Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Институт повышения квалификации
и переподготовки кадров

Кафедра «Информатика»

ТЕХНОЛОГИИ КОМПОНЕНТНОГО ПРОГРАММИРОВАНИЯ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ
по одноименной дисциплине
для слушателей специальности 1-40 01 73
«Программное обеспечение информационных систем»
заочной формы обучения

Гомель 2014

УДК 004.43(075.8)
ББК 32.973-018я73
Т38

Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем ГГТУ им. П. О. Сухого
(*протокол № 5 от 30.12.2013 г.*)

Составитель: Н. В. Самовендюк

Рецензент: доц. каф. «Информационные технологии» ГГТУ им. П. О. Сухого
канд. техн. наук *К. С. Курочка*

Технологии компонентного программирования : лаборатор. практикум по одноим. дисциплине для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем» заоч. формы обучения / сост. Н. В. Самовендюк. – Гомель : ГГТУ им. П. О Сухого, 2014. – 89 с. Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: http://library.gstu.by. – Загл. с титул. экрана.

Т38

В лабораторном практикуме на практических примерах рассматриваются базовые понятия Com-технологий, вопросы использования интерфейсов, взаимодействия клиентских приложений с Com-серверами, реализованных в виде динамически загружаемых библиотек или во внешней программе, управления Com-объектами во время выполнения программы, создания собственных Com-серверов.

Изучение дисциплина проводится на базе объектно-ориентированного языка Delphi.

Для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем» заочной формы обучения ИПК и ПК.

УДК 004.43(075.8)
ББК 32.973-0186я73

Содержание

Лабораторная работа № 1
тема: «Создание пользовательских классов»

Цель работы: получить навыки создания и использования собственных классов объектов

Краткие теоретическая сведения:

Классы

Для поддержки ООП (объектно-ориентированное программирование) в язык Delphi введены объектные типы данных, с помощью которых одновременно описываются данные и операции над ними. Объектные типы данных называют классами, а их экземпляры — объектами.

Классы объектов определяются в секции **type** глобального блока. Описание класса начинается с ключевого слова **class** и заканчивается ключевым словом **end.** По форме объявления классы похожи на обычные записи, но помимо полей данных могут содержать объявления пользовательских процедур и функций. Такие процедуры и функции обобщенно называют методами, они предназначены для выполнения над объектами различных операций. Ниже приведено описание класса **Pryamougolnik**, который предназначен для вычисления площади и периметра прямоугольника

```
Type
    TPryamougolnik=class
        //Поля
        hir,vis:real;
        //Методы
        function Ploshad:real;
        function Perimetr:real;
end;
```

Класс содержит поля (hir, vis) и методы (Ploshad, Perimetr). Заголовки методов, всегда следующие за списком полей, играют роль упреждающих описаний. Программный код методов пишется отдельно от определения класса.

Объекты

Чтобы от описания класса перейти к объекту, следует выполнить соответствующее объявление в секции **var**:

Var a,b:TPryamougolnik;

Объекты в среде Delphi являются динамическими данными, т.е. распределяются в динамической памяти. Поэтому переменная **a** — это просто ссылка на экземпляр (объект в памяти), которого физически еще не существует. Чтобы сконструировать объект (выделить память для экземпляра) класса **TPryamougolnik** и связать с ним переменную Reader, нужно в тексте программы поместить следующий оператор:

a:=TPryamougolnik.Create;

Create — это так называемый конструктор объекта; он всегда присутствует в классе и служит для создания и инициализации экземпляров. При создании объекта в памяти выделяется место только для его полей. Методы, как и обычные процедуры и функции, помещаются в область кода программы; они умеют работать с любыми экземплярами своего класса и не дублируются в памяти.

После создания объект можно использовать в программе: получать и устанавливать значения его полей, вызывать его методы. Доступ к полям и методам объекта происходит с помощью уточненных имен, например: **a.Ploshad**.

Если объект становится ненужным, он должен быть удален вызовом специального метода **Destroy**, например: **a.Destroy**.

Destroy — это так называемый деструктор объекта; он присутствует в классе наряду с конструктором и служит для удаления объекта из динамической памяти.

Конструкторы и деструкторы

Объявление конструкторов и деструкторов похоже на объявление обычных методов с той лишь разницей, что вместо зарезервированных слов **function** и **procedure** используются слова **constructor** и **destructor**.

Приведем возможную реализацию конструктора:

```
constructor TPryamougolnik.Create;
begin
        //инициализация полей экземпляра класса
        hir:=1;
        vis:=1;
 end;
```

Методы

Процедуры и функции, предназначенные для выполнения над объектами действий, называются методами. Предварительное объявление методов выполняется при описании класса в секции **interface** модуля, а их программный код записывается в секции **implementation**. Однако в отличие от обычных процедур и функций заголовки методов должны иметь уточненные имена, т.е. содержать наименование класса. Приведем возможную реализацию одного из методов в классе TPryamougolnik:

```
function TPryamougolnik.Ploshad:real;
begin
        result:=hir*vis;
end;
```

Обратите внимание, что внутри методов обращения к полям и другим методам выполняются как к обычным переменным и подпрограммам без уточнения экземпляра объекта. Такое упрощение достигается путем использования в пределах метода псевдопеременной **Self** (стандартный идентификатор). Физически **Self** представляет собой дополнительный неявный параметр, передаваемый в метод при вызове. Этот параметр и указывает экземпляр объекта, к которому данный метод применяется.

Свойства

Помимо полей и методов в объектах существуют свойства. При работе с объектом свойства выглядят как поля: они принимают значения и участвуют в выражениях. Но в отличие от полей свойства

не занимают места в памяти, а операции их чтения и записи ассоциируются с обычными полями или методами. Это позволяет создавать необходимые сопутствующие эффекты при обращении к свойствам.

Объявление свойства выполняется с помощью зарезервированного слова **property**, например:

```
property Fhir:real read hir write hir;
property Fvis:real read vis write vis;
```

Ключевые слова **read** и **write** называются спецификаторами доступа. После слова read указывается поле или метод, к которому происходит обращение при чтении (получении) значения свойства, а после слова write — поле или метод, к которому происходит обращение при записи (установке) значения свойства. Чтобы имена свойств не совпадали с именами полей, последние принято писать с буквы **F** (от англ. field).

Обращение к свойствам выглядит в программе как обращение к полям:

```
a.Fhir:=strtofloat(edit1.text);
a.Fvis:=strtofloat(edit2.text);
```

Технология объектно-ориентированного программирования в среде Delphi предписывает избегать прямого обращения к полям, создавая вместо этого соответствующие свойства. Это упорядочивает работу с объектами, изолируя их данные от непосредственной модификации.

Наследование

В языке Delphi существует предопределенный класс **TObject**, который служит неявным предком тех классов, для которых предок не указан. Это означает, что объявление

```
Type
    TPryamougolnik = class
    ……
end;
```

эквивалентно следующему:

```
Type
        TPryamougolnik = class(TObject)
        ……
end;
```

Класс TObject выступает корнем любой иерархии классов. Он содержит ряд методов, которые по наследству передаются всем остальным классам. Среди них конструктор Create, деструктор Destroy, метод Free и некоторые другие методы.

Классы в программных модулях

Классы очень удобно собирать в модули. При этом их описание помещается в секцию **interface**, а код методов — в секцию **implementation**.

Программист может разграничить доступ к атрибутам своих объектов для других программистов (и себя самого) с помощью специальных ключевых слов: **private, protected, public, published**.

Private. Все, что объявлено в секции private недоступно за пределами модуля. Секция private позволяет скрыть те поля и методы, которые относятся к так называемым особенностям реализации.

Public. Поля, методы и свойства, объявленные в секции public не имеют никаких ограничений на использование, т.е. всегда видны за пределами модуля. Все, что помещается в секцию public, служит для манипуляций с объектами и составляет программный интерфейс класса.

Protected. Поля, методы и свойства, объявленные в секции protected, видны за пределами модуля только потомкам данного класса; остальным частям программы они не видны. Так же как и private, директива protected позволяет скрыть особенности реализации класса, но в отличие от нее разрешает другим программистам порождать новые классы и обращаться к полям, методам и свойствам, которые составляют так называемый интерфейс разработчика. В эту секцию обычно помещаются виртуальные методы.

Published. Устанавливает правила видимости те же, что и директива public. Особенность состоит в том, что для элементов,

помещенных в секцию published, компилятор генерирует информацию о типах этих элементов. Эта информация доступна во время выполнения программы, что позволяет превращать объекты в компоненты визуальной среды разработки. Секцию published разрешено использовать только тогда, когда для самого класса или его предка включена директива компилятора **$TYPEINFO**.

Задание: Требуется разработать проект в СП Delphi, в котором должен быть создан класс в соответствии с вариантом.

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:

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

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

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

a и b – длины сторон прямоугольника;

d – длина диагонали прямоугольника;

S – площадь прямоугольника.

Вариант 9

Разработайте класс *Параллелограмм*. Свойства: длины сторон параллелограмма и острый угол. Методы: площадь и периметр параллелограмма.

На основе разработанного класса решите следующую задачу: для заданных сторон и острого угла двух параллелограммов определите, у какого параллелограмма больший периметр, а у какого большая площадь. Ответ выведите на форму.

Формулы для расчета:

$S = ab \sin \alpha$, P = 2(a+b)

где a и b – длины сторон параллелограмма;

α – острый угол параллелограмма;

P – периметр параллелограмма;

S – площадь параллелограмма.

Вариант 10

Разработайте класс *Шар*. Свойства: радиус. Методы: площадь поверхности и объём шара.

На основе разработанного класса решите следующую задачу: для заданных радиусов двух шаров определите, у какого шара больший объём, а у какого большая площадь поверхности. Ответ выведите на форму.

Формулы для расчета:

$$S = 4\pi R^2,\ V = \frac{4}{3}\pi R^3$$

где R – радиус шара;

Пояснения к выполнению 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.

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

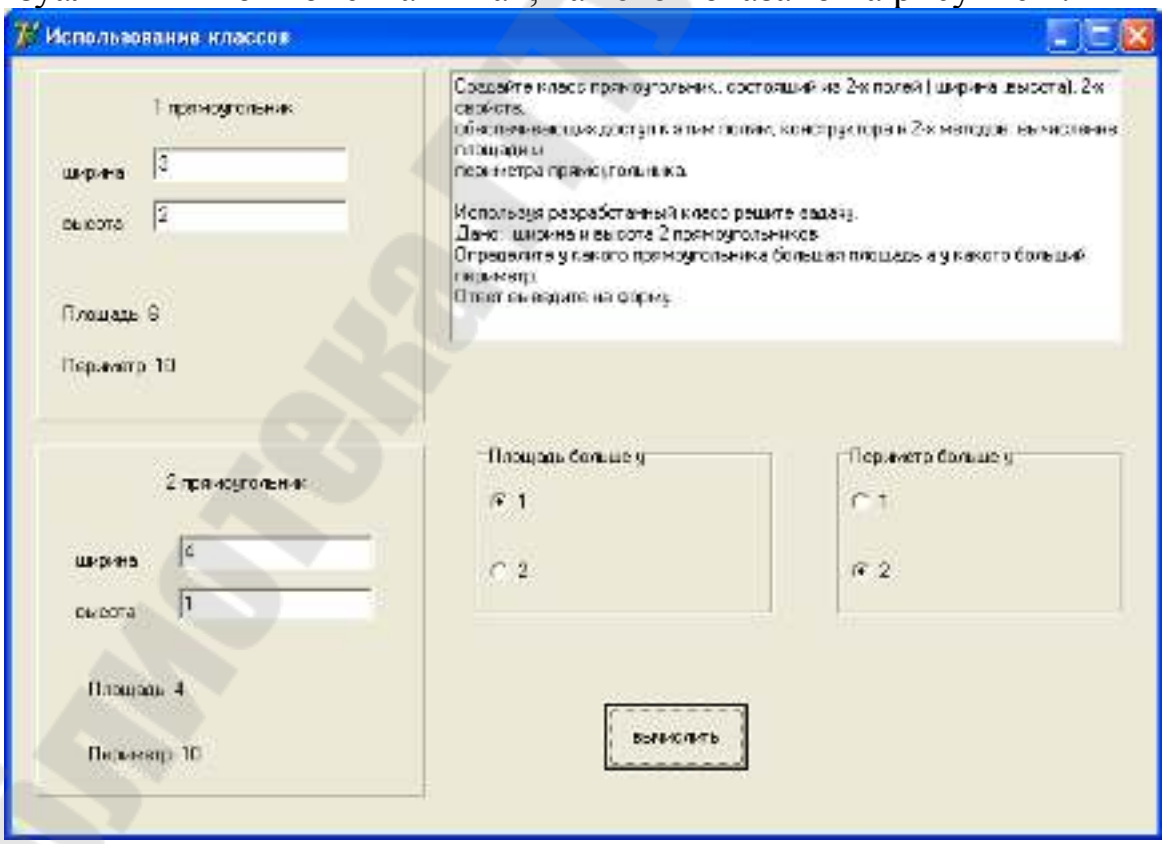


Рисунок 1 – **Интерфейс проекта создания и использования класса**

2. ***Описание пользовательского класса***. В конец интерфейсной части проекта необходимо записать конструкции, описывающие создаваемый класс *Pramougolnik*, включающее описание полей свойств и объявление методов класса. Кроме того, описаны переменные *a* и *b*, предназначенные для работы с двумя объектами – прямоугольниками.

```
TPryamougolnik=class
   private
    hir,vis:real;
   published
     property Fhir:Real read hir write hir;
     property FVis:real read vis write vis;
   public
    Constructor Create;
    function Ploshad:real;
    function Perimetr:real;
 end;

var
 Form1: TForm1;
 a,b:TPryamougolnik;
```

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

```
constructor TPryamougolnik.Create;
 begin
 inherited;
   hir:=1;
   vis:=1;
 end;
function TPryamougolnik.Ploshad:real;
  begin
    result:=hir*vis;
  end;
function TPryamougolnik.Perimetr:real;
  begin
    result:=2*(hir+vis);
  end;
```

4. ***Использование пользовательского класса***. Использование объектов созданного класса происходит в процедуре обработки щелчка по кнопке *Вычислить* (см. 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.

```
procedure TForm1.Button1Click(Sender: TObject);

begin
 a:=TPryamougolnik.Create;
 b:=TPryamougolnik.Create;
 a.Fhir:=StrTofloat(Edit1.text);
 a.FVis:=strtofloat(edit2.text);
 b.Fhir:=StrTofloat(Edit3.text);
 b.FVis:=strtofloat(edit4.text);
 if a.Ploshad>b.Ploshad then Radiogroup1.ItemIndex:=0 else
Radiogroup1.ItemIndex:=1;
 if a.Perimetr>b.Perimetr then Radiogroup2.ItemIndex:=0 else
Radiogroup2.ItemIndex:=1;
 label5.Caption:=label5.Caption+'  '+ floattostr(a.Ploshad);
 label6.Caption:=label6.Caption+'  '+ floattostr(a.Perimetr);
 label9.Caption:=label9.Caption+'  '+ floattostr(b.Ploshad);
 label10.Caption:=label10.Caption+'  '+ floattostr(b.Perimetr);
end;
```

Контрольные вопросы

1. Что такое объектный тип данных?
2. Что включается в объявление класса?
3. Как создать экземпляр класса?
4. Что такое конструктор?
5. Что такое деструктор?
6. Что такое методы класса?
7. В какой части модуля описывается реализация методов?
8. Что такое свойство объекта?
9. Какие спецификаторы доступа Вы знаете?
10. Что такое наследование?

