МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТРАНСПОРТА»

Кафедра изысканий и проектирования дорог

А. К. ГОЛОВНИЧ

АВТОМАТИЗАЦИЯ РАСЧЕТОВ И ПРОЕКТИРОВАНИЯ ЖЕЛЕЗНЫХ ДОРОГ

Гомель 2010

УДК 625.1.001.2 (075.8)
ББК 39.20
Г61

Р е ц е н з е н т ы: начальник Брестского информационно-вычислительного центра Белорусской железной дороги *В. А. Падалица*; заведующий кафедрой «Информационные технологии» канд. экон. наук, доцент *Л. А. Иоффе* (УО «БелГУТ»).

Головнич, А. К.

Г61 Автоматизация расчетов и проектирования железных дорог : учеб.-метод. пособие по дисциплине «Автоматизированное проектирование дорог» / А. К. Головнич ; М-во образования Респ. Беларусь, Белорус. гос. ун-т трансп. – Гомель : БелГУТ, 2010. – 145 с.

ISBN 978-985-468-709-4

Рассмотрены теоретические и практические вопросы комплексного использования пакета визуального программирования Visual Basic и автоматизированного проектирования AutoCAD для решения задач расчета и проектирования элементов железных дорог. Представлен анализ функциональных возможностей данных программных продуктов и особенностей их применения для решения конкретных задач расчетного и проектного характера на железнодорожном транспорте.

Предназначено для студентов дневной и безотрывной форм обучения по специальности «Строительство железных дорог, путь и путевое хозяйство».

УДК 625.1.001.2(075.8)
ББК 39.20

ISBN 978-985-468-709-4

ВВЕДЕНИЕ

Проектирование – наиболее сложная сфера человеческой деятельности. Практически любой объект искусственного происхождения проходит этап проектирования – создания бумажного прототипа его существования. Опыт человеческой цивилизации доказывает нам необходимость такого этапа, который является началом жизнедеятельности любого продукта работы интеллекта. Проектировщик дает путевку в жизнь идее, возникшей как потребность в постоянном развитии, которое, в свою очередь, является необходимым условием благополучного существования всего общества.

Сложность проектирования представляется как объективный процесс всестороннего анализа перспектив и возможностей будущего объекта, необходимости учета влияния на его функционирование ряда позитивных и негативных факторов. Это ответственная и кропотливая работа, требующая высокой профессиональной подготовки и использования всего арсенала средств, обеспечивающего в конечном итоге качество результатов проектирования. Однако следует признать, что существенное увеличение производительности труда не является показательным для процесса проектирования. Если за последние 100 лет производительность труда в различных отраслях народного хозяйства возросла от 100 до 1000 %, то в области проектирования – только на 15–20 %. Результаты исследований американской аналитической компании Gardner Group показывают, что качество проектирования имеет принципиальное значение для всего последующего жизненного цикла объекта. Если на этапе проектирования замечена некоторая ошибка, стоимость устранения которой равна 1 доллару, то в случае ее обнаружения после сооружения данного объекта стоимость ее ликвидации оценивается уже в 100000 долларов.

Железнодорожный транспорт привносит свои особенности в процесс проектирования. Все проекты по сооружению новых участков железных дорог и станций, реконструкции существующих перегонов и раздельных пунктов уникальны. Ни один уже исполненный проект не повторится в будущем. Всегда находятся принципиальные различия в топографии, геологии, экономике, которые требуют оригинальных методов разработки и

принятия проектных решений. В таких условиях подготовка проектов должна проводиться с активным применением всех средств, способствующих высокой эффективности процесса проектирования и качества полученного результата. В настоящее время таким средством являются информационные технологии, выступающие в виде высокопроизводительного программного обеспечения систем автоматизированного проектирования (САПР) и технического обеспечения этапов подготовки проектов (электронные геодезические приборы для проведения полевых измерений, сканеры, принтеры, плоттеры, устройства функционального прототипирования). К сожалению, современный уровень САПР и ТОПР (технического обеспечения проектов) недостаточно высок для комплексного решения всех проблем связной технологической цепочки «задание на проект – инженерно-геодезические и геологические изыскания – разработка вариантов развития транспортного объекта – анализ вариантов – выбор проектного решения – выдача технической документации». Единая технология подготовки проектов – пока лишь перспектива дальнейшего развития САПР. В настоящее время различные программные средства позволяют достаточно эффективно решать задачи расчетного характера, формировать исходную базу данных для последующего графического моделирования. Кроме этого имеются продуктивные графические оболочки, способные при незначительных затратах времени решать задачи проектного характера и получать приемлемые визуализируемые результаты, близкие к требованиям технической документации.

Использование информационных технологий в теории и практике проектирования невозможно без изучения основ алгоритмизации и программной разработки основных расчетных процедур и функций. Поэтому важно ознакомиться с соответствующим инструментарием, который несложен в изучении, но весьма эффективен при использовании.

Разработка проектных решений по реконструкции путевого развития станций и перегонов связывается с решением многочисленных взаимоувязанных задач расчетного и проектного характера. Их сложность и необходимость использования опыта проектировщика приводят к тому, что попытка применения информационных технологий и компьютерных методов решения разбивает целостную проблему на ряд конкретных, слабо зависящих друг от друга задач. Существует такое понятие – диссипация связей, которое связывают с разрывом технологической цепочки в системе поиска эффективных проектных решений.

В общей постановке задача проектирования состоит из целого ряда расчетных, проектных, логических, эвристических фрагментов, связанных между собой исходными и результирующими данными. Однако методы решения этих отдельных фрагментов различны, поэтому при использовании компьютерных технологий целостная проектная задача распадается на

несколько модулей. Наиболее крупными из таких модулей являются расчетная и проектная части. Расчетная часть переведена в соответствующие алгоритмические формы достаточно просто. Проектная часть специфична, требует использования особых приемов и методов, навыков работы в графической среде автоматизированного проектирования. Однако при этом следует не терять связей с результатами расчетного этапа. Более того, результаты компьютерных расчетов должны быть исходными данными для этапа компьютерного проектирования. И перенос значений параметров из одного этапа в другой должен осуществляться автоматически, без всякого участия пользователя. Это условие выполнить непросто, учитывая различия в средствах реализации расчетов и проектирования.

Решение этой сложной задачи лежит в плоскости использования интегрированной программной среды, объединяющей в себе расчетные и проектные процедуры и функции.

Предлагаемое пособие представляет методические основы автоматизированного подхода системного проведения расчетов и проектирования элементов железных дорог. Особенностью решения данной задачи является целостность и преемственность операций определения основных параметров плана и профиля и последующей разработки его геометрической структуры. Реализация стандартных методик с применением информационных технологий позволит получить практические навыки программирования расчетных процедур с использованием среды Visual Basic и на основе полученных результатов выполнить проектную работу в среде AutoCAD.

Для проектировщика все эти операции представляются как связный комплекс отдельных видов расчетных и проектных работ. Данное методическое пособие преследует цели получения навыков разработки подобных интегрированных программ и их практического использования. Методики могут быть применимы для решения программно-ориентированных задач расчетно-проектного характера в других отраслях знаний.

Предлагаемые методики используются в процессе изучения дисциплины «Автоматизированное проектирование дорог». Соответствующая рабочая программа по данной дисциплине приведена в приложении А пособия.

1 РАЗРАБОТКА ИНТЕРФЕЙСА ПРОГРАММЫ РАСЧЕТА ЭЛЕМЕНТОВ ПЛАНА ЛИНИИ С ИСПОЛЬЗОВАНИЕМ ВОЗМОЖНОСТЕЙ СРЕДЫ VISUAL BASIC

1.1 Сведения из теории

Visual Basic – мощная среда разработки программ различного уровня сложности, которая основана на объектном подходе. Удобство использования данной среды заключается в том, что начальным этапом разработки любой программы является не программирование, а проектирование внешнего вида, схемы взаимного расположения отдельных конструктивных элементов (интерфейса). Разработчику необходимо определить состав объектов и их взаимное расположение на экране дисплея. Эти операции в значительной степени автоматизированы, так как Visual Basic включает обширный инструментарий, который позволяет формировать программно завершенные графические объекты.

Использование различных графических шаблонов значительно сокращает время выбора формы эффективного визуального представления, места расположения отдельных программных компонентов. Часто решение конкретной задачи автоматизированным способом определяется не только алгоритмом ее решения, но и характером представления на экране дисплея.

В последние годы большое значение придается информативности окна программы, искусству программиста раскрыть содержательную сторону решаемой задачи, объединяя общие характеристики, но при этом оставляя пользователю возможность быстро понять структуру информации, не заставляя его читать громоздкие инструкции, сложные пояснения и контекстные подсказки. Ориентация программы на конечного пользователя, наиболее полный учет его запросов и потребностей – залог востребованности программного продукта у массового потребителя. Формирование отдельных программных окон как расчетных и проектных процедур, определенных проектировщиками в качестве отдельных этапов разработки проекта, позволяет моделировать творческий процесс достижения некоторого результата, приблизить работу системы

автоматизированного проектирования к реальным условиям проектной работы.

Преимущество визуальных сред программирования заключается в том, что на стадии разработки программы можно заниматься только конструированием ее внешнего вида. Сначала следует решать конкретную задачу, и для проектировщика становится важным предметная среда, а не навыки использования языка программирования. Важно определить форму, внешний вид, который помог бы компактно и одновременно наглядно охватить все основные моменты расчета и проектирования сложной, комплексной задачи. Нельзя забывать, что мы имеем дело со специфическими транспортными проектными проблемами, для решения которых необходимо использовать достаточно широкий диапазон исходных данных топографического, климатического, экономического, экологического, социального характера. Поэтому приоритетной становится выбор программной среды представления исходных, расчетных данных и результатов проектирования.

Важнейшей особенностью визуальных сред программирования, к числу которых принадлежит Visual Basic, является минимальный объем программного кода, необходимого для формирования работоспособного приложения. На рисунке 1.1 представлен интерфейс подобной программы, разработанный на уровне проектирования ее внешнего вида без единой строки кода.

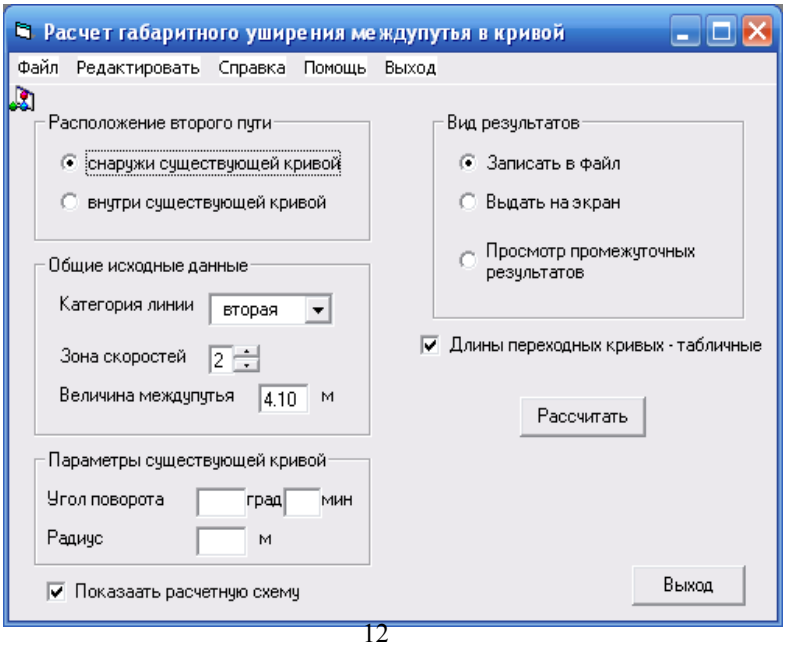


Рисунок 1.1 – Интерфейс программы расчета габаритного уширения пути

Анализ внешнего вида данной программы показывает, что все параметры, используемые в методике расчета габаритного уширения пути, входят в состав отдельных блоков, определяющих расположение второго пути, параметры существующей кривой, вид выдаваемого результата и др. В данном визуальном представлении программы выделено 4–5 блоков, в каждом из которых по 2–5 параметров. Наглядно можно оценить полученную визуальную схему как приемлемую, обеспечивающую достаточно полный охват и восприятие параметров, распределенных и сгруппированных по отдельных признакам. Большее количество приведет к перегрузке информационных форм.

С другой стороны, использование подобных программ не ограничивает число последовательно вложенных друг в друга информационных окон, вызываемых нажатием различных функциональных клавиш и графических кнопок. Например, реализация программы рисунка 1.1 способна активизировать вызов других окон при нажатии кнопок «Рассчитать», «Выход» и отдельных опций меню «Файл», «Редактировать», «Справка», «Помощь».

Возможности визуальной среды Visual Basic весьма обширны. Для их практического использования необходимо кратко ознакомиться с ресурсами программного комплекса, обеспечивающих решение прикладных задач. После загрузки Visual Basic и активизации Standard EXE опции «New» (рисунок 1.2) получаем исходный основной экран, обеспечивающий разработку прикладной программы (рисунок 1.3).

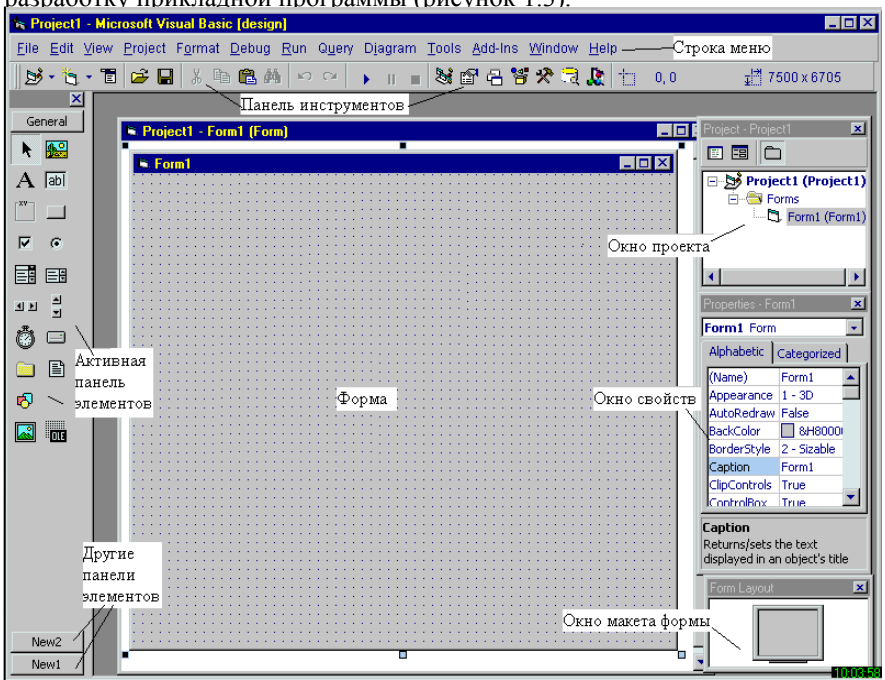


Рисунок 1.2 – Окно загрузки среды Visual Basic

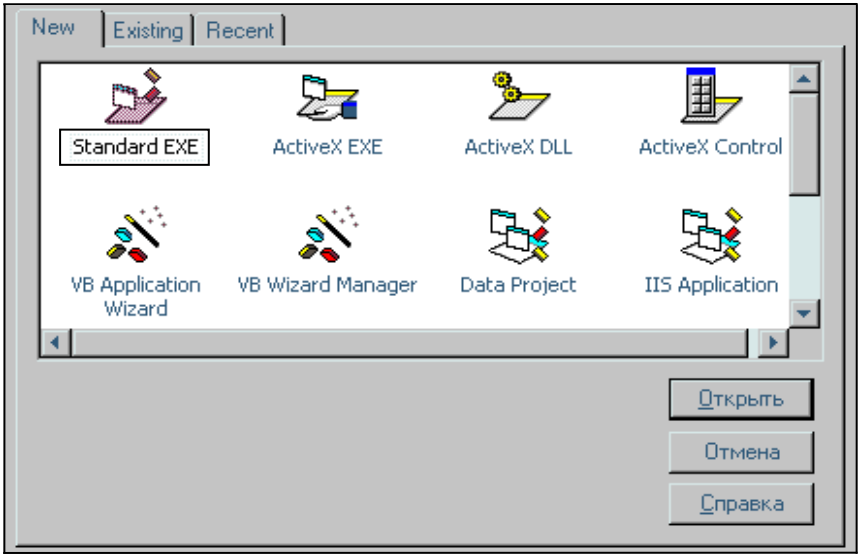


Рисунок 1.3 – Исходный экран объектов среды Visual Basic

Все операции по конструированию программы производятся на части рабочего поля экрана дисплея, который называется формой. С помощью простых манипуляций с перетягиванием края формы указателем курсора и нажатой левой клавишей мыши можно изменять размеры формы. Соотношение между размерами формы и размерами рабочего поля экрана отображается в окне макета формы. Программа может включать не одно, а несколько форм, и вся структура визуально отображается в окне проекта. Окно свойств определяет перечень характеристик, которыми обладает выделенный элемент. Для рисунка 1.3 представлены свойства основной формы, однако элементами могут быть и другие объекты, которые входят в состав активной панели элементов. По существу, эти элементы и являются конструктивными блоками, с помощью которых формируется внешний вид любой прикладной программы.

Данные объекты называют элементами управления, так как посредством их активизации (нажатием левой клавиши мыши при указании курсора на данном элементе) можно управлять программой. В состав данной панели включено 20 элементов (в общем случае их может быть произвольное количество). Для переноса необходимого элемента на форму необходимо щелкнуть на его изображении левой клавишей мыши и повторно нажать левую клавишу мыши в точке установки элемента на форме.

Представим краткую характеристику элементов управления:

 – знак отмены выбора элемента на панели. Отмену выбора можно осуществить также указанием курсора на другой элемент;

 – окно с рисунком (PictureBox) отображает на экране рисунки, загруженные из графических файлов, и позволяет рисовать на своей поверхности с помощью графических методов. Этот элемент можно использовать для объединения других элементов;

 – объект «метка» (Label), позволяющий в выделенном окне формы написать с помощью клавиатуры текст, который при работе программы нельзя изменить;

 – текстовое поле (TextBox) является элементом управления, предназначенным для ввода данных;

 – элемент управления «рамка» (Frame), объединяющий в группу несколько других элементов управления (см. на рисунке 1.1 рамки «Расположение второго пути», «Общие исходные данные», «Параметры существующей кривой», «Вид результатов»). Важной особенностью данного элемента является связанное положение находящихся на ней объектов. Перемещение рамки по форме приводит к изменению положения всех расположенных на ней других элементов. Однако для фиксирования выбранных объектов на данной рамке необходимо после выбора из панели элементов при нажатой левой клавиши мыши очертить поле на рамке, которое укажет место и размеры данного элемента. В противном случае элемент будет располагаться не на рамке, а на форме. Способность рамки объединять ряд элементов переводит ее в категорию элементов-контейнеров;

 – кнопка (CommandButton) – элемент, который используется для выполнения какого-либо процесса (см. на рисунке 1.1 кнопки «Рассчитать», «Выход»);

 – «флажок» (CheckBox), указывающий наличие или отсутствие действия, определяемого расположенным рядом текстом. Флажок может быть в активном, неотмеченном и неопределенном (недоступном) состояниях. На рисунке 1.1 флажок, определяющий табличные длины переходных кривых, находится в активном состоянии (т. е. утверждается, что эти кривые не следует

рассчитывать). Если необходимо выключить действие флажков, то достаточно щелчком левой клавиши мыши снять указатель соответствующего флажка;

 – переключатель (OptionButton) – элемент управления, предназначенный для установки или исключения функции, указанной в поясняющем тексте. Обычно несколько переключателей помещаются на одну рамку;

 – поле со списком (ComboBox), представляющее собой комбинацию двух элементов управления – списка с определенными значениями и отдельного (верхнего) поля для возможного ввода некоторого текста. На рисунке 1.1 категория линии определяется таким полем со списком возможных значений. При щелчке по указанной стрелке раскрывается весь список;

 – список (ListBox) позволяет пользователю выбирать из списка один или несколько элементов. В данный список можно добавлять новые элементы или удалять существующие. Если не все элементы могут одновременно отобразиться в поле списка, то автоматически отображаются полосы прокрутки;

 – горизонтальная (HScrollBar) и вертикальная (VScrollBar) полосы прокрутки, позволяющие отображать и вводить числовые значения только с помощью указателя мыши. Зона скоростей на рисунке 1.1 определяется с помощью такой полосы прокрутки;

 – таймер (Timer) – элемент, генерирующий события через заданные промежутки времени;

 – список устройств (DriveListBox), относящийся к группе элементов управления, предназначенных для отображения и работы с дисками, каталогами и файлами. DriveListBox служит для вывода списка всех доступных дисков и устройств системы и обеспечивает возможность их выбора;

 – список каталогов (DirListBox или DirectoryListBox) элемент управления, предназначенный для выбора файлов. Он отображает структуру выбранного диска и позволяет осуществлять выбор и смену каталога;

 – список файлов (FileListBox) – элемент управления, который можно использовать для выбора файлов. Он отображает файлы текущего каталога;

 – объект управления "фигура" (Shape), который может принимать вид определённой фигуры и служит для представления на форме геометрических фигур;

 – объект "линия" (Line) служит для отображения на форме линии;

 – графический образ (Image) выводит на экране рисунок. Image использует меньше ресурсов и поэтому быстрее, чем PictureBox,

однако в отличие от последнего является объектом-контейнером;

– OLE-контейнер, который способен объединять на программном уровне связываемые и внедряемые объекты.

1.2 Примеры использования элементов управления Visual Basic

Окно с рисунком (PictureBox) является удобным инструментом визуализации расчетных схем станций и их фрагментов, различных этапов проведения расчетов, вариантов проектных решений. Важным элементом является графический образ Image, который позволяет наглядно представлять рисунки, схемы, чертежи и т. д. Без данных элементов управления расчетно-проектные задачи выполнить затруднительно. В учебных целях графические элементы управления помогут студентам самостоятельно понять стандартные методики расчета и проектирования. На рисунке 1.4 представлен вариант обучающей программы, иллюстрирующей методику расчета габаритного уширения пути у кривых способом Д. Г. Голованова.

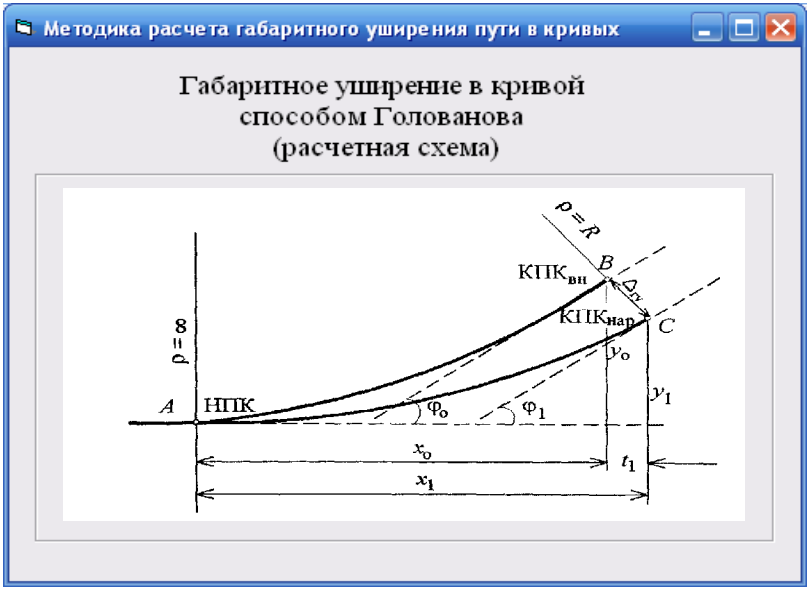


Рисунок 1.4 – Расчетная схема на программной форме с элементом PictureBox

Объект Label позволяет оформить все текстовые сопровождения программной формы. Принципиально можно всю программную форму заполнить только данными метками, которые тем не менее будут

достаточно информативными и необходимыми. Пример такого программного окна приведен на рисунке 1.5.

Рисунок 1.5 – Использование объекта Label
при конструировании интерфейса программы

Следует обратить внимание на то, что текстовые строки данного элемента управления могут растягиваться на всю длину информационного окна текущей программной формы. Так, строка указания величины руководящего уклона содержит 10 числовых характеристик (6–15), а каждая из строк указания массы состава включает 11 элементов (вместе с типом локомотива). Для удобства оформления таблицы использован также элемент Line, обеспечивающий разбиение позиций на фиксированные по длине строки и столбцы.

Для указания пользователем величин параметров применяется элемент управления «Текстовое поле» (TextBox) (рисунок 1.6).

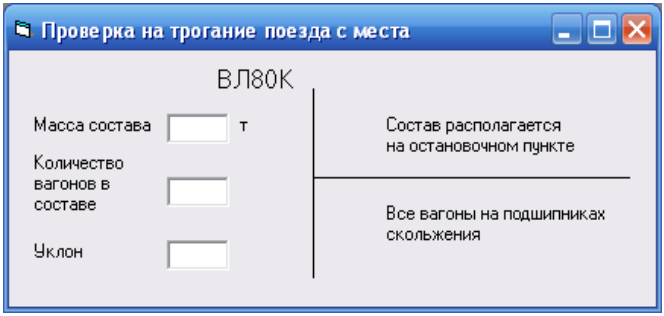


Рисунок 1.6 – Выделенные поля для ввода данных

Целесообразно выделять текстовые поля, фиксируя их положение по одной линии строки или столбца. Это может быть удобно для визуального контроля полноты и правильности заполнения всех необходимых полей перед проведением различных расчетов. Текстовое поле нужно указывать таким, чтобы все цифры величины соответствующего параметра были видимыми на выделенном сегменте. Поэтому при построении интерфейса с применением текстовых полей важно определить количество позиций

символом, которые вводятся пользователем в некоторую область программного окна.

Элемент управления «Рамка» (Frame) позволяет формировать структурно завершенные информационные объекты, объединяя несколько элементов управления в одну связную композицию. Например, таблица рисунка 1.5 с использованием рамки может выглядеть так, как показано на рисунке 1.7.

По возможности на поле программного окна следует компоновать рамочные объекты, объединяемые между собой по какому-либо заданному признаку.

Кнопка (CommonButton) позволяет передавать управление некоторому процессу (визуализация, расчет, запись, выход). Если имеется ряд подобных кнопок, то целесообразно компоновать их на одной программной рамке (рисунок 1.8).

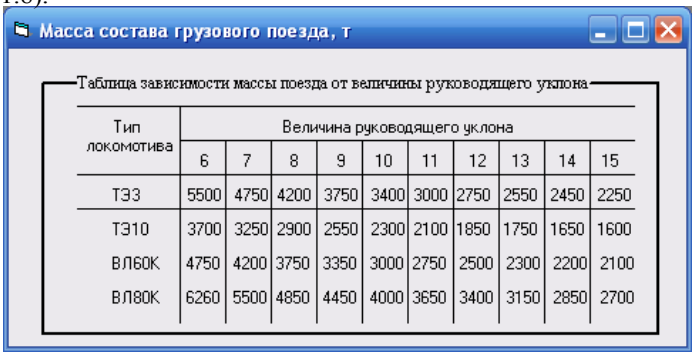


Рисунок 1.7 – Структурно обособленный объект таблицы

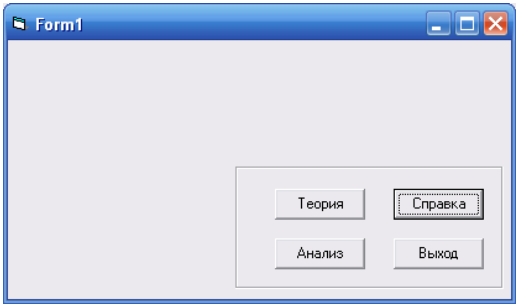


Рисунок 1.8 – Связная позиция кнопок
на программной форме

При наличии на одной форме 5 и более кнопок необходимо изменить внешний вид программного окна, перемещая менее важные и

необязательные кнопки на другие вложенные окна или в меню. Однако могут быть случаи, когда большое количество кнопок оправдывается удобством быстрого вывода промежуточных результатов, необходимостью визуального вывода величин расчетных параметров по шагу изменения исходных данных.

Элемент управления «Флажок» (CheckBox) обеспечивает выбор условия проведения дальнейших расчетов. Например, интерфейс рисунка 1.6 существенно можно изменить в лучшую сторону, применяя данный программный инструмент (рисунок 1.9).

При неактивных флажках предполагается, что состав поезда находится на перегоне, а все вагоны в данном составе имеют подшипники качения.

Переключатель (OptionButton) позволяет осуществлять выбор из ряда возможных условий. Например, предыдущий интерфейс программы с использованием переключателей приведен на рисунке 1.10.

Список и поле со списком часто используются при необходимости выбора некоторой конкретной позиции из перечня наименований параметров или их свойств. Различие этих элементов управления заключается в том, что список раскрывается полностью с выделением указываемого поля некоторым цветом, отличным от фона. Поле со списком имеет похожий графический вид, однако после выбора конкретной позиции список сворачивается, а в программном окне остается только строка с активной позицией и указателем в правой части элемента, позволяющем раскрыть список полностью.

Примеры использованием данных элементов управления приведены на рисунке 1.11.

Вертикальная и горизонтальная полосы прокрутки являются удобным инструментом при установке величин параметров в текстовом окне с выбором из некоторого ряда (рисунок 1.12, *а*) и чтении длинных строк в текстовом поле ограниченной длины (рисунок 1.12, *б*).

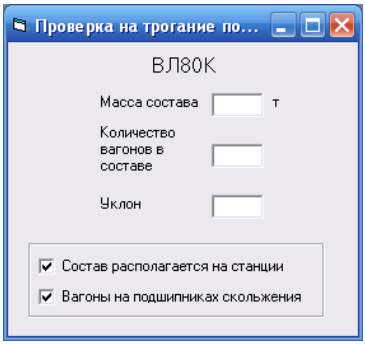


Рисунок 1.9 – Фиксирование данных на флажках

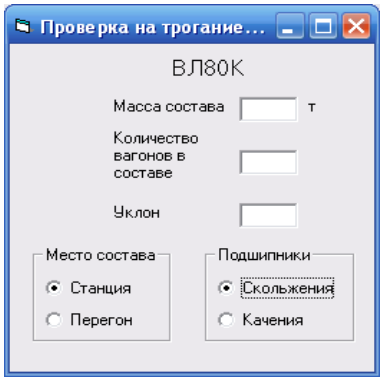


Рисунок 1.10 – Использование переключателей

для выбора условий

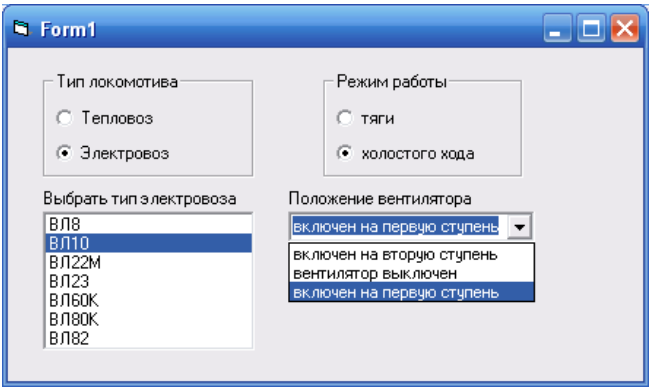
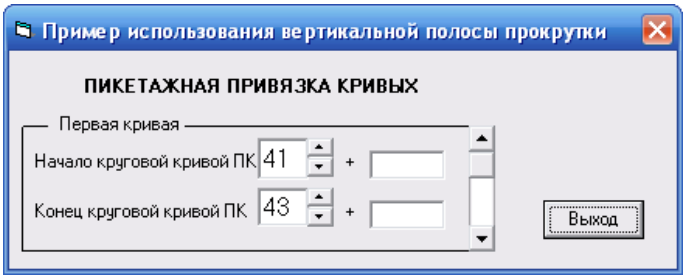


Рисунок 1.11 – Применение списка и поля со списком

а)



б)

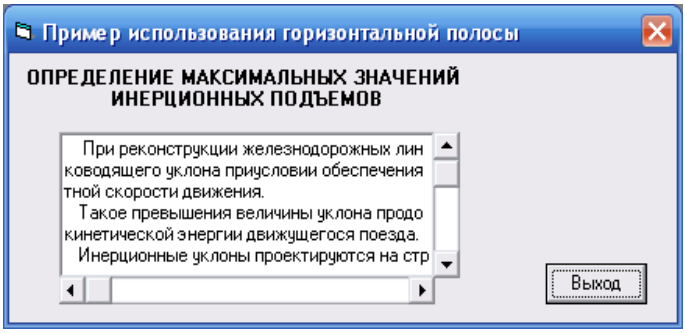


Рисунок 1.12 – Примеры использования полос прокрутки:
а – вертикальной; *б* – горизонтальной и вертикальной

Вертикальные полосы прокрутки используются при переборе значений пикетной привязки. Пользователь имеет возможность выбора фиксированных значений, что исключает задания неверного параметра.

Часто набор разрешенных состояний в текстовом поле закольцовывают, т. е. после попытки установки следующего за максимальным значением в текстовом поле появляется минимальное значение ряда.

Пример рисунка 1.12, *а* интересен двойным применением вертикальной полосы прокрутки. Кроме выбора значений пикетов начала и конца круговой кривой можно перемещаться по различным характеристикам кривых. При активизации внешней вертикальной полосы прокрутки происходит визуализация фреймов следующих кривых.

Рисунок 1.12, *б* иллюстрирует способ применения полос прокрутки при чтении текста из окна фиксированного размера, меньшего формата записи самого текста.

1.3 Задание свойств элементов управления

Загрузка среды Visual Basic позволяет увидеть в правой половине проекта (см. рисунок 1.2) окно свойств элементов управления. При переносе из активной панели конкретного элемента на программную форму в окне свойств формируется соответствующий список всех его свойств. Выбор соответствующих позиций из набора свойств приводит к визуальному изменению формы, места размещения, вида элементов управления. Например, свойства программной формы (рисунок 1.13) позволяют создавать интерфейс программы без разработки программного кода.

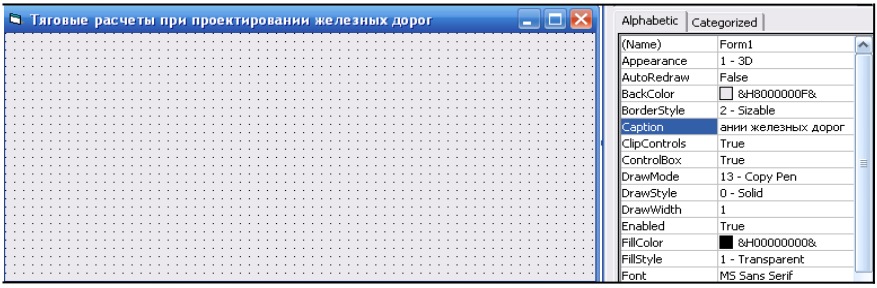


Рисунок 1.13 – Изменение внешнего вида программной формы с помощью изменения ее свойств

Иногда отображение всех трех кнопок справа в верху программной формы может привести к некоторым неудобствам (например, неполной видимости наименования формы при небольших размерах самой формы, излишнему количеству неприменяемых значков). Существует ряд свойств,

которые помогают удалить из формы кнопки «Свернуть» и «Развернуть». Для этого следует на этапе редактирования и отладки программы установить свойства формы MaxButton и MinButton в False.

Объект управления «Метка» является наиболее простым элементом. На рисунке 1.14 приведен способ управления его свойством «Font» с заданием шрифта, начертания и размера.

Важным свойством объекта Label является задание типа границы Border-Style. При выборе опции Fixed Single внешне метка становится похожей на текстовое поле с записью, которую нельзя редактировать после исполнения программы (рисунок 1.15).

Текстовое поле имеет такие важные свойства как Locked и MultiLine. Установка свойства Locked в True запрещает изменение пользователем значения параметра в данном текстовом поле в процессе исполнения программы пользователем. Свойство MultiLine = True позволяет вводить многостроковые тексты с автоматическим переносом слов на следующую строку (рисунок 1.16).

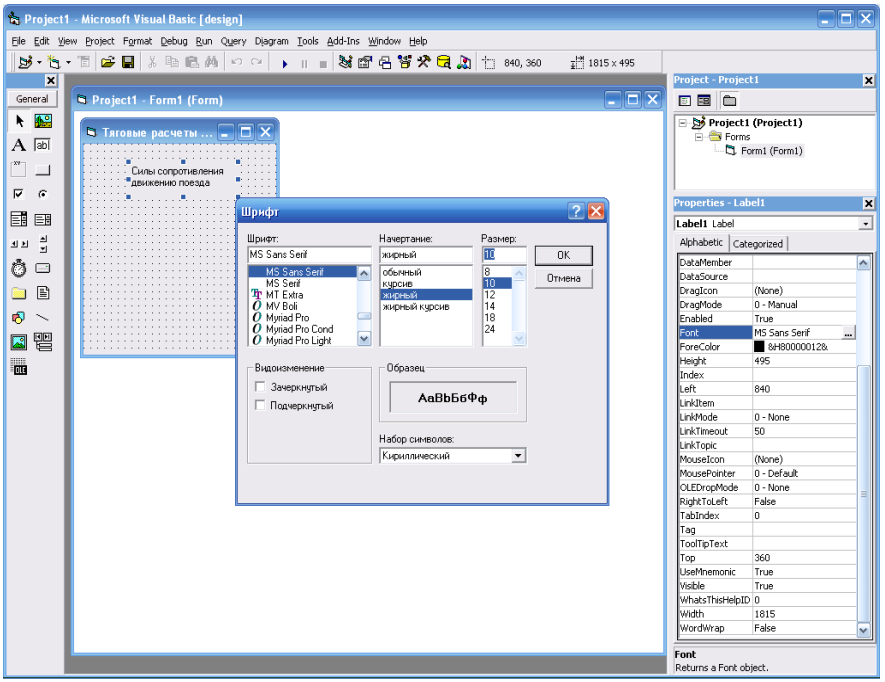


Рисунок 1.14 – Изменение свойства «Font» объекта Label

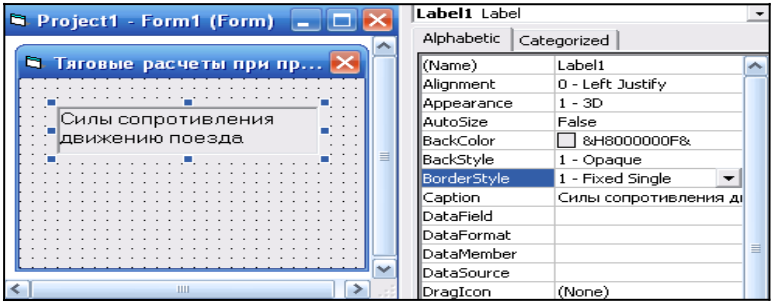


Рисунок 1.15 – Изменение внешнего вида записи в поле «Метка»

Наиболее интересным свойством элемента управления «Рамка» является MousePointer, с помощью которого можно задавать вид указателя курсора при перемещении его над полем рамки (рисунок 1.17).

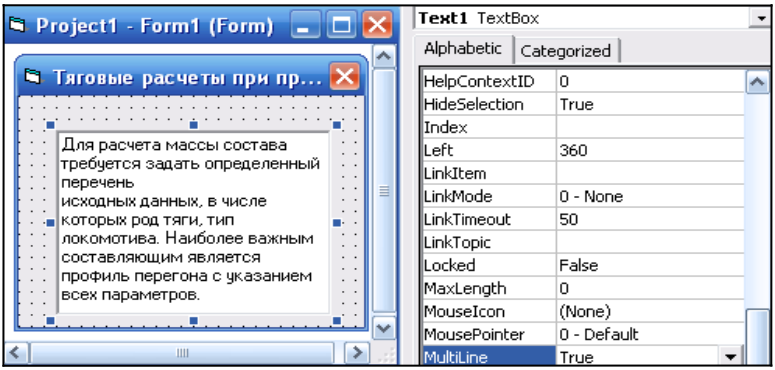


Рисунок 1.16 – Пример работы со свойствами текстового поля

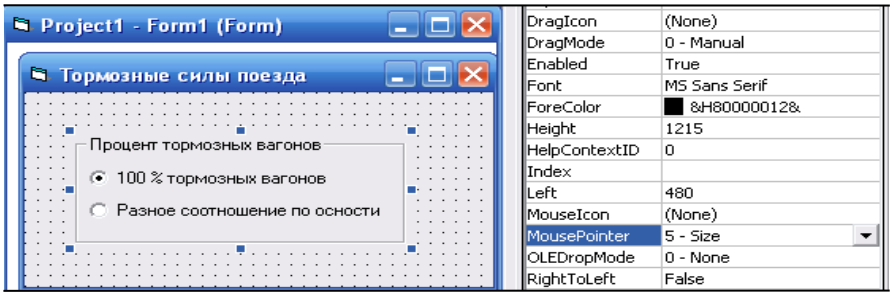


Рисунок 1.17 – Изменение вида курсора с помощью свойства MousePointer

На элементе CommonButton («Кнопка») можно разместить графическое изображение. Для этого следует установить свойство Style в положение «1 – Graphical», а по свойству Picture выбрать графический файл (рисунок 1.18).

