
  Table1.Refresh;
end;
```

Рисунок 19 – **Текст процедуры, реализующей запрос на обновление**

Индивидуальные задания

Вариант 1

Создайте в системе Paradox базу данных со следующими полями:

- номер ученика;
- фамилия;
- класс;
- адрес;
- год рождения;
- средний балл.

С использованием технологии BDE подключитесь к этой базе данных из СП Delphi. Создайте процедуры, реализующие в режиме диалога следующие функции для работы с базой данных:

- вычислить количество человек в заданном классе со средним баллом выше заданного;

- удалить данные о конкретном ученике;
- для заданного ученика обновить все сведения;
- вычислить средний балл у учеников заданного года рождения.

Вариант 2

Создайте в системе Paradox базу данных со следующими полями:
- код товара;
- наименование товара;
- цена, р.;
- дата производства;
- срок годности, дни;
- производитель;
- количество, кг.

С использованием технологии BDE подключитесь к этой базе данных из СП Delphi. Создайте процедуры, реализующие в режиме диалога следующие функции для работы с базой данных:
- вычислить количество просроченного товара;
- удалить данные о конкретном товаре;
- для заданного кода товара обновить все сведения;
- вычислить среднюю цену товара заданного производителя.

Вариант 3

Создайте в системе Paradox базу данных со следующими полями:
- код книги;
- название книги;
- автор;
- жанр;
- издательство;
- цена, р.;
- количество, штук.

С использованием технологии BDE подключитесь к этой базе данных из СП Delphi. Создайте процедуры, реализующие в режиме диалога следующие функции для работы с базой данных:
- вычислить среднюю цену книг заданного автора;
- удалить данные о книгах заданного автора;
- для заданного кода книги обновить все сведения;
- вычислить количество книг заданного жанра, выпущенных заданным издательством.

Вариант 4

Создайте в системе Paradox базу данных со следующими полями:

- код больного;
- ФИО больного;
- адрес;
- место работы;
- день рождения;
- ФИО участкового врача.

С использованием технологии BDE подключитесь к этой базе данных из СП Delphi. Создайте процедуры, реализующие в режиме диалога следующие функции для работы с базой данных:

- вычислить количество пациентов у заданного врача;
- удалить данные о заданном больном;
- для заданного кода больного обновить все сведения;
- вычислить количество больных с заданным годом рождения.

Вариант 5

Создайте в системе Paradox базу данных со следующими полями:

- код студента;
- ФИО студента;
- факультет;
- группа;
- специальность;
- средний балл;
- год поступления.

С использованием технологии BDE подключитесь к этой базе данных из СП Delphi. Создайте процедуры, реализующие в режиме диалога следующие функции для работы с базой данных:

- вычислить средний балл студентов заданного факультета;
- удалить данные о студентах заданной группы;
- для заданного кода студента обновить все сведения о нем;
- на заданном факультете вычислить количество студентов, поступивших в заданный год.

Вариант 6

Создайте в системе Paradox базу данных со следующими полями:

- код работника;
- ФИО работника;

- год поступления на работу;
- подразделение;
- должность;
- оклад, р.;
- образование.

С использованием технологии BDE подключитесь к этой базе данных из СП Delphi. Создайте процедуры, реализующие в режиме диалога следующие функции для работы с базой данных:

- вычислить количество человек со стажем более 10 лет;
- удалить данные о людях, работающих в заданном подразделении;
- для заданного кода работника обновить все сведения о нем;
- вычислить количество человек в заданном подразделении, работающих на заданной должности.

Вариант 7

Создайте в системе Paradox базу данных со следующими полями:

- код заказа;
- ФИО заказчика;
- адрес;
- рубрика раздела объявлений;
- текст объявления;
- дата выхода объявления

С использованием технологии BDE подключитесь к этой базе данных из СП Delphi. Создайте процедуры, реализующие в режиме диалога следующие функции для работы с базой данных:

- вычислить количество объявлений в заданной рубрике в заданную дату;
- удалить данные о заказах в заданную дату;
- для заданного кода заказа дату выхода сдвинуть на неделю вперед;
- вычислить количество объявлений у заданного заказчика.

Вариант 8

Создайте в системе Paradox базу данных со следующими полями:

- код группы;
- название группы;
- количество студентов в группе;
- название дисциплины;
- дата экзамена;

• количество студентов, не допущенных к экзамену;
• количество неудовлетворительных оценок на экзамене.

С использованием технологии BDE подключитесь к этой базе данных из СП Delphi. Создайте процедуры, реализующие в режиме диалога следующие функции для работы с базой данных:

• вычислить процент сдачи экзамена заданной группой по заданному предмету;
• удалить данные о экзаменах в заданную дату;
• для заданного экзамена для заданной группы изменить количество не допущенных к экзамену на заданное значение;
• вычислить количество экзаменов у заданной группы.

Лабораторная работа 5
ФИЛЬТРАЦИЯ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ BDE

***Цель работы*:** поиск данных в БД Paradox с применением механизмов фильтрации.

Задание. В проект, разработанный в лабораторной работе 4, добавьте функции, реализующиеся через механизмы фильтрации.

Пример выполнения работы

1. *Модификация интерфейса проекта*. Добавим в окно формы кнопки и надписи, приведенные на рисунке 20.

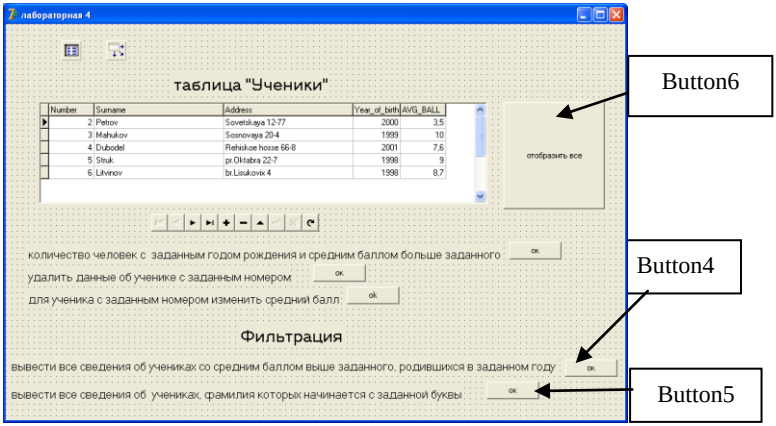

Рисунок 20 – **Окно формы проекта с фильтрацией данных**

2. *Создание процедуры фильтрации по числовому параметру*. Создадим процедуру обработки щелчка по кнопке *Button4* для отбора сведений об учениках со средним баллом выше заданного, родившихся в заданном году. Текст процедуры приведен на рисунке 21.

```
procedure TForm1.Button4Click(Sender: TObject);
//Фильтрация по числовому параметру
var
  s1,s2:string;
begin
  s1:=InputBox('Ввод','Введите год рождения','');
  s2:=InputBox('Ввод','Введите средний балл','');
  Table1.Filtered:=False;
  Table1.FilterOptions:=[FoCaseInsensitive];
  Table1.Filter:='Year_of_birth='+s1+' and AVG_ball>'+s2;
  Table1.Filtered:=True;
end;
```

Рисунок 21 – **Текст процедуры, реализующей фильтрацию данных по числовому параметру**

3. *Создание процедуры фильтрации по текстовому параметру*. Создадим процедуру обработки щелчка по кнопке *Button5* для отбора сведений об учениках, фамилии которых начинаются с заданных букв. Текст процедуры приведен на рисунке 22.

```
procedure TForm1.Button5Click(Sender: TObject);
//Фильтрация по текстовому параметру
var s1:string;
begin
  s1:=InputBox('Ввод','Введите Фио','');
  Table1.Filtered:=False;
  Table1.FilterOptions:=[FoCaseInsensitive];
  Table1.Filter:='Surname='''+s1+'*''';
  Table1.Filtered:=True;
end;
```

Рисунок 22 – **Текст процедуры, реализующей фильтрацию данных по текстовому параметру**

4. *Создание процедуры отмены фильтрации*. Создадим процедуру обработки щелчка по кнопке *Button6* для отмены фильтрации. Текст процедуры приведен на рисунке 23.

```
procedure TForm1.Button6Click(Sender: TObject);
//Отмена фильтрации
begin
  Table1.Filtered:=false;
end;
```

Рисунок 23 – **Текст процедуры для отмены фильтрации**

Индивидуальные задания

Вариант 1

Разработать процедуры, реализующие отмену фильтрации и фильтрацию данных по следующим условиям:

- вывести все сведения об учениках заданного класса со средним баллом выше заданного;
- вывести все сведения об ученике с заданной фамилией.

Вариант 2

Разработать процедуры, реализующие отмену фильтрации и фильтрацию данных по следующим условиям:

- вывести все сведения о просроченных товарах;
- вывести все сведения о товарах заданного производителя в заданной ценовой категории.

Вариант 3

Разработать процедуры, реализующие отмену фильтрации и фильтрацию данных по следующим условиям:

- вывести все сведения о книгах заданного автора;
- вывести все сведения о книгах заданного жанра в заданном ценовом диапазоне.

Вариант 4

Разработать процедуры, реализующие отмену фильтрации и фильтрацию данных по следующим условиям:

- для заданного врача вывести все сведения о пациентах, родившихся после заданного года;
- вывести все сведения о заданном пациенте.

Вариант 5

Разработать процедуры, реализующие отмену фильтрации и фильтрацию данных по следующим условиям:

- вывести все сведения о студентах заданного факультета, поступивших в заданный год;
- вывести сведения о заданном студенте.

Вариант 6

Разработать процедуры, реализующие отмену фильтрации и фильтрацию данных по следующим условиям:

- вывести все сведения о работниках заданного подразделения, работающих на заданной должности;
- вывести все сведения о заданном работнике.

Вариант 7

Разработать процедуры, реализующие отмену фильтрации и фильтрацию данных по следующим условиям:

- вывести все сведения о заказах заданной рубрики в заданную дату;
- вывести все сведения о заданном заказе.

Вариант 8

Разработать процедуры, реализующие отмену фильтрации и фильтрацию данных по следующим условиям:

- вывести все сведения об экзаменах для заданной группы;
- вывести все сведения о заданном экзамене для всех групп, в которых были получены двойки на экзаменах.

Лабораторная работа 6
ОБРАБОТКА ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ

***Цель работы*:** повышение надежности программ за счет использования глобального и локального обработчиков исключительных ситуаций.

Задание. Требуется разработать проект в СП Delphi, который в режиме диалога открывает заранее подготовленный текстовый файл, содержащий набор целых чисел, считывает эти числа в компонент *StringGrid* и выполняет над ними действия в соответствии с вариантом индивидуального задания.

Следует составить два варианта программы: с использованием глобального обработчика исключительных ситуаций и с использованием локального обработчика исключительных ситуаций.

Обе программы должны обрабатывать следующие исключения:

- отсутствует заданный файл;
- данные в файле не являются целыми числами;
- заданное число не является целым;
- в файле нет чисел, удовлетворяющих заданному условию.

Порядок выполнения работы

Порядок выполнения работы рассмотрим на примере вычисления среднего арифметического тех чисел из текстового файла, которые меньше заданного числа и находятся на четных местах.

1. *Создание текстового файла*. В текстовом редакторе *Блокнот* создадим текстовый файл, содержащий целые числа. Каждое число должно находиться в отдельной строке. Сохраним файл в ту папку, в которой впоследствии будет находиться проект СП Delphi. Пример файла приведен на рисунке 24.

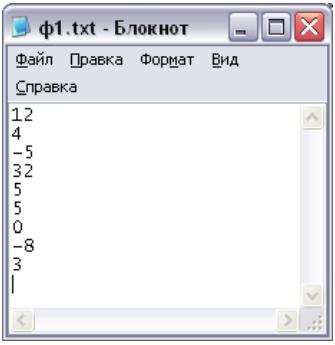

Рисунок 24 – **Текстовый файл с целыми числами**

2. *Отключение обработчика событий редактора СП Delphi*. После вызова СП Delphi и сохранения пока еще пустого проекта снимите флажок *Integrated debugging* в диалоговом окне команды *Tools / Debugger Optios*. Это делается с целью отключения отладчика СП Delphi от обработки исключительных ситуаций.

3. *Создание интерфейса проекта с глобальным обработчиком исключительных ситуаций*. Поместим в окно формы объекты, показанные на рисунке 25.

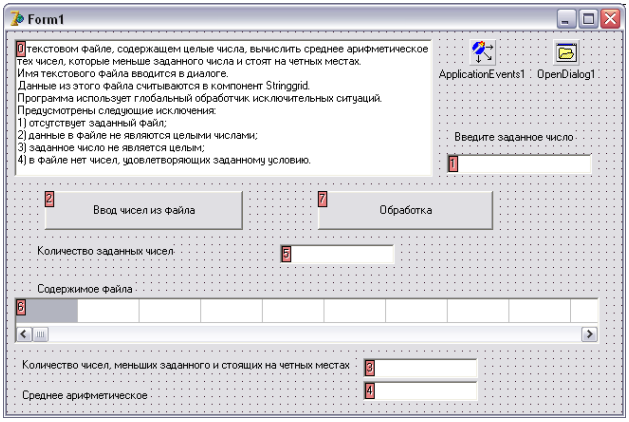

Рисунок 25 – **Окно формы проекта с глобальным обработчиком исключительных ситуаций**

Объект *Application Events1* находится на странице *Additional* палитры компонентов. Для события *onException* этого объекта надо в инспекторе объектов указать метод *ApplicationEvents1Exception*. Указанный метод является глобальным обработчиком исключительных ситуаций.

Объект *OpenDialog1* находится на странице *Dialogs* палитры компонентов. В качестве значения свойства *FileName* этого объекта надо указать путь к созданному текстовому файлу.

В окне формы находятся две командные кнопки: одна служит для считывания данных из текстового файла в компонент *StringGrid1*, а другая – для вычисления среднего арифметического тех чисел из текстового файла, которые меньше заданного числа и находятся на четных местах.

4. *Разработка процедуры глобальной обработки исключительных ситуаций*. Шаблон процедуры *ApplicationEvents1Exception* следует заполнить операторами, обеспечивающими вывод сообщений об исключительных ситуациях (рисунок 26). Сообщения выводятся в диалоговые окна с помощью процедуры *ShowMessage*.

```
procedure TForm1.ApplicationEvents1Exception(Sender: TObject;
  E: Exception);
//Обработка исключительных ситуаций
begin
  if E is EMathError
    then ShowMessage('В массиве нет подходящих чисел.'+
                     ' Ошибка деления на 0.');
  if E is EInOutError
    then
      begin
        if EInOutError(E).ErrorCode=2
          then ShowMessage('Ошибка открытия файла.');
        if EInOutError(E).ErrorCode=106
          then ShowMessage('Ошибка чтения данных из файла.');
      end;
  if E is EConvertError
    then  ShowMessage('Ошибка переобразования типа.');
end;
```

Рисунок 26 – **Процедура глобального обработчика исключительных ситуаций**

5. *Разработка процедуры ввода чисел из файла*. Процедура, обеспечивающая ввод данных из текстового файла, их подсчет и занесение в компонент *StringGrid1* приведена на рисунке 27.

```
var
  Form1: TForm1;
  m:integer; //Для считывания числа из файла
  n:integer; //Количество чисел в файле
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
{Ввод информации из текстового файла
 в Edit2 и StringGrid1}
begin
  if OpenDialog1.Execute
    then
      begin
        AssignFile(input,OpenDialog1.FileName);
        Reset(input);
        n:=0;
        while not(EOF(input)) do
          begin
            Readln(input,m);
            StringGrid1.Cells[n,0]:=IntToStr(m);
            n:=n+1;
          end;
        StringGrid1.ColCount:=n;
        Edit2.Text:=IntToStr(n);
        CloseFile(input);
      end;
end;
```

Рисунок 27 – **Процедура ввода данных из текстового файла**

6. *Разработка процедуры обработки массива*. Процедура вычисления среднего арифметического тех чисел из текстового файла, которые меньше заданного числа и находятся на четных местах в таблице *StringGrid1*, приведена на рисунке 28.

```
procedure TForm1.Button2Click(Sender: TObject);
{Нахождение среднего арифметического чисел, меньших
 заданного числа и стоящих на четных местах}
 var
   a:array[1..20]of integer;
   b,i,k,sm:integer;
   sr:real;
begin
  b:=StrToInt(Edit1.Text);
  for i:=1 to n do
    a[i]:=StrToInt(StringGrid1.Cells[i-1,0]);
  k:=0;
  sm:=0;
  i:=2;
  while i<=n do
    begin
      if a[i]<b
        then
          begin
            sm:=sm+a[i];
            k:=k+1
          end;
      i:=i+2
    end;
  sr:=sm/k;
  Edit3.Text:=IntToStr(k);
  Edit4.Text:=FloatToStr(sr);
end;
```

Рисунок 28 – **Процедура обработки чисел из текстового файла в проекте с глобальным обработчиком исключительных ситуаций**

7. *Тестирование проекта с глобальным обработчиком исключительных ситуаций*. Для тестирования созданного проекта с глобальным обработчиком исключительных ситуаций надо не только варьировать числа в текстовом файле и заданное число, но и проверить программную обработку каждого из четырех запланированных исключений. Таким образом, надо проверить следующие случаи:

- в диалогом окне задания имени текстового файла указать имя не существующего файла, а затем имя файла, не являющегося текстовым;
- в текстовый файл занести вещественное число, а затем строку;
- в качестве заданного числа задать вещественное число;
- подобрать такое заданное число, чтобы количество чисел в файле, меньших его и стоящих на четных местах, было равно нулю.

8. *Создание интерфейса проекта с локальным обработчиком исключительных ситуаций.* Для создания проекта с локальным обработчиком исключительных ситуаций скопируем папку проекта с глобальным обработчиком и переименуем ее. Из окна формы удалим объект *Application Events1*, а из текста модуля – процедуру обработки исключительных ситуаций.

9. *Разработка процедуры ввода чисел из файла с локальной обработкой ошибок*. Существующий текст процедуры (рисунок 27), обеспечивающей ввод данных из текстового файла, их подсчет и занесение в компонент *StringGrid1*, надо включить в блок *try-except-end* так, как это показано на рисунке 29.

