View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Electronic National Technical University "Kharkiv...

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ХАРЬКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам
«Элементы управления среды Visual Basic.
Ввод и вывод информации»
из раздела «Программирование в среде Visual Basic.
Интегрированная среда разработки»
дисциплины «Информатика»
для студентов направления подготовки
6.050801 «Микро- и наноэлектроника»

Харьков НТУ «ХПИ» 2013

Методические указания к лабораторным работам «Элементы управления среды Visual Basic. Ввод и вывод информации» из раздела «Программирование в среде Visual Basic. Интегрированная среда разработки» дисциплины «Информатика» для студентов направления подготовки 6.050801 «Микро- и наноэлектроника» / Состав.: В.И. Шкалето, Р.В. Зайцев, Г.С. Хрипунов. – Харьков: НТУ «ХПИ», 2013. – 59 с.

Составители: В.И. Шкалето,
Р.В. Зайцев
Г.С. Хрипунов

Рецензент доц. Г.И. Копач

Кафедра физического материаловедения для электроники и гелиоэнергетики

ВВЕДЕНИЕ

Методические указания к лабораторным работам по разделу «Программирование в среде Visual Basic. Интегрированная среда разработки» дисциплины «Информатика» касаются трех лабораторних работ: «Основные элементы управления в Visual Basic», «Элементы управления Windows в Visual Basic. Массивы элементов управления» и «Работа с графикой в Visual Basic. Экран, клавиатура и мышь».

Так что же такое Visual Basic? Слово "Visual" относится к методу, используемому для создания того, что видит пользователь — графического пользовательского интерфейса, или GUI (graphical user interface). Слово "Basic" относится к аббревиатуре BASIC (Beginners All-Purpose Symbolic Instruction Code — многоцелевой код символьных инструкций для начинающих) языка программирования, который используется программистами намного чаще, чем любой другой язык в истории вычислений. Для создания различных полезных программ достаточно изучить лишь некоторые из его возможностей. Приведенные ниже лабораторные работы содержат материалы для начинающих программистов в среде Visual Basic 6.0 (VB6.0), в каждой из которых изучаются определенные вопросы технологии работы со средой программирования VB6.0 и решаются различные варианты учебной задачи.

В результате изучения теоретического материала и выполнения лабораторных работ студент должен изучить:

- этапы разработки программ;
- основы алгоритмического языка VB6.0;
- назначение, функции и структуру среды программирования VB6.0;

а также уметь:

- проводить физический и математический анализ простейших прикладных задач и разрабатывать алгоритмы их решения;
- вести отладку программ в среде программирования VB6.0.

Указанное позволит преорести первоначальные навыки работы в среде программирования VB6.0

ЛАБОРАТОРНАЯ РАБОТА 1

ОСНОВНЫЕ ЭЛЕМЕНТЫ УПРАВЛЕНИЯ В VISUAL BASIC

Цель работы – Научиться работать с различными элементами управления в Visual Basic 6. Создать проект на Visual Basic 6, содержащий формы, различные элементы управления, процедуры и функции. Научиться работать с графическими и математическими данными

1.1 Общие сведения

1.1.1 Общие сведения об элементах управления

1.1.1.1 Использование элементов управления

Создание Windows приложений в Visual Basic практически невозможно без использования элементов управления, так как они позволяют пользователю взаимодействовать с этими приложениями. Набор таких элементов управления не ограничен и может расширяться за счет так называемых пользовательских элементов управления (Custom Controls).

С элементами управления вы уже сталкивались при выполнении предыдущих лабораторных работ (разработке проектов). Однако Visual Basic позволяет обращаться к элементам управления не только при разработке приложения, но и во время выполнения программы.

Главное, что следует знать при работе с элемент ми управления, – то, что к ним можно обращаться как к переменной, присваивая значения определенным свойствам или считывая их.