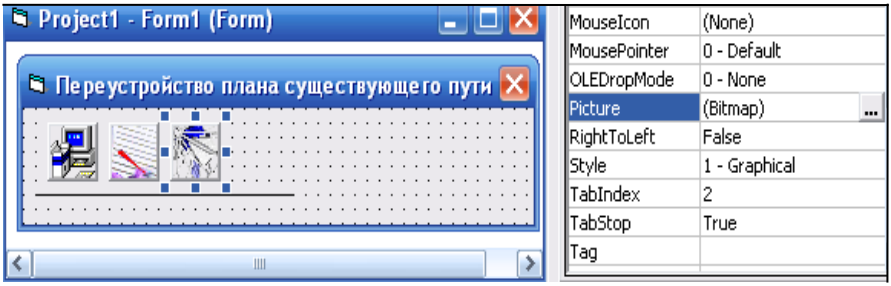
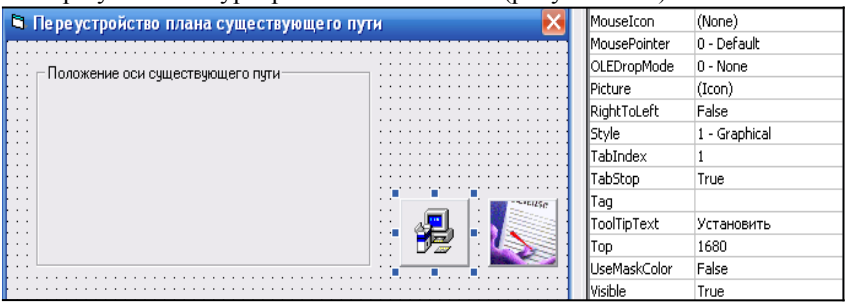


Рисунок 1.18 – Создание графической кнопки

Свойство ToolTipText позволяет формировать текстовую подсказку для кнопки при указании курсором на поле кнопки (рисунок 1.19).

а)



б)

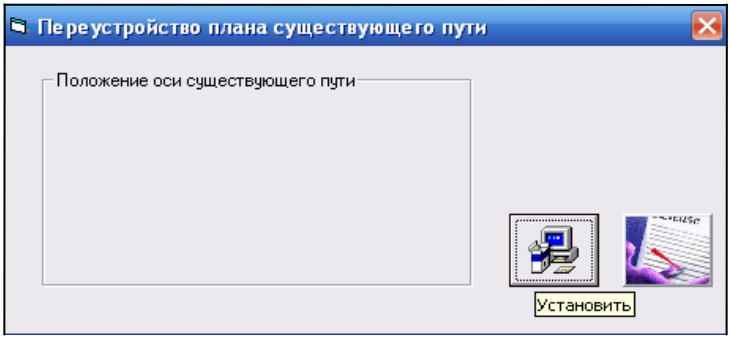


Рисунок 1.19 – Текстовая подсказка для CommonButton: *а* – установка свойства ToolTipText; *б* – внешний вид работающей кнопки

Элемент CheckBox также имеет свойства, которые переводят флажок в графическую форму с несколько иным внешним видом. Если установить Style в Graphical, то данный элемент становится похожим при Value = 0 на выпуклую (неактивную) кнопку, а при Value = 1 – на нажатую (активную) кнопку (рисунок 1.20).

а) *б)*

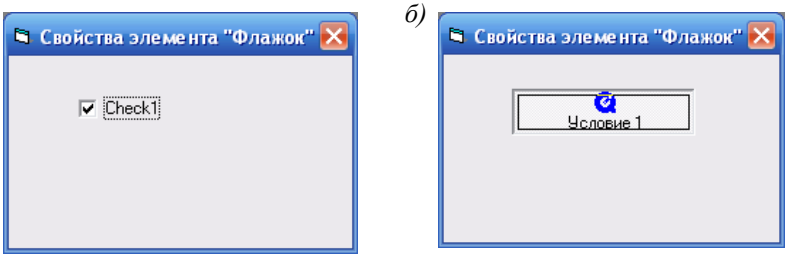


Рисунок 1.20 – Внешний вид элемента CheckBox:
а – стандартный (Style = 0, Value = 1);
б – графический (Style = 1, Picture = ...bmp, MousePointer = HourGlass, ToolTipText = «Проверка на трогание с места»)

Программный переключатель OptionButton имеет аналогичные свойства, которые могут перевести стандартную форму представления элемента в графическую.

Объект управления Shape («Фигура») обладает свойствами, позволяющими задавать вид фигуры (окружность, эллипс, прямоугольник с прямыми или скругленными краями и др.), толщину линии, тонировку внутреннего поля и др.

Для вывода графических изображений необходимо пользоваться элементами PictureBox и Image. Они имеют общие свойства (Name, BorderStyle, Picture, ToolTipText). Однако элемент Image имеет более широкие возможности отображения (рисунок 1.21) и его целесообразно использовать при программной настройке в ходе выполнения расчетов и проектирования.

Как уже указывалось, элемент Image не является контейнером. Это значит, что при размещении на его поле других элементов (например, кнопки) не формируется общая графическая среда связанных объектов. Элемент Picture может включать другие элементы (флажки, кнопки) и в дальнейшем работать как сложный скомпонованный

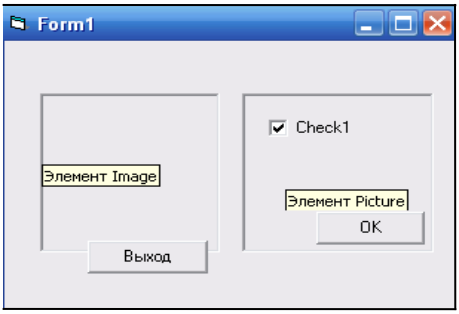


Рисунок 1.21 – Сравнительная

характеристика
элементов управления PictureBox и Image

Задачи

Задача 1.1 Заменить на рисунке 1.1 элемент TextBox, определяющий величину междупутья, на поле со списком (ComboBox).

Задача 1.2 Изменить интерфейс рисунка 1.1 посредством удаления рамки «Вид результатов». Вместо нее поместить флажки (CheckBox) с соответствующими указаниями.

Задача 1.3 Вынести за пределы фрейма «Общие исходные данные» элемент «Категория линии» и разработать отдельный фрейм.

Задача 1.4 Поместить на программную форму рисунка 1.1 вместо компонента «Длины переходных кривых – табличные» таблицу зависимости длин переходных кривых от радиуса круговых кривых (по аналогии с таблицей 3.2 [2, с. 20].

Задача 1.5 Вывести на форму рисунка 1.1 ряд кнопок с надписями «Записать в файл», «Выдать на экран», «Просмотр промежуточных результатов» вместо программных переключателей элемента управления «Вид результатов».

Задача 1.6 Распечатать интерфейс программы, полученный после всех изменений по результатам решения задач 1.1–1.5.

Указание – Распечатать интерфейс программы можно путем переноса изображения в буфер (нажатием кнопки Print Screen) и вставки его на страницу файла в текстовом редакторе Word.

Задача 1.7 Как может выглядеть интерфейс программы, изображенной на рисунке 1.1, если не использовать элемент управления OptionButton?

Задача 1.8 Разработать интерфейс программы расчета смещения оси пути, используя элемент управления SSTab, который активизируется в библиотеке Microsoft Tabbed Dialog. Вызов данного элемента осуществляется из меню Project-Components...-Microsoft Windows Common Controls 6.0 (SP4). Этот элемент является контейнером и состоит из ряда закладок. На каждую из них можно помещать другие элементы управления. Общий вид формы с одной активной закладкой приведен на рисунке 1.22.

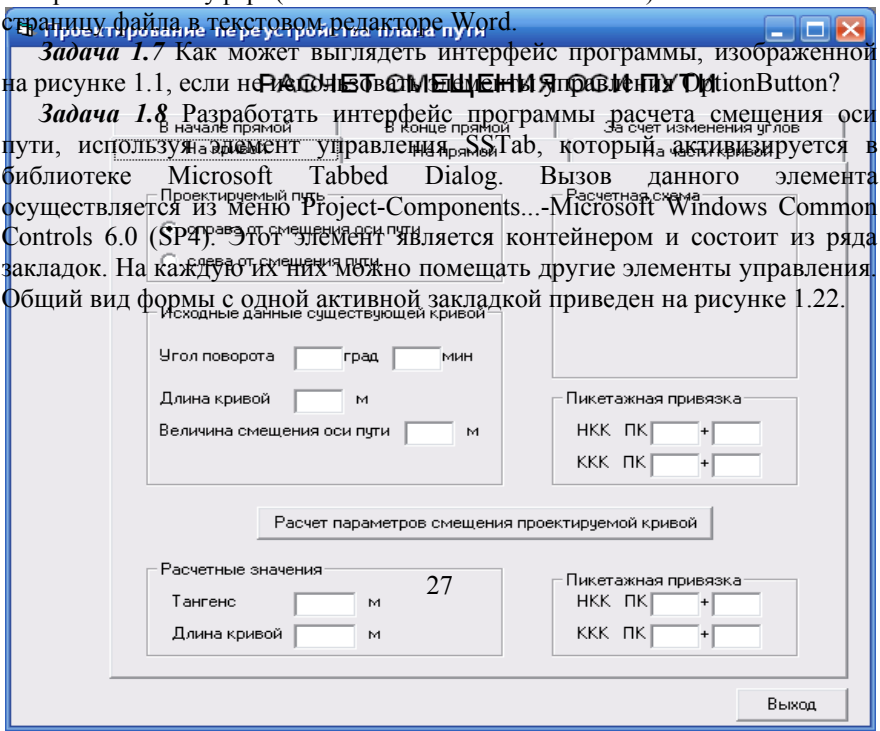


Рисунок 1.22 – Вид программы с активной закладкой

Указание – Добавление закладок выполняется путем выбора строки Properties выпадающего меню после нажатия правой кнопкой мыши на закладке. Переход от одной закладки к другой производится посредством активизации левой кнопкой мыши заголовка соответствующей страницы.

Задача 1.9 Изменить внешний вид формы рисунка 1.4 путем разделения закладки «на кривой» на две части. В левой половине следует поместить исходные данные, в правой – расчетные.

Задача 1.10 Дополнить программную форму рисунка 1.1 фреймами (рамками) с выводом пикетажных значений начала и конца переходных кривых.

Тесты

Тест 1.1 Visual Basic как инструментальная среда разработки программ узкопрофильного назначения:

1) требует от пользователя глубоких знаний языка программирования;

2) позволяет разрабатывать программы широкого назначения без единой строки кода;

3) помогает проектировать внешний вид программ и существенно сокращает затраты времени на программирование;

4) может использоваться как исходная основа для получения интерфейса.

Тест 1.2 Характеристика элементов управления определяется перечнем:

1) окна проекта;

2) окна свойств;

3) окна макета формы;
4) активной панели элементов.

Тест 1.3 Ввод данных обеспечивается элементом управления:
1) TextBox;
2) Label;
3) CommonButton;
4) OptionButton.

Тест 1.4 Элемент-контейнер:
1) замещает на форме последний объект, выбранный из панели элементов;
2) связывает текст с именем, автоматически размещая его вместе с данным объектом;
3) может объединять другие элементы и помещать их на данный элемент как на выделенную форму;
4) условное название определенного класса объектов, с которыми автоматически связывается определенный программный код.

Тест 1.5 Элемент OptionButten («переключатель»), определенный на форме рисунка 1.4 как «слева от смещения оси пути», является:
1) активным;
2) неотмеченным;
3) недоступным;
4) неопределенным.

Тест 1.6 Кнопки «Расчет», «Справка», «Запись в файл», «Выход» удобно поместить на объект-контейнер:
1) Frame;
2) Image;
3) CommonButton;
4) TextBox.

Тест 1.7 Элементы управления, недоступные для редактирования после загрузки программы (см. рисунок 1.1):
1) «Вид результатов»;
2) «Расположение второго пути»;
3) «Показать расчетную схему»;
4) «Справка».

Тест 1.8 В состав панели элементов может быть включено:
1) любое число объектов;
2) не более 20 объектов;
3) число элементов, графические изображения которых помещаются в пределах отведенной площади экрана дисплея;

4) не менее 10.

Тест 1.9 Окно свойств определяет характеристики:

1) основной программной формы;

2) выделенного элемента управления;

3) последнего элемента, перенесенного с панели на форму;

4) нескольких элементов, расположенных на контейнере.

Тест 1.10 Кроме расчетных параметров на программных формах Visual Basic можно выполнять проектирование элементов путевого развития:

1) в упрощенном виде по причине отсутствия инструментальных средств;

2) только с привлечением средств программирования графики;

3) посредством использования элементов управления Line, Shape, PictureBox, Image и последующим вызовом графических файлов;

4) с использованием символов клавиатуры как графических элементов.

Контрольные вопросы

1 Преимущество визуальных средств программирования.

2 Что такое интерфейс программы?

3 Какие элементы управления могут размещаться на рамке (Frame)?

4 Какие изменения последуют в расчетах при активизации кнопки «Криволинейная трасса» (см. рисунок 1.1)?

5 В чем отличие таких элементов как «Флажок» (CheckBox) и «Переключатель» (OptionButton)? Показать на примере рисунка 1.1.

6 Где размещаются элементы управления после выбора их из панели?

7 Что содержит окно свойств?

8 Чем отличаются элементы управления «Поле со списком» и «Список»?

9 Какие фигуры включает элемент Shape?

10 Можно ли программно изменить любую надпись на форме, выполненную с помощью элемента Label?

2 ИСПОЛЬЗОВАНИЕ СТРУКТУРЫ МЕНЮ, ДИАЛОГОВЫХ ОКОН И ВЛОЖЕННЫХ ФОРМ

2.1 Основные программные средства

Многие прикладные программы включают список команд меню, с помощью которых оказывается удобным управлять различными функциями. Этот список всегда располагается в верхней части загружаемой программной формы (см. рисунок 1.1) и при активизации одной из конкретных позиций может иметь сложную вложенную структуру (рисунок 2.1).

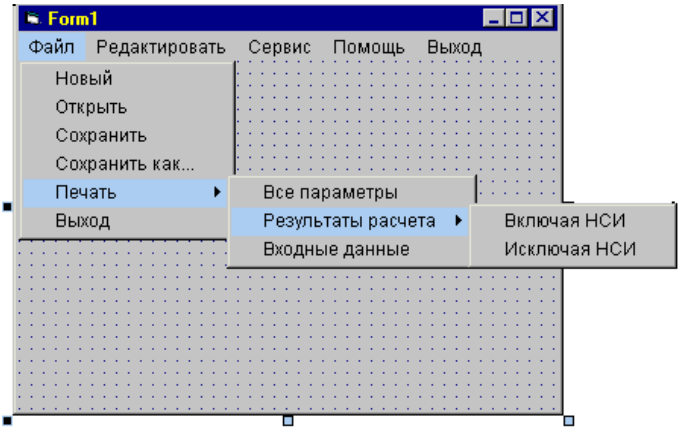


Рисунок 2.1 – Пример структуры меню прикладной программы

Структурно меню формируется как система горизонтальных и вертикальных уровней строк, по которым можно переходить с помощью манипулятора мыши или стрелок клавиатуры «вверх» и «вниз». При этом выпадающие области меню могут закрывать программную форму и представленные ней информационные области. Рекомендуется не выделять место на программной форме для выпадающих областей меню (пример рисунка 2.1 не является образцом и демонстрирует только вид меню).

Меню является также важным элементом управления, которое целесообразно проектировать практически всегда на выделенной программной форме. Редактор меню (Menu Editor) может быть вызван:

- командой Menu Editor, входящей в состав основного меню Tools загрузочной формы Visual Basic;

- нажатием кнопки Menu Editor на панели инструментов программной среды Visual Basic;

- нажатием правой клавиши указателя мыши на активной форме и выбором строки Menu Editor в раскрывшемся новом информационном окне (рисунок 2.2).

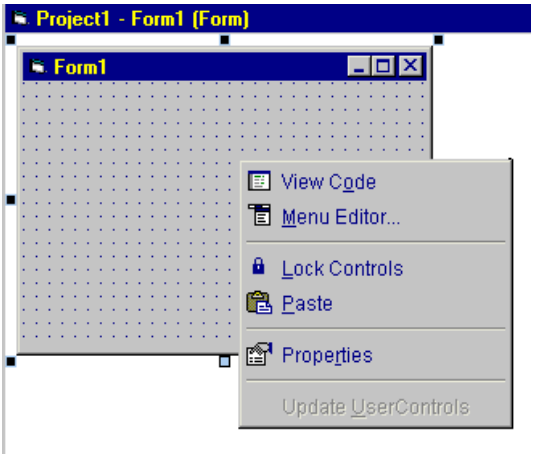


Рисунок 2.2 – Окно выбора конструктора меню

В результате любого из трех перечисленных действий появляется форма конструктора меню (рисунок 2.3).

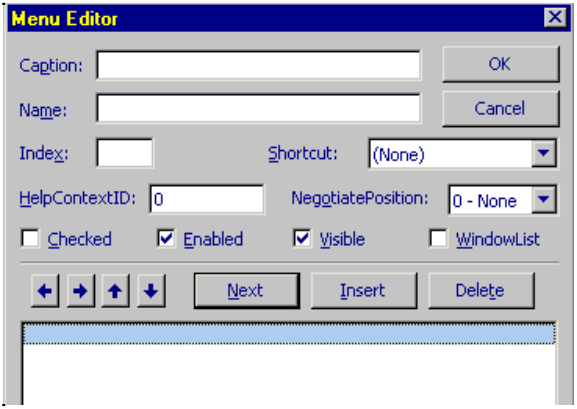


Рисунок 2.3 – Внешний вид конструктора меню

В нижней части формы размещаются элементы меню. В зависимости от вида различают заголовки и пункты меню, заголовки и пункты подменю. На рисунках 2.4 и 2.5 представлены фрагмент проектирования меню и его рабочий вид.

Для получения требуемого меню необходимо заполнить поле формы конструктора меню (см. рисунок 2.3).

Поле Caption определяет имя заголовка меню, отражаемого в выделенной строке нижней части конструктора.

Поле Name указывает имя, по которому впоследствии производится ссылка на данный пункт меню в программе. Следует отметить, что имя нужно набирать латиницей. Начало поля Name целесообразно всегда определять символами mnu…, по которым можно легко распознать наименование заголовка меню.

Поле Index заполняется значениями 1, 2, 3, …, если требуется создание массива элементов управления меню. В случае простого использования меню это поле можно оставлять незаполненным.

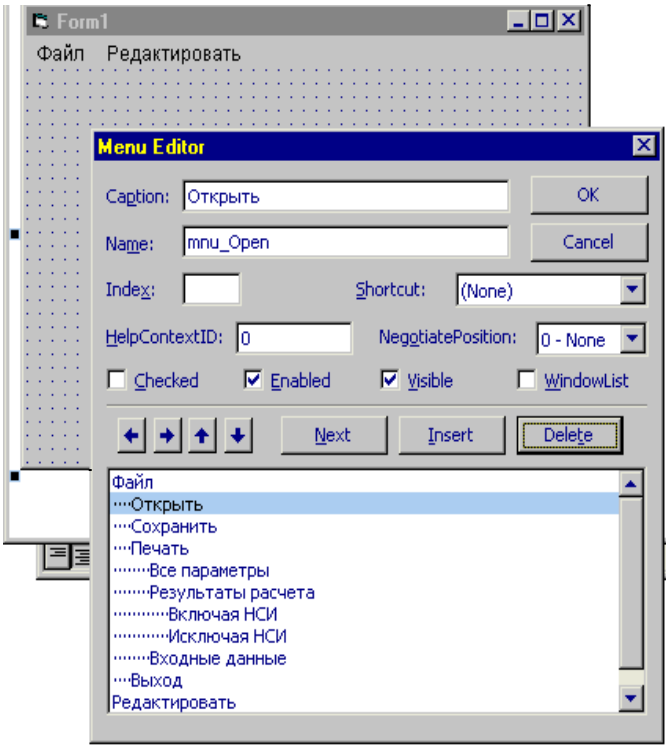


Рисунок 2.4 – Форма меню на этапе проектирования

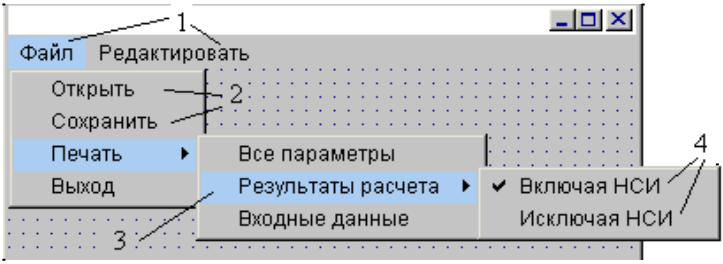


Рисунок 2.5 – Пример меню загруженной программы;
1 – заголовки меню; 2 – пункты меню; 3 – заголовок подменю;
4 – пункты подменю

Поле Shortcut можно заполнить для быстрого вызова заголовка меню по сочетанию клавиш из указанного выпадающего списка.

Заполнение поля HelpContextID позволяет связать данный заголовок меню с определенным справочным файлом.

Поле NegotiatePosition определяет расположение меню верхнего уровня при использовании нескольких форм.

Флажок Checked может помещать метку на элемент меню, указывая на использование некоторого свойства. Например, на рисунке 2.5 элемент меню «Включая НСИ» помечен флажком, указывающим на то, что все параметры нормативно-справочной информации используются в программе.

Установленный Enabled флажок указывает на активность данной опции меню. Если снять этот флажок, то заголовок меню останется видимым, но недоступным (т. е. курсор на нем нельзя зафиксировать).

Флажок Visible определяет видимость или невидимость данной опции меню.

Флажок WindowList активизирует свойство отображения списка открытых подчиненных форм. В общем случае его можно оставлять не установленным в активном положении.

Кнопки  позволяют перемещать позиции курсора в окне, расположенном ниже, в соответствующее место.

Кнопка Next переводит курсор в следующую позицию.

Кнопка Insert вставляет новую строку, Delete − удаляет активную строку в позиции курсора конструктора меню.

После заполнения всех данных по каждой опции меню необходимо нажать на кнопку ОК. В результате конструктор меню окажется закрытым, а запроектированное меню станет видимым в верхней части формы.

В пользовательских программах, разработанных с использованием среды Visual Basic, различают две разновидности диалоговых окон: ввода и

сообщения. Примеры соответствующих диалоговых окон приведены на рисунке 2.6.

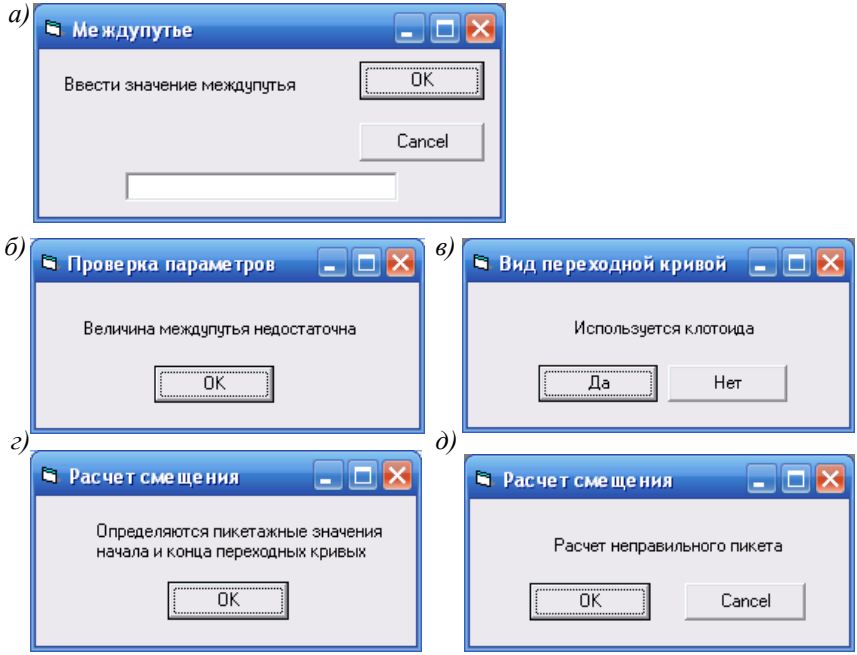


Рисунок 2.6 – Диалоговые окна ввода и сообщений:
а – простое окно ввода; *б* – сообщение с предупреждением; *в*, *д* – сообщения с выбором; *г* – информационное окно

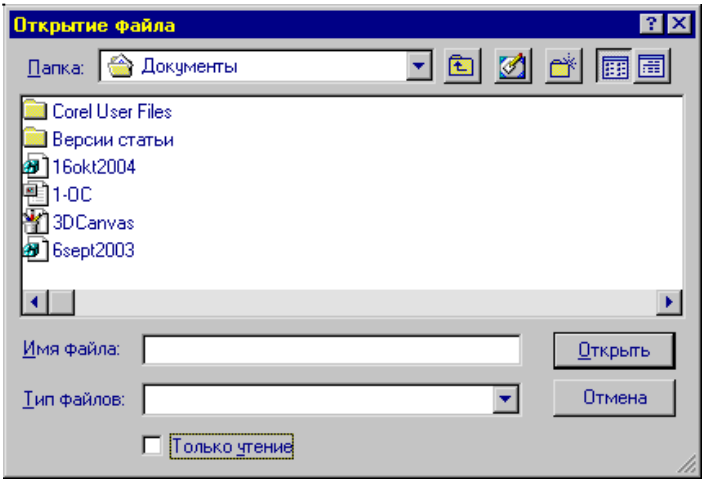
Кроме указанных программно предопределенных окон диалога используются стандартные окна, которые встречаются в приложениях Word, Excel: Print (Печать), FileOpen (Открыть файл). Эти окна диалога создаются с помощью элемента управления общим диалогом (CommonDialog). Если этот объект не входит в состав палитры элементов, то его можно вызвать из библиотеки компонентов (меню Visual Basic → опция Project→ Components…, на вкладке найти строку Microsoft Tabbed Dialog Control 6.0, отметить соответствующий флажок, нажать «Применить» и «ОК»).

Стандартные окна диалога можно использовать в пользовательских программах. Они позволяют быстро разрабатывать интерфейсы информационных окон, отличаются высокой эффективностью и надежностью в работе. Примеры работы таких элементов приведены на рисунке 2.7.

Применение этих окон требует прямого программирования и будет рассмотрено в других разделах данного пособия.

В состав программы может входить одна и более форм. Необходимость использования нескольких форм часто возникает при решении сложных задач многоэтапного расчета и проектирования элементов путевого развития железнодорожных станций и узлов.

а)



б)

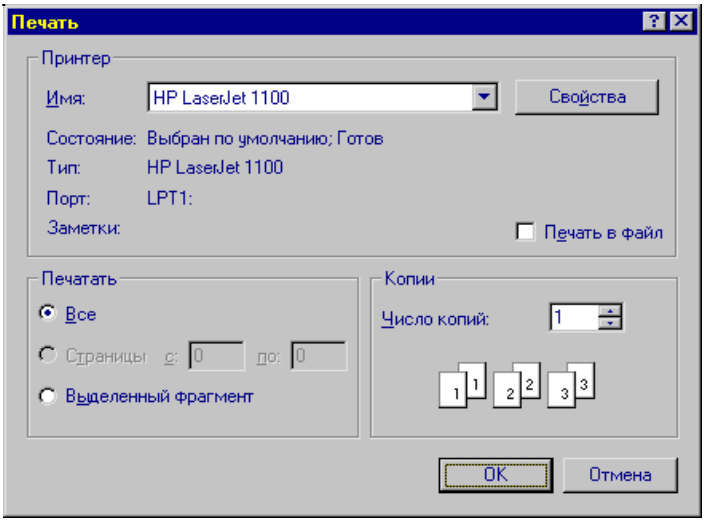


Рисунок 2.7 – Диалоговые окна элемента CommonDialog:
а – открытие файла; *б* – печать файла

Экспертно можно отметить, что новая программная форма требуется при достаточно плотном заполнении поля текущей формы, применении большого количества различных элементов управления Visual Basic.

Первая (основная) форма выводится автоматически после открытия среды Visual Basic.

Для открытия новой (одной или нескольких) форм в основном меню Project Visual Basic необходимо активизировать строку Add Form (добавить форму), а в появившемся окне New указать курсором на первую позицию Form (рисунок 2.8).

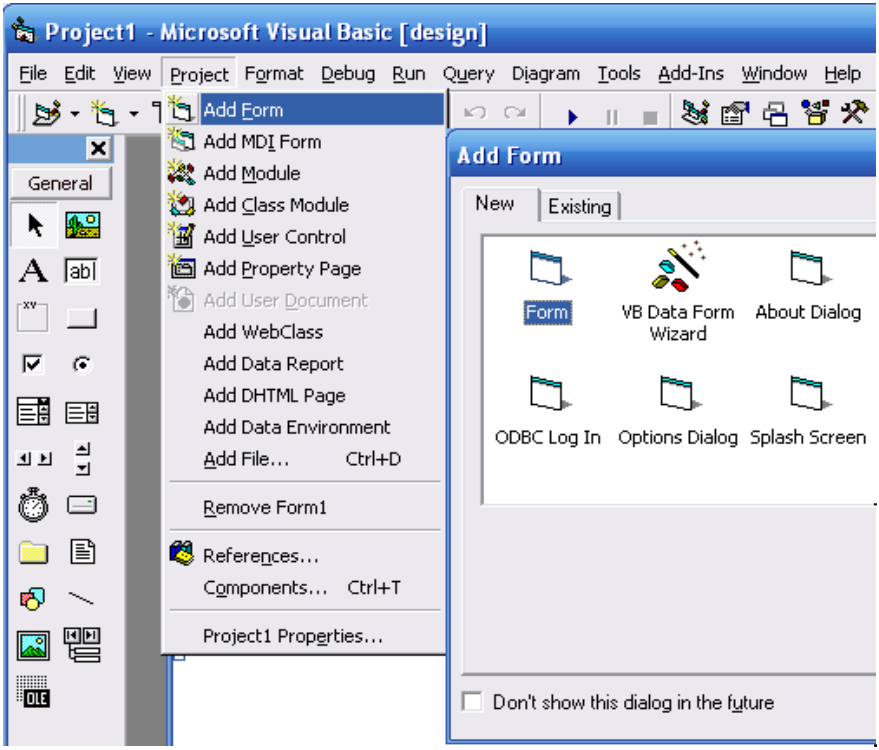


Рисунок 2.8 – Последовательность операций выбора новой программной формы

В результате поверху Form1 возникнет новая форма Form2, а в окне проекта добавится соответствующая новая строка (рисунок 2.9).

На каждом из окон можно помещать с палитры элементов любые объекты. Различают модальные и немодальные формы. Модальные формы ограничивают действия пользователя только размерами данной активной

формы. Переход на другие вложенные формы запрещен. Только после закрытия модальной формы можно осуществлять операции на основной (или любой другой) форме. В дальнейшем будет определено, как программно можно связать несколько форм между собой посредством разрешенных переходов, которые обусловливаются логикой связи элементов на формах методикой проектирования.

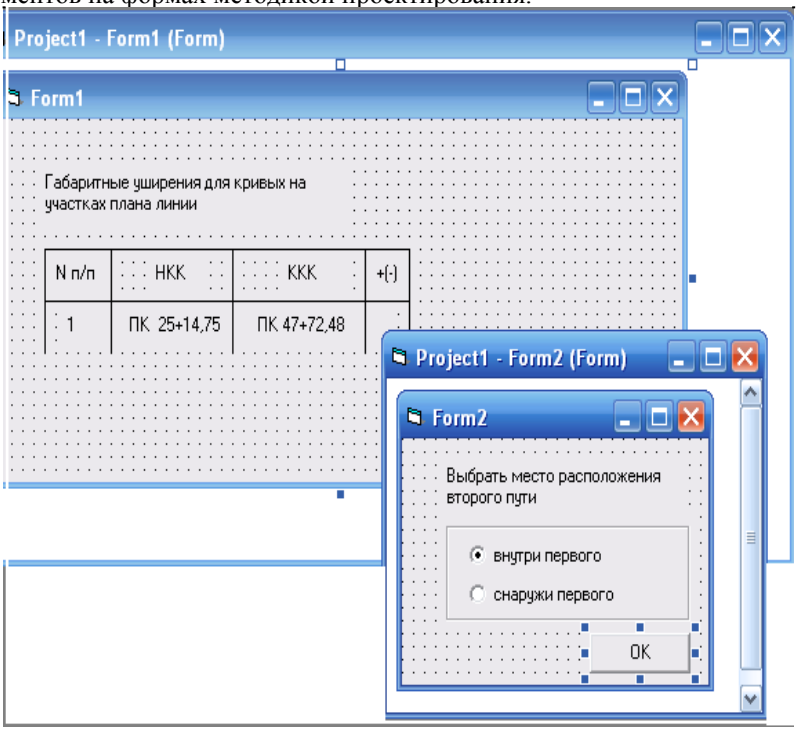


Рисунок 2.9 – Фрагмент окна программы с двумя формами

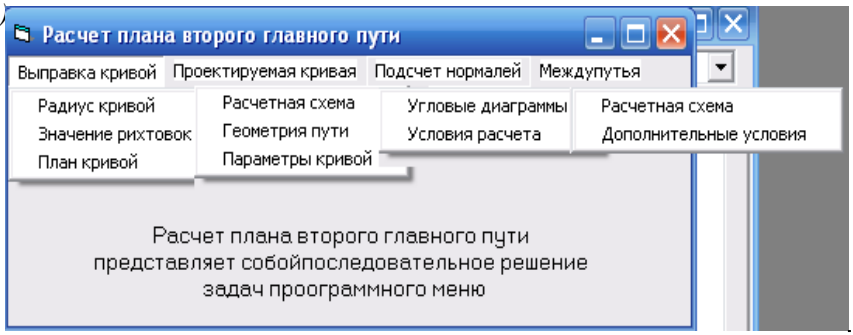
2.2 Примеры использования программных меню и вложенных форм

Программные меню представляют собой набор фиксированных позиций, расположенных в верху основной формы прикладной программы. Как правило, необходимость разработки меню возникает при сложных вычислениях с записью результатов расчета в файл, вызовом справочных данных и др. Программные меню можно использовать не только при проведении расчетов, но и при подготовке вариантов проектных решений

по выбору трассы, удлинению существующих и проектированию новых станционных и главных путей.

Меню всегда рассматривается как дополнительное средство расширения возможностей пользовательской программы. Важно точно определить название головных пунктов меню и их содержание. На рисунке 2.10 представлены различные варианты сконструированного программного меню для решения конкретной задачи расчета плана проектируемого второго пути на перегоне.

а)



б)

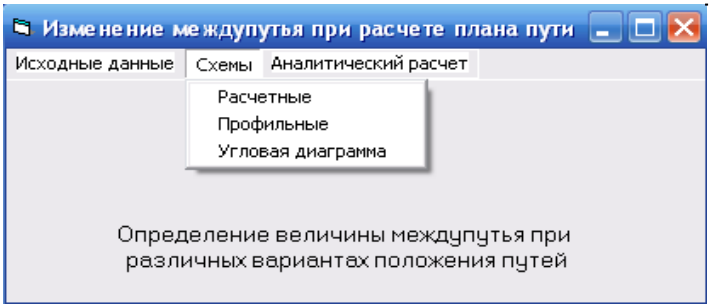


Рисунок 2.10 – Примеры программных меню:
а – удачного; *б* – неудачного

Меню рисунка 2.10, *а* включает основные задачи расчета плана второго пути, последовательность решения которых отражается соответствующими пунктами. В случае рисунка 2.10, *б* попытка компактного представления меню приводит к громоздкому и непонятному для пользователя виду. Следует отметить, что в исходной форме при загрузке программы отображается только основная строка меню. Раскрытие отдельных позиций происходит при активизации пользователем соответствующего пункта. Поэтому для рисунка 2.10, *б* следует изменить порядок следования пунктов меню и производить расчет в соответствии с методикой (рисунок 2.11).

В данном примере программное меню раскрывает ряд нисходящих задач, связанных друг с другом. В учебных целях подобная структура предпочтительна, и соответствующие обучающие и тестовые программы следует разрабатывать с использованием предложенного подхода.

В исключительных случаях программные меню можно использовать без заполнения основной формы Visual Basic. Такой подход может оказаться лучшим в случае решения изолированных задач, результаты которых целесообразно просматривать на экране дисплея и возможностью оперативной корректировки.

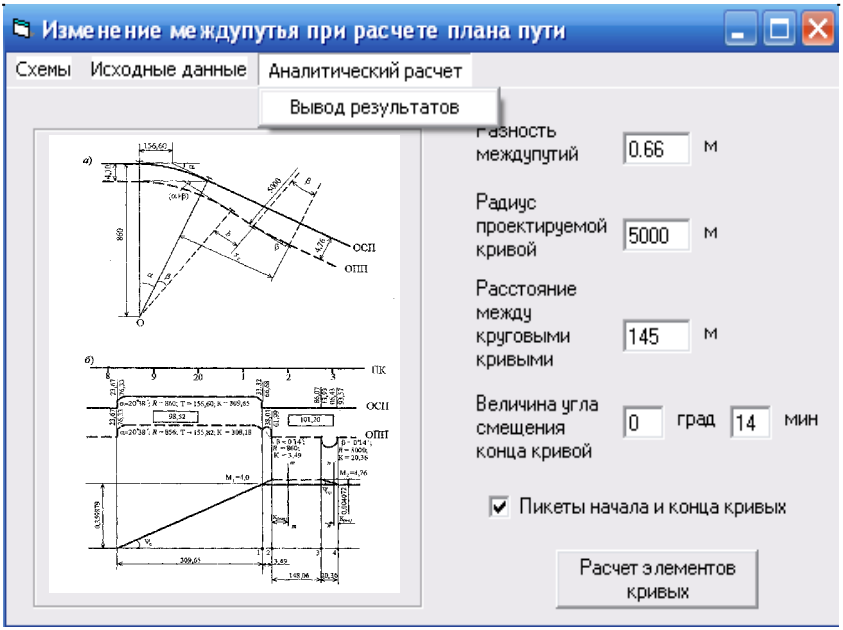


Рисунок 2.11 – Цепочечная схема расчетов

Как показывает практика, это позволяет сократить время получения окончательного результата на 10–15 % за счет отсутствия повторных расчетов и корректировок неточных промежуточных решений.

Выдачу опорных результатов в дополнительное окно следует признать нормальн Пример интерфей

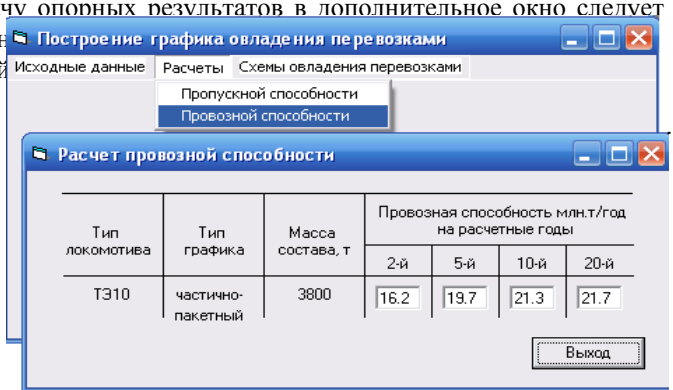


Рисунок 2.12 – Программные формы реализации пунктов меню

В этом случае ввод исходных данных и выполнение расчетов пропускной способности производятся на вложенной форме, которая удаляется после ее закрытия. Активная строка меню «Расчеты провозной способности» приводит к формированию нового вложенного окна, на котором отображается расчетная таблица, полученная по параметрам исходных данных. Если нажать на кнопку «Выход», то окно «Расчеты провозной способности» исчезнет, но результаты сохранятся в программе. Главная форма «Построение графика овладения перевозками» снова будет иметь незаполненное поле кроме пунктов меню. В дальнейшем можно переходить к последнему пункту меню «Схемы овладения перевозками» и решать задачу по аналогичной схеме вызова следующего вложенного окна.