```
procedure TForm1.Button1Click(Sender: TObject);
{Ввод информации из текстового файла в Edit5 и StringGrid1}
var
  m:integer;
  n:integer;
begin
  try
    if OpenDialog1.Execute
      then
        begin
          AssignFile(input,OpenDialog1.FileName);
          Reset(input);
          n:=0;
          while not(EOF(input)) do
            begin
              Readln(input,m);
              StringGrid1.Cells[n,0]:=IntToStr(m);
              n:=n+1;
            end;
          StringGrid1.ColCount:=n;
          Edit2.Text:=IntToStr(n);
          CloseFile (input)
        end;
  except
    on EO:EConvertError do
       ShowMessage('Ошибка преобразования  '+ EO.Message);
    on EO: EInOutError do
      begin
        if EO.ErrorCode=2
          then ShowMessage('Ошибка открытия файла');
        if EO.ErrorCode=106
          then  ShowMessage('Ошибка чтения данных из файла');
      end
    else ShowMessage('Ошибка не определена');
  end;
end;
```

Рисунок 29 – **Процедура ввода данных из текстового файла с локальной обработкой исключительных ситуаций**

10. *Разработка процедуры обработки чисел из файла с локальной обработкой ошибок*. Аналогично преобразуется текст процедуры обработки элементов массива, представленный на рисунке 28. Полученный текст приведен на рисунке 30.

```
procedure TForm1.Button2Click(Sender: TObject);
{Нахождение среднего арифметического чисел, меньших
 заданного числа и стоящих на четных местах}
 var
   a:array[1..20]of integer;
   b,i,k,n,sm:integer;
   sr:real;
begin
  try
    n:=StrToInt(Edit2.Text);
    b:=StrToInt(Edit1.Text);
    for i:=1 to n do
      a[i]:=StrToInt(StringGrid1.Cells[i-1,0]);
    k:=0;
    sm:=0;
    i:=2;
    while i<=n do
      begin
        if a[i]<b
          then
            begin
              sm:=sm+a[i];
              k:=k+1
            end;
       i:=i+2
      end;
    sr:=sm/k;
    Edit3.Text:=IntToStr(k);
    Edit4.Text:=FloatToStr(sr);
  except
    on EMathError do
       begin
         ShowMessage('Попытка деления на 0');
         Edit4.SetFocus; Edit4.Text:='Ошибка';
       end;
    on EO:EConvertError do
       ShowMessage('Ошибка преобразования  '+ EO.Message);
    else ShowMessage('Ошибка не определена');
  end;
end;
```

Рисунок 30 – **Процедура обработки чисел из текстового файла в проекте с локальным обработчиком исключительных ситуаций**

11. *Тестирование проекта с локальным обработчиком исключительных ситуаций*. Тестирование проекта с локальным обработчиком событий следует провести для тех же тестов, что и проект с глобальным обработчиком (см. пункт 7).

Индивидуальные задания

Вариант 1

Вычислить среднее арифметическое положительных чисел, содержащихся в текстовом файле и меньших заданного числа.

Вариант 2

Вычислить среднее арифметическое чисел, содержащихся в текстовом файле в записях с четными номерами и меньших заданного числа.

Вариант 3

Вычислить среднее арифметическое чисел, содержащихся в текстовом файле и больших заданного числа.

Вариант 4

Вычислить среднее арифметическое четных чисел, содержащихся в текстовом файле и больших заданного числа.

Вариант 5

Вычислить среднее арифметическое чисел, содержащихся в текстовом файле и принадлежащих заданному промежутку $[x, y]$.

Вариант 6

Вычислить среднее арифметическое чисел, содержащихся в текстовом файле, стоящих после первого максимального элемента и меньших заданного числа.

Вариант 7

Вычислить среднее арифметическое отрицательных чисел, содержащихся в текстовом файле и больших заданного числа.

Вариант 8

Вычислить среднее арифметическое чисел, содержащихся в текстовом файле, кратных 3 и больших заданного числа.

Лабораторная работа 7
ЗАПОЛНЕНИЕ, ПРОСМОТР И РЕДАКТИРОВАНИЕ ИНФОРМАЦИИ В ТАБЛИЦАХ БАЗЫ ДАННЫХ СУБД ACCESS НА ОСНОВЕ ТЕХНОЛОГИИ ADO

***Цель работы*:** получение навыков использования технологии ADO для обеспечения универсального доступа к данным из приложений БД на примере СУБД Access.

Задание. Для указанного преподавателем варианта создайте в СУБД MS Access базу данных из четырех таблиц. Создайте в СП Delphi проект, позволяющий заполнять, просматривать и редактировать данные из таблиц БД.

Порядок выполнения работы

1. *Создание БД в MS Access*. В среде СУБД MS Access опишем структуру четырех таблиц и создадим схему данных БД *Клиенты-Заказы*. В таблице *Клиенты* содержатся сведения о клиентах торговой фирмы, в таблице *Товары* – сведения о продаваемых товарах и их наличии на складе фирмы. В таблицах *Заказы* и *СтрокиЗаказов* содержатся сведения из заказов на поставки, пример соответствующего документа приведен на рисунке 31.

Заказ № 4 от 04.03.2011 г.
Клиент – ОАО «Гомельские баранки»

Наименование товара	Количество
Желе вишневое	12
Желе апельсиновое	4

Рисунок 31 – **Пример документа «Заказ клиента»**

Структура таблиц БД приведена на рисунке 32, а схема данных – на рисунке 33.

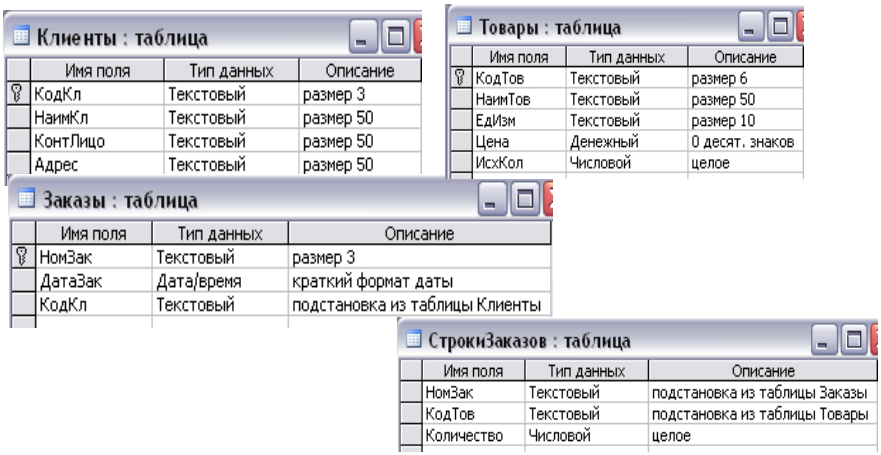

Рисунок 32 – **Структура таблиц БД *Клиенты-Заказы***

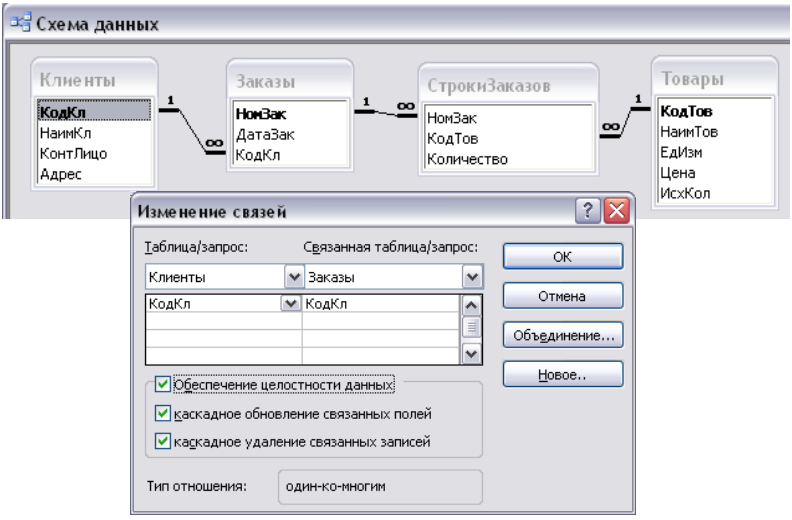

Рисунок 33 – **Схема данных БД *Клиенты-Заказы***

2. *Создание интерфейса проекта*. Создадим интерфейс проекта в соответствии с образцом, приведенном на рисунке 34.

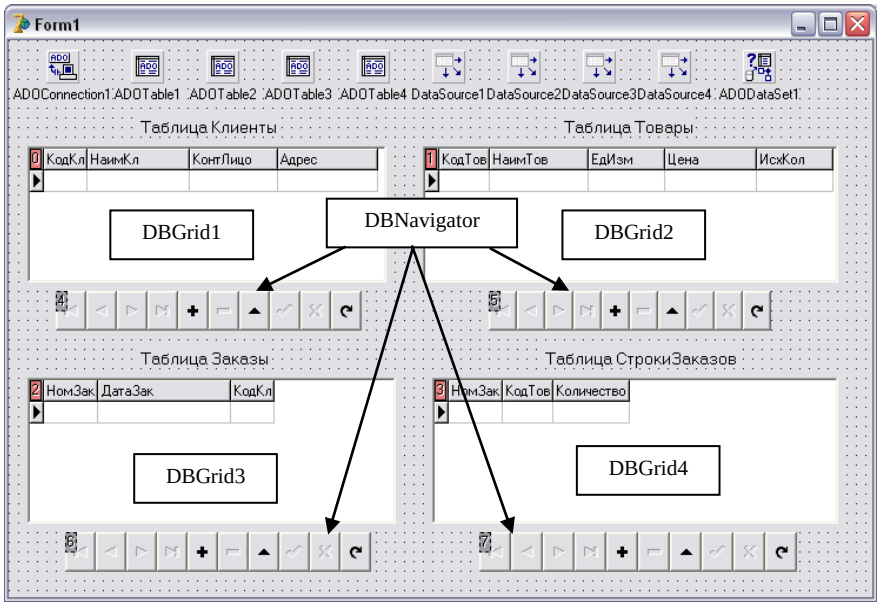

Рисунок 34 – **Окно формы проекта использования БД *Клиенты-Заказы***

Поместим в окно формы компонент *ADOConnection* (страница *ADO* палитры компонентов), который обеспечивает соединение приложения с хранилищем данных. Установим следующие значения для свойств этого компонента:

• *False* для свойства *LoginPromt* (беспарольный доступ к БД).

• Для свойства *ConnectedString* надо сначала указать поставщика данных *MS Jet 4.0 OLE DB Provider* по команде…/ *Build* / *Поставщик данных*, затем по команде / *Подключение* необходимо указать путь к БД Access и по команде *Проверка подключения* проверить подключение, после чего последовательно три раза щелкнуть по кнопке *OK*.

• *True* для свойства *Connected*.

Поместим в окно формы четыре компонента *ADOTable* (страница *ADO* палитры компонентов), которые обеспечат использование в проекте Delphi всех таблиц БД, подключенных через *ADO*. Установим следующие значения для свойств каждого из этих компонентов:

• *ADOConnection1* для свойства *Connection*.

• Для свойства *TableName* надо выбрать из списка одну из таблиц БД (*ADOTable1 – Клиенты, ADOTable2 – Товары, ADOTable3 – Заказы, ADOTable4 – СтрокиЗаказов*).

• *ClUseServer* для свойства *CursorLocation*.

• *True* для свойства *Active*.

На следующем этапе разработки интерфейса необходимо перенести на форму и настроить четыре компонента *DataSource* (страница *Data Access* палитры компонентов). Для связи набора данных с компонентом *DataSource* используется свойство *DataSet*, в качестве значения которого надо указать *ADOTable1*, *ADOTable2*, *ADOTable3* и *ADOTable4* соответственно.

Компонент *DBGrid* (страница *Data Controls* палитры компонентов) представляет данные в виде таблицы. В столбцах таблицы размещаются поля набора данных, а в строках – записи. Связь с набором данных устанавливается значением *DataSource1* (2, 3 или 4) для свойства *DataSource*. Итак, надо установить четыре таких компонента.

Поместим в окно формы четыре компонента *DBNavigator* (страница *Data Controls* палитры компонентов) для перемещения по записям каждого из наборов данных. Связь с набором данных устанавливается значением *DataSource1* (2, 3 или 4) для свойства *DataSource*.

Наконец, поместим в окно формы компонент *ADODataSet1*, который обеспечит ввод значений внешнего ключа в подчиненную таблицу из списка значений первичного ключа главной таблицы. Установим значение *ADOConnection1* для свойства *Connection*.

Заметим, что при выполнении проекта в окне приложения из перечисленных выше компонентов будут видны только *DBGrid* и *DBNavigator*.

3. *Ввод информации в таблицы, не содержащие внешних ключей*. Запустим проект на выполнение и с помощью кнопок компонента *DBNavigator* занесем пять записей в таблицу *Клиенты* и восемь записей в таблицу *Товары* (рисунок 35). Завершим выполнение проекта и убедимся в СУБД Access, что вся информация, введенная при выполнении проекта СП Delphi, занесена в таблицы БД.

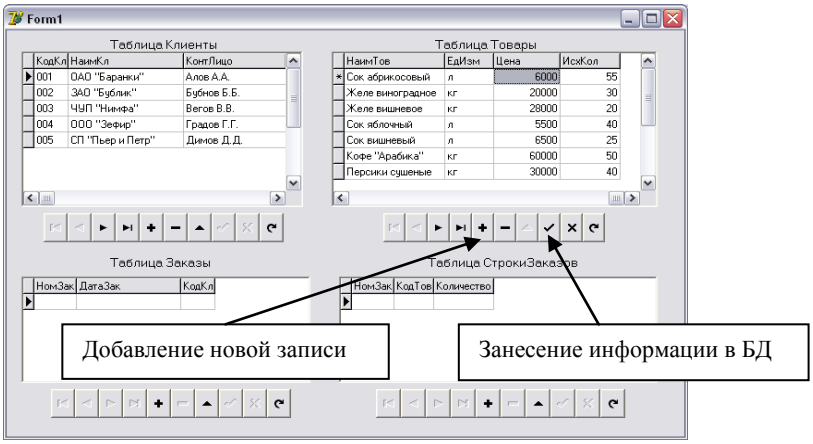

Рисунок 35 – **Занесение информации в БД с использованием навигатора**

4. *Создание процедур для ввода значений внешних ключей*. В нашей базе данных три внешних ключа:

- поле *КодКл* в таблице *Заказы* – значения этого поля должны выбираться из поля *КодКл* таблицы *Клиенты*;
- поле *НомЗак* в таблице *СтрокиЗаказов* – значения этого поля должны выбираться из поля *НомЗак* таблицы *Заказы*;
- поле *КодТов* в таблице *СтрокиЗаказов* – значения этого поля должны выбираться из поля *КодТов* таблицы *Товары*.

Для создания списков значений первичных ключей и занесения информации из них в поля внешних ключей создадим процедуру *ObnovlenieSpiska*. Заголовок процедуры надо поместить в интерфейсную часть модуля (рисунок 36), а текст ее в исполняемую часть модуля (рисунок 37).

```
var
  Form1: TForm1;
  procedure ObnovlenieSpiska;
implementation
{$R *.dfm}
procedure ObnovlenieSpiska;
{обновление списков выбора для полей внешних ключей}
 begin
```

Рисунок 36 – **Описание процедуры в интерфейсной части модуля**

```
procedure ObnovlenieSpiska;
{Обновление списков выбора для полей внешних ключей}
begin
  with Form1 do
    begin
      //Очистка списков для внешних полей
      DBGrid3.Columns[2].PickList.Clear;
      DBGrid4.Columns[0].PickList.Clear;
      DBGrid4.Columns[1].PickList.Clear;
      //Связывание поля КодКл таблицы Заказы с таблицей Клиенты
      ADODataSet1.Active:=False;
      ADODataSet1.CommandText:='Select КодКл From Клиенты';
      ADODataSet1.Active:=True;
      While not ADODataSet1.Eof do
        begin
          DBGrid3.Columns[2].PickList.Add(ADODataSet1.FieldValues['КодКл']);
          ADODataSet1.Next;
        end;
      //Связывание поля НомЗак таблицы СтрокиЗаказов с таблицей Заказы
      ADODataSet1.Active:=False;
      ADODataset1.CommandText:='Select НомЗак From Заказы';
      ADODataSet1.Active:=True;
      While not ADODataSet1.Eof do
        begin
          DBGrid4.Columns[0].PickList.Add(ADODataSet1.FieldValues['НомЗак']);
          ADODataSet1.Next;
        end;
      //Связывание поля КодТов таблицы СтрокиЗаказов с таблицей Товары
      ADODataSet1.Active:=False;
      ADODataset1.CommandText:='Select КодТов From Товары';
      ADODataSet1.Active:=True;
      While not ADODataSet1.Eof do
        begin
          DBGrid4.Columns[1].PickList.Add(ADODataSet1.FieldValues['КодТов']);
          ADODataSet1.Next;
        end;
    end;
end;
```

Рисунок 37 – **Процедура обновления списков для внешних ключей**

Запуск процедуры *ObnovlenieSpiska* надо выполнять при возникновении следующих событий:

- при активации формы;
- при изменении значений поля *КодКл* в таблице *Клиенты* – в этом случае надо после обновления списка обновить содержимое таблицы *Заказы*;
- при изменении значений поля *КодТов* в таблице *Товары* – в этом случае надо после обновления списка обновить содержимое таблицы *СтрокиЗаказов*;

• при изменении значений поля *НомЗак* в таблице *Заказы* – в этом случае надо после обновления списка обновить содержимое таблицы *СтрокиЗаказов*.

Таким образом, необходимо создать следующие три процедуры:

• процедуру *Form1Activate*, вызываемую при активации формы;

• процедуру *ADOTable1AfterDelete*, вызываемую при удалении (*AfterDelete*) или после редактирования (*AfterPost*) информации в таблице *Клиенты* (*ADOTable1*);

• процедуру *ADOTable2AfterDelete*, вызываемую при удалении (*AfterDelete*) или после редактирования (*AfterPost*) информации в таблице *Товары* (*ADOTable2*), а также при аналогичных действиях в таблице *Заказы* (*ADOTable3*).

Тексты перечисленных процедур приведены на рисунке 38.

