ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

«МЕЖДУНАРОДНЫЙ КОЛЛЕДЖ «ПОЛИГЛОТ»

КУРСОВАЯ РАБОТА

Тема: Проектирование приложения для подсчёта электроэнергии

Студент: Гречкин Владислав Александрович

ФИО подпись

Специальность: Программирование в компьютерных системах

Группа 4 ПКС

Руководитель: Кубанова Джамиля Аубекировна

ФИО подпись

Черкесск

2020г.

Содержание

Введение 3

Глава 1. Постановка задачи 6

1.1. Что из себя представляет подсчет электроэнергии 6

1.2. Описание среды разработки 7

1.3. Панели инструментов Lazarus 9

Глава 2. Процесс создания программы 12

2.1. Описание блок схемы алгоритма. 12

2.2. Создание визуальной формы 16

2.3. Процесс построения функций 17

Заключение 19

Список источников и литературы 20

Приложения 22

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 модульного программирования.

Целью данной курсовой работы является проектирование приложения, для подсчёта электроэнергии и для достижения поставленной цели, необходимо решить следующие задачи;

- Создание визуальной модели;
- Описание алгебраических функций

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.
- Наследование - Когда класс определяется путем наследования существующего функционала родительского класса, то говорят, он

наследуется. В этом случае дочерний класс наследует все или несколько функций-членов и переменных родительского класса. Объекты могут также быть унаследованы.

- Родительский класса - класс, который передается по наследству другому классу. Он также называется базовым классом или суперклассом.
- Дочерний класс - класс, который наследуется от другого класса. Он также называется подклассом или производным классом.

Глава 1. Постановка задачи

В данной курсовой работе будет описан подробный процесс создания программы для подсчета электроэнергии и подсчет стоимости за период пользуясь базовыми знаниями объектно-ориентированного языка Object Pascal. Рассмотрим методы и атрибуты, которые будут использовать в процессе разработки и создания программы.

Основные цели курсовой работы;

- Знакомство с базовыми аспектами языка Object Pascal;
- Создание визуальной формы проекта;
- Проектирование алгоритма работы программы:

1.1. Что из себя представляет подсчет электроэнергии

Электричество давно играет важную роль в жизни современного человека, сопровождая его повсюду. Любой из нас пользуется лифтом, бытовой техникой, телефонами, банкоматами, компьютерами – все эти привычные каждому человеку вещи, облегчающие нашу жизнь, не способны функционировать без постоянного электроснабжения. При этом количество электроприборов вокруг нас не становится меньше, оно постоянно растет из года в год. Электрический свет, горячая вода, тепло, которые так необходимы нам всем тоже поступают к нам благодаря электроэнергии.

Делая свою жизнь все более комфортней, человек становится все более зависим от электричества. Любые отключения электроэнергии, пусть даже и кратковременные, имеют весьма негативные последствия. Особенно, когда речь заходит о загородном доме или дачных поселках. При этом не стоит

забывать про промышленность и социально значимые объекты, где наличие электроэнергии является необходимостью.

Так же как и все услуги электроснабжение имеет свою цену и свои нормативные нормы. Например, для удобства клиента энергетические компании взымают плату за электроснабжение исходя из показаний счётчика, но и бывают исключения, например, когда счетчик неисправен или же показывает неверное значение, так же бывают случаи, когда в многоквартирном доме стоит счетчик на всех жильцов, а бывает так, что его вообще нет. Для всех случаев предусмотрены нормативные формы оплаты. В данной курсовой работе мы рассмотрим первый вариант, самый популярный, когда оплата производится по счетчику.

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(Рисунок 1)

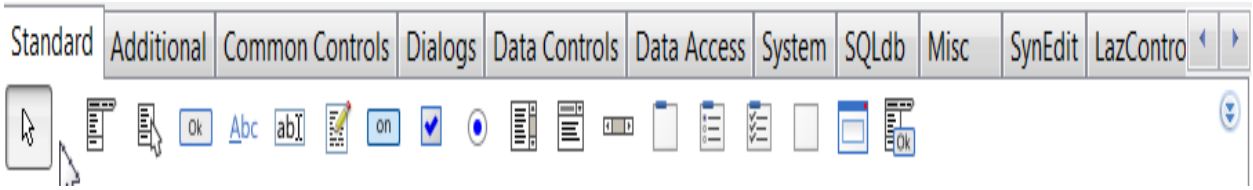


Рисунок 1 – Палитра объектов

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 (или нажатие правой кнопки мыши на компоненте) появляется подсказка – название компоненты.

1.3. Панели инструментов Lazarus

С помощью окна(рис.2) 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).