11. От какого класса наследуются все остальные?
12. Какие основные методы класса TObject используются в классах-наследниках?

Лабораторная работа № 2
тема: «Создание и вызов динамических библиотек»

Цель работы: получение навыков создания, статического и динамического вызова подпрограмм динамических библиотек при разработке проектов

Краткие теоретические сведения:

Динамически загружаемые библиотеки

Динамически загружаемая библиотека (от англ. dynamically loadable library) — это библиотека подпрограмм, которая загружается в оперативную память и подключается к использующей программе во время ее работы (а не во время компиляции и сборки). Файлы динамически загружаемых библиотек в среде Windows обычно имеют расширение .dll (от англ. Dynamic-Link Library). Для краткости мы будем использовать термин динамическая библиотека, или просто библиотека, подразумевая DLL-библиотеку.

Несколько разных программ могут использовать в работе общую динамически загружаемую библиотеку. При этом операционная система в действительности загружает в оперативную память лишь одну копию библиотеки и обеспечивает совместный доступ к ней со стороны всех программ. Кроме того, такие библиотеки могут динамически загружаться и выгружаться из оперативной памяти по ходу работы программы, освобождая ресурсы системы для других задач.

Одно из важнейших назначений динамически загружаемых библиотек — это взаимодействие подпрограмм, написанных на разных языках программирования. Например, вы можете свободно использовать в среде Delphi динамически загружаемые библиотеки, разработанные в других системах программирования с помощью языков C и C++. Справедливо и обратное утверждение — динамически загружаемые библиотеки, созданные в среде Delphi, можно подключать к программам на других языках программирования.

Структура библиотеки

По структуре исходный текст библиотеки похож на исходный текст программы, за исключением того, что текст библиотеки начинается с ключевого слова **library**, а не слова **program**. Например:

```
library SortLib;
```

После заголовка следуют секции подключения модулей, описания констант, типов данных, переменных, а также описания процедур и функций. Процедуры и функции — это главное, что должно быть в динамически загружаемой библиотеке, поскольку лишь они могут быть экспортированы.

Если в теле библиотеки объявлены некоторые процедуры:

```
procedure BubleSort(var Arr: array of Integer);
procedure QuickSort(var Arr: array of Integer);
```

необходимо поместить имена процедур в специальную секцию **exports**, например:

```
exports
    BubleSort,
    QuickSort;
```

Перечисленные в секции exports процедуры и функции отделяются запятой, а в конце всей секции ставится точка с запятой. Секций exports может быть несколько, и они могут располагаться в программе произвольным образом.

Ниже приведен пример исходного текста простейшей динамически загружаемой библиотеки SortLib. Она содержит единственную процедуру BubleSort, сортирующую массив целых чисел методом «пузырька»:

```
library SortLib;

procedure BubleSort(var Arr: array of Integer);
var
    I, J, T: Integer;
```

```
begin
        for I := Low(Arr) to High(Arr) - 1 do
                for J := I + 1 to High(Arr) do
                        if Arr[I] > Arr[J] then
                        begin
                        T := Arr[I];
                        Arr[I] := Arr[J];
                        Arr[J] := T;
                        end;
end;

exports
        BubleSort;

begin
end.
```

Исходный текст динамически загружаемой библиотеки заканчивается операторным блоком begin...end, в который можно вставить любые операторы для подготовки библиотеки к работе. Эти операторы выполняются во время загрузки библиотеки основной программой. Наша простейшая библиотека SortLib не требует никакой подготовки к работе, поэтому ее операторный блок пустой.

Соглашения о вызове подпрограмм

В различных языках программирования используются различные правила вызова подпрограмм. Для совместимости с ними в языке Delphi существуют директивы **register, stdcall, pascal и cdecl**. Применение этих директив становится особенно актуальным при разработке динамически загружаемых библиотек, которые используются в программах, написанных на других языках программирования.

Механизм вызова подпрограмм основан на использовании **стека**.

Стек — это область памяти, в которую данные помещаются в прямом порядке, а и извлекаются в обратном. Очередность работы с элементами в стеке обозначается термином LIFO (от англ. Last In, First Out — последним вошел, первым вышел).

Для каждой программы на время работы создается свой стек. Через него передаются параметры подпрограмм и в нем же сохраняются адреса возврата из этих подпрограмм. Именно благодаря стеку подпрограммы могут вызывать друг друга, или даже рекурсивно сами себя.

Вызов подпрограммы состоит из «заталкивания» в стек всех аргументов и адреса следующей команды (для воврата к ней), а затем передачи управления на начало подпрограммы. По окончании работы подпрограммы из стека извлекается адрес воврата с передачей управления на этот адрес; одновременно с этим из стека выталкиваются аргументы. Происходит так называемая очистка стека. Это общая схема работы и у нее бывают разные реализации. В частности, аргументы могут помещаться в стек либо в прямом порядке (слева направо, как они перечислены в описании подпрограммы), либо в обратном порядке (справа налево), либо вообще, не через стек, а через свободные регистры процессора для повышения скорости работы. Кроме того, очистку стека может выполнять либо вызываемая подпрограмма, либо вызывающая программа. Выбор конкретного соглашения о вызове обеспечивают директивы **register, pascal, cdecl и stdcall**. Их смысл поясняет следующая таблица.

Директива	Порядок занесения аргументов в стек	Кто отвечает за очистку стека	Передача аргументов через регистры
register	Слева направо	Подпрограмма	Да
pascal	Слева направо	Подпрограмма	Нет
cdecl	Справа налево	Вызывающая программа	Нет
stdcall	Справа налево	Подпрограмма	Нет

Соглашение stdcall, изначально предназначенное для вызова подпрограмм операционной системы, лучше всего подходит для взаимодействия программ и библиотек, написанных на разных языках программирования. Все программы так или иначе используют функции операционной системы, следовательно они обязательно поддерживают соглашение stdcall.

Использование библиотеки в программе

Для того чтобы в прикладной программе воспользоваться процедурами и функциями библиотеки, необходимо выполнить так называемый импорт. Импорт обеспечивает загрузку библиотеки в оперативную память и привязку записанных в программе команд вызова к адресам соответствующих процедур и функций библиотеки. Существуют два способа импорта, отличающихся по удобству и гибкости программирования:

- статический импорт (обеспечивается директивой компилятора **external**);
- динамический импорт (обеспечивается функциями **LoadLibrary** и **GetProcAddress**).

Статический импорт

Статический импорт является более удобным, а динамический — более гибким.

При статическом импорте все действия по загрузке и подключению библиотеки выполняются автоматически операционной системой во время запуска главной программы. Чтобы задействовать статический импорт, достаточно просто объявить в программе процедуры и функции библиотеки как внешние. Это делается с помощью директивы external, например:

```
procedure BubleSort(var Arr: array of Integer); stdcall; external 'SortLib.dll';
```

После ключевого слова external записывается имя двоичного файла библиотеки в виде константной строки или константного строкового выражения.

Поместив в программу приведенное выше объявления, можно вызвать процедуры BubleSort, как будто она являются частью самой программы.

Динамический импорт

Действия по загрузке и подключению библиотеки (выполняемые при статическом импорте автоматически) можно проделать самостоятельно, обратившись к стандартным функциям

операционной системы. Таким образом, импорт можно произвести динамически во время работы программы (а не во время ее запуска).

Для динамического импорта необходимо загрузить библиотеку в оперативную память вызовом функции **LoadLibrary**, а затем извлечь из нее адреса подпрограмм с помощью функции **GetProcAddress**. Полученные адреса нужно сохранить в процедурных переменных соответствующего типа. После этого вызов подпрограмм библиотеки может выполняться путем обращения к процедурным переменным. Для завершения работы с библиотекой необходимо вызвать функцию **FreeLibrary**.

Рассмотрим динамический импорт на примере консольного приложения **TestDynamicImport**:

```
program TestDynamicImport;

{$APPTYPE CONSOLE}

uses
      Windows;

type
   TBubleSortProc = procedure (var Arr: array of Integer); stdcall;

var
   BubleSort: TBubleSortProc; // указатель на функцию BubleSort
   LibHandle: HModule;        // описатель библиотеки
   Arr: array [0..9] of Integer;
   I: Integer;

begin
   LibHandle := LoadLibrary('SortLib.dll');
   if LibHandle <> 0 then
      begin
      @BubleSort := GetProcAddress(LibHandle, 'BubleSort');
      if (@BubleSort <> nil) then
            begin
                  Randomize;
                  for I := Low(Arr) to High(Arr) do
                  Arr[I] := Random(100);
```

```
                BubleSort(Arr);
                for I := Low(Arr) to High(Arr) do
                Write(Arr[I], ' ');
                Writeln;
            end
        else
        Writeln('Ошибка отсутствия процедуры в библиотеке.');
    FreeLibrary(LibHandle);
    end
    else
    Writeln('Ошибка загрузки библиотеки.');
    Writeln('Press Enter to exit...');
    Readln;
end.
```

В программе определен процедурный тип данных, который по списку параметров и правилу вызова (stdcall) соответствуют подпрограмме сортировки BubleSort в библиотеке:

```
type
    TBubleSortProc = procedure (var Arr: array of Integer); stdcall;
```

Этот тип данных нужен для объявления процедурной переменных, в которой сохраняется адрес подпрограммы:

```
var
    BubleSort: TBubleSortProc;
```

В секции var объявлена также переменная для хранения целочисленного описателя библиотеки, возвращаемого функцией LoadLibrary:

```
var
    LibHandle: HModule;
```

Программа начинает свою работу с того, что вызывает функцию LoadLibrary, в которую передает имя файла DLL-библиотеки. Функция возвращает описатель библиотеки, который сохраняется в переменной LibHandle:

```
LibHandle := LoadLibrary('SortLib.dll');
    if LibHandle <> 0 then
    begin
    ...
    end
```

Если значение описателя отлично от нуля, значит библиотека была найдена на диске и успешно загружена в оперативную память. Убедившись в этом, программа обращается к функции GetProcAddress за адресом подпрограммы. Полученный адрес сохраняется в соответствующей процедурной переменной:

```
@BubleSort := GetProcAddress(LibHandle, 'BubleSort');
```

Обратите внимание на использование символа @ перед именем переменной. Он говорит о том, что выполняется не вызов подпрограммы, а работа с ее адресом.

Если этот адрес отличен от значения **nil**, значит подпрограмма с указанным именем была найдена в библиотеке и ее можно вызвать путем обращения к процедурной переменной:

```
if (@BubleSort <> nil) then
    begin
    ...
    BubleSort(Arr);
    end
```

По окончании сортировки программа выгружает библиотеку вызовом функции FreeLibrary.

Динамический импорт в сравнении со статическим требует значительно больше усилий на программирование, но он имеет ряд преимуществ:

- Более эффективное использование ресурсов оперативной памяти по той причине, что библиотеку можно загружать и выгружать по мере надобности;
- Динамический импорт помогает в тех случаях, когда некоторые процедуры и функции могут отсутствовать в библиотеке. При статическом импорте такие ситуации обрабатывает операционная система, которая выдает

сообщение об ошибке и прекращает работу программы. Однако при динамическом импорте программа сама решает, что ей делать, поэтому она может отключить часть своих возможностей и работать дальше.

Динамический импорт отлично подходит для работы с библиотеками драйверов устройств. Он, например, используется самой средой Delphi для работы с драйверами баз данных.

Задание: Требуется разработать проект в СП Delphi, использующий динамически загружаемую библиотеку, в которой должен быть описан класс в соответствии с вариантом с возможностью экспорта методов класса. Подключение библиотеки осуществить с помощью динамического импорта.

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.

Вариант 9

Разработайте класс *Параллелограмм*. Свойства: длины сторон параллелограмма и острый угол. Методы: площадь и периметр параллелограмма.

На основе разработанного класса решите следующую задачу: для заданных сторон и острого угла двух параллелограммов определите, у какого параллелограмма больший периметр, а у какого большая площадь. Ответ выведите на форму.

Формулы для расчета:

$S = ab\sin\alpha$, P = 2(a+b)

где a и b – длины сторон параллелограмма;

α – острый угол параллелограмма;

P – периметр параллелограмма;

S – площадь параллелограмма.

Вариант 10

Разработайте класс *Шар*. Свойства: радиус. Методы: площадь поверхности и объём шара.

На основе разработанного класса решите следующую задачу: для заданных радиусов двух шаров определите, у какого шара больший объём, а у какого большая площадь поверхности. Ответ выведите на форму.

Формулы для расчета:

$$S = 4\pi R^2,\ V = \frac{4}{3}\pi R^3$$

где R – радиус шара;

Пояснения к выполнению работы:

Задание: Требуется разработать проект в СП Delphi, использующий динамически загружаемую библиотеку, в которой должен быть описан класс *Прямоугольник*, состоящий из двух полей (ширина, высота), двух свойств, обеспечивающих доступ к этим полям, конструктора и двух методов: вычисление площади и периметра прямоугольника, с возможностью экспорта методов класса 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. Подключение библиотеки осуществить с помощью статического импорта для первого экземпляра класса и с помощью динамического импорта для второго экземпляра класса.

1. ***Создание динамически загружаемой библиотеки.***

Запускаем систему Delphi и выбераем в меню команду File | New | Other... . В диалоговом окне, которое откроется на экране, выбераем значок с подписью DLL Wizard и нажимем кнопку OK (рисунок 1).