2.3 Примеры использования диалоговых окон

Диалоговые окна ввода редко используются для определения значений исходных параметров. В учебных целях рекомендуется использовать этот вид диалоговых окон при необходимости пояснений, которые помогают пользователю правильно ввести значения параметра. Диалог повышает внимание пользователя при проведении расчетов с использованием программной среды. Это дает возможность активно применять сложные, информационно насыщенные структуры. В определенный расчетный момент времени одна из таких программных структур становится приоритетной и посредством диалогового окна фиксируется соответствующий элемент управления.

Данный подход позволяет решать поставленную задачу поэтапно в строгом соответствии с установившейся методикой и нормативными положениями. На рисунке 2.13 приведена иллюстрация программного интерфейса расчета параметров кривой при технико-экономическом обосновании величины радиуса посредством диалога с пользователем.

Следует обратить внимание на наличие вертикальной полосы прокрутки у рамки «План вариантов трассы». Посредством скроллинга можно изучать

другие области данной карты или выбирать другие карты, схемы прокладки трассы и т. п.

Диалоговые окна сообщений являются эффективным средством поддержки обратной связи с пользователем. Вызов всех диалоговых программно предопределенных окон производится с помощью строки кода:

MsgBox *сообщение*, *атрибуты*, *заголовок*,

где *сообщение* – основной текст, который отражается в диалоговом окне;

атрибуты – особенности окна выдачи, определяющие количество кнопок и графические иконки;

заголовок – название окна диалога.

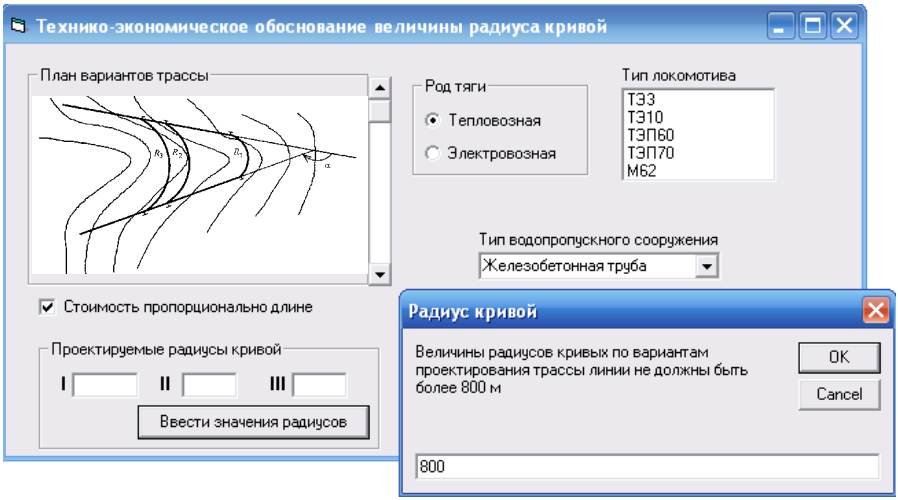


Рисунок 2.13 – Вариант использования диалогового окна ввода данных

Например, строка

MsgBox *"Введено недопустимое значение угла", 16, "Ошибочный параметр"*

приводит к появлению следующего окна (рисунок 2.14).

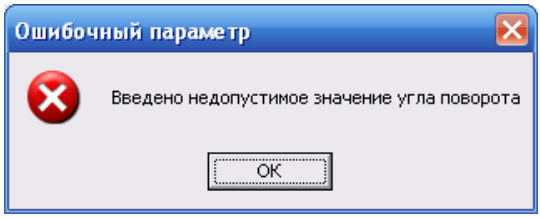


Рисунок 2.14 – Пример сообщения с предупреждением

Другие примеры реализованных строк программного кода с выводом различных диалоговых окон приведены на рисунке 2.15.

Диалоговые окна рисунка 2.15, *а* и *в* имеют принципиальные различия. Первое окно может прервать работу программы по указанию пользователя. Во втором случае программа продолжит работу с выводом (или без вывода) результатов расчета на схеме.

Необходимость использования диалоговых окон возникает в таких случаях, когда требуется сравнить программную ситуацию с каким-либо критерием. Например, задача поиска файлов на различных логических дисках винчестера и внешних накопителях памяти ЭВМ связывается со сложным анализом корректности обращения к физическим и логическим приводам.

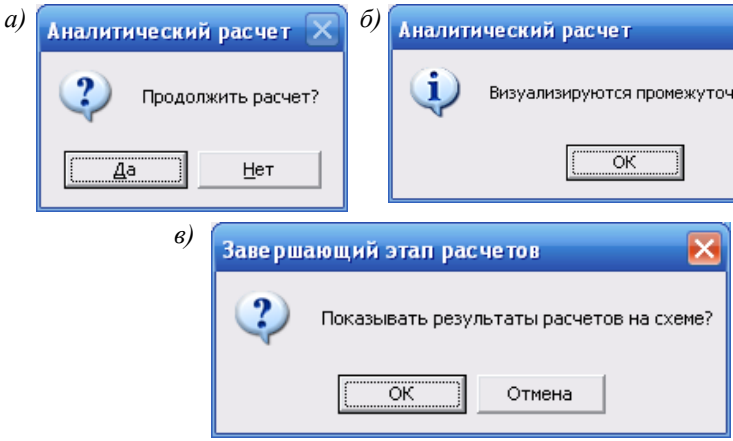


Рисунок 2.15 – Примеры диалоговых окон:
а – с выбором пользователя; *б* – с предупреждением;
в – с выводом информации

В числе важных элементов управления Visual Basic имеются такие объекты как DriveListBox (список дисков), DirListBox (список каталогов) и FileListBox (список файлов). Выводим эти элементы на поле формы (рисунок 2.16).

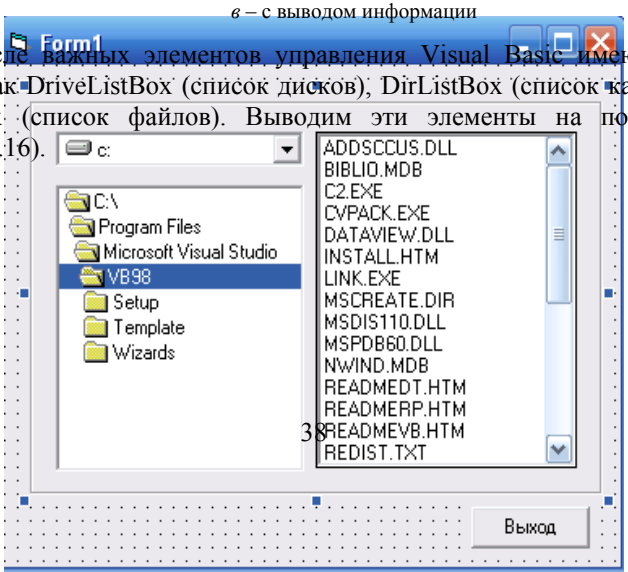


Рисунок 2.16 – Элементы управления накопителями
на программной форме

Если связать их между собой программно, то мы получим достаточно эффективное средство поиска необходимых файлов. При двойном нажатии указателем мыши на поле объекта Drive1 в процедуру Drive1_Change вводим соответствующий код, который характеризуется наличием программной метки (Label5).

```
Private Sub Drive1_Change()
  On Error GoTo Label5
    Dir1.Path = Drive1.Drive
Exit Sub
Label5:
  MsgBox "Нет диска в накопителе", 64, "Ошибка обращения"
  Drive1.Drive = Dir1.Path
  Exit Sub
End Sub
```

При двойном нажатии на поле объекта Dir1 в процедуру Dir1_Change вводим код:

```
Private Sub Dir1_Change()
  File1.Path = Dir1.Path
End Sub
```

Исполнение данной программы позволяет просмотреть все каталоги, папки и файлы на дисках винчестера и внешних накопителях. При отсутствии дискеты или диска в дисководе происходит корректная обработка исключительной ситуации с выдачей соответствующего сообщения диалогового окна (рисунок 2.17).

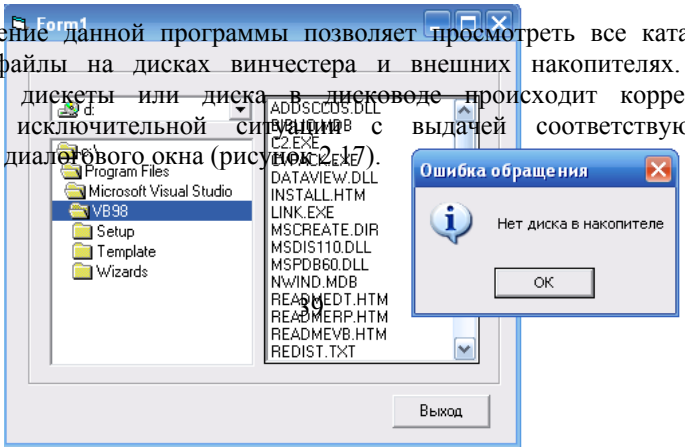


Рисунок 2.17 – Диалоговое окно обработки
исключительной программной ситуации

Следует отметить важное свойство элемента управления «Список файлов». Если при двойном нажатии указателя мыши на данной форме в открывшееся окно ввести строку *File1.Pattern = "*.dll",*то после исполнения программы в окне файлов будут показаны только файлы с расширением dll (рисунок 2.18).

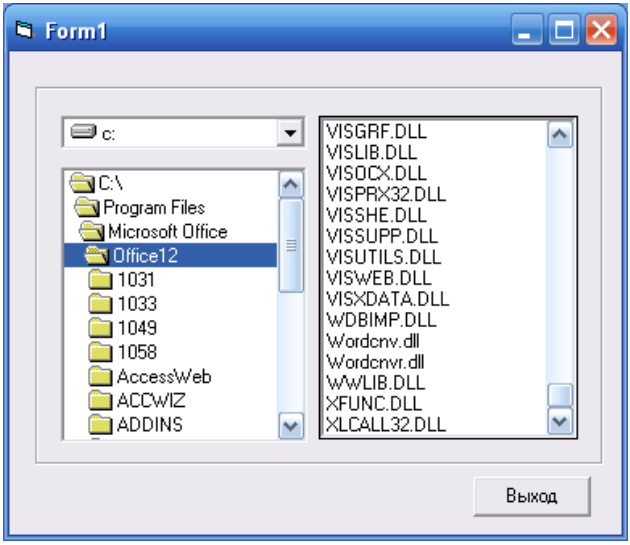


Рисунок 2.18 – Выбор типа файлов в окне элемента

Чтобы в списке файлов указывались системные, скрытые и «Только для чтения», необходимо установить следующие свойства:

File1.System = True

```
File1.Hidden = true
File1.ReadOnly = True.
```

Для подсчета числа найденных файлов в текущем окне списка следует дополнить программную форму одним текстовым окном и редактированным кодом:

```
Private Sub Dir1_Change()
  File1.Path = Dir1.Path
  Text1.Text = File1.ListCount
End Sub
Private Sub Drive1_Change()
  On Error GoTo Label5
    Dir1.Path = Drive1.Drive
Exit Sub
Label5:
  MsgBox "Нет диска в накопителе", 64, "Ошибка обращения"
  Drive1.Drive = Dir1.Path
  Exit Sub
End Sub

' Private Sub Form_Load()
'   File1.Pattern = "*.dll"
' End Sub

Private Sub Form_Load()
Text1.Text = File1.ListCount        ; определение количества файлов в окне
End Sub
```

Исполнение данной программы приводит к следующей визуализированной форме (рисунок 2.19).

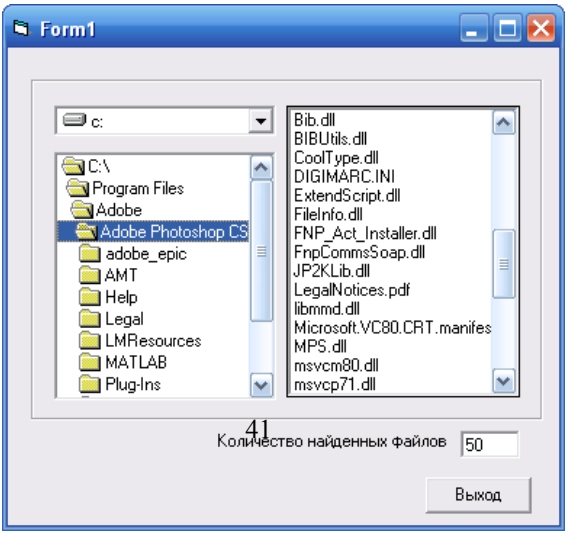


Рисунок 2.19 – Интерфейс программы с подсчетом
числа файлов

Благодаря корректному объединению различных элементов управления на форме при изменении поля файлов происходит постоянный перерасчет количества файлов в текущем окне.

Задачи

Задача 2.1 Разработать внешний вид программы с двумя формами (по образцу программы рисунка 2.9) и вводом данных по пикетажной привязке НКК и ККК в таблицы на отдельных модальных формах.

Задача 2.2 Разработать программную форму, эквивалентную вложенному меню, представленному на рисунке 2.20.

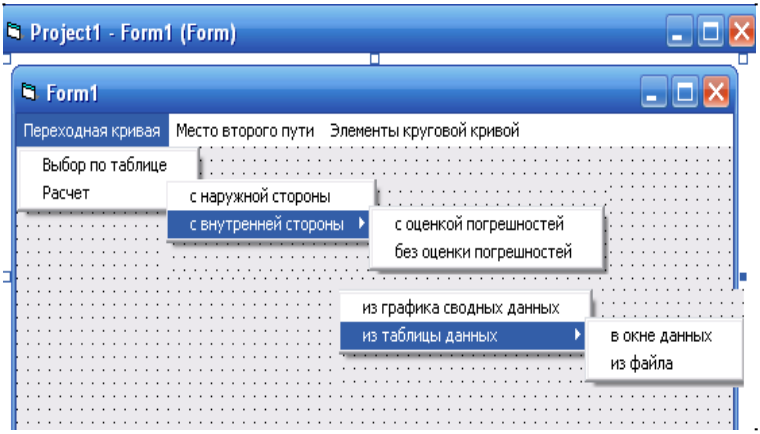


Рисунок 2.20 – Структура меню расчета кривой

Задача 2.3 Провести анализ эффективности использования программных компонентов по решениям рисунка 2.20 и задачи 2.2. Разработать новую форму с использованием меню и соответствующих программных компонентов.

Задача 2.4 Разработать систему модальных окон, на которых изображены расчетные схемы смещения пути на части кривой при различном расположении переустраиваемого пути согласно рисункам 4.11–4.14 [2].

Задача 2.5 Определить возможное содержание пунктов программного меню при расчете контрольных междупутий по графику сводных данных.

Задача 2.6 Скорректировать вид программного меню предыдущей задачи в местах расположения искусственных сооружений.

Задача 2.7 Определить вид программного меню для подъемок и срезок графика сводных данных.

Задача 2.8 Отразить в программном меню перечень элементов, которые определяются при расчете плана второго пути и реконструируемого первого пути (см. «Общие положения» [2]).

Задача 2.9 Разработать две программные формы для расчета значений величин подъемки и срезки бровок земляного полотна (см. формулы (2.3), (2.4) [2]).

Задача 2.10 Разработать две программные формы с параметрами круговой кривой и графическим представлением соответствующего фрагмента плана пути.

Тесты

Тест 2.1 Меню как элемент управления может быть создано с помощью:

1) специальных средств, не входящих в состав типовой среды Visual Basic;
2) редактора меню Menu Editor;
3) последовательного переноса на форму элементов управления Label;
4) вложенных форм с элементами управления Label.

Тест 2.2 Диалоговые окна служат:

1) для передачи управления определенному элементу формы;
2) отображения данных из файла;
3) ввода данных и выдачи сообщений;
4) подтверждения или отмены принятого решения проектировщиком.

Тест 2.3 Вызов окон диалога осуществляется с помощью элемента:

1) CommonButton;
2) ComboBox;
3) CommonDialog;
4) FileOpen.

Тест 2.4 Информационное диалоговое окно содержит кнопки:

1) «ОК», «Отмена»;
2) «Отмена»;
3) «Да», «Нет»;
4) «ОК».

Тест 2.5 Диалоговое окно ввода содержит кнопки:
1) «ОК», «Cancel»;
2) «ОК»;
3) «Cancel»;
4) «Да».

Тест 2.6 Меню программы включает:
1) заголовки и пункты меню;
2) перечень команд;
3) последовательность элементов управления;
4) имена элементов управления.

Тест 2.7 Элемент меню может содержать метку, которая устанавливается:
1) в поле Name;
2) флажком Checked;
3) флажком Enabled;
4) флажком WindowList.

Тест 2.8 Курсор в окне редактора меню перемещается с помощью:
1) кнопки Next;
2) кнопки Insert;
3) кнопок «Влево», «Вправо», «Вверх», «Вниз»;
4) нажатия клавиши <ENTER>.

Тест 2.9 Применение стандартных окон диалога обеспечивается:
1) разработкой программного кода;
2) выбором свойств элемента управления CommonDialog;
3) загрузкой соответствующих файлов;
4) проектированием интерфейса формы, на которой располагается соответствующее окно диалога.

Тест 2.10 Имена заголовков и пунктов меню на форме определяются:
1) значением поля Name;
2) значением поля Caption;
3) установкой флажка Enabled;
4) кнопкой ОК.

Контрольные вопросы

1 Где располагается меню на программной форме?

2 Перечислить элементы конструктора меню, с помощью которых проектируется вид меню.

3 Как установить метку на элемент меню?

4 Как удалить пункт меню в позиции курсора конструктора меню?

5 Привести примеры использования диалоговых окон ввода при разработке программы расчета элементов плана линии.

6 Какие окна диалога относятся к стандартным?

7 Чем отличаются модальные формы от немодальных?

8 Как удалить программную форму?

9 Можно ли помещать на стандартные диалоговые окна, вызываемые с помощью элемента управления CommonDialog, другие элементы управления?

10 Как можно использовать в одной программе меню, диалоговые окна и две формы (на примере решения какой-либо задачи данной темы)?

3 ОСНОВЫ ПРОГРАММИРОВАНИЯ В СРЕДЕ VISUAL BASIC

3.1 Основные программные средства

Программа, запроектированная только с использованием различных объектов панели элементов, может быть исполнена после нажатия кнопки Start на панели инструментов Visual Basic (см. рисунок 1.3), строки меню Run→Start или функциональной клавиши F5. Однако эффективность работы такой программы оказывается крайне низкой: кнопки при нажатии на них не отвечают, информация в текстовых окнах может быть изменена. Полезные действия с окном программы выполнить нельзя по причине отсутствия указаний, каким образом должна реагировать программа на запросы пользователя.

Для получения работоспособной и высокоэффективной программы необходимо к каждому (или выделенным) элементу управления, расположенному на форме, прикрепить соответствующий программный код. Это оказывается не трудно исполнить, так как среда автоматизированного проектирования Visual Basic предоставляет широкие возможности разработки и тестирования программы в окне редактора кода, который можно рассматривать как особую форму, раскрывающуюся после двойного щелчка левой кнопки мыши на любом элементе (рисунок 3.1).

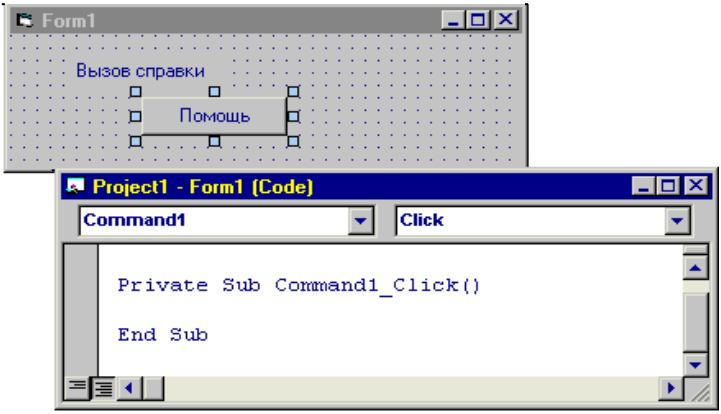


Рисунок 3.1 – Связь элемента формы и редактора кода

При этом следует обратить внимание на то, что автоматически формируется название процедуры, указывающей на способ ее реализации. Кнопка как элемент управления по умолчанию имеет название Command1, а в программном коде ее рекомендуется определять с именем cmd. Код, помещенный между строками Private Sub Command1_Click и End Sub, будет исполняться после нажатия левой клавиши мыши на эту кнопку (событие Click). Поэтому имя данной процедуры Command1_Click. Для большей информативности программные имена объектов целесообразно выбирать в соответствии с их назначением. Например, в данном случае (см. рисунок 3.1) имя кнопки лучше определить как CmdHelp. Его нужно записать в поле Name окна свойств элемента управления. После изменения имени следует обратить внимание на то, что изменилось имя процедуры в окне редактора кода.

Каждый из элементов, помещенных на форму (в том числе и сама форма), имеет целый ряд свойств, указанных в окне свойств. Их можно изменять не только в процессе редактирования программы, но и в процессе ее исполнения. Например, если дважды нажать на кнопку «Помощь» (см. рисунок 3.1) и на месте курсора в окне редактора кода записать

Label1.Caption = “Обращение к справке”,

то после исполнения кода первоначальный текст метки на форме изменится на программный.

Так как программа должным образом не завершена, то выйти из загруженного фрагмента можно нажатием на кнопку (справа вверху формы), а поместить форму над редактором кода можно посредством активизации строки Form1(Form1) двойным нажатием левой кнопки мыши в окне проекта. Таким образом следует управлять объектами на форме посредством указания имени изменяемого элемента и его свойства, записываемого через точку. Если имеется несколько вложенных свойств, то каждое последующее по отношению к предыдущему записывается также через точку. Используя полученные сведения, разработаем программу, интерфейс которой приведен на рисунке 3.2.

Разработка данной программы требует выполнения ряда операций по размещению на форме различных элементов: меток, текстовых полей, фреймов, кнопок. Как видно из рисунка 3.2, необходимо поместить на форму 25 текстовых полей (TextBox), а соответствующие значения записать в свойство Text данных элементов. Следует обратить внимание на то, что форма называется не Form1, а «Расчет элементов плана переустраиваемого пути».

Для этого нужно указать данный текст в поле Caption при активной форме (Верхняя строка формы выделена темным цветом). Метки

«Исходные данные» и «Результаты расчетов» указываются на форме увеличенным шрифтом.

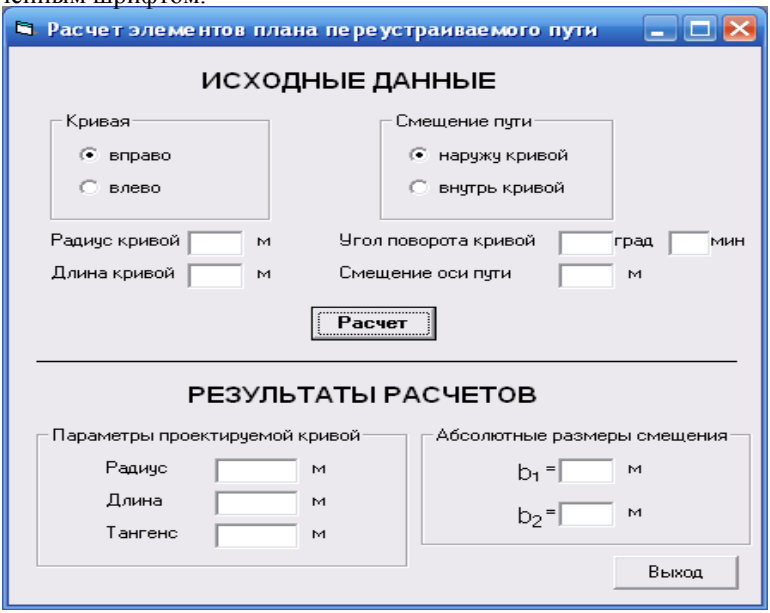


Рисунок 3.2 – Интерфейс программы расчета
плана переустраиваемого пути

Для оформления внешнего вида данных объектов необходимо открыть при активном объекте «Метка» свойство Font, нажать на кнопку с многоточием, выбрать начертание шрифта «Жирный», размер – 14 пт и нажать кнопку «ОК». Аналогичным образом следует изменить свойство шрифта на жирный для кнопки «Расчет».

Для удобной навигации по различным элементам формы следует называть эти элементы понятными именами. Например, поле для ввода радиуса кривых исходных данных по умолчанию имеет имя Text1 (рисунок 3.3).

Целесообразно изменить поле Name с Text1 на более понятное имя Txt_Radius_Tek (поле ввода радиуса существующей кривой). Аналогичным образом нужно изменить имена других параметров: для переменной длины существующей кривой с Text2 на Txt_Dlina_Tek, угла поворота кривой – с Text3 и Text4 на Txt_Ugol_Grad и Txt_Ugol_Min, смещения пути – с Text5 на Txt_Delta. Для параметров проектируемой кривой вводятся следующие переменные: тангенса – Txt_Tangens_pr, длины кривой – Txt_Kriv_pr, радиуса – Txt_Radius_pr.

С этих позиций рекомендуется изменять все используемые в программе переменные, придавая им осмысленные наименования. Форма, на которой помещены все программные элементы, имеет имя Form_Puti.

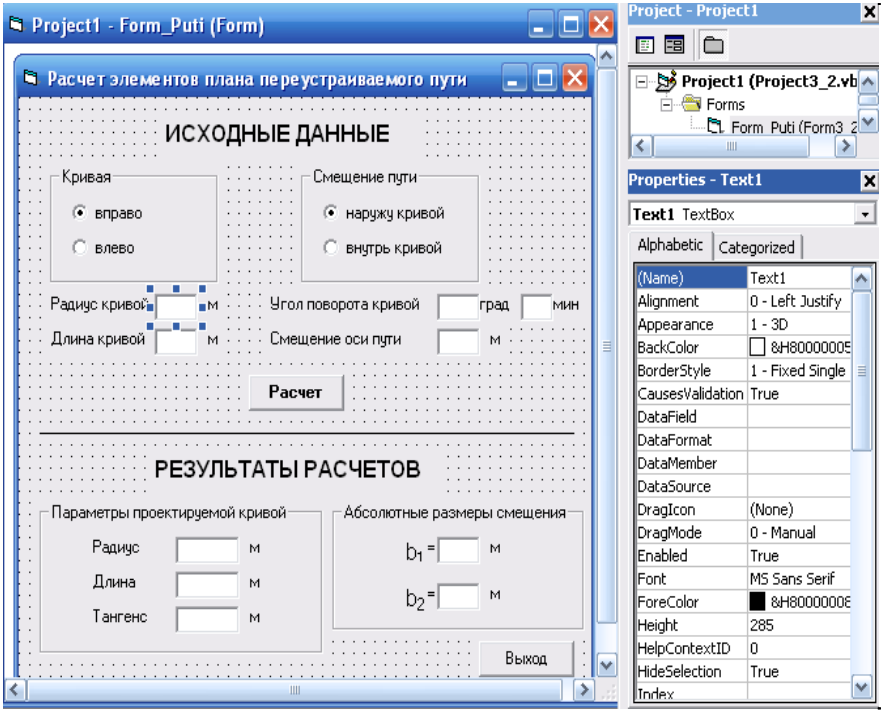


Рисунок 3.3 – Связь поля ввода радиуса кривой с его именем Text1

По умолчанию загрузочное окно программы устанавливается в верхнем левом углу экрана дисплея. Однако удобнее помещать данное окно по центру экрана. Это достигается с помощью небольшой процедуры, добавляемой к событию, определяющему загрузку формы в самом начале работы программы. Если дважды нажать левой клавишей мыши в любом месте формы, свободном от других элементов, то откроется редактор кода с двумя заголовками: Form и Load. В правом заголовке после нажатия на стилизованный треугольник раскрывается список, в котором следует найти строку Resize (рисунок 3.4).

При активизации этой строки путем нажатия левой клавиши мыши раскроется поле для ввода программного кода с курсором, указывающим место ввода текущего кода. В нашем случае нужно ввести строки кода, которые позволят при запуске программы помещать форму по центру

экрана дисплея. Между уже существующими строками Private Sub Form_ Resize () и End Sub нужно поместить следующий код:

Left = (Screen.Width - Width) / 2

Top = (Screen.Height - Height) / 2

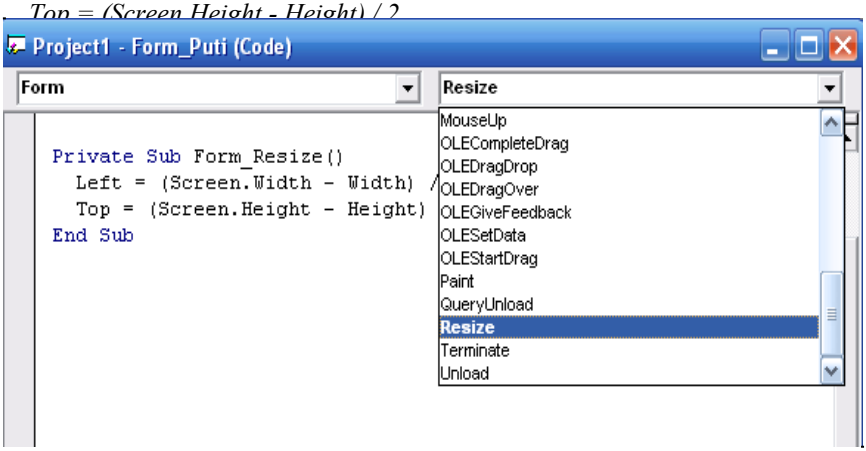


Рисунок 3.4 – Вид программного окна с записью кода

Корректный выход из программы обеспечивается следующим программным кодом, который вводится после двойного клика мышью на кнопке «Выход» и предыдущем именовании ее как CmdExit:

Private Sub CmdExit_Click()

Unload Me

Set FormPuti = Nothing

End Sub

В программе необходимо указать типы переменных (целый, дробный, строковый), так как описание всех переменных всегда предшествует их использованию в расчетах. Следующий важный шаг заключается в преобразовании значений из текстовых окон в числовые параметры. Все числа в текстовых окнах TextBox, Label воспринимаются программой на Visual Basic как строки, однако в формулах должны использоваться только числовые значения. Поэтому необходимо применить функцию преобразования текстовой строки (Val) в число, а после проведения вычислений – произвести обратное преобразование для записи в текстовые окна (функция CStr).

После проведения всех описанных шагов согласно методике [2, с. 34] получаем следующий программный код, приведенный на рисунке 3.5.

Для удобства чтения исходного кода целесообразно дополнять каждую строку поясняющим текстом (комментариями). Возможный вариант оформления программного кода представлен ниже:

```
Private Sub Form_Resize()
  Left = (Screen.Width - Width) / 2
  Top = (Screen.Height - Height) / 2
End Sub

Private Sub Cmd_Exit_Click()
  Unload Me
  Set FormPuti = Nothing
End Sub
```

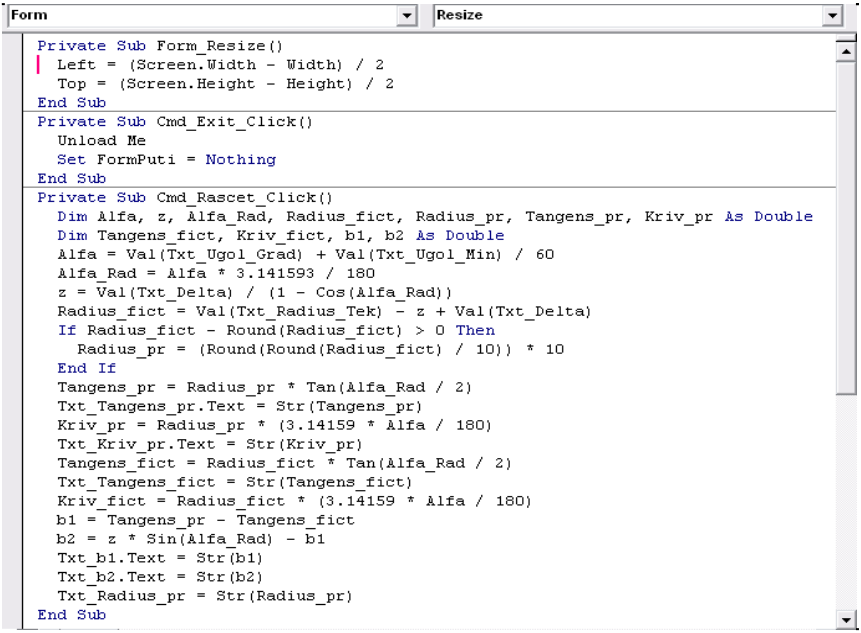


Рисунок 3.5 – Программный код исполнения задачи

```
Private Sub Cmd_Rascet_Click()
  Dim Alfa, z, Alfa_Rad, Radius_fict, Radius_pr, As Double     ; описание
  Dim Tangens_pr, Kriv_pr As Double                            ; всех
  Dim Tangens_fict, Kriv_fict, b1, b2 As Double                ; переменных
  Alfa = Val(Txt_Ugol_Grad) + Val(Txt_Ugol_Min) / 60           ; угол α, рад
  Alfa_Rad = Alfa * 3.141593 / 180                             ; расчет z, рад
  z  = Val(Txt_Delta) / (1 - Cos(Alfa_Rad))                    ; расчет Rф
  Radius_fict = Val(Txt_Radius_Tek) - z + Val(Txt_Delta)       ; округление Rф
  If Radius_fict - Round(Radius_fict) > 0 Then                 ; до целого
    Radius_pr = (Round(Round(Radius_fict) / 10)) * 10          ; значения Rпр
```



```
                                                               ; печать Rпр
                                                               ; расчет Tпр
                                                               ; печать Tпр
                                                               ; расчет Kпр
```

```
    End If
    Txt_Radius_pr = CStr(Radius_pr)
    Tangens_pr = Radius_pr * Tan(Alfa_Rad / 2)
    Txt_Tangens_pr.Text = CStr(Tangens_pr)
    Kriv_pr = Radius_pr * (3.14159 * Alfa / 180)
    Txt_Kriv_pr.Text = CStr(Kriv_pr)                    ; печать Kпр
    Tangens_fict = Radius_fict * Tan(Alfa_Rad / 2)      ; расчет Tф
    Kriv_fict = Radius_fict * (3.14159 * Alfa / 180)    ; расчет Kф
    b1 = Tangens_pr - Tangens_fict                      ; расчет b1
    b2 = z * Sin(Alfa_Rad) - b1                         ; расчет b2
    Txt_b1.Text = CStr(b1)                              ; печать b1
    Txt_b2.Text = CStr(b2)                              ; печать b2
End Sub
```

Результаты работы программы по исходным данным примера 1 [2, с. 33] приведены на рисунке 3.6.

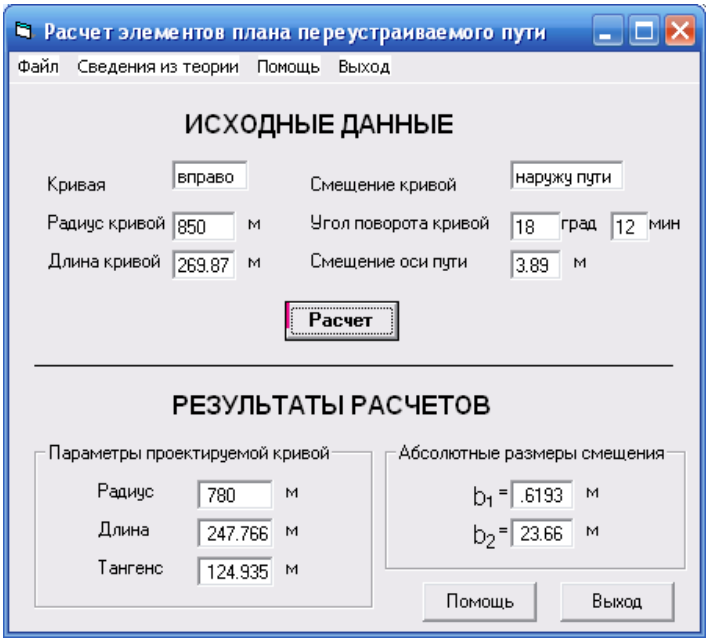


Рисунок 3.6 – Общий вид программной формы с решением задачи

Данную программу можно дополнить формой, на которую вынести сведения, имеющие, например, справочный характер. Поместим рядом с

кнопкой «Выход» кнопку «Помощь» с программным именем CmdHelp. Создадим вторую форму с именем FrmHelp (см. рисунок 2.7).

Связь форм обеспечивается строкой *FrmHelp.Show vbModal,* связанной с кнопкой «Помощь».

На вторую форму помещаем кнопку «Выход» с именем CmdExit. Закрываем данную форму кодом, аналогичным закрытию программы:

```
Unload Me
Set FrmHelp = Nothing
```

Если добавить соответствующую информацию на вторую форму, то запуск программы и последовательное нажатие кнопок «Расчет» и «Помощь» приводит к следующему виду (рисунок 3.7).

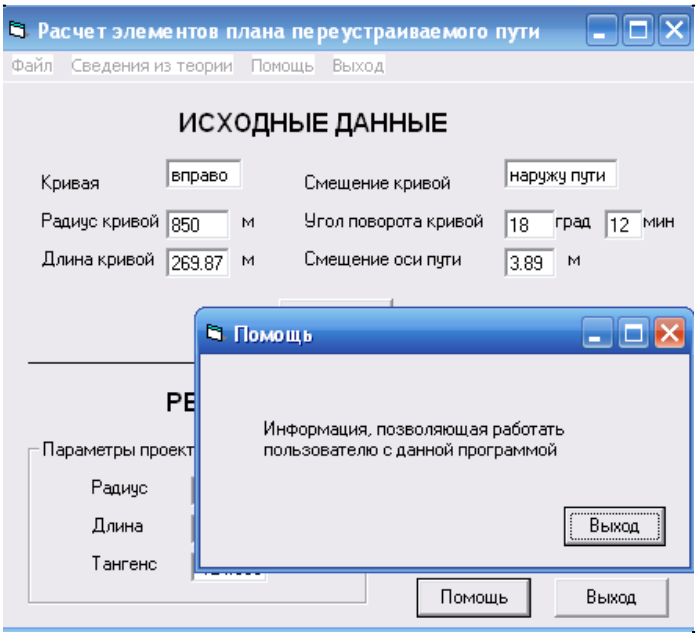


Рисунок 3.7 – Программа с двумя формами

Следует отметить, что вторая форма «Помощь» является модальной, что отражается командой Show vbModal. Если не закрыть данную форму, то никакие другие операции, кроме как на данной форме, невыполнимы.

Немодальная форма определяется командой FrmHelp.Show (без указания vbModal). В данном случае можно работать с другими формами, однако немодальная остается на экране дисплея до момента нажатия на кнопку «Выход», расположенную на этой форме.

Взаимодействие пользователя и программы очень удобно производить с помощью окон диалога (см. раздел 2 пособия). Пусть средствами Visual Basic на форме рисунке 3.3 создано меню (рисунок 3.8).

С опциями «Открыть», «Сохранить…», «Печать» связываем стандартные диалоговые окна FileOpen, Save, Print. Для этого можно поместить на форму элемент управления CommonDialog.

