**ЛАБОРАТОРНАЯ РАБОТА № 3**

**ОСНОВЫ ОРГАНИЗАЦИИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА В ПРОЕКТАХ LAZARUS**

## ТЕОРИЯ И ПОЯСНЕНИЯ

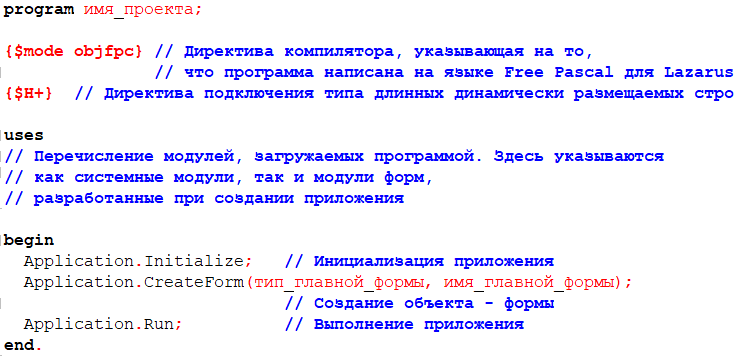
### *Приложение в Lazarus*

***Приложение*,** содержащеестандартный для Windows графический пользовательский интерфейс GUI, создается в Lazarus как ***проект LCL Application*** (Файл – Создать – Project – Application). Среда разработки позволяет программисту применять в проекте любые стандартные элементы интерфейса. Они описаны в библиотеке визуальных компонентов Lazarus Component Library. Окно на рабочем столе Windows, с помощью которого пользователь будет взаимодействовать с приложением, создается как главная форма проекта. Форма – это визуальный компонент, который всегда предоставляется средой разработки при создании нового проекта. Остальные элементы интерфейса добавляются на форму в режиме визуального проектирования, при этом среда автоматически создает соответствующие части программного кода приложения.

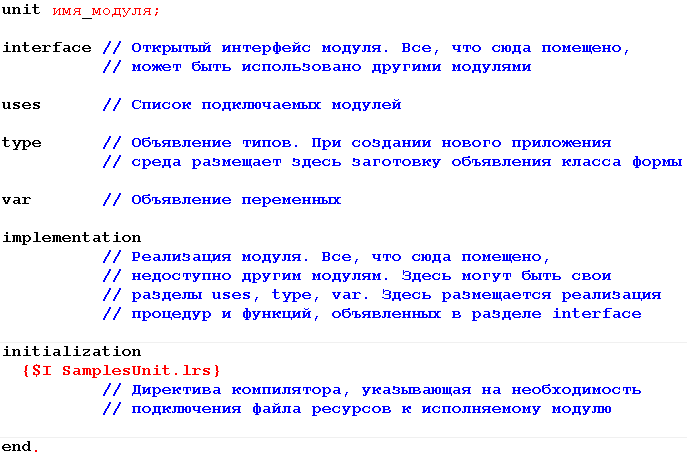
***Файлы проекта***

|  |  |
| --- | --- |
| Файл главной программы проекта (.lpr) | Текстовый файл с исходным кодом основной программы проекта. |
| Файл проекта (.lpi) | Основной файл с описанием проекта в формате XML. |
| Файл проекта (.compiled) | Текстовый файл с установками проекта, в частности, в нем хранятся используемые директивы компилятора. |
| Файл ресурсов (.lrs) | Файл, который содержит описание ресурсов проекта. |
| Файл модуля (.pas) | Текстовый файл модуля. Каждой форме соответствует файл модуля, в котором хранится исходный код. Многие из функций и процедур Lazarus описаны в модулях. |
| Файл формы (.lfm) | Текстовый файл с описанием внешнего вида формы. Файл формы находится в соответствии с файлом модуля. |
| Файл модуля (.o) | Объектный файл модуля (implementation). |
| Файл модуля (.ppu) | Объектный файл модуля (interface). |
| Исполняемый файл (.exe) | Автономный исполняемый файл. |

***Структура файла главной программы проекта***

****

***Структура файла модуля проекта***

****

***Замечание. Имя проекта и имена всех модулей данного проекта должны быть разными.*** *При сохранении проекта сначала запрашивается имя модуля с главной формой, а затем имя самого проекта.*

### *Основные понятия объектно-ориентированного программирования*

**ПРОГРАММИРОВАНИЕ В WINDOWS** производится с учетом следующих положений**:**

**–** разрабатываемый программный код состоит из процедур обработки сообщений, которые Windows посылает окнам приложения;

**–** Windows реагирует на возникающие в программах и аппаратуре события и затем посылает приложению соответствующие сообщения;

– работающее приложение находится в рабочей области памяти и ожидает сообщений от Windows, на которые оно должно реагировать. Таким образом, можно сказать, что приложение находится в бесконечном цикле ожидания сообщений от операционной среды.

Windows предоставляет разработчику **управляемую событиями среду,** в которой код программы исполняется как реакция на события.