```
procedure TForm1.FormActivate(Sender: TObject);
//Запуск процедуры обновления списков при активации формы
begin
   ADOConnection1.Connected:=True;
   Adotable1.Active:=True;
   ADOTable2.Active:=True;
   ADOTable3.Active:=True;
   ADOTable4.Active:=True;
   ObnovlenieSpiska;
end;
procedure TForm1.ADOTable1AfterDelete(DataSet: TDataSet);
{После удаления (AfterDelete) и редактирования (AfterPost)
 информации в таблице Клиенты}
begin
  //Обновление списков подстановки
  ObnovlenieSpiska;
  //Обновление информации в таблице Заказы
  ADOTable3.Refresh;
end;
procedure TForm1.ADOTable2AfterDelete(DataSet: TDataSet);
{После удаления (AfterDelete) и редактирования (AfterPost)
 информации в таблице Товары, а также
 после удаления (AfterDelete) и редактирования (AfterPost)
 информации в таблице Заказы}
begin
  //Обновление списков подстановки
  ObnovlenieSpiska;
  //Обновление информации в таблице СтрокиЗаказов
  ADOTable4.Refresh;
end;
```

Рисунок 38 – **Процедуры, вызывающие обновление списков для внешних ключей**

5. *Ввод информации в таблицы, содержащие внешние ключи*. Запустим проект на выполнение и заполним таблицы *Заказы* и

СтрокиЗаказов информацией с придуманных пяти заказов, в каждом из которых содержится от одной до трех строк с наименованиями товаров и их количеством. При заполнении полей внешних ключей будем использовать раскрывающиеся списки с соответствующими значениями ключей главных таблиц. Окно приложения в процессе заполнения с отображением списков значений первичных ключей приведено на рисунке 39.

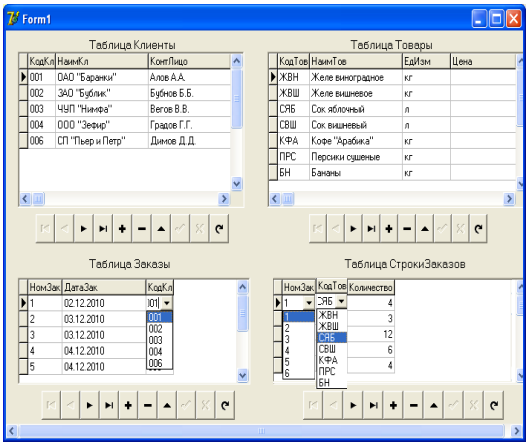

Рисунок 39 – **Окно приложения в процессе ввода информации в таблицы *Заказы* и *СтрокиЗаказов***

Проведем следующий эксперимент: изменим значение кода клиента в таблице *Клиенты* и убедимся, что автоматически изменилось значение (значения) соответствующего поля в таблице *Заказы* (рисунок 40). Аналогично – при изменении значения поля *КодТов* в таблице *Товары* изменится значение кода этого товара в таблице *СтрокиЗаказов*, а при изменении номера заказа в таблице *Заказы* изменяется это значение и в таблице *СтрокиЗаказов*.

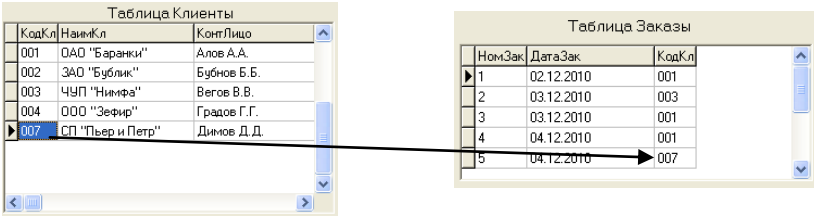

Рисунок 40 – **Изменение значения внешнего ключа при изменении значения первичного ключа**

Проверим, что при удалении какого-либо значения первичного ключа из главной таблицы автоматически будут удалены записи из подчиненных таблиц, связанные с этим значением. Например, при удалении из таблицы *Клиенты* записи с кодом клиента 007 будут удалены из таблицы *Заказы* записи, которые связаны с этим клиентом, а из таблицы *СтрокиЗаказов* – записи, относящиеся к удаляемым заказам.

Варианты индивидуальных заданий

Вариант 1. Прием материальных ценностей

Материальные ценности (инвентарные объекты), приобретаемые организацией делятся на инвентарные группы: литература, средства связи, мебель, вычислительная техника и другие. Они распределяются между подразделениями организации и учитываются в бухгалтерии по материально ответственным лицам на основании документа «Акт приема материальных ценностей». Пример этого документа приведен на рисунке 41.

Акт № 175 от 12.02.2011 г.
приема материальных ценностей Грушиным А.Б. Плановый отдел

Инв. номер	Название инвентарного объекта	Наименование инвентарной группы	Балансовая стоимость, р.
ИПО123	Принтер LBP-810	Вычислительная техника	520000
ИПО348	Стул	Мебель	18050
ИПО349	Кресло	Мебель	36800

Рисунок 41 – **Пример документа «Акт приема материальных ценностей»**

Задание. Необходимо разработать в среде СУБД Access базу данных *Материальные ценности*, в которой должны быть отражены сущности *Инвентарный объект* и *Материально ответственные лица*, содержащие нормативно-справочную информацию, а также сущности *Акты* и *Строки актов*, основанные на документе «Акт приема материальных ценностей». Сущность *Акты* содержит информацию из заголовочной части документов, а *Строки актов* – из их табличных частей. При выполнении задания надо учесть следующие обстоятельства (условия применения):

• номера актов не повторяются на протяжении всего периода учета;

• каждый инвентарный объект идентифицируется уникальным инвентарным номером;

• один и тот же объект может упоминаться в разных актах;

• в одном акте могут быть отражены поступления нескольких объектов, каждый из которых относится к своей инвентарной группе;

• все объекты одного акта принимаются одним материально ответственным лицом (МОЛ);

• в один день может быть оформлено несколько актов.

В результате логического проектирования БД был определен набор из четырех таблиц, структура которых описана ниже.

Структура таблицы *ИнвентОбъект* включает следующие атрибуты:

• *ИнвНомер* – инвентарный номер, первичный ключ, текстовый, до шести символов;

• *НазвИнвОб* – название инвентарного объекта, текстовый, до 50 символов;

• *БалансСтоим* – балансовая стоимость, денежный, ноль цифр в десятичной части;

• *НаимИнвГр* – наименование инвентарной группы, текстовый, до 50 символов.

Структура таблицы *МОЛ* включает следующие атрибуты:

• *ТабНом* – табельный номер МОЛ, первичный ключ, текстовый, до трех символов;

• *ФИОМОЛ* – фамилия МОЛ, текстовый, до 50 символов;

• *Подразд* – подразделение, в котором работает МОЛ, текстовый, до 50 символов.

Структура таблицы *Акты* включает следующие атрибуты:

• *НомерАкт* – номер акта, первичный ключ, текстовый, до трех символов;

• *ДатаАкт* – дата акта, дата/время, краткий формат даты;

• *ТабНом* – табельный номер МОЛ, текстовый, до трех символов, внешний ключ, подстановка из таблицы *МОЛ*.

Структура таблицы *СтрокиАктов* включает следующие атрибуты:

• *НомерАкт* – номер акта, текстовый, до трех символов, внешний ключ, подстановка из таблицы *Акты*;

• *ИнвНомер* – инвентарный номер, текстовый, до шести символов, внешний ключ, подстановка из таблицы *ИнвентОбъект*.

Опишите в среде СУБД Access структуру перечисленных таблиц, создайте схему данных и заполните таблицы информацией в соответствии со следующими условиями:

• информация в БД должна быть взята из четырех-пяти актов приема;

• два из этих актов должны быть выписаны в один и тот же день;

• справочник *МОЛ* должен содержать сведения о четырех сотрудниках, двое из которых должны работать в одном подразделении;

• в справочнике *ИнвентОбъект* должна находиться информация не менее чем о восьми объектах, из которых две пары должны иметь одинаковые названия, но разные инвентарные номера.

Вариант 2. Издержки обращения в торговле

Под издержками обращения в торговле понимают затраты по реализации товаров. К ним относятся транспортные расходы, затраты на электроэнергию, плата за аренду помещений и т. д. Учет издержек обращения в торговле проводится на основе документа «Ведомость издержек обращения в торговле». Пример этого документа приведен на рисунке 42.

Ведомость № 38 от 14.02.2011 г.
издержек обращения по подразделению: *Склад*

Статья издержек	Сумма, р.	Примечание
Транспорт	690000	Перевозка продуктов
Электроэнергия	147000	Освещение, холодильные установки
Спецодежда	109000	Спецодежда для грузчиков

Рисунок 42 – **Пример документа «Ведомость издержек обращения»**

Задание. Необходимо разработать в среде СУБД Access базу данных *Издержки обращения*, в которой должны быть отражены сущности *Статьи издержек* и *Подразделения организации*, содержащие нормативно-справочную информацию, а также сущности *Ведомости* и *Строки ведомостей*, основанные на документе «Ведомость издержек обращения в торговле». Сущность *Ведомости* содержит информацию из заголовочной части документов, а *Строки ведомостей* – из их табличных частей. При выполнении задания надо учесть следующие обстоятельства (условия применения):

• номера ведомостей не повторяются на протяжении всего периода учета;

• в одной ведомости может быть оценено несколько статей издержек;

• не все статьи из справочника должны быть оценены в одной ведомости;

• некоторые статьи издержек могут быть не оценены ни в одной из ведомостей;

• дважды одна статья не может быть оценена в одной ведомости;

• в один день в каждом подразделении должна быть оформлена только одна ведомость.

В результате логического проектирования БД был определен набор из четырех таблиц, структура которых описана ниже.

Структура таблицы *СтатьиИздержек* включает следующие атрибуты:

• *КодСт* – код статьи издержек, первичный ключ, текстовый, до двух символов;

• *НаимСтИзд* – наименование статьи издержек, текстовый, до 50 символов.

Структура таблицы *Подразделения* включает следующие атрибуты:

• *КодПодр* – код подразделения, первичный ключ, текстовый, до трех символов;

• *НазвПодр* – название подразделения, текстовый, до 50 символов.

Структура таблицы *Ведомости* включает следующие атрибуты:

• *НомерВед* – номер ведомости, первичный ключ, текстовый, до трех символов;

• *ДатаВед* – дата ведомости, дата/время, краткий формат даты;

• *КодПодр* – код подразделения, текстовый, до трех символов, внешний ключ, подстановка из таблицы *Подразделения*.

Структура таблицы *СтрокиВедомостей* включает следующие атрибуты:

• *НомерВед* – номер ведомости, текстовый, до трех символов, внешний ключ, подстановка из таблицы *Ведомости*;

• *КодСт* – код статьи издержек, текстовый, до двух символов, внешний ключ, подстановка из таблицы *СтатьиИздержек*;

• *Сумма* – стоимость издержек по статье, денежный, ноль десятичных цифр;

• *Примечание* – пояснительный текст, текстовый, до 50 символов.

Опишите в среде СУБД Access структуру перечисленных таблиц, создайте схему данных и заполните таблицы информацией в соответствии со следующими условиями:

• информация в БД должна быть взята из четырех-пяти ведомостей;

• две из этих ведомостей должны быть оформлены в один и тот же день (для разных подразделений);

• в каждой ведомости должно быть не менее двух строк;

• каждый из справочников *СтатьиИздержек* и *Подразделения* должен содержать не менее четырех записей.

Вариант 3. Оптовая торговля

Торговая фирма занимается оптовой торговлей. У нее имеется склад для хранения товаров. Основная деятельность фирмы состоит в том, что она закупает товары у одних контрагентов (поставщиков) и продает их другим контрагентам (покупателям). Некоторые контрагенты могут быть одновременно и поставщиками, и покупателями.

Поступающие от контрагентов и продаваемые контрагентам товары сопровождаются документами, которые называются накладными. Пример этого документа приведен на рисунке 43.

Накладная № 38 от 14.02.2011 г.
о поступлении товаров от ОАО «Мир»

Наименование товара	Количество, шт.	Цена, р.	Стоимость, р.
Персональный компьютер	10	800000	80000000
Блокнот	250	5000	1250000
Шариковая ручка	250	7000	1750000

Рисунок 43 – **Пример документа «Накладная»**

Задание. Необходимо разработать в среде СУБД Access базу данных *Торговая фирма*, в которой должны быть отражены сущности *Контрагенты* и *Товары*, содержащие нормативно-справочную информацию, а также сущности *Накладные* и *Строки накладных*, основанные на документе «Накладная». Сущность *Накладные* содержит информацию из заголовочной части документов, а *Строки накладных* – из их табличных частей. При выполнении задания надо учесть следующие обстоятельства (условия применения):

• номера накладных не повторяются на протяжении всего периода учета;

• накладная выписывается в точности одним контрагентом (приход для фирмы) или в точности одному контрагенту (расход для фирмы);

• одним контрагентом или одному контрагенту может быть выписано несколько накладных;

• в одной накладной может быть перечислено несколько товаров;

• для каждого товара отводится отдельная строка в табличной части накладной;

• при продаже товара его цена увеличивается на торговую надбавку, содержащуюся в справочнике товаров.

В результате логического проектирования БД был определен набор из четырех таблиц, структура которых описана ниже.

Структура таблицы *Контрагенты* включает следующие атрибуты:

• *КодКА* – код контрагента, первичный ключ, текстовый, до трех символов;

• *НаимКА* – наименование контрагента, текстовый, до 50 символов;

• *Адрес* – адрес контрагента, текстовый, до 50 символов;

• *Телефон* – номер телефона, текстовый, до 14 символов.

Структура таблицы *Товары* включает следующие атрибуты:

• *КодТов* – код товара, первичный ключ, текстовый, до трех символов;

• *НаимТов* – наименование товара, текстовый, до 50 символов;

• *Цена* – цена за единицу товара, денежный, ноль десятичных цифр;

• *ТоргНадб* – процент торговой надбавки при продаже товара, числовой, одинарное с плавающей точкой, одна десятичная цифра.

Структура таблицы *Накладные* включает следующие атрибуты:

• *НомерНак* – номер накладной, первичный ключ, текстовый, до трех символов;

• *ДатаНак* – дата накладной, дата/время, краткий формат даты;

• *ПрихРасх* – признак приходной или расходной накладной, логический, ИСТИНА – поступление товаров, ЛОЖЬ – продажа товаров;

• *КодКА* – код контрагента, текстовый, до трех символов, внешний ключ, подстановка из таблицы *Контрагенты*.

Структура таблицы *СтрокиНакладных* включает следующие атрибуты:

• *НомерНак* – номер накладной, текстовый, до трех символов, внешний ключ, подстановка из таблицы *Накладные*;

• *КодТов* – код товара, текстовый, до трех символов, внешний ключ, подстановка из таблицы *Товары*;

• *Количество* – объем поставки или продажи штучного товара, числовой, целое.

Опишите в среде СУБД Access структуру перечисленных таблиц, создайте схему данных и заполните таблицы информацией в соответствии со следующими условиями:

• информация в БД должна быть взята из четырех-пяти накладных; накладные должны быть приходные и расходные;

• две из этих накладных должны быть оформлены в один и тот же день (для разных контрагентов);

• в каждой накладной должно быть не менее двух строк;

• каждый из справочников *Контрагенты* и *Товары* должен содержать не менее четырех записей.

Вариант 4. Учет нематериальных активов

К нематериальным активам относят приобретенные организацией патенты, технологии, права на использование земельных участков, авторские права, программное обеспечение компьютеров и др. Для учета нематериальных активов в бухгалтерии составляется документ «Учетная карта нематериальных активов». Пример этого документа приведен на рисунке 44.

Учетная карта нематериальных активов
№ НА45 от 26.02.2011 г.

Инвентарный номер	Название нематериальных активов	Наименование вида	ФИО МОЛ	Балансовая стоимость, р.
ИН057	MS Access-2000	Программы	Гейц Б.	50000
ИН124	Государственный акт № ДА 43675	Права на землю	Батый Х.	18000000
ИН387	Пакет «Тест»	Программы	Лавров С.	20000

Рисунок 44 – **Пример документа «Учетная карта нематериальных активов»**

Задание. Необходимо разработать в среде СУБД Access базу данных *Нематериальные ценности*, в которой должны быть отражены сущности *Нематериальные активы* и *Материально ответственные лица*, содержащие нормативно-справочную информацию, а также сущности *Учетные карты* и *Строки учетных карт*, основанные на документе «Учетная карта нематериальных активов». Сущность *Учетные карты* содержит информацию из заголовочной части документов, а *Строки учетных карт* – из их табличных частей. При выполнении работы надо учесть следующие обстоятельства (условия применения):

• номера учетных карт не повторяются на протяжении всего периода учета;

• в одной учетной карте один объект нематериальных активов может быть упомянут только один раз;

• в один день может быть составлено несколько учетных карт;

• в одной карте могут быть упомянуты несколько материально ответственных лиц.

В результате логического проектирования БД был определен набор из четырех таблиц, структура которых описана ниже.

Структура таблицы *НематерАктивы* включает следующие атрибуты:

• *ИнвНомер* – инвентарный номер, первичный ключ, текстовый, до шести символов;

• *НазвНА* – название нематериального актива, текстовый, до 50 символов;

• *БалансСтоим* – балансовая стоимость, денежный, ноль цифр в десятичной части;

• *НаимВидаНА* – наименование вида нематериальных активов, текстовый, до 50 символов.

Структура таблицы *МОЛ* включает следующие атрибуты:

• *ТабНом* – табельный номер МОЛ, первичный ключ, текстовый, до трех символов;

• *ФИОМОЛ* – фамилия МОЛ, текстовый, до 50 символов;

• *Подразд* – подразделение, в котором работает МОЛ, текстовый, до 50 символов.

Структура таблицы *УчКарты* включает следующие атрибуты:

• *НомерУК* – номер учетной карты, первичный ключ, текстовый, до трех символов;

• *ДатаУК* – дата учетной карты, дата/время, краткий формат даты.

Структура таблицы *СтрокиУК* включает следующие атрибуты:

• *НомерУК* – номер учетной карты, текстовый, до трех символов, внешний ключ, подстановка из таблицы *УчКарты*;

• *ИнвНомер* – инвентарный номер, текстовый, до шести символов, внешний ключ, подстановка из таблицы *НематерАктивы*;

• *ТабНом* – табельный номер МОЛ, текстовый, до трех символов, внешний ключ, подстановка из таблицы *МОЛ*.

Опишите в среде СУБД Access структуру перечисленных таблиц, создайте схему данных и заполните таблицы информацией в соответствии со следующими условиями:

• информация в БД должна быть взята из четырех-пяти учетных карт;

• две из этих учетных карт должны быть составлены в один и тот же день;

• справочник *МОЛ* должен содержать сведения о четырех сотрудниках, двое из которых должны работать в одном подразделении;

• в справочнике *НематерАктивы* должна находиться информация не менее чем о восьми объектах, из которых две пары должны принадлежать одному виду нематериальных активов.

Вариант 5. Учет банковских кредитов

Для каждого кредита, предоставляемого какой-то организации, банк составляет договор, имеющий уникальный номер для банка. В этом договоре точно определяются сумма кредита, срок погашения (возврата кредитованной суммы) и годовой банковский процент за предоставление кредита. Периодически в банке составляется документ «Ведомость поступления денежных средств по договорам кредитов за период». Пример этого документа приведен на рисунке 45.

Ведомость № 31 за период от 02.02.2011 до 06.02.2011 г.
поступления денежных средств по договорам кредитов

Организация	№ договора	Оплаченная сумма, р.	Дата оплаты
ООО «Мир»	54	20000000	02.02.2011
ООО «Мир»	59	10000000	02.02.2011
АО «Земля»	67	15000000	06.02.2011

Рисунок 45 – **Пример документа «Ведомость поступления денежных средств по договорам кредитов за период»**

Задание. Необходимо разработать в среде СУБД Access базу данных *Кредиты*, в которой должны быть отражены сущности *Заемщики* и *Договора*, содержащие нормативно-справочную информацию, а также сущности *Ведомости* и *Строки ведомостей*, основанные на документе «Ведомость поступления денежных средств по договорам кредитов за период». Сущность *Ведомости* содержит информацию из заголовочной части документов, а *Строки ведомостей* – из их табличных частей. При выполнении задания надо учесть следующие обстоятельства (условия применения):

• номера ведомостей не повторяются на протяжении всего периода учета;

• в одной ведомости могут быть зафиксированы поступления денежных средств не по всем заключенным договорам;

• в одной ведомости ссылка на один договор может встретиться несколько раз, если заемщик сделал несколько взносов в рассматриваемый период;

• одна организация может иметь с банком несколько договоров о кредитовании;

• в одну ведомость включаются все сведения об оплатах, произведенных в указанный период.

В результате логического проектирования БД был определен набор из четырех таблиц, структура которых описана ниже.

Структура таблицы *Заемщики* включает следующие атрибуты:

• *КодОрг* – код организации, первичный ключ, текстовый, до трех символов;

• *НазвОрг* – название организации, текстовый, до 50 символов;

• *АдресОрг* – адрес организации, текстовый, до 50 символов.

Структура таблицы *Договора* включает следующие атрибуты:

• *НомерДог* – номер договора о кредитовании, первичный ключ, текстовый, до трех символов;

• *ДатаДог* – дата заключения договора о кредитовании, дата/время, краткий формат даты;

• *КодОрг* – код организации, текстовый, до трех символов, внешний ключ, подстановка из таблицы *Заемщики*;

• *СуммаКр* – сумма кредита, денежный, ноль десятичных цифр;

• *Срок* – дата погашения кредита, дата/время, краткий формат даты;

• *Ставка* – годовая процентная ставка за кредит, числовой, одинарное с плавающей точкой, одна десятичная цифра.

Структура таблицы *Ведомости* включает следующие атрибуты:

• *НомерВед* – номер ведомости, первичный ключ, текстовый, до трех символов;

• *НачДата* – начальная дата периода учета, дата/время, краткий формат даты;

• *КонДата* – конечная дата периода учета, дата/время, краткий формат даты.

Структура таблицы *СтрокиВедомостей* включает следующие атрибуты:

• *НомерВед* – номер ведомости, текстовый, до трех символов, внешний ключ, подстановка из таблицы *Ведомости*;

• *НомерДог* – номер договора о кредитовании, текстовый, до трех символов, внешний ключ, подстановка из таблицы *Договора*;

• *СуммаОпл* – оплаченная сумма в счет кредита, денежный, ноль десятичных цифр;

• *ДатаОпл* – дата оплаты, дата/время, краткий формат даты.

Опишите в среде СУБД Access структуру перечисленных таблиц, создайте схему данных и заполните таблицы информацией в соответствии со следующими условиями:

• информация в БД должна быть взята из четырех-пяти ведомостей;

• по крайней мере, по двум договорам о кредитовании должно быть несколько оплат;

• в каждой ведомости должно быть не менее двух строк;

• каждый из справочников *Договора* и *Заемщики* должен содержать не менее четырех записей.

Вариант 6. Расчеты с подотчетными лицами

Организации выдают своим сотрудникам наличные деньги под отчет на командировочные, операционные и хозяйственные расходы. Расходовать выданные под отчет суммы допускается лишь на те цели, на которые они выданы. Подотчетное лицо передает в бухгалтерию отчет, где документально должна быть подтверждена каждая потраченная сумма. Если выданный аванс превышает сумму потраченных средств, то сотрудник возвращает разницу в кассу предприятия. Однако, если аванс не покрыл все оправданные затраты, то подотчетному лицу доплачивается эта разница. Периодически в бухгалтерии составляется документ «Учетная ведомость по операциям с подотчетными лицами». Пример этого документа приведен на рисунке 46.

Учетная ведомость № 52 от 06.02.2011 г.

Фамилия И.О.	Номер документа	Дата документа	Операция	Сумма, р.
Сайков С.С.	67	30.01.2011	аванс	20000
Букин Б.Б.	69	30.01.2011	аванс	50000
Сайков С.С.	73	02.02.2011	отчет	19900
Сайков С.С.	74	02.02.2011	расчет	19900
Букин Б.Б.	76	03.02.2011	отчет	52500
Сайков С.С.	101	04.02.2002	возврат	100
Букин Б.Б.	102	04.02.2002	расчет	52500
Букин Б.Б.	103	05.02.2002	доплата	2500

Рисунок 46 – **Пример документа «Учетная ведомость по операциям с подотчетными лицами»**

Необходимо разработать в среде СУБД Access базу данных *Расчеты с подотчетными лицами*, в которой должны быть отражены сущности *Операции* и *Подотчетные лица*, содержащие нормативно-справочную информацию, а также сущности *Ведомости* и *Строки ведомостей*, основанные на документе «Учетная ведомость по операциям с подотчетными лицами». Сущность *Ведомости* содержит информацию из заголовочной части документов, а *Строки ведомостей* – из их табличных частей. При выполнении задания надо учесть следующие обстоятельства (условия применения):

- номера ведомостей не повторяются на протяжении всего периода учета;
- в ведомости может быть несколько записей на одну дату с одной и той же фамилией подотчетного лица, но с разными операциями;
- в один день может быть оформлено несколько ведомостей.

В результате логического проектирования БД был определен набор из четырех таблиц, структура которых описана ниже.

Структура таблицы *Операции* включает следующие атрибуты:

- *КодОп* – код операции, первичный ключ, текстовый, один символ;
- *НазвОп* – название операции, текстовый, до 50 символов.

Структура таблицы *ПодотчЛица*:

- *ТабНом* – табельный номер подотчетного лица, первичный ключ, текстовый, до трех символов;
- *ФИОПЛ* – фамилия подотчетного лица, текстовый, до 50 символов;
- *Подразд* – подразделение, в котором работает подотчетное лицо, текстовый, до 50 символов.

Структура таблицы *Ведомости* включает следующие атрибуты:

- *НомерВед* – номер ведомости, первичный ключ, текстовый, до трех символов;
- *ДатаВед* – дата составления ведомости, дата/время, краткий формат даты.

Структура таблицы *СтрокиВедомостей* включает следующие атрибуты:

- *НомерВед* – номер ведомости, текстовый, до трех символов, внешний ключ, подстановка из таблицы *Ведомости*;
- *ТабНом* – табельный номер подотчетного лица, текстовый, до трех символов, внешний ключ, подстановка из таблицы *ПодотчЛица*;
- *НомерДок* – номер документа по операциям с подотчетным лицом, текстовый, до трех символов;
- *ДатаДок* – дата документа по операциям с подотчетным лицом, дата/время, краткий формат даты;

• *КодОп* – код операции, текстовый, один символ, внешний ключ, подстановка из таблицы *Операции*;

• *СуммаОп* – сумма операции, денежный, ноль десятичных цифр.

Опишите в среде СУБД Access структуру перечисленных таблиц, создайте схему данных и заполните таблицы информацией в соответствии со следующими условиями:

• информация в БД должна быть взята из трех-четырех ведомостей, а каждая ведомость должна содержать не менее четырех строк;

• справочник *Операции* должен содержать пять записей: аванс, отчет, расчет, возврат, доплата;

• справочник *ПодотчЛица* должен содержать не менее четырех записей;

• при заполнении ведомости следует учесть, что операция «расчет» для конкретного подотчетного лица должна следовать за операцией «отчет» для этого сотрудника, а операции «возврат» и «доплата» – только после операции «расчет».

Вариант 7. Передача материальных ценностей

Материальные ценности (инвентарные объекты) находятся на учете в подразделениях организации и учитываются в бухгалтерии по материально-ответственным лицам (МОЛ). Материальные ценности делятся на инвентарные группы: литература, средства связи, мебель, вычислительная техника и другие. Инвентарные объекты могут передаваться из одного подразделения в другое. Факт передачи отражается в документе «Ведомость передачи материальных ценностей». Пример этого документа приведен на рисунке 47.

Ведомость № 17 от 28.02.2011 г.
передачи материальных ценностей за февраль 2011 г.

Инв. номер	Название инвентарного объекта	От кого		Кому		Балансовая стоимость, р.
		ФИО МОЛ	Подразделение	ФИО МОЛ	Подразделение	
ИПО123	Принтер LBP-810	Сидорова С.С.	Склад	Грушина Г.Г.	Плановый отдел	520000
ИПО348	Стул	Метелина М.М.	Отдел кадров	Демин Д.Д.	Лаборатория	18050
ИПО349	Кресло	Сидорова С.С.	Склад	Демин Д.Д.	Лаборатория	36800

Рисунок 47 – **Пример документа «Ведомость передачи материальных ценностей»**

Задание. Необходимо разработать в среде СУБД Access базу данных *Передача МЦ*, в которой должны быть отражены сущности *Инвентарный объект* и *Материально ответственные лица*, содержащие нормативно-справочную информацию, а также сущности *Ведомости* и *Строки ведомостей*, основанные на документе «Ведомость передачи материальных ценностей». Сущность *Ведомости* содержит информацию из заголовочной части документов, а *Строки ведомостей* – из их табличных частей. При выполнении работы надо учесть следующие обстоятельства (условия применения):

- номера ведомостей не повторяются на протяжении всего периода учета;
- каждый инвентарный объект идентифицируется уникальным инвентарным номером;
- один и тот же объект может упоминаться в разных ведомостях;
- в одной ведомости могут быть отражены передачи нескольких объектов, каждый из которых относится к своей инвентарной группе;
- ведомость оформляется один раз в месяц.

В результате логического проектирования БД был определен набор из четырех таблиц, структура которых описана ниже.

Структура таблицы *ИнвентОбъект* включает следующие атрибуты:

- *ИнвНомер* – инвентарный номер, первичный ключ, текстовый, до шести символов;
- *НазвИнвОб* – название инвентарного объекта, текстовый, до 50 символов;
- *БалансСтоим* – балансовая стоимость, денежный, ноль цифр в десятичной части;
- *НаимИнвГр* – наименование инвентарной группы, текстовый, до 50 символов.

Структура таблицы *МОЛ* включает следующие атрибуты:

- *МОЛ* (*ТабНом*, *ФИОМОЛ*, *Подразд*),
- *ТабНом* – табельный номер МОЛ, первичный ключ, текстовый, до трех символов;
- *ФИОМОЛ* – фамилия МОЛ, текстовый, до 50 символов;
- *Подразд* – подразделение, в котором работает МОЛ, текстовый, до 50 символов.

Структура таблицы *Ведомости* включает следующие атрибуты:

- *НомерВед* – номер ведомости, первичный ключ, текстовый, до трех символов;

• *ДатаВед* – дата составления ведомости, дата/время, краткий формат даты;

• *ОтчМесяц* – отчетный месяц, текстовый, до восьми символов;

• *ОтчГод* – год отчетного месяца, числовой, целое.

Структура таблицы *СтрокиВедомостей* включает следующие атрибуты:

• *НомерВед* – номер ведомости, текстовый, до трех символов, внешний ключ, подстановка из таблицы *Ведомости*;

• *ИнвНомер* – инвентарный номер, текстовый, до шести символов, внешний ключ, подстановка из таблицы *ИнвентОбъект*;

• *ТНПеред* – табельный номер материально ответственного лица, передающего инвентарный объект, текстовый, до трех символов, внешний ключ, подстановка из таблицы *МОЛ*;

• *ТНПрин* – табельный номер материально ответственного лица, принимающего инвентарный объект, текстовый, до трех символов, внешний ключ, подстановка из таблицы *МОЛ*.

Опишите в среде СУБД Access структуру перечисленных таблиц, создайте схему данных и заполните таблицы информацией в соответствии со следующими условиями:

• информация в БД должна быть взята из трех-четырех ведомостей, каждая из которых фиксирует три-четыре передачи инвентарных объектов;

• справочник *МОЛ* должен содержать сведения о четырех сотрудниках, двое из которых должны работать в одном подразделении;

• в справочнике *ИнвентОбъект* должна находиться информация не менее чем о восьми объектах, из которых две пары должны иметь одинаковые названия, но разные инвентарные номера.

Вариант 8. Ремонт бытовой техники

Фирма по ремонту бытовой техники принимает заказы у физических и юридических лиц. На каждый заказ оформляется квитанция. Существует категория срочных заказов, стоимость их выполнения расценивается на 10% больше, чем обычных заказов. Пример квитанции приведен на рисунке 48.

Квитанция № 38 от 14.02.2011 г.
Заказчик Иванов И.И., телефон 71-00-00
Категория заказа: срочный

Наименование бытовой техники	Вид ремонта	Тариф, р.	Стоимость, р.
Персональный компьютер	Замена чипа ОП	50000	70000
Персональный компьютер	Санитарное обслуживание	5000	10000
Пылесос	Устранение обрыва кабеля	20000	20000
Всего к оплате			110000

Рисунок 48 – **Пример квитанции**

Задание. Необходимо разработать в среде СУБД Access базу данных *Ремонт бытовой техники*, в которой должны быть отражены сущности *Виды бытовой техники* и *Прейскурант*, содержащие нормативно-справочную информацию, а также сущности *Квитанции* и *Строки квитанций*, основанные на документе «Квитанция». Сущность *Квитанции* содержит информацию из заголовочной части документов, а *Строки квитанций* – из их табличных частей. При выполнении задания надо учесть следующие обстоятельства (условия применения):

- номера квитанций не повторяются на протяжении всего периода учета;
- квитанция выписывается в точности одному заказчику;
- одному заказчику может быть выписано несколько квитанций, в том числе в один и тот же день;
- в одной квитанции может быть перечислено несколько видов бытовой техники и несколько видов ремонта;
- для каждого вида ремонта конкретного вида бытовой техники отводится отдельная строка в табличной части квитанции;
- в одной квитанции не может быть две строки, в которых совпадают вид бытовой техники и вид ремонта одновременно.

В результате логического проектирования БД был определен набор из четырех таблиц, структура которых описана ниже.

Структура таблицы *БытоваяТехника* включает следующие атрибуты:

- *КодБТ* – код бытовой техники, первичный ключ, текстовый, до трех символов;
- *НаимБТ* – наименование вида бытовой техники, текстовый, до 50 символов;

• *Модель* – модель вида бытовой техники, текстовый, до 50 символов.

Структура таблицы *Прейскурант* включает следующие атрибуты:

• *КодВР* – код вида ремонта или технического обслуживания, первичный ключ, текстовый, до трех символов;

• *НаимВР* – наименование вида ремонта или технического обслуживания, текстовый, до 50 символов;

• *Тариф* – стоимость выполнения данного вида ремонта или технического обслуживания, денежный, ноль десятичных цифр.

Структура таблицы *Квитанции* включает следующие атрибуты:

• *НомерКв* – номер квитанции, первичный ключ, текстовый, до трех символов;

• *ДатаКв* – дата выписки квитанции, дата/время, краткий формат даты;

• *Заказчик* – фамилия физического лица или наименование юридического лица, текстовый, до 50 символов;

• *Телефон* – номер телефона, текстовый, до 14 символов;

• *Категория* – признак срочности выполнения заказа, логический, ИСТИНА – срочный, ЛОЖЬ – обычный.

Структура таблицы *СтрокиКвитанций* включает следующие атрибуты:

• *НомерКв* – номер квитанции, текстовый, до трех символов, внешний ключ, подстановка из таблицы *Квитанции*;

• *КодБТ* – код бытовой техники, текстовый, до трех символов, внешний ключ, подстановка из таблицы *БытоваяТехника*;

• *КодВР* – код вида ремонта или технического обслуживания, текстовый, до трех символов, внешний ключ, подстановка из таблицы *Прейскурант*.

Опишите в среде СУБД Access структуру перечисленных таблиц, создайте схему данных и заполните таблицы информацией в соответствии со следующими условиями:

• информация в БД должна быть взята из четырех-пяти квитанций, заказы должны быть срочные и обычные;

• две из этих квитанций должны быть оформлены в один и тот же день (для разных заказчиков);

• в каждой квитанции должно быть не менее двух строк, причем в некоторых из них для одного и того же вида бытовой техники следует указать несколько видов ремонта или обслуживания;

• каждый из справочников *БытоваяТехника* и *Прейскурант* должен содержать не менее четырех записей.

Лабораторная работа 8
ФИЛЬТРАЦИЯ ИНФОРМАЦИИ В БД НА ОСНОВЕ ТЕХНОЛОГИИ ADO

***Цель работы*:** получение навыков использования технологии ADO для обеспечения универсального доступа к данным из СУБД Access с применением фильтров.

Пример выполнения работы

Задание. Создайте копию папки, содержащей проект лабораторной работы 7, и переименуйте эту папку. В окно формы для каждой главной таблицы добавьте поле для ввода шаблона значения некоторого атрибута. Включите в проект процедуры, обеспечивающие фильтрацию записей соответствующей таблицы по введенному шаблону. Кроме того, включите в проект процедуры, обеспечивающие фильтрацию записей подчиненной таблицы при перемещении указателя текущей записи в главной таблице.

Порядок выполнения работы

1. *Модификация окна формы*. Поместим в окно формы, приведенного на рисунке 34, поля для ввода наименования клиента, наименования товара и номера заказа. Полученное окно формы приведено на рисунке 49.

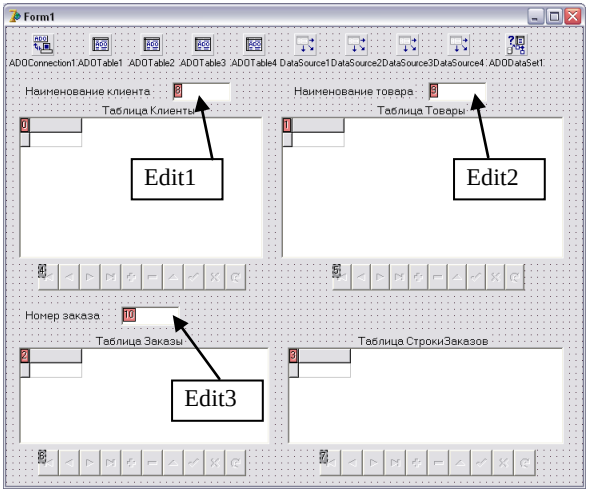

Рисунок 49 – **Окно формы для реализации фильтрации таблиц БД**

2. *Создание процедур для фильтрации по шаблону.* Двойным щелчком по полю *Edit1* создадим заготовку процедуры для обработки события *Change* (изменить). Заполним ее операторами, обеспечивающими фильтрацию таблицы *Клиенты* по полю *НаимКл* в соответствии с шаблоном, находящимся в поле *Edit1*. Полученный текст процедуры приведен на рисунке 50.