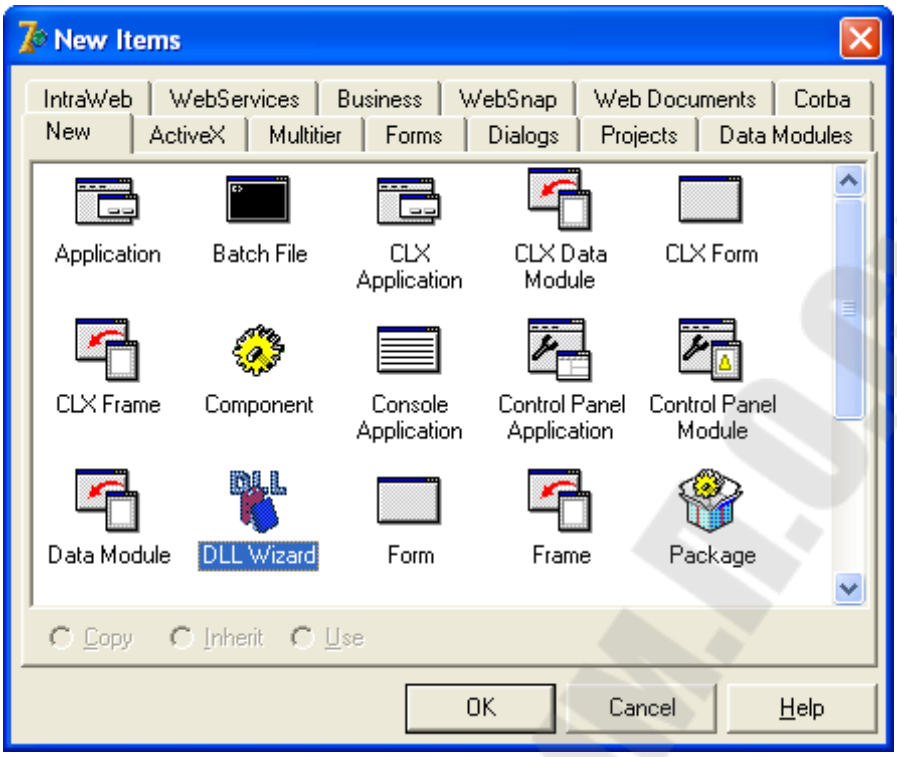


Рисунок 1 - Окно выбора нового проекта, в котором выделен пункт DLL Wizard

В описание типов описываем класс, реализующий поставленную задачу

```
type Prym2=class(TObject)
  private
  fa,fb:real;
  published
  public
  constructor Create;
  procedure setarg(a,b:real);
  function Plosh:real;
  function Per:real;
  end;
```

Описываем экземпляр нашего класса

```
Var Pr:Prym2;
```

Имплементируем методы класса

```
constructor Prym2.Create;
begin
inherited Create;
fa:=1;
fb:=1;
end;

procedure Prym2.setarg(a,b:real);
begin
fa:=a;
fb:=b;
end;

function Prym2.Plosh:real;
begin
result:= fa*fb;
end;

function Prym2.Per:real;
begin
result:= 2*(fa+fb);
end;
```

Добавляем функции, которые будут экспортироваться библиотекой

```
procedure setarg(a,b:real);
begin
Pr.setarg(a,b);
end;

function pl:real;
begin
result:=Pr.Plosh;
end;

function per:real;
begin
result:=Pr.Per;
```

```
end;

procedure createobject;
begin
Pr:=Prym2.Create;
end;

procedure releaseobject;
begin
Pr.Free;
end;
```

Добавляем описанные функции в блок экспорта

```
exports
setarg,
pl,
per,
createobject,
releaseobject;
```

Сохраняем проект под необходимым именем. Для завершения создания библиотеки используют команду **Project | Build Project**.

2. ***Создание интерфейса пользователя проекта***.

Создаем новый проект в Delphi, заполним окно формы визуальными компонентами так, как это показано на рисунке 2.

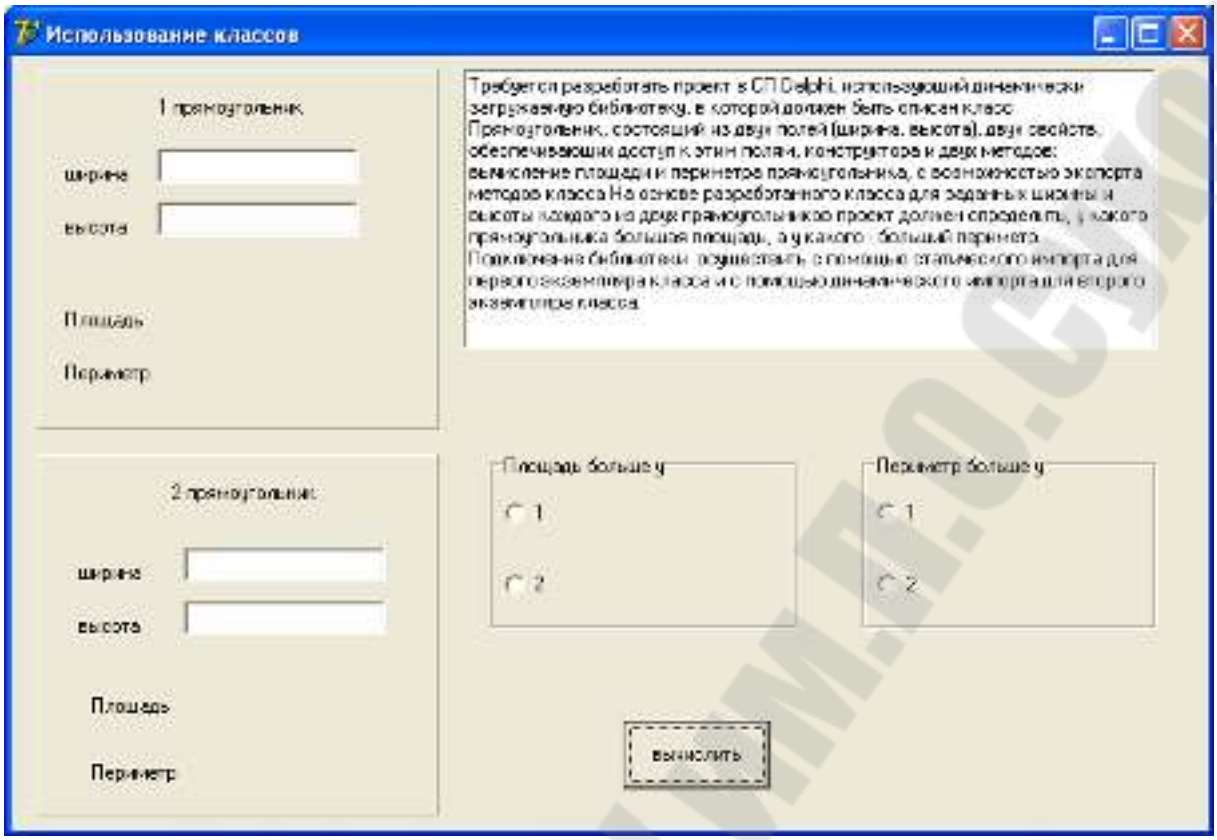


Рисунок 2 – **Интерфейс проекта использования динамической библиотеки**

4. ***Использование динамической библотеки***. Использование библиотеки происходит в процедуре обработки щелчка по кнопке *Вычислить*. Ниже приведен код обработчика события. Обратите внимание на то, как загружается библиотека и как происходит доступ к внешним функциям.

```
procedure TForm1.Button1Click(Sender: TObject);
var
//Индентификатор загружаемой библиотеки
_Mod:Integer;
//Описываем указатели на функции, которые будем получать
pl,per:function:real;
CreateObj,ReleaseObj:procedure;
setarg:procedure(a,b:real);
//Описываем дополнительные переменные
x,y:real;
S1,S2,P1,P2:real;
```

```
begin
//Загружаем библиотеку в память
_Mod:=LoadLibrary('Project2_dll.dll');
 //Получаем адреса загруженных функций
 @CreateObj:=GetProcAddress(_Mod,'createobject');
 @ReleaseObj:=GetProcAddress(_Mod,'releaseobject');
 @pl:=GetProcAddress(_Mod,'pl');
 @per:=GetProcAddress(_Mod,'per');
 @setarg:=GetProcAddress(_Mod,'setarg');

//Создаем экземпляр класса, вызывая функцию по адресу
CreateObj;
x:=strtofloat(edit1.Text);
y:=strtofloat(edit2.Text);
//Устанавливаем значения полей согласно введенным данным
setarg(x,y);
//Запоминаем значения площади и периметра для введенных данных
S1:=pl;
P1:=per;
x:=strtofloat(edit3.Text);
y:=strtofloat(edit4.Text);
//Устанавливаем и запоминаем значения для новых данных
setarg(x,y);
S2:=pl;
P2:=per;
//Уничтожаем объект
ReleaseObj;
//Выгружаем библиотеку из памяти
FreeLibrary(_Mod);

 if S1>S2 then Radiogroup1.ItemIndex:=0 else Radiogroup1.ItemIndex:=1;
 if P1>P2 then Radiogroup2.ItemIndex:=0 else Radiogroup2.ItemIndex:=1;
 label5.Caption:=label5.Caption+'  '+ floattostr(S1);
 label6.Caption:=label6.Caption+'  '+ floattostr(P1);
 label9.Caption:=label9.Caption+'  '+ floattostr(S2);
```

```
 label10.Caption:=label10.Caption+'  '+ floattostr(P2);
end;
```

Контрольные вопросы

1. Что такое динамически загружаемая библиотека?
2. Какова структура библиотеки?
3. Что описывается в секции export?
4. Для чего используются директивы register, stdcall, pascal и cdecl и в чем их отличие?
5. Как создать динамическую библиотеку в Delphi?
6. Что такое импорт библиотеки?
7. Как осуществляется статический импорт?
8. Как осуществляется динамический импорт?
9. В чем разница между статическим и динамическим импортом?
10. Как получить адрес функции при динамическом импорте?
11. Как осуществляется выгрузка библиотеки из оперативной памяти?

Лабораторная работа № 3
тема: «Использование интерфейсов»

Цель работы: получение навыков описания и реализации интерфейсов.

Краткие теоретические сведения:

Понятие интерфейса

При программировании нередко возникает необходимость выполнить обращение к объекту, находящемуся в другом загрузочном модуле, например EXE или DLL. Для решения поставленной задачи компания Microsoft разработала технологию COM (Component Object Model) — компонентную модель объектов. Технология получила такое название благодаря тому, что обеспечивает создание программных компонентов — независимо разрабатываемых и поставляемых двоичных модулей. Поскольку объекты различных программ разрабатываются на различных языках программирования, например Delphi, C++, Visual Basic и др., технология COM стандартизирует формат взаимодействия между объектами на уровне двоичного представления в оперативной памяти. Согласно технологии COM взаимодействие между объектами осуществляется посредством так называемых интерфейсов.

Интерфейс - это список функций. С точки зрения двоичного кода интерфейс представляет собой таблицу, содержащую указатели на функции. Количество элементов в этой таблице соответствует количеству функций в интерфейсе.

Каждый интерфейс имеет имя. Имя интерфейса должно быть идентификатором, корректным с точки зрения языка, на котором разрабатывается клиент или сервер. Так как различные языки имеют разные требования к идентификаторам, в имени интерфейса рекомендуется использовать только английские буквы и символ подчёркивания "_". Названия интерфейсов принято начинать с буквы "I".

В языке Delphi интерфейсы описываются в секции type глобального блока. Описание начинается с ключевого слова **interface**

и заканчивается ключевым словом **end**. По форме объявления интерфейсы похожи на обычные классы, но в отличие от классов:

- интерфейсы не могут содержать поля;
- интерфейсы не могут содержать конструкторы и деструкторы;
- все атрибуты интерфейсов являются общедоступными (public);
- все методы интерфейсов являются абстрактными.

```
type

  ITest=interface
  ['{5629C5BE-359A-4996-AB37-B58667D4935B}']
  procedure setarg(a,b:real);
  function Plosh:real;
  function Per:real;
  end;
```

Интерфейс можно создать также путем расширения уже существующего интерфейса. В этом случае в описании интерфейса после слова **interface** указывается имя базового интерфейса.

В языке Delphi существует предопределенный интерфейс **IInterface**, который служит неявным базовым интерфейсом для всех остальных интерфейсов.

```
type

  ITest = interface
    ...
  end;
```

эквивалентно следующему:

```
type
  ITest = interface(IInterface)
    ...
  end;
```

Рекомендуется использовать вторую, более полную форму записи.

Описание интерфейса IInterface находится в стандартном модуле System:

```
type
  IInterface = interface
    ['{00000000-0000-0000-C000-000000000046}']
    function QueryInterface(const IID: TGUID; out Obj): HResult;
stdcall;
    function _AddRef: Integer; stdcall;
    function _Release: Integer; stdcall;
  end;
```

Методы интерфейса IInterface явно или неявно попадают во все интерфейсы и имеют особое назначение. Метод **QueryInterface** нужен для того, чтобы, имея некоторый интерфейс, запросить у объекта другой интерфейс. Этот метод автоматически вызывается при преобразовании одних интерфейсов в другие. Метод **_AddRef** автоматически вызывается при присваивании значения интерфейсной переменной. Метод **_Release** автоматически вызывается при уничтожении интерфейсной переменной. Последние два метода позволяют организовать подсчет ссылок на объект и автоматическое уничтожение объекта, когда количество ссылок на него становится равным нулю. Вызовы всех трех методов генерируются компилятором автоматически, и вызывать их явно нет необходимости, однако программист должен позаботиться об их реализации.

Глобально-уникальный идентификатор интерфейса

Интерфейс является особым типом данных: он может быть реализован в одной программе, а использоваться из другой. Для этого нужно обеспечить идентификацию интерфейса при межпрограммном взаимодействии. Понятно, что программный идентификатор интерфейса для этого не подходит — разные программы пишутся разными людьми, а разные люди подчас дают одинаковые имена своим творениям. Поэтому каждому интерфейсу выдается своеобразный «паспорт» — **глобально-уникальный идентификатор (Globally Unique Identifier — GUID**).

Глобально-уникальный идентификатор — это 16-ти байтовое число, представленное в виде заключенной в фигурные скобки последовательности шестнадцатеричных цифр:

{DC601962-28E5-4BF7-9583-0CE22B605045}

Если глобально-уникальный идентификатор назначается интерфейсу, то он записывается после ключевого слова interface и заключается в квадратные скобки (смотри описание интерфейса IInterface).

Генерация глобально-уникальных идентификаторов осуществляется системой Windows по специальному алгоритму, в котором задействуется адрес сетевого адаптера, текущее время и генератор случайных чисел. Для генерации GUID в среде Delphi необходимо нажмите в редакторе кода комбинацию клавиш **Ctrl+Shift+G.**

Реализация интерфейса

Реализацией интерфейса занимается класс. Если класс реализует интерфейс, то интерфейс может использоваться для доступа к объектам этого класса. При объявлении класса имя реализуемого интерфейса записывается через запятую после имени базового класса:

```
type
  TTest = class(TObject, ITest)
    ...
  end;
```

Такая запись означает, что класс TTest унаследован от класса TObject и реализует интерфейс ITest.

Класс, реализующий интерфейс, должен содержать код для всех методов интерфейса.

Методы **QueryInterface**, **_AddRef** и **_Release** тоже должны быть реализованы. Разработчики системы Delphi реализовали методы интерфейса **IInterface** в классе **TInterfacedObject**.

Использование интерфейса

Для доступа к объекту через интерфейс нужна интерфейсная переменная:

```
var
  Intf: ITextReader;
```

Интерфейсная переменная занимает в оперативной памяти четыре байта, хранит ссылку на интерфейс объекта и автоматически инициализируется значением **nil**.

Перед использованием интерфейсную переменную инициализируют значением объектной переменной:

```
var
  Obj: TTest;  // объектная переменная
  Intf: ITest; // интерфейсная переменная
begin
  ...
  Intf := Obj;
  ...
end;
```

После инициализации интерфейсную переменную Intf можно использовать для вызова методов объекта Obj. Через интерфейсную переменную доступны только те методы и свойства объекта, которые есть в интерфейсе.

Один класс может содержать реализацию нескольких интерфейсов. Такая возможность позволяет воплотить в классе несколько понятий.

Представление интерфейса в памяти

Интерфейс по сути выступает дополнительной таблицей виртуальных методов, ссылка на которую укладывается среди полей объекта (рисунок 1). Эта таблица называется таблицей методов интерфейса. В ней хранятся указатели на методы класса, реализующие методы интерфейса.

Интерфейсная переменная хранит ссылку на скрытое поле объекта, которое содержит указатель на таблицу методов интерфейса. Когда интерфейсной переменной присваивается значение объектой переменной,

Intf := Obj; // где Intf: ITest и Obj: TTest

к адресу объекта добавляется смещение до скрытого поля внутри объекта и этот результат заносится в интерфейсную переменную.