**КЛАСС** – это тип данных, определяемый пользователем. Для объявления класса используется служебное слово ***class***. Принято в начале имени класса ставить букву T. Класс включает в себя поля, методы и свойства. Поля – это данные: переменные и константы. Методы – это описанные в классе процедуры и функции. Свойства – это управляемые поля данных, которые влияют на внешний вид и поведение объекта данного класса.

В Lazarus имеется очень большое количество предопределенных классов. Они образуют обширную иерархию, в которой каждый класс создан на основе другого класса. Новый класс будет обладать всеми полями, методами и свойствами класса, на основе которого он создан. Например, класс, на основе которого описывается главная форма приложения, входит в следующую последовательность объявлений классов:

TObject

TPersistent

TComponent

TLCLComponent

TControl

TWinControl

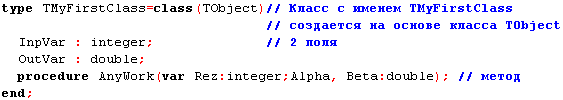
TCustomControl

TScrollingWinControl

TCustomForm

**TForm**

Пример объявления нового класса:



**ОБЪЕКТ** – это экземпляр класса. Объект описывается как переменная типа класса и создается выделением для него памяти в куче (heap) – динамически распределяемой области памяти. Переменная является указателем на начало области памяти, выделенной для объекта. В программе можно использовать любое количество объектов одного и того же класса.

Пример описания переменной типа класса TMyFirstClass:



Эту переменную можно будет использовать в программе для выделения памяти объекту – экземпляру класса TMyFirstClass – и дальнейшей работы с ним.

Обращение к полям этого объекта выполняется с помощью уточняющих имен, т.е. так же, как при работе с записями:



В реализации процедуры-метода перед именем процедуры появляется имя класса:



**КОМПОНЕНТ** – это любой из классов, входящих в состав библиотеки LCL. Каждый элемент пользовательского интерфейса разрабатываемого приложения является экземпляром соответствующего компонента, т.е. объектом. В литературе термин «компонент» часто применяется и к классу, и к объекту этого класса. Различия следует выявлять из контекста.

Свойства многих компонентов можно настраивать визуально в ходе разработки программы. Для каждого визуального компонента (класса) в среде разработки имеется инструментальная кнопка на одной из палитр компонентов: **Standart, Additional, Common Controls, Dialogs** и т.д. Нажатие на кнопку делает ее активной, по щелчку левой кнопкой мыши в нужном месте формы появляется окно компонента**.** Внешний вид компонента (объекта) описывается набором **Свойств**. Поведение компонента (объекта) во время работы приложения программируется в виде процедуры или функции как реакция на возникновение соответствующего **События**.

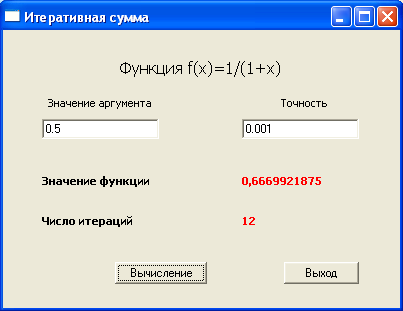
Каждый компонент (объект) имеет свойство ***Name***, его значение – имя переменной, являющейся указателем на этот объект, оно будет использоваться в программном коде приложения. Описание этой переменной среда Lazarus АВТОМАТИЧЕСКИ добавляет в модуль. Многие компоненты (объекты) имеют свойство ***Caption***, его значение – текст, который отображается как заголовок окна компонента. Например, надпись на кнопке задается значением свойства Caption компонента TButton. Набор остальных свойств у разных компонентов варьируется. Задание значений свойств производится в окне **Инспектора объектов.** ИЗМЕНЕНИЯ значений свойств во время визуального проектирования нужно производить ТОЛЬКО В ОКНЕ ИНСПЕКТОРА объектов. Выбранные разработчиком значения свойств компонентов, размещенных на форме, сохраняются в файле .lfm.

Программирование поведения компонента (объекта) производится в следующем порядке. На закладке **События** выполняется двойной щелчок левой кнопкой мыши напротив нужного события. Среда создает заготовку для процедуры обработки этого события. По щелчку на кнопке с многоточием текст заготовки процедуры открывается в окне **Редактора исходного кода**. Программист заполняет процедуру нужными операторами.

## ЗАДАНИЯ

### ИТЕРАТИВНОЕ ВЫЧИСЛЕНИЕ ЗНАЧЕНИЯ ФУНКЦИИ В ЗАДАННОЙ ТОЧКЕ И С ЗАДАННОЙ ТОЧНОСТЬЮ. *Это задание нужно выполнить, если в работе «1 Рекуррентные алгоритмы» НЕ выполнялось* задание *«II.2. Итеративное вычисление бесконечных сумм».*

Создайте приложение для вычисления значения известной функции (лабораторная работа № 1) при заданном значении аргумента и заданной точности вычислений. Выведите число итераций, при котором была достигнута заданная точность результата. Ниже приводится возможный вид формы.