```
procedure TForm1.Edit1Change(Sender: TObject);
{Фильтрация таблицы Клиенты по полю НаимКл
 в соответствии с шаблоном из Edit1}
begin
  ADOTable1.Filtered:=false;
  if Edit1.Text<>''
    then
      begin
        ADOTable1.Filter:='НаимКл like '+#39+'%'+
                          Edit1.Text+'%'+#39;
        ADOTable1.Filtered:=True;
      end;
end;
```

Рисунок 50 – **Процедура фильтрации таблицы *Клиенты* по полю *НаимКл* в соответствии с шаблоном из *Edit1***

В тексте процедуры символы «#39» представляют собой код одинарной кавычки и используются для того, чтобы текст после оператора *like* оказался заключенным в эти кавычки. Вместо символов «#39» можно написать три одинарные кавычки подряд. Символ «%» по синтаксису языка СП Delphi означает последовательность произвольной длины из любых символов. Таким образом, этот символ соответствует символу «*» в Windows.

Аналогичным образом создаются процедуры фильтрации таблицы *Товары* по полю *НаимТов* и таблицы *Заказы* по полю *НомЗак* в соответствии с шаблонами из полей *Edit2* и *Edit3*.

3. *Проверка работы фильтрации по шаблону*. Запустим проект на выполнение и введем последовательно в поля *Edit1*, *Edit2* и *Edit3* шаблоны для фильтрации. Окно приложения с результатом фильтрации приведено на рисунке 51. До фильтрации содержимое таблиц БД приведено на рисунке 39.

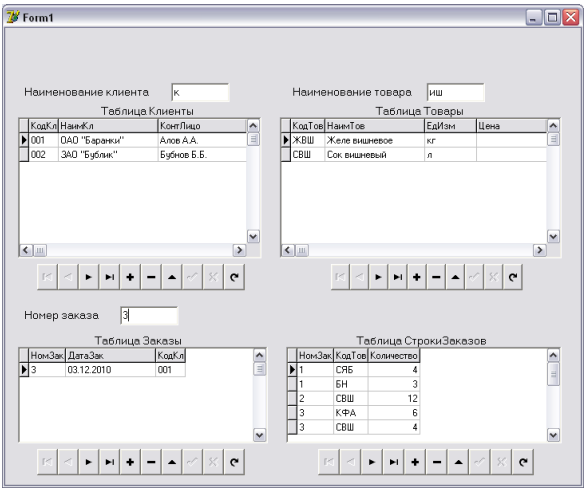

Рисунок 51 – **Окно приложения при фильтрации трех таблиц в соответствии с введенными шаблонами**

4. *Создание процедур для фильтрации подчиненных таблиц*. Теперь создадим процедуру, которая будет отфильтровывать записи подчиненной таблицы *Заказы* в соответствии с положением указателя текущей записи таблицы *Клиенты*. Для этого надо создать заготовку процедуры для события *AfterScroll* объекта *ADOTable1*. Текст искомой процедуры приведен на рисунке 52.