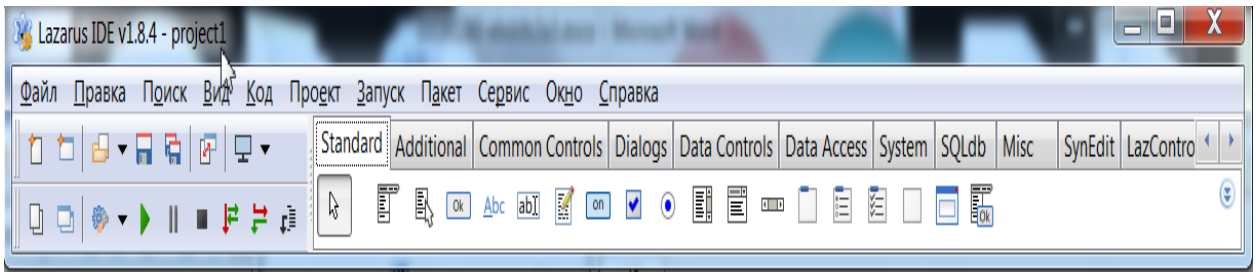


Рисунок 2 – Главное окно

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(рисунок 3).

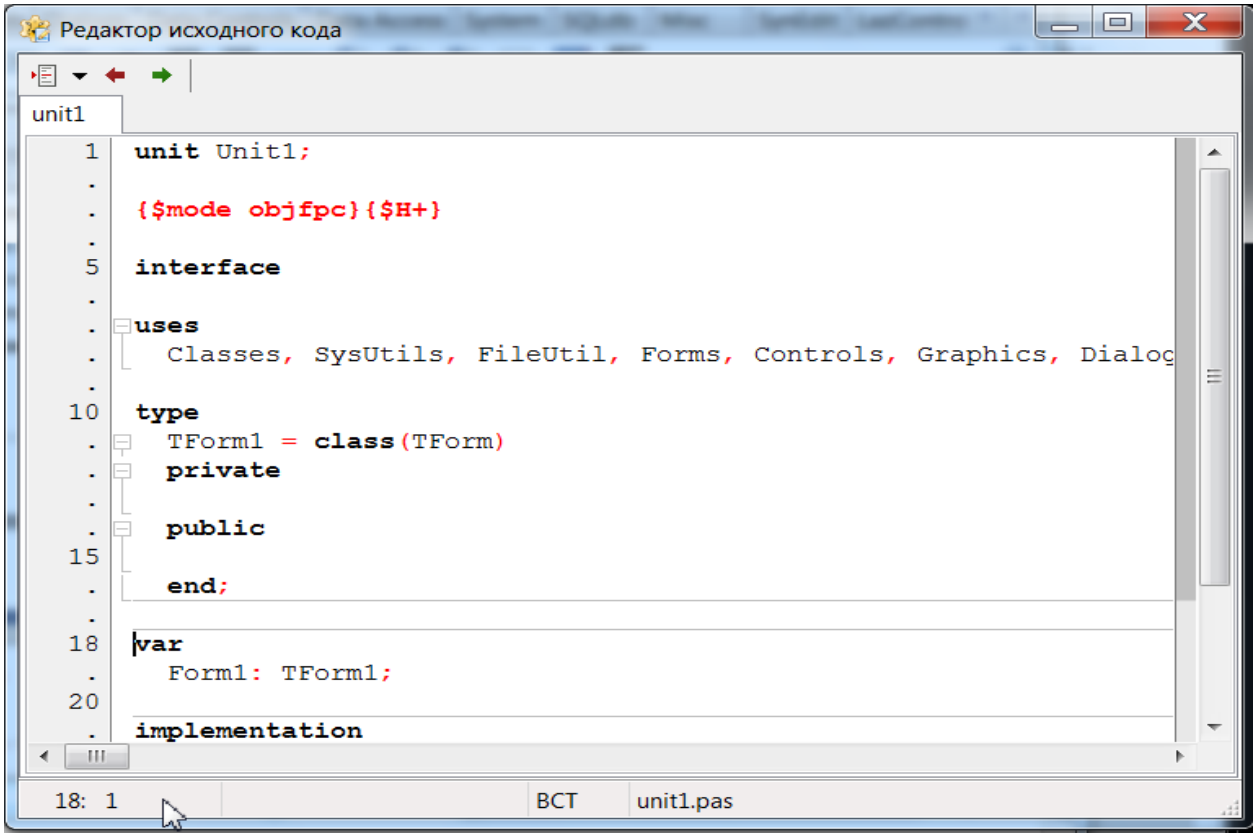


Рисунок 3 – Редактор исходного кода

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).