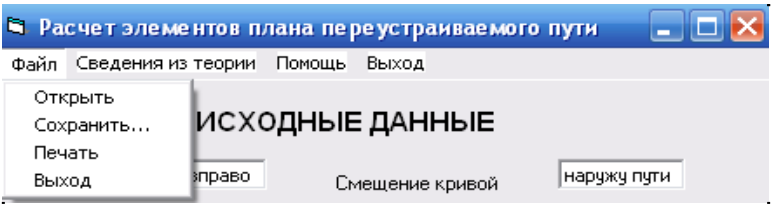


Рисунок 3.8 – Меню основной программной формы

При этом в режиме редактирования следует нажать левую клавишу мыши на строку меню «Открыть» и в открывшемся окне записать следующий код:

CommonDialog1.ShowOpen,

а в редакторе кода для строк «Сохранить…» и «Печать» – соответственно

CommonDialog1.ShowSave

и

CommonDialog1.ShowPrinter.

После выхода из редактора кода и запуска пользовательской программы на исполнение можно вызывать стандартные окна при нажатии на строки меню «Открыть», «Сохранить…» и «Печать». Следует отметить, что после выбора параметров в данных окнах и нажатия на кнопки Open, Save или Print выполнение соответствующей операции не последует, так как необходимо разработать дополнительный программный код для обработки принятых параметров.

На кнопку меню «Выход» можно записать

Unload Me
Set FormPuti = Nothing.

Эти строки позволяют корректно завершить работу программной формы и удалить ее из памяти.

3.2 Разработка программных функций

Выполнение расчетов в программных процедурах представляет собой достаточно трудную задачу. Следует отметить, что результатом работы процедуры удобно иметь некоторое визуально наблюдаемое изменение (появление новых объектов на программной форме, отображение результатов). Проведение расчетов целесообразно выделять в особую категорию программных функций. Поэтому различие между процедурой и функцией проявляется в том, что функция всегда завершает свою работу получением одного или нескольких количественных значений. Ранее мы знакомились со структурой программной процедуры Visual Basic. Аналогично уже известной записи процедуры можно определить соответствующий вид программной функции:

Private Sub RasRadius_Click(),

Private Function DigRadius() As Double.

Процедура не имеет типа результирующего значения, определение функции всегда завершается указанием типа возвращающего аргумента. Кроме того, рядом с именем функции может быть указан один или несколько параметров, которые передаются в данную функцию для проведения расчетов как исходные данные. Например, при расчете плана переустраиваемого пути требуется определить расстояние между центрами существующей и фиктивной кривых по формуле

$$z = y / (1 - \cos\alpha).$$

Тогда для программного расчета параметра z можно разработать функцию следующего вида:

Private z(y As Double, Alfa As Double) As Double.

Особенностью работы функций является их широкое использование. В программе может потребоваться многократный расчет параметра z при различных исходных данных y и *Alfa*. Поэтому необходим доступ к результатам проведения расчетов в других процедурах и функциях. Зарезервированное слово Private указывает на узкую область видимости данной переменной. Чтобы можно было использовать результаты работы функции в других модулях, вместо Private записывают Public:

Public z(y As Double, Alfa As Double) As Double.

Таким образом, полная программная функция определения расстояния между центрами существующей и фиктивной кривых при аналитическом расчете элементов плана переустраиваемого пути может быть:

```
Public z(y As Double, Alfa As Double) As Double
  z = y / (1 – cos(Alfa)
```

End Function

Это наиболее простая функция, состоящая всего из одной строки кода, однако при разработке любой другой сложной функции увеличивается только количество строк. Первая строка наименования функции и последняя строка завершения функции всегда присутствуют в теле любой функции. Все арифметические операции, большинство тригонометрических функций предопределены в Visual Basic и их можно использовать без всякого дополнительного описания и разработки.

Чтобы правильно записать программную функцию, следует в главном окне Visual Basic выбрать в основном меню следующий путь:

Project-Add Module-Открыть.

В результате таких действий рядом с основной программной формой откроется новое окно Module1(Code), а справа вверху в окне проекта программы дополнится еще одной связью –Modules-Module1 (рисунок 3.9).

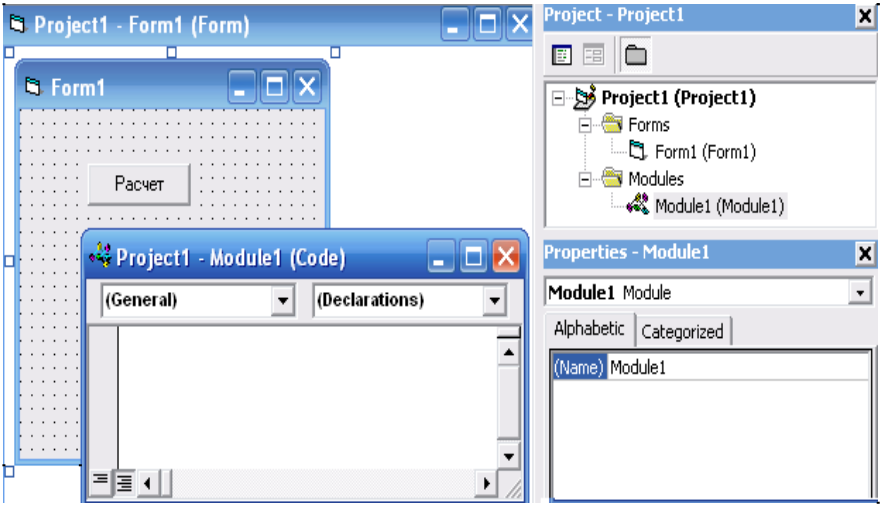


Рисунок 3.9 – Включение функции в состав пользовательской программы

Вынос функциональных модулей за пределы основной программы приводит к облегчению содержания основного поля кода. Целесообразно все расчеты перемещать в отдельный модуль функций.

Кроме переменных входных данных и результата в функции могут использоваться заранее определенные константы и промежуточные параметры. Их следует определять в блоке Dim:

Public z(y As Double, Alfa_Grad As Integer, Alfa_Min As Integer) As Double

```
Const Pi = 3.141593
Dim Alfa As Double
Alfa = (Alfa_Grad + Alfa_Min / 60) * Pi / 180
z = y / (1 – cos(Alfa)
End Function
```

Для использования такой функции в любой другой процедуре следует отдельной строкой кода указать:

Txt_z.Text = CStr(z(3.89, 18, 12)),

и в текстовом окне переменной Txt_z будет показан результат. Программная среда Visual Basic без участия пользователя (и разработчика программы тоже) найдет функцию

z(y As Double, Alfa_Grad As Integer, Alfa_Min As Integer),

произведет соответствующий расчет, передаст полученный результат в текстовое окно переменной и визуально отобразит его. Практическая реализация этого процесса заключается в том, что при нажатии на кнопку «Расчет *z*» рядом в текстовом окне сразу возникнет результат (рисунок 3.10).

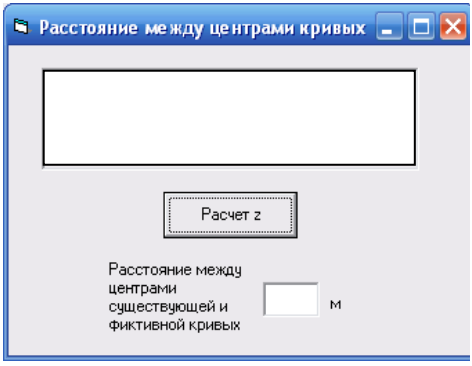


Рисунок 3.10 – Визуализация результата расчета функции на программной форме

При проведении сложных расчетов разрабатываются многочисленные функции. В состав одной программной структуры Module1 может входить несколько функций. Если функции различны по своему назначению, целесообразно формировать другие модули и называть их удобными и легко распознаваемыми именами, например, Module_Radius, Mod-

Пользовательские функции можно оформлять в библиотеки. Описание функции *z* может быть следующим:

z(y, Alfa_Grad, Alfa_Min).

Функция используется для определения расстояния между центрами существующей и фиктивной кривых при аналитическом расчете элементов плана переустраиваемого пути. Возвращает значение типа Double. Имеет три обязательных параметра:

y – разность междупутий типа Double;
Alfa_Grad – градусы угла поворота кривой (тип Integer);
Alfa_Min – минуты угла поворота кривой (тип Integer).

Пример:

```
Dim DeltaPuti As Double
DeltaPuti = z(0.66, 20, 38)
Print DeltaPuti
```

При решении задач изыскания и проектирования железных дорог используются различные методики расчета параметров. Поэтому сведение основных расчетных формул в программные функции и библиотеки функций позволит автоматизировать процесс получения исходных данных для последующего этапа проектирования.

3.3 Стандартные функции Visual Basic

Встроенные функции Visual Basic позволяют быстро и эффективно разработать достаточно сложные пользовательские программы. В данном методическом пособии приведем только те функции, которые наиболее часто применимы для целей автоматизации расчетов и проектирования железных дорог.

Abs(Number)

Функция Abs (Absolute) служит для вычисления модуля числа. В результате действия функции Abs возвращается значение, тип которого совпадает с типом переданного аргумента и равняется абсолютному значению числа, указанного в аргументе. Обязательный аргумент Number может представлять любое допустимое числовое выражение. Если аргумент имеет значение Null или представляет собой не определенную переменную, то возвращается нулевое значение.

Пример:

```
Dim A
Dim B
A = Abs(50.3)        ' возвращается 50.3
B = Abs(-38.4)       ' возвращается 38.4
```

Array(ParamArray)

Функция Array служит для обработки массивов и объявляет переменную типа *Variant*, присваиваемую массиву. Значения аргумента означают размерность массива. Обязательный аргумент представляет собой разделенный запятыми список значений, присваиваемых массиву. Если аргумент ParamArray не указывается – Array(), то создается массив нулевой длины. Номер первого элемента массива начинается с 0.

Пример:

```
Dim A
Dim B
A = Array(5, 7, 12)
B=A(2)          ' в переменную B передается второй элемент массива A
Print B         ' возвращается 7
```

Atn(Number)

Функция Atn определяет арктангенс величины угла в радианах. Возвращает значение типа Double в диапазоне от –pi/2 до pi/2 радиан.

Пример:

```
Dim pi
pi=4*Atn(1)          'вычисляет значение pi
```

CByte(Expression)

Функция CByte (Conversion to Byte) преобразует выражение в новое выражение типа Byte. Обязательный аргумент представляет собой любое числовое выражение или строку, представляющую собой число без знака в диапазоне значений от 0 до 255. Если Expression является дробным числом, CByte округляет его следующим образом:

- если дробная часть меньше 0,5, то она отбрасывается;
- если больше 0,5, то к целой части прибавляется единица;
- если равна точно 0,5, то округляется до ближайшего четного числа;
- если аргумент имеет тип данных String, то он должен представлять число;
- если Expression меньше 0 или больше 255, генерируется ошибка стадии выполнения.

Пример:

```
Dim A, retval
A = 123.4567          ' имеет тип Double
retval = CByte(A)     ' переменная retval содержит 123
Debug.Print retval    ‘ результат появляется внизу в окне Immediate
```

CDate(Expression)

Функция CDate (Conversion to Date) преобразует выражение в новое выражение типа Date. Обязательный аргумент представляет собой любое выражение, выражающее календарную дату. Последовательность элементов даты (день, месяц, год) в аргументе Expression, распознаваемом как дата, определяется установками текущего компьютера.

Пример:

```
Dim A, retval
A = "Июль 27,1966"          ' выбираем дату
```

```
retval = CDate(A)              ' преобразуем результат в значение типа
Date
Debug.Print retval
```

Choose(Index,item1 [, item2 [,..., [itemN]])

Функция Choose выполняет выборку элемента по его порядковому номеру (индексу) в списке элементов, возвращая значение выбранного элемента. Функция Choose возвращает значение Null, если значение Index меньше 1 или больше числа элементов в списке. Если значение Index не целое число, то перед обработкой списка оно округляется до ближайшего целого числа. Обязательный аргумент представляет собой выражение типа Single. Обязательный аргумент item1 представляет собой список типа Variant возвращаемых элементов, разделенных запятой. Необязательные аргументы Item2,...itemN аналогичны аргументу item1.

Пример:

```
Dim retval
retval = Choose(3, "выемка", "насыпь", "косогор")
Debug.Print retval ' косогор          ' результат появляется внизу
                                      ' в окне Immediate
```

CInt(Expression)

Функция CInt (Conversion to Integer) используется для преобразования числа в новое число типа Integer. Функция CInt возвращает значение аргумента, преобразованное в числовой целый тип данных Integer. Дробная часть при этом округляется. Любое числовое выражение Expression или строка представляет собой число от –32768 до 32767. Если числовое значение Expression не попадает в диапазон допустимых значений, происходит ошибка стадии выполнения. Если Expression имеет тип данных String, то он также должен быть числом, иначе возникает ошибка стадии выполнения. Если дробная часть Expression равна 0,5, CInt всегда округляет его до ближайшего четного числа, т. е. CInt(2,5) = 2, а CInt(1,5) = 2.

Пример:

```
Dim A
Dim retval
A = 1234.5678                 ' A имеет тип Double
retval = CInt(A)              ' преобразуем в Integer
Debug.Print retval            ' получаем 1235
```

CLng(Expression)

Функция CLng (Conversion to Long) используется для приведения выражения к типу Long. Она возвращает значение Expression, преобразованное в числовой длинный тип данных Long. Дробная часть при

этом округляется. Expression представляет собой любое числовое выражение или строку в виде числа от −2147483648 до 2147483647. Если Expression имеет тип данных String, то он должен быть числом, иначе возникает ошибка стадии выполнения. Если дробная часть Expression равна 0,5, CLng всегда округляет его до ближайшего четного числа, т. е. CLng(0,5) = 0, а CLng(1,5) = 2.

Пример:

```
Dim A, retval
A = 1234.5678              ' A имеет тип Double
retval = CLng(A)           ' преобразуем в Long
Debug.Print retval         ' получаем 1235
```

Cos(Number)

Функция Cos вычисляет значение косинуса угла. Аргумент задается в радианах. Функция возвращает величину с типом данных Double, лежащую в диапазоне от –1 до 1. Обязательный аргумент Number представляет собой значение типа Double, задающее угол в радианах. Если указанный аргумент не может быть оценен как число, генерируется соответствующая ошибка времени исполнения.

Пример:

```
Dim MyAngle, retval
Dim GradToRad
Const pi = 3.14159265358979     ' константа pi
MyAngle = 64                    ' задаем угол в градусах
GradToRad = MyAngle*pi/180      ' переводим градусы в радианы
retval = Cos(GradToRad)         ' вычисляем косинус прямого угла
Debug.Print retval
```

CStr(Expression)

Функция CStr (Conversion to String) используется для приведения числового выражения в строку (тип String). Функция CStr возвращает значение Expression, преобразованное в строковый тип данных String.

Пример:

```
Dim retval
retval = CStr(1234.56789)       ' преобразуем Double в String
Text1.Text = retval             ' выводим строку в текстовом окне
```

Date()

Функция Date позволяет получить текущую системную дату по календарю компьютера. Функция не имеет аргументов.

Пример:

```
Dim Today
Today = Date                    ' определяем  текущую системную дату
```

Form1.Caption = CStr(Today) ' выводим в заголовке формы

Day(Date)

Функция Day используется для получения дня месяца из даты. Установлено, что отсчет дат начинается с 31 декабря 1899 года, поэтому нулевым днем будет 30 декабря. Возвращается значение типа Variant (Integer) от 1 до 31, которое представляет день месяца. Обязательный аргумент Date представляет собой любое выражение, оцениваемое как дата. Допустимый диапазон дат: от 01.01.0000 до 31.12.9999.

Пример:

Print Day("27 Июля 1966") ' извлекаем день месяца из строки

Dir [(PathName[, Attributes])]

Функция Dir служит для проверки существования каталога или файла, отвечающих заданному образцу. Она поддерживает использование подстановочных знаков для нескольких символов (*) и одиночного символа (?) для указания некоторого шаблона файлов. Функция возвращает первое имя файла, соответствующего аргументу PathName. Для получения остальных файлов, имена которых соответствуют PathName, следует повторно вызвать функцию Dir без аргументов. Последовательные вызовы функции без аргументов возможны до тех пор, пока имеются файлы или папки, соответствующие образцу первого вызова (с аргументами). Рекурсивные (вложенные) вызовы функции Dir запрещены. Функция возвращает данные типа String, структурно представляющего имя файла или папки, которые удовлетворяют указанному шаблону имени файла, набору атрибутов файла или метке тома на диске. Если аргумент PathName не найден, то функция Dir возвращает пустую строку. Необязательный аргумент PathName является строковым выражением, указывающим имя файла. Этот аргумент также может содержать имя каталога или папки и диска. При использовании имен файлов или папок, содержащих пробелы, следует использовать дополнительные кавычки. Например:

Dir(""E:\Тяговые расчеты\t_r.exe"").

PathName является необязательным аргументом, однако он обязателен при первом вызове функции, а также в случаях, если задан аргумент Attributes. Этот параметр определяется как необязательный аргумент, содержащий константу или числовое выражение, описывающее атрибуты файла. Если этот аргумент опущен, возвращаются все файлы, имена которых удовлетворяют содержимому аргумента PathName. Допустимые значения Attributes:

vbNormal = 0 ' обычное состояние файла.
vbReadOnly = 1 ' атрибут только для чтения
vbHidden = 2 ' скрытый атрибут

vbSystem = 4 ' системный атрибут
vbVolume = 8 ' метка тома
vbDirectory = 16 ' каталог или папка

Можно также указывать комбинации атрибутов путем их суммирования. Например, vbHidden + vbDirectory выводит все скрытые папки.

Пример:

```
Dim retval
retval = Dir("e:\user5\geodez\\*.txt")    ' возвращает имя с расширением txt.
                                          ' При нескольких файлах возвращается
                                          ' первый найденный файл
Print retval
```

EOF(FileNumber)

Функция EOF(End Of File) проверяет, достигнут ли конец файла. С помощью этой функции можно избежать ошибок, возникающих при попытках чтения или записи после достижения конца файла. Функция EOF возвращает значение False до тех пор, пока не будет достигнут конец файла. Возвращается значение типа Integer, содержащее логическое значение True при достижении конца файла. При обработке файлов, открытых с помощью Output, функция EOF всегда возвращает True. Обязательный аргумент FileNumber имеет тип Integer, указывающий на любой допустимый номер файла.

Пример:

```
' В данном примере функция EOF
' используется для обнаружения конца файла.
' Предполагается, что MYFILE
' является текстовым файлом,
' который содержит
' несколько строк текста
Dim InputData
Open "MYFILE" For Input As #1          ' открываем файл для чтения
Do While Not EOF(1)                    ' достигнут ли конец файла?
        Line Input #1, InputData       ' читаем строку данных
        Debug.Print InputData          ' выводим в окно отладки
Loop                                   ' закрываем файл
Close #1
```

Error[(ErrorNumber)]

При анализе ошибочной ситуации по заданному номеру ошибки функция Error позволяет получать стандартный текст сообщения об ошибке без необходимости их генерации. Функция возвращает значение типа Variant (String), содержащее стандартное описание ошибки по заданному

номеру. Необязательный аргумент ErrorNumber содержит номер любой доступной ошибки. Если этот номер допустим, но текст ошибки не определен, то функция Error возвращает строку

“Application-defined or object-defined error”.

Пример:

```
Dim ErrorNumber
For ErrorNumber = 51 To 64  ' выводим сообщения,
                            ' соответствующие номерам ошибки
                            ' с циклом по номерам 51-64
  Debug.Print Error(ErrorNumber)
Next ErrorNumber
```

Exp(Number)

Функция Exp используется для вычисления основания натуральных логарифмов *e*. Функция Exp является обратной к функции Log, поэтому ее иногда называют антилогарифмом. Обязательный аргумент Number представляет собой значение типа Double. Аргумент не может превышать число 709 782712893. Если аргумент не может быть оценен как число, то генерируется ошибка времени выполнения.

Пример:

```
retval = Exp(2)
Debug.Print retval
```

Fix(Number)

Функция Fix (Fixed) отбрасывает дробную часть числа и возвращает целое значение. Функция аналогична функции Int. Различие между ними состоит в том, что для отрицательного аргумента функция Int возвращает ближайшее отрицательное целое число, меньшее либо равное указанному, а Fix – ближайшее отрицательное целое число, большее либо равное указанному. Например, функция Int преобразует –8.4 в –9, функция Fix преобразует –8,4 в –8.

Выражение Fix(Number) эквивалентно следующему:

```
Sgn(Number) * Int(Abs(Number)).
```

Функция возвращает значение типа, совпадающего с типом аргумента, который содержит целую часть числа. Обязательный аргумент Number может представлять любое допустимое числовое выражение или число типа Double. Если аргумент имеет значение Null, то возвращается также Null. Если значение аргумента не попадает в диапазон допустимых значений Double, то генерируется ошибка стадии выполнения. Если аргумент имеет тип данных String, то он должен представлять собой число, иначе генерируется ошибка.

Пример:

```
Dim MyNumber
MyNumber = Int(99.8)          ' возвращает 99
MyNumber = Fix(99.2)          ' возвращает 99
MyNumber = Int(-99.8)         ' возвращает –100
MyNumber = Fix(-99.8)         ' возвращает –99
MyNumber = Int(-99.2)         ' возвращает –100
MyNumber = Fix(-99.2)         ' возвращает –99
```

Hour(Time)

Функция Hour используется для получения значения часов из выражения времени. Вне зависимости от формата выражения времени аргумента и системных установок функция Hour всегда возвращает время в 24-часовом формате. Возвращается параметр типа Variant (Integer) от 0 до 23, который переводит часы в значение времени. Обязательный аргумент Time представляет собой любое значение типа Variant, числовое, строковое выражение или любое их сочетание, которое представляет значение времени. Если время содержит значение Null, возвращается значение Null.

Пример:

```
Dim MyTime, MyHour
MyTime = #4:35:17 PM#         ' присваиваем время
MyHour = Hour(MyTime)         ' MyHour содержит 16
Print MyHour
```

Input(Number, [#]FileNumber)

Служит для чтения символьных или байтовых данных из файла, открытого инструкцией Open в режиме текстового ввода или бинарного доступа. Возвращает значение типа String, содержащее символы или байты из файла, открытого в режиме Input или Binary. Обязательный параметр Number представляет собой любое действительное числовое выражение, указывающее число возвращаемых символов или байтов. Параметр FileNumber указывает на любой действительный номер файла. Данные, считываемые с помощью функции Input, обычно записываются в файл с использованием оператора Print # или Put. Эта функция применима только к файлам, открытым в режиме Input или Binary. В отличие от оператора Input #, функция Input возвращает все считанные символы, в том числе запятые, символы возврата каретки, символы перевода строки, кавычки и начальные пробелы. Для файлов, открытых для доступа в режиме Binary, попытка чтения файла с помощью функции Input при возвращении функцией EOF значения True, приводит к ошибке. При чтении двоичных файлов с помощью функции Input следует вместо функции EOF использовать

функции LOF и Loc или применять с функцией EOF оператор Get. Для байтовых данных, содержащихся в текстовых файлах, следует использовать функцию InputB. В этом случае параметр Number указывает число байт (а не символов), которые следует вернуть.

Пример:

```
' В данном примере функция Input
' применяется для последовательного чтения символов
' из файла и вывода их в окно Immediate.
' Предполагается, что текстовый файл TESTFILE
' существует и содержит несколько строк данных
Dim MyChar
Open "TESTFILE" For Input As #1   ' открываем файл
Do While Not EOF(1)               ' цикл до конца файла
  MyChar = Input(1, #1)           ' читаем один символ
  Debug.Print MyChar              ' выводим в окно Immediate
Loop
Close #1                          ' закрываем файл
```

InputBox(Prompt[, Title] [, Default] [, XPos] [, YPos] [, HelpFile, Context])

Функция выводит на экран диалоговое окно с кнопкой закрытия, содержащее заданное сообщение, поле ввода, кнопки OK, Cancel и опционально заголовок и/или кнопку Help, ожидая от пользователя ввода текста или щелчка кнопки. При задании нескольких параметров необходимо использовать функцию InputBox в выражении. Для пропуска некоторых параметров нужно включить соответствующие разделители в виде запятых. Возвращает значение типа String, включающее содержимое окна текста. Функция содержит следующие аргументы:

Prompt – обязательный параметр, представляющий собой строковое выражение, отображаемое как сообщение в диалоговом окне. Максимальная длина параметра Prompt составляет 1024 символа и зависит от ширины используемых символов. Строковое значение Prompt может содержать нескольких физических строк. Для разделения строк допускается использование символа возврата каретки, символа перевода строки или комбинации этих символов;

Title – необязательный параметр в виде строкового выражения, отображаемого в строке заголовка диалогового окна. Если этот параметр опущен, в строку заголовка помещается имя приложения. Максимальное число символов заголовка достигает 50;

Default – необязательный параметр как строковое выражение, отображаемое в окне текста. Представляет собой ответ, используемый по умолчанию, если пользователь не введет другую строку. Если этот параметр опущен, окно текста отображается пустым;

XPos – необязательный параметр, представляющий собой числовое выражение, задающее расстояние по горизонтали между левой границей диалогового окна и левым краем экрана. Если этот параметр опущен, то диалоговое окно выравнивается по центру экрана по горизонтали;

YPos – необязательный параметр в виде числа, задающего расстояние по вертикали между верхней границей диалогового окна и верхним краем экрана. Если этот параметр опущен, то диалоговое окно помещается по вертикали на расстоянии примерно на одну треть высоты экрана;

HelpFile – необязательное строковое выражение, определяющее имя файла справки, содержащего контекстно-зависимую информацию о данном диалоговом окне. Если этот параметр указан, то необходимо задать также параметр Context;

Context – необязательное числовое выражение, определяющее номер соответствующего раздела справочной системы. Если этот параметр указан, то необходимо задать также параметр HelpFile.

Если указаны параметры HelpFile и Context, то пользователь имеет возможность нажатием клавиши F1 вызвать соответствующую контекстную справку. Некоторые главные приложения, например, Microsoft Excel, также автоматически добавляют в диалоговое окно кнопку «Справка». Если пользователь щелчком мыши активизирует кнопку OK или нажимает ENTER, функция InputBox возвращает содержимое поля ввода. Если пользователь нажимает на кнопку Cancel, то функция возвращает пустую строку. Если пользователь не введет в поле ввода строку, то функция также вернет пустую строку. В этом случае будет затруднительно определить, какую кнопку (OK или Cancel) нажал пользователь. В действительности Cancel возвращает параметр vbNullString, который воспринимается средой Visual Basic как пустая строка. Можно воспользоваться недокументированной функцией StrPtr и получить указатель на строку. Например, так

```
Dim strInput As String
strInput = InputBox("")
If StrPtr(strInput) = 0 Then
        MsgBox "Нажата кнопка Cancel"
End If
```

Пример:

```
Dim Message, Title, Default, MyValue
Message = "Укажите значение междупутья"
Title = "Расчет уширения междупутья"     ' заголовок
Default = "1"                            ' значение по умолчанию
MyValue = InputBox(Message, Title, Default)
' Получение справки. Кнопка "Справка" добавляется автоматически
```

MyValue = InputBox(Message, Title, , , , "DEMO.HLP", 10)
' Размещаем верхний левый угол окна диалога в точке 100, 100
MyValue = InputBox(Message, Title, Default, 100, 100)

Int(Number)

Функция Int(Integer) отбрасывает дробную часть числа и возвращает целое значение. Функция аналогична функции Fix. Различие между ними состоит в том, что для отрицательного значения аргумента число функция Int возвращает ближайшее отрицательное целое число, меньшее либо равное указанному, а Fix – ближайшее отрицательное целое число, большее либо равное указанному. Например, функция Int преобразует –8,45 в –9, а функция Fix преобразует –8,4 в –8. Функция возвращает значение типа, совпадающего с типом аргумента, которое содержит целую часть числа. Обязательный аргумент Number может представлять любое допустимое числовое выражение или число типа Double. Если аргумент имеет значение Null, то возвращается также Null. Если значение аргумента не попадает в диапазон допустимых значений Double, то генерируется ошибка стадии выполнения. Если аргумент имеет тип данных String, то он должен представлять собой число, иначе генерируется ошибка.

Пример:

Dim MyNumber
MyNumber = Int(99.8) *' Возвращает 99*
MyNumber = Fix(99.2) *' Возвращает 99*
MyNumber = Int(-99.8) *' Возвращает –100*
MyNumber = Fix(-99.8) *' Возвращает –99*
MyNumber = Int(-99.2) *' Возвращает –100*
MyNumber = Fix(-99.2) *' Возвращает –99*

IsArray(VarName)

Служит для проверки, является ли переменная массивом. Функция IsArray часто используется для переменных типа Variant, содержащих массивы. Возвращает значение типа Boolean и указывает, является ли переменная массивом. Если является, то возвращается True, иначе – False. Обязательный параметр VarName является идентификатором, задающим переменную.

Пример:

Dim MyArray(1 To 5) As Integer, YourArray, MyCheck
YourArray = Array(1, 2, 3)
MyCheck = IsArray(MyArray) *' Возвращает True*
MyCheck = IsArray(YourArray) *' Возвращает True*

Join(SourceArray[, Delimiter])

Данная функция служит для слияния элементов заданного массива в одну строку со вставкой между ними необязательного разделителя. Возвращает строку, созданную путем объединения множества подстрок, содержащихся в массиве с разделителем между ними. SourceArray представляет собой обязательный аргумент в виде одномерного массива, содержащего объединяемые подстроки. Delimiter – необязательный аргумент, являющийся символом, используемым для разграничения подстрок в возвращаемой строке. Если параметр опущен, применяется символ пробела. Если аргумент является строкой нулевой длины, то производится конкатенация (слияние) всех элементов списка без добавления разделителей.

Пример:

```
Dim vArr(1 to 2)
Dim retval as String
vArr(1) = "Условия"
vArr(2) = "проектирования"
retval = Join(vArr,"_")
Print retval
```

LCase(String)

Функция LCase (Lower Case) преобразует символы строки в строчные. Все буквы в нижнем регистре и небуквенные символы остаются неизмененными. Вместо LCase можно использовать универсальную функцию преобразования строк StrConv. В качестве результата возвращается строка, которая отличается от входной тем, что все буквы преобразованы в строчные. Если параметр String содержит Null, то и возвращаемым значением является Null. Обязательный аргумент String может представлять любое строковое выражение.

Пример:

```
Form1.Caption = LCase("ПЕРЕХОДНАЯ КРИВАЯ")
```

Left(String, Length)

Функция Left служит для уменьшения длины исходной строки до заданной. Для определения числа символов в строке следует использовать функцию Len. Возвращает строку прочитанных данных. Данные могут быть символами или байтами. Функция содержит следующие аргументы:

String – обязательный аргумент со значением типа Variant (Long). Число указывает количество возвращаемых символов. Если оно равно 0, то возвращается пустая строка. Если значение Length больше либо равно числу символов в исходной строке, то возвращается вся строка. При отрицательных значениях аргумента генерируется ошибка. Если аргумент Null, то также генерируется ошибка;

Length – обязательный аргумент в виде строки, из которой извлекаются символы. Если выражение имеет значение Null, то возвращается Null.

Пример:

```
Dim strMyString, retval
strMyString = "Указать параметры проектирования"
retval = Left(strMyString, 7)        ' Возвращает "Указать"
retval = Left(strMyString, 17)       ' Возвращает "Указать параметры"
```

Len(String | Varname)

Функция Len вычисляет число символов в строке или размер заданной переменной. Из двух возможных аргументов должен быть указан только один. Для определяемых пользователем типов Len возвращает размер, который требуется для записи переменной в файл. В результате вычислений возвращается значение типа Long, содержащее число символов в строке или число байт, необходимое для размещения переменной. Обязательный аргумент String – любое допустимое строковое выражение. Varname – обязательный аргумент, представляющий собой любое допустимое имя переменной. Если выражение имеет значение Null, то возвращается Null. Если же аргумент имеет значение типа Variant, то функция Len обрабатывает его так же, как и значение типа String, и всегда возвращает число содержащихся в нем символов.

Пример:

```
Dim A
Dim B
' Вычисляем длину указанной строки
Print Len("Основные параметры расчета")
```

LoadPicture([FileName], [Size], [ColorDepth], [X, Y])

Используя свойство Picture, функция загружает графическое изображение в форму или элементы управления PictureBox и Image. Visual Basic различает следующие форматы графических файлов: растровые файлы (.bmp), значки (.ico), файлы (.rle) с групповым кодированием (run-length encoded), метафайлы (.wmf), улучшенные метафайлы (.emf), файлы типа GIF (.gif) и JPEG (.jpg). Функция использует следующие параметры:

FileName – необязательный параметр в виде некоторой строковой переменной, определяющей имя загружаемого графического файла. Может включать имя папки и диска. Если не указывается название (наименование) файла, то функция LoadPicture очищает элементы управления Image или PictureBox;

Size – необязательный (опционный) параметр типа Variant. Если FileName является курсором или значком файла, то параметр определяет

желаемый размер требуемого изображения. Допустимые значения приведены в таблице 3.1.

Т а б л и ц а 3.1 – **Установки необязательного параметра Size**

Константа	Значение	Описание
vbLPSmall	0	Системный значок малого размера
vbLPLarge	1	Системный значок большого размера
vbLPSmallShell	2	Значок малого размера для оболочки
vbLPLargeShell	3	Значок большого размера для оболочки
vbLPCustom	4	Определяемый размер, задается аргументами X и Y

ColorDepth – необязательный параметр типа Variant. Если FileName является курсором или значком файла, то данный параметр определяет желаемую глубину цвета. Допустимые значения приведены в таблице установок 3.2.

Т а б л и ц а 3.2 – Установки необязательного параметра ColorDepth

Константа	Значение	Описание
vbLPDefault	0	Оптимальное значение при использовании определенных файлов
vbLPMonochrome	1	2 цвета
vbLPVGAColor	2	16 цветов
vbLPColor	3	256 цветов

X – необязательный параметр типа Variant. Этот параметр необходимо указывать, если задан параметр Y. Если FileName является курсором или значком файла, то параметр X определяет желаемую ширину. Значения X и Y используются, если установлено значение vbLPCustom аргумента ColorDepth. Для значков файлов наибольшее возможное значение 255.

Y – необязательный параметр типа Variant. Обязателен, если указан параметр X. Если FileName является курсором или значком файла, то параметр определяет желаемую высоту.

Чтобы загрузить графическое изображение в элементы управления PictureBox и Image или же использовать его в качестве фона формы, следует присвоить возвращаемое значение функции LoadPicture значению свойства Picture данных элементов. Например:

Set Picture = LoadPicture("PARTY.BMP"),
Set Picture1.Picture = LoadPicture("PARTY.BMP").

Чтобы прикрепить к форме значок, следует приравнять значение свойства Icon данной формы значению, возвращаемому функцией LoadPicture:

Set Form1.Icon = LoadPicture("MYICON.ICO").

Значки могут быть присвоены для свойства DragIcon любого элемента управления, за исключением элементов управления Timer и Menu. Например:

Set Command1.DragIcon = LoadPicture("MYICON.ICO").

С помощью функции LoadPicture можно загрузить графический файл в системный буфер обмена:

Clipboard.SetData LoadPicture("PARTY.BMP").

LoadResData(id, type)

Загружает данные нескольких возможных типов из файла ресурсов (.res). Функция LoadResData может загружать до 64Кб данных. Использование функции LoadResData облегчает локализацию приложений среды проектирования Visual Basic. Возвращает массив данных типа Byte. Имеем два обязательных параметра:

id – целое число или строка, которые определяют идентификатор (id) данных в файле ресурсов. Ресурс с идентификатором, равным 1, зарезервирован для значка приложения;

type – значение, которое указывает фиксированный формат возвращаемых данных. Значение может быть также строкой, задающей имя определенного пользователем ресурса. Допустимые значения параметра type представлены в таблице 3.3.

При использовании функции LoadResData с ресурсами растр, значок и курсор возвращается строка, содержащая число бит в ресурсе. При работе с этими ресурсами следует применять функцию LoadResPicture.

Т а б л и ц а 3.3 – **Установки обязательного параметра Type**

Установка	Описание	Установка	Описание
1	Ресурс «Курсор»	7	Ресурс «Каталог шрифта»
2	Ресурс «Растр»	8	Ресурс «Шрифт»
3	Ресурс «Значок»	9	Таблица «Акселератор»
4	Ресурс «Меню»	10	Определяется пользователем
5	Ресурс «Диалоговое окно»	12	Групповой курсор
6	Ресурс "Строка"	13	Групповой значок

LoadResPicture(id, restype)

Загружает растр, значок или курсор из файла ресурсов (.res). Функция использует два обязательных параметра:

id – целое число или строка, указывающие на идентификатор данных в файле ресурсов;

restype – значение или константа, которые указывают формат возвращаемых данных. Допустимые значения представлены в таблице 3.4.

Таблица 3.4 – **Установки обязательного параметра Restype**

Константа	Значение	Описание
vbResBitmap	0	Ресурс «Растр»
vbResIcon	1	Ресурс «Значок»
vbResCursor	2	Ресурс «Курсор»

LOF(FileNumber)

Функция LOF (Len Of File) определяет длину файла (количество символов), открытого инструкцией Open, в байтах. Однако для определения фиксированного размера файла, который не открыт, рекомендуется использовать функцию FileLen. Функция возвращает значение типа Long, содержащее размер в байтах для данного файла, открытого с помощью оператора Open. Параметр FileNumber является обязательным со значением типа Integer, в котором содержится действительный номер файла.

Пример:

```
Dim FileLength
Open "TESTFILE" For Input As #1      ' Открываем файл
FileLength = LOF(1)                  ' Определяем размер файла
Close #1                                   ' Закрываем файл
```

Log(Number)

Функция вычисляет значение натурального логарифма. Возвращает значение типа Double, содержащее натуральный логарифм числа. Параметр Number является значением типа Double или любым действительным числовым выражением, значение которого больше нуля. Если аргумент не является числом, то генерируется ошибка. Если аргумент равен нулю или отрицателен, то также генерируется ошибка.

Пример:

```
Static Function Log10(X)
Log10 = Log(X) / Log(10#)
End Function
```

LTrim(String)

Функция LTrim (Left Trim) удаляет пробелы в начале строки. Возвращает исходную строку типа Variant (String) без начальных пробелов. Если аргумент имеет значение Null, то возвращается Null. Параметр String является обязательным и может представлять любое строковое выражение.

Пример:

```
Dim retval
retval=LTrim("  <-Радиус кривой->  ")
Print retval
```

Mid(String, Start, [Length])

Функция Mid (Middle) используется для считывания заданного числа символов или байтов подряд от заданной позиции в строке слева направо. Нумерация символов в строке всегда начинается с единицы. Для определения числа символов в строке следует использовать функцию Len. Возвращает значение типа Variant (String), содержащее указанное число символов строки. Функция содержит следующие аргументы:

String – обязательный аргумент в виде строки, из которой извлекаются символы. Если аргумент имеет значение Null, возвращается Null;

Start – обязательный аргумент со значением типа Long. Указывает на позицию символа в строке String, с которого начинается нужная подстрока. Если Start больше числа символов в строке String, то функция Mid возвращает пустую строку. Если аргумент имеет отрицательное значение или равен 0, то генерируется ошибка;

Length – необязательный аргумент, имеющий значение типа Variant (Long). Указывает число возвращаемых символов. Если этот аргумент опущен или превышает число символов, расположенных справа от позиции Start, то возвращаются все символы от позиции Start до конца строки.

Пример:

```
Dim Mystring, retval
Mystring = "Тяговые расчеты"
retval = Mid(Mystring, 1, 3)
Print retval ' возвращает Тяг
```

Minute(Time)

Функция Minute используется для получения значения минут часа из выражения времени. Возвращает значение типа Variant(Integer) от 0 до 59, содержащее число минут неполного часа. Функция имеет обязательный параметр Time, представляющий собой любое значение типа Variant, числовое или строковое выражение, любое их сочетание, которое представляет значение времени. Если время содержит значение Null, возвращается значение Null.

Пример:

Dim MyMinute
MyTime = Now
MyMinute = Minute(MyTime)
Print MyMinute

Month(Date)

Функция Month получает номер месяца для заданной даты. Возвращает значение типа Variant(Integer), содержащее целое число (от 1 до 12 включительно), которое представляет месяц в значении даты. Параметр Date является обязательным аргументом и может быть любым значением типа Variant, числовым выражением, строковым выражением или любым их сочетанием, которое представляет дату. Если дата содержит значение Null, возвращается значение Null.

