ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

«МЕЖДУНАРОДНЫЙ КОЛЛЕДЖ «ПОЛИГЛОТ»

КУРСОВАЯ РАБОТА

Тема: Проектирование приложения «Инженерный калькулятор»

Студент: Рябцев Максим Александрович

ФИО подпись

Специальность: Программирование в компьютерных системах

Группа 4 ПКС

Руководитель: Кубанова Джамиля Аубекировна

ФИО подпись

Черкесск

2020г.

Содержание

Введение

Язык Паскаль был разработан в 1970 г. Никлаусом Виртом как язык, обеспечивающий строгую типизацию и интуитивно понятный синтаксис. Он был назван в честь французского математика, физика и философа Блеза Паскаля.

Одной из целей создания языка Паскаль Никлаус Вирт считал обучение студентов структурному программированию. До сих пор Паскаль заслуженно считается одним из лучших языков для начального обучения программированию. Его современные модификации, такие как Object Pascal, широко используются в промышленном программировании

Актуальность данной курсовой работы состоит в возможности рассмотрения языка программирования Object Pascal в качестве первого языка программирования, исходя из простоты написания программ используя данный синтаксис.

Object Pascal – это строго типизированный язык высоко уровня, который поддерживает структурное и объектно-ориентированное проектирование. Преимущество языка состоит из легко читаемого кода, быстрой компиляции, и использования нескольких модульных файлов для программирования.

Целью данной курсовой работы является проектирование приложения, «Инженерный калькулятор» и для достижения поставленной цели, необходимо решить следующие задачи;

- Создание формы простого режима;
- Создание формы перевода системы счисления;
- Описание необходимых математических функций;
- Проектировка конвертора систем счисления:

Прежде чем перейти к деталям, нужно определить термины Pascal, связанные с объектно-ориентированным Pascal. Object - Объект представляет собой особый вид записи, который содержит поля, такие как записи; Однако, в отличие от записи, объекты также содержат процедуры и функции, являющиеся частью объекта. Такие процедуры и функции рассматриваются как указатели на методы, связанные с типом объекта;

- Класс - Класс определяется почти так же, как объект, но есть разница в способе их создания. Класс выделяется в куче программы, в то время как объект выделяется на стеке. Класс является указателем на объект, а не самим объектом.
- Создание экземпляров класса - означает создание переменной этого типа класса. Поскольку класс является просто указателем, когда переменная типа класса объявляется, тогда выделяется память только для указателя, а не для всего объекта. Память выделяется для объекта, только когда экземпляр использует один из его конструкторов. Экземпляры класса также называются «объектами», но не путайте их с Object Pascal объектами. Мы будем писать «объект» для Pascal Object и "объект" для экземпляра концептуального объекта или класса.
- Переменные-члены - Это переменные, определенные внутри класса или объекта.
- Функции-члены - Это функции или процедуры, определенные внутри класса или объекта и используются для доступа к данным объекта.
- Видимость членов - члены объекта или класса также называют полями. Эти поля имеют различные уровни видимости. Видимость относится к доступности членов, т.е. именно там, где эти члены будут доступны для обращения. Объекты имеют три уровня видимости: public, private and protected. Классы имеют пять типов Видимость: public, private, strictly private, protected and published.

Глава 1. Постановка задачи

Необходимо разработать приложение, «Инженерный калькулятор», следовательно, в данной курсовой работе мы рассмотрим подробный процесс создания приложения пользуясь базовыми знаниями объектно-ориентированного языка Object Pascal. Рассмотрим методы и атрибуты, которые будут использовать в процессе разработки и создания программы.

По окончанию процесса разработки приложения должно быть:

- Приложение состоящие из двух независящих друг от друга форм
- Кнопка переключения между этими формами
- Панель ввода чисел и выбор математических функций

1.1. Основополагающая приложения «Инженерный калькулятор»

Калькулятор — электронное вычислительное устройство для выполнен ия операций над числами или алгебраическими формулами.

Калькулятор заменил механические вычислительные устройства, такие, как абаки, счёты, логарифмические линейки, механические или электромеханические арифмометры, а также математические таблицы (прежде всего — таблицы логарифмов).

В зависимости от возможностей и целевой сферы применения калькуляторы делятся на простейшие, бухгалтерские, инженерные (научные), финансовые. В отдельные классы обычно выделяют программируемые калькуляторы, дающие возможность выполнения сложных вычислений по предварительно заложенной программе, а также графические — поддерживающие построение и отображение графиков. Специализированные

калькуляторы предназначены для выполнения вычислений в достаточно узкой сфере (финансовые, строительные и т. п.)

В данной курсовой работе мы будем реализовывать простой калькулятор и калькулятор систем счисления.

1.2. Описание среды разработки

Среда разработки, которая будет использована для реализации, описанной выше задачи, называется Lazarus.

Lazarus это бесплатный инструмент разработки с открытым кодом. Он представляет собой среду с графическим интерфейсом для быстрой разработки программ, аналогичную Delphi, и базируется на оригинальной кроссплатформенной библиотеке визуальных компонентов Lazarus Component Library, совместимых с Delphi. В состав входят и не визуальные компоненты. Такого набора достаточно для создания программ с графическим интерфейсом и приложений, работающих с базами данных и Интернетом.

В среде Lazarus используются собственный формат управления пакетами и свои файлы проектов.

Lazarus это стабильная богатая возможностями среда разработки для создания самостоятельных графических и консольных приложений. В настоящее время она работает на Linux, FreeBSD и Windows и предоставляет настраиваемый редактор кода и визуальную среду создания форм вместе с менеджером пакетов, отладчиком и графическим интерфейсом, полностью интегрированным с компилятором FreePascal.

Рассмотрим основные элементы среды разработки Lazarus. Среда Lazarus состоит из нескольких, не связанных окон.

Lazarus использует в своей работе объектно=ориентированный принцип программирования.

Объектно-ориентированное программирование – это методика разработки программ, в основе которой лежит понятие объекта.

Объект – это совокупность свойств параметров, определенных сущностей и методов их обработки программных средств.

Каждый объект имеет:

- Свойства – характеристики объекта (автомобиль характеризуется маркой, цветом, салоном, местонахождением руля, коробкой передач и т.д.).
- Методы – действия объекта, что объект может делать
- События – изменения в окружающей объект обстановке
- Реакция на события – описания действий, которые необходимо совершить при данном событии.