Свойства определяют внешний вид и функционирование элемента управления. Например, если требуется установить новую надпись, то следует изменить свойство Caption. Как это делается на стадии проектирования, вам уже известно. Но как изменять свойства во время выполнения приложения? Это несложно, если рассматривать элементы управления как переменные – так, для изменения надписи кнопки Command1 используется ее свойство Caption:

```
Command1.Caption = "Новая надпись"
```

В данном примере свойству Caption объекта Command1 присваивается значение *Новая надпись*. При этом имя объекта и свойство разделяются точкой:

```
Control.Свойство = Значение
```

Проект обычно состоит из нескольких форм, каждая из которых может содержать элементы управления с одинаковым именем. Поэтому синтаксис обращения к свойствам следует несколько расширить:

```
[Форма.]Control.Свойство = Значение
```

Имя формы указывать не обязательно, если обращаются к элементу управления, принадлежащему этой форме.

Значения свойств элементов управления считываются аналогичным образом. Каждое свойство является как бы внутренней переменной элемента управления, значение которой можно не только установить, но и считать. Поэтому, чтобы узнать, например, текст на командной кнопке, достаточно записать:

```
Надпись$ = Command1.Caption
```

В этом примере переменная Надпись$ после присваивания содержит текст надписи на командной кнопке. В общем случае значение свойства считывается следующим образом:

```
Значение =[Форма.]Объект.Свойство
```

Большинство свойств элементов управления доступно как для считывания, так и для изменения. Но есть свойства, которые во время выполнения доступны только для чтения (Read Only); другие же могут быть недоступны при проектировании. Сведения о доступности свойств (для чтения или для изменения, при проектировании или во время выполнения) содержатся в справочном файле Visual Basic.

Запомнить все свойства всех элементов управления практически невозможно. Для получения информации о каком-либо элементе управления, его свойствах, методах и событиях следует обратиться к справке. Для этого выделите соответствующий элемент управления на панели элементов и нажмите клавишу [F1]. После этого Visual Basic предоставит всю необходимую информацию.

1.1.1.2 Основные свойства элементов управления

Ниже рассматриваются свойства, которыми обладает большинство элементов управления. Позицию элемента управления определяют четыре свойства: Left, Top, Height и Width. Эти значения по умолчанию используют в качестве единицы измерения твип (twip – это экранно-независимая единица измерения, равная 1/20 точки принтера и гарантирующая независимость отображения элементов приложения от разрешения дисплея).

Свойства Top и Left задают координаты верхнего левого угла элемента управления, свойства Height и Width – его высоту и ширину. Стандартный отсчет в системе координат экрана и других элементов управления ведется сверху вниз (Y) и слева направо (X).

Управление цветовым оформлением элементов осуществляется с помощью свойств BackColor, FillColor и ForeColor, которым по умолчанию назначаются стандартные цвета Windows. Цвет фона устанавливается с помощью свойства BackColor. При проектировании цвет выбирают в диалоговом окне настройки цвета, а во время работы приложения цвета задаются либо с использованием цветовой схемы RGB, либо цветовой функцией QBColor (функция из прежней версии языка QBasic), либо константами библиотеки VBRUN. С помощью свойств ForeColor можно определить или установить цвет, используемый для отображения текста и графики в элементе управления, с помощью свойства FillColor – установить цвет заполнения так называемых Shapes (рисованных объектов).

Вид шрифта в элементах управления выбирается путем установки значений свойства Font (табл. 1.1).

Таблица 1.1 - Параметры шрифта

Свойство	Значение
Font.Name	Имя шрифта
Font.Size	Размер шрифта
Font.Bold	Полужирный
Font.Italic	Курсив
Font.Underline	Подчеркивание
Font.StrikeThrough	Перечеркивание
Font.Weight	Толщина символа

Часто при работе приложения требуется сделать недоступными для пользователя некоторые элементы управления. Для этого используют два свойства – Enabled и Visible. Свойство Enabled определяет, будет ли элемент управления реагировать на событие или нет. Если значение свойств равно False, элемент управления будет недоступен и пользователь не сможет его использовать. Обычно при этом элемент подсвечивается серым цветом, так же, как элементы меню, которые нельзя выбрать. Свойство Visible позволяет сделать элемент управления невидимым. Если его значение равно False, то он не виден, и обратиться к нему нельзя.