Пример:

Form1.Caption = Month("27/07/66") ' порядковый номер месяца июля

MonthName(Month[, Abbreviate])

Функция получает полное или сокращенное название месяца по его номеру. Возвращает строку, содержащую полное или сокращенное название месяца. Параметр Month является обязательным аргументом Long, определяющим номер месяца (1-12). Abbreviate – необязательный аргумент типа Boolean, указывающий на вид возвращаемого месяца. По умолчанию установлен в False, что означает вывод полного имени месяца.

Пример:

Dim A
Form1.Caption = MonthName(7) ' полное название седьмого месяца

MsgBox(Prompt, [Buttons], [Title], [HelpFile], [Context])

Функция выводит на экран диалоговое окно, содержит сообщение и устанавливает режим ожидания нажатия кнопки пользователем. Следует отметить широкое распространение данной функции как удобное средство разработки интерфейса связи с пользователем программы. Возвращает значение типа Integer, указывающее, какая кнопка была нажата. Данная функция имеет следующие параметры:

Prompt – обязательный в виде строкового выражения, отображаемого как сообщение в диалоговом окне. Максимальная длина строки составляет приблизительно 1024 символа. Длинный текст разбивается автоматически, но можно задавать разбиение строки явно, используя символы возврата каретки и перевода строки (vbCrLf);

Buttons – необязательный аргумент как целочисленная константа, которая является суммой VB-констант, определяющих ряд характеристик

диалогового окна (число и тип кнопок, тип значка, основная кнопка, модальность окна сообщения). Значение по умолчанию равно 0. Константы, используемые в аргументе Buttons для задания вида выводимых кнопок, приведены в таблице 3.5.

Title – необязательный параметр в виде некоторого строкового выражения, отображаемого в верху строки заголовка диалогового окна. Максимальное число символов для заголовка принимается около 50. Этот параметр может быть опущен, тогда в строку заголовка помещается имя приложения;

HelpFile – необязательный параметр, представляющий собой строковое выражение, определяющее имя файла справки. Этот параметр содержит контекстно-зависимую справку о данном диалоговом окне. Если он указан, то необходимо задать также и параметр Context;

Context – необязательный параметр, рассматриваемый как соответствующее числовое выражение, определяющее номер соответствующего раздела справочной системы (или любое строковое выражение, определяемое разработчиком программы. Если этот параметр указан, то необходимо задать также параметр HelpFile. В таблице 3.6 указаны значения, содержащие код нажатой кнопки.

Если используется кнопка «Отмена» (Cancel), то можно вместо нее нажимать на клавишу Esc. В отличие от InputBox окно MsgBox не позиционируется на экране, а всегда располагается в центре экрана.

Т а б л и ц а 3.5 – Значения констант аргумента Buttons

Константа	Значение	Описание
vbOKOnly	0	В окне функции отображается только кнопка "OK"
vbOKCancel	1	То же "OK" и "Отмена" (Cancel)
vbAbortRetryIgnore	2	" "Прервать" (Abort), "Повторить" (Retry) и "Пропустить" (Ignore)
vbYesNoCancel	3	" "Да" (Yes), "Нет" (No) и "Отмена" (Cancel)
vbYesNo	4	" "Да" (Yes) и "Нет" (No)
vbRetryCancel	5	" "Повторить" (Retry) и "Отмена" (Cancel)
vbCritical	16	Информационный значок "Критическое сообщение"
vbQuestion	32	То же "Предупреждающий запрос"
vbExclamation	48	" "Предупреждение"

vbInformation	64	" "Информационное сообщение"
vbDefaultButton1	0	Основной в программном окне является первая кнопка
vbDefaultButton2	256	То же вторая кнопка
vbDefaultButton3	512	" третья кнопка
vbDefaultButton4	768	" четвертая кнопка
vbApplicationModal	0	Модальное окно на уровне программного приложения
vbSystemModal	4096	Модальное окно на уровне системы
vbMsgBoxHelpButton	16384	Добавляется информационная кнопка Справка (Help)
VbMsgBoxSetForground	65536	Аналогично программному параметру vbApplicationModal
vbMsgBoxRight	524288	Текст выравнивается по правому краю поля сообщения
vbMsgBoxRtlReading	1048576	Текст для формирования строки арабской системы и иврита

Т а б л и ц а 3.6 – **Значения кода нажатой кнопки функции MsgBox**

Константа	Значение	Нажатая кнопка
vbOK	1	OK
vbCancel	2	Отмена (Cancel)
vbAbort	3	Прервать (Abort)
vbRetry	4	Повторить (Retry)
vbIgnore	5	Пропустить (Ignore)
vbYes	6	Да (Yes)
vbNo	7	Нет (No)

Пример:

```
Dim A
Dim retval As Integer
retval = MsgBox("Определить условия проектирования" _
vbQuestion + vbYesNo, "по выбранному варианту")
```

Now()

Функция Now позволяет быстро узнать текущую дату и время по системному календарю и часам компьютера. Чтобы получить только

системную дату без времени, необходимо использовать функцию Date. При этом можно самостоятельно устанавливать системное время и дату на своем компьютере, следовательно, параметры даты и времени могут отличаться от реальной даты и времени. Такая возможность может оказаться весьма полезной при моделировании процессов проектирования в реальном масштабе времени. Возвращает значение типа Variant (Date), содержащее текущую дату и время. Функция не имеет аргументов.

Пример:

```
Dim A
Dim Today
Today = Now                ' текущая системная дата и время
Print CStr(Today)          ' преобразование в строку и вывод на форме
```

QBColor(Color)

Функция QBColor получает RGB-код цвета из кодировки цвета. Следует обратить внимание, что функция RGB обладает большими возможностями для управления цветом. Функция QBColor возвращает значение типа Long, содержащее RGB-код заданного цвета. Обязательный аргумент Color типа Integer может представлять целое число в диапазоне от 0 до 15. Данная функция может быть полезна при оформлении внешнего вида программы при выборе цвета объектов (шрифта, обрамлений и др.). В таблице 3.7 указаны коды цветовой палитры.

Т а б л и ц а 3.7 – Бейсик-кодировка параметра Color

Номер	Цвет	Номер	Цвет
0	Черный	8	Светло-серый
1	Темно-синий	9	Синий
2	Темно-зеленый	10	Зеленый
3	Бирюзовый	11	Голубой
4	Малиновый	12	Красный
5	Сиреневый	13	Розовый
6	Оливковый	14	Желтый
7	Темно-серый	15	Белый

Пример:

```
Form1.BackColor = QBColor(9)          ' фон формы синего цвета
```

Replace(Expression, Find, Replace[, Start[, Count[, Compare]]])

В результате действия функции возвращается исходная строка с замененным строковым фрагментом. Используются следующие параметры:

Expression – обязательный аргумент-строка, в котором требуется замена;
Find – обязательный аргумент-подстрока, который нужно заменить;
Replace – обязательный аргумент-подстрока замены;
Start – необязательный аргумент, указывает позицию замены;
Count – необязательный аргумент, указывающий число замен;
Compare – необязательный аргумент, определяющий вид сравнения.

Пример:

```
Dim sample$, findstr$, newstr$, retval$
sample = "Равнинный профиль"          ' строка с заменой
findstr = "Равнинный"                 ' подстрока для замены
newstr = "Пересеченный"               ' новая подстрока для замены
retval = Replace (sample, findstr, newstr)   ' замена
Debug.Print retval
```

RGB(Red, Green, Blue)

Функция RGB получает RGB-код цвета из основных компонентов. Возвращает значение типа Long, представляющее цвет в модели RGB. Значение RGB указывает относительную интенсивность красного, зеленого и синего компонентов, образующих отображаемый цвет. RGB-код цвета получается из компонентов по формуле

*RGB = red + (green * 256) + (blue * 65536).*

Параметры Red ,Green, Blue представляют собой аргументы типа Variant (Integer). Числа в интервале от 0 до 255 включительно, представляющие соответственно красный, зеленый и синий компоненты цветов. Значение любого аргумента функции RGB, превышающее 255, считается равным 255.

Пример:

```
Dim Red
Red = RGB(255,0,0)                ' устанавливаем красный цвет
Form1.BackColor = Red             ' красный фон формы
```

Right(String, Length)

Функция Right служит для отсечения справа исходной строки заданной длины. Число символов в строке определяется с помощью функцию Len. Возвращает значение типа Variant (String), содержащее указанное число последних символов или байтов строки. Функция содержит параметры:

String – строковое выражение, из которого извлекаются символы. Если выражение имеет значение Null, то возвращается Null;

Length – обязательный аргумент, имеющий значение типа Variant (Long). Числовое выражение, указывающее число возвращаемых символов. Если равно 0, то возвращается пустая строка. Если значение Length больше либо

равняется числу символов в исходной строке, то возвращается вся строка. При отрицательных значениях аргумента генерируется ошибка.

Пример:

```
Dim strMyString, retval
strMyString = "Автоматизированное_проектирование"
retval = Right(strMyString, 14)    ' "проектирование"
retval =Right(strMyString, 33)   ' "Автоматизированное проектирование"
```

Rnd[(Number)]

Функция Rnd (Random) служит для генерации случайных чисел, возвращает значение в диапазоне от 0 до 1 типа Single, содержащее случайное число (1 не входит в этот диапазон, 0 – входит). Строго говоря, функция возвращает псевдослучайные числа. При каждом запуске программы функция генерирует одну и ту же последовательность. Во избежание этого явления нужно использовать инструкцию Randomize. Чтобы получить значения случайных чисел в интервале от min до max, следует применять формулу:

*Int((max – min + 1) * Rnd + min),*

где *min* и *max* – соответственно минимальное и максимальное числа диапазона. Параметр Number является необязательным аргументом и представляет собой число типа Single или любое допустимое числовое выражение.

Пример:

```
Dim MyValue
MyValue = Int((7 * Rnd) + 1)        ' генерируем случайные числа от 1 до 7
Print MyValue                       ' выводим число на форме
```

Round(Number [, NumDigitAfterDecimal])

Функция служит для округления чисел до заданной точности (число значащих цифр в дробной части). Данная Функция Round(Number, 2) эквивалентна функции Format (Number,"#.##"). В результате действия функции возвращается округленное число, тип которого совпадает с типом переданного аргумента. Параметр Number является обязательным аргументом и может представлять любое допустимое числовое выражение. Параметр NumDigitAfterDecimal – необязательный аргумент, представляющий собой целое положительное число, которое указывает количество знаков после запятой. Если аргумент опущен, то дробная часть отбрасывается.

Пример:

```
Dim A, retval
A = 123.456789
```

```
retval = Round(A)                ' возвращается 123
retval = Round(A, 3)             ' возвращается 123.457
```

RTrim(String)

Функция RTrim (Right Trim) удаляет пробелы в конце строки. Возвращает исходную строку типа Variant (String) без завершающих пробелов. Если аргумент имеет значение Null, то возвращается Null. Параметр String является обязательным аргументом в виде любого строкового выражения.

Пример:

```
Dim retval
retval = RTrim("  <-ПРОДОЛЬНЫЙ ПРОФИЛЬ->                    ")
Print retval
```

Second(Time)

Функция Second используется для получения значения секунд минуты из выражения времени. Возвращает значение типа Variant (Integer) от 0 до 59, содержащее число секунд неполной минуты. Параметр Time – обязательный аргумент, представляющий собой любое значение типа Variant, числовое или строковое выражения, которые представляют значение времени. Если время содержит значение Null, возвращается значение Null.

Пример:

```
Dim MyTime, MySecond
MyTime = #4:35:17 PM#            ' присваиваем время
MySecond = Second(MyTime)        ' MySecond содержит 17
Print MySecond
```

Seek(FileNumber)

Функция Seek определяет текущее положение указателя чтения или записи внутри файла, открытого с помощью инструкции Open. Возвращает значение типа Long в интервале от 1 до 2 147 483 647 включительно, определяющее текущее положение указателя чтения/записи внутри файла, открытого с помощью инструкции Open. Кроме того, могут быть использованы инструкции Binary, Output, Append, Input. Первому байту файла соответствует номер 1, второму 2 и т. д. Параметр FileNumber является обязательным и является выражением типа Integer, содержащим допустимый номер файла.

Пример:

```
 'В данном примере функция Seek
' используется для определения текущего
' положения указателя внутри файла.
' Предположим, что файл TESTFILE содержит
```

```
' несколько записей определенного
' пользователем типа Record
Type Record                 ' Тип, определенный пользователем
        ID As Integer
        Name As String * 20
End Type
                            ' Для файлов, открытых в режиме Random,
                            ' Seek возвращает номер следующей записи
Dim MyRecord As Record      ' Объявляем переменную
Open "TESTFILE" For Random As #1 Len = Len(MyRecord)
Do While Not EOF(1)         ' Цикл до конца файла
        Get #1, , MyRecord  ' Читаем следующую запись
        Debug.Print Seek(1) ' Выводим номер записи в окно
Loop
Close #1                    ' Закрываем файл
' Для файлов, открытых в других режимах,
' Seek возвращает номер байта, с которого
' будет начато выполнение следующей операции.
' Предположим, что файл TESTFILE содержит
' несколько строк текста
Dim MyChar
Open "TESTFILE" For Input As #1          ' Открываем файл для чтения
Do While Not EOF(1)                      ' Цикл до конца файла
        MyChar = Input(1, #1)            ' Читаем следующий символ данных
        Debug.Print Seek(1)              ' Выводим номер байта в окно
Loop
Close #1                                 ' Закрываем файл
```

Sgn(Number)

Функция Sgn определяет знак аргумента и возвращает величину с типом данных Variant (Integer) согласно приведенному примеру:

Пример:

```
Dim Znak
Znak = Sgn(27)          ' возвращает 1
Znak = Sgn(-66)         ' возвращает –1
Znak = Sgn(-0,3)        ' возвращает –1
Znak = Sgn(0)           ' возвращает 0
```

Shell(PathName, [WindowStyle])

Функция Shell служит для запуска другой программы из VB-программы. При успешном запуске программы возвращает значение типа Variant

(Double), представляющее идентификатор программы. Если функция Shell не может запустить указанную программу, то возникает ошибка. Параметр PathName является обязательным аргументом типа Variant (String). Имя выполняемой программы, любые требуемые аргументы, ключи командной строки допускают включение каталога, папки и диска. Функция содержит следующие аргументы:

WindowStyle – необязательный аргумент типа Variant (Integer), соответствующий типу окна, в котором выполняется программа. Именованный аргумент WindowStyle имеет значения, приведенные в таблице 3.8.

Таблица 3.8 – **Значения констант аргумента WindowStyle**

Константа	Значение	Описание
vbHide	0	Окно скрыто, фокус передается скрытому окну
vbNormalFocus	1	Окно имеет фокус и восстанавливает свои стандартные размер и положение
vbMinimizedFocus	2	Окно отображается в виде значка с фокусом
vbMaximizedFocus	3	Окно развертывается на экран с фокусом
vbNormalNoFocus	4	Восстанавливаются предыдущие
vbMinimizedNoFocus	6	Окно отображается в виде значка.

Пример:

```
Dim RetVal
RetVal = Shell("D:\Izyskanie\lokomotive.exe", 1)
```

Sin(Number)

Функция вычисляет значение синуса угла в радианах и возвращает величину с типом данных Double, лежащую в диапазоне от –1 до 1. Обязательный аргумент Number представляет собой значение типа Double, задающее величину угла в радианах. Если аргумент не может быть оценен функцией как число, то генерируется ошибка.

Пример:

```
Dim MyAngle
Dim retval
Dim GradToRad
Const pi = 3.14159265358979
MyAngle = 44
GradToRad = MyAngle*pi/180
retval = Sin(GradToRad)
Debug.Print retval
```

Space(Number)

Функция Space используется для создания строки с указанным числом пробелов. Функция возвращает значение типа Variant(String), содержащее указанное число пробелов. Number представляет собой любое положительное целое число. Если аргумент равен нулю, то возвращается пустая строка. Дробные значения аргумента округляются. При отрицательных значениях аргумента генерируется ошибка.

Пример:

Dim MyString

MyString = "ГЛАВНЫЙ" & Space(2) & "ХОД" ' 2 пробела между словами

MsgBox MyString

Split(Expression, [Delimiter], [Limit], [Compare])

Функция Split используется для разбиения строки на субстроки с использованием разделителя субстрок. Функция возвращает одномерный массив с типом данных Variant (String), содержащий в качестве элементов найденные субстроки. Функция использует следующие параметры:

Expression, являющийся обязательным аргументом в виде строки, которую нужно разделить. Если аргумент содержит нулевую строку, то возвращается пустой массив;

Delimiter, относящийся к необязательным (опциональным) аргументам, представляет собой символы типа String, которые используются в качестве разделителя строки. Если аргумент опущен, то используется символ пробела. Если аргумент содержит нулевую строку, то возвращается массив, содержащий целую строку;

Limit – необязательный аргумент, содержащий число возвращаемых субстрок (частей основной строки). Если аргумент опущен или равен –1, то обрабатывается вся строка;

Compare – необязательный аргумент в виде числовой константы, определяющей вид сравнения (таблица 3.9).

Таблица 3.9 – **Значения констант аргумента Compare**

Константа	Значение	Описание
vbUseCompareOption	-1	Используется по умолчанию метод сравнения, заданный инструкцией Option Compare
vbBinaryCompare	0	Двоичный метод сравнения. В таком случае буквы разных регистров считаются разными
		Текстовый метод сравнения.

vbTextCompare	1	Сравнение производится без учета регистров
vbDatabaseCompare	2	Только для Microsoft Access. Сравнение с использованием информации базы данных

Sqr(Number)

Функция Sqr вычисляет величину квадратного корня некоторого значения Number. Возвращает значение типа Double.

Str(Number)

Функция Str (String) используется для приведения числового выражения типа Long в строку (тип String). Функция возвращает значение Number, преобразованное в текстовый тип String. Если число положительно, то в этом месте будет зафиксирован пробел, если число отрицательно, то выводится знак минус.

В качестве десятичного разделителя дроби функция Str воспринимает только точку. При использовании других десятичных разделителей (например, запятой) следует использовать функцию CStr.

Пример:

```
Dim retval
retval = Str(123)          ' результат " 123"
retval = Str(–12.3)        ' результат "–12.3"
```

StrComp(String1, String2[, Compare])

Функция StrComp служит для сравнения двух строк. Возвращает значение типа Variant (Integer), представляющее результат сравнения указанных строк. При этом строки должны иметь ненулевые значения. Функция содержит следующие аргументы:

String1, String2 – обязательные аргументы в виде любых допустимых строковых выражений;

Compare – необязательный аргумент, указывающий на способ сравнения строк (таблица 3.10).

Т а б л и ц а 3.10 – **Возвращаемые значения аргумента Compare**

Константа	Значение	Описание
vbBinaryCompare	0	Двоичное сравнение по внутренним кодам символов. Буквы разных регистров считаются разными. Установлен по умолчанию
vbTextCompare	1	Текстовое сравнение строк без учета регистра на основе системной национальной настройки

vbDatabaseCompare	2	По установкам базы данных. Используется только в Microsoft Access

Значение функции зависит от соотношения сравниваемых аргументов String1 и String2 (таблица 3.11).

Таблица 3.11 – **Возвращаемые значения функции StrComp**

Соотношение сравниваемых строк	Значение функции
string1 меньше чем string2	−1
string1 равняется string2	0
string1 больше, чем string2	1
string1 или string2 имеет значение Null	Null

String(Number,Character)

Функция String используется для создания строки из одинаковых символов. Возвращает значение типа Variant (String), содержащее повторяющуюся строку указанной длины. Функция содержит следующие обязательные аргументы:

Number, представляющий собой число типа Long, определяющее длину возвращаемой строки. Если аргумент имеет значение Null, то возвращаемое значение тоже Null;

Character в виде параметра со значением типа Variant.

StrReverse(Expression)

Функция используется для построения заданной строки в обратном порядке и возвращает строку Expression в обратном порядке.

Tab([n])

Функция Tab используется вместе с инструкцией Print # или методом Print для указания позиции вывода. При использовании метода Print и функции Tab поле печати разбивается на позиции фиксированной ширины, которая равняется средней ширине всех символов текущего размера в используемом шрифте. Аргумент *n* задает номер столбца, к которому следует перейти перед выводом на экран или печать следующего выражения из списка. Если аргумент отсутствует, Tab устанавливает курсор в начало следующей зоны печати. Это позволяет использовать функцию Tab вместо запятой в качестве разделителя списка, если в текущей национальной настройке запятая используется в качестве десятичного разделителя. Если позиция печати на текущей строке больше *n*, функция Tab вызывает переход к *n*-му столбцу на следующей строке. Если *n* меньше 1, Tab переходит к столбцу 1. Например, если ширина поля печати равна 80,

то выражение Tab(90) установит следующую позицию печати равной 10 (остаток от деления 90 на 80). При печати в файл с помощью инструкции Print # крайней правой позицией печати является текущая ширина результирующего файла, которая устанавливается с помощью инструкции Width #.

Пример:

```
' В данном примере функция Tab
' используется для позиционирования вывода
' в файле и в окне Отладка
' Функцию Tab можно использовать
' в инструкции Print #
Open "TESTFILE" For Output As #1  ' Открываем файл для записи
Print #1, "N"; Tab(20); "Форма"       ' Второе слово – в столбце 20
Print #1, "Расчетная"; Tab; "Схема"
Close #1                          ' Закрываем файл
' Следующая инструкция выводит текст
' в окно Отладка с помощью метода Print,
' печатая текст, начиная со столбца 10
Debug.Print Tab(10); "Проектная линия"
```

Tan(Number)

Функция Tan вычисляет значение тангенса угла, выраженного в радианах. Возвращает значение типа Double. Обязательный аргумент Number имеет тип Double или любое допустимое числовое значение, задающее угол в радианах. Если аргумент не может быть оценен как число, генерируется ошибка.

Пример:

```
Dim A
Dim MyAngle
Dim retval
Dim GradToRad
Const pi = 3.14159265358979
MyAngle = 45                      ' угол в градусах
GradToRad = MyAngle*pi/180        ' перевод угла  в радианы
retval = Tan(GradToRad)           ' вычисление тангенса угла
Debug.Print retval
```

Time()

Функция Time используется для получения текущего времени по системным часам компьютера. Для формированияя системной даты и времени в одной переменной следует использовать функцию Now. Возвращает значение типа Variant (Date), содержащее текущее время по системным часам компьютера. Данная функция не содержит аргументов.

Пример:
MyTime = Time ' Возвращает текущее системное время

Trim(String)

Функция Trim объединяет действия двух функций LTrim и RTrim, удаляя пробелы справа и слева. Возвращает исходную строку типа Variant (String) без начальных и завершающих пробелов. Если аргумент имеет значение Null, то возвращается Null. Обязательный аргумент String может представлять любое строковое выражение.

Пример:
Dim retval
retval=Trim(" <-ВЫЗОВ AUTOCAD-> ")
Print retval

UCase(String)

Функция UCase (Upper Case) преобразует символы строки в верхний регистр. Вместо UCase можно использовать универсальную функцию преобразования строк StrConv. Возвращает строку, которая отличается от входной тем, что все буквы преобразованы в прописные. Обязательный аргумент String может представлять любое строковое выражение.

Пример:
Form1.Caption = UCase("autocad") ' Преобразует строку в верхний регистр

Val(String)

Функция Val (Value) служит для преобразования аргумента в числовой тип данных. Данная функция прекращает чтение строки на первом символе, который она не может распознать в качестве части числа. Символы, которые рассматриваются в качестве частей числовых значений, типа знака доллара и запятых, не распознаются. Пробелы, символы табуляции и символы перевода строк удаляются из значения параметра. Функция Val распознает в качестве разделителя целой и дробной частей только точку. Если используются другие разделители целой и дробной частей, следует применять для преобразования строки в число функцию CDbl. Возвращает числовое представление аргумента с подходящим типом данных. Обязательный аргумент String является любым допустимым строковым выражением.

Пример:
Print Val(" 32-й ") ' возвращается числовое значение 32

***Weekday*Name**(Weekday[, Abbreviate [, FirstDayOfWeek]])

Функция получает полное или сокращенное название дня недели по его номеру. Возвращает строку, содержащую полное или сокращенное название дня недели. Имеет следующие параметры:

Weekday – обязательный аргумент типа Long, определяющего день недели от 1 до 7. Допускается использование констант, указанных в таблице 3.12.

Т а б л и ц а 3.12 – Значения аргумента Weekday

Параметр	Значение параметра	День недели
vbSunday	1	Воскресенье
vbMonday	2	Понедельник
vbTuesday	3	Вторник
vbWednesday	4	Среда
vbThursday	5	Четверг
vbFriday	6	Пятница
vbSaturday	7	Суббота

Abbreviate – необязательный аргумент типа Boolean, указывающий на вид возвращаемого дня недели. По умолчанию установлен в False, что означает вывод полного дня недели;

FirstDayOfWeek – необязательный аргумент, определяющий числовую константу и указывающий, какой день недели считать первым. Значения данного аргумента указаны в таблице 3.13.

Т а б л и ц а 3.13 – Значения констант аргумента FirstDayOfWeek

Константа	Значение константы	Условия применения
vbUseSystem	0	Используется соответствующая системная информация
vbFirstJan1	1	Неделя, обязательно содержащая 1 января
vbFirstFourDays	2	Первая неделя, содержащая не менее 4 дней нового года
vbFirstFullWeek	3	Первая полная неделя текущего или указанного года

Пример:

```
    ' Вывод информации в текстовое окно
Form1.Caption = WeekdayName(7, True)      ' Сокращенное название июля
```

Year(Date)

Функция используется для получения года из заданной даты. Возвращает значение типа Variant (Integer), содержащее целое число, представляющее год. Обязательный параметр Date является числовым

выражением, строковым выражением или любой комбинацией, с помощью которой может быть отражена дата. Допустимый диапазон дат от 01.01.0000 до 31.12.9999.

Пример:

```
Dim MyDate
Dim MyYear
MyDate = #7/27/1966#                ' Указание даты
MyYear = Year(MyDate)
Print MyYear
```

Задачи

Задача 3.1 Чем можно заменить фреймы «Кривая» и «Смещение пути» на рисунке 3.2?

Задача 3.2 Дополнить блок «Результаты расчетов» соответствующими элементами $R_ф$, $T_ф$ и $K_ф$ и изменить исходный код программы.

Задача 3.3 Составить таблицу связи всех параметров программы с именами переменных по образцу, приведенному в таблице 3.14.

Т а б л и ц а 3.14 – Соответствие параметров и переменных программы

Параметр	Область размещения соответствующего элемента	Переменная
Радиус кривой	Исходные данные	Txt_Radius_Tek
b_1	Абсолютные размеры смещения	Txt_b1
...	…	…

Задача 3.4 Как изменить код процедуры Form_Resize () (см. рисунок 3.4), чтобы исполняемая форма программы находилась у левого края экрана дисплея?

Задача 3.5 Расширить состав параметров и переменных задачи рисунка 3.3, дополнив ее определением нормалей на пикетах. В качестве исходных данных принять значения примера 1 [2, с. 33–36].

Задача 3.6 Дополнить программу введением новой связанной формы с информацией по теории расчета элементов плана пути при нажатии на пункт меню «Сведения из теории» (см. рисунок 3.8).

Задача 3.7 Разработать программный код по интерфейсу, представленному на рисунке 3.2.

Задача 3.8 Дополнить внешний вид программы кнопкой «Справка», расположенной в нижнем правом углу рядом с кнопкой «Выход». Связать кнопку «Справка» с пунктом меню «Справка» таким образом, чтобы нажатие на них указателя мыши вызывало одну новую форму с одним содержанием.

Задача 3.9 Аналогично предыдущей задаче связать кнопку и пункт меню «Выход».

Задача 3.10 Провести компиляцию полученного кода программы. Для этого в меню File пакета Visual Basic активизировать пункт Make File_Name.txt...

Тесты

Тест 3.1 Разработка программного кода производится:
1) посредством вызова соответствующего файла и его редактирования;
2) в редакторе кода;
3) с помощью дополнительных окон элемента управления TextBox;
4) переносом на форму специального элемента управления.

Тест 3.2 Двойное нажатие на элемент управления:
1) порождает событие;
2) приводит к появлению на форме второго элемента;
3) увеличивает размеры элемента на форме в 2 раза;
4) удаляет элемент управления с формы.

Тест 3.3 Выход из загруженной программы без записи программного кода осуществляется нажатием:
1) на кнопку  формы;
2) на кнопку  формы;
3) на кнопку  формы;
4) на клавишу <ENTER>.

Тест 3.4 Изменение шрифта метки на форме производится с помощью вызова свойства:
1) Font;
2) Caption;
3) Name;
4) Alignment.

Тест 3.5 Информация в окнах элементов управления TextBox и Label записывается как:
1) числа;
2) логические переменные;
3) данные неопределенного типа;
4) строки.

Тест 3.6 Указать неправильную запись:
1) Dim X As Byte
   X = 5

Label32.Caption = CStr(X);

2) Dim Q_A As Integer
Q_A = 72
TextBox11.Text = CStr(Q_A);

3) Dim B As String
B = “1”
TextBox5.Text = B;

4) Dim B As String
B = “1”
TextBox7.Text = Val(B).

Тест 3.7 Строка

Text11.Text = CStr(Round(Val (Text6.Text)*Val(Text9.Text))),

записанная на кнопку CommonButton1, позволяет получить результат умножения двух значений из кнопок Text6 и Text9 в виде:

1) целого числа на кнопке CommonButton1;
2) целого числа в окне элемента управления Text11;
3) действительного числа в окне элемента Text6;
4) действительного числа в окне элемента Text9.

Тест 3.8 Размещение формы в центре экрана дисплея после загрузки программы достигается с помощью кода:

1) Unload Me
Set FrmHelp = Nothing;

2) Public Function CenterForm As Boolean
CenterForm = True
End Function;

3) Left = (Screen.Width - Width) / 2
Top = (Screen.Height - Height) / 2;

4) Private Sub CmdOk_Click()
Dim Center1 As Byte
Center1 = 1
Left = 200
Top = 300
End Sub.

Тест 3.9 Указать правильную запись

1)TextBox5.Locked = 1;
2) CmdExit.SetFocus;
3) Label5.Caption = Val(TextBox1.Text);
4) N_PK = Command3.Caption.

Тест 3.10 Результатом использования команды FormExit.Show является:

1) немодальная форма;
2) модальная форма;
3) вызов графического файла;
4) аварийное завершение программы.

Контрольные вопросы

1 Для чего служит окно редактора кода?

2 Можно ли исключить на форме кнопку «Выход», обеспечивающую завершение работы программы?

3 Как производится запись рассчитанного значения $K_{пр}$ в соответствующее поле программной формы? Указать соответствующие строки кода в процедуре Cmd_Rascet_Click().

4 Как можно охарактеризовать имя элемента Cmd_Exit?

5 Как будет работать программа, если расчетное значение $R_ф$ окажется равным целому числу?

6 Как можно объединить в одну строку фрагмент программы

```
b2 = z * Sin(Alfa_Rad) - b1
Txt_b2.Text = Str(b2)
```
?

7 Какое различие между модальными и немодальными формами?

8 Почему целесообразно выносить все функции за пределы кода основной программы?

9 В чем недостаток такой функции:

```
Public R_Fict(R_s  As Double, y As Double, z As Double, Alfa_Grad As Integer,
Alfa_Min As Integer) As Double
    R_Fict = R_s – y  – z * cos((Alfa_Grad + Alfa_Min / 60) * 3.141593 / 180)
End Function
```
?

10 Какой результат будет иметь попытка замены стандартной функции Visual Basic пользовательской функцией с таким же именем?

4 ОБЩАЯ СХЕМА КОМПЬЮТЕРНОГО РЕШЕНИЯ ЗАДАЧИ КОМПЛЕКСНОГО РАСЧЕТА И ПРОЕКТИРОВАНИЯ СМЕЩЕНИЯ ПУТИ НА СТАНЦИИ И ПЕРЕГОНЕ

4.1 Особенности решения проектных задач с использованием компьютерных технологий

Отличительной особенностью задачи проектного характера является обязательное наличие графической компоненты. В проектах сооружения новых и реконструкции существующих путей железных дорог, как правило, это фрагмент масштабного плана путевого развития станции или перегона, в котором рассчитываются все элементы прямых и криволинейных участков пути. Графический материал следует различать двух основных видов: расчетные схемы и масштабные планы.

Расчетные схемы представляют собой рисунки с обозначением параметров и их значений на выносных линиях. При этом допускаются другие поясняющие надписи и обозначения, которые помогают однозначно зафиксировать задачу на уровне зрительного образа. К схеме не предъявляются никакие дополнительные требования, в том числе и соблюдения масштаба. Важно только, чтобы все элементы схемы были графически соразмерны по значениям параметров. Например, если междупутье $M_1 > M_2$ по величине, то и на схеме визуально должно соблюдаться данное соотношение. Если стрелочные переводы *STR*(1) и *STR*(2) укладываются с маркой крестовины 1/9, то и на схеме углы поворота на боковой путь по стрелочным переводам *STR*(1) и *STR*(2) должны быть примерно одинаковы на вид. Длины кривых и их радиусы также должны быть сопоставимы по значениям и соответствующим визуальным образам.

Масштабный план в этом отношении должен строго соответствовать указанному масштабу без всяких исключений. При этом вертикальный и горизонтальный масштабы должны быть одинаковы. Проектировщики работают с масштабными планами 1:500, 1:1000 и 1:2000. Точная фиксация отдельных элементов путевого развития возможна при координатной привязке в условной прямоугольной (или принятой) системе координат (одной и той же в пределах данного чертежа). Такой подход позволяет перевести величины расчетных параметров в их адекватные графические аналоги.

Расчет координат контрольных точек элементов путевого развития (начала и конца участков прямых и кривых, начала и конца кривых, угла поворота кривых) позволяет вычертить фрагмент плана в некотором графическом пакете САПР (например, AutoCAD).

Пусть некоторая исходная точка фрагмента путевого развития станции A имеет координаты $A(0,\ 0)$. Тогда при известных значениях длины всех связных элементов легко рассчитать остальные координаты ключевых точек. В качестве примера рассматривается координатная привязка параллельного смещения пути, расчетная схема которого приведена на рисунке 4.1.

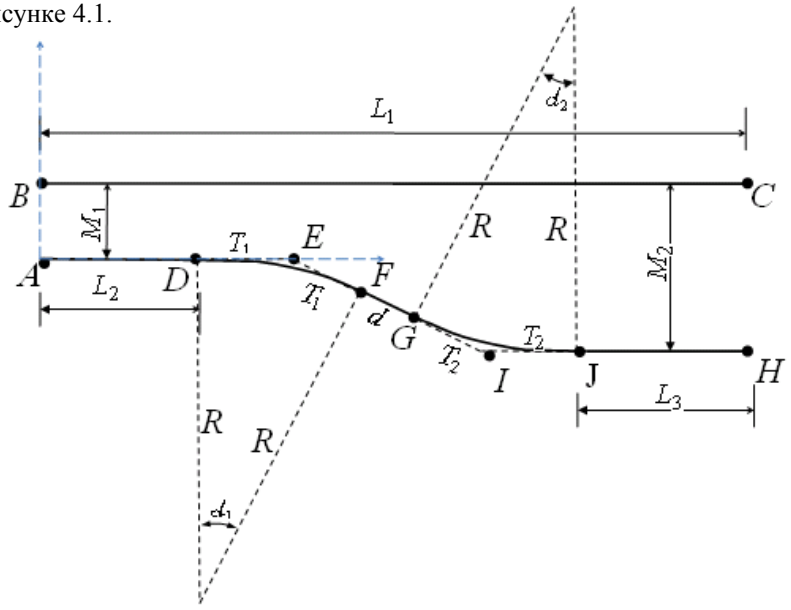


Рисунок 4.1 – Расчетная схема параллельного смещения пути

Следует отметить, что данная схема иллюстрирует вариант смещения пути на железнодорожной станции, где отсутствуют переходные кривые, прямая вставка фиксируется как некоторая рубка, R = 200...300 м, $T_1 = T_2 = T$ и $\alpha_1 = \alpha_2 = \alpha$.

Согласно рисунку 4.1 при $X_A = 0$ и $Y_A = 0$:

$X_B = X_A = 0$,
$Y_B = Y_A + M_1$,
$X_C = X_B + L_1$,
$Y_C = Y_B$,
$X_D = X_A + L_2$,
$Y_D = Y_{A1}$,
$X_E = X_D + T_1$,
$Y_E = Y_D$,

$X_F = X_E + T_1\cos\alpha$,
$Y_F = Y_E - T_1\cos\alpha$,
$X_G = X_F + (T_1 + d)\cos\alpha$,
$Y_G = Y_F + (T_1 + d)\sin\alpha$,
$X_I = X_G + (2T_1 + d)\cos\alpha$,
$Y_I = Y_G + (2T_1 + d)\cos\alpha$,
$X_J = X_I + L_3$,
$Y_J = Y_I$,

$X_H = X_J + L_3$,
$Y_H = Y_J$.