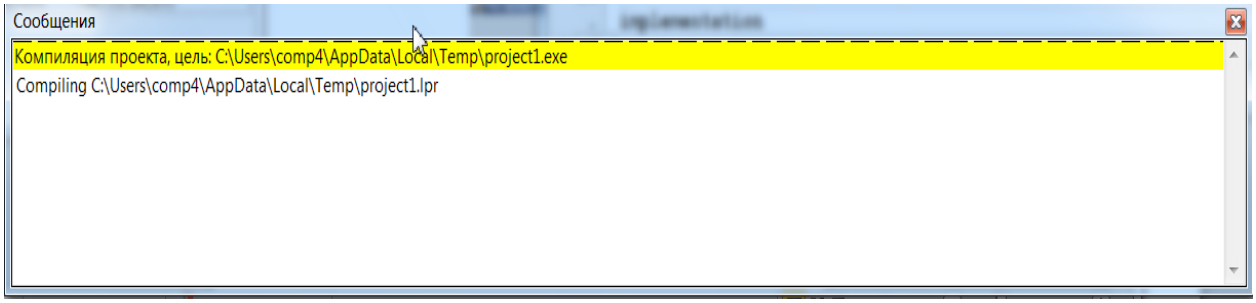


Рисунок 4 – Окно сообщений

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

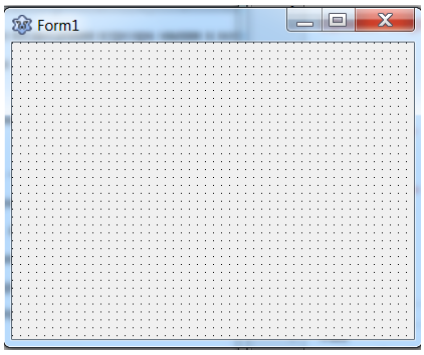


Рисунок 5 – Окно формы

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).

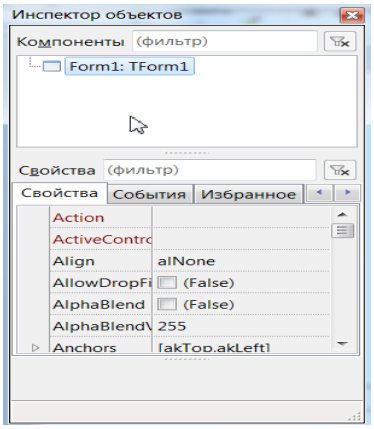


Рисунок 6 – Инспектор объектов

Глава 2. Процесс создания программы

В самом начале, чтобы приступить к созданию программы, первым шагом создадим алгоритм работы программы, для того чтобы наглядно видеть структуру программы и в будущем не запутаться, и не забыть о мелких деталях, но в данном случае начнем сразу с реализации, так как программа не требует использовать сложные объекты или классы.

Приложение будет состоять одной маленькой формы, на которой можно будет проводить вычисления, следовательно, в данной курсовой работе будет описан подробный процесс создания обеих этих форм.

2.1. Описание блок схемы алгоритма.

Алгоритм — это последовательность команд, предназначенная исполнителю, в результате выполнения которой он должен решить поставленную задачу. Алгоритм должен описываться на формальном языке, исключающем неоднозначность толкования.

Исполнитель — это человек, компьютер, автом. устройство и т. д. Исполнитель должен уметь выполнять все команды, составляющие алгоритм. Множество возможных команд конечно и изначально строго задано. Действия, которые выполняет исполнитель по этим командам называются элементарными. Запись алгоритма на формальном языке называется программой.

Алгоритм обладает следующими свойствами:

- Дискретность - процесс решения задачи должен быть разбит на последовательность отдельных шагов-команд, которые выполняются

одна за другой. Только после завершения одной команды начинается выполнение следующей.

- Понятность - алгоритм должен содержать только те команды, которые известны исполнителю.
- Детерминированность - каждый шаг и переход от шага к шагу должны быть точно определены, чтобы его мог выполнить любой другой человек или механическое устройство. У исполнителя нет возможности принимать самостоятельное решение (алгоритм исполняется ф
- Конечность - обычно предполагают, что алгоритм заканчивает работу за конечное число шагов. Результат работы алгоритма также должен быть получен за конечное время. Можно расширить понятие алгоритма до понятия процесса, который по различным каналам получает данные, выводит данные и потенциально может не заканчивать свою работу.
- Массовость - алгоритм должен решать не одну частную задачу, а класс задач. Не имеет смысла строить алгоритм нахождения наибольшего общего делителя только для чисел 10 и 15.
- При описании алгоритмов давно и успешно используются блок-схемы (Basic Flowchart). Построение блок-схем алгоритмов регламентируется ГОСТ 19.701-90 (ИСО 5807-85) «Единая система программной документации».
- Схемы алгоритмов программ, данных и систем. Условные обозначения и правила выполнения". Данный государственный стандарт составлен на основе международного стандарта "ISO 5807-85. Information processing - Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts".
- Согласно ГОСТ 19.701-90 под схемой понимается графическое представление определения, анализа или метода решения задачи. С помощью схем можно отобразить как статические, так и динамические