Выбор свойства Visible либо свойства Enabled остается за вами. Если вы выбираете свойство Enabled, это означает, что элемент управления есть, но обратиться к нему пока невозможно. А свойство Visible позволяет "скрыть" элемент от пользователя.

```
Private Sub Cominand1_Click ()
Command1.Enabled = False
End Sub
Private Sub Command2_Click()
Command2.Visible = False
End Sub
```

Свойство Name играет особую роль. Ошибки при его задании часто приводят к серьезным последствиям. Имя является идентификатором элемента управления. Если в приведенном выше примере изменить имя второй кнопки на Comniand2, то код больше не будет выполняться, так как элемента с именем Comniand2 нет. Окажется невозможным и изменить свойства кнопки Command2. Поэтому сначала всегда следует задавать имя элемента управления и лишь, затем писать для него код обработки его события.

Большинство элементов управления имеет свойство Appearance, отвечающее за отображение элемента управления (без визуальных эффектов или в трехмерном виде).

Кроме того, для большинства элементов управления можно установить значение свойства ToolTipText. Введенный текст отображается в

подсказке, которая появляется, если пользователь установит указатель мыши на элементе управления в форме.

Большая часть элементов управления Visual Basic имеет также свойства Parent и Container. Свойство Parent указывает на родительский объект. Благодаря этому свойству возможен доступ к его методам или свойствам. В следующем примере строка заголовка формы, которой принадлежит соответствующая кнопка, сохраняется в переменной strCaption:

```
strCaption$ = Command1.Parent.Caption
```

Свойство Container, на первый взгляд, действует аналогично. Однако в отличие от свойства Parent, доступного только для чтения, свойство Container позволяет не только считывать, но и изменять контейнер элемент управления. В примере путем изменения свойства Container кнопка перемещается в элемент pictureBox:

```
Set Command1.Contanier = Picture1
```

Следующие элементы управления могут служить контейнером для других элементов управления: Form, Frame, Picture, Toolbar.

В отличие от других свойств, Visual Basic не использует свойство Tag для управления элементом управления – это свойство предназначено для хранения любых дополнительных данных, необходимых разработчику:

```
Text.Tag ="Ввод имени по-русски"
```

1.1.1.3 Основные события

В этом разделе рассматриваются события, которые могут обрабатываться большинством элементов управления.

Есть два события, вызываемые щелчком мыши:Click и DblClick.

Событие Click вызывается, как только пользователь выполнит щелчок на элементе управления. Событие DblClick вызывается двойным щелчком кнопкой мыши на элементе управления. Временной интервал между двумя щелчками двойного щелчка устанавливается в панели управления Windows. Параметры для процедур обработки этих событий не передаются. Кроме основных событий Click и DblClick, есть еще три.

Событие MouseDown вызывается при нажатии кнопки мыши. При этом процедуре обработки события передается несколько параметров:

```
Private Sub Command1_MouseDown(Button As Integer,_
        Shift As Integer, X As Single, Y As Single)
End Sub
```

Передаваемые параметры (табл. 1.2) определяют состояние кнопок мыши (Button), управляющих клавиш (Shift) и позицию курсора (X и Y).Параметры X и Y определяют позицию курсора мыши на экране относительно левой верхней точки элемента управления.

Событие MouseUp вызывается при отпускании кнопки мыши. Его параметры аналогичны и приведены в табл. 1.2.

Событие MouseMove вызывается, когда пользователь передвигает курсор мыши. Синтаксис процедуры обработки этого события следующий:

```
Private Sub Control_MouseMove (Button As Integer,_
        Shift As Integer, X As Single, Y As Single)
End Sub
```

Таблица 1.2 - Параметры событий MouseUp и MouseDown

Парметр	Значение
Button	Нажата кнопка мыши: 1 = левая, 2 = правая, 4 = средняя
Shift	Нажата клавиша: 0 = ничего, l = [Shift], 2 = [Ctrl], 4 = [Alt], а также их комбинации
X	Координата X
Y	Координата Y

Подобно событиям, связанным с мышью, есть также события, связанные с клавиатурой: KeyPress, KeyUp и KeyDown. Обычно событие вызывается для активного элемента управления. Если свойству формы KeyPreview присвоить значение True, то событие, связанное с клавиатурой, передается сначала форме, затем текущему элементу управления.