Неизвестными параметрами могут быть некоторые два из перечисленных и используемых при расчетах. Сумма проекций отдельных элементов на оси ОХ и ОУ позволяет определить систему двух уравнений:

$$\begin{cases} L_2 + T + (2T + d)\cos\alpha + T + L_3 = L_1, \\ M_1 + (2T + d)\sin\alpha = M_2. \end{cases} \tag{4.1}$$

Если искомыми неизвестными являются R и d, то решение системы уравнений следующее:

$$T = R\operatorname{tg}\frac{\alpha}{2}, \qquad d = \frac{M_2 - M_1}{\sin\alpha} - 2R\operatorname{tg}\frac{\alpha}{2}, \tag{4.2}$$

$$L_2 + 2T + (2T + d)\cos\alpha + L_3 = L_1,$$
$$L_2 + 2T + 2T\cos\alpha + d\cos\alpha + L_3 = L_1,$$
$$2T + 2T\cos\alpha = L_1 - L_2 - L_3 - d\cos\alpha,$$
$$T(2 + \cos\alpha) = L_1 - L_2 - L_3 - d\cos\alpha,$$
$$T = \frac{L_1 - L_2 - L_3 - d\cos\alpha}{2 + \cos\alpha}, \quad R\operatorname{tg}\frac{\alpha}{2} = \frac{L_1 - L_2 - L_3 - d\cos\alpha}{2 + \cos\alpha},$$
$$R = \frac{L_1 - L_2 - L_3 - d\cos\alpha}{(2 + \cos\alpha)\operatorname{tg}\frac{\alpha}{2}}. \tag{4.3}$$

Подстановка (4.3) в (4.2) приводит к следующим результатам:

$$d = \frac{M_2 - M_1}{\sin\alpha} - 2\operatorname{tg}\frac{\alpha}{2} \cdot \frac{L_1 - L_2 - L_3 - d\cos\alpha}{(2 + \cos\alpha)\operatorname{tg}\frac{\alpha}{2}},$$
$$d = \frac{M_2 - M_1}{\sin\alpha} - 2 \cdot \frac{L_1 - L_2 - L_3 - d\cos\alpha}{2 + \cos\alpha},$$
$$d = \frac{M_2 - M_1}{\sin\alpha} - 2 \cdot \frac{L_1 - L_2 - L_3}{2 + \cos\alpha} + \frac{d\cos\alpha}{2 + \cos\alpha},$$
$$d - \frac{d\cos\alpha}{2 + \cos\alpha} = \frac{M_2 - M_1}{\sin\alpha} - 2 \cdot \frac{L_1 - L_2 - L_3}{2 + \cos\alpha},$$

$$\frac{(2+\cos\alpha)d - d\cos\alpha}{2+\cos\alpha} = \frac{M_2 - M_1}{\sin\alpha} - 2\cdot\frac{L_1 - L_2 - L_3}{2+\cos\alpha},$$

$$\frac{2d + d\cos\alpha - d\cos\alpha}{2+\cos\alpha} = \frac{M_2 - M_1}{\sin\alpha} - 2\cdot\frac{L_1 - L_2 - L_3}{2+\cos\alpha},$$

$$\frac{2}{2+\cos\alpha} + 2\cdot\frac{L_1 - L_2 - L_3}{2+\cos\alpha} = \frac{M_2 - M_1}{\sin\alpha},$$

$$\frac{2 + 2(L_1 - L_2 - L_3)}{2+\cos\alpha} = \frac{M_2 - M_1}{\sin\alpha},$$

Если $L_1 - L_2 - L_3 = \Delta L$ и $M_2 - M_1 = \Delta M$, то

$$\frac{2 + 2\Delta L}{2+\cos\alpha} = \frac{\Delta M}{2\sin\alpha}.$$

Если требуется определить угол α и величину прямой вставки d, то порядок вычислений следующий:

из первого уравнения системы (4.1)

$$L_2 + 2T + 2T\cos\alpha + d\cos\alpha + L_3 = L_1,$$

$$d\cos\alpha = L_1 - L_2 - L_3 - 2T - 2T\cos\alpha,$$

(4.4)

$$d = \frac{L_1 - L_2 - L_3 - 2T - 2T\cos\alpha}{\cos\alpha}.$$

Из второго уравнения системы (4.1)

$$M_1 + 2T\sin\alpha + d\sin\alpha = M_2,\ d\sin\alpha = M_2 - M_1 - 2T\sin\alpha,$$

$$d = \frac{M_2 - M_1 2T\sin\alpha}{\sin\alpha}.$$

(4.5)

Из выражения (4.4) находим

$$d\cos\alpha + 2T\cos\alpha = L_1 - L_2 - L_3 - 2T,$$

$$\cos\alpha(d + 2T) = L_1 - L_2 - L_3 - 2T,$$

$$\cos\alpha = \frac{L_1 - L_2 - L_3 - 2T}{d + 2T},$$

$$\cos^2\alpha = \left(\frac{L_1 - L_2 - L_3 - 2T}{d + 2T}\right)^2.$$

(4.6)

Из уравнения (4.5) определяем:

$$2T\sin\alpha + d\sin\alpha = M_2 - M_1,\ \sin\alpha(2T + d) = M_2 - M_1,$$

$$\sin\alpha = \frac{M_2 - M_1}{2T + d},\ \sin^2\alpha = \left(\frac{M_2 - M_1}{2T + d}\right)^2,$$

$$1 - \cos^2\alpha = \left(\frac{M_2 - M_1}{2T + d}\right)^2,$$

$$\cos^2\alpha = 1 - \left(\frac{M_2 - M_1}{2T + d}\right)^2.$$

Из уравнения (4.6)

$$\left(\frac{L_1 - L_2 - L_3 - 2T}{d + 2T}\right)^2 = 1 - \left(\frac{(M_2 - M_1)}{d + 2T}\right)^2,$$

$$\left(\frac{L_1 - L_2 - L_3 - 2T}{d + 2T}\right)^2 + \left(\frac{M_2 - M_1}{d + 2T}\right)^2 = 1,$$

$$(L_1 - L_2 - L_3 - 2T)^2 + (M_2 - M_1)^2 = (d + 2T)^2,$$

$$(L_1 - L_2 - L_3 - 2T)^2 + (M_2 - M_1)^2 = 4T^2 + 4Td + d^2,$$

$$-d^2 - 4Td - 4T^2 + (L_1 - L_2 - L_3 - 2T)^2 + (M_2 - M_1)^2 = 0,$$

$$d^2 + 4Td + 4T^2 - (L_1 - L_2 - L_3 - 2T)^2 - (M_2 - M_1)^2 = 0.$$

Если $(L_1 - L_2 - L_3 - 2T)^2 - (M_2 - M_1)^2 = W$, то

$$d^2 + 4Td + W = 0,\ d_{1,2} = \frac{-4T \pm \sqrt{16T^2 - 4W}}{2}.$$

Так как $d \geq 0$, то

$$d = \frac{-4T + \sqrt{16T^2 - 4W}}{2},$$

$$d = \frac{-4T + \sqrt{16T^2 - 4[4T^2 - (L_1 - L_2 - L_3 - 2T)^2 - (M_2 - M_1)^2]}}{2},$$

$$d = -2T + \sqrt{4T^2 - [4T^2 - (\Delta L - 2T)^2 - \Delta M^2]},$$

$$d = \sqrt{4T^2 - 4T^2 + (\Delta L - 2T)^2 - \Delta M^2} - 2T = \sqrt{(\Delta L - 2T)^2 - \Delta M^2} - 2T,$$

$$d = \sqrt{(\Delta L - 2T)^2 - \Delta M^2} - 2T,$$

$$d = \sqrt{\left(\Delta L - 2R\operatorname{tg}\frac{\alpha}{2}\right)^2 - \Delta M^2} - 2R\operatorname{tg}\frac{\alpha}{2}.$$

(4.7)

Если требуется определить α и *R*, то последовательность вычислительных операций следующая:

из второго уравнения системы (4.1)

$$(2R\operatorname{tg}\frac{\alpha}{2} + d)\sin\alpha = M_2 - M_1,\ 2R\operatorname{tg}\frac{\alpha}{2} + d = \frac{M_2 - M_1}{\sin\alpha},$$

$$2R\operatorname{tg}\frac{\alpha}{2} = \frac{M_2 - M_1}{\sin\alpha} - d,\ 2R\operatorname{tg}\frac{\alpha}{2} = \frac{M_2 - M_1 - d\sin\alpha}{\sin\alpha},$$

$$R = \frac{M_2 - M_1 - d\sin\alpha}{2\sin\alpha \cdot \operatorname{tg}\frac{\alpha}{2}}. \qquad (4.8)$$

Из первого уравнения системы (4.1)

$$L_2 + 2R\operatorname{tg}\frac{\alpha}{2} + \left(2R\operatorname{tg}\frac{\alpha}{2} + d\right)\cos\alpha + L_3 = L_1,$$

$$2R\operatorname{tg}\frac{\alpha}{2} + 2R\operatorname{tg}\frac{\alpha}{2}\cos\alpha = L_1 - L_2 - L_3 - d\cos\alpha,$$

$$R\left(2\operatorname{tg}\frac{\alpha}{2} + 2\operatorname{tg}\frac{\alpha}{2}\cos\alpha\right) = L_1 - L_2 - L_3 - d\cos\alpha,$$

$$R = \frac{L_1 - L_2 - L_3 - d\cos\alpha}{2\operatorname{tg}\frac{\alpha}{2}(1 + \cos\alpha)}. \qquad (4.9)$$

Приравниваем правые части уравнений (4.8) и (4.9):

$$\frac{M_2 - M_1 - d\sin\alpha}{2\operatorname{tg}\frac{\alpha}{2}\sin\alpha} = \frac{L_1 - L_2 - L_3 - d\cos\alpha}{2\operatorname{tg}\frac{\alpha}{2}(1 + \cos\alpha)},$$

$$\frac{M_2 - M_1 - d\sin\alpha}{\sin\alpha} = \frac{L_1 - L_2 - L_3 - d\cos\alpha}{1 + \cos\alpha}.$$

Полученная окончательная формула не является функциональным выражением $\alpha = f(Par_i)$. Для нахождения значения α необходимо итерационным способом достигать равенства левой и правой частей уравнения при некотором значении α.

Возможно решение задачи поиска некоторого функционального выражения зависимости трех неизвестных параметров. Так, при необходимости нахождения соотношения α, R и d этапная последовательность вывода необходимых зависимостей может быть следующей:

из первого уравнения системы (4.1)

$$2R\operatorname{tg}\frac{\alpha}{2} + \left(2R\operatorname{tg}\frac{\alpha}{2} + d\right)\cos\alpha = \Delta L,$$

$$2R\operatorname{tg}\frac{\alpha}{2} + 2R\operatorname{tg}\frac{\alpha}{2}\cos\alpha + d\cos\alpha = \Delta L,$$

$$2R\operatorname{tg}\frac{\alpha}{2}(1+\cos\alpha) + d\cos\alpha = \Delta L,$$

$$2R\operatorname{tg}\frac{\alpha}{2} = \frac{\Delta L - d\cos\alpha}{1+\cos\alpha}. \quad (4.10)$$

Из второго уравнения системы (4.1)

$$\left(2R\operatorname{tg}\frac{\alpha}{2} + d\right)\sin\alpha = \Delta M,\ 2R\operatorname{tg}\frac{\alpha}{2}\sin\alpha + d\sin\alpha = \Delta M,$$

$$2R\operatorname{tg}\frac{\alpha}{2}\sin\alpha = \Delta M - d\sin\alpha,$$

$$2R\operatorname{tg}\frac{\alpha}{2} = \frac{\Delta M - d\sin\alpha}{\sin\alpha}. \quad (4.11)$$

Приравнивая правые части выражений (4.10) и (4.11), получаем:

$$\frac{\Delta L - d\cos\alpha}{1+\cos\alpha} = \frac{\Delta M - d\sin\alpha}{\sin\alpha},$$

$$(\Delta L - d\cos\alpha)\sin\alpha = (1+\cos\alpha)(\Delta M - d\sin\alpha),$$

$$\Delta L\sin\alpha - d\sin\alpha\cos\alpha = \Delta M - d\sin\alpha + \Delta M\cos\alpha - d\sin\alpha\cos\alpha,$$

$$\Delta L\sin\alpha - \Delta M + d\sin\alpha - \Delta M\cos\alpha = 0,$$

$$\Delta L\sin\alpha + d\sin\alpha - \Delta M - \Delta M\cos\alpha = 0,$$

$$\sin\alpha(\Delta L + d) - \Delta M(1+\cos\alpha) = 0,$$

$$\frac{\Delta L + d}{1+\cos\alpha} = \frac{\Delta M}{\sin\alpha}, \quad \frac{\Delta L}{1+\cos\alpha} + \frac{d}{1+\cos\alpha} = \frac{\Delta M}{\sin\alpha},$$

$$\frac{d}{1+\cos\alpha} = \frac{\Delta M}{\sin\alpha} - \frac{\Delta L}{1+\cos\alpha},$$

$$d = \left(\frac{\Delta M}{\sin\alpha} - \frac{\Delta L}{1+\cos\alpha}\right)(1+\cos\alpha).$$

(4.12)

Полученное уравнение может использоваться для исследований зависимости параметра d от α (сравнить с выражением (4.7)).

4.2 Программный поиск параметров смещения пути

Наиболее простая задача заключается в выборе параметров путей L_1 и M_2, что достигается решением системы уравнений (4.1) относительно этих переменных. Для представления способа автоматизации расчетов разработаем следующую программную форму с использованием среды Visual Basic (рисунок 4.2).

На данную форму помещаем расчетную схему параллельного смещения пути. Этот шаг реализуется с помощью инструмента Image с последующей активизацией свойства Picture и вводом имени соответствующего графического файла. Такой подход следует признать достаточно удобным, позволяющим сопоставлять исходные и расчетные параметры графическому аналогу, указывающему на место расположения контрольных точек и протяженных объектов (M_1, M_2, L_1, L_2, L_3, d) на схеме. Пря… M_2, опр…у, скла…ие меж… с нор…ти сме…

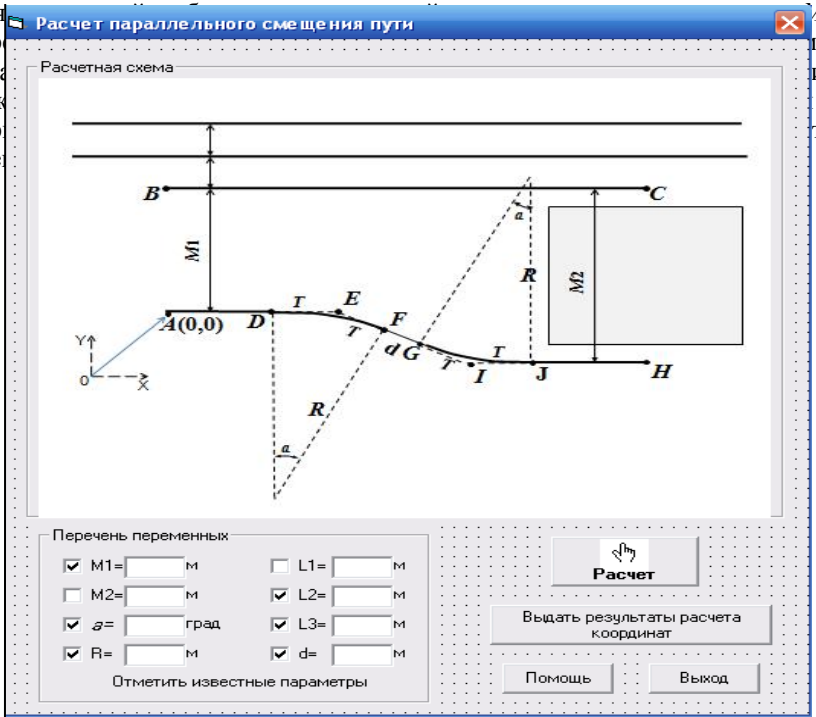


Рисунок 4.2 – Интерфейс задачи расчета параллельного смещения пути

Фрейм «Перечень переменных» включает 8 параметров, каждый из которых охватывает два основных программных компонента: Check и TextBox. По условию задачи из указанных восьми переменных требуется определить две – L_1 и M_2. Эти переменные не отмечены активными флажками. Пусть требуется программным способом скрыть поля искомых переменных Txt_L1 и Txt_M2 на этапе ввода данных.

Правило удаления этих полей с программной формы состоит в следующем: если соответствующий флажок не отмечен, то рядом стоящее текстовое поле должно быть невидимым. Это правило применяем к каждой переменной:

```
If Check_Alfa.Value = 0
  Then Txt_Ugol.Visible = False
  Else Txt_Ugol.Visible = True

If Check_M1.Value = 0
  Then Txt_M1.Visible = False
  Else Txt_M1.Visible = True

If Check_M2.Value = 0
  Then Txt_M2.Visible = False
  Else Txt_M2.Visible = True

If Check_Radius.Value = 0
  Then Txt_Radius.Visible = False
  Else Txt_Radius.Visible = True

If Check_L1.Value = 0
  Then Txt_L1.Visible = False
  Else Txt_L1.Visible = True

If Check_L2.Value = 0
```

```
    Then Txt_L2.Visible = False
    Else Txt_L2.Visible = True                                    (4.13)

  If Check_L3.Value = 0
    Then Txt_L3.Visible = False
    Else Txt_L3.Visible = True

  If Check_d.Value = 0
    Then Txt_d.Visible = False
    Else Txt_d.Visible = True.
```

Удаление текстовых полей должно производиться на этапе исполнения программы. Для этого указанный код следует поместить в процедуру загрузки формы (дважды щелкнуть мышью на форме «Расчет параллельного смещения пути» и в открывшейся процедуре Form_Load () ввести приведенные выше строки).

После загрузки программы можно выбирать исходные и расчетные данные из фрейма «Перечень переменных». Чтобы сразу изменялась видимость соответствующих текстовых полей, необходимо выполнять аналогичную проверку с компонентами флажков. Дважды нажимая мышью на эти компоненты, вводим следующие блоки программного кода:

```
Private Sub Check_Alfa_Click()
  If Check_Alfa.Value = 0
    Then Txt_Ugol.Visible = False Else Txt_Ugol.Visible = True
End Sub
Private Sub Check_d_Click()
  If Check_d.Value = 0
    Then Txt_d.Visible = False Else Txt_d.Visible = True
End Sub
Private Sub Check_L1_Click()
  If Check_L1.Value = 0
    Then Txt_L1.Visible = False Else Txt_L1.Visible = True
End Sub
Private Sub Check_L2_Click()  If Check_L2.Value = 0
    Then Txt_L2.Visible = False Else Txt_L2.Visible = True
End Sub
Private Sub Check_L3_Click()
  If Check_L3.Value = 0
    Then Txt_L3.Visible = False Else Txt_L3.Visible = True
End Sub
Private Sub Check_M1_Click()
  If Check_M1.Value = 0
```

```
    Then Txt_M1.Visible = False Else Txt_M1.Visible = True
End Sub
Private Sub Check_M2_Click()
  If Check_M2.Value = 0
    Then Txt_M2.Visible = False Else Txt_M2.Visible = True
End Sub
Private Sub Check_Radius_Click()  If Check_Radius.Value = 0
    Then Txt_Radius.Visible = False Else Txt_Radius.Visible = True
End Sub
```

В предлагаемом варианте программы исходные данные записываются как значения текстовых полей в процедуру Form_Load (). Пусть они будут следующими:

```
Txt_M1.Text = "7.50"
Txt_Ugol.Text = "6.2"
Txt_Radius.Text = "200"
Txt_L2.Text = "25"
Txt_L3.Text = "25"
Txt_d.Text = "6.25".
```

После загрузки программы эти данные переходят на соответствующие текстовые поля. Данные поля остаются доступными для изменения, поэтому на любом этапе работы программы можно корректировать все исходные данные.

Кнопка «Расчет» имеет программное имя Cmd_Rascet. Открываем поле ввода программного кода в процедуру Cmd_Rascet ():

```
Private Sub Cmd_Rascet_Click()
  Dim Alfa, Alfa_Rad, Tangens, L1, M2 As Double
  Alfa = Val(Txt_Ugol)    Alfa_Rad = Alfa * 3.141593 / 180
  Tangens = Val(Txt_Radius.Text) * Tan(Alfa_Rad / 2)
    L1 = Val(Txt_L2.Text) + (2 * Tangens + d) * Cos(Alfa_Rad) +
Val(Txt_L3.Text)
  M2 = Val(Txt_M1.Text) + (2 + Tangens + d) * Sin(Alfa_Rad)
   Txt_L1.Visible = True    Txt_M2.Visible = True    Txt_L1.Text =
CStr(L1)
  Txt_M2.Text = CStr(M2)
End Sub
```

Общий вид загрузочного модуля программы приведен на рисунке 4.3.

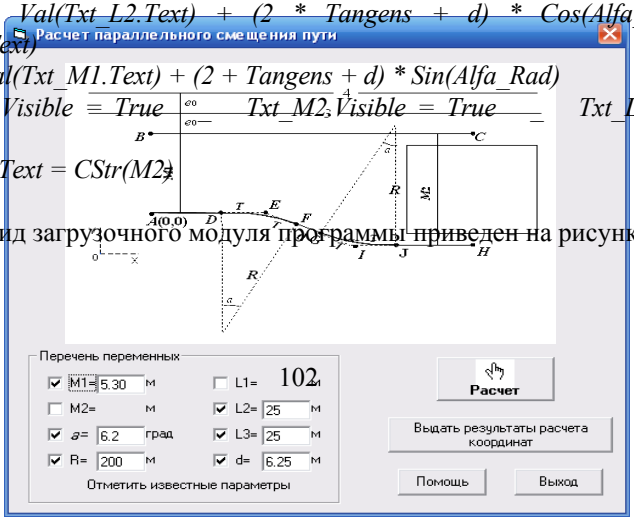


Рисунок 4.3 – Интерфейс программы расчета смещения пути

При нажатии на кнопку «Расчет» происходит заполнение текстовых полей расчетных параметров (рисунок 4.4).

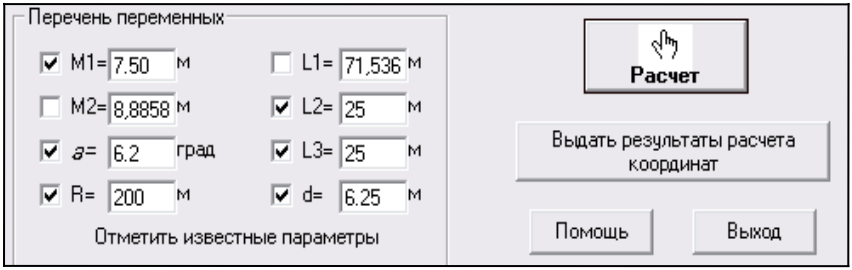


Рисунок 4.4 – Результаты работы программы

По сравнению с предыдущим экраном (см. рисунок 4.3) визуализируются текстовые окна параметров L_1 и M_2, в которые записываются соответствующие рассчитанные значения.

4.3 Программный расчет координат контрольных точек смещения пути

Расчет заключается в определении координат контрольных точек *A*, *B*, *C*, *D*, *E*, *F*, *G*, *I*, *J*, *H*. Выдача результатов будет производиться на основную форму и в дополнительное программное окно второй модальной формы. Представленный ниже программный код позволяет рассчитать координаты контрольных точек:

Const X_A = 0 _ Const Y_A = 0

```
Dim X_B, Y_B, X_C, Y_C, X_D, Y_D, X_E, Y_E As Double
Dim X_F, Y_F, X_G, Y_G, X_I, Y_I, X_J, Y_J, X_H, Y_h As Double
X_B = X_A _  Y_B = Y_A + M1 _   X_C = X_B + L1 _    Y_C = Y_B
X_D = X_A + L2 _   Y_D = Y_A _    X_E = X_D + Tangens _  Y_E = Y_D
X_F = X_E + Tangens * Cos(Alfa_Rad)
Y_F = Y_E - Tangens * Sin(Alfa_Rad)
X_G = X_F + d * Cos(Alfa_Rad)
Y_G = Y_F - d * Sin(Alfa_Rad)
X_I = X_G + Tangens * Cos(Alfa_Rad)
Y_I = Y_G - Tangens * Sin(Alfa_Rad)
X_J = X_I + Tangens
Y_J = Y_I
X_H = X_J + L3
Y_h = Y_J
End Sub.
```

Результаты расчета координат контрольных точек можно вывести на некоторое поле формы (рисунок 4.5).

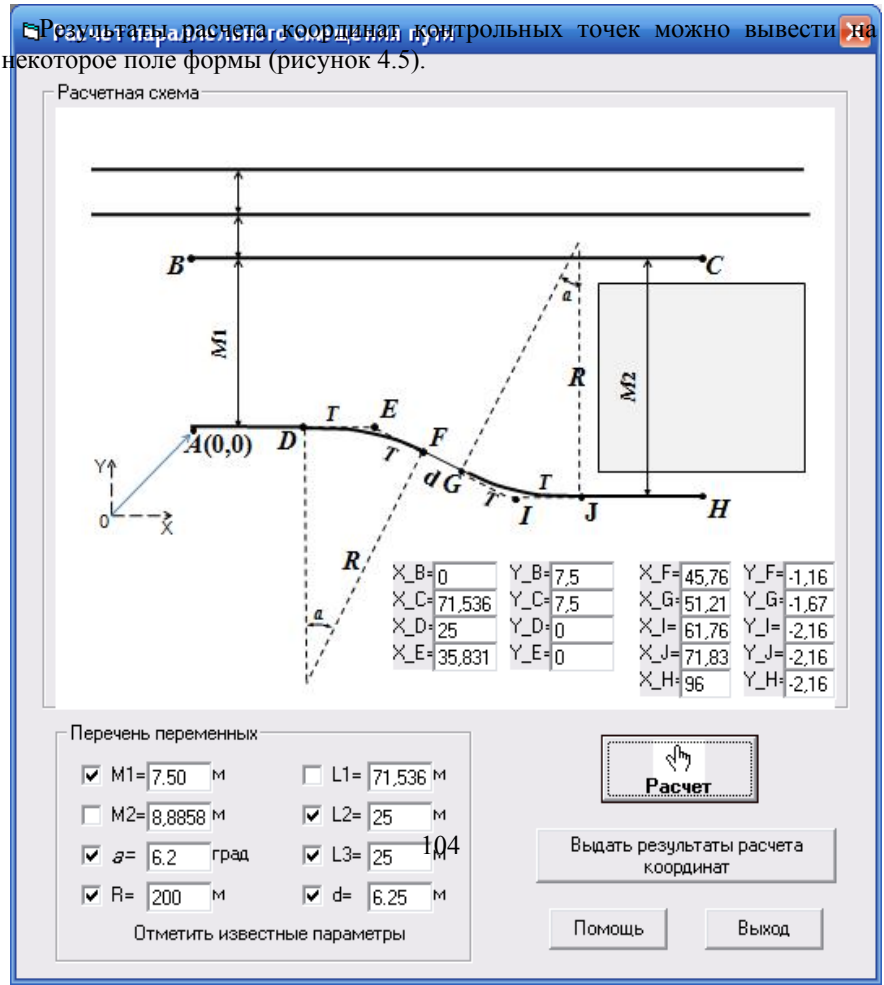


Рисунок 4.5 – Поле координат контрольных точек

Вторая модальная форма формируется с помощью выбора в основном меню среды проектирования Visual Basic ряда опций Project–Add Form–Form. Новую программную форму фиксируем с соответствующим именем Frm_Shema. В редакторе Visual Basic в окне справа отображается дополнительная строка с наименованием полученной программной формы (рисунок 4.6).

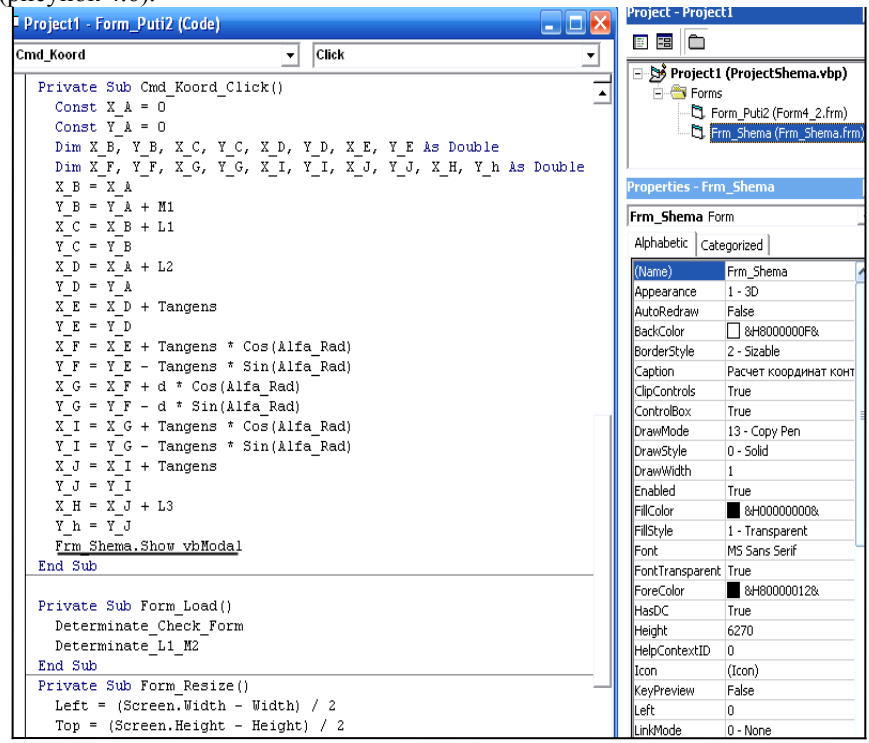


Рисунок 4.6 – Программный код с вызовом новой формы

Переключаться с одной формы на другую в редакторе удобно посредством двойного клика мышью на соответствующую строку правого верхнего окна (см. рисунок 4.6). Следует обратить внимание на определенную сложность использования результатов расчета, выполненных в программной форме Form_Puti2. Все переменные, описанные в некоторой процедуре с помощью оператора Dim, могут быть использованы только в этой процедуре. Существует ряд способов переноса переменных в область глобального доступа. Наиболее простым является считывание значений текстового поля одной формы и запись его в текстовое поле другой формы. Например, при заполненном текстовом поле Txt_M1.Text = “7.50” программной формы Form_Puti2 записать это значение на текстовое поле формы Form_Shema можно следующим образом:

Form_Shema.Txt_M1.Text = Form_Puti2.Txt_M1.Text.

При этом, как можно заметить, имена текстовых полей на разных программных формах одинаковы (в данном случае это помогает ориентироваться в большом количестве однотипных переменных, однако в общем случае лучше давать всем переменным уникальные имена).

Наиболее наглядным может быть выдача координат данных точек на графическое поле. В приведенном примере рассмотрим возможность формирования вложенного окна с координатами точек (рисунок 4.7).

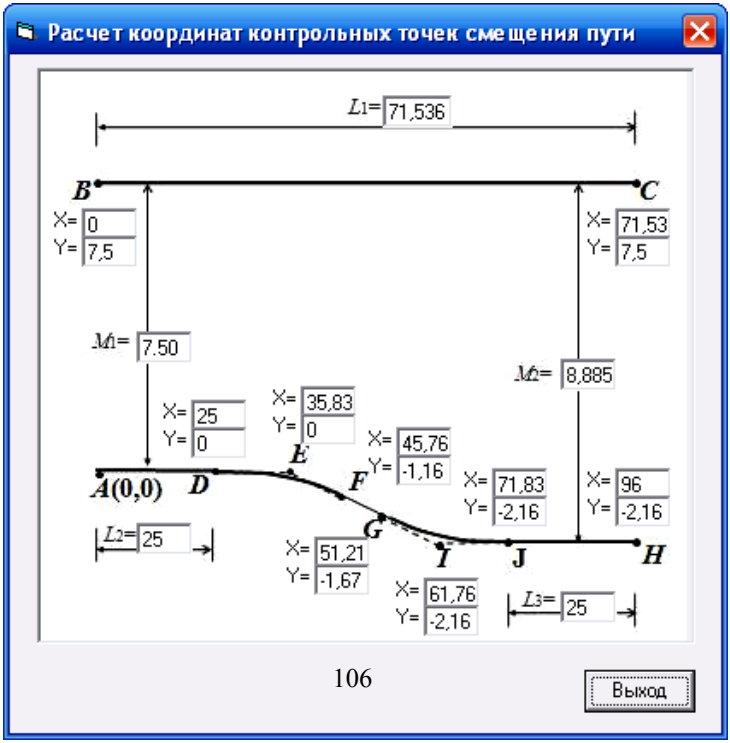


Рисунок 4.7 – Второе окно формы с результатами расчета длин и координат

Это окно можно считать наиболее информативным, так как его содержание конкретно. Оно концентрирует внимание на точках привязки элементов путевого развития. Рядом с каждой контрольной точкой указываются значения пары координат. Если создать координатную сетку с достаточно малым шагом (h = 0,001 м), то можно было бы вычертить в масштабе полученный фрагмент путевого развития. Однако обычными средствами Visual Basic это сделать чрезвычайно трудно.

Для решения проектной части поставленной задачи необходимо использовать среду автоматизированного вычерчивания графических примитивов. Такой средой может быть AutoCAD.

Задачи

Задача 4.1 Как изменится содержание процедуры Cmd_Rascet (), если перенести начало координат в точку B (см. рисунок 4.2)?

Задача 4.2 Разработать алгоритм решения задачи параллельного смещения пути при неизвестных параметрах R и d.

Задача 4.3 Предложить интерфейс программы предыдущей задачи при различных параметрах S-образной кривой параллельного смещения пути (α_1, α_2, T_1, T_2, R_1, R_2).

Задача 4.4 Разработать одну процедуру расчета координат точек D, E, F, G, I, J, H (см. рисунок 4.7) для условий $M_1 > M_2$ и $M_1 < M_2$.

Задача 4.5 Разработать программное окно с отображением расчетной схемы согласно выбранному варианту проведения расчетов по интерфейсу, приведенному на рисунке 4.8.

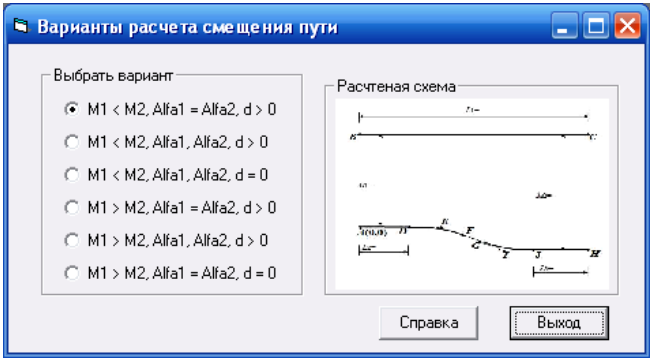


Рисунок 4.8 – Интерфейс программной формы альтернатив смещения пути

Рисунок 4.9 – Графиковая сетка точек координат

Задача 4.6 Разработать график зависимости *X*-координаты точки *H* при различных углах поворота и константных значениях остальных параметров. Вынести полученный график на вторую программную форму средствами Visual Basic по предлагаемой схеме рисунка 4.9.

Согласно условию предлагается нанести поле линий с минимальным шагом

По результатам проведенных расчетов определим минимальные и максимальные значения *X*-координат точки *H* и углов и α_{min} и α_{max}. Пусть получены следующие значения:

$$H_{min} = 75{,}12,\ H_{min} = 124{,}74,\ \alpha_{min} = 4 \text{ град},\ \alpha_{max} = 24 \text{ град}.$$

Фиксируем границы по оси ОУ с крайним левым и правым значениями, умноженными на 100 для исключения дробных величин.

Получаем соответственно 7512 и 12474. Диапазон колебания данного параметра составляет 12474 – 7512 = 4962. Если на поле формы проведено 20 вертикальных и горизонтальных линий (см. рисунок 4.9), то шаг приращения по оси ОУ 4962 / 20 = 240 единиц. По оси ОХ имеем следующие результаты: (24 – 4) / 20 = 1 единица. Тогда при α = 6,2 град и X_H = 96 (см. рисунок 4.7) номер горизонтальной линии с α = 6 определится как

$$(4 – 2) / 1 + 1 = 3 \text{ линия}.$$

Номер вертикальной линии с X_H = 96

$$(9600 – 7512) / 240 + 1 = 5 \text{ линия}.$$

На пересечении этих линий следует поместить точку с α = 6,2 град и X_H = 96. Аналогично требуется найти координаты остальных точек.

Примечание - Оценить сложность решения такой задачи указанными средствами и точность ее решения.

Задача 4.7 Разработать расчетную схему параллельного смещения пути и программу расчета параметров при $L_3 = 0$.

Задача 4.8 Разработать функцию расчета приращения координат

$$+–Tangens * cos(Alfa\_rad) [* sin(Alfa\_rad)]$$

и использовать ее для вычисления X_F, Y_F, X_I, Y_I по схеме:

private Function Delta_Koord (n, m) As Double

*If n = 1 And m = 1 Then Delta_Koord = - Tangens * cos(Alfa_rad)*
*If n = 2 And m = 2 Then Delta_Koord = Tangens * sin(Alfa_rad)*
End Function

Задача 4.9 Согласно расчетам по схеме рисунка 4.7 *X_C* = 71,53, а *X_H* = 96. Следовательно, схема некорректно фиксирует точки *C* и *H*. Последняя должна быть на 25 м правее, а ее место должна занимать точка *J*. Исправить расчетную схему рисунка 4.7 с корректным положением *H*.

Задача 4.10 Определить и обработать несколько исключительных ситуаций при проведении расчетов координат. Например, если ввести отрицательное значение параметра M_1, то весь последующий расчет будет неверным. Поэтому после ввода данных и перед расчетом необходимо проверить условие ввода положительных значений, а при нарушении этого условия – выдать информационное окно и предложить повторить ввод.

Примечание - Проверку лучше проводить после ввода каждого значения.

Тесты

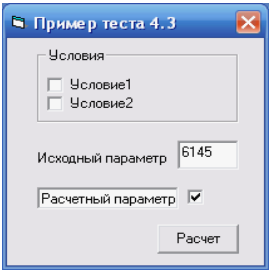
Тест 4.1 Если условия (4.13) не проверять для каждой переменной, то:
1) это повлияет на правильность расчета координат;
2) программа аварийно завершит работу;
3) некоторые расчетные данные не будут визуализированы;
4 сложно реализовать программный расчет координат в дальнейшем.

Тест 4.2 Одна из следующих строк программного кода не является правильной:
1) *Txt_L3.Text = “125”;*
2) *txt_L1.Visible = 0;*
3) *Alfa_Grad.Text = CStr(Alfa * 180 / 3.141593);*
4) *D = val(Z5.Text).*

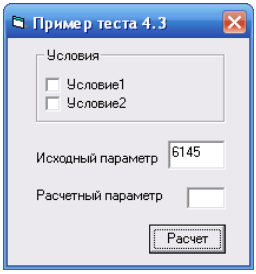
Тест 4.3 Укажите правильный вид экрана, обеспечивающий программную реализацию следующей процедуры:

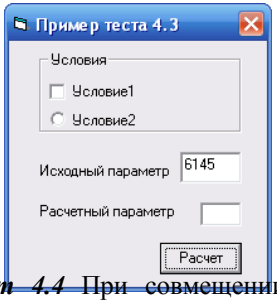
Private Sub Primer ()
*If Check1_Value = 1 Then Txt_W7.Text = 512 * Val(P_Txt.Text) / 5*
*If Check1_Value = 0 Then Txt_W7.Text = 512 * Val(P_Txt.Text) / 10*
*If Check2_Value = 1 Then Txt_W7.Text = 512 * Val(P_Txt.Text) / 15*
*If Check2_Value = 0 Then Txt_W7.Text = 512 * Val(P_Txt.Text) / 25*
End Sub

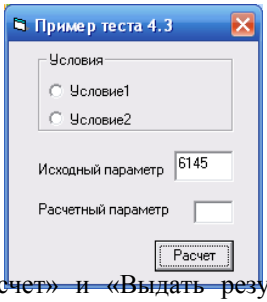
1)



2)



3) 

4) 

Тест 4.4 При совмещении кнопок «Расчет» и «Выдать результаты расчета координат» (см. рисунок 4.3):

1) сложно реализовать алгоритм расчета;
2) программа аварийно завершит свое исполнение;
3) следует аккуратно перенести все данные на одно программное окно;
4) результат будет идентичен исходному.

Тест 4.5 Если в коде процедуры Cmd_Koord_Click() (см. рисунок 4.6) строку

Frm_ Shema.Show vbModal

поместить перед расчетами, то на вызываемой второй форме:

1) не отразится ни один расчетный параметр;
2) программа завершится аварийно;
3) не возникнет никаких проблем;
4) переменные, рассчитываемые после вызова второй формы, будут иметь нулевые значения.

Тест 4.6 В чем различие записей

*L1 = 11 * Val(Txt_Z.Text)*

и

*L1 = Val(CStr(11 * Val(Txt_Z.Text)))*:

1) вторая запись некорректна;
2) первая запись некорректна;
3) никаких различий;
4) обе записи неверны.

Тест 4.7 В чем неточность следующего программного фрагмента:

Const A = 10
Dim B, C As Byte
Dim K As String
.....
B = A + Val(Txt33.K)
C = B / 4.15
.....

1) ошибки отсутствуют;
2) неверное вычисление переменной *C*;
3) результат будет корректным при условии задания значения Txt33.K;
4) строка *C = B* / 4.15 неверна.

Тест 4.8 Определить только ключевые точки, координаты которых позволяют восстановить геометрию пути смещения при известных значениях параметров α и *R* (см. рисунок 4.2):

1) *D*, *E*, *I*, *J*;
2) *A*, *E*, *I*, *H*;
3) все точки;
4) *A*, *B*, *C*, *H*.

Тест 4.9 Возможна ли разработка графических номограмм как визуального представления зависимостей трех и более параметров решения системы уравнений (4.1):

1) задача технически нереализуема;
2) возможна и в некоторых случаях может быть полезна;
3) только при условии использования целого ряда дополнительных данных;
4) если $M_2 > M_1$.

Тест 4.10 Сколько точек с рассчитанными координатами нужно получить, чтобы запроектировать корректный геометрический образ параллельного смещения пути:

1) не менее трех точек;
2) 4 точки;
3) все 8 точек, указанных на расчетной схеме рисунка 4.1;
4) не менее 4 точек.

Контрольные вопросы

1 Являются ли условные обозначения сигналов масштабными изображениями на плане станции?

2 В чем недостаток схемного представления с горизонтальным масштабом 1:2000 и вертикальным 1: 1000?

3 Какие причины могут привести к необходимости изменения междупутья и проектирования смещения пути?

4 Чем будут отличаться схемы по сравнению с рисунком 4.1 при $\alpha_1 > \alpha_2$ и $\alpha_1 < \alpha_2$?

5 Какой минимальный набор координат точек характеризует параллельное смещение пути?

6 Какой визуальный опыт будет получен при реализации *Check_M2.Grayed?*

7 Какие функции может выполнять кнопка «Помощь», помещаемая на программную форму?

8 Можно ли определить положение контрольных точек без координатной привязки в прямоугольной системе? Если можно, то как?

9 В чем различие записей $Z = 5.30$ и $Z =$ “5.30”?

10 Почему на форме Frm_Shema в процедуре Form_Load () нельзя вставить строку *Frm _Shema.Txt_Xd.text = CStr(Xd),*основываясь на результатах расчета процедуры Cmd_Koord_Click() программной формы Form_Puti2 (см. рисунок 4.6)?