Все объекты реализованы в Lazarus в виде палитры визуальных компонентов(рисунок1)

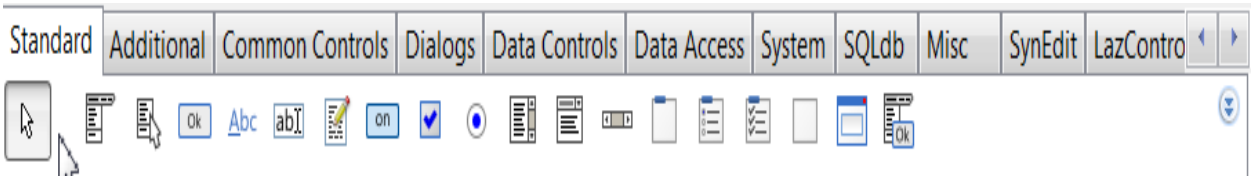


Рисунок 1 – Палитра объектов

Компоненты сгруппированы в отдельные страницы, каждая из которых снабжена закладкой. Если щелкнуть мышью на одну из закладок, то можно перейти на соответствующую ей страницу. При подведении курсора мыши к компоненте появляется подсказка – название компоненты.

1.3. Панели инструментов Lazarus

С помощью окна можно управлять процессом разработки приложения. В нем предусмотрены команды управления файлами, компиляцией,

редактированием, окнами и т.д. Окно разбито на три функциональных блока: Главное меню. В нем расположены команды управления файлами, команды управления компиляцией и свойствами всего приложения, команды управления окнами и настройками среды и многое другое. Меню располагается в верхней части основного окна.

- Панель инструментов. Панель инструментов предоставляет быстрый доступ к основным командам главного меню. Она расположена в левой части главного окна, под главным меню.
- Палитра компонентов. Предоставляет доступ к основным компонентам среды разработки, например, поле ввода, надпись, меню и т.п.(рисунок 2)

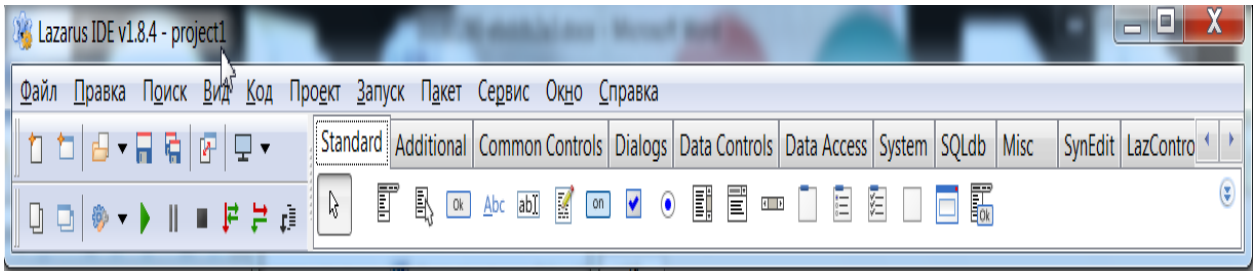


Рисунок 2 – Главное окно

Именно в этом окне мы будем набирать тексты своих программ. Многие функции и возможности этого редактора совпадают с возможностями обычных текстовых редакторов, например, Блокнота. Текст в редакторе можно выделять, копировать, вырезать, вставлять. Кроме того, в редакторе можно осуществлять поиск заданного фрагмента текста, выполнять вставку и замену. Но, конечно, этот редактор исходных текстов Lazarus обладает еще рядом дополнительных возможностей для комфортной работы применительно к разработке программ. Основное преимущество редактора заключается в том, что он обладает возможностями подсветки синтаксиса, причем не только Pascal, но и других языков, а также рядом других удобств. В частности, выделенный фрагмент текста можно сдвигать вправо или влево на количество позиций, указанных в настройках. Отступ блока, что очень удобно для форматирования с целью структурирования кода. Выделенный фрагмент

можно закомментировать или раскомментировать, перевести в верхний или нижний регистр и т.д(рисунок3).