```
procedure TForm1.ADOTable1AfterScroll(DataSet: TDataSet);
{При перемещении по записям таблицы Клиенты в таблице
 Заказы отображаются только те записи, в которых
 КодКл совпадает с текущим значением таблицы Клиенты}
 var
   s1:string;
begin
  ADOTable3.Active:=True;
  ADOTable3.Filtered:=False;
    if ADOTable1.FieldValues['КодКл']<>null
      then
        s1:= ADOTable1.FieldValues['КодКл']
      else
        s1:='';
    ADOTable3.Filter:='КодКл='''+s1+'''';
    ADOTable3.Filtered:=True;

end;
```

Рисунок 52 – **Процедура фильтрации таблицы *Заказы* в соответствии с положением указателя текущей записи в таблице *Клиенты***

Аналогичным образом создаются процедуры фильтрации таблицы *СтрокиЗаказов* в соответствии с положением указателя текущей записи таблицы *Товары* и таблицы *Заказы*. Поскольку отображением записей таблицы *СтрокиЗаказов* управляют два указателя текущей записи разных таблиц, то результат определяется последним щелчком по одной из главных таблиц.

5. *Проверка работы фильтрации подчиненных таблиц*. Запустим проект на выполнение и будем перемещать указатель текущей записи в таблице *Клиенты*. При этом в таблице *Заказы* будут отображаться только записи, относящиеся к текущему значению поля *КодКл*. Аналогично перемещение указателя текущей записи в таблицах *Товары* и *Заказы* будет управлять отображением записей в таблице *СтрокиЗаказов*. На рисунке 53 приведено окно приложения в момент, когда последней просматривалась таблица *Товары*.

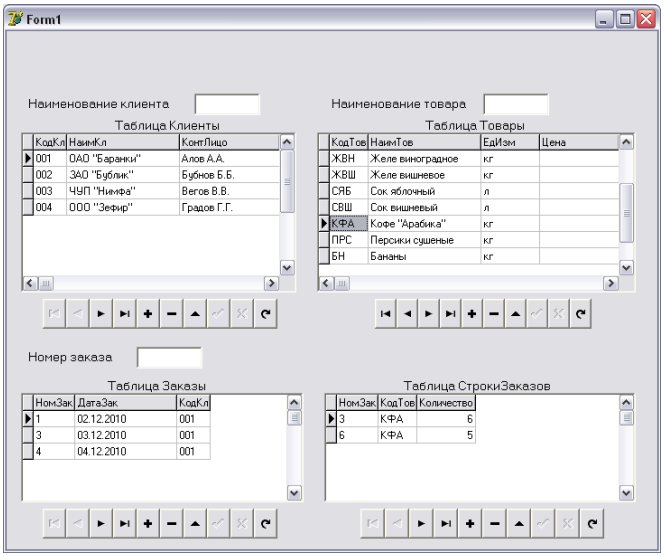

Рисунок 53 – **Окно приложения при фильтрации таблиц *Заказы* и *СтрокиЗаказов* в соответствии с положением указателя текущей записи в главных таблицах**

Лабораторная работа 9
РЕАЛИЗАЦИЯ ЗАПРОСОВ К БД НА ОСНОВЕ ТЕХНОЛОГИИ ADO

***Цель работы*:** получение навыков создания процедур, реализующих запросы к БД Access с использованием технологии ADO.

***Задание*:** в проект СП Delphi, созданный в лабораторной работе № 7, добавьте процедуры, обеспечивающие выполнение запросов в соответствии с вариантом индивидуального задания.

Пример выполнения работы

1. *Создание основы проекта*. Создадим копию папки, содержащей проект лабораторной работе № 7, и переименуем эту папку. Для компонента *ADOConnection* в свойстве *ConnectedString* подкорректируем путь к БД Access в связи с переносом БД в новую папку.

2. *Модификация интерфейса проекта*. Поместим в окно формы компонент *PageControl* (страница *Win32* палитры компонентов). С помощью команды *New Page* контекстного меню этого компонента образуем четыре страницы. Свойству *Caption* каждой из страниц дадим имена *Таблицы БД* (*TabSheet1*), *Запросы на выборку* (*TabSheet2*), *Перекрестные запросы* (*TabSheet3*) и *Активные запросы* (*TabSheet4*). С помощью окна *Object Tree View* перекрепим от *Form1* к *TabSheet1* все объекты *Label*, *DBGrid* и *DBNavigator* (всего 12 объектов). Полученное дерево объектов приведено на рисунке 54.

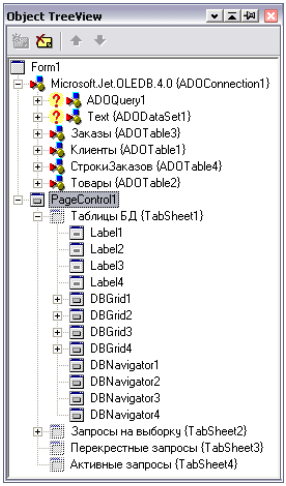

Рисунок 54 – **Вид окна дерева объектов после перезакрепления объектов**

Поместим на страницу *TabSheet2* компонент *ADOQuery* (страница *ADO* палитры компонентов), который обеспечивает применение запросов *SQL* на выборку при работе с данными через *ADO*. Соединение с БД задается значением *ADOConnection1* для свойства *Connection*.

На следующем этапе модификации интерфейса перенесем на форму и настроим компонент *DataSource5*, для свойства *DataSet* которого установим значение *ADOQuery1*.

Для представления результата выполнения запроса в виде таблицы поместим на страницу *TabSheet2* компонент *DBGrid5*. Установим значение *DataSource5* свойства *DataSource* этого компонента.

Поместим на страницу *TabSheet2* компонент *DBNavigator5*, который предназначен для перемещения по записям набора данных. Связь с набором данных устанавливается значением *DataSource5* свойства *DataSource* этого компонента.

Поместим на страницу командную кнопку Button1, для свойства *Caption* которой установим значение *Заказы заданного клиента*. Вид страницы *TabSheet2* приведен на рисунке 55.

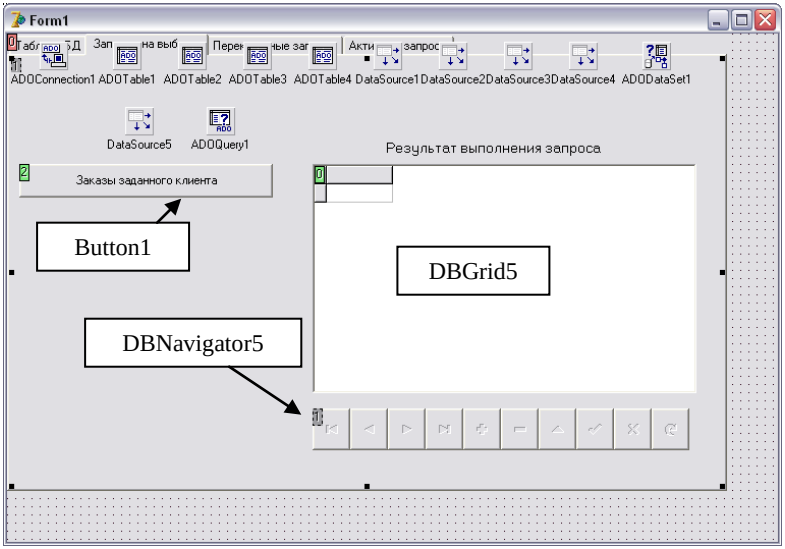

Рисунок 55 – **Вид страницы *TabSheet2* проекта**

В существующую процедуру *Form1Activate* добавим оператор *TabSheet1.Show*, чтобы при активации формы отображалась страница *Таблицы БД*.

3. *Создание процедуры для выполнения запроса на выборку с параметром*. Свяжем с кнопкой *Button1* запуск процедуры, реализующей получение списка заказов для клиента, код которого задается в диалоговом окне.

Общий вид процедуры, реализующей запрос на выборку с параметром, приведен на рисунке 56.

```
procedure TForm1.Button1Click(Sender: TObject);
//Структура процедуры реализации запроса на выборку с параметром
 var
     s,s1:string;
begin
  DataSource5.DataSet:=ADOQuery1;
  ADOQuery1.Active:=False;
  ADOQuery1.SQL.Clear;
  //Ввод параметра в строку s1
  s1:=InputBox('Окно ввода', {'Сообщение для ввода'}, '');
  //Присвоение строке s текста SQL-запроса
  s:='{Текст SQL-запроса}';
  ADOQuery1.SQL.Add(s);
  ADOQuery1.Active:=True;
end;
```

Рисунок 56 – **Структура процедуры, реализующей запрос на выборку с параметром**

Создадим процедуру, приведенную на рисунке 56, и в качестве строки «Сообщение для ввода» напишем строку *Введите код клиента*.

Для получения текста запроса на языке SQL создадим в СУБД Access искомый запрос. Вид данного запроса в режиме конструктора приведен на рисунке 57.

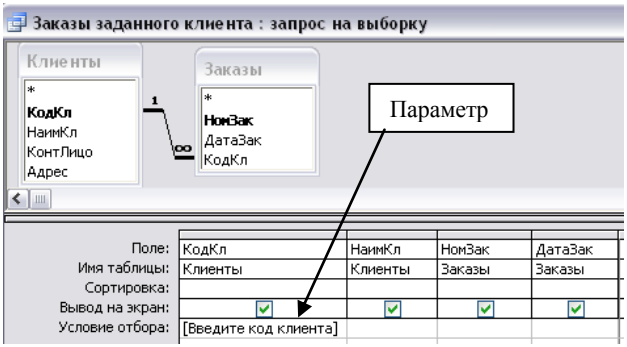

Рисунок 57 – **Запрос на выборку с параметром в режиме конструктора**

Проверим работу запроса в СУБД Access, а затем получим текст этого запроса на языке SQL. Процесс перехода в режим SQL показан на рисунке 58.

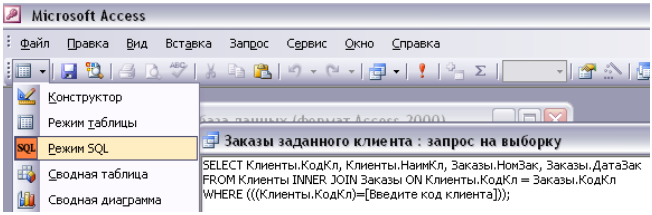

Рисунок 58 – **Запрос на выборку с параметром в режиме SQL**

Скопируем текст запроса в правую часть оператора присваивания переменной *s* процедуры, приведенной на рисунке 56. При этом в SQL-запросе выражение, заключенное в квадратные скобки, надо заменить переменной *s1* с соблюдением правил записи строк в языке Паскаль. Длинную строку заменим конкатенацией нескольких строк с использованием оператора «+». Обратите внимание, что служебные слова языка SQL выделяются пробелами. Окончательный текст процедуры реализации искомого запроса приведен на рисунке 59.