Событие KeyPress возвращает код ASCII нажатой клавиши. При этом не перехватываются специальные клавиши, такие как [PrintScreen] или [Alt],только [Enter], [Esc] и [Backspace]. Процедура передает параметр KeyASCII, содержащий код ASCII нажатой клавиши. Этот параметр передается как значение, т.е. его можно изменять. Это можно использовать, например, для фильтрации вводимых пользователем символов – если символ недопустимый, то установив значение KeyASCII равным нулю, вы предотвратите его передачу для дальнейшей обработки (отображение и т.п.).

```
Control_KeyPress (KeyAscii As Integer)
```

События KeyDown, KeyUp вызываются при нажатии (KeyDown) или отпускании (KeyUp) клавиши. Они происходят даже при нажатии специальных клавиш управления, например функциональных клавиш. При этом передаются два параметра: KeyCode и Shift. Параметр KeyCode содержит клавиатурный код (а не ASCII) нажатой клавиши, например vbKeyFl, а параметр Shift информирует о состоянии клавиш [Shift],[Ctrl] и [Alt].

```
Control_ KeyUp(Key Code As Integer, Shift As Integer)
Control_ KeyDown (KeyCode As Integer, Shift As Integer)
```

После нажатия клавиши события наступают в такой последовательности: KeyDown, KeyPress и KeyUp.

1.1.1.4 Фокус

Фокус – это одно из важных понятий при обращении к элементам управления в Windows. Как уже упоминалось, система Windows решает, какому приложению передавать нажатие клавиши – управление получает активный элемент, т.е. элемент, имеющий фокус. Если элемент получает фокус, то это соответствующим образом отображается на экране – тексто-

вое поле отображается с мерцающим маркером ввода, командная кнопка выделяется пунктирной рамкой вокруг надписи.

Visual Basic позволяет обрабатывать два события, связанных с передачей фокуса: LostFocus и GotFocus. Если перейти от одного элемента управления к другому, то для предыдущего элемента вызывается событие LostFocus, а для нового – GotFocus.

1.1.2 Форма

Форма дает возможность сообщаться с "внешним миром", т.е. с пользователем, и поэтому стоит познакомиться с ней поближе. Как уже говорилось, каждая форма сохраняется в проекте в виде отдельного файла. Этот файл содержит описание рабочей среды и код, относящийся к элементам управления и форме. Формы сохраняются как обычные текстовые файлы.

Кроме стандартных свойств, таких как Caption, BackColor, Font и т.д., формы имеют и свои собственные свойств, присущие только им. Эти свойств рассматриваются ниже. Для просмотра свойств формы в окне свойств нужно либо щелкнуть в пустом месте формы (но не в строке заголовка), либо выбрать форму из списка объектов в окне свойств.

Стандартное окно имеет рамку (border).С ее помощью пользователь может изменять размеры окна. Для этого в системном меню имеется соответствующая команда. Вид рамки можно изменить с помощью свойства BorderStyle, которое может принимать одно из следующих значений (табл. 1.3). С помощью рамки можно изменять не только внешний вид окна, но и его размеры. А это важно, т.к. как содержимое окна не подгоняется автоматически к его измененному размеру. Это может привести к тому, что элемент управления после изменения размера будет находиться вне видимой области и поэтому "забудется". Не включится также и полоса прокрутки.

Свойство ControlBox определяет, отображается ли системное меню, с помощью которого пользователь может выйти из программы ([Alt+F4]). Если же системное меню удаляется, пользователю следует обеспечить другой способ выхода из программы.

Кнопкой максимизации пользователь может увеличить окно до размера экрана. Ее наличие определяется свойством MaxButton формы. Если присвоить этому свойству значение False, то соответствующая кнопка будет отсутствовать, а команда Maximize (Развернуть) удаляется из системного меню.