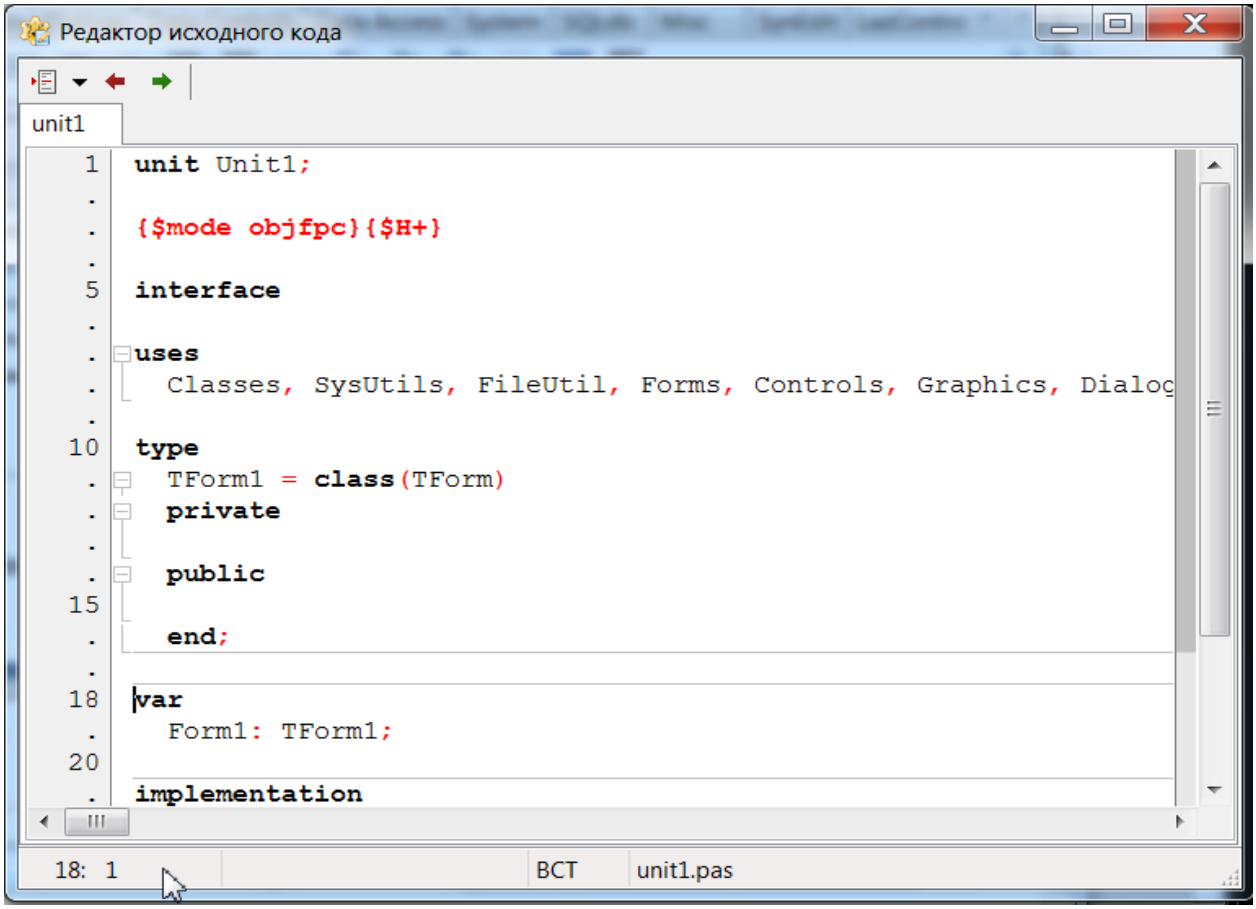


Рисунок 3 – Редактор исходного кода

В этом окне выводятся сообщения компилятора, компоновщика и отладчика. Окно использует различные цвета для объяснения действий компилятора, красный означает ошибку, жёлтый работу компилятора, а зеленый успех компиляции(рисунок 4).

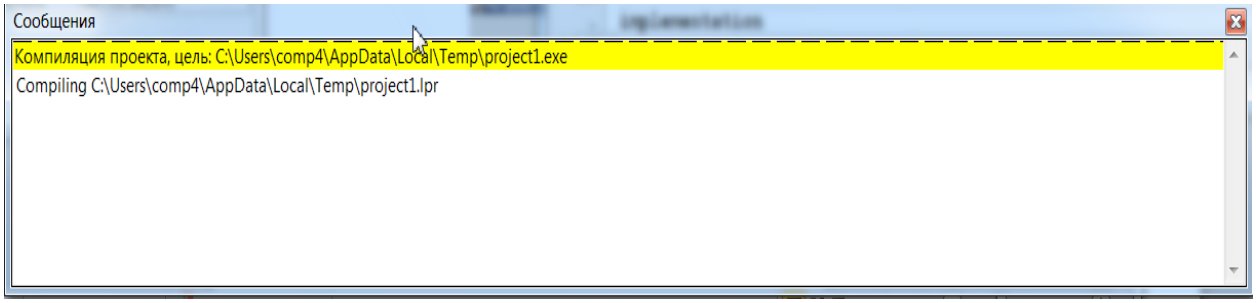


Рисунок 4 – Окно сообщений

Окно формы представляет собой проект окна Windows: имеет заголовок, кнопку вызова системного меню, кнопку закрытия окнак. На форме размещаются компоненты(рисунок 5).

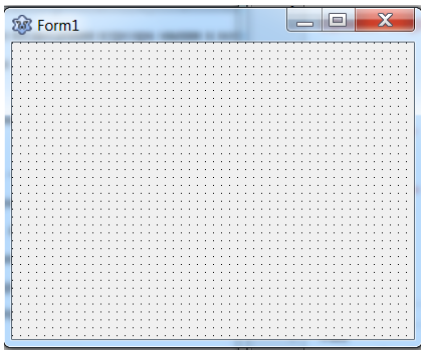


Рисунок 5 – Окно формы

В верхней части окна показывается иерархия объектов, а снизу, расположены три вкладки: "Свойства", "События", "Избранное". Назначение инспектора объекта – это просмотр всех свойств и методов объектов. На вкладке "Свойства" перечисляются все свойства выбранного объекта. На вкладке "Избранное" избранные свойства и методы(рисунок 6).

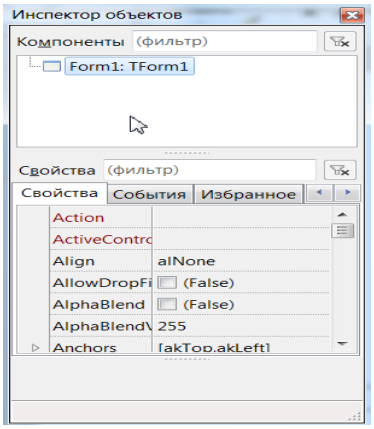


Рисунок 6 – Инспектор объектов

Глава 2. Процесс создания программы «Инженерный калькулятор»

Обычно процесс проектирования приложения или какой-либо программы начинается с создания алгоритма, для того, чтобы наглядно видеть структуру проектано в данном случае начнем сразу с реализации, так как программа не требует использовать сложные объекты или классы.

Приложение будет состоять из двух форм калькулятора, «простого» и «калькулятора систем счисления», следовательно, в данной курсовой работе будет описан подробный процесс создания обеих этих форм.

2.1. Проектирование простого калькулятора

Начальным этапом будет создание простого калькулятора, который сможет производить простые математические вычисления, такие как сложение, вычитание, умножение, деление и т.д.

После создания проекта, автоматически создается одна форма и открывается так называемое «Окно формы» и на нём будем расставлять наши объекты(рисунок 7)..

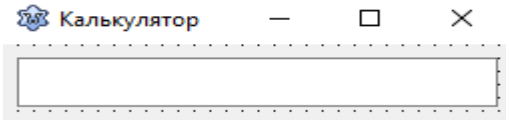


Рисунок 7 – TEdit

Для начала создадим элемент «TEdit», который позволит нам записывать данные с клавиатуры или же с кнопок управления и после переводить их в машинных код и обратно.

Далее немного увеличив размер формы в ход за «строчкой» пойдут кнопки, те самые, которыми мы будем управлять самим калькулятором. Эти кнопки называются «TButton» и по своей сути могут выполнять большой спектр функций, начиная с банального элемента ввода текста в строку, до вычисления сложных функций одним нажатием(рисунок 8).

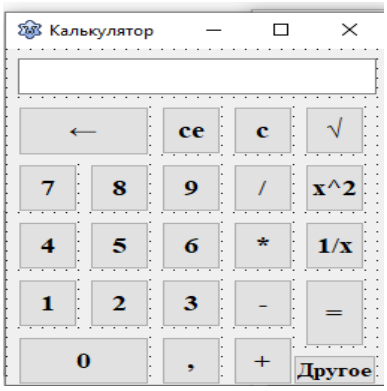


Рисунок 8 – Окно простого калькулятора

После расположения кнопок на форме, нужно им задать название. При помощи свойства «Caption» задаем название для каждой кнопки на форме, так же оставляем место для дополнительной кнопки, которая будет перемещать нас на вторую форму где будут располагаться дополнительные функции.