```
procedure TForm1.Button1Click(Sender: TObject);
{Реализация запроса о получении списка заказов
 для клиента по заданному коду клиента}
 var
     s,s1:string;
begin
  DataSource5.DataSet:=ADOQuery1;
  ADOQuery1.Active:=False;
  ADOQuery1.SQL.Clear;
  s1:=InputBox('Окно ввода', 'Введите код клиента', '');
  s:='SELECT Клиенты.КодКл, Клиенты.НаимКл, Заказы.НомЗак,'+
     ' Заказы.ДатаЗак FROM Клиенты INNER JOIN Заказы ON'+
     ' Клиенты.КодКл = Заказы.КодКл'+
     ' WHERE (((Клиенты.КодКл)='''+s1+'''))';
  ADOQuery1.SQL.Add(s);
  ADOQuery1.Active:=True;
end;
```

Рисунок 59 – **Текст процедуры, реализующей запрос на выборку с параметром**

Запустим проект на выполнение и проверим работу процедуры. Пример ее выполнения приведен на рисунке 60.

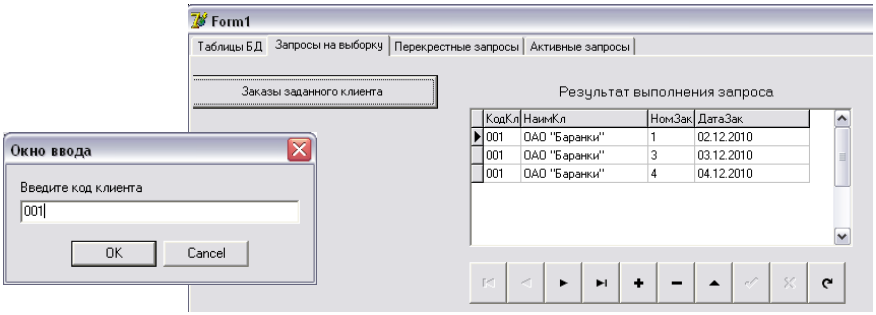

Рисунок 60 – **Пример выполнения процедуры, реализующей запрос на выборку с параметром**

Аналогичным образом создаются процедуры, реализующие другие запросы на выборку. Для каждого запроса надо помещать на страницу *TabSheet2* новую командную кнопку. И создавать процедуру, структура которой приведена на рисунке 56.

4. *Создание процедуры для выполнения перекрестного запроса.* Результатом перекрестного запроса является таблица, строки и столбцы которой соответствуют разным значениям некоторых полей исходных таблиц. В качестве примера создадим перекрестный запрос для получения таблицы, строки которой соответствуют наименованиям клиентов, а столбцы – номерам заказов. В каждой ячейке таблицы должна находиться общая стоимость заказа для клиента.

Поместим на страницу *TabSheet3* компонент *ADOQuery2* и установим значение *ADOConnection1* для свойства *Connection*.

Перенесем на форму и настроим компонент *DataSource6*, для свойства *DataSet* которого установим значение *ADOQuery2*.

Для представления результата выполнения запроса в виде таблицы поместим на страницу *TabSheet3* компонент *DBGrid6*. Установим значение *DataSource6* свойства *DataSource* этого компонента.

Поместим на страницу *TabSheet3* компонент *DBNavigator6*, который предназначен для перемещения по записям набора данных. Связь с набором данных устанавливается значением *DataSource6* свойства *DataSource* этого компонента.

Поместим на страницу *TabSheet3* командную кнопку *Button2*, для свойства *Caption* которой установим значение *Заказы-Клиенты-Стоимость*. Структура процедуры *Button2Click* аналогична той, что приведена на рисунке 56. Отличие состоит в том, что здесь не нужна строка *s1*, т. к. в рассматриваемом запросе отсутствует параметр. Структура процедуры *Button2Click* приведена на рисунке 61.

```
procedure TForm1.Button2Click(Sender: TObject);
{Реализация перекрестного запроса
 Заказы-Клиенты-Стоимость}
  var
     s:string;
begin
  DataSource6.DataSet:=ADOQuery2;
  ADOQuery2.Active:=False;
  ADOQuery2.SQL.Clear;
  s:='{Текст SQL-запроса}';
  ADOQuery2.SQL.Add(s);
  ADOQuery2.Active:=True;
end;
```

Рисунок 61 – **Структура процедуры, реализующей перекрестный запрос**

Для получения текста запроса на языке SQL создадим в СУБД Access искомый запрос. Вид данного запроса в режиме конструктора приведен на рисунке 62.

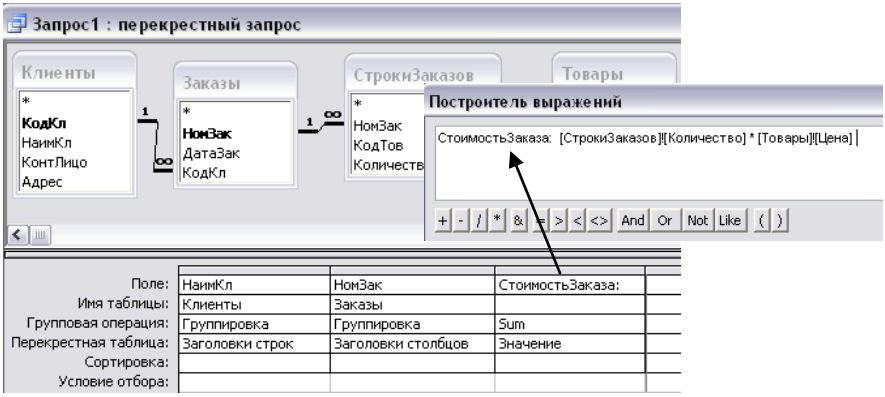

Рисунок 62 – **Перекрестный запрос в режиме конструктора**

Этот же запрос в режиме SQL приведен на рисунке 63.

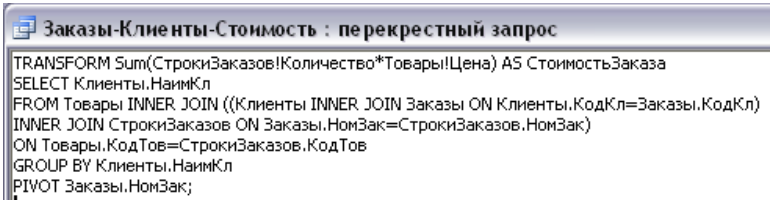

Рисунок 63 – **Перекрестный запрос в режиме SQL**

Скопируем текст запроса в правую часть оператора присваивания переменной s процедуры, приведенной на рисунке 61. Длинную строку заменим конкатенацией нескольких строк с использованием оператора «+». Окончательный текст процедуры реализации перекрестного запроса приведен на рисунке 64.

```
procedure TForm1.Button2Click(Sender: TObject);
{Реализация перекрестного запроса
 Заказы-Клиенты-Стоимость}
  var
     s:string;
begin
  DataSource6.DataSet:=ADOQuery2;
  ADOQuery2.Active:=False;
  ADOQuery2.SQL.Clear;
  s:='TRANSFORM Sum(СтрокиЗаказов!Количество*Товары!Цена)'+
     ' AS СтоимостьЗаказа SELECT Клиенты.НаимКл'+
     ' FROM Товары INNER JOIN ((Клиенты INNER JOIN '+
     'Заказы ON Клиенты.КодКл=Заказы.КодКл)'+
     ' INNER JOIN СтрокиЗаказов ON Заказы.НомЗак='+
     'СтрокиЗаказов.НомЗак) ON Товары.КодТов='+
     'СтрокиЗаказов.КодТов GROUP BY Клиенты.НаимКл'+
     ' PIVOT Заказы.НомЗак';
  ADOQuery2.SQL.Add(s);
  ADOQuery2.Active:=True;
end;
```

Рисунок 64 – **Текст процедуры, реализующей перекрестный запрос**

Результат выполнения созданной процедуры приведен на рисунке 65.

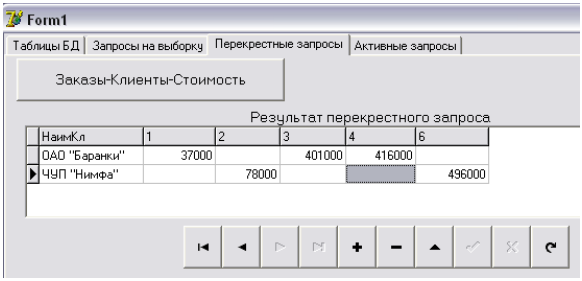

Рисунок 65 – **Результат выполнения процедуры для перекрестного запроса**

5. *Создание процедуры для выполнения запроса на обновление*. Запрос на обновление относится к активным запросам, т. е. запросам, изменяющим таблицы БД. Создадим процедуру, реализующую запрос, увеличивающий все цены на заданную процентную величину.

Для выполнения активных запросов предназначен специальный компонент *ADOCommand* (страница *ADO* палитры компонентов). Поместим его в окно формы и установим значение *ADOConnection1* для свойства *Connection*, которое обеспечит соединение с БД.

Поместим на страницу *TabSheet4* командную кнопку *Button3*, для свойства *Caption* которой установим значение *ОбновлениеЦены*. Структура процедуры, реализующей любой активный запрос, приведена на рисунке 66.

```
procedure TForm1.Button3Click(Sender: TObject);
//Структура активного запроса
 var
   s,s1:string;
begin
  s1:=InputBox('Окно ввода', '{Название параметра}', '');
  s:={текст SQL-запроса};
  ADOCommand1.CommandText:=s;
  ADOCommand1.Execute
end;
```

Рисунок 66 – **Структура процедуры, реализующей активный запрос**

Для получения текста SQL-запроса создадим в СУБД Access требуемый запрос, вид которого в режиме конструктора и в режиме SQL приведен на рисунке 67.

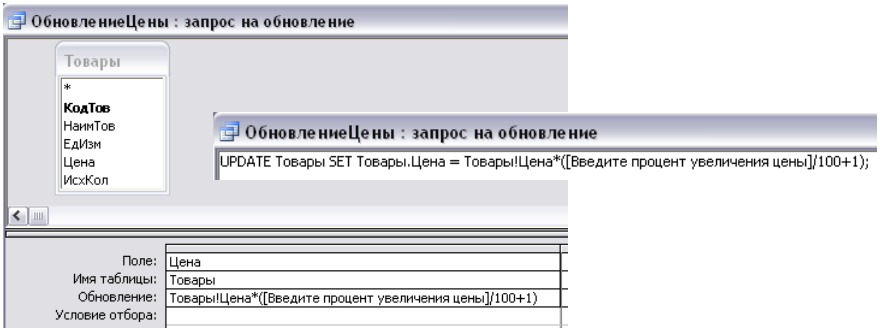

Рисунок 67 – **Активный запрос на обновление записей в режиме конструктора и в режиме SQL**

После вставки полученного текста SQL-запроса в заготовку процедуры получим текст, приведенный на рисунке 68.

```
procedure TForm1.Button3Click(Sender: TObject);
//Увеличение цены на заданную величину
 var
   s,s1:string;
begin
  s1:=InputBox('Окно ввода', 'Введите процент увеличения цены', '');
  s:='UPDATE Товары SET Товары.Цена = Товары!Цена'+
     '*('+s1+'/100+1)';
  ADOCommand1.CommandText:=s;
  ADOCommand1.Execute
end;
```

Рисунок 68 – **Процедура, реализующая активный запрос на обновление с параметром**

Запустим проект на выполнение, перейдем на страницу *TabSheet4* и щелкнем по командной кнопке *Button3*. В открывшемся диалоговом окне введем процентное значение увеличения цены.

Для проверки результата выполнения процедуры надо перейти на страницу *TabSheet1* и с помощью навигатора обновить таблицу *Товары*. Пример выполнения процедуры приведен на рисунке 69.

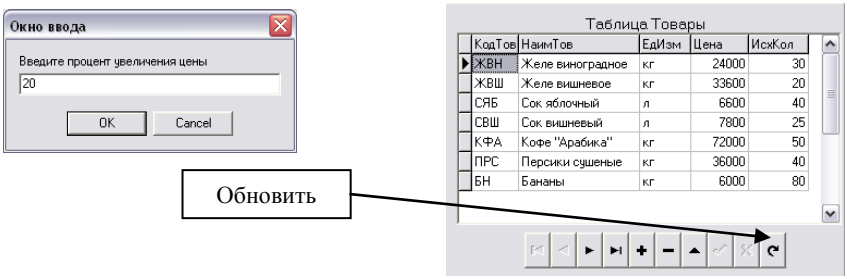

Рисунок 69 – **Пример выполнения запроса на обновление**

6. *Создание процедуры для выполнения запроса на удаление*. Запрос на удаление относится к активным запросам. Создадим процедуру для удаления из таблиц БД записей с информацией о всех заказах клиента с заданным номером.

Поместим на страницу *TabSheet4* командную кнопку *Button4*, для свойства *Caption* которой установим значение *УдалениеЗаказовКлиента*. Структура процедуры *Button4Click* приведена на рисунке 66.

Для получения текста SQL-запроса создадим в СУБД Access требуемый запрос, вид которого в режиме конструктора и в режиме SQL приведен на рисунке 70.

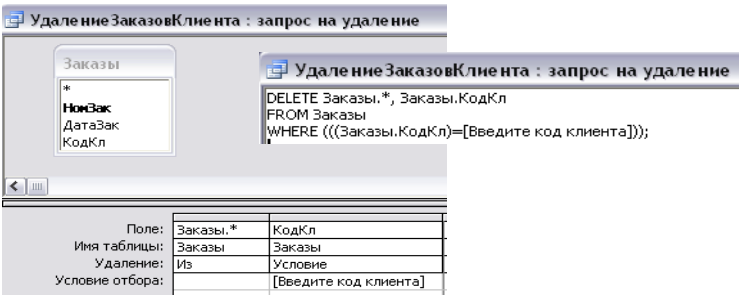

Рисунок 70 – **Активный запрос на удаление записей в режиме конструктора и в режиме SQL**

В результате выполнения этого запроса удаляются все записи о заказах указанного клиента из таблицы *Заказы* и соответствующие строки удаляемых заказов из таблицы *СтрокиЗаказов*.

После вставки полученного текста SQL-запроса в заготовку процедуры получим текст, приведенный на рисунке 71.

```
procedure TForm1.Button4Click(Sender: TObject);
{Удаление всех сведений о заказах заданного клиента}
 var
   s,s1:string;
begin
  s1:=InputBox('Окно ввода', 'Введите код клиента', '');
  s:='DELETE Заказы.*, Заказы.КодКл FROM Заказы'+
     ' WHERE (((Заказы.КодКл)='''+s1+'''))';
  ADOCommand1.CommandText:=s;
  ADOCommand1.Execute
end;
```

Рисунок 71 – **Процедура, реализующая активный запрос на удаление с параметром**

Для проверки результата выполнения процедуры надо перейти на страницу *TabSheet1* и с помощью навигатора обновить таблицу *Заказы*. При этом в таблице *СтрокиЗаказов* отображаются все записи, существовавшие до удаления. Чтобы обновить содержимое таблицы *СтрокиЗаказов*, надо прекратить выполнение проекта и запустить его заново.

Обновление отображения таблиц *Заказы* и *СтрокиЗаказов* можно осуществить добавлением в процедуру операторов обновления, как это показано на рисунке 72.