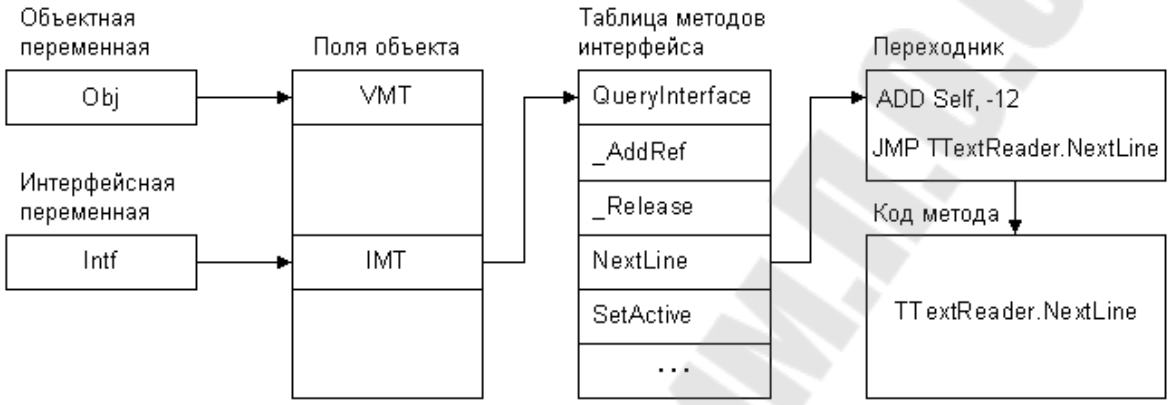


Рисунок 1 - Представление интерфейса в памяти

Алгоритм вызова метода интерфейса такой же, как алгоритм вызова метода класса. Когда через интерфейсную переменную выполняется вызов метода,

Intf.NextLine;

реализуется следующий алгоритм:

1. Из интерфейсной переменной извлекается адрес (по нему хранится адрес таблицы методов интерфейса);
2. По полученному адресу извлекается адрес таблицы методов интерфейса;
3. На основании порядкового номера метода в интерфейсе из таблицы извлекается адрес соответствующей подпрограммы;
4. Вызывается код, находящийся по этому адресу. Этот код является переходником от метода интерфейса к методу объекта. Его задача — восстановить из ссылки на интерфейс значение указателя Self (путем вычитания заранее

известного значения) и выполнить прямой переход на код метода класса.

Задание: Требуется разработать проекты в СП Delphi, использующие интерфейсы, для реализации задач в соответствии со своим вариантом. В первом проекте реализация интерфейсов, выполняется в главном модуле проекта, во втором проекте реализация интерфейсов, описывается классом, находящемся в DLL.

Вариант 1

С использованием интерфейсов вычислить длину окружности и площадь круга, ограниченного окружностью заданного радиуса, а также вычислить длину гипотенузы и площадь треугольника при заданных значениях длин двух катетов.

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.

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

где a и b – длины катетов;
c – длина гипотенузы;
S – площадь прямоугольного треугольника.

Вариант 2

С использованием интерфейсов вычислить длину гипотенузы и площадь треугольника при заданных значениях длин двух катетов, а также площадь поверхности и объем куба с заданной значением стороны.

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.

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

где a – длина стороны куба;
S – площадь поверхности куба;
V – объем куба.

Вариант 3

С использованием интерфейсов вычислить площадь поверхности и объем куба с заданным значением стороны, а также площадь треугольника и величину третьего угла при заданных значениях одной стороны и двух прилежащих к заданной стороне углов.

Формулы для расчета:

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

где a – длина стороны куба;
S – площадь поверхности куба;
V – объем куба.

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

где a – длина стороны треугольника;
α и β – углы, прилежащие к стороне a, град.;
γ – третий угол треугольника, град.;
S – площадь треугольника.

Вариант 4

С использованием интерфейсов вычислить площадь треугольника и величину третьего угла при заданных значениях одной стороны и двух прилежащих к заданной стороне углов, а также длину отрезка и осуществить проверку параллельности оси OX при заданных координатах концов отрезков.

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.

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

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

Условие параллельности отрезка оси OX:

$$y1 = y2.$$

Вариант 5

С использованием интерфейсов вычислить длину отрезка и осуществить проверку параллельности оси OX при заданных координатах концов отрезков, а также площадь треугольника и длину большей стороны при заданных координатах вершин треугольника.

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:

$$y1 = y2.$$

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

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 6

С использованием интерфейсов вычислить площадь треугольника и длину большей стороны при заданных координатах вершин треугольника, а также длину окружности и площадь круга, ограниченного окружностью заданного радиуса.

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.

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

где d – диаметр окружности;

C – длина окружности;

S – площадь круга.

Вариант 7

С использованием интерфейсов вычислить длину окружности и площадь круга, ограниченного окружностью заданного радиуса, а также площадь прямоугольника и длина диагонали при трех заданных координат вершин треугольника.

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.

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

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

a и b – длины сторон прямоугольника;

d – длина диагонали прямоугольника;

S – площадь прямоугольника.

Вариант 8

С использованием интерфейсов вычислить площадь прямоугольника и длина диагонали при трех заданных координат вершин треугольника, а также площадь и периметр параллелограмма при заданных значениях сторон и острого угла параллелограмма.

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 (*x*1, *y*1), (*x*2, *y*2) и (*x*3, *y*3) – координаты вершин прямоугольника;

a и *b* – длины сторон прямоугольника;

d – длина диагонали прямоугольника;

S – площадь прямоугольника.

$S = ab\sin\alpha$, P = 2(a+b)

где *a* и *b* – длины сторон параллелограмма;

α – острый угол параллелограмма;

P – периметр параллелограмма;

S – площадь параллелограмма.

Вариант 9

С использованием интерфейсов вычислить площадь и периметр параллелограмма при заданных значениях сторон и острого угла параллелограмма, а также площадь поверхности и объём шара заданного радиуса.

Формулы для расчета:

$S = ab\sin\alpha$, P = 2(a+b)

где *a* и *b* – длины сторон параллелограмма;

α – острый угол параллелограмма;

P – периметр параллелограмма;

S – площадь параллелограмма.

$$S = 4\pi R^2,\ V = \frac{4}{3}\pi R^3$$

где *R* – радиус шара;

Вариант 10

С использованием интерфейсов вычислить площадь поверхности и объём шара заданного радиуса, а также длину окружности и площадь круга, ограниченного окружностью заданного радиуса.

Формулы для расчета:

$$S = 4\pi R^2,\ V = \frac{4}{3}\pi R^3$$

где *R* – радиус шара;

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

где *r* – радиус окружности;

C – длина окружности;

S – площадь круга.

Пояснения к выполнению работы:

Задание: Требуется разработать проекты в СП Delphi, использующие интерфейсы, для вычисления площади и периметра прямоугольника, а также вычисления длины окружности и площади круга.

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

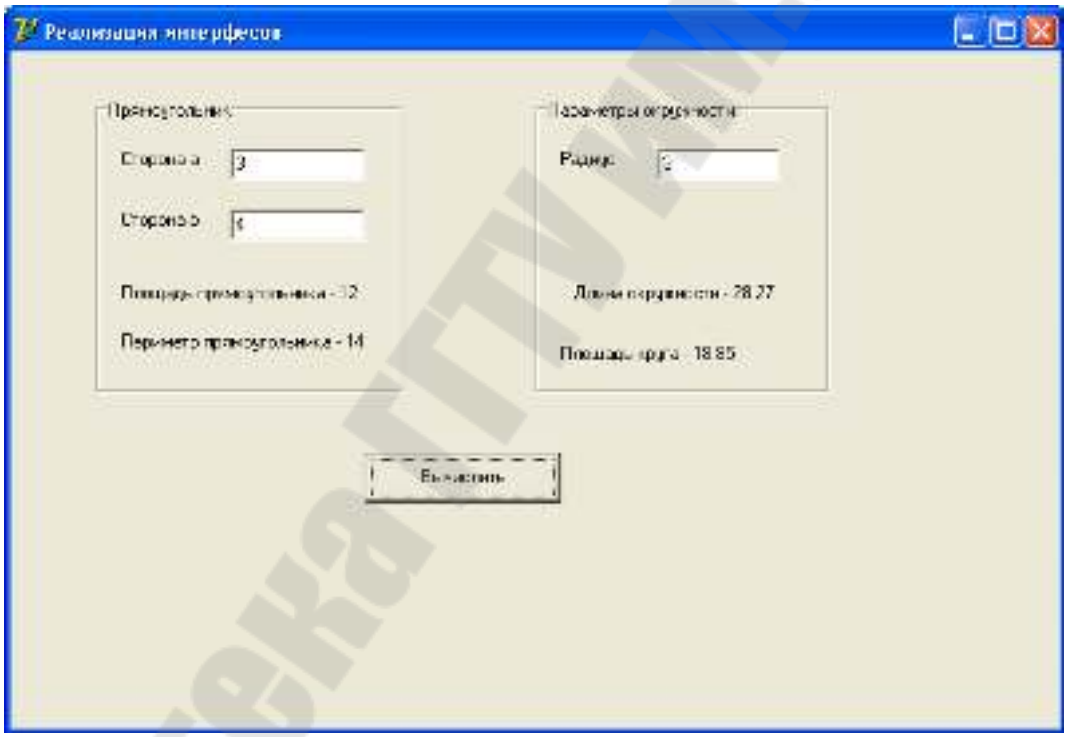


Рисунок 1 – **Интерфейс проекта реализации интерфейсов**

2. ***Описание интерфейсов***. В начале интерфейсной части добавим два интерфейса, использующие методы вычисления площади и периметра прямоугольника, а также методы вычисления длины окружности и площади круга:

```
type
  Itest1=interface
```

```
function Plosch(a,b:real):real;
function Perimetr(a,b:real):real;
end;

Itest2=interface
function Plosh(r:real):real;
function Dlina(r:real):real;
end;
```

3. ***Описание класса, реализующего методы описанных интерфесов***. В интерфейсной части добавим описание класса, реализующего методы двух интерфейсов.

```
TTest=class(TInterfacedObject,IInterface,Itest1,Itest2)
 function Plosch(a,b:real):real;
 function Perimetr(a,b:real):real;
 function Plosh(r:real):real;
 function Dlina(r:real):real;
 end;
```

Обратите внимание, что constructor и destructor наследуется от класса TInterfacedObject и не нуждаются в дополнительном описании и реализации.

4. ***Описание методов класса***. В начало исполняемой части проекта поместим описание четырех методов.

```
function TTest.Plosch(a,b:real):real;
    begin
    result:=a*b;
    end;
function TTest.Perimetr(a,b:real):real;
    begin
    result:=2*(a+b);
    end;
function TTest.Plosh(r:real):real;
    begin
    result:=pi*sqr(r);
```

```
    end;
function TTest.Dlina(r:real):real;
    begin
    result:=2*pi*r;
    end;
```

5. ***Использование интерфейсов***. Использование интерфейсов происходит в процедуре обработки щелчка по кнопке ***Вычислить***

```
procedure TForm1.Button1Click(Sender: TObject);
var a,b,r,S,P,Skr,Dl:real;
Pr:ITest1;Kr:ITest2;
begin
a:=strtofloat(edit1.text);
b:=strtofloat(edit2.text);
r:=strtofloat(edit3.text);
Kr:=TTest.Create;
Skr:=Kr.Plosh(r);
Dl:=Kr.Dlina(r);
Pr:=TTest.Create;
S:=Pr.Plosch(a,b);
P:=Pr.Perimetr(a,b);
Label5.Caption:='Площадь прямоугольника - '+FloatToStr(S);
Label6.Caption:='Периметр прямоугольника - '+FloatToStr(P);
Label7.Caption:='Площадь круга - '+FloatToStr(Skr);
Label8.Caption:='Длина окружности - '+FloatToStr(Dl);
end;
```

Контрольные вопросы

1. Что понимается под интерфейсным типом данных?
2. Как описывается интерфейс?
3. Что понимается под расширением интерфейса?
4. Что такое глобально-уникальный идентификатор интерфейса?
5. Как генерируется глобально-уникальный идентификатор интерфейса в Delphi?
6. Как реализуется интерфейс?
7. Как осуществляется доступ к объекту через интерфейс?

8. Как осуществляется реализация нескольких интерфейсов?
9. Какой интерфейс является базовым в Delphi?
10. Какие методы интерфейса IInterface обеспечивают подсчет ссылок на объект?
11. Для чего предназначен метод QueryInterface интерфейса IInterface?
12. Как представляется интерфейса в памяти?

Лабораторная работа № 4
тема: «Создание внутреннего com-сервера без библиотеки типов»

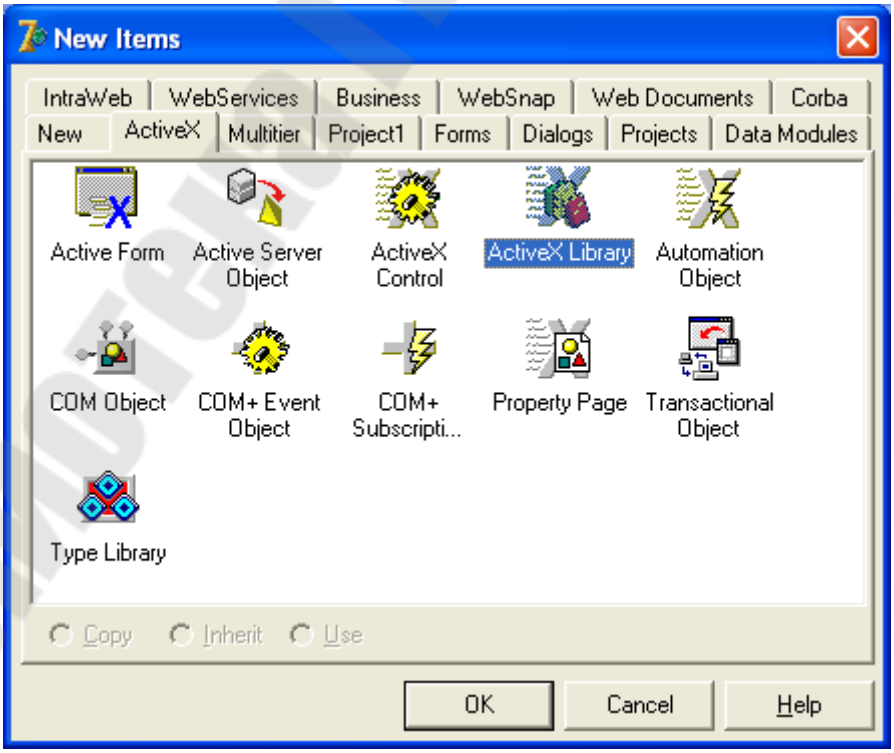
Цель работы: получение навыков создания и регистрации внутренних Com-серверов без использования библиотеки типов, Com-клиентов, использующие возможности Com-серверов.