2.2. Процесс построения функций

Для того, чтобы кнопки не были простыми объектами им нужно при помощи редактора исходного кода присвоить определенную функцию. В данном случае они будут выступать кнопками управления.

В данной курсовой работе будет использована процедура, при которой значения записанные в свойстве «Caption» будут использоваться как переменные(рисунок 9).

```
procedure TForm1.ClickZnak(Sender: TObject);
begin
   a := StrToFloat(Edit1.Text);
   Edit1.Clear;

   znak :=(Sender as TButton).Caption;

end;
```

Рисунок 9 – Процедура управления кнопками

Дальше будем настраивать кнопку, которая будет стирать по одному знаку в строчке, так называемый «Backspace». Процедура работает по типу удаление последнего символа, введенного в строку(рисунок 10).

```
procedure TForm1.But22Click(Sender: TObject);
var
  str : String;
begin
   str := Edit1.Text;
   if str <> '' then
    Delete(str, Length(str),1);
   Edit1.Text:= str;
end;
```

Рисунок 10 – Процедура кнопки Backspace

Следующем шагом будет создание кнопок удаление всех символов в строке и полное удаление. Работают они практически одинаково, исключение состоит в том, что последняя помимо удаления символом в строке приводит все переменные к исходному состоянию(рисунок 11).

```
procedure TForm1.But21Click(Sender: TObject);
begin
   Edit1.Clear;
end;

procedure TForm1.But16Click(Sender: TObject);
begin
   b := StrToFloat(Edit1.Text);
   Edit1.Clear;
   case znak of
   '+' : c := a+b;
   '-' : c := a-b;
   '*' : c := a*b;
   '/' : if a=0 then
   showmessage ('Незя так') else
     c := a/b;

   end;

   Edit1.Text:= FloatToStr(c);

end;
```

Рисунок 11 – Кнопки удаления

Теперь, так как у нас готовы основные кнопки начнем создавать простейшие алгебраические функции и выполняться они будут при помощи оператора выбора «case of»(рисунок 12).

```
procedure TForm1.But16Click(Sender: TObject);
begin
  b := StrToFloat(Edit1.Text);
  Edit1.Clear;
  case znak of
  '+' : c := a+b;
  '-' : c := a-b;
  '*' : c := a*b;
  '/' : if a=0 then
  showmessage ('Нельзя так') else
    c := a/b;

  end;

  Edit1.Text:= FloatToStr(c);

end;
```

Рисунок 12 – Кнопка равно

Работают эти функции по типу кнопок управления и при нажатии на кнопку равенства, оператор считывает символ в переменной и выбирает правильную алгебраическую функцию, после чего выводит ответ.

Следующей кнопкой будет кнопка, которая отвечает за возведение числа в квадрат или же в вторую степень. Выполняется путем использования команды «sqr»(рисунок 13).

```
procedure TForm1.But18Click(Sender: TObject);
begin
 a := StrToFloat(Edit1.Text);
 a := sqr(a);
 Edit1.Text:=FloatToStr(a);
 a := 0;
end;
```

Рисунок 13 – Кнопка возведения в квадрат

По аналогии создадим кнопку, которая будет выводить число из-под корня, путем изменения команды из «sqr» в «sqrt».(рисунок 14).

```
procedure TForm1.But19Click(Sender: TObject);
begin
  a := StrToFloat(Edit1.Text);
  a := sqrt(a);
  Edit1.Text:=FloatToStr(a);
  a := 0;

end;
```

Рисунок 14 – Кнопка вывода из корня

И последней кнопкой, имеющей алгебраическую функцию будет кнопка дроби, где единица делится на введенное нами значение. Выполняется банальным деление единицы на значение переменной(рисунок 15).

```
procedure TForm1.But17Click(Sender: TObject);
begin
  a := StrToFloat(Edit1.Text);
  a := 1/(a);
  Edit1.Text:=FloatToStr(a);
  a := 0;
end;
```

Рисунок 15 – Кнопка деления единицы на переменную

Последней же кнопкой на этой форме является кнопка перехода в другой режим при помощи команд закрытия текущей формы и открытия новой(рисунок 16)

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    form1.hide;
    form2.show;
end;
```

Рисунок 16 – Кнопка перехода

2.3. Проектирование калькулятора «СС»

Данный режим предназначен для того, что автоматически перевод числа из разных систем счисления.

Система счисления – это метод записи числа при помощи указанного набора специальных знаков.

На данный момент мы имеем четыре основных систем счисления;

- Десятичная;
- Двоичная;
- Восьмеричная;
- Шестнадцатеричная:

По аналогии предыдущей формы создаем идентичную и так же добавляем на нее сточку для ввода и сами кнопки управления. Так же добавим элементы, называемые «Radiobutton», их будет четыре, по одной на каждую систему счисления(рисунок 17).

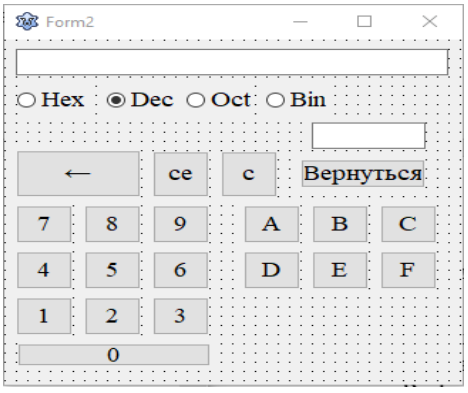


Рисунок 17 – Окно калькулятора СС

Radiobutton – это элемент имеющий два состояния «true» и «false». Некий переключатель, который поможет пользователю упростить использование данной функции, хоть и займет больше сил на его описание.

Так же, как и на предыдущей форме создаем кнопки управления и добавляем к ним несколько дополнительных кнопок отвечающих за числа шестнадцатеричной системе счисления.

В самом начале задаем программе функцию, которая будет отвечать за то чтобы при запуске переключатель отвечающий за десятичную систему счисления. Так же добавляем отдельную, независимую переменную, которая будет отвечать за то, какая система счисления выбрана на данный момент, добавим к ней отдельную строку для вывода, чтобы наглядно видеть работают ли переключатели и отключить те кнопки, которые не используются в шестнадцатеричной системе счисления(рисунок 18).