```
procedure TForm1.Button4Click(Sender: TObject);
{Удаление всех сведений о заказах заданного клиента
 с обновлением таблиц на странице TabSheet1}
 var
   s,s1:string;
begin
  s1:=InputBox('Окно ввода', 'Введите код клиента', '');
  s:='DELETE Заказы.*, Заказы.КодКл FROM Заказы'+
     ' WHERE (((Заказы.КодКл)='''+s1+'''))';
  ADOCommand1.CommandText:=s;
  ADOCommand1.Execute;
  //Обновление таблиц
  ADOConnection1.Connected:=False;
  ADOConnection1.Connected:=True;
  Adotable1.Active:=True;
  ADOTable2.Active:=True;
  ADOTable3.Active:=True;
  ADOTable4.Active:=True;
  ObnovlenieSpiska;
  TabSheet1.Show
end;
```

Рисунок 72 – **Процедура, реализующая активный запрос на удаление с параметром с обновлением таблиц**

7. *Создание процедуры для выполнения запроса на выборку, основанного на другом запросе*. Создадим процедуру для получения следующей информации о каждом товаре: исходное количество, заказано, стоимость заказов, остаток на складе.

Создание такого запроса в СУБД Access произведем в два этапа. Сначала создадим активный запрос с группировкой для создания таблицы *Вспомогательная*, в которой для каждого товара будет получено общее количество и общая стоимость заказанных товаров (рисунки 73 и 74).

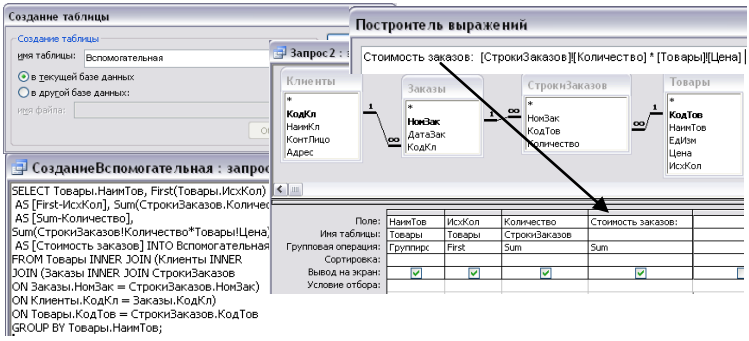

Рисунок 73 – **Запрос на создание таблицы *Вспомогательная* в режиме конструктора и в режиме SQL**

Вспомогательная : таблица

	НаимТов	First-ИсхКол	Sum-Количество	Стоимость заказов
	Бананы	80	6	36 000,00р.
	Желе виноградное	30	4	96 000,00р.
	Желе вишневое	20	19	638 400,00р.
	Кофе "Арабика"	50	11	792 000,00р.
	Сок вишневый	25	16	124 800,00р.
	Сок яблочный	40	4	26 400,00р.

Рисунок 74 –**Таблица *Вспомогательная* после создания**

На втором этапе надо создать запрос на выборку из таблицы *Вспомогательная* для вычисления остатка на складе как результат вычитания заказанного количества из исходного. Указанный запрос в режиме конструктора и в режиме SQL приведен на рисунке 75.

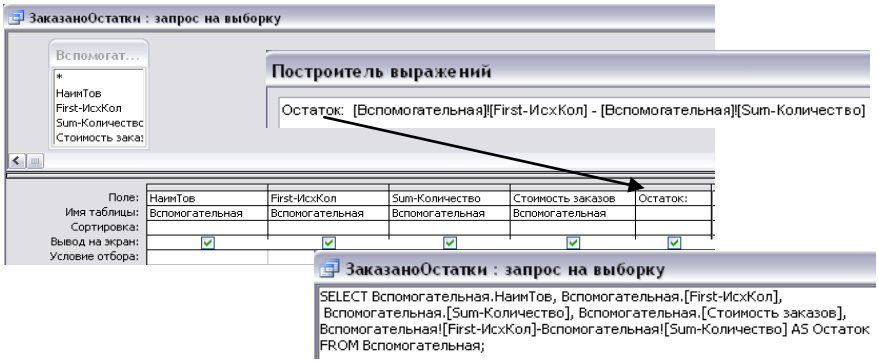

Рисунок 75 – **Запрос на выборку из таблицы *Вспомогательная* в режиме конструктора и в режиме SQL**

Поместим на страницу *TabSheet2* командную кнопку *Button5*, для свойства *Caption* которой установим значение *ЗаказаноОстаток*. Процедура *Button5Click*, реализующая искомый запрос, будет состоять из трех частей:

- операторы, реализующие активный запрос создания таблицы *Вспомогательная*;
- операторы, реализующие запрос на выборку;
- операторы, реализующие активный запрос на удаление таблицы *Вспомогательная* из базы данных.

Текст искомой процедуры приведен на рисунке 76.

```
procedure TForm1.Button5Click(Sender: TObject);
{Определение для каждого товара заказанного количества,
 стоимости заказов и остатка на складе}
 var
   s:string;
begin
  //Создание таблицы Вспомогательная
  s:='SELECT Товары.НаимТов, First(Товары.ИсхКол) AS [First-ИсхКол],'+
     ' Sum(СтрокиЗаказов.Количество) AS [Sum-Количество],'+
     'Sum(СтрокиЗаказов!Количество*Товары!Цена) AS [Стоимость заказов]'+
     ' INTO Вспомогательная FROM Товары INNER JOIN (Клиенты INNER'+
     ' JOIN (Заказы INNER JOIN СтрокиЗаказов'+
     ' ON Заказы.НомЗак = СтрокиЗаказов.НомЗак)'+
     ' ON Клиенты.КодКл = Заказы.КодКл)'+
     ' ON Товары.КодТов = СтрокиЗаказов.КодТов'+
     ' GROUP BY Товары.НаимТов';
  ADOCommand1.CommandText:=s;
  ADOCommand1.Execute;
  //Запрос на выборку из таблицы Вспомогательная
  DataSource5.DataSet:=ADOQuery1;
  ADOQuery1.Active:=False;
  ADOQuery1.SQL.Clear;
  s:='SELECT Вспомогательная.НаимТов, '+
     'Вспомогательная.[First-ИсхКол], Вспомогательная.[Sum-Количество],'+
     ' Вспомогательная.[Стоимость заказов],'+
     'Вспомогательная![First-ИсхКол]-Вспомогательная![Sum-Количество]'+
     ' AS Остаток FROM Вспомогательная';
  ADOQuery1.SQL.Add(s);
  ADOQuery1.Active:=True;
  //Удаление из базы данных таблицы Вспомогательная
  s:='DROP TABLE Вспомогательная';
  ADOCommand1.CommandText:=s;
  ADOCommand1.Execute;
end;
```

Рисунок 76 – **Процедура, реализующая запрос на выборку, основанный на другом запросе**

Запустим проект на выполнение и проверим корректность работы созданной процедуры. Ее результат будет отображен на странице *TabSheet2* (рисунок 77).

Результат выполнения запроса

НаимТов	First-ИсхКол	Sum-Количество	Стоимость заказов	Остаток
Бананы	80	6	36000	74
Желе виноградно	30	4	96000	26
Желе вишневое	20	19	638400	1
Кофе "Арабика"	50	11	792000	39
Сок вишневый	25	16	124800	9
Сок яблочный	40	4	26400	36

Рисунок 77 – **Пример выполнения процедуры, реализующей запрос на выборку, основанный на другом запросе**

Индивидуальные задания

Вариант 1. Прием материальных ценностей

В проект СП Delphi, созданный в лабораторной работе 7, добавьте процедуры, обеспечивающие выполнение следующих запросов:

1. Запросы на выборку с параметром:

- для заданного наименования инвентарной группы выдать список находящихся на учете инвентарных объектов с указанием их названия, стоимости и материально ответственного лица;
- для заданного периода выдать список принятых на учет инвентарных объектов с указанием их названия, стоимости, материально ответственного лица и номера акта;
- выдать список всех имеющихся на учете инвентарных объектов (инвентарный номер, название, балансовая стоимость, наименование инвентарной группы), балансовая стоимость которых больше заданного значения;
- для заданного названия инвентарного объекта выдать список материально ответственных лиц, у которых есть данный объект, с указанием фамилии, подразделения, номера и даты акта приема.

2. Запросы с группировкой:

- для каждого наименования инвентарной группы указать общую стоимость находящихся на учете во всей организации инвентарных объектов, относящихся к заданной группе;
- для каждого подразделения указать общее количество находящихся на учете инвентарных объектов.

3. Перекрестный запрос:

- для каждого наименования инвентарной группы и для каждого материально ответственного лица указать общую стоимость находящихся у него на учете инвентарных объектов, относящихся к заданной группе.

4. Запросы на обновление данных:

- изменить заданную фамилию материально ответственного лица;
- уменьшить на 10% балансовую стоимость всех находящихся на учете в организации инвентарных объектов, относящихся к заданной инвентарной группе и принятых на учет после заданной даты.

5. Запросы на удаление:

- удалить из БД все сведения об инвентарном объекте с заданным инвентарным номером;
- удалить из БД все сведения об актах, относящихся к заданной дате.

Вариант 2. Издержки обращения в торговле

В проект СП Delphi, созданный в лабораторной работе 7, добавьте процедуры, обеспечивающие выполнение следующих запросов:

1. Запросы на выборку с параметром:

- для заданного наименования статьи издержек выдать список строк ведомостей, в которых она упоминается, с указанием номера и даты ведомости, названия подразделения и суммы;
- для заданного периода выдать список оформленных ведомостей с указанием их номера, даты и названия подразделения;
- для заданного подразделения выдать список всех строк составленных для него ведомостей с указанием номера и даты ведомости, наименования статьи издержек, суммы и примечания;
- из всех ведомостей выделить строки, в которых расходы превышают заданную величину.

2. Запросы с группировкой:

- для каждой статьи издержек указать общую стоимость расходов по всем подразделениям за весь период учета;
- для каждого подразделения указать общую стоимость расходов по всем статьям издержек за весь период учета.

3. Перекрестный запрос:

- для каждой статьи издержек и для каждого подразделения указать стоимость расходов за весь период учета.

4. Запросы на обновление данных:

- изменить название заданного подразделения;
- увеличить на 15% стоимость расходов для заданной статьи издержек, совершенных после заданной даты.

5. Запросы на удаление:

- удалить из БД все сведения, содержащиеся в ведомости с заданным номером;
- удалить из БД все сведения о расходах по заданной статье издержек.

Вариант 3. Оптовая торговля

В проект СП Delphi, созданный в лабораторной работе 7, добавьте процедуры, обеспечивающие выполнение следующих запросов:

1. Запросы на выборку с параметром:

- для заданного наименования товара выдать список строк накладных, в которых он упоминается, с указанием номера и даты

накладной, ее типа (приходная или расходная), наименования контрагента и стоимости;

• для заданного периода выдать список накладных с указанием их номера, даты и типа, а также наименования контрагента;

• для заданного наименования контрагента выдать список полученных от него товаров с указанием номера и даты накладной, наименования и стоимости товара;

• для заданного наименования контрагента выдать список проданных ему товаров, с указанием номера и даты накладной, наименования и стоимости товара.

2. Запросы с группировкой:

• для каждой накладной указать общую стоимость перечисленных в ней товаров;

• для каждого контрагента указать стоимости полученных от него и проданных ему товаров.

3. Перекрестный запрос:

• для каждого контрагента и для каждого наименования товара указать сальдо.

4. Запросы на обновление данных:

• изменить наименование заданного контрагента;

• увеличить на 5%-ную надбавку для заданного наименования товара.

5. Запросы на удаление:

• удалить из БД все сведения, содержащиеся в накладной с заданным номером;

• удалить из БД все сведения о продажах заданному контрагенту.

Вариант 4. Учет нематериальных активов

В проект СП Delphi, созданный в лабораторной работе 7, добавьте процедуры, обеспечивающие выполнение следующих запросов:

1. Запросы на выборку с параметром:

• для заданного наименования вида нематериальных активов выдать список находящихся на учете объектов с указанием их названия, стоимости и материально ответственного лица;

• для заданного периода выдать список принятых на учет нематериальных активов с указанием их названия, стоимости, материально ответственного лица и номера учетной карты;

• выдать список всех имеющихся на учете нематериальных активов (инвентарный номер, название, балансовая стоимость, наимено-

вание вида нематериальных активов), балансовая стоимость которых больше заданного значения;

• для заданного названия нематериального актива выдать список материально ответственных лиц, у которых есть данный объект, с указанием фамилии, подразделения, номера и даты учетной карты.

2. Запросы с группировкой:

• для каждого наименования вида нематериальных активов указать общую стоимость находящихся на учете во всей организации нематериальных активов, относящихся к заданному виду;

• для каждого подразделения указать общее количество находящихся на учете нематериальных активов.

3. Перекрестный запрос:

• для каждого наименования вида нематериальных активов и для каждого материально ответственного лица указать общую стоимость находящихся у него на учете объектов, относящихся к заданному виду.

4. Запросы на обновление данных:

• изменить заданную фамилию материально ответственного лица;

• уменьшить на 20% балансовую стоимость всех находящихся на учете в организации объектов, относящихся к заданному виду нематериальных активов и принятых на учет после заданной даты.

5. Запросы на удаление:

• удалить из БД все сведения о нематериальном активе с заданным инвентарным номером;

• удалить из БД все сведения об учетных картах, относящихся к заданной дате.

Вариант 5. Учет банковских кредитов

В проект СП Delphi, созданный в лабораторной работе 7, добавьте процедуры, обеспечивающие выполнение следующих запросов:

1. Запросы на выборку с параметром:

• для заданного названия организации выдать список строк ведомостей, в которых оно упоминается, с указанием номера ведомости, номера договора, суммы и даты оплаты;

• для заданного периода выдать список произведенных оплат с указанием номера договора, суммы и даты оплаты;

• для заданного номера договора выдать список всех строк ведомостей, с указанием номера ведомости, названия организации, суммы и даты оплаты;

• из всех ведомостей выделить строки, в которых сумма оплаты меньше заданной величины.

2. Запросы с группировкой:

• для каждого договора о кредитовании указать общую сумму оплаты за весь период учета и сумму кредита;

• для каждой организации указать общую сумму оплаты за весь период учета и общую сумму кредита по всем договорам.

3. Перекрестный запрос:

• для каждой организации и для каждого договора о кредитовании указать сумму оплаты за весь период учета.

4. Запросы на обновление данных:

• изменить название заданной организации;

• увеличить на 1% ставку по кредиту для заданного договора, заключенных после заданной даты.

5. Запросы на удаление:

• удалить из БД все сведения, содержащиеся в ведомости с заданным номером;

• удалить из БД все сведения об оплате по кредиту для заданного договора.

Вариант 6. Расчеты с подотчетными лицами

В проект СП Delphi, созданный в лабораторной работе 7, добавьте процедуры, обеспечивающие выполнение следующих запросов:

1. Запросы на выборку с параметром:

• для заданной операции выдать список строк из всех ведомостей, в которых она упоминается, с указанием номера и даты ведомости, фамилии подотчетного лица, номера и даты документа, даты и суммы операции;

• для заданного периода выдать список подотчетных лиц, получивших аванс, с указанием их табельного номера, фамилии, подразделения и суммы аванса;

• выдать список подотчетных лиц, сумма отчета которых больше заданного значения;

• для заданного подразделения выдать список подотчетных лиц, получивших аванс, с указанием их табельного номера, фамилии, подразделения и суммы аванса.

2. Запросы с группировкой:

• для каждого подотчетного лица указать общую сумму возвратов и общую сумму доплат;

• для каждой операции указать общую сумму для всех подотчетных лиц за весь период учета.

3. Перекрестный запрос:

• для каждого подотчетного лица и для каждой операции указать общую сумму за весь период учета.

4. Запросы на обновление данных:

• изменить заданную фамилию подотчетного лица;

• увеличить на 20% сумму отчета для каждого подотчетного лица, дата отчета которого была после заданной даты.

5. Запросы на удаление:

• удалить из БД все записи с операцией «возврат», относящиеся к заданной дате;

• удалить из БД все записи о заданном подотчетном лице.

Вариант 7. Передача материальных ценностей

В проект СП Delphi, созданный в лабораторной работе 7, добавьте процедуры, обеспечивающие выполнение следующих запросов:

1. Запросы на выборку с параметром:

• для заданного наименования инвентарной группы выдать список передававшихся инвентарных объектов с указанием их названия, стоимости и материально ответственных лиц;

• для заданного месяца выдать список переданных инвентарных объектов с указанием их названия, стоимости, передающего и принимающего материально ответственного лица и номера ведомости;

• выдать список всех передававшихся инвентарных объектов (инвентарный номер, название, балансовая стоимость, наименование инвентарной группы), балансовая стоимость которых больше заданного значения;

• для заданного названия инвентарного объекта выдать список материально ответственных лиц, которые передавали данный объект, с указанием фамилии, подразделения, номера и даты ведомости передачи.

2. Запросы с группировкой:

• для каждого наименования инвентарной группы указать общую стоимость передававшихся инвентарных объектов, относящихся к заданной группе в заданный месяц;

• для каждого подразделения указать общее количество передававшихся инвентарных объектов в указанном году.

3. Перекрестный запрос:

для каждого наименования инвентарной группы и для каждого материально-ответственного лица указать общую стоимость передававшихся ему инвентарных объектов в указанном году.

4. Запросы на обновление данных:

• изменить заданную фамилию материально ответственного лица;

• уменьшить на 10% балансовую стоимость всех находящихся на учете в организации инвентарных объектов, относящихся к заданной инвентарной группе.

5. Запросы на удаление:

• удалить из БД все сведения об инвентарном объекте с заданным инвентарным номером;

• удалить из БД всю информацию, связанную с ведомостью заданного отчетного месяца.

Вариант 8. Ремонт бытовой техники

В проект СП Delphi, созданный в лабораторной работе 7, добавьте процедуры, обеспечивающие выполнение следующих запросов:

1. Запросы на выборку с параметром:

• для заданного наименования бытовой техники выдать список строк квитанций, в которых он упоминается, с указанием номера и даты квитанции, ее категории (срочный или обычный заказ), наименования заказчика и стоимости (с учетом категории);

• для заданного периода выдать список квитанций с указанием их номера, даты и категории, а также наименования заказчика;

• для заданного наименования заказчика выдать список полученных от него заказов, с указанием номера и даты квитанции, наименования бытовой техники и стоимости выполненных работ с учетом срочности;

• для заданного вида работ выдать список строк квитанций, в которых он упоминается, с указанием номера и даты квитанции, наименования бытовой техники и стоимости выполненных работ с учетом срочности.

2. Запросы с группировкой:

• для каждой квитанции указать общую стоимость перечисленных в ней работ с учетом срочности;

• для каждого заказчика указать общую стоимость выполненных для него работ с учетом срочности.

3. Перекрестный запрос:

• для каждого заказчика и для каждого наименования бытовой техники указать стоимость работ с учетом срочности.

4. Запросы на обновление данных:

• изменить наименование заданного вида бытовой техники;

• увеличить на 5% тариф для заданного наименования вида работ.

5. Запросы на удаление:

• удалить из БД все сведения, содержащиеся в квитанции с заданным номером;

• удалить из БД все сведения о работах для заданного заказчика.

Лабораторная работа 10
ФОРМИРОВАНИЕ ОТЧЕТОВ С ИСПОЛЬЗОВАНИЕМ COM-ТЕХНОЛОГИЙ

***Цель работы*:** получение навыков использования технологии COM для создания и управления документов MS WORD и MS EXCEL из проектов СП Delphi.

Задание. Требуется в проект, разработанный в лабораторной работе № 9, добавить функции автоматического формирования документов MS WORD и MS EXCEL с результатами выполнения первых двух запросов на выборку индивидуального задания. Каждый документ должен содержать форматированный заголовок отчета и результаты соответствующего запроса, оформленные в виде форматированной таблицы.

Порядок выполнения работы

1. *Модификация интерфейса проекта*. Добавьте к проекту компонент *WordApplication1* (страница *Servers* палитры компонентов). Установите значение *True* для свойств *AutoConnect* и *AutoQuit* (эти свойства отвечают за автоматическую загрузку и выгрузку из памяти сервера автоматизации после запуска приложения).

Добавьте к проекту компонент *WordDocument1* (страница *Servers* палитры компонентов).

Пометите в окно проекта компонент *MainMenu1* и создайте пункт меню *Отчеты* с подпунктами, заголовки которых соответствуют названиям отчетов, как показано на рисунке 78.

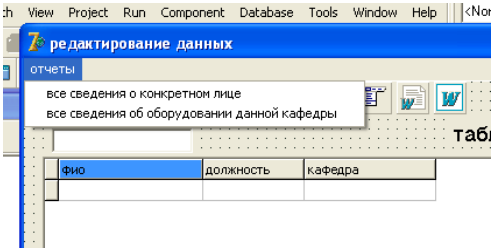

Рисунок 78 – **Фрагмент окна проекта с использованием COM-технологии**

2. *Создание процедуры формирования отчета в* MS Word. Создайте процедуру формирования отчета в MS Word по результатам

выполнения первого запроса на выборку. Код процедуры сформируйте на основе файла, содержащего пример кода формирования отчета и приведенного на рисунках 79–82. Фрагмент процедуры, приведенный на рисунке 79, можно переписать в свой проект без изменения (возможно изменение параметров шрифта).