Краткие теоретические сведения:

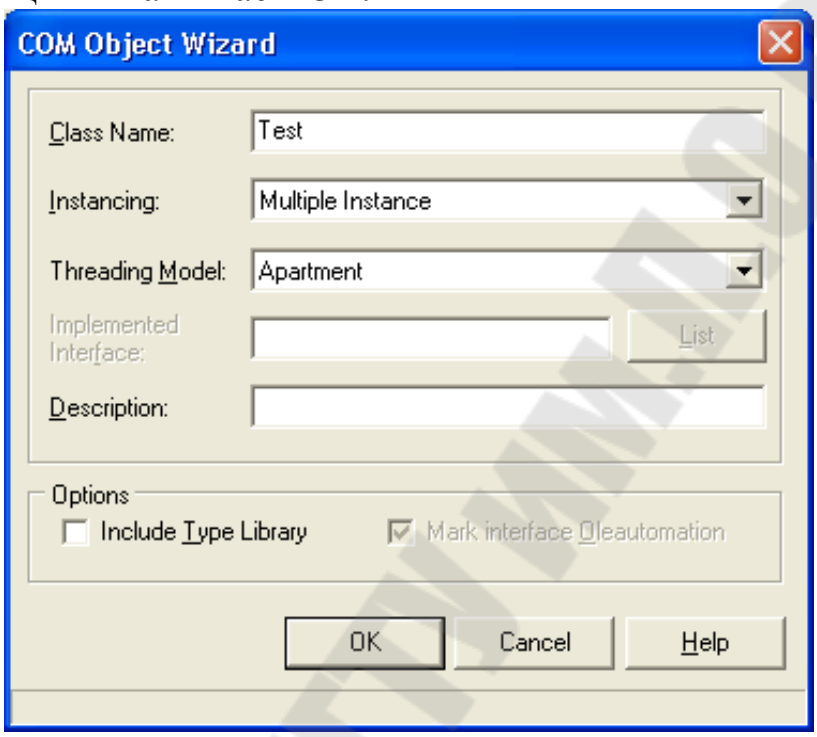
Рассмотрим процедуру создания Com-сервера и Com-клиента в Delphi на примере конкретного задания.

Задание: Требуется разработать Com-сервер, реализующий перевод целого числа в двоичный, восьмеричный и шестнадцатеричный вид. Работоспособность сервера проверить подключением Com-клиента.

1. ***Создание Com-сервера***. Для создания встроенного сервера (реализуется в виде DLL) выбираем в меню File/New. В появившемся диалоговом окне New Item открываем закладку ActiveX и выбираем элемент ActiveX Library.



2. ***Создание Com-объекта***. Для создания COM-объекта выбираем в меню File/New. В появившемся диалоговом окне New Item открываем закладку ActiveX и выбираем элемент COM Object. В появившемся диалоговом окне вводим название КоКласса, снимаем флажки с опций и нажимаем ОК.



Дописываем описание методов, которые будут использоваться для перевода заданного числа, в интерфейсную часть и их реализацию в разделе implementation.

```
type
  TTest = class(TComObject,IConvert)
  protected
  function Bin(const N:Word; var S:WideString):Boolean; stdcall;
  function Oct(const N:Word; var S:WideString):Boolean; stdcall;
  function Hex(const N:Word; var S:WideString):Boolean; stdcall;
  end;

implementation

function TTest.Bin(const N:Word; var S:WideString):Boolean;
var S1:string;
    m:Word;
```

```
begin
result:=true;
try
S1:='';
m:=n;
repeat
S1:=chr((m mod 2)+48)+S1;
m:=m div 2;
until m=0;
S:=WideString(S1);
except
result:=false;
end;
end;

function TTest.Oct(const N:Word; out S:WideString):Boolean;
var S1:string;
   m:Word;
begin
result:=true;
try
S1:='';
m:=n;
repeat
S1:=chr((m mod 8)+48)+S1;
m:=m div 8;
until m=0;
S:=WideString(S1);
except
result:=false;
end;
end;

function TTest.Hex(const N:Word; out S:WideString):Boolean;
var S1:string;
   m:Word;
begin
result:=true;
try
```

```
m:=n;
S1:='';
while m>15 do
begin
i:=m mod 16;
m:=m div 16;
if i<10 then s1:=chr(i+48)+s
else S1:=chr(i+55)+s;
end;
if m<10 then S1:=chr(m+48)+s
else S1:=chr(m+55)+s;
S:=WideString(S1);
except
result:=false;
end;
end;
```

3. ***Создание Интерфейса***. Для создания интерфейса добавим в проект сервера новый модуль и сохраним под именем server_interface.pas. В тексте модуля описываем интерфейс

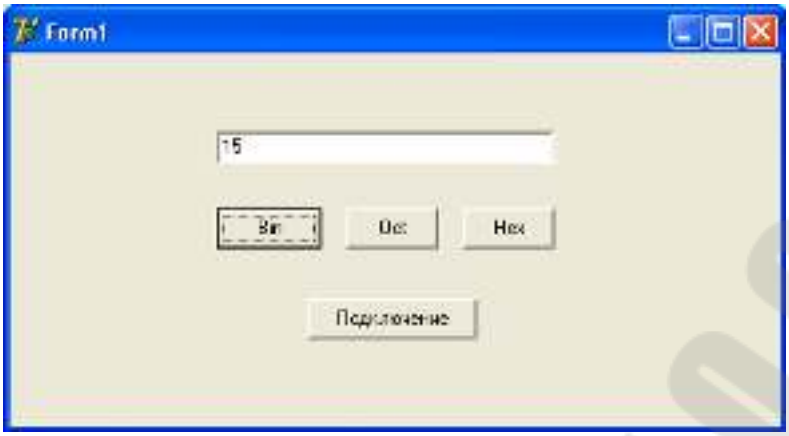
```
IConvert=Interface
  ['{738810F5-FE8D-49C0-B44B-4EA81B5AC167}']
  function Bin(const N:Word; var S:WideString):Boolean; stdcall;
  function Oct(const N:Word; var S:WideString):Boolean; stdcall;
  function Hex(const N:Word; var S:WideString):Boolean; stdcall;
  end;
```

В разделе подключения модулей Com-объекта дописываем модули

```
uses
  Windows, ActiveX, Classes, ComObj, server_interface;
```

4. ***Регистрация Com-сервера***. Выбираем команду Project – Build Project2. Для регистрации сервера запускаем команду RegSvr32.exe Project2.dll.

5. ***Создание Com-клиента***. Создаем новое приложение вида



В тексте модуля подключаем **ComObj, server_interface** и описываем реализацию Com-клиента

```
unit client_for_server;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  StdCtrls, ComObj, server_interface, dialogs;

Const
//Эта строка копируется из модуля КоКласса
  Class_Test: TGUID = '{37D69CC7-7971-4FD8-8FAC-
96E34C5E4A63}';
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    procedure Button4Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
```

```
    { Private declarations }

    test1:IConvert;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button4Click(Sender: TObject);
Var t1:TGUID;
begin
t1:= Class_Test;
showmessage(GUIDToString(t1));
test1 := CreateComObject(t1) as Iconvert;
  if test1= nil then
    ShowMessage('сервер не подключен')
 else ShowMessage('сервер подключен');
end;

procedure TForm1.Button1Click(Sender: TObject);
Var N:word;
Str:WideString;
begin
N:=StrToInt(Edit1.Text);
if test1.Bin(N,Str) then ShowMessage(Str);
end;

procedure TForm1.Button2Click(Sender: TObject);
Var N:word;
Str:WideString;
begin
N:=StrToInt(Edit1.Text);
if test1.Oct(N,Str) then ShowMessage(Str);
```

```
end;

procedure TForm1.Button3Click(Sender: TObject);
Var N:word;
Str:WideString;
begin
N:=StrToInt(Edit1.Text);
if test1.Hex(N,Str) then ShowMessage(Str);
end;

end.
```

Индивидуальные задания:

Разработать Com-сервер, в соответствии с вариантом. Разработать Com-клиент для подключения к Com-серверу.

Вариант 1

Разработать Com-сервер для вычисления длины окружности и площадь круга, ограниченного окружностью заданного радиуса.

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

Разработать Com-сервер для вычисления длины гипотенузы и площади треугольника при заданных значениях длин двух катетов.

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

Разработать Com-сервер для вычисления площади поверхности и объема куба с заданным значением стороны.

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

Разработать Com-сервер для вычисления площади треугольника и величины третьего угла при заданных значениях одной стороны и двух прилежащих к заданной стороне углов.

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

Разработать Com-сервер для вычисления длины отрезка и осуществления проверки параллельности оси OX при заданных координатах концов отрезков.

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

Разработать Com-сервер для вычисления площади треугольника и длины большей стороны при заданных координатах вершин треугольника.

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

Разработать Com-сервер для вычисления длины окружности и площади круга, ограниченного окружностью заданного радиуса.

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

Разработать Com-сервер для вычисления площади прямоугольника и длины диагонали при трех заданных координат вершин треугольника.

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.

Вариант 9

Разработать Com-сервер для вычисления площади и периметра параллелограмма при заданных значениях сторон и острого угла параллелограмма.

Формулы для расчета:

$S = ab\sin\alpha$, P = 2(a+b)

где a и b – длины сторон параллелограмма;

α – острый угол параллелограмма;

P – периметр параллелограмма;

S – площадь параллелограмма.

Вариант 10

Разработать Com-сервер для вычисления площади поверхности и объёма шара заданного радиуса.

Формулы для расчета:

$$S = 4\pi R^2,\ V = \frac{4}{3}\pi R^3$$

где R – радиус шара;

Контрольные вопросы

1. Как создать внутренний Com-сервер?
2. Как описывается CoClass?
3. Как регистрируется Com-сервер в реестре Windows?
4. Как подключиться к зарегистрированному Com-серверу?
5. Как создается Com-объект?
6. Какие модули необходимы для работы с Com-объектами?

Лабораторная работа № 5
тема: «Создание Com-сервера с использованием библиотеки типов»

Цель работы: получение навыков создания и регистрации внутренних Com-серверов с использованием библиотеки типов, Com-клиентов, использующие возможности Com-серверов.

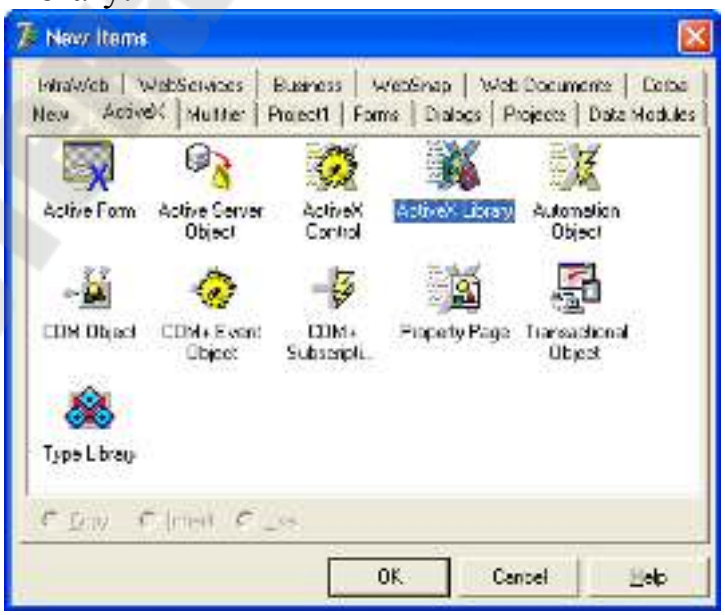
Краткие теоретические сведения:

Создание com-сервера с использованием библиотеки типов

Рассмотрим процедуру создания Com-сервера с использованием библиотеки типов и Com-клиента в Delphi на примере конкретного задания.

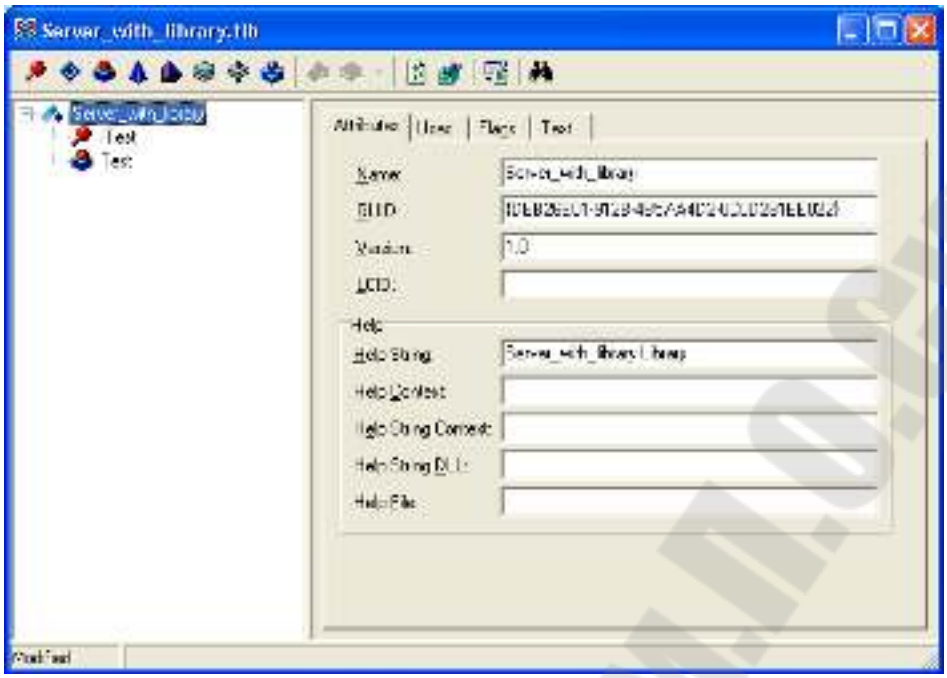
Задание: Требуется разработать Com-сервер, реализующий перевод целого числа в двоичный, восьмеричный и шестнадцатеричный вид. Работоспособность сервера проверить подключением Com-клиента.

1. ***Создание Com-сервера***. Для создания встроенного сервера (реализуется в виде DLL) выбираем в меню File/New. В появившемся диалоговом окне New Item открываем закладку ActiveX и выбираем элемент ActiveX Library.

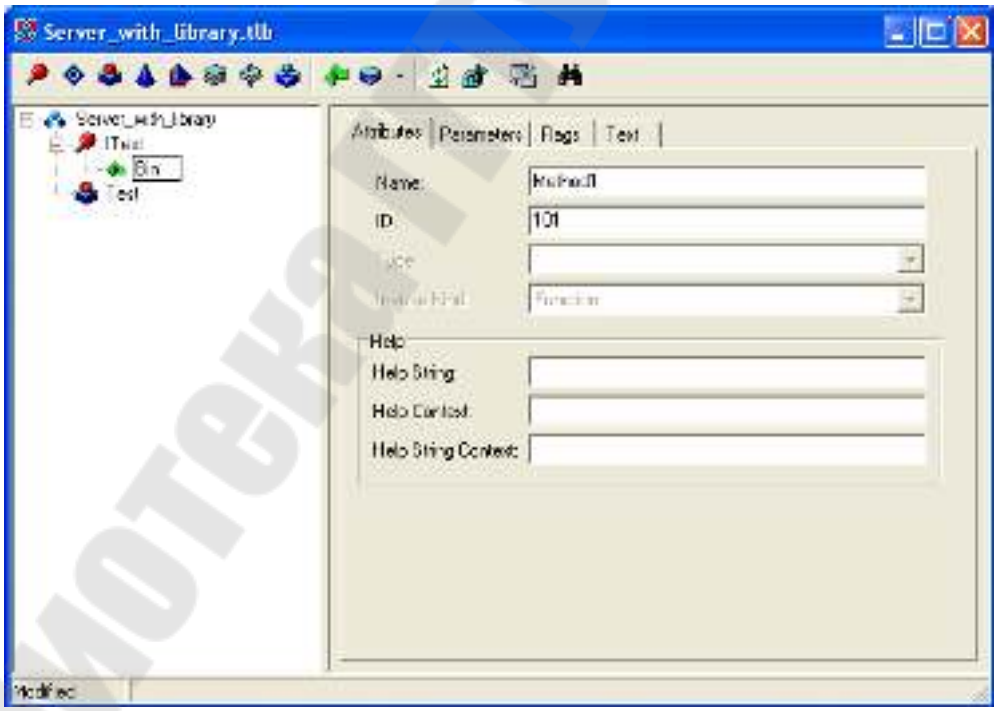


2. ***Создание Com-объекта***. Для создания COM-объекта выбираем в меню File/New. В появившемся диалоговом окне New Item открываем закладку ActiveX и выбираем элемент COM Object. В появившемся диалоговом окне вводим название КоКласса, в опциях устанавливаем флаг Include Type Library, флаг Mark interface Oleautomation снимаем.