```
procedure TForm2.FormCreate(Sender: TObject);
begin
    SS1:=10;
  Edit2.Text:=floattostr(SS1);
    Button10.Enabled:=false;
  Button11.Enabled:=false;
  Button12.Enabled:=false;
  Button13.Enabled:=false;
  Button14.Enabled:=false;
  Button15.Enabled:=false;
  end;
```

Рисунок 18 – Процедура при создании формы

Для начала напишем процедуру, которая будет переводить числа из десятичной в любые другие системы счисления, путем создания конвертора.

Чтобы конвертор работал правильно, нужно присвоить константу символов и чисел, с которыми будет работать процедура(рисунок 19).

```
const
digit: string[36] = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ';
```

Рисунок 19 – Описание константы

Метод работы процедуры будет заключаться на алгебраической функции, то есть в зависимости от переменной отвечающей за то, какая система счисления будет использоваться процедура будет делить число, записанное нами на нужную переменную. Если перевод идет из десятичной в двоичную, то процедура делит переменную на два, до того момента пока не получит ответ, который не сможет поделить на два. Так же по аналогии с остальными системами счисления(рисунок 20).

```
function FromDec(n, r: longint): string;
var
   s: string;
begin
   s := '';
   repeat
     s := digit[(n mod r) + 1] + s;
     n := n div r;
   until n = 0;
   FromDec := s;
end;
```

Рис. 20. «Конвертор из десятичной СС»

Теперь по антологии нужно написать процедуру, которая будет переводить число из любой системы счисления в десятичную(рисунок 21).

```
begin
   if SS1=16 then
begin
   k:=0;
   s:=Edit1.Text;
   n:=Length(s);
   for m:=1 to n do begin
     k:=k*16;
     case s[m] of
       '0': k:=k+0;
       '1': k:=k+1;
       '2': k:=k+2;
       '3': k:=k+3;
       '4': k:=k+4;
       '5': k:=k+5;
       '6': k:=k+6;
       '7': k:=k+7;
       '8': k:=k+8;
       '9': k:=k+9;
       'A': k:=k+10;
       'B': k:=k+11;
       'C': k:=k+12;
       'D': k:=k+13;
       'E': k:=k+14;
       'F': k:=k+15;
     end;
   end;
   Edit1.Text:=IntToStr(k);
```

Рисунок 21 – Конвертор из любой в СС

Вместо вызова процедуры, будем писать функцию внутри каждого переключателя, не очень практично, но зато будет работать стабильно. Функция работает по принципу представления каждого символа в строке под нужный описанный в операторе выбора и сложенного в единое число в результате.

Данный пример выполнен для перевода из шестнадцатеричной в десятичную, но для перевода из других систем мы будем использовать эту же функцию, но при этом нужно учитывать систему счисления и изменять шаг и количество символов в операторе, то есть если нужна двоичная система, то будет использоваться только ноль и единица.

Следующим шагом будет присваивание определенных функций каждому переключателю, путем внедрения функции перевода из любой системы счисления в десятичную, а после перевода в нужную нам систему. Но из-за исходя из особенности определения системы нужно в каждом переключателе описать три разных цикла, при котором будет выбран нужный путем определения той самой переменной, которая отвечает за нужную систему счисления(рисунок 22).

```
begin
 if SS1=10 then
   begin
    n:=StrToInt(Edit1.Text);
    Edit1.text:=FromDec(n,16);
   end
 else if SS1=2 then
   begin
    k:=0;
    s:=Edit1.Text;
    n:=Length(s);
    for m:=1 to n do begin
      k:=k*2;
      case s[m] of
        '0': k:=k+0;
        '1': k:=k+1;
      end;
    end;
    Edit1.Text:=IntToStr(k);
    n:=StrToInt(Edit1.Text);
  Edit1.text:=FromDec(n,16);
  end
  SS1:=16;
  Edit2.Text:=inttostr(SS1);
```

Рисунок 22 – Пример работы radiobutton

Из приведенного выше примера можно понять, что программа узнает в начале какая система счисления выбрана на данный момент, после выбирает нужный вариант, проводит вычисления, а потом выводит результат в строку и записывает в переменную отвечающую за систему счисления нужное значение. По такому же принципу описываем для каждого переключателя.

И на последнем этапе создаем кнопку, которая позволит нам вернуться обратно в простой режим при помощи тех же команд открытия и закрытия форм(рисунок 23).