5 СОПРЯЖЕНИЕ РАСЧЕТНОЙ СРЕДЫ VISUAL BASIC И ПРОЕКТНОЙ СРЕДЫ AUTOCAD

5.1 Основные функциональные возможности AutoCAD

AutoCAD обладает широкими возможностями автоматизации проведения проектных работ, связанных с расчетом и проектированием элементов путевого развития и технического оснащения станций и перегонов. Как показывает проведенный анализ, набор соответствующих активных инструментов достаточно велик. Их состав можно увидеть после вызова из строки меню ряда опций «Вид – Панель инструментов». В версии AutoCAD 14 насчитывается 14, AutoCAD 2000 – 24, AutoCAD 2002 – 27, AutoCAD 2004 – 29, а AutoCAD 2009 – более 40 различных панелей инструментов. Поочередно нажимая на пустые ячейки слева от названий информационных панелей, можно получить их изображение как фиксированных или плавающих форм.

По существу проектная работа, связанная с подготовкой чертежной документации, обладает общими признаками независимо от ее предметной направленности. Поэтому из ряда инструментальных средств широкого диапазона функций AutoCAD можно выделить такие, которые могут быть с высокой эффективностью использоваться для подготовки цифровых моделей схем железнодорожных станций.

Загрузка среды AutoCAD приводит к возникновению на экране дисплея рабочего стола САПР (рисунок 5.1).

Рабочий стол AutoCAD состоит из следующих основных элементов:

1 – строки меню, включающей основные команды управления пакетом САПР;

2 – стандартной панели инструментов, охватывающей графические изображения кнопок, встречающихся в других программных пакетах (Word, Excel, PowerPoint);

3 – дополнительных панелей инструментов, которые могут быть фиксированными и плавающими. Перемещение панелей по рабочей области экрана дисплея можно осуществить с помощью активной (нажатой) левой клавиши манипулятора мыши и с указателем курсора на названии панели или левом ее крае;

4 – рабочего графического поля (области экрана, используемого для размещения чертежа);

5 – экранного меню;

6 – окна командных строк, в которое можно вводить различный перечень команд;

7 – строки состояния, фиксирующей и контролирующей некоторые особенности использования отдельных команд в данный момент времени.

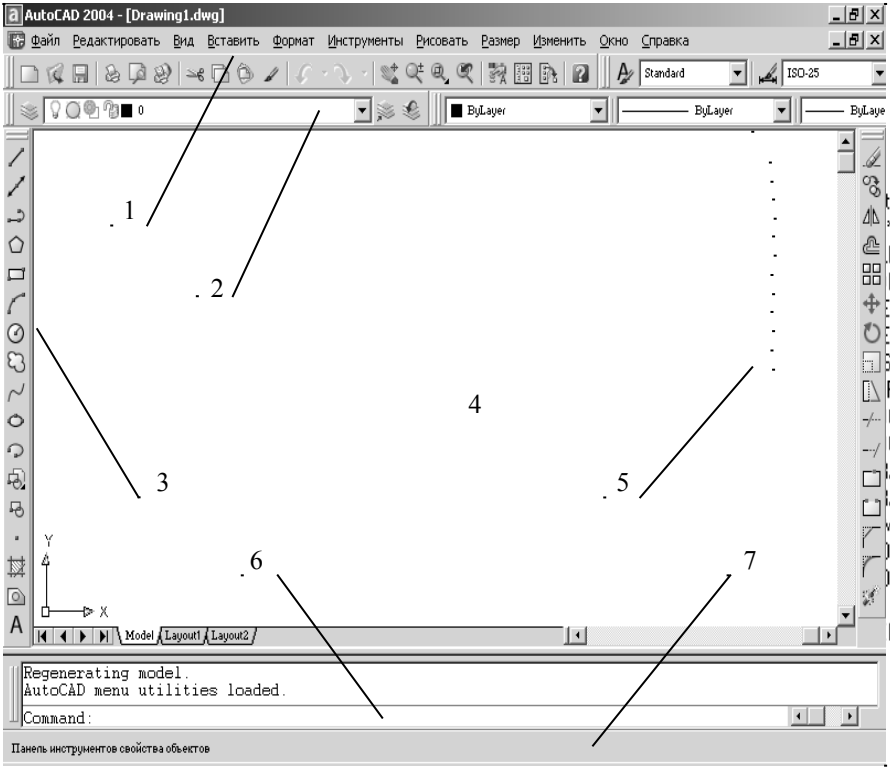


Рисунок 5.1 – Основные элементы загрузочного экрана AutoCAD

Панели инструментов включают различное количество отдельных операций, которые наглядно фиксируются соответствующими графическими значками в виде кнопок. Например, вычерчивание прямой линии определяется некоторым знаком, нажатие на который вызывает соответствующую программную функцию. Если задержать указатель курсора на данном значке, то рядом с ним и внизу в строке состояния появится надпись, подсказывающая проектировщику ожидаемое действие после нажатия на данную кнопку.

Для увеличения видимой области поля можно удалить экранное меню с помощью вызова из строки меню «Инструменты – Соглашения – Экран», и открывшемся окне снять флажки у строк «Показать экранное меню AutoCAD в окне рисования» и «Показать линейки прокрутки в окне рисования» (рисунок 5.2).

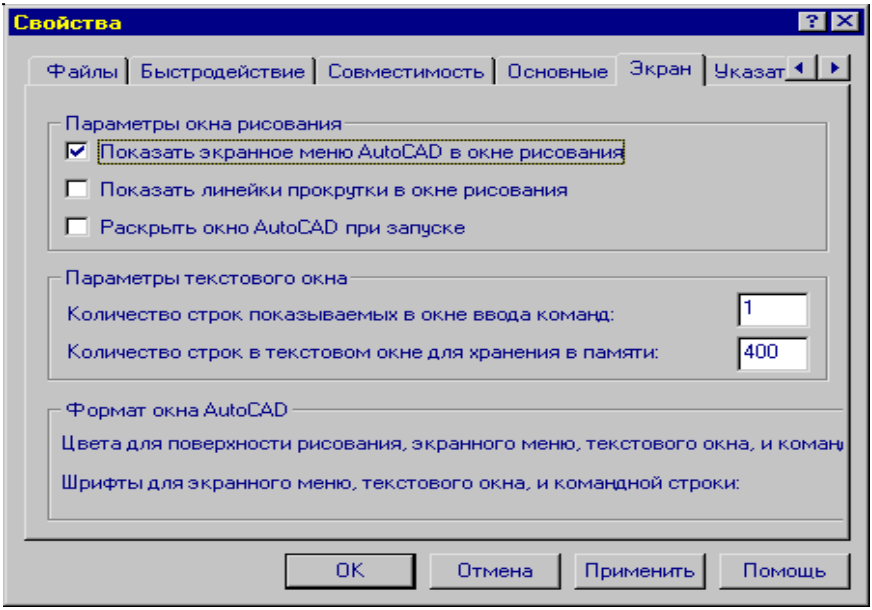


Рисунок 5.2 – Установка параметров окна рисования AutoCAD

Анализ возможностей AutoCAD показывает, что только незначительная их часть может активно использоваться для автоматизированного проектирования портов. Поэтому на экране рекомендуется закрепить только те графические изображения инструментов, которые автоматизируют процесс подготовки цифровой схемы железнодорожной станции.

Важными элементами на рабочем столе AutoCAD является маркер, положение которого изменяется при перемещении мыши, с которым связывается отображение координат положения маркера в строке состояния. Эти данные можно удалять или восстанавливать с помощью клавиши <F6>.

Отсчет координат при загрузке AutoCAD производится от нулевой точки, находящейся в нижнем левом углу графического поля. В данном месте располагается также изображение системы координат.

Инструменты проектирования отрезка и полилинии можно использовать для укладки прямолинейных и кривых участков. Вызов первого инструмента (или ввод в командной строке команды “LINE” + <ENTER>) требует указания

точки на графическом поле (From point). Задание исходной точки приводит к появлению луча с динамическим изменением длины отрезка при изменении положения курсора на экране дисплея. Ответ на запрос “To point:” путем указания второй точки приводит к построению требуемого участка прямого пути. При этом команда вычерчивания отрезка не завершает своей работы, а в командной строке снова выводится сообщение “To point:”. Выбор новой точки на экране приводит к продолжению построения ломаных отрезков. Однако в нашем случае следует после построения конца первого отрезка завершить действие команды LINE нажатием клавиш <ENTER> либо <ESC>, так как участки могут не сопрягаться посредством ломаных линий.

Активизация кнопки полилинии или указание в командной строке команды PLINE требует задания исходной точки отрезка: “From point:”, что аналогично действию команды LINE. После выбора места положения начальной точки на графическом поле необходимо выбрать одно из возможных продолжений строки команд, указываемых в командной строке:

Arc/Close/Halfwidth/Length/Undo/Width/<Endpoint of line>:

по начальным символам (*A, C, H, L*, …). Вычерчивание кривой обеспечивается выбором команды Arc и вводом в командную строку символа *A*. В результате укладывается кривая с началом в исходной точке, определенным направлением и радиусом. Направление кривой определяется положением касательной к проектируемой кривой, проходящей через начальную точку. Радиус пропорционален расстоянию между начальной и конечной точками, фиксируемыми на экране дисплея как хорды переменной длины.

Таким образом, команда PLINE является универсальной, обеспечивающей построение как прямых, так и криволинейных участков. Данный факт оказывается очень важным, так как позволяет проектировать связные участки, состоящие из элементов прямых и кривых, автоматически сопрягаемых по соответствующим касательным.

Проектирование технического оснащения станций (зданий, сооружений), имеющих прямоугольную форму, удобно производить с помощью команды RECTANG или инструмента  . После его активизации нужно выбрать места размещения противоположных точек контура объекта.

Если требуется повернуть запроектированный объект прямоугольной формы на определенный угол, то следует применить команды «Изменить-Повернуть» из основного меню. В командной строке отражается запрос Select objects (Выбрать объекты), а указатель курсора на экране превращается в небольшой прямоугольник. После выбора одного или нескольких объектов нажимаем на правую кнопку мыши и отвечаем на программный запрос Base point (Базовая точка поворота объекта). Следующий запрос Rotation angle (Угол поворота) требует указания угла поворота в градусах. После выполнения указанных операций на экране вместо

выделенного одного или нескольких объектов, к которым были применены команды «Изменить-Повернуть», возникают новые объекты с установленными свойствами (размерами и углом поворота).

Для полноты схемы важно использовать соответствующий инструмент формирования поясняющих надписей (наименований зданий и сооружений, парков станций, подъездных путей и др.). Ввод текста на графическое поле обеспечивается выбором из меню последовательности команд «Рисовать-Текст-Однострочный текст». На запрос <Start point>: следует указать курсором точку вставки текста. Высота символов определяется следующим параметром Height. В командной строке необходимо указать требуемый размер или подтвердить предлагаемый нажатием клавиши <ENTER>. Рекомендуемая высота вспомогательного текста на схеме станции 1,8–3,0 единицы. Строку текста можно повернуть на любой угол, который указывается параметром Rotation angle командной строки. Текст вводится в командную строку после указанной команды TEXT. Ввод завершается двойным нажатием клавиши <ENTER>.

Многострочный текст можно вводить с использованием такой же последовательности команд «Рисовать-Текст-Однострочный текст» и набором строк после каждого нажатия клавиши <ENTER>. Аналогичные возможности предоставляет команда TEXT, которая вводится в командной строке.

Если возникает необходимость изменить размер строки текста, выведенной на графическое поле, то нужно выделить данную строку (щелкнуть левой кнопкой мыши с активным указателем курсора на тексте). В основном меню AutoCAD выбрать «Изменить-Масштабировать». При этом контуры строки текста изменятся на штриховые. Далее нужно активизировать курсор в области размещения текста и перемещением мыши выбрать новый масштаб изображения строки текста.

Редактировать выведенную на графическое поле строку текста можно с помощью команд основного меню «Изменить-Объект-Текст». После указания курсором на некоторый текст появляется дополнительное окно, в котором можно отредактировать строку текста. Динамическое уменьшение и увеличение изображения на экране достигается использованием соответствующей графической кнопки и перемещением курсора по оси ОУ или вращением колеса мыши.

Экранное представление схемы станции, как правило, не позволяет расположить весь чертеж в границах графического поля рабочего стола AutoCAD. Для просмотра оставшейся за пределами окна части схемы служит соответствующий инструментарий. После его активизации в положении курсора вместо перекрестья появляется другое изображение. При нажатой левой кнопке мыши можно переместить всю электронную схему на экране с исходным масштабом в любом направлении.

Использование базовых примитивов AutoCAD для проектирования путевого развития (Line, Arc) позволяет с ограниченными затратами времени вычертить электронный аналог схемы станции. Однако лишь в первом приближении такой цифровой чертеж можно применять для решения практических задач. Как правило, требуется проведение ряда конструкционно-геометрических мер, способствующих улучшению наглядного представления схемы. Требованиями по проектированию железнодорожных станций устанавливается необходимость выделения главных путей (жирными линиями), узкой колеи (штриховыми линиями), удлиняемых и вновь укладываемых путей (красными линиями). Поэтому после проведения основных проектных операций необходимо использовать дополнительные программные средства AutoCAD для работы над проблемными объектами, не соответствующими нормам проектирования.

Для укладки железнодорожных путей на электронной схеме станции можно использовать команду PLINE с выбором параметра Width (ширина) и указанием значений 0,35 для начальной и конечной точек (Starting width, Ending width). В этом случае железнодорожные пути на схеме заметно выделяются по сравнению с шириной станционных путей. Штриховые линии проектируются на схеме с помощью кнопки  или команды LINETYPE. Результатом активизации данной кнопки или команды является информационное окно (рисунок 5.3).

Если в указанном поле нет требуемой штриховой линии, то следует нажать кнопку «Загрузить», выбрать тип Dashed и активизировать кнопку ОК. Данный тип линии отобразится в основном окне (см. рисунок 5.3). Далее следует отметить строку с типом Dashed и нажать на кнопку ОК.

В стандартной панели инструментов нужно указать на графическое изображение команды «Управление типами линий»

и из предлагаемого перечня выбрать тип Dashed. В дальнейшем использование инструментов LINE, PLINE, RECTANG, CIRCLE будет приводить к построению объектов со штриховыми контурами.

При переустройстве станций и необходимости укладки дополнительного путевого развития парков и станций, выделенного красным цветом, необходимо использовать инструмент «Управление цветом»

Из выпадающего списка нужно отметить строку «Красный» или при отсутствии данного цвета «Другие». Теперь до отмены данной команды все проектируемые элементы станции будут отображаться красным цветом.

Требованиями ЕСКД устанавливается внешний вид изображений технических устройств, зданий и сооружений, вычерчиваемых на

соответствующей схеме. Складские помещения, производственные и административные здания, грузовые и пассажирские платформы дополняются определенной штриховкой.

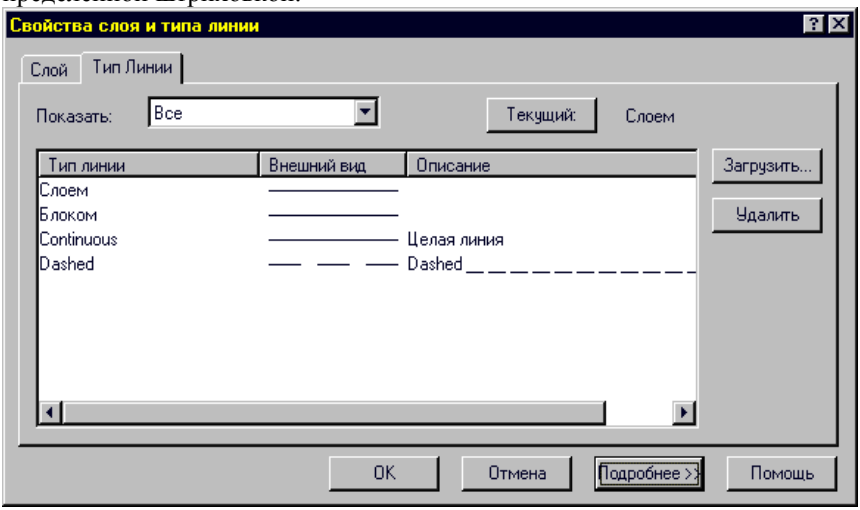


Рисунок 5.3 – Окно выбора типа линий

AutoCAD имеет графические инструменты, позволяющие выполнить данную работу в автоматизированном режиме. Первоначально нужно подготовить замкнутый контур, обеспечиваемый инструментами

или соответствующими командами LINE, PLINE, RECTANG, CIRCLE. Далее следует активизировать кнопку «Штриховка» или указать в командной строке команду BHATCH, что приведет к появлению нового программного окна выбора типа шаблона (штриховки), его свойств (масштаба, угла наклона линий штриховки), точки внутри замкнутого контура или всех границ контура. Рекомендуется использовать возможность задания одной точки, после указания которой достаточно щелкнуть правой кнопкой мыши и перейти к окну выбора характеристик шаблона с установленными параметрами. После нажатия на кнопку «Применить» результат действия команды отображается на электронной схеме. Если требуется перенести данный тип штриховки на несколько объектов, то следует многократно повторить выбор точек внутри геометрически заданных контуров и нажать правую кнопку мыши.

Схема в цифровом виде представляет собой чертеж, выполненный средствами САПР, визуально определяемый на графическом поле экрана дисплея. Из-за большой протяженности изображения путевого развития

электронная схема может быть зафиксирована на экране дисплея только как определенный фрагмент, видимый в некотором окне. Средствами перемещения схемы по графическому полю можно просмотреть заполнение всей станционной площадки. Однако для распечатки всей схемы принтера формата А4, как правило, недостаточно. Попытка разместить схему станции в пределах размеров листа приводит к очень мелкому масштабу с неразличимыми деталями. Например, результатом распечатки схемы станции средствами лазерного принтера формата А4 является чертежный аналог масштаба менее 1:10 000.

Выдачу на бумагу протяженных чертежей обеспечивает плоттер, поддерживающий рулонную печать. Особенностью работы плоттера является обязательный выбор масштаба чертежа, являющегося основной характеристикой печатаемого на плоттере документа. Для проведения операций печати схемы станции на плоттере необходимо установить плоттер как активное устройство из меню «Сервис-Установка-Принтер» (рисунок 5.4).

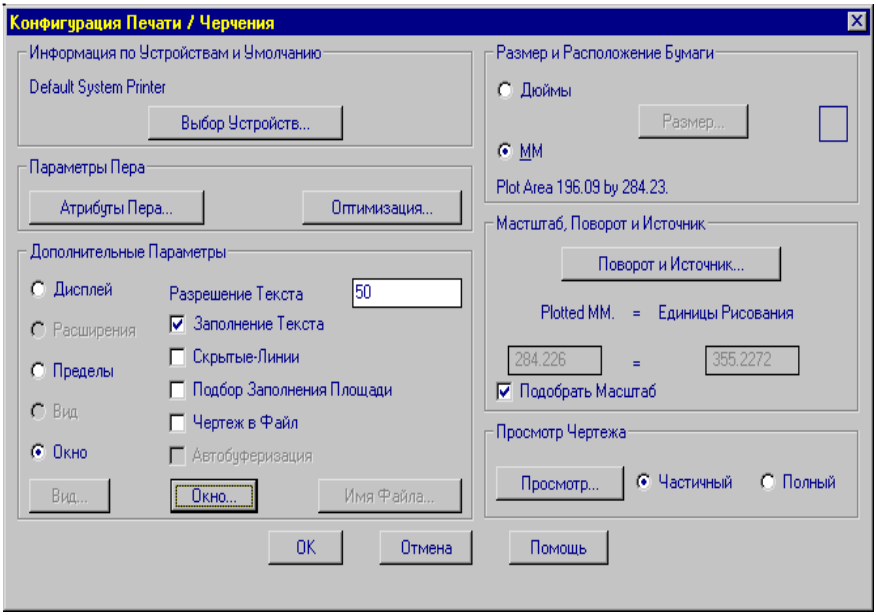


Рисунок 5.4 – Вид основного окна печати документа

Выбор параметров печати производится посредством вызова основного окна печати («Файл-Печать-Окно»). Опция «Выбрать» позволяет определить границы чертежа, в пределах которых документ должен быть выдан на печать. Кнопка «Просмотр» с активной опцией «Полный» позволяет предварительно до печати просмотреть вид чертежа, каким он будет на твердой копии. Кнопка

«Размер» устанавливает размеры листа, на котором производится печать. Высота фиксирует ширину рулона, а длина оценивается экспертно с учетом масштаба печати.

Масштаб определяется путем сравнения числа точек экрана («Plotted MM») и соответствующего числа точек печатной копии («Единицы рисования»). Например, если требуется распечатать схему станции в масштабе 1:2000, то значения параметров Plotted MM = 1, «Единицы рисования» = 2. Если флажок «Подобрать масштаб» активен, то производится программный подбор масштаба печати, при котором выбранный фрагмент полностью размещается в пределах указанного размера листа. С помощью опций печати «Файл-Печать-Окно-Выбор» указывается требуемый фрагмент схемы с последующим выводом его на принтер.

5.2 Совместное использование результатов работы программы Visual Basic и AutoCAD

Среда визуального программирования, построенная на основе Visual Basic (VB) и используемая для расширения возможностей других приложений (Word, Excel, AutoCAD), называется Visual Basic for Application (VBA). По существу – это VB с добавлением специальных функций, обеспечивающих тесную интеграцию данной среды программирования с конкретным приложением. Так как VB (или VBA) обеспечивают расчет устройств, а проектирование осуществляется в AutoCAD после получения результатов программы, то загружать AutoCAD можно посредством функций на VBA. Поэтому следует иметь доступ от VBA к AutoCAD. Такая возможность осуществляется активизацией в программе на VBA библиотеки типов объектов AutoCAD. Для AutoCAD 14 в меню Project необходимо выбрать опцию References и в выпадающем диалоговом окне установить флажок напротив строки «AutoCAD R14 Object Library» (рисунок 5.5).

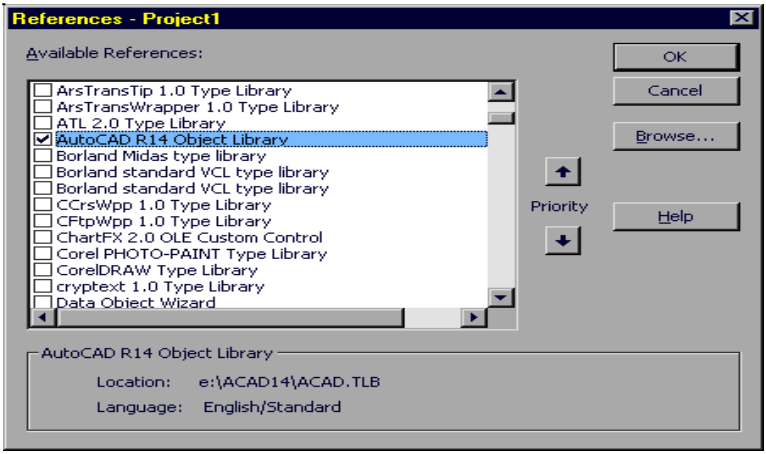


Рисунок 5.5 – Активизация библиотеки объектов AutoCAD в VBA

После обеспечения доступа к объектам AutoCAD в редакторе VB можно просмотреть соответствующий список посредством нажатия функциональной клавиши <F2> или в меню View выбрать опцию Object Browser. Слева после прокрутки указанного ряда Classes можно увидеть имена, начинающиеся с «IAcad…» (рисунок 5.6).

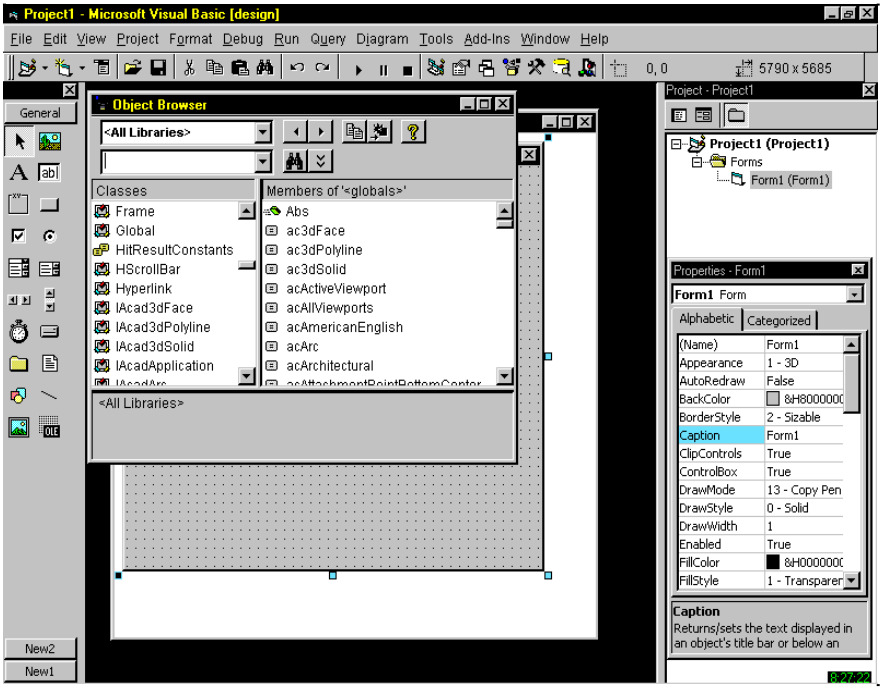


Рисунок 5.6 – Перечень классов работы с AutoCAD, доступных из среды VBA

Следует обратить внимание, что классы работы с AutoCAD помещаются в Browser не в начале, а после основных рабочих классов VBA. Соответствующий указатель вертикальной линейки прокрутки (см. рисунок 5.6) находится на расстоянии 1/3 от всей длины бегунка.

Все свойства и методы указанной функции AutoCAD можно получить из данного окна нажатием клавиши <F1>. При необходимости можно загрузить пример, который имеется в каждом вспомогательном информационном окне. Для получения доступа к функциям можно из программы на VBA выполнять

любые действия, управляющие средой проектирования. Например, можно загрузить и активизировать AutoCAD.

Допустим, на основе фрагмента программной формы, приведенной на рисунке 5.7, следует обеспечить доступ к AutoCAD.

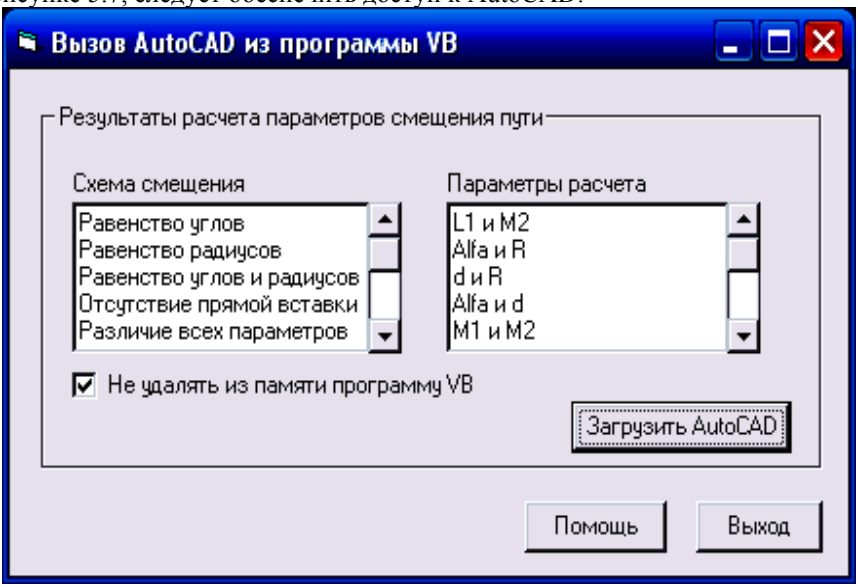


Рисунок 5.7 – Общий вид программы сопряжения с AutoCAD

Исходный код соответствующих функций имеет вид:

***- загрузки AutoCAD*:**

```
Public Acad As Object
Private Sub Command1_Click()
On Error Resume Next
Set Acad = GetObject(, "AutoCAD.Application")
If Err Then
Err.Clear
Set Acad = CreateObject("AutoCAD.Application")
If Err Then
MsgBox "Unable to connect to AutoCAD"
Exit Sub
End If
End If
cmdQuit.Enabled = True
End Sub
```

***- активизации AutoCAD*:**

```
Private Sub_Check1_Check()
```

Acad.Visible = True
End Sub

Для версий AutoCAD 2000 и далее вызов редактора VBA производится нажатием клавиш <ALT+F11>. В результате открывается новое окно, в меню которого следует открыть новую форму активизацией пунктов меню Insert-UserForm (рисунок 5.8).

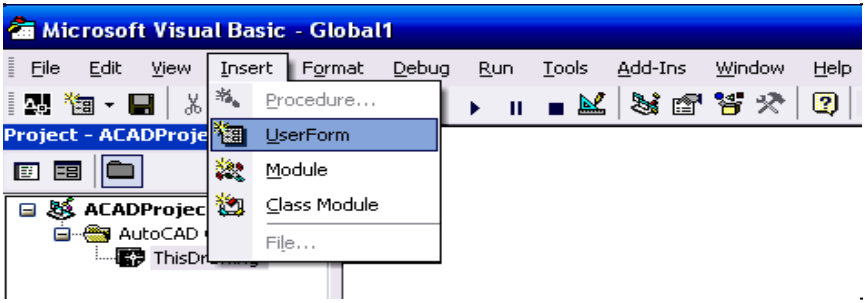


Рисунок 5.8 – Вызов программной формы из меню VBA

Вместе с формой на экране дисплея появляется ряд инструментов, идентичных редактору Visual Basic (рисунок 5.9).

Определим общий вид программы построения кривого участка пути по заданным параметрам, вычисленным ранее (рисунок 5.10).

Исходный код данной программы следующий:

Private Sub CommandButton1_Click()
Dim ArcObj As AcadArc
Dim CenterPoint(0 To 2) As Double _ Dim Radius As Double
Dim StartAngle As Double _ Dim EndAngle As Double
CenterPoint(0) = Val(TextBox2.Text)_ CenterPoint(1)=Val(TextBox3.Text)
CenterPoint(2) = Val(TextBox4.Text) _ Radius = Val(TextBox1.Text)
*StartAngle = (Val(TextBox5.Text)) * 3.141593 / 180#*
*EndAngle = (Val(TextBox6.Text)) * 3.141593 / 180#*
***Set ArcObj** = ThisDrawing.ModelSpace.**AddArc**(CenterPoint, Radius,*
End Sub *StartAngle, EndAngle)*

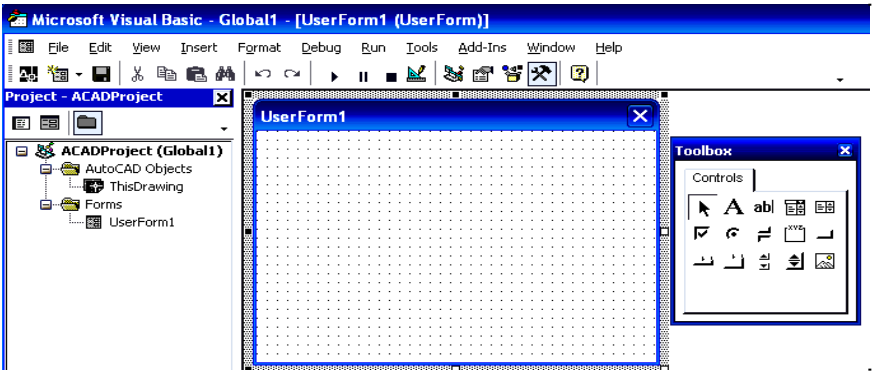


Рисунок 5.9 – Общий вид окна VBA в AutoCAD

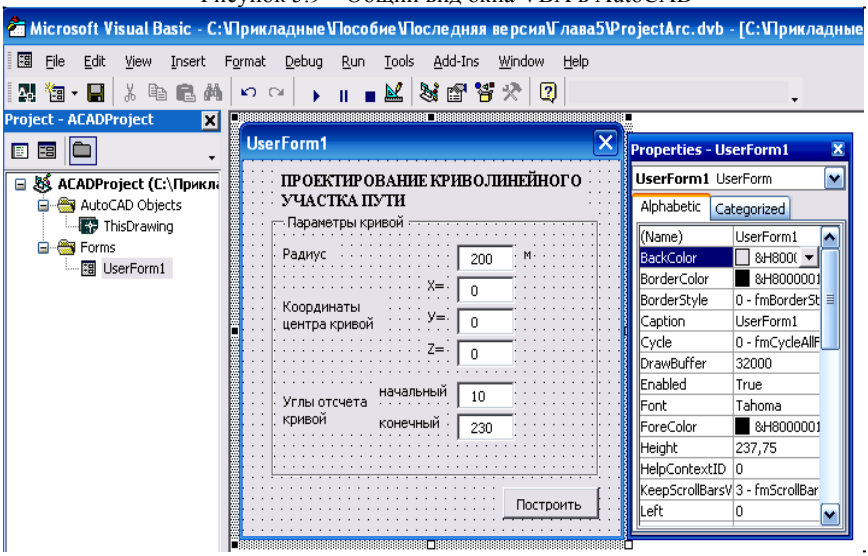


Рисунок 5.10 – Интерфейс программы построения кривого участка пути

После исполнения данной программы на рабочем поле AutoCAD можно зафиксировать построенную дугу, которая по заявленным параметрам будет характеризовать соответствующий участок криволинейного пути станции или перегона.

Таким образом можно вычерчивать в среде AutoCAD любые графические элементы, идентифицирующие соответствующие объекты путевого развития перегона и станции. Следует отметить, что особое внимание при таком программировании нужно обратить на обязательное указание метода, посредством которого будет производиться вывод графического примитива в AutoCAD. В приведенном примере выделены две программные строки описания метода

Dim ArcObj As AcadArc

и его использования

Set ArcObj = ThisDrawing.ModelSpace.AddArc(CenterPoint, Radius, StartAngle, EndAngle).

Так, для вывода окружности требуется использование метода AddCircle, для вывода прямой линии – метод AddLine, для текстовой строки – метода AddText. После активизации программы на Visual Basic, вызываемой из AutoCAD, следует скрыть программную форму VB посредством строки кода:

UserForm1.Hide.

Восстановление окна достигается с помощью строки *UserForm1.Show*.

Разработка программ, в которых выполняются проектные и расчетные задачи, производится в среде AutoCAD с последующим вызовом VBA. Такое соединение двух оболочек обеспечивает решение сложных проблем многоуровневых расчетов и проектирования с интерактивным анализом проектных решений. Диалоговый режим работы достигается оперативным изменением или отладкой расчетных модулей и последующим репроектированием полученного ранее решения. Как показывает практика, это наиболее эффективный путь, позволяющий в кратчайшие сроки провести тестирование сложной расчетно-проектной задачи.

5.3 Характеристика языка AutoLisp

5.3.1 Общие положения

Система автоматизированного проектирования элементов железных дорог может быть построена на базе AutoCAD. Средства расчета различных параметров парков базируются на языке программирования AutoLISP, который органично связан со средой AutoCAD и позволяет развивать существующие базовые функции и создавать новые, способствующие эффективному решению возникающих проектных задач узкопрофильного характера.

Язык AutoLISP является модификацией распространенного языка логического программирования LISP, который позиционируется как программная среда обработки символьной информации и естественных языков, создания экспертных систем и искусственного интеллекта. Этот язык обладает мощными и одновременно простыми и понятными средствами, позволяющими выполнять в автоматизированном режиме расчеты и проектирование станционных объектов. LISP – это язык, имеющий много диалектов, включая MacLISP, InterLISP, ZetaLISP, CommonLISP. AutoLISP - наиболее близок по синтаксису и соглашениям к CommonLISP, но является небольшой его частью, хотя имеет много дополнительных функций, отражающих специфику AutoCAD.

Отличительной особенностью AutoLISP является развитая возможность работы со списками. Значительное число функций AutoLISP обеспечивает обработку списков (выделение элементов, слияние, перестановка, сортировка). Список является всеобъемлющим понятием этого языка программирования. Его название LISt Processing – это прямое указание на методологическую направленность его функционирования. Даже записи команд AutoLISPa оказываются одним большим списком. Такая приверженность единому подходу в представлении обрабатываемой информации является преимуществом данного языка, так как позволяет программно вносить изменения в исходный код, формируя таким образом самоизменяющиеся (полиморфные) функции, которые работают каждый раз по-разному в зависимости от складывающейся проектной ситуации. Программно можно накапливать базу знаний, используемую полиморфной оболочкой, что позволит ей работать в наиболее эффективном режиме. Поэтому нет принципиальных ограничений на разработку самосовершенствующихся программных функций с использованием AutoLISP.

Список в AutoLISP формируется с помощью определенных команд, которые объединяют последовательность элементов в связное подмножество. По внешнему виду список – это набор символов, разделенных пробелами и заключенных в круглые скобки. Однако простая запись вида

(*a b c d e*)

не является списком в смысле понимания его AutoLISP. Обязательно к такой исходной записи следует применить команду конструирования списка. Часто в примерах можно встретить упрощенную запись сконструированного списка в виде

'(a b c d e).

AutoLISP обладает всеми базовыми возможностями стандартного LISP. В некотором смысле AutoLISP оказывается более предпочтительным для автоматизации проектирования, так как он непосредственно связан с AutoCAD. Практически любая функция AutoCAD может быть вызвана программно соответствующей строкой кода на AutoLISP. Данный симбиоз позволяет автоматизировать сложные цепочки последовательности проектных операций и значительно сократить время проектного процесса за счет выполнения потокового проектирования. Как показывают исследования, именно благодаря пакетному режиму работы программы, когда ввод блока исходных данных обеспечивает проектирование сложной многообъектной структуры, достигается сокращение продолжительности разработки проекта в 3–3,5 раза.

Соединение в одной функции на AutoLISP расчетных и проектных операций позволяет реализовать процесс в «фоновом» режиме. На экране дисплея формируется результат проведенных расчетов, сравнений возникающих вариантов, их оценки по определенным критериям. Программные функции могут работать и в интерактивном режиме, останавливаясь после каждого единичного расчетного или проектного шага и ожидая разрешения проектировщика на проведение следующего шага. Эти два режима легко реализуются программно на AutoLISP, практически не отличаясь друг от друга по виду исходного кода.

САПР может определиться как программная оболочка благодаря возможностям AutoLISP. Модульная структура функций позволяет легко модифицировать их, изменяя в соответствии с нормами проектирования и дополнительными требованиями. Поэтому САПР является открытой средой, которая может (и должна) развиваться на основе возможностей языка AutoLISP.

AutoLISP поддерживает несколько различных типов данных, основными из которых являются:

- целые числа;
- действительные числа;
- строковые константы;
- списки.

Ввод данных в программе на AutoLISP может осуществляться с клавиатуры в сеансе работы AutoCAD, считываться из файла или из строковой константы. При разработке программ на AutoLISP должны соблюдаться следующие условия:

- имена символов могут состоять из любой последовательности печатных знаков, исключая такие символы как ( ) . ' " ;

- выражения могут занимать несколько строк;

- несколько пробелов между символами эквивалентны одному пробелу. Отступы в строках не требуются, однако их можно использовать для повышения наглядности структуры функций;

- имена символов и функций в AutoLISP могут быть набраны прописными и строчными символами и не могут начинаться с цифры;

- целые константы могут начинаться с необязательных символов "+" или "-";

- буквенные (строковые) константы как последовательность знаков, взятая в кавычки. Внутри строковых констант можно ввести соответствующие управляющие символы, пользуясь знаком обратной черты (\). Коды символов следующие: \\ означает знак \, \e эквивалентен символу escape-последовательности, \n − новой строке (newline), \r − возврату каретки (return), \t − табуляции (tab).

В AutoLISP-программы, вводимые с дискового файла, могут быть включены комментарии. Комментарии начинаются с точки с запятой и продолжаются до конца строки.

AutoLISP имеет специфическую программную структуру, называемую вычислителем. Вычислитель анализирует строку, введенную пользователем, на предмет наличия ошибок, вычисляет ее и возвращает определенный результат.

Особо следует отметить правила записи выражений. В командах принята, так называемая, обратная польская запись, в которой математическая операция предваряет числа или переменные, участвующие в операции.