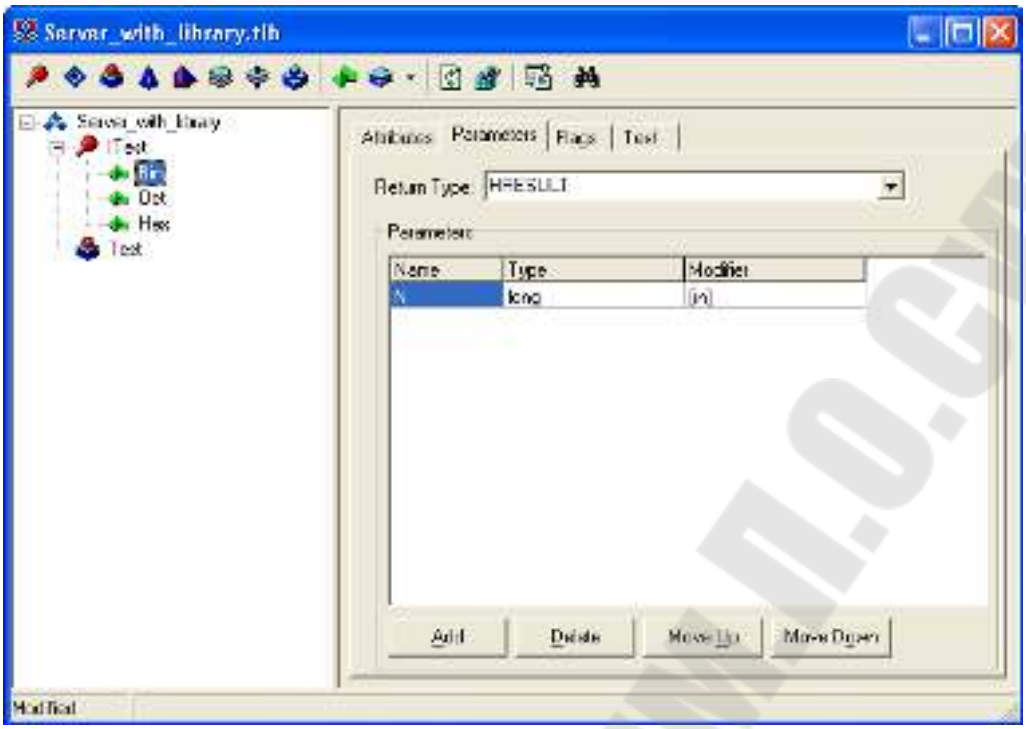
3. ***Создание Интерфейса***. После нажатия ОК в предыдущем пункте открывается редактор библиотеки типов



Выбираем интерфейс ITest и с помощью контекстного меню (или кнопки на панели инструментов) выбираем команду New Method.



На вкладке Parameters определяем входные параметры соответствующего метода.



Создаем методы Oct и Hex аналогичным образом. После этого нажимаем кнопку Refresh (Обновить) для записи методов в код модуля. Обратите внимание, что к модулю Com-объекта уже добавлены модули **ComObj, Server_with_library_TLB**. Последний является описанием библиотеки типов.

4. ***Реализация методов интерфейса***. В разделе implementation модуля Com-объекта необходимо описать реализацию методов

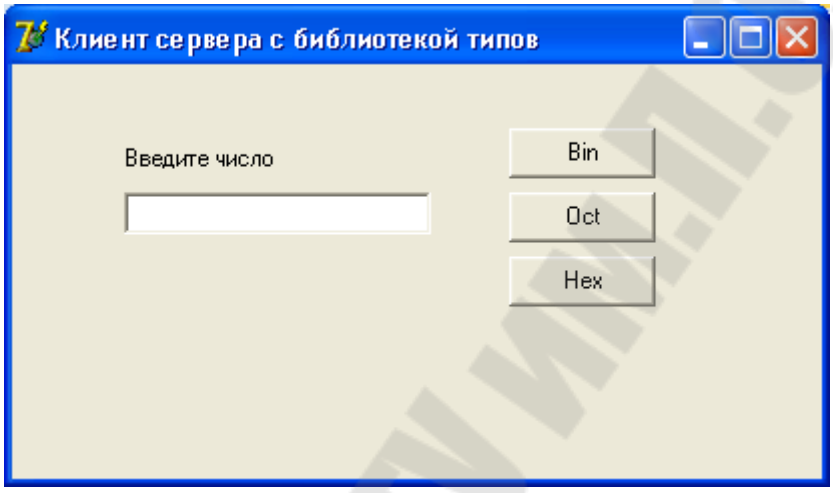
```
function TTest.Bin(N: Integer): HResult;
var S:String;
   m:Word;
begin
S:='';
m:=N;
repeat
S:=chr((m mod 2)+48)+S;
m:=m div 2;
until m=0;
showmessage(widestring(s));
Result := S_OK;
```

```
end;

function TTest.Hex(N: Integer): HResult;
var S:string;
    m,k:Word;
begin
S:='';
m:=N;
repeat
S:=chr((m mod 16)+48)+S;
k:=m div 16;
if k<10 then S:=chr(k+48)+S
        else S:=chr(k+57)+S;
m:=m div 16;
until m=0;
showmessage(widestring(s));
Result := S_OK;
end;

function TTest.Oct(N: Integer): HResult;
var S1:string;
   m, i:Word;
begin
result:=true;
try
m:=n;
S1:='';
while m>15 do
begin
i:=m mod 16;
m:=m div 16;
if i<10 then s1:=chr(i+48)+s
else S1:=chr(i+55)+s;
end;
if m<10 then S1:=chr(m+48)+s
else S1:=chr(m+55)+s;
S:=WideString(S1);
except
result:=false;
```

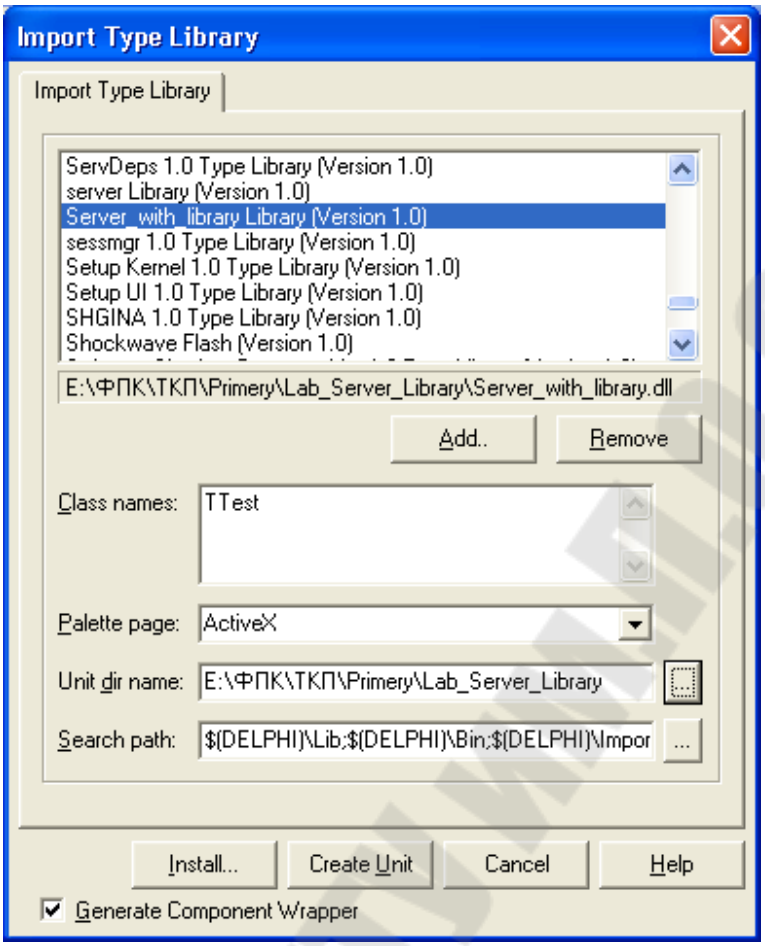
end;

5. ***Регистрация Com-сервера***. Запускаем процесс компиляции или выбираем команду Project – Build server_with_library. Для регистрации сервера в редакторе библиотеки типов выбираем команду Register Type Library.

6. ***Создание Com-клиента***. Создаем новое приложение вида



Выбираем команду Project – Import Type Library.



В диалоговом окне выбираем соответствующую библиотеку. Прописываем путь для записи файла библиотеки типов с модулем Com-объекта. Снимаем флаг с Generate Component Wrapper. Нажимаем кнопку Create Unit. В окне Com-клиента в разделе **uses** подключаем библиотеку типов и описываем последовательность действий клиента.

```
procedure TForm1.Button1Click(Sender: TObject);
Var t1:TGUID;
N:integer;
begin
t1:= Class_Test;
test2 := CreateComObject(t1) as ITest;
  if test2= nil then
    ShowMessage('сервер не подключен')
 else ShowMessage('сервер подключен');
N:=strtoInt(Edit1.Text);
test2.Bin(N);
```

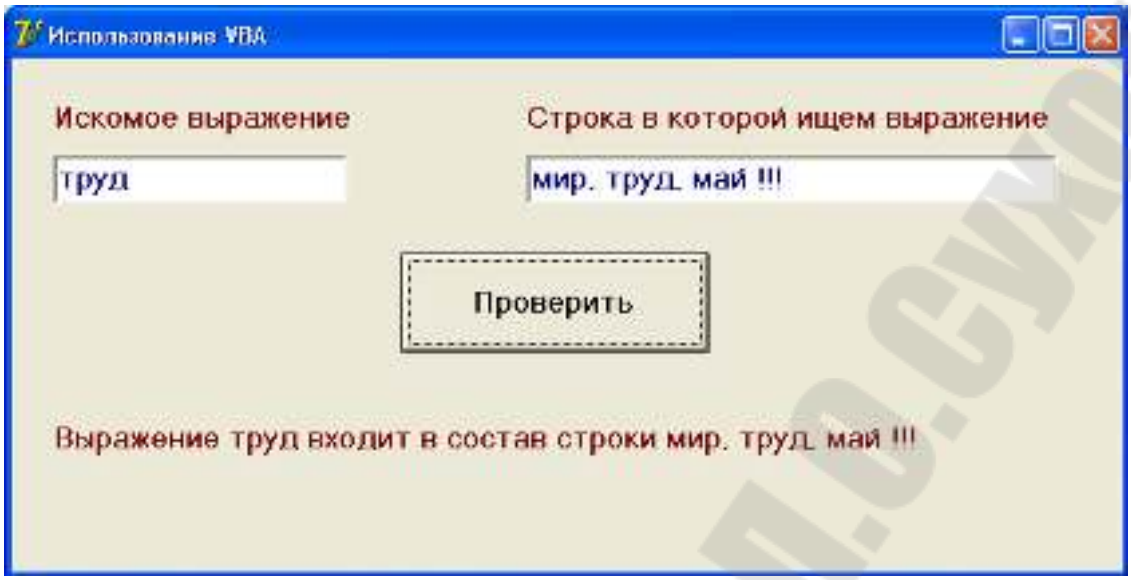
```
end;

procedure TForm1.Button2Click(Sender: TObject);
Var t1:TGUID;
N:integer;
begin
t1:= Class_Test;
test2 := CreateComObject(t1) as ITest;
  if test2= nil then
    ShowMessage('сервер не подключен')
 else ShowMessage('сервер подключен');
N:=strtoInt(Edit1.Text);
test2.Oct(N);

end;

procedure TForm1.Button3Click(Sender: TObject);
Var t1:TGUID;
N:integer;
begin
t1:= Class_Test;
test2 := CreateComObject(t1) as ITest;
  if test2= nil then
    ShowMessage('сервер не подключен')
 else ShowMessage('сервер не подключен");
N:=strtoInt(Edit1.Text);
test2.Hex(N);
end;
```

Подключение к зарегистрированной библиотеке типов

Задание: Необходимо создать приложение проверяющее вхождение подстроки1 в строку2. Приложение в своей работе должно использовать процессор регулярных выражений VBScript.

Порядок выполнения

1. Создадим новое приложение вида



2. Сохраним проект.
3. Импортируем библиотеку типов Microsoft VBScript Regular Expressions, выбрав команду Project - Import Type Library в главном меню среды разработки.
4. Указываем путь для записи фала описания библиотеки, снимаем флажок Generate Component Wrapper и нажимаем Create Union. В указанном месте будет создан файл VBScript_RegExp_TLB.pas, в котором описаны все методы.
5. Добавим следующий код обработки нажатия кнопки

```
implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var
RE: IRegExp;
s1,s2:string;
begin
s1:=edit1.Text;
s2:=edit2.Text;
RE := CoRegExp.Create;
RE.Pattern := s1;
if RE.Test(s2) then
label3.Caption := 'Выражение '+s1+' входит в состав строки '+s2
else
label3.Caption := 'Выражение '+s1+' не входит в состав строки '+s2;
end;


end.
```

6. Запустим приложение на выполнение.

Индивидуальные задания:

Разработать Com-сервер с использованием библиотеки типов, в соответствии с вариантом. Разработать Com-клиент для подключения к Com-серверу.

Вариант 1

Разработать Com-сервер для вычисления длины окружности и площадь круга, ограниченного окружностью заданного радиуса.

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

Разработать Com-сервер для вычисления длины гипотенузы и площади треугольника при заданных значениях длин двух катетов.

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

Разработать Com-сервер для вычисления площади поверхности и объема куба с заданным значением стороны.

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

Разработать Com-сервер для вычисления площади треугольника и величины третьего угла при заданных значениях одной стороны и двух прилежащих к заданной стороне углов.

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

Разработать Com-сервер для вычисления длины отрезка и осуществления проверки параллельности оси OX при заданных координатах концов отрезков.

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

Разработать Com-сервер для вычисления площади треугольника и длины большей стороны при заданных координатах вершин треугольника.

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

Разработать Com-сервер для вычисления длины окружности и площади круга, ограниченного окружностью заданного радиуса.

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

Разработать Com-сервер для вычисления площади прямоугольника и длины диагонали при трех заданных координат вершин треугольника.

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.

Вариант 9

Разработать Com-сервер для вычисления площади и периметра параллелограмма при заданных значениях сторон и острого угла параллелограмма.

Формулы для расчета:

$S = ab\sin\alpha$, P = 2(a+b)

где a и b – длины сторон параллелограмма;

α – острый угол параллелограмма;

P – периметр параллелограмма;

S – площадь параллелограмма.