```
procedure TForm2.Button16Click(Sender: TObject);
begin
    Form2.hide;
    Form1.show;
end;
```

Рисунок 23 – Кнопка перехода в простой режим

Заключение

В ходе выполнения курсовой работы было разработано приложение «Инженерный калькулятор». Данное приложение было разработано и реализовано в среде разработки Lazarus на языке Object Pascal.

Поставленные цели и задачи данной курсовой работы были достигнуты, путём проведения следующий действий;

- Создание формы простого режима;
- Создание формы перевода системы счисления;
- Описания необходимых математические функции;
- Проектировка конвертора систем счисления:

Данный формат разработки приложения помогает разграничить необходимый функционал для пользователя и материал для работы программисту.

Преимущества;

- Ускоренная разработка ПО;
- Визуальное удобство при проектировке структуры программы;
- Легкость внесения изменений в программу и поиск ошибок.

Данная курсовая работа позволила повторить основные аспекты языка Pascal и узнать некоторые особенности языка Object Pascal, разобраться в особенностях поставленной задачи и ознакомиться с средой разработки Lazarus.

Список источников и литературы

1. Г. Г. Рапаков, С. Ю. Ржеуцкая- Программирование на языке Pascal. 2016. – 425 с;
2. Бонд Д. Object Pascal. Программирование от идеи до реализации. – Прогресс книга. 2018. – 928 с;
3. Вандевурд Д., Джосаттис Н., Грегор Д. Шаблоны Pascal. Справочник разработчика. – Альфа – книга. 2015. – 848 с;
4. Васильев А. Программирование на Object Pascal для начинающих. Особенности языка. 2016. – 528 с;
5. 4Васильев А.Н. Программирование на Pascal в примерах и задачах: Учебное пособие. 2016. – 368 с;
6. Виллемер А. Программирование на Pascal. 2018. – 528 с;
7. Иванова Г. Программирование. Учебник. 2019. – 426 с;
8. Комлев Н. Объектно-ориентированное Программирование. Настольная книга программиста. – Салон Пресс. 2017. – 298 с;
9. Лафоре Р. Объектно-ориентированное программирование в Object Pascal. –Питер СПб. 2015. – 928 с;
10. Липпман С., Лажойе Ж., Му Б. Язык программирования Object Pascal Pascal. Базовый курс. – Диалектика. 2018. – 1120 с;
11. Магда Ю. Программирование и отладка приложений. – ДМК Пресс. 2017. – 168 с;
12. Павловская Т. Object Pascal Программирование на языке высокого уровня. – Питер СПб. 2019. – 432 с;
13. Подбельский В. Язык Object Pascal. Решение задач. – Финансы и статистика. 2014;
14. Шилдт Г. Object Pascal для начинающих. – ЭКОМ. 2017. – 640 с;
15. https://ru.bmstu.wiki/Object_Pascal;
16. https://www.calc.ru/Sistemy-Schisleniya-Osnovnyye-Ponyatiya.html;
17. https://ru.wikipedia.org/wiki/Паскаль_(язык_программирования) ;

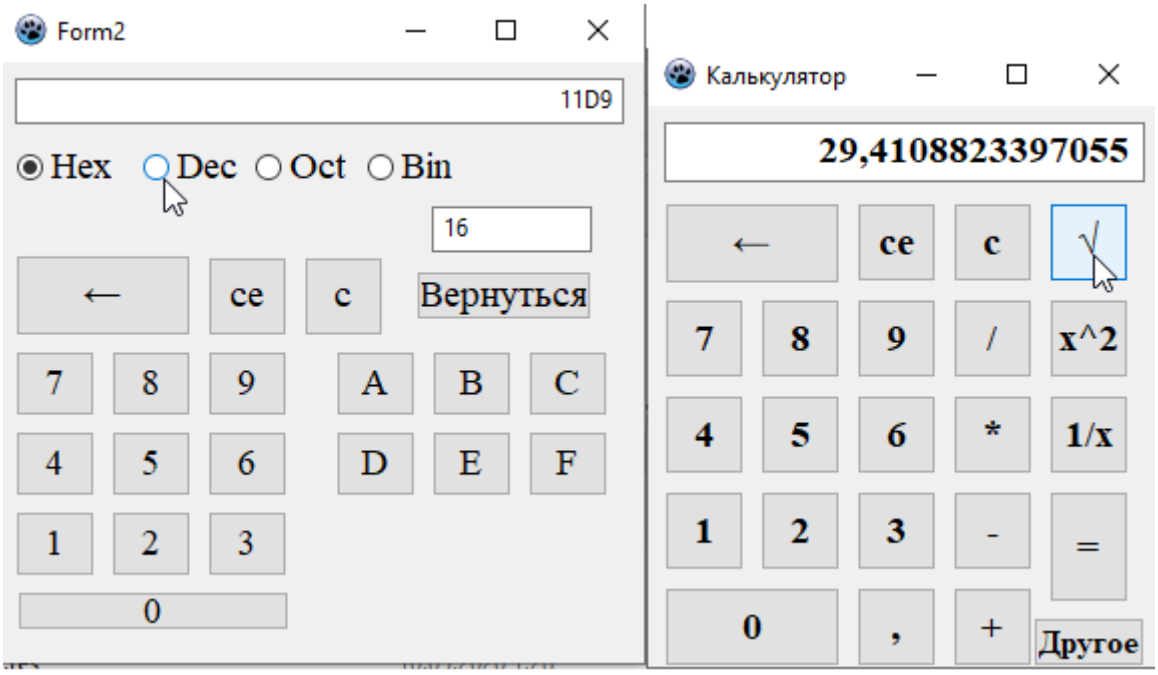
18.https://ru.wikipedia.org/wiki/Free_Pascal;

19.https://ru.wikipedia.org/wiki/Lazarus;

20.https://ru.wikipedia.org/wiki/Delphi_(среда_разработки):

Приложения

Приложение 1 – Внешний вид программы



Приложение 2 – Листинг первой формы

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls;

type
  { TForm1 }
  TForm1 = class(TForm)
    But1: TButton;
```

```
But10: TButton;
But11: TButton;
But12: TButton;
But13: TButton;
But14: TButton;
But15: TButton;
But16: TButton;
But17: TButton;
But18: TButton;
But19: TButton;
But2: TButton;
But20: TButton;
But21: TButton;
But22: TButton;
But3: TButton;
But4: TButton;
But5: TButton;
But6: TButton;
But7: TButton;
But8: TButton;
But9: TButton;
Button1: TButton;
```

```
    Edit1: TEdit;
    procedure But16Click(Sender: TObject);
    procedure But17Click(Sender: TObject);
    procedure But18Click(Sender: TObject);
    procedure But19Click(Sender: TObject);
    procedure But20Click(Sender: TObject);
    procedure But21Click(Sender: TObject);
    procedure But22Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure ClickBut(Sender: TObject);
    procedure ClickZnak(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;
var
  Form1: TForm1;
  a, b, c : Real;
  znak : String;
```

```
implementation

uses unit2;

{$R *.lfm}

{ TForm1 }

procedure TForm1.ClickZnak(Sender: TObject);

begin

  a := StrToFloat(Edit1.Text);

  Edit1.Clear;

  znak :=(Sender as TButton).Caption;

end;

procedure TForm1.Edit1Change(Sender: TObject);

begin

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

end;

procedure TForm1.ClickBut(Sender: TObject);

begin

  Edit1.Text:=Edit1.Text + ( Sender as TButton).Caption;

end;


procedure TForm1.But22Click(Sender: TObject);
```

```
var
 str : String;
begin
  str := Edit1.Text;
  if str <> '' then
   Delete(str, Length(str),1);
  Edit1.Text:= str;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
   form1.hide;
   form2.show;
end;
procedure TForm1.But21Click(Sender: TObject);
begin
  Edit1.Clear;
end;
procedure TForm1.But16Click(Sender: TObject);
begin
  b := StrToFloat(Edit1.Text);
  Edit1.Clear;
  case znak of
```

```
  '+' : c := a+b;
  '-' : c := a-b;
  '*' : c := a*b;
  '/' : if a=0 then
  showmessage ('Незя так') else
    c := a/b;
  end;
  Edit1.Text:= FloatToStr(c);
end;
procedure TForm1.But17Click(Sender: TObject);
begin
 a := StrToFloat(Edit1.Text);
 a := 1/(a);
 Edit1.Text:=FloatToStr(a);
 a := 0;
end;
procedure TForm1.But18Click(Sender: TObject);
begin
 a := StrToFloat(Edit1.Text);
 a := sqr(a);
 Edit1.Text:=FloatToStr(a);
 a := 0;
```

```
end;
procedure TForm1.But19Click(Sender: TObject);
begin
 a := StrToFloat(Edit1.Text);
 a := sqrt(a);
 Edit1.Text:=FloatToStr(a);
 a := 0;
end;
procedure TForm1.But20Click(Sender: TObject);
begin
  Edit1.Clear;
  a:=0;
  b:=0;
  c:=0;
end;
end.
```