аспекты системы. Символы, приведенные в государственном стандарте, могут использоваться в следующих типах схем:

- Схемы данных - определяют последовательность обработки данных и их носители;
- Схемы программ - отображают последовательность операций в программе (по сути, это и есть блок-схемы алгоритмов в традиционном понимании);
- Схемы работы системы - отображают управление операциями и потоки данных в системе;
- Схемы взаимодействия программ - отображают путь активации программ (модулей) и их взаимодействие с соответствующими данными;
- Схемы ресурсов системы - отображают конфигурацию блоков данных и обрабатывающих блоков.

Как видно из приведенных выше типов схем, они могут использоваться не только для моделирования поведенческого аспекта, но и для задач функционального, информационного и компонентного проектирования.

При построении поведенческой модели системы используются основные принципы структурного подхода - принципы декомпозиции и иерархического упорядочения. Поведенческая модель представляет собой набор взаимосвязанных схем (диаграмм) с разным уровнем детализации, причем с каждым новым уровнем детализации система приобретает все более законченные очертания.

На схемах могут присутствовать следующие элементы графической нотации:

- Переменные-члены - Это переменные, определенные внутри класса или объекта.

- Символы данных - указывают на наличие данных, вид носителя или способ ввода-вывода данных;
- Символы процесса - указывают операции, которые следует выполнить над данными;
- Символы линий - указывают потоки данных между процессами и/или носителями данных, а также потоки управления между процессами;
- Специальные символы - используются для облегчения написания и чтения схем.

Кроме деления по смысловому содержанию, каждую категорию символов (кроме специальных) делят на основные и специфические символы. Основной символ используется в тех случаях, когда точный вид процесса или носителя данных неизвестен или отсутствует необходимость в описании фактического носителя данных (процесса). Специфический символ используется в тех случаях, когда известен точный вид процесса или носителя данных и это необходимо отобразить на схеме. В следующей таблице приводятся элементы графической нотации блок-схем.

Данной программе не используется сложных функций из-за чего алгоритм самого приложения будет линейным и будет состоять из пары блоков(рисунок 7).

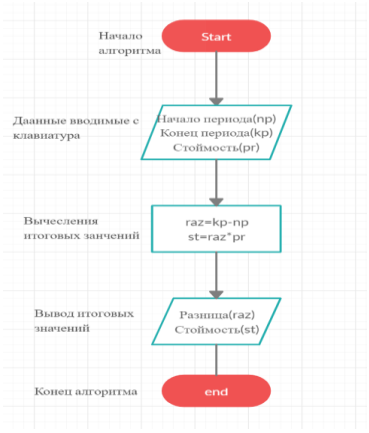


Рис. 7. «Алгоритм программы»

2.2. Создание визуальной формы

После создания проекта, автоматически создается одна форма и открывается так называемое «Окно формы» и на нём будем расставлять наши объекты. Для начала создадим несколько элементов «TEdit», которые позволя нам записывать данные и после переводить их в машинных код, после всех вычислений получать данные обратно(рисунок 8).

Рис. 8. «TEdit»

Так же, для того чтобы не запутаться создадим элементы «TLabel», над каждой строчкой, что позволит нам уточнить какая стока, за что отвечает. После создания элемента, меняем его свойство «Text» на нужное нам.

Далее для того, чтобы программа была полноценной осталось добавить кнопку, которая будет отвечать за выполнение функции. При помощи свойства кнопки «Caption» вписываем нужное нам слово, после чего появляется на форме внутри самого объекта(рисунок 9).

Рис. 9. «Tbutton»

2.3. Процесс построения функций

Так как в данном приложении присутствует только одна кнопка, нужно ей присвоить определенную функцию, чтобы научить эту кнопку выполнять определенные действия.

Для начала опишем переменные, которые будем использовать, а так же опишем их тип(рисунок 10).

```
var np,kp,raz:integer;
  pr,st:extended;
```

Рис. 10. «Описание переменных»

В данном случае, чтобы не путать опишем каждую переменную в форме сокращения, то есть «np» – это переменная описывающая начало периода, а «kp» – это конец периода и так далее.