Вариант 10

Разработать Com-сервер для вычисления площади поверхности и объёма шара заданного радиуса.

Формулы для расчета:

$$S = 4\pi R^2,\ V = \frac{4}{3}\pi R^3$$

где R – радиус шара;

Контрольные вопросы

1. Что такое библиотека типов?
2. Как создаются методы Com-объекта?
3. Где определяются параметры методов?
4. Как описать входные параметры?
5. Как описать выходные параметры?
6. Как регистрируется Com-сервер с библиотекой типов?
7. Как подключить библиотеку типов Com-сервера?
8. Для чего используется кнопка Refresh диалогового окна библиотеки типов?

Лабораторная работа № 6
тема: «Создание сервера автоматизации»

Цель работы: получение навыков создания серверов и контроллеров автоматизации.

Краткие теоретические сведения:

Основные понятия

Автоматизация (automation или OLE-automation) - технология, позволяющая приложениям и библиотекам (DLL) предоставлять свои программируемые объекты с целью их использования в других приложениях.

Сервера автоматизации (automation servers) - приложения и библиотеки (DLL), предоставляющие свои программируемые объекты.

Контроллеры автоматизации (automation controllers) - приложения, получающие доступ к управлению программируемыми объектами, предоставляемыми серверами автоматизации.

Контроллеры автоматизации (**КА**) способны "программировать" сервер автоматизации (**СА**) с помощью некоторого макроязыка, предлагаемого СА.

Объекты автоматизации (ОА) - это обычные COM-объекты, в которых помимо интерфейса **IUnknown** реализован интерфейс **IDispatch**. Интерфейс **IDispatch** определён в модуле System следующим образом:

```
type
  IDispatch = interface(IUnknown)
    ['{00020400-0000-0000-C000-000000000046}']
    function GetTypeInfoCount(out Count: Integer): HResult; stdcall;
    function GetTypeInfo(Index, LocaleID: Integer; out TypeInfo):
     HResult; stdcall;
    function GetIDsOfNames(const IID: TGUID; Names: Pointer;
      NameCount, LocaleID: Integer; DispIDs: Pointer): HResult; stdcall;
    function Invoke(DispID: Integer; const IID: TGUID; LocaleID:
     Integer;
```

Flags: Word; var Params; VarResult, ExcepInfo, ArgErr: Pointer):
HResult; stdcall;
end;

При разработке серверов и контроллеров автоматизации средствами Delphi работа программиста максимально упрощается, так как Delphi инкапсулирует автоматизацию.

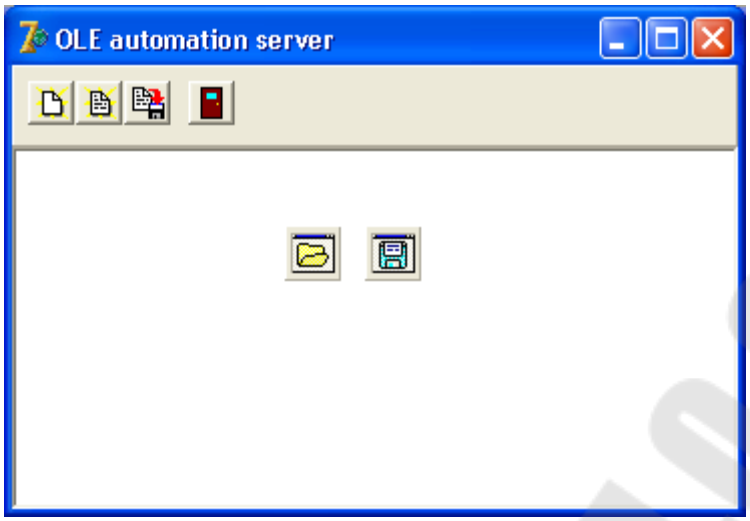
Внешний сервер автоматизации (LocalServer) - внешние СА являются выполняемыми файлами, которые могут создавать объекты автоматизации для использования их другими приложениями. Из названия понятно, что внешние СА выполняются в контексте своего собственного процесса. Как и все COM-объекты СА должны быть зарегистрированы, т.е. должны создавать такие же записи в реестре, как и все остальные COM-объекты, плюс два дополнительных параметра. В Delphi регистрация происходит автоматически при первом запуске СА при выполнении Application.Initialize().

Создание простейшего внешнего сервера автоматизации

Пусть требуется разработать сервер автоматизации, на основе приложения (текстовый редактор), которое содержит панель инструментов с четырьмя кнопками и компонент TMemo, а также диалоговые окна открытия и сохранения файла.

Порядок выполнения работы

7. ***Создание Текстового редактора***. Создадим обычное приложение вида



В тексте модуля напишем коды обработчиков событий, связанные со щелчками на кнопках

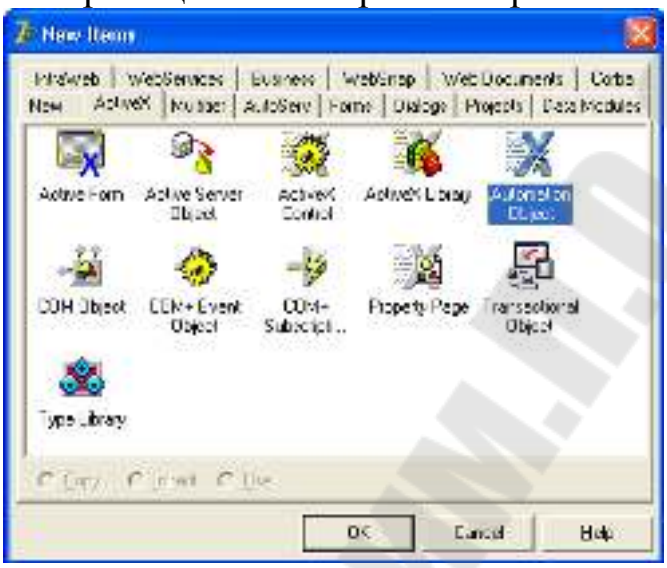
```
unit UServ1;

interface

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

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Memo1: TMemo;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    SpeedButton3: TSpeedButton;
    SpeedButton4: TSpeedButton;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    procedure SpeedButton1Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
    procedure SpeedButton3Click(Sender: TObject);
private
    { Private declarations }
```

```
public
    { Public declarations }
end;

var
  Form1: TForm1;

implementation
{$R *.dfm}

procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  Memo1.Lines.Clear;
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
  if OpenDialog1.Execute then
Memo1.Lines.LoadFromFile(OpenDialog1.FileName);
end;

procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
  if SaveDialog1.Execute then
Memo1.Lines.SaveToFile(OpenDialog1.FileName);
end;

procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
  Close;
end;

end.
```

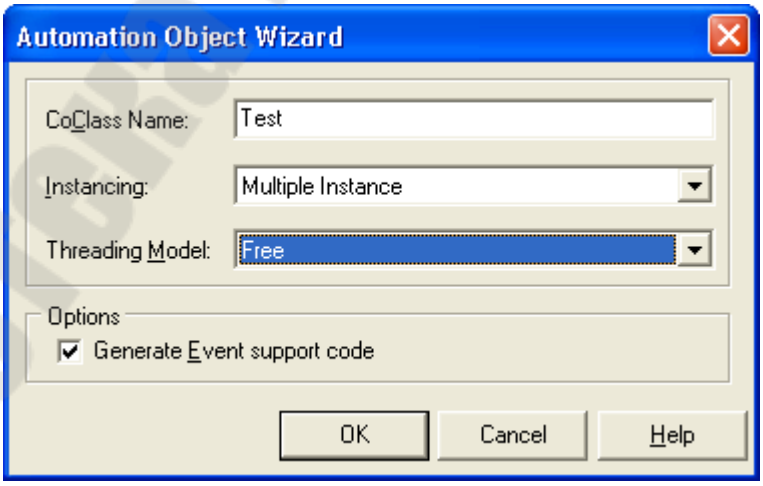
Сохраним проект под именем AutoServer

8. ***Создание Сервера автоматизации***. Отметим, что созданный нами текстовый редактор представляет собой обычное Windows-приложение и не является сервером автоматизации.

Для создания ***Сервера автоматизации*** добавим объект Automation Object на странице ActiveX репозитария объектов

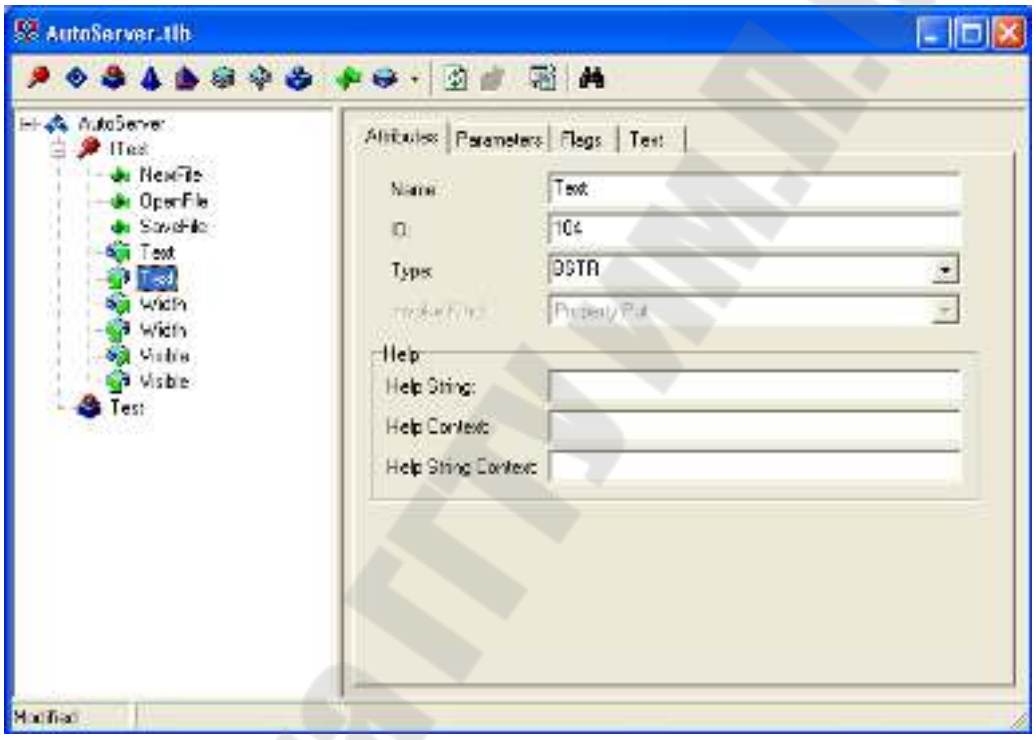


В открывшемся диалоговом окне введем имя, под которым данный класс COM-объектов будет зарегистрирован в реестре. В раскрывающемся списке Threading Model выберем пункт Free, означающий выбор модели свободных потоков (free-threaded model). В раскрывающемся списке Instancing выбираем Multiple Instance.



В редакторе библиотеки типов определяем свойства и методы созданного класса COM-объектов. Для нашего сервера опишем методы FileNew, FileOpen, FileSave, AddLine и их параметры, а также

свойства Text, Width и Visible. Метод NewFile параметров не имеет. Методы OpenFi1e и SaveF1e имеют один строковый параметр типа BSTR (WideString) - имя файла. Метод AddLine также имеет один строковый параметр, задающий добавляемую строку. Свойство Text доступно для чтения и записи и имеет тип BSTR(WideString). Свойство Visible имеет логический тип VARIANT_BOOL(WordBool) и тоже доступно для чтения и записи. Свойство Width имеет целый тип long (integer), определяет число пикселов и также доступно как для чтения, так и для записи.



9. ***Реализация методов сервера автоматизации***. Перейдем к реализации методов. Для этой цели в окне редактора библиотеки типов следует щелкнуть на кнопке Refresh панели инструментов. В модуль реализации (сгенерированный ранее мастером, использованным при создании сервера) будут добавлены. Добавим соответствующий код в заготовки методов.

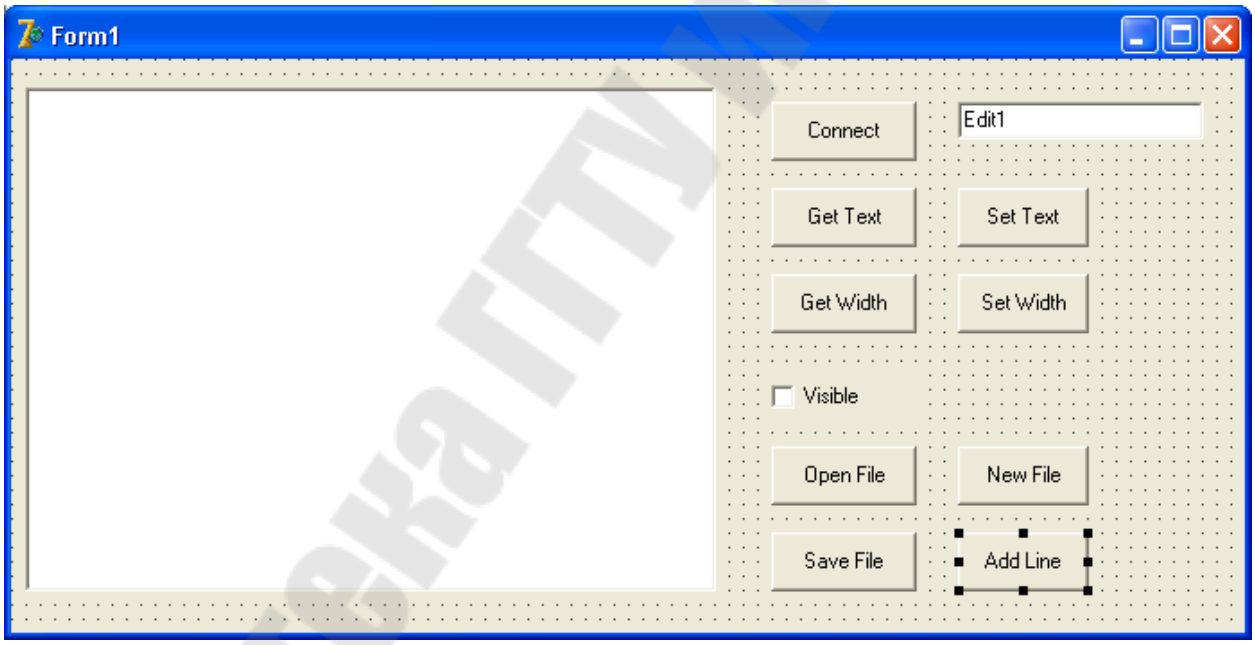
```
function TTest.Get_Text: WideString;
begin
Result := Form1.Memo1.Text;
end;
```

```
function TTest.Get_Visible: WordBool;
Result:= Form1.Visible;
end;

function TTest.Get_Width: Integer;
begin
Result := Form1.Width;
end;

procedure TTest.AddLine(const S: WideString);
begin
Form1.Memo1.Lines.Add(S);
end;

procedure TTest.NewFile;
begin
Form1.SpeedButton1Click(nil);
end;

procedure TTest.OpenFile(const S: WideString);
if Length(S)>0 then
Form1.Memo1.Lines.LoadFromFile(S)
else
Form1.SpeedButton2Click(nil);
end;

procedure TTest.SaveFile(const S: WideString);
begin
if Length(S) > 0 then
Form1.Memo1.Lines.SaveToFile(S)
else
Form1.SpeedButton3Click(nil);
end;

procedure TTest.Set_Text(const Value: WideString);
Form1.Memo1.Text := Value;
end;

procedure TTest.Set_Visible(Value: WordBool);
```

```
begin
Form1.Visible := Value;
end;

procedure TTest.Set_Width(Value: Integer);
begin
Form1.Width:=Value;
end;
```

10. ***Регистрация сервера автоматизации***. Регистрация сервера автоматизации происходит автоматически после запуска на выполнение.