Приложение 3 – Листинг второй формы

```
unit Unit2;
{$mode objfpc}{$H+}
interface
uses
```

```
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, StdCtrls, ExtCtrls, Buttons, Math,
  ComCtrls,clipbrd;
type
  { TForm2 }
  TForm2 = class(TForm)
    But20: TButton;
    But21: TButton;
    But22: TButton;
    Button1: TButton;
    Button10: TButton;
    Button11: TButton;
    Button12: TButton;
    Button13: TButton;
    Button14: TButton;
    Button15: TButton;
    Button16: TButton;
    Button0: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
```

```
Button6: TButton;
Button7: TButton;
Button8: TButton;
Button9: TButton;
Edit1: TEdit;
Edit2: TEdit;
RadioButton1: TRadioButton;
RadioButton2: TRadioButton;
RadioButton3: TRadioButton;
RadioButton4: TRadioButton;
procedure But20Click(Sender: TObject);
procedure But21Click(Sender: TObject);
procedure But22Click(Sender: TObject);
procedure Button16Click(Sender: TObject);
procedure Button9Click(Sender: TObject);
procedure ClickBut(Sender: TObject);
procedure ClickZnak(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Edit2Change(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure RadioButton1Change(Sender: TObject);
procedure RadioButton2Change(Sender: TObject);
```

```
    procedure RadioButton3Change(Sender: TObject);
    procedure RadioButton4Change(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;
var
  Form2: TForm2; SS:extended;
  SS1:integer;
  const
  digit: string[36] = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ';
implementation
uses unit1;
{$R *.lfm}
//dec v luboi ss
function FromDec(n, r: longint): string;
var
  s: string;
begin
  s := '';
  repeat
```

```
    s := digit[(n mod r) + 1] + s;
    n := n div r;
  until n = 0;
  FromDec := s;
end;
{ TForm2 }
procedure TForm2.ClickBut(Sender: TObject);
begin
  Edit1.Text:=Edit1.Text + ( Sender as TButton).Caption;
end;
procedure TForm2.Button9Click(Sender: TObject);
begin
end;
procedure TForm2.Button16Click(Sender: TObject);
begin
   Form2.hide;
   Form1.show;
end;
procedure TForm2.But22Click(Sender: TObject);
var
 str : String;
begin
```

```
  str := Edit1.Text;
  if str <> '' then
   Delete(str, Length(str),1);
  Edit1.Text:= str;
end;
procedure TForm2.But21Click(Sender: TObject);
begin
  Edit1.Clear;
end;
procedure TForm2.But20Click(Sender: TObject);
begin
  Edit1.Clear;
  a:=0;
  b:=0;
  c:=0;
end;
procedure TForm2.ClickZnak(Sender: TObject);
begin
  a := StrToFloat(Edit1.Text);
  Edit1.Clear;
  znak :=(Sender as TButton).Caption;
end;
```

```
procedure TForm2.Edit1Change(Sender: TObject);
begin
end;
procedure TForm2.Edit2Change(Sender: TObject);
begin
end;
procedure TForm2.FormCreate(Sender: TObject);
begin
   SS1:=10;
  Edit2.Text:=floattostr(SS1);
   Button10.Enabled:=false;
  Button11.Enabled:=false;
  Button12.Enabled:=false;
  Button13.Enabled:=false;
  Button14.Enabled:=false;
  Button15.Enabled:=false;
  end;
procedure TForm2.RadioButton2Change(Sender: TObject);
var s: string;
   n,m,k: integer;
begin
  if SS1=16 then
```

```
begin
  k:=0;
  s:=Edit1.Text;
  n:=Length(s);
  for m:=1 to n do begin
    k:=k*16;
    case s[m] of
      '0': k:=k+0;
      '1': k:=k+1;
      '2': k:=k+2;
      '3': k:=k+3;
      '4': k:=k+4;
      '5': k:=k+5;
      '6': k:=k+6;
      '7': k:=k+7;
      '8': k:=k+8;
      '9': k:=k+9;
      'A': k:=k+10;
      'B': k:=k+11;
      'C': k:=k+12;
      'D': k:=k+13;
      'E': k:=k+14;
```

```
      'F': k:=k+15;
    end;
  end;
  Edit1.Text:=IntToStr(k);
end
else if SS1=8 then
  begin
    k:=0;
    s:=Edit1.Text;
    n:=Length(s);
    for m:=1 to n do begin
      k:=k*8;
      case s[m] of
        '0': k:=k+0;
        '1': k:=k+1;
        '2': k:=k+2;
        '3': k:=k+3;
        '4': k:=k+4;
        '5': k:=k+5;
        '6': k:=k+6;
        '7': k:=k+7;
      end;
```

```
    end;
    Edit1.Text:=IntToStr(k);
  end
else if SS1=2 then
    begin
      k:=0;
      s:=Edit1.Text;
      n:=Length(s);
      for m:=1 to n do begin
        k:=k*2;
        case s[m] of
          '0': k:=k+0;
          '1': k:=k+1;
        end;
      end;
      Edit1.Text:=IntToStr(k);
end;
 SS1:=10;
  Edit2.Text:=inttostr(SS1);
  Button1.Enabled:=true;
  Button2.Enabled:=true;
  Button3.Enabled:=true;
```

```
 Button4.Enabled:=true;
 Button5.Enabled:=true;
 Button6.Enabled:=true;
 Button7.Enabled:=true;
 Button8.Enabled:=true;
 Button9.Enabled:=true;
 Button0.Enabled:=true;
 Button10.Enabled:=false;
 Button11.Enabled:=false;