Для создания формы вам понадобятся компоненты, обозначенные на схеме:

Компонент **МЕТКА**

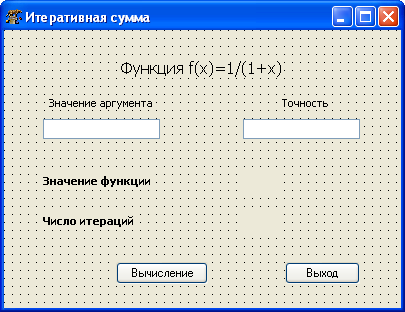
Класс ***TLabel***

Свойства ***Caption, Name, Font***

Компонент **ФОРМА**

Класс ***TForm***

Свойства ***Caption, Name***



Компонент **МЕТКА**

Класс ***TLabel***

Свойства ***Caption, Name, Font***

Компонент **КНОПКА**

Класс ***TButton***

Свойства ***Caption, Name***

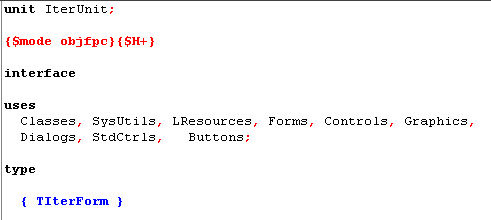
Событие ***OnClick***

Компонент **СТРОКА ВВОДА**

Класс ***TEdit***

Свойства ***Name, Text***

***Программный код модуля созданной формы:***

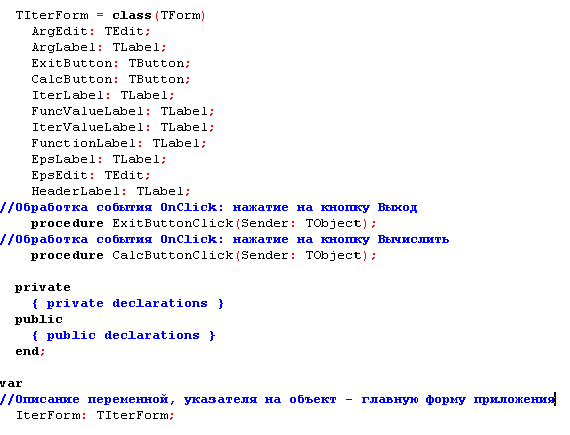


*Эти описания среда разработки записывает автоматически при создании проекта и помещении на форму каждого компонента.*

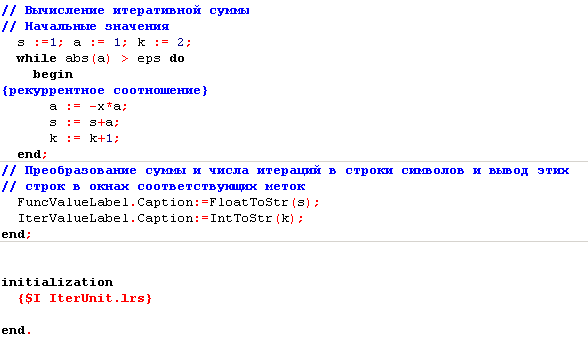
*Свойства компонента настраиваются в окне Инспектора Объектов и отображаются в файле .lfm.*

*Реакция компонента на возникающие события оформляется в виде процедур. Заготовки для них среда автоматически записывает в исходный программный код после того, как разработчик выберет в окне Инспектора Объектов нужное событие и напишет имя процедуры обработки этого события.*

***Замечание****. Любые переименования компонентов или процедур, которые могут потребоваться в процессе разработки приложения, следует выполнять в окне Инспектора Объектов. В этом случае среда сама внесет изменения в программный код, построенный ею раньше.* ***НО*** *эти изменения не коснутся тех строк кода, которые набирал разработчик.*

**



**

### ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ ДЛЯ ЗАДАЧИ ВЫЧИСЛЕНИЯ ПРИБЛИЖЕННЫХ ЗНАЧЕНИЙ ФУНКЦИИ, ПРЕДСТАВЛЕННОЙ В ВИДЕ РЯДА. *Это задание нужно выполнить, если в работе «1 Рекуррентные алгоритмы» выполнялось задание «II.2. Итеративное вычисление бесконечных сумм».*

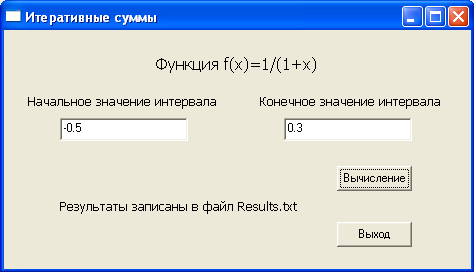
Создайте приложение для вычисления значений известной функции (лабораторная работа № 1) в десяти равноотстоящих друг от друга точках указанного диапазона возможных значений аргумента.

Для каждого значения аргумента выполняйте вычисления до достижения каждой из заданных границ точности и одновременно считайте соответствующее число итераций.

На форме **вводятся** два числа: начальная и конечная точки указанного диапазона возможных значений аргумента.

Результаты вычислений сохраните в текстовом файле.

Ниже приводится возможный вид формы.



Перед выполнением этого задания ознакомьтесь с пояснениями к заданию II.1.