Например, результатом записи

(– 15 6)

является число 9, а

(+ 5 6 7) – число 18.

Можно формировать достаточно сложные выражения:

(* (+ 7 (– 6 4)) (+ 8 (– 4 1))).

Аналогом этой записи является принятая форма записи:

((6 – 4) + 7) * ((4 – 1) + 8)) = 99.

Контроль правильности записи как списка обеспечивают левые и правые скобки, которых обязательно должно быть одинаковое количество в каждой записи.

5.3.2 Основные команды AutoLISP

По назначению функции языка AutoLISP разделяются на следующие основные группы:

- ввода данных;
- вывода данных и результатов;
- операций с данными и выражениями;
- преобразования аргументов;
- операций со списками;
- операций со строками;
- операций выполнения функций;
- проверки выполнения условий.

Функции ввода данных.

GETINT – ввод целого числа.

Форма записи соответствующей команды следующая:

(GETINT “Число путей парка приема = ”).

После загрузки данной команды программа будет ожидать ввода проектировщиком любого целого числа.

GETREAL – ввод действительного числа.

Команда выглядит следующим образом:

(GETREAL “Значение коэффициента неравномерности прибытия = ”).

GETSTRING – ввода строки текста.

(GETSTRING “Тип маневрового локомотива ”).

GETPOINT – ввод списка значений координат точки на экране дисплея.

(GETPOINT “Ввести точку установки осветительной мачты в горловине”).

Операции вывода данных и результатов.

ALERT – отображает на экране окно с определенным сообщением.

(ALERT “Междупутье недостаточно для сооружения платформы”).

PRIN1 – возвращает и выводит указанное выражение в командную строку экрана.

(PRIN1 “Проектирование входной горловины парка приема завершено”).

PRINC – команда работает аналогично предыдущей за исключением того, что указанное выражение не переносится на новую строку.

PRINT – команда работает аналогично предыдущей с переносом выражения на следующую командную строку.

PROMPT – выводит указанное сообщение в командной строке экрана дисплея и возвращает nil.

(PROMPT “Величина уширенного междупутья не задана”).

Операции с данными и выражениями.

SETQ – присвоение переменной значения или результата выражения.

(SETQ *a* 5) – присвоение переменной *a* значения, равного 5.

(SETQ *b* "*XYZ*") – присвоение переменной *b* значения текстовой константы, равной "*XYZ*".

(SETQ *c* (SIN 0.2)) – присвоение переменой *c* результата расчета (SIN 0.2).

Если в программе следуют несколько присвоений переменных (как в приведенном выше примере), то возможны следующие эквивалентные варианты записи этих выражений:

1. (SETQ *a* 5
   b "*XYZ*"
   c (SIN 0.2)
)
2. (SETQ *a* 5 *b* "*XYZ*" *c* (SIN 0.2)).

DEFUN – определение названия функции. Любая функция в AutoLISP начинается командой DEFUN (DEfinition FUNction). Имя функции может не иметь параметров, содержать один или несколько параметров. Например,

(DEFUN Start (A1 A2 / A3 A4)

…….,

где Start – имя функции;

*A*1, *A*2 – имена глобальных переменных;

*A*3, *A*4 – имена локальных переменных.

При вызове функции Start требуется ввод значений двух переменных *A*1 и *A*2. Для переменных *A*3, *A*4, указываемых после знака «/», резервируется память, однако они никаким образом не фигурируют при вызове данной функции. Поэтому указанную функцию можно вызывать следующим образом: (Start 2 3).

LOAD – загрузка указанного файла AutoLISP в память.

GETVAR – возвращение значения системной переменной AutoCAD.

(SETQ *R* (GETVAR "FILLETRAD")) – в переменную *R* считывается значение радиуса сопряжения, сохраняемого AutoCAD.

SETVAR – присвоение нового значения указанной системной переменной.

(SETVAR "FILLETRAD" 600) – радиус сопряжения равен 600 м.

PROGN – вычисление связной последовательности выражений.

```
(PROGN
  (SETVAR "OSMODE" 0)
  (SETQ w 1
        x 0
  )
)
```

COMMAND – выполнение указанной команды AutoCAD.

(COMMAND “LINE” ‘(1 1) ‘(10 10) “”) – вычерчивание участка прямого пути с координатами начала (1 1) и конца (10 10).

Преобразование аргументов.

ANGTOS – преобразование значения угла в радианах в строковую константу с заданной формой представления и точностью.

(ANGTOS 0,785398 0 4) – угол в радианах переводит в градусы с точностью до четырех знаков после запятой и возвращает результат “45.0000”.

ATOF – преобразование строковой константы в вещественное число.

(ATOF “5”) приводит к результату 5.0.

ATOI – преобразование строковой константы в целое число.

(ATOI “44”) возвращает значение 44,

(ATOI “22.74”) возвращает значение 22.

ITOA – преобразование целого числа в строковую константу.

(ITOA 23) возвращает значение строки “23”.

FIX – преобразование вещественного числа в целое (выделение целого числа из вещественного).

(FIX 5.8) возвращает значение 5.

FLOAT – преобразование целого числа в вещественное.

(FLOAT 2) возвращает значение 2.0.

RTOS – преобразование вещественного числа в строку.

Операции со списками.

APPEND – объединяет заданное количество списков в один.

(APPEND ‘(4 3) ‘(6 5)) возвращает результат ‘(4 3 6 5).

CAR – выбирает из указанного списка первый элемент.

(CAR ‘(2 3 4)) возвращает в качестве результата элемент, равный 2.

CDR – возвращает список без первого элемента.

(CDR ‘(2 4 6 8)) определяет результат ‘(4 6 8).

CONS – формирует новый список из двух указанных в порядке следования их как аргументов данной функции.

(CONS ‘(*x*) ‘(*y z*)) возвращает новый список ‘(*x y z*).

LAST – выбирает из указанного списка последний элемент.

(LAST’(2 3 4 5 6)) возвращает 6.

LENGTH – определение числа элементов в списке.

(LENGTH ‘(1 2 3 5 6 7)) возвращает значение 6.

LIST – формирует список из указанного числа элементов в порядке их следования как аргументов данной функции.

(LIST 2.55 1.71 6.58) возвращает ‘(2.55 7.71 6.58).

LISTP – проверяет, является ли указанный аргумент списком.

(LISTP ‘(*a b c*)) возвращает *T*.

(LISTP 5.4) возвращает nil.

MEMBER – анализ списка и возвращение части списка, начинающегося с найденного выражения.

(MEMBER ‘*a* ‘(*z w v a b c*)) возвращает ‘(*a b c*).

NTH – выбор из списка элемента с указанным номером по порядку их следования в списке. Первый элемент в списке имеет номер 0.

(NTH 0 ‘(*v w x y z*)) возвращает *v*.

REVERSE – расстановка элементов списка в обратном порядке.

(REVERSE ‘(2 3 4 5)) возвращает список ‘(5 4 3 2).

SUBST – замена новым элементом старого в указанном списке.

(SUBST ‘*x* ‘*a* ‘(*a y z*)) возвращает новый список ‘(*x y z*).

Операции со строками.

STRLEN – определение длины строковой константы.

(STRLEN “стрелочный перевод”) – возвращает значение 18.

STRCAT – объединение нескольких строк в одну.

(STRCAT “веерная” “ ” “стрелочная ” “улица”) – возвращает строку «веерная стрелочная улица».

SUBSTR – выделение части строки, начиная с указанного символа.

(SUBSTR “allnumber” 4) – возвращает строку “number”.

(SUBSTR “allnumber” 4 7) – возвращает строку “numb”.

Операции выполнения функций.

ABS – вывод абсолютного значения числа.

(ABS –4) возвращает значение 4.

ATAN – вычисление арктангенса аргумента, выраженного в радианах.

COS – вычисление косинуса угла, выраженного в радианах.

EXP – возведение основания натурального логарифма *e* в степень.

(EXP 2.2) возвращает результат вычисления $e^{2.2}$.

EXPT – возведение указанного числа в степень.

(EXPT 2 5) возвращает результат вычисления 2^5.

LOG – вычисление натурального логарифма указанного числа.

(LOG 4.2) возвращает результат вычисления *ln* 4.2.

MAX – определение большего из числа заданных аргументов.

(MAX 3 4 5) возвращает значение 5.

MIN – определение минимального из числа заданных аргументов.

(MIN 5 6 7 8) возвращает значение 5.

REM – определение остатка от деления двух чисел.

(REM 7 3) возвращает значение 1.

SIN – вычисление синуса угла, выраженного в радианах.

SQRT – определение значения квадратного корня из аргумента.

(SQRT 2) возвращает значение 1.41421.

Функции проверки выполнения условий.

IF – сравнение указанных выражений.

(IF (> *a b*)… – сравнивает значения *a* и *b* (если *a* > *b*, то …).

EQ – сравнение двух элементов на идентичность.

AND – возвращение результата выполнения операции логического И над указанными элементами. Результат может быть nil (если хотя бы один элемент равен nil) или *T*.

COND – сравнение ряда условий и выбор выполняемого на данный момент.

(COND ((= *a* “Д”) 1)
((= *a* “д”) 1)
((= *a* “*Y*”) 1)
((= *a* “*y*”) 1)
((= *a* “Н”) 0)
((= *a* “н”) 0)
((= *a* “*N*”) 0)
((= *a* “*n*”) 0)

) – возвращает 0 или 1 в зависимости от того, чему равно значение *a*.

OR – возвращение результата выполнения операции логического ИЛИ над указанными элементами. Результат может быть *T* (если хотя бы один элемент не равен nil).

REPEAT – выполнение заданных команд указанное количество раз.

(SETQ *a* 2)
(REPEAT 5
(SETQ *a* (+ *a* 1))
) – возвращает результат 7.

WHILE – выполнение заданных команд вычисляемое количество раз.

(SETQ *i* 1)
(WHILE (<= *i* 10)
‘ выполнять некоторые команды
) – возвращает результат последнего (десятого) выполнения команд.

NOT – сравнение с условием nil.

(SETQ *a* nil)
(NOT *a*) – возвращает *T*.

5.3.3 Разработка функций в составе AutoCAD

В качестве примера программы, использующей возможности AutoLISP, рассмотрим процесс разработки функции расчета параметров стрелочной улицы под углом ($\alpha_1 + \alpha_2$) (рисунок 5.11).

Рисунок 5.11 – Схема стрелочной улицы под углом ($\alpha_1 + \alpha_2$)

Необходимо определить возможность проектирования данной стрелочной улицы при определенных параметрах α_1, α_2, d, $e_Г$, $b_{пл}$. Другими словами, следует разработать программу расчета стрелочной улицы с обоснованием допустимости укладки стрелочного перевода 2 по схеме взаимного расположения в соответствии с требованиями норм проектирования. Согласно приведенной расчетной схеме (см. рисунок 5.11)

$$T_1 = R\,\mathrm{tg}\frac{\alpha_1}{2},\ e_u = b_{пл} + 2e_Г,\ (b_1 + d + a_2 + b_2 + d_0 + T_1)\sin\alpha_1 = e_u,$$

$$(b_1 + b_2 + d + a_2 + T_1)\sin\alpha_1 + d_0\sin\alpha_1 = b_{пл} + 2e_Г,$$

$$d_0 = \frac{b_{пл} + 2e_Г - (b_1 + b_2 + d + a_2 + T_1)\sin\alpha}{\sin\alpha_1}.$$

Если $d_0 < d$, то данная конструкция стрелочной улицы невозможна.

Соответствующие функции на AutoLISP имеют вид:

```
(defun tan (alfa)
   (/ (sin alfa) (cos alfa))
)
(defun rad (aa)
   (* pi (/ aa 180))
)
(defun NeedleStreet (/ alfa1 alfa2 T1 b_pl e_u e_g R d d0)
(setq s1 (getstring "Марка перевода 1 = 1/9 или 1/11 ")
 s2 (getstring "Марка перевода 2 = 1/9 или 1/11 ")
s3 (getstring "Вид тяги: тепловозная – 1, электровозная – 2")
 R (getint "Радиус кривой=")
 b_pl (getint "Ширина платформы=")
 )
 (if (= s1 "1/9")
 (setq alfa1 6.3 a1 15.76 b1 15.66 d 6.25)
 (setq alfa1 5.2 a1 13.78 b1 19.35 d 12.50)
 )
 (if (= s2 "1/9")
  (setq alfa2 6.3 a2 15.76 b2 15.66 d 6.25)
```

```
20    (setq alfa2 5.2 a2 13.78 b2 19.35 d 12.50)
21  )
22  (if (= s3 "1")
23      (setq e_g 1.745)
24      (setq e_g 1.920)
25  )
26  (setq T1 (* R (tan (rad (/ alfa1 2)))))
27  (setq e_u (+ b_pl (*2 e_g)))
28  (setq d0 (/ (- (+ b_pl (* 2 e_g)) (* ( + b1 b2 d a1 T1)
    (sin (rad alfa1)))) (sin (rad alfa1))))
29  (if (< d0 d)
30      (alert "Уширенное междупутье недостаточно")
31       (progn
32         (setq m (strcat "Прямая вставка d0=" (rtos d0) " м"))
33         (alert m)
34       )
35   )
36  )
```

Основная расчетная функция, обеспечивающая выполнение поставленной задачи, называется NeedleStreet. Функции tan и rad являются вспомогательными, используемыми в основной функции при расчете вставки d_0. Каждая функция – это одна большая запись, в которой количество открытых и закрытых скобок должно быть равным. Следует отметить, что общее число скобок и место их расположения во многом определяют правильность выполнения сложных расчетов аналитических выражений. Поэтому следует по возможности упрощать громоздкие формулы, разбивая их на несколько простых.

Все параметры функции NeedleStreet локальные, т. е. для них только резервируется соответствующая память. Данная функция состоит из трех основных частей. Первая часть (строки 8–13) формирует блок исходных данных, запрашиваемых у проектировщика, которые требуется вводить в командную строку AutoCAD. Вторая часть (строки 14–25) проверяет значения полученных данных и определяет соответствующие величины нормативных параметров (стрелочных переводов, прямых вставок, габаритных расстояний).

Необходимо указать, что приведенная схема анализа данных является наиболее простой. Предполагается, что проектировщик вводит данные строго по указанному формату (1/9, 1/11, 1, 2 и т.д.). При недопустимом вводе (например, вместо марки перевода 1/9 последует ввод 9) требуются более сложные методы программного анализа, корректирующие ввод пользователя или выводящие на экран дисплея информационные сообщения. Третья часть (строки 26–35) выполняет расчет прямой ставки и анализирует полученные результаты.

Как было отмечено ранее, скобки можно расставлять без учета числа пробелов, сообразуясь только с условием наглядности и возможности быстрой проверки правильности их расстановки. Поэтому следует обратить внимание на пары открытых и закрытых скобок, стоящих друг под другом (1 и 3, 4 и 6, 8 и 13, 14 и 17, 18 и 21, 22 и 25, 29 и 35, 31 и 34, 7 и 36).

Исходный код указанных трех функций вводится в любом текстовом имеющемся редакторе (желательно в том, который не дополняет текст управляющими символами, не видимыми на экране дисплея). Наиболее простым и подходящим является «Блокнот».

После набора функций полученный код следует записать в папку AutoCAD (/Support или /Bonus/CadTools) с некоторым именем и расширением *.lsp (например, street.lsp).

5.3.4 Вызов функций через панели инструментов

Для пользования полученной функцией необходимо при загруженном AutoCAD ввести в командную строку

(load “street”).

Результатом правильной загрузки данного файла в командной строке AutoCAD появляется имя последней функции файла (NEEDLESTREET). Если область командных строк ограничена, то просмотреть результаты загрузки файла можно путем нажатия функциональной клавиши <F2>. Далее в командную строку следует ввести имя функции *(needlesstreet)*. При этом функция начинает исполняться в порядке записи команд.

Приведенный прием работы с пользовательскими функциями AutoLISP оказывается сложным при одновременной работе со многими задачами расчета и проектирования. Нужно помнить имена файлов, где записаны необходимые функции, точно воспроизводить имена самих функций, указывать значения вызываемых параметров. AutoCAD позволяет упростить и автоматизировать вызов требуемых функций. Для этого можно занести все вызовы файлов и функций в специальный программный блок и связать его с определенным графическим изображением, похожим на палитру инструментов AutoCAD.

Пользовательскую функцию (needlesstreet) можно добавить в состав типовых средств системы автоматизированного проектирования. Формирование новой графической кнопки, активизация которой будет тождественна последовательному набору команд по вызову данной функции, производится посредством следующих операций:

- в меню AutoCAD «Вид» активизировать опцию «Панель инструментов» (рисунок 5.12);

- нажать кнопку «Новый» и в указанном окне «Имя панели» ввести название «Стрелочная улица». В результате на экране появляется плавающее окно;

- на панели инструментов нажать кнопку «Настроить»;

- справа от выделенного поля нажать на кнопку треугольника и перейти к строке «Выборка»;

- из двух указанных кнопок выбрать первую путем нажатия на ее левой кнопкой мыши и переноса на поле плавающего окна. На данном окне закрепляется пустая кнопка;

- закрыть информационные панели «Настройку панелей» и «Панель инструментов»;

- дважды нажать на пустую кнопку новой панели;

- в открывшемся окне «Свойства кнопки» заполнить поля «Имя» и «Помощь». Количество символов в имени ограничено предельным значением 32. Имя является подсказкой, которая возникает при задержке указателя мыши над данной кнопкой. Помощь может быть более информативной. Введенная в данное поле строка текста появляется в строке статуса (нижняя строка AutoCAD);

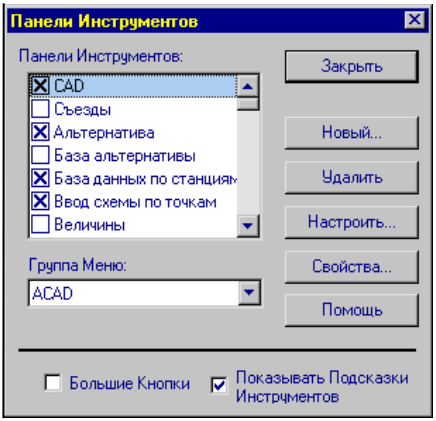


Рисунок 5.12 – Панель инструментов AutoCAD

- в поле макроса после символов ^C^C (без пробелов) ввести

(load "street") (NeedleStreet);

- для формирования внешнего вида иконки кнопки нажать «Редактировать» (рисунок 5.13);

- в «Редакторе кнопок» определить соответствующее изображение;

- нажать кнопку «Сохранить как ...» и в соответствующей папке записать имя картинки с расширением * bmp;

- нажать клавишу «Закрыть» в «Редакторе иконок»;

- нажать клавишу «Применить» в «Свойствах кнопки»;

- закрыть окно «Свойства кнопки»;

- перенести новый графический инструмент САПР в область расположения остальных панелей инструментов AutoCAD.

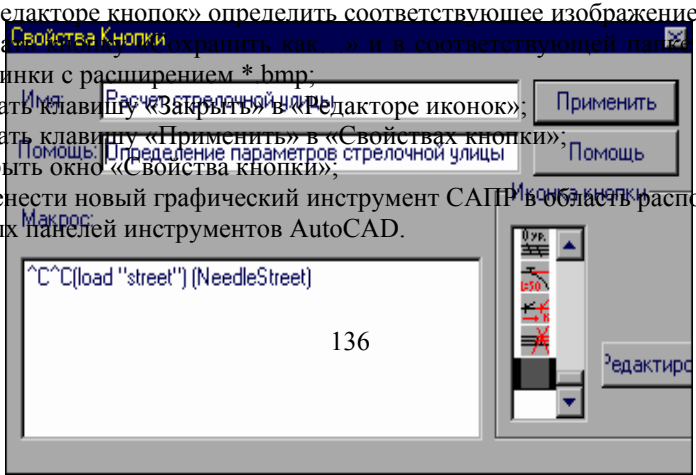


Рисунок 5.13 – Панель «Свойства кнопки»

Разработанный графический инструмент может использоваться для расчета параметров сокращенной улицы. Для этого необходимо нажать на данную кнопку и следовать указаниям программы в командной строке AutoCAD.

Прикладные функции оказываются достаточно удобными для решения целого ряда задач автоматизированного проектирования станций и их элементов. Пример оформления специальной функции определения места положения предельного столбика по заявленному положению шаблона стрелочного перевода рассмотрим на схеме примыкания к задним стыкам крестовины перевода только прямых участков путей (рисунок 5.14).

Координаты предельного столбика определяются по формулам:

$$\Delta b = 2{,}05\cos ec\frac{\alpha}{2} - b,\quad X_4 = X_1 + 2{,}05\cos ec\frac{\alpha}{2}\cos\left(\frac{\alpha}{2} + \beta\right),$$

$$Y_4 = Y_1 + 2{,}05\cos ec\frac{\alpha}{2}\sin\left(\beta + \frac{\alpha}{2}\right),$$

$$\beta = \operatorname{arctg}\frac{Y_3 - Y_1}{X_3 - X_1}.$$

Вид формул инвариантен по отношению к сторонности стрелочного перевода, что дает возможность получить единый формат пользовательской функции определения положения предельного столбика для всех несимметричных шаблонов стрелочных переводов. При этом единственной сложностью является расчет значения угла β, который фиксируется контрольными точками шаблона $A(X_1, Y_1)$ и $B(X_3, Y_3)$. В шаблоне перевода,

который используется САПР, записаны имена выходных точек как атрибуты, координаты которых можно извлечь функцией AutoLISP (entnext).

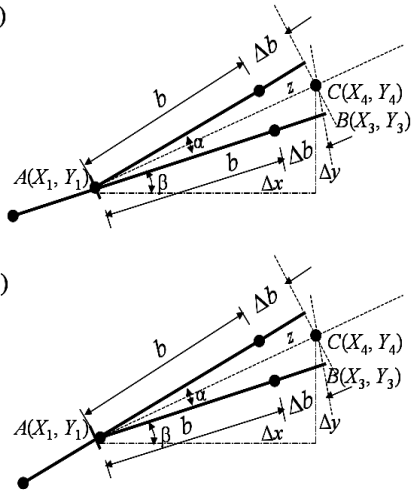


Рисунок 5.14 – Расчетные схемы для определения координат предельного столбика шаблона стрелочного перевода: *а* – левостороннего; *б* – правостороннего

Набор программных строк выбора координат контрольных точек следующий:

```
(setq a22x (entnext a21)                ; первый атрибут
      b22x (entget a22x)                ; выбор записей атрибута
      b221x (nth 7 b22x)                ; координаты контрольной точки 1
      b222x (nth 9 b22x)                ; координаты номера точки
      e222x (cdr b222x)                 : номер точки
      c21x (nth 1 b221x)                ; координата X контрольной точки 1
      c21y (nth 2 b221x))               ; координата У контрольной точки 1
(setq a23x (entnext a22x)               ; второй атрибут
      b23x (entget a23x)                ; выбор записей атрибута
      b231x (nth 7 b23x)                ; координаты контрольной точки 2
      b232x (nth 9 b23x)                ; координаты номера точки
      e232x (cdr b232x)                 ; номер точки
      c22x (nth 1 b231x)                ; координата X контрольной точки 2
      c22y (nth 2 b231x))               ; координата У контрольной точки 2
(setq a24x (entnext a23x)               ; третий атрибут
      b24x (entget a24x)                ; выбор записей атрибута
                                        ; координаты контрольной точки 3
                                        ; координаты номера точки
                                        ; номер точки
                                        ; координата X контрольной точки 3
                                        ; координата У контрольной точки 3
```

b241x (nth 7 b24x)
b242x (nth 9 b24x)
e242x (cdr b242x)
c23x (nth 1 b241x)
c23y (nth 2 b241x))

Следует отметить, что положение точки *B* на рисунке 5.14, *а* соответствует контрольной точке 2, а на рисунке 5.14, *б* – контрольной точке 3. Таким образом, в переменных c22x, c22y сохраняются координаты X_3, Y_3 для левосторонних, а в c23x, c23y – правосторонних стрелочных переводов.

По предложенной выше схеме (см. рисунки 5.12, 5.13) можно разработать пользовательскую функцию, которая на экране по указанию оператора рассчитывала бы координаты предельного столбика и фиксировала его положение на рабочем поле.

Задачи

Задача 5.1 Разработать программную функцию установки тангенсов, определяющих начало и конец кривой, с помощью средств Visual Basic.

Задача 5.2 Решить задачу 5.1, используя возможности AutoLISP. Сравнить затраты на данные варианты решения и эффективность работы полученных программных продуктов.

Задача 5.3 Найти различия в свойствах и возможностях панелей элементов среды Visual Basic и VBA AutoCAD.

Задача 5.4 Разработать функцию расчета параллельного смещения с неизвестными параметрами M_2 и L_1 (см. рисунок 4.2) с вызовом в VBA AutoCAD опции Module (см. рисунок 5.8).

Задача 5.5 По результатам решения предыдущей задачи запроектировать в AutoCAD средствами VBA полученное параллельное смещение пути.

Задача 5.6 Разработать новое программное решение по интерфейсу рисунка 5.10, в котором углы отсчета заменяются координатами начала и конца кривой.

Примечание − Координаты начала и конца кривой следует аналитически получить по известным параметрам радиуса, длины и центра кривой.

Задача 5.7 С помощью команд AutoCAD и AutoLISP рассчитать и построить сопряжение двух прямых участков пути кривой заданного радиуса.

Задача 5.8 Разработать на AutoLISP программу расчета параметров смещения пути, не выражаемых их уравнений (4.1) аналитически (например, определить любые три неизвестные параметра).

Задача 5.9 Средствами AutoCAD разработать шаблоны (блоки) стрелочных переводов и в дальнейшем с помощью программы на AutoLISP

использовать полученные шаблоны для соединения стрелочных переводов между собой по соответствующим схемам.

Задача 5.10 Разработать интерфейс совместного использования VBA и AutoCAD в программе, изменяющей значения параметров по движению полосы прокрутки и визуализации полученных результатов на рабочем поле AutoCAD.

Тесты

Тест 5.1 Окно командных строк AutoCAD используется:
1) для активизации VBA;
2) визуализации панелей элементов Visual Basic;
3) ввода команд AutoLISP;
4) разработки пользовательского программного меню.

Тест 5.2 Инструмент проектирования PLINE (полилиния) позволяет:
1) вычерчивать длинные кривые участки;
2) укладывать сопрягаемые между собой прямые и кривые;
3) соединять прямые и кривые без сопряжения;
4) сопрягать существующие прямые и кривые.

Тест 5.3 Наиболее быстрый способ масштабирования в AutoCAD:
1) нажатие знака «+» на клавиатуре;
2) ввод команды SCALE;
3) вращение колеса на манипуляторе мыши;
4) нажатие правой кнопки мыши.

Тест 5.4 Нельзя вызвать:
1) VBA из AutoCAD;
2) AutoLISP из VB;
3) AutoCAD из VB;
4) AutoLISP из AutoCAD.

Тест 5.5 Вызов VBA из AutoCAD осуществляется посредством нажатия:
1) STRL + F11;
2) ESC + F11;
3) Space + F11;
4) ALT + F11.

Тест 5.6 Правильный вызов из VBA метода построения криволинейного пути:
1) ***Set ArcObj*** = *ThisDrawing.ModelSpace.**AddCircle**(CenterPoint, Radius, StartAngle, EndAngle);*
2) ***Set ArcObj*** = *ThisDrawing.ModelSpace.**AddArc**(CenterPoint, Radius, StartAngle, EndAngle);*

3) ***Set ArcObj*** = *ThisDrawing.ModelSpace.**AddLine**(CenterPoint, Radius, StartAngle, EndAngle);*

4) ***Set ArcObj*** = *ThisDrawing.**AddArc**(CenterPoint, Radius, StartAngle, EndAngle).*

Тест 5.7 AutoLISP использует:
1) синтаксис Visual Basic;
2) элементы Visual Basic;
3) списки;
4) некоторые функции VBA.

Тест 5.8 Результат вычисления выражения (*(+5(+6 4))(-8(-4 1))) равен:
1) 60;
2) 52;
3) 85;
4) 75.

Тест 5.9 Результат вычисления выражений
(setq a 2 b 3 c 4)
(print (*(+a b)(-c b))
равен:
1) 4;
2) -5;
3) 5;
4) 0.

Тест 5.10 Окно загрузки команд AutoCAD вызывается нажатием клавиши:
1) F1;
2) F2;
3) F3;
4) F11.

Контрольные вопросы

1 Чем отличается объект путевого развития от графического примитива AutoCAD?

2 Какой способ решения расчетно-проектных задач предпочтительнее: вызов AutoCAD из программы Visual Basic или вызов VBA из AutoCAD?

3 Почему возникает необходимость масштабирования изображений в AutoCAD?

4 Чем отличается команда LINE от PLINE?

5 Что собой представляют строки кода на AutoLISP?

6 Почему рекомендуется все команды AutoCAD в программах AutoLISP записывать как _. + «команда»?

7 Как удалить с рабочего поля AutoCAD экранное меню?

8 Как увеличить поле окна командных строк AutoCAD?

9 Что называется рабочим столом AutoCAD?

10 Как реализуется диалог программной среды и проектировщика в программе с использованием AutoCAD и VBA?

СПИСОК ИСПОЛЬЗОВАННОЙ И РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1 Железнодорожные станции и узлы (задачи, примеры, расчеты) / под общ. ред. Н. В. Правдина. – М. : Маршрут, 2007. – 455 с.

2 **Ахраменко, Г. В.** Проектирование плана и земляного полотна второго пути / Г. В. Ахраменко, В. А. Вербило, Н. В. Довгелюк. – Гомель : БелГУТ, 2009. – 64 с.

3 **Ахраменко, Г. В.** Расчеты изменения междупутий при проектировании второго пути : учеб.-метод. пособие по курсовому и дипломному проектированию / Г. В. Ахраменко. – Гомель : БелГУТ, 2009. – 67 с.

4 **Довгелюк, Н. В.** Тяговые расчеты при проектировании железных дорог / Н. В. Довгелюк, В. А. Вербило, Г. В. Ахраменко. – Гомель : БелГУТ, 2009. – 74 с.

5 **Браун, С.** Visual Basic. Учебный курс / С. Браун. – СПб. : Питер, 2000. – 576 с.

6 Сайлер, Б. Использование Visual Basic 6 /Б. Сайлер, Д. Споттс. – М. : Издательский дом «Вильямс», 2005. – 832 с.

7 Visual Basic на практике / под ред. Г. И. Магданурова. – СПБ. : БХВ-Петербург, 2008. – 480 с.

8 **Патрик, Т.** Visual Basic 2005. Рецепты программирования / Т. Патрик, К. Крейг. – СПБ. : БХВ-Петербург, 2008. – 752 с.

9 **Тику, Ш.** AutoCAD 2005 / Ш. Тику. – СПб. : Питер, 2005. – 1088 с.

10 **Полещук, Н.** AutoCAD: Разработка приложений. Настройка и адаптация / Н. Полещук. – СПб. : БХВ-Петербург, 2006. – С. 457–503.

11 **Зуев, С.** САПР на базе AutoCAD – как это делается / С. Зуев, Н. Полещук. – СПб. : БХВ-Петербург, 2004. – 1168 с.

12 **Свет, В.** AutoCAD: язык макрокоманд и создание кнопок / В. Свет. – СПб. : БХВ-Петербург, 2004. – 320 с.

13 **Кудрявцев, Е.** AutoLISP: Программирование в AutoCAD / Е. Кудрявцев. – М. : ДМК, 1999. – 368 с.

14 **Полещук, Н.** Visual LISP и секреты адаптации AutoCAD / Н. Полещук. – СПб. : БХВ-Петербург, 2001. – 576 с.

15 Приемы объектно-ориентированного проектирования / Э. Гамма [и др.] – СПб. : Питер, 2007. – 366 с.

16 **Басс, Л.** Архитектура программного обеспечения / Л. Басс, П. Клеменс, Р. Кацман. – СПб. : Питер, 2006 . – 575 с.

17 **Тецци, К.** Основы инженерии программного обеспечения / К. Тецци, М. Джазайери, Д. Мандриоли. – СПб. : БХВ-Петербург, 2005. – 832 с.

ПРИЛОЖЕНИЕ А
(справочное)

РАБОЧАЯ ПРОГРАММА ПО ДИСЦИПЛИНЕ «АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ ДОРОГ»

1 ЦЕЛЬ И ЗАДАЧИ ДИСЦИПЛИНЫ В УЧЕБНОМ ПРОЦЕССЕ

1.1 Цель преподавания дисциплины

Дисциплина «Автоматизированное проектирование дорог» является одной из новых дисциплин, утвержденных согласно учебному плану и предлагаемых к изучению студентам, обучающимся на строительном факультете.

Предметом изучения курса являются теоретические и практические основы автоматизированных расчетов элементов дорог и их проектирование с помощью программного обеспечения общего и специального назначения персональных ЭВМ. Данная дисциплина базируется на активном применении математических и вычислительных методов. При этом изучению подлежат как системы автоматизированного проектирования, обеспечивающие сокращение продолжительности проектирования за счет машинных расчетов, подготовки рабочих чертежей, так и автоматизированные системы проектирования дорог, способствующие снижению временных затрат на предварительных этапах проектирования (автоматизация обработки данных аэрогеодезических съемок участка прокладки дороги, использование программ определения объемов земляных работ, расчет кривых участков пути и т. д.).

Цель преподавания дисциплины − изучение теоретических принципов автоматизации расчетов и проектирования элементов дорог, получение практических навыков использования ЭВМ для выполнения расчетов и проектирования.

1.2 Задачи изучения дисциплины

Основные задачи курса «Автоматизированное проектирование дорог»:

1 Ознакомление с существующими информационно-техническими средствами и устройствами общего специального назначения, обеспечивающими автоматизацию проектирования транспортных объектов.

2 Характеристика и применение существующих пакетов прикладных программ автоматизации расчетов.

3 Анализ используемых в пакетах прикладных программ математических и вычислительных методов, обеспечивающих эффективные и оптимизирующие проектные решения.

4 Рассмотрение перспективных аспектов дальнейшего совершенствования методик расчета элементов дорог и принципов возможного построения соответствующих алгоритмов и формализованных процедур.

5 Анализ существующих методик определения экономической эффективности САПР.

6 Оценка качества проектной работы, выполненной с помощью прикладных пакетов автоматизированного проектирования (продолжительность проектирования, производительность труда проектировщиков, выдача проектной документации).

1.3 Перечень дисциплин, усвоение которых необходимо для изучения дисциплины

1 Изыскания и проектирование железных дорог.

2 Математика (матричное исследование, линейная алгебра, теория графов, теория вероятностей, математическая статистика, теория принятия решений).

3 Информатика (структурное программирование, архитектура ПЭВМ и периферия).

Для изучения данной дисциплины в течение VIII семестра предусматривается 48 аудиторных часов, из них 34 часа − лекции, 14 часов − практические занятия. В конце семестра зачет.

2 СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

2.1 Содержание лекционных занятий

Принципы проектирования и структура автоматизированных систем проектирования – 6 ч.

Назначение САПР. Систематические и эвристические САПР. Классификация САПР. Основные принципы функционирования САПР. Структура задач САПР. Особенности САПР Д.

Базовое обеспечение САПР ЖД – 6 ч.

Информационное обеспечение. Математическое обеспечение. Программное обеспечение. Техническое обеспечение. Лингвистическое обеспечение. Методическое обеспечение. Организационное обеспечение.

Применение пакетов прикладных программ для автоматизированных расчетов и проектирования железных дорог – 8 ч.

Возможности САПР общего назначения. Использование AutoCAD. Программные надстройки над типовыми САПР. Сопряжение результатов работы нескольких программных сред. Передача параметров выполненных расчетов в проектную среду автоматизации.

Взаимодействие проектировщика (технического дизайнера проекта) и программной среды САПР – 6 ч.

Роль дизайнера проекта в общем процессе разработки проектного решения. Мониторинг технического дизайнера контрольных точек процесса автоматизированной подготовки проекта. Вариантные решения и летопись проекта в САПР железных дорог. Перспективы разработки и применения информационно-управляющих систем в автоматизированной подготовке проектов по проектированию и реконструкции железных дорог.

Комплексные решения расчетных и проектных задач по проектированию и реконструкции железных дорог с использованием сопряженных сред – 6 ч.

Анализ возможностей пакетов Visual Basic и AutoCAD по сопряжению результатов расчета и проектирования. Алгоритмизация и программирование задач в среде VB. Схема передачи результатов расчета из программы на VB в AutoCAD. Получение проектного решения в AutoCAD.

Технико-экономические расчеты эффективности внедрения существующих САПР – 4 ч.

Понятие эффективности как критерия внедрения САПР. Анализ существующих методик определения экономической эффективности САПР.

2.2 Перечень практических занятий (14 ч)

1 Алгоритмизация задачи определения объема земляных работ поперечных профилей земляного полотна по средним рабочим отметкам – 4 ч.

2 Автоматизация тяговых расчетов – 4 ч.

3 Программный расчет параметров путепроводной развязки – 2 ч..

4 Программный расчет площади угловой диаграммы проектируемой кривой – 2 ч.

5 Программный расчет смещения круговой кривой при увеличении радиуса – 2 ч.

3 ОСНОВНАЯ ЛИТЕРАТУРА

1 Изыскания и проектирование железных дорог : учеб. для вузов ж.-д. трансп. / И. В. Турбин [и др.] ; под ред. И. В. Турбина. − М. : Транспорт, 1989. – 479 с.

2 **Кантор, И. И.** Изыскания и проектирование железных дорог / И. И. Кантор. – М. : ИКЦ «Академкнига», 2003. − 288 с.

3 Лабораторный практикум на базе учебно-исследовательской САПР : практ. пособие / под ред. А. В. Петрова. − М. : Высш. шк., 1991. – 78 с.

4 **Головнич, А. К.** Автоматизация проектирования транспортных коммуникаций / А. К. Головнич. – Гомель : БелГУТ, 2002. − 52 с.

5 **Головнич, А. К**. Применение элементов САПР в расчетах транспортной развязки. − Гомель: БелГУТ, 2002. − 167 с.

6 **Бучкин, В. А.** Автоматизированное проектирование реконструкции плана и продольного профиля железных дорог : учеб. пособие / В. А. Бучкин, Г. Г. Иванов, В. С. Ми-ронов. − М., 1994. − 70 с.

7 **Бучкин, В. А.** Автоматизированное проектирование продольного профиля железных дорог : учеб. пособие / В. А. Бучкин, Г. Г. Иванов. − М., 1994. − 43 с.

8 **Першин, С. П.** Автоматизированное проектирование организации строительства железных дорог / С. П. Першин, М. И. Иванов, А. Ф. Акуратов. − М. : Транспорт, 1991. − 261 с.

9 Технология и автоматизация железнодорожных изысканий : учеб. пособие / В. С. Миронов [и др.] ; под общ. ред. В. С. Миронов. − М. : МИИТ, 1994. − 88 с.

10 **Головнич, А. К**. Автоматизация проектирования железнодорожных станций и узлов: уч.-метод. пособие. Ч.1 / А. К. Головнич. – Гомель : БелГУТ, 2006. − 100 с.

11 **Правдин, Н. В.** Компьютерное проектирование железнодорожных станций / Н. В. Правдин, А. К. Головнич, С. П. Вакуленко. − М. : ГОУ УМЦ, 2008. − 472 с.

ОГЛАВЛЕНИЕ

Учебное издание

ГОЛОВНИЧ Александр Константинович

АВТОМАТИЗАЦИЯ РАСЧЕТОВ И ПРОЕКТИРОВАНИЯ ЖЕЛЕЗНЫХ ДОРОГ

Учебно-методическое пособие

Редактор Т. М. Р и з е в с к а я
Технический редактор В. Н. К у ч е р о в а

Подписано в печать 05.07.2010 г. Формат 60х84 1/16
Бумага офсетная. Гарнитура Times. Печать на ризографе.
Усл. печ. л. 8,14. Уч.-изд. л. 9,01. Тираж 150 экз.
Зак. № . Изд. № 36.

Издатель и полиграфическое исполнение
Белорусский государственный университет транспорта:

ЛИ № 02330/0552508 от 09.07.2009 г.
ЛП № 02330/0494150 от 03.04.2009 г.
246653, г. Гомель, ул. Кирова, 34