 Button12.Enabled:=false;
 Button13.Enabled:=false;
 Button14.Enabled:=false;
 Button15.Enabled:=false;
 end;
procedure TForm2.RadioButton1Change(Sender: TObject);
var s: string;
    n,m,k: integer;
begin
 if SS1=10 then
   begin
    n:=StrToInt(Edit1.Text);
    Edit1.text:=FromDec(n,16);
```

```
 end
else if SS1=2 then
 begin
  k:=0;
  s:=Edit1.Text;
  n:=Length(s);
  for m:=1 to n do begin
    k:=k*2;
    case s[m] of
      '0': k:=k+0;
      '1': k:=k+1;
    end;
  end;
  Edit1.Text:=IntToStr(k);
  n:=StrToInt(Edit1.Text);
 Edit1.text:=FromDec(n,16);
 end
else if SS1=8 then
 begin
 k:=0;
 s:=Edit1.Text;
 n:=Length(s);
```

```
  for m:=1 to n do begin
    k:=k*8;
    case s[m] of
      '0': k:=k+0;
      '1': k:=k+1;
      '2': k:=k+2;
      '3': k:=k+3;
      '4': k:=k+4;
      '5': k:=k+5;
      '6': k:=k+6;
      '7': k:=k+7;
    end;
  end;
  Edit1.Text:=IntToStr(k);
  n:=StrToInt(Edit1.Text);
  Edit1.text:=FromDec(n,16);
end;


  SS1:=16;
  Edit2.Text:=inttostr(SS1);
 Button1.Enabled:=true;
 Button2.Enabled:=true;
```

```
  Button3.Enabled:=true;
  Button4.Enabled:=true;
  Button5.Enabled:=true;
  Button6.Enabled:=true;
  Button7.Enabled:=true;
  Button8.Enabled:=true;
  Button9.Enabled:=true;
  Button0.Enabled:=true;
  Button10.Enabled:=true;
  Button11.Enabled:=true;
  Button12.Enabled:=true;
  Button13.Enabled:=true;
  Button14.Enabled:=true;
  Button15.Enabled:=true;
  end;
procedure TForm2.RadioButton3Change(Sender: TObject);
var s: string;
    n,m,k: integer;
begin
 if SS1=10 then
   begin
    n:=StrToInt(Edit1.Text);
```

```
  Edit1.text:=FromDec(n,8);
 end
else if SS1=2 then
 begin
  k:=0;
  s:=Edit1.Text;
  n:=Length(s);
  for m:=1 to n do begin
    k:=k*2;
    case s[m] of
      '0': k:=k+0;
      '1': k:=k+1;
    end;
  end;
  Edit1.Text:=IntToStr(k);
  n:=StrToInt(Edit1.Text);
 Edit1.text:=FromDec(n,8);
 end
else if SS1=16 then
 begin
 k:=0;
 s:=Edit1.Text;
```

```
n:=Length(s);
for m:=1 to n do begin
  k:=k*16;
  case s[m] of
    '0': k:=k+0;
    '1': k:=k+1;
    '2': k:=k+2;
    '3': k:=k+3;
    '4': k:=k+4;
    '5': k:=k+5;
    '6': k:=k+6;
    '7': k:=k+7;
    '8': k:=k+8;
    '9': k:=k+9;
    'A': k:=k+10;
    'B': k:=k+11;
    'C': k:=k+12;
    'D': k:=k+13;
    'E': k:=k+14;
    'F': k:=k+15;
  end;
end;
```

```
  Edit1.Text:=IntToStr(k);
  n:=StrToInt(Edit1.Text);
  Edit1.text:=FromDec(n,8);
end;
  SS1:=8;
  Edit2.Text:=inttostr(SS1);
  Button1.Enabled:=true;
  Button2.Enabled:=true;
  Button3.Enabled:=true;
  Button4.Enabled:=true;
  Button5.Enabled:=true;
  Button6.Enabled:=true;
  Button7.Enabled:=true;
  Button8.Enabled:=false;
  Button9.Enabled:=false;
  Button0.Enabled:=true;
  Button10.Enabled:=false;
  Button11.Enabled:=false;
  Button12.Enabled:=false;
  Button13.Enabled:=false;
  Button14.Enabled:=false;
  Button15.Enabled:=false;
```

```
end;
procedure TForm2.RadioButton4Change(Sender: TObject);
var s: string;
    n,m,k: integer;
begin
 if SS1=10 then
   begin
    n:=StrToInt(Edit1.Text);
    Edit1.text:=FromDec(n,2);
   end
 else if SS1=8 then
   begin
    k:=0;
    s:=Edit1.Text;
    n:=Length(s);
    for m:=1 to n do begin
      k:=k*8;
      case s[m] of
        '0': k:=k+0;
        '1': k:=k+1;
        '2': k:=k+2;
        '3': k:=k+3;
```

```
      '4': k:=k+4;
      '5': k:=k+5;
      '6': k:=k+6;
      '7': k:=k+7;
    end;
  end;
  Edit1.Text:=IntToStr(k);
  n:=StrToInt(Edit1.Text);
 Edit1.text:=FromDec(n,2);
 end
else if SS1=16 then
 begin
 k:=0;
 s:=Edit1.Text;
 n:=Length(s);
 for m:=1 to n do begin
  k:=k*16;
  case s[m] of
    '0': k:=k+0;
    '1': k:=k+1;
    '2': k:=k+2;
    '3': k:=k+3;
```

```
      '4': k:=k+4;
      '5': k:=k+5;
      '6': k:=k+6;
      '7': k:=k+7;
      '8': k:=k+8;
      '9': k:=k+9;
      'A': k:=k+10;
      'B': k:=k+11;
      'C': k:=k+12;
      'D': k:=k+13;
      'E': k:=k+14;
      'F': k:=k+15;
    end;
  end;
  Edit1.Text:=IntToStr(k);
  n:=StrToInt(Edit1.Text);
  Edit1.text:=FromDec(n,2);
end;
  SS1:=2;
  Edit2.Text:=inttostr(SS1);
  Button1.Enabled:=true;
  Button2.Enabled:=false;
```

```
  Button3.Enabled:=false;
  Button4.Enabled:=false;
  Button5.Enabled:=false;
  Button6.Enabled:=false;
  Button7.Enabled:=false;
  Button8.Enabled:=false;
  Button9.Enabled:=false;
  Button0.Enabled:=true;
  Button10.Enabled:=false;
  Button11.Enabled:=false;
  Button12.Enabled:=false;
  Button13.Enabled:=false;
  Button14.Enabled:=false;
  Button15.Enabled:=false;
end;
end.
```