Далее следует научить программу распознавать информацию, которую мы ей преподносим, если говорить точнее, то следует присвоить к кнопке команду при которой она будет считывать информацию с нужных нам строк.

Для этого воспользуемся командой «strtoint», которая позволит нам присваивать значения к переменным используя лишь одну кнопку управления. У команды имеются особенности зависящие от типа переменной, то есть, если мы используем переменную типа «integer», то команда будет выглядеть прежней, но если тип переменной изменится, то заним и поменятся команда, например, если используем тип «extended» то и команда будет изменена на более общий формат «strtofloat»(рис.11).

```
np:=strtoint(Edit1.text);
kp:=strtoint(Edit2.text);
pr:=strtofloat(Edit3.text);
```

Рис. 11. «Перевод данных в машинный код»

Следующим нашим шагом будет выяснение конечных значений. Так как количество потраченной электроэнергии и её стоимость узнают путем вычисления разницы между началом периода и его концом, а после перемножения этой разницы на стоимость за один киловатт, будем использовать простые алгебраические функции сложения и умножения(рисунок 12).

```
raz:=kp-np;
st:=raz*pr;
```

Рис. 12. «Вычисление конечных значений»

И последний штрих, присвоение кнопке функции, которая будет выводить итоговые значения. Делать это будем, используя все ту же команду «strtoint», вот только нужно ее изменить так как данный ее вид позволяет компьютеру читать введенные нами данные, следовательно просто перевернем команду, чтобы получилось «inttostr» для переменных типа «integer» и соответственно, «floattostr» для переменных типа «extenden». По такому же принципу меняется местами переменные, если в прошлый раз переменные стоят в начале, то теперь они будут описываться в конце, так как формально мы присваиваем значение внутрь самих строк(рисунок 13).

```
Edit4.text:=inttostr(raz);
Edit5.Text:=floattostr(st);
```

Рис. 13. «Вывод конечных значений»

Заключение

В ходе выполнения курсовой работы было разработано приложение для подсчёта электроэнергии. 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 визуальной формы проекта;
- Проектирование алгоритма работы программы:
- Создан рабочий функционал программы;

Данный формат разработки приложения помогает разграничить необходимую информацию 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) ;

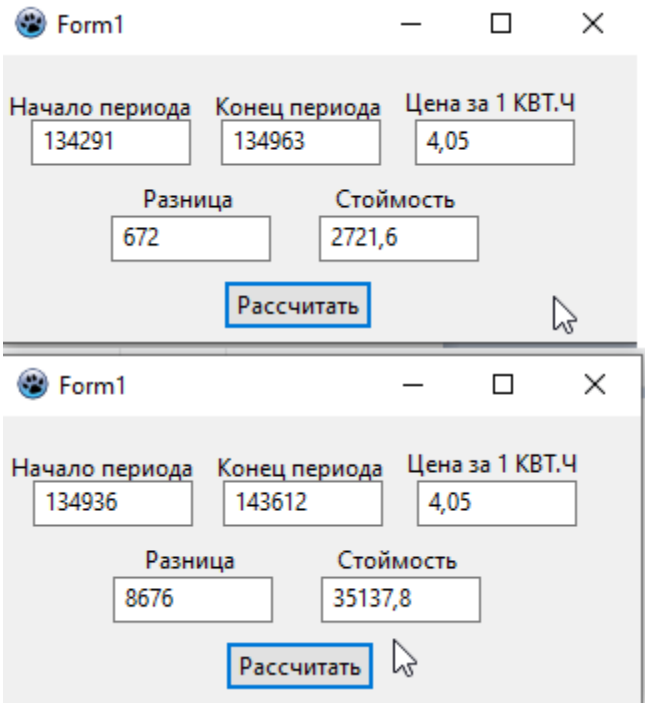
18.https://ru.wikipedia.org/wiki/Free_Pascal;

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 программы

```
unit Unit1;

{$mode objfpc}{$H+}

interface

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

type
  { TForm1 }


  TForm1 = class(TForm)
```

```
    Button1: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.lfm}
{ TForm1 }
```

```
procedure TForm1.Button1Click(Sender: TObject);
var np,kp,raz:integer;
  pr,st:extended;
begin
np:=strtoint(Edit1.text);
kp:=strtoint(Edit2.text);
pr:=strtofloat(Edit3.text);
raz:=kp-np;
st:=raz*pr;
Edit4.text:=inttostr(raz);
Edit5.Text:=floattostr(st);
end;
end.
```