11. ***Создание контроллера автоматизации***. Создаем новое приложение вида



В секции **uses** добавляем **ComObj** и **Variants**. В секции **private** класса **TForml** объявляем переменную **FServ** типа **Variant**. Создаем обработчики событий, связанные со щелчками на кнопках.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if vartype(FServ)=varDispatch then begin
    FServ:=Unassigned;
```

```
    Button1.Caption:='Connect';
  end else begin
    FServ:=CreateOLEObject('AutoServ.Test');
    Button1.Caption:='Disconnect';
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  if vartype(FServ)=varDispatch then
  Memo1.Text:=FServ.Text;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  if vartype(FServ)=varDispatch then
  FServ.Text:=Memo1.Text;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  if vartype(FServ)=varDispatch then
  Edit1.Text:=FServ.Width;
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
  if vartype(FServ)=varDispatch then
  FServ.Width:=StrToInt(Edit1.Text);
end;

procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  if vartype(FServ)=varDispatch then
  FServ.Visible:=CheckBox1.Checked;
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
```

```
  if vartype(FServ)=varDispatch then
  FServ.OpenFile(Edit1.Text);
end;

procedure TForm1.Button7Click(Sender: TObject);
begin
  if vartype(FServ)=varDispatch then
  FServ.SaveFile(Edit1.Text);
end;

procedure TForm1.Button8Click(Sender: TObject);
begin
  if vartype(FServ)=varDispatch then
  FServ.NewFile;
end;

procedure TForm1.Button9Click(Sender: TObject);
begin
  if vartype(FServ)=varDispatch then
  FServ.AddLine(Edit1.Text);
end;
```

Сохраняем проект под именем ClientAuto и запускаем на выполнение. Проверяем работоспособность сервера и контроллера автоматизации.

Контрольные вопросы

1. Что такое сервер автоматизации?
2. Что такое контроллер автоматизации?
3. Что такое объект автоматизации?
4. Для чего предназначен интерфейс IDispatch?
5. Для чего предназначен метод Invoke интерфейса IDispatch?
6. Что такое вариантный тип?
7. Что такое раннее связывание?
8. Что такое позднее связывание?
9. Как создать сервер автоматизации в Delphi?
10. Как регистрируется сервер автоматизации в реестре Windows?

Лабораторная работа №7
тема: «Создание контроллеров автоматизации приложений MS OFFICE»

Цель работы: получение навыков создания контроллеров автоматизации MS Office.

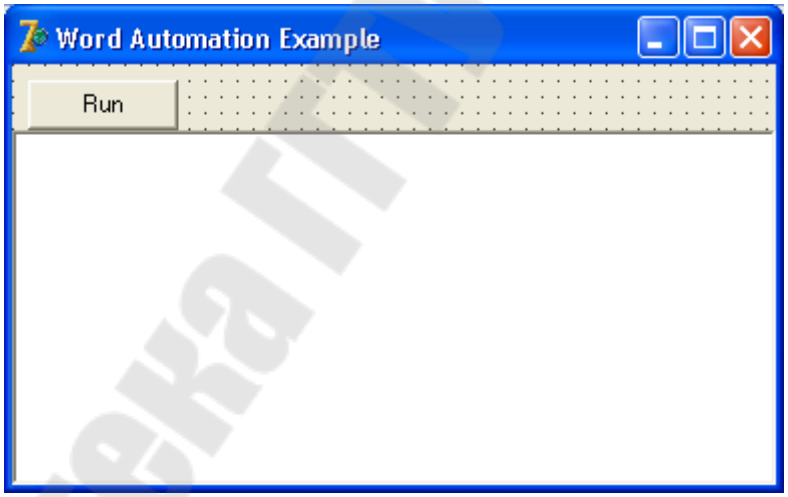
Краткие теоретические сведения:

,

Задание: Требуется разработать контроллеры автоматизации для MS Word и MS Excel

Порядок выполнения работы

1. ***Создание контроллера автоматизации MS Word.*** Создадим обычное приложение вида



В тексте модуля напишем код обработчика события, связанного со щелчком на кнопке RUN

```
unit wordauto;

interface

uses
```

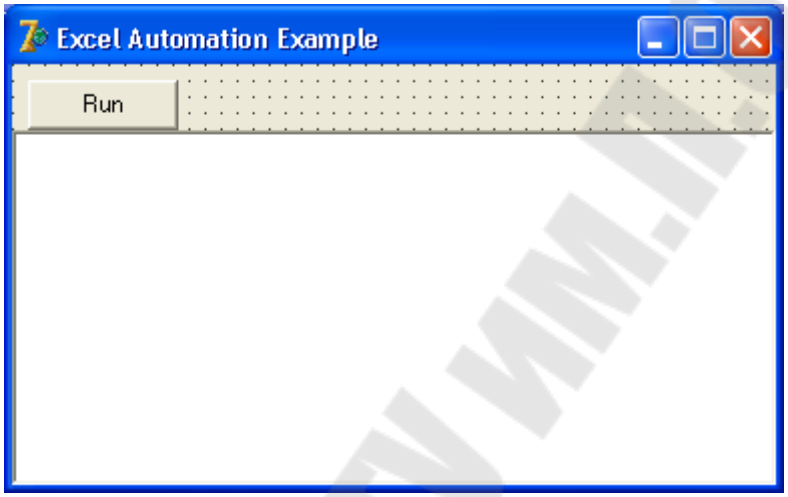
```
    Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
    Dialogs, StdCtrls, ComObj, ActiveX;

  type
    TForm1 = class(TForm)
      Button1: TButton;
      Memo1: TMemo;
      procedure Button1Click(Sender: TObject);
    private
      { Private declarations }
    public
      { Public declarations }
    end;

  var
    Form1: TForm1;

  const
    wdPropertyTitle         = $00000001; // название
    wdPropertySubject       = $00000002; // назначение
    wdPropertyAuthor        = $00000003; // автор
    wdPropertyKeywords      = $00000004; // ключевые слова
    wdPropertyComments      = $00000005; // комментарии
    wdPropertyTemplate      = $00000006; // шаблон
    wdPropertyLastAuthor    = $00000007; // автор, последним
редактировавший текст
    wdCollapseStart         = $00000001; // новый объект
находится в начале фрагмента
    wdCollapseEnd           = $00000000; // новый объект
находится в конце фрагмента

  implementation

  {$R *.dfm}

  procedure TForm1.Button1Click(Sender: TObject);
  var
    ServerIsRunning      : boolean;
```

```
    Unknown          : IUnknown;
    Result           : HResult;
    AppProgID        : String;
    App              : Variant;
    Rng, Sel         : Variant;
    I                : Integer;
  begin
    // Указать программный идентификатор приложения-сервера
    AppProgID := 'Word.Application';
    ServerIsRunning := False;
    Result :=
GetActiveObject(ProgIDToClassID(AppProgID),nil,Unknown);
    if (Result = MK_E_UNAVAILABLE) then
      // Создать один экземпляр сервера
      App := CreateOleObject(AppProgID)
    else
      begin
      // Соединиться с уже запущенной копией сервера
      App := GetActiveOleObject(AppProgID);
      ServerIsRunning := True;
    end;
    // показать окно приложения на экране
    App.Visible := True;
    //-----------------------------------------------------
    //
    // Здесь выполняются другие действия
    // с объектами приложения Office
     App.Documents.Add();
     App.Documents.Item(1).Activate;
     //Вставка и форматирование текста
     App.ActiveDocument.Paragraphs.Add;
     Rng :=  App.ActiveDocument.Paragraphs.Item(1).Range;
     Rng.InsertAfter('Это вставляемый текст '+Chr(13)+Chr(10));
     Rng.InsertAfter('Это вставляемый текст '+Chr(13)+Chr(10));
     Rng.InsertAfter('Это вставляемый текст '+Chr(13)+Chr(10));
     Rng.InsertAfter('Это вставляемый текст '+Chr(13)+Chr(10));
     Rng.InsertAfter('Это вставляемый текст '+Chr(13)+Chr(10));
     App.ActiveDocument.Paragraphs.Item(3).Range.Select;
     Sel := App.Selection;
```

```
        Sel.TypeText('Это текст, которым мы заменим выделенный
фрагмент');
        Sel.TypeParagraph;
        Sel.Font.Bold := True;
        Sel.TypeText('Это текст, который выделен полужирным
шрифтом.');
        Sel.Font.Bold := False;
        Sel.TypeParagraph;
        Sel.Style := 'Заголовок 1';
        Sel.TypeText('Это текст, который станет заголовком');
        Sel.TypeParagraph;
        //Испльзование буфера обмена
        App.ActiveDocument.Paragraphs.Item(3).Range.Select;
        Rng := App.Selection.Range;
        Rng.Copy;
        Rng :=  App.ActiveDocument.Paragraphs.Item(3).Range;
        Rng.Paste;
        Rng := App.Selection.Range;
        //Вставка таблицы
        Rng.Collapse(wdCollapseEnd);
        Rng.InsertAfter('1, 2, 3');
        Rng.InsertParagraphAfter;
        Rng.InsertAfter('4,5,6');
        Rng.InsertParagraphAfter;
        Rng.InsertAfter('7,8,9');
        Rng.InsertParagraphAfter;
        Rng.ConvertToTable(',');
        //Свойства документа
        Memo1.Lines.Add('Название - ' + App.ActiveDocument.
         BuiltInDocumentProperties[wdPropertyTitle].Value);
        Memo1.Lines.Add('Автор - ' + App.ActiveDocument.
         BuiltInDocumentProperties[wdPropertyAuthor].Value);
        Memo1.Lines.Add('Шаблон - ' + App.ActiveDocument.
         BuiltInDocumentProperties[wdPropertyTemplate].Value);
        //Сохранение документа
        App.ActiveDocument.SaveAs('C:\MyWordFile.doc');
       //
       //-------------------------------------------------
       if not ServerIsRunning then App.Quit;
```

```
  App:=Unassigned;
end;

end.
```

2. ***Создание контроллера автоматизации MS Excel***. Создадим обычное приложение вида



В тексте модуля напишем код обработчика события, связанного со щелчком на кнопке RUN

```
unit excelauto;

interface

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

type
  TForm1 = class(TForm)
    Button1: TButton;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
```

```
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
const
  xl3DColumn = -4100;
  xl3DBar = -4099;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var
    ServerIsRunning      : boolean;
    Unknown              : IUnknown;
    Result           : HResult;
    AppProgID            : String;
    App, Ch          : Variant;
    Rng, Sel         : Variant;
    I                    : Integer;
begin
  // Указать программный идентификатор приложения-сервера
  AppProgID := 'Excel.Application';
  ServerIsRunning := False;
  Result :=
GetActiveObject(ProgIDToClassID(AppProgID),nil,Unknown);
  if (Result = MK_E_UNAVAILABLE) then
    // Создать один экземпляр сервера
    App := CreateOleObject(AppProgID)
  else
    begin
    // Соединиться с уже запущенной копией сервера
    App := GetActiveOleObject(AppProgID);
    ServerIsRunning := True;
  end;
  // показать окно приложения на экране
```

```
        App.Visible := True;
        //-----------------------------------------------------
        //
        // Здесь выполняются другие действия
        // с объектами приложения Office
         App.Workbooks.Add();
         I:=App.Workbooks.Count;
         App.Workbooks[I].Activate;
         //Обращение к страницам
         App.ActiveWorkbook.WorkSheets[1].Name := 'Страница 1';
         //Заполнение ячеек данными и формулами
         App.ActiveWorkbook.WorkSheets['Страница 1'].Cells[1,2].Value
:= '25';
         App.ActiveWorkbook.WorkSheets['Страница 1'].Cells[2,2].Value
:= '25';
         App.ActiveWorkbook.WorkSheets['Страница 1'].Cells[3,2].Value
:=  '=SUM(B1:B2)';
         //Форматирвоание ячеек
         App.ActiveWorkbook.WorkSheets[1].Cells[3,2].Interior.Color :=
clYellow;
         App.ActiveWorkbook.WorkSheets[1].Cells[3,2].Font.Color :=
clRed;
         App.ActiveWorkbook.WorkSheets[1].Cells[3,2].Font.Name :=
'Courier';
         App.ActiveWorkbook.WorkSheets[1].Cells[3,2].Font.Size := 16;
         App.ActiveWorkbook.WorkSheets[1].Cells[3,2].Font.Bold :=
True;
         App.ActiveWorkbook.WorkSheets[2].Range['A1:C5'].Value :=
'Test';
         App.ActiveWorkbook.WorkSheets[2].Range['A1:C5'].Font.Color
:= clRed;
         //Применение буфера обмена
         App.ActiveWorkbook.WorkSheets[2].Range['A1:C5'].Copy;
         App.ActiveWorkbook.WorkSheets[3].Paste;
         //Подготовка данных для построения графика
         Randomize;
         For i:=1 to 5 do begin
          App.ActiveWorkbook.WorkSheets[2].Cells[I,2].Value :=
Random(100);
```

```
          App.ActiveWorkbook.WorkSheets[2].Cells[I,3].Value :=
Random(100);
        end;
        //Строим график
        Ch :=
App.ActiveWorkbook.WorkSheets[2].ChartObjects.Add(20,70,400,200);
      Ch.Chart.ChartWizard(App.ActiveWorkbook.WorkSheets[2].Range[
'A1:C5'],xl3dColumn);
        Ch.Chart.HasTitle              := 1;
        Ch.Chart.HasLegend             := False;
        Ch.Chart.ChartTitle.Text       := 'Пример диаграммы Excel';
        Ch.Chart.Axes(1).HasTitle      := True;
        Ch.Chart.Axes(1).AxisTitle.Text      := 'Подпись вдоль оси
абсцисс';
        Ch.Chart.Axes(2).HasTitle      := True;
        Ch.Chart.Axes(2).AxisTitle.Text      := 'Подпись вдоль оси
ординат';
        //Сохраняем файл
        App.DisplayAlerts:=False;
        App.ActiveWorkBook.SaveAs('C:\MyWorkBook.xls');
       //
       //-------------------------------------------------
       if not ServerIsRunning then App.Quit;
       App:=Unassigned;
      end;

      end.
```

1. Как запустить сервер автоматизации MS Word?
2. Как сохранить документ MS Word?
3. Как открыть документ MS Word?
4. Как вставить параграф?
5. Как применяется форматирование текста?
6. Как создается таблица в MS Word?
7. Как вывести данные в таблицу MS Excel?
8. Как построить диаграмму в MS Excel?

ТЕХНОЛОГИИ КОМПОНЕНТНОГО ПРОГРАММИРОВАНИЯ

Лабораторный практикум
по одноименной дисциплине
для слушателей специальности 1-40 01 73
«Программное обеспечение информационных систем»
заочной формы обучения

Составитель: **Самовендюк** Николай Владимирович

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 28.05.14.
Рег. № 75Е.
http://www.gstu.by