```
procedure TForm1.N2Click(Sender: TObject);
var
  dial,NumColumns,filename,tempo,separator:oleVariant;
  s,text,text1:WideString;
  MyRange : Range;    {Область документа}
  Tabl : Table;       {Одна таблица}
  Pars : Paragraphs; {Массив параграфов}
  Par  : Paragraph;  {Один параграф}
  i,k:integer;
begin
  try
    //Задание параметров приложения
    WordApplication1.AutoQuit:=True;
    WordApplication1.ConnectKind:=ckRunningOrNew;
    WordApplication1.AutoConnect:=True;
    //Создание документа Word и его отображение
    WordApplication1.Documents.Creator;
    WordApplication1.Visible:=True;
    Worddocument1.Activate;
    //Отключение проверки орфографии и граматики в Word
    WordApplication1.Options.CheckSpellingAsYouType:=false;
    WordApplication1.Options.CheckGrammarAsYouType:=false;
    {Определение области документа и очистка ее
    на тот случай, если был выбран существующий файл}
    MyRange:=WordDocument1.Range(EmptyParam,EmptyParam);
    Tempo:=MyRange;
    MyRange.Select;
    MyRange.Cut;
    //Определение массива параграфов
    Pars:=WordDocument1.Paragraphs;
    //Добавление к массиву первого параграфа и задание его параметров
    Par:=Pars.Add(Tempo);
    //Выравнивание параграфа
    Par.Alignment:=wdAlignParagraphCenter;
    Par.Range.Font.Bold:=1; //Шрифт жирный
    Par.Range.Font.Size:=14;//Размер шрифта
```

Рисунок 79 – **Начальный фрагмент процедуры формирования отчета в Word с описанием переменных и заданием основных параметров области документа**

В фрагмент процедуры, приведенный на рисунке 80, необходимо внести следующие изменения:

- организовать ввод параметра искомого запроса;
- сформулировать заголовок отчета;
- ввести названия граф таблицы;
- получить текст запроса на языке SQL.

```
//Ввод параметра запроса и формирование заголовка отчета
s:=InputBox('Ввод','Введите фио','');
Par.Range.InsertBefore('Все сведения об оборудовании, числящимся за '+s);
//Определение конца области параграфа}
Tempo:=Par.Range.Get_End_;
MyRange:=WordDocument1.Range(Tempo);
{Формирование строки заголовков граф таблицы
 с использованием в качестве разделителя символа @}
text:=' № п/п @ Фамилия  @ Должность     @ Кафедра   '+
      '@ Наименование оборудования @ Количество @ Стоимость @';
//Реализация запроса на выборку из базы данных
AdoQuery1.SQL.Clear;
text1:='SELECT ответственный.фио, ответственный.должность,'+
       ' ответственный.кафедра, оборудование.наименование,'+
       ' оборудование.количество, оборудование.цена'+
       ' FROM ответственный INNER JOIN оборудование ON '+
       'ответственный.фио = оборудование.фио'+
       ' WHERE (((ответственный.фио)="'+s+'"));';
AdoQuery1.SQL.Add(text1);
AdoQuery1.Active:=True;
```

Рисунок 80 – **Фрагмент процедуры формирования отчета в Word с кодом реализации запроса на выборку**

В фрагмент процедуры, приведенный на рисунке 81, необходимо внести названия полей искомого запроса.

Текст заключительного фрагмента процедуры, приведенный на рисунке 82, необходимо скорректировать с учетом числа полей искомого запроса и требуемой ширины столбцов.

```
    //Формирование строки с результатами запроса
    for k:=1 to AdoQuery1.RecordCount do
      begin
         text1:='';
         text1:=text1+IntToStr(k)+'@';
         text:=text+text1;
         text1:=AdoQuery1.FieldByName('фио').Value+'@';
         text:=text+text1;
         text1:=AdoQuery1.FieldByName('должность').Value+'@';
         text:=text+text1;
         text1:=AdoQuery1.FieldByName('кафедра').Value+'@';
         text:=text+text1;
         text1:=AdoQuery1.FieldByName('наименование').Value+'@';
         text:=text+text1;
         text1:=IntToStr(AdoQuery1.FieldByName('количество').Value)+'@';
         text:=text+text1;
         text1:=IntToStr(AdoQuery1.FieldByName('цена').Value)+'@';
         text:=text+text1;
         AdoQuery1.Next;
      end;
    //Передача строки текста в Word и создание нового параграфа
    Tempo:=MyRange;
    Par:=Pars.Add(Tempo);
    Par.Range.Font.Size:=10;
    Par.Range.InsertBefore(Text);
```

Рисунок 81 – **Фрагмент процедуры формирования отчета в Word с кодом передачи результата запроса на выборку в таблицу**

```
    //Конвертация текста в таблицу
    Separator:='@';
    NumColumns:=7;
    MyRange.ConvertToTable(Separator,EmptyParam,NumColumns,
            EmptyParam,EmptyParam,EmptyParam,EmptyParam,
            EmptyParam,EmptyParam,EmptyParam,EmptyParam,
            EmptyParam,EmptyParam,EmptyParam,EmptyParam,EmptyParam);
    {Связывание переменной Tabl и таблицы, изменение размеров столбцов
     и выравнивание по центру}
    Tabl:=WordDocument1.Range.Tables.Item(1);
    Tabl.Columns.Item(1).SetWidth(30,wdAdjustNone);
    Tabl.Columns.Item(2).SetWidth(80,wdAdjustNone);
    Tabl.Columns.Item(3).SetWidth(80,wdAdjustNone);
    Tabl.Columns.Item(4).SetWidth(80,wdAdjustNone);
    Tabl.Columns.Item(5).SetWidth(80,wdAdjustNone);
    Tabl.Columns.Item(6).SetWidth(70,wdAdjustNone);
    Tabl.Columns.Item(7).SetWidth(70,wdAdjustNone);
    Tabl.Range.Paragraphs.Format.Alignment:=wdAlignParagraphCenter;
    Tempo:=Par.Range.Get_End_; //Определение конца области
  finally
    //Сохранение файла активного документа
    Wordapplication1.Activate;
    Dial:= wdDialogFileSaveAs;
    WordApplication1.Dialogs.Item(Dial).Show(EmptyParam);
  end;
end;
```

Рисунок 82 – **Заключительный фрагмент процедуры формирования отчета в Word**

3. *Создание процедуры формирования отчета в MS Excel.* Создайте процедуру формирования отчета в MS Excel по результатам выполнения второго запроса на выборку. Код процедуры сформируйте на основе файла, содержащего пример кода формирования отчета и приведенного на рисунках 83–86. Фрагмент процедуры, приведенный на рисунке 83, можно переписать в свой проект без изменения (возможно изменение адреса ячейки с заголовком таблицы).

```
procedure TForm1.N3Click(Sender: TObject);
var
  border,e,myrange,name,sheet,mysell:variant;
  s,s2:string;
  p1,p,i,j:longint;
begin
  //Создание объекта
  name:=CreateOleObject('Excel.Application');
  //Добавление рабочей книги
  name.WorkBooks.Add;
  //Добавление листа
  name.Sheets.Add;
  //Занесение в переменную sheet первого листа
  sheet:=name.Sheets.Item[1];
  {Организация доступа к ячейке, в которую
   будет помещен заголовок таблицы}
  mysell:=Sheet.Cells[1,5];
```

Рисунок 83 – **Начальный фрагмент процедуры формирования отчета в Excel с описанием переменных и заданием основных параметров области документа**

В фрагмент процедуры, приведенный на рисунке 84, необходимо внести следующие изменения:

- организовать ввод параметра искомого запроса;
- сформулировать заголовок отчета;
- получить текст запроса на языке SQL.

```
//Ввод параметра запроса и формирование заголовка отчета
s:=InputBox('Ввод','Введите код клиента','');
//Задание параметров ячейки с заголовком таблицы
mysell.Value:='Все сведения о заказах клиента с номером '+s;
mysell.HorizontalAlignment:=xlHAlignCenter;
mysell.VerticalAlignment:=xlVAlignCenter;
mysell.Font.Size:=20;
mysell.Font.Bold:=True;
//Реализация запроса на выборку из базы данных
s2:='SELECT Клиенты.КодКл, Заказы.ДатаЗак, Заказы.НомЗак,'+
    ' СтрокиЗаказов.КодТов, СтрокиЗаказов.Количество'+
    ' FROM Товары INNER JOIN ((Клиенты INNER JOIN Заказы'+
    ' ON Клиенты.КодКл = Заказы.КодКл) INNER JOIN '+
    'СтрокиЗаказов ON Заказы.НомЗак = СтрокиЗаказов.НомЗак)'+
    ' ON Товары.КодТов = СтрокиЗаказов.КодТов'+
    ' WHERE (((Клиенты.КодКл)='''+s+'''));';
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(s2);
ADOQuery1.Active:=True;
p:=ADOQuery1.FieldCount;//Количество граф в таблице
```

Рисунок 84 – **Фрагмент процедуры формирования отчета в Excel с кодом реализации запроса на выборку**

В коде фрагментов процедуры, приведенных на рисунках 85 и 86, можно изменить параметры ячеек.

```
i:=2;
//Формирование шапки (головки) таблицы
for j:=1 to p do
  begin
    mysell:=Sheet.Cells[i,j];
    //Параметры границы
    border:=mysell.Borders;
    border.Weight:=4;
    //Значение ячейки с названием графы таблицы
    mysell.Value:=Adoquery1.Recordset.Fields[j-1].Name;
    //Выравнивание информации по центру
    mysell.Horizontalalignment:=xlHAlignCenter;
    mysell.Verticalalignment:=xlVAlignCenter;
    //Установка размера шрифта
    mysell.Font.Size:=12;
    //Автоподбор ширины ячейки
    mysell.Columns.AutoFit;
    //Дополнительное расширение ячейки
    mysell.ColumnWidth:=mysell.ColumnWidth+10;
    //Установка вида и цвета шрифта
    mysell.Font.Bold:=True;
    mysell.Font.Color:=Clred;
  end;
```

Рисунок 85 – **Фрагмент процедуры формирования отчета в Excel с кодом создания шапки таблицы**

```
  //Заполнение таблицы
  p1:=ADOQuery1.RecordSet.RecordCount;
  for i:=3 to p1+2 do
    begin
      for j:=1 to p do
        begin
          mysell:=Sheet.Cells[i,j];
          border:=mysell.Borders;
          border.Weight:=2;
          mysell.Value:=ADOQuery1.RecordSet.Fields[j-1].Value;
          mysell.HorizontalAlignment:=xlHAlignCenter;
          mysell.VerticalAlignment:=xlVAlignCenter;
          mysell.Font.Size:=12;
        end;
      ADOQuery1.RecordSet.MoveNext;
    end;
  //Отображение документа
  name.Visible:=True;
end;
```

Рисунок 86 – **Заключительный фрагмент процедуры формирования отчета в Excel**

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Автоматизация работы пользователя в среде СУБД Access : пособие для студентов всех специальностей : В 2 ч. / авт.-сост. Л. М. Ашарчук, Т. В. Астапкина, И. В. Дубинина. - Гомель : ГКИ, 2000. – 48 с.

Алгоритмизация и программирование : пособие для студентов специальности 1-25 01 07 «Экономика и управление на предприятии» специализации 1-25 01 07 02 «Экономическая информатика», специальности 1-26 03 01 «Управление информационными ресурсами» / авт.-сост. : С.М. Мовшович, О.А. Кравченко. – Гомель : Бел. торгово-экон. ун-т потребит. кооп., 2008. – 104 с.

Вальвачев, А. Н. Программирование на языке Паскаль для персональных ЭВМ ЕС : справ. пособие / А. Н. Вальвачев, В. С. Крисевич. – Минск : Выш. шк., 1989. – 223 с.

Гофман, В. Delphi 6. Наиболее полное руководство / В. Гофман. – СПб. : BHV, 2001. – 1129 с.

Дарахвелидзе, П. Г. Программирование в Delphi 5 / П. Г. Дарахвелидзе. – СПб. : BHV, 2000. – 767 с.

ER-метод проектирования баз данных и его реализация в среде СУБД Access : пособие для студентов экон. спец. / С. М. Мовшович, К. Г. Сулейманов. – Гомель : Бел. торгово-экон. ун-т потребит. кооп., 2003. – 140 с.

Корняков, В. Программирование документов MSOffice в Delphi / В. Корняков. – СПб. : BHV, 2005. – 489 с.

Культин, Н. Б. Основы программирования в Delphi 7 / Н. Б. Культин. – СПб. : BHV, 2003. – 566 с.

Левчук, Е. А. Технологии организации, хранения и обработки данных : учеб. пособие для вузов / Е. А. Левчук. – Минск : Выш. шк., 2005. – 239 с.

Левчук, Е. А. Основы информатики и вычислительной техники: пособие для студентов / Е.А. Левчук. – Гомель : Бел. торгово.-экон. ун-т потреб. кооп., 2003. – 128 с.

Обработка экономической информации в системе программирования Delphi : пособие по дисциплине «Алгоритмизация и программирование» для студентов специальности 1-26 03 01 «Управление информационными ресурсами» / авт.-сост. : С. М. Мовшович, О. А. Кравченко. – Гомель : учреждение образования «Белорусский торгово-экономический университет потребительской кооперации», 2011. – 128 с.

Фаронов, В. В. Турбо Паскаль 7.0. Начальный курс : учеб. пособие / В. В. Фаронов. – М. : КНОРУС, 2006. – 576 с.

Фаронов, В. В. Delphi. Программирование на языке высокого уровня / В. В. Фаронов. – СПб. : Питер, 2004. – 640 с.

СОДЕРЖАНИЕ

Учебное издание

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Практикум
для студентов специальности 1-26 03 01
«Управление информационными ресурсами»

Авторы-составители:
Коробейникова Евгения Васильевна
Мовшович Семен Михайлович
Кравченко Ольга Алексеевна

Редактор Н. В. Коптелова
Технический редактор Н. Н. Короедова
Компьютерная верстка Л. Г. Макарова

Подписано в печать 20.03.12. Бумага типографская № 1.
Формат 60 × 84 $^{1}/_{16}$. Гарнитура Таймс. Ризография.
Усл. печ. л. 5,81. Уч.-изд. л. 5,8. Тираж 110 экз.
Заказ № 19-03-12.

Учреждение образования
«Белорусский торгово-экономический
университет потребительской кооперации».
ЛИ № 02330/0494302 от 04.03.2009 г.
246029, г. Гомель, просп. Октября, 50.

Отпечатано в учреждении образования
«Белорусский торгово-экономический
университет потребительской кооперации».
246029, г. Гомель, просп. Октября, 50.

БЕЛКООПСОЮЗ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БЕЛОРУССКИЙ ТОРГОВО-ЭКОНОМИЧЕСКИЙ
УНИВЕРСИТЕТ ПОТРЕБИТЕЛЬСКОЙ КООПЕРАЦИИ»

Кафедра информационно-вычислительных систем

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Практикум
для студентов специальности 1-26 03 01
«Управление информационными ресурсами»

Гомель 2012