Если для свойства MinButton задать значение False, то кнопка затемняется, из системного меню удаляется строка Minimize (Свернуть).

С некоторыми из событий формы Вы уже знакомы. Но есть еще несколько событий, на которые необходимо обратить особое внимание. Следует отметить одну особенность событий формы – синтаксис проце-

дуры обработки события формы отличается от синтаксис процедур обработки событий элементов управления.

```
Form_Событие([Аргументы...])
```

Таблица 1.3 - Установка рамки окна

Константа	Значение	Вид окна
VbBSNone	0	Окно без рамки. Размер окна изменяться не может. Нет строки заголовка. Окно не может перемещаться. Минимизация и максимизация окна также невозможны.
VbFixedSingle	1	Фиксированная рамка. Есть заголовок, кнопки минимизации и максимизации, но размер окна изменяться не может.
VbSizable	2	Значение по умолчанию. Возможны все изменения размера окна.
VbFixedDialog	3	Окно окаймляется толстой рамкой. Изменения размера невозможны. Нет кнопок минимизации и максимизации. Минимизировать и максимизировать можно только из системного меню.
VbFixedToolWindow	4	Поведение такое же, как vbFixedSingle.
VbSizableToolWindow	5	Поведение такое же, как VbSizable, но строка заголовка более узкая и имеет меньший шрифт.

Имя процедуры обработки события формы всегда содержат Form. При этом не важно, как фактически называется форма.

Одним из наиболее используемых событий формы является Load. Это событие происходит при загрузке формы в память. Поэтому Load лучше всего подходит для инициализации объектов и переменных, принадлежащих форме. Событие Unload вызывается, если форма удаляется из памяти. С помощью параметра Cancel можно отменить удаление формы с экрана. Более практичным, чем Unload, является событие QueryUnload. Наряду с параметром Cancel в процедуру обработки события передается и параметр UnloadMode, указывающий причину возникновения события (табл. 1.4).

Проблема, возникающая при изменении размеров формы, уже рассматривалась. Решить ее помогает событие Resize, которое вызывается при любом изменении размеров формы. При этом следует учитывать два аспекта. Во-первых, во время обработки события Load форма еще не видна. Во-вторых, при загрузке и отображении формы всегда возникает и событие Resize, так как при запуске размеры формы изменяются от нулевых до заданных в свойствах формы. При создании процедур для связанных событий типа Activate, GotFocus, Paint и Resize убедитесь, что их дей-

ствия не находятся в противоречии друг с другом и что они не вызывают рекурсивных событий.

Таблица 1.4 - Параметры UnloadMode в QueryUnload

Константа	Значение	Причина возникновения события
VbFormControlMenu	0	Пользователь закрыл приложение посредством [Alt+F4], кнопки Close (Закрыть) окна или одноименной команды системного меню
VbFormCode	1	В коде выполняется команда Unload
VbAppWindows	2	Завершение сеанса Windows
VbAppTaskManager	3	Выход из приложения с помощью менеджера задач
VbFormMDIForm	4	Дочерняя форма MDI закрыта, так как закрыта вышестоящая форма MDI

1.1.3 Основные элементы управления

1.1.3.1 Кнопка (CommandButton)

Этот элемент управления используется для того, чтобы начать, прервать или закончить какой-либо процесс. Кнопка встречается практически во всех приложениях Windows.

Главным событием для кнопки является Click. Кроме этого события, у кнопки могут быть и другие, но они применяются редко. Для вызова события Click имеются разные способы. Самый простой – непосредственный щелчок на кнопке мышью. Это же событие вызывается также, если с помощью клавиши [Tab] переместить фокус на кнопку, затем нажать клавишу [Enter]. Можно программно вызвать событие Click, установив равным True значение свойства Value, доступного только во время выполнения. Есть два интересных свойства кнопки, связанных с событием Click. Свойство Default определяет, что данная кнопка является кнопкой, активной по умолчанию. Если это свойство равно True, то нажатием клавиши [Enter] автоматически генерируется событие Click этой кнопки независимо от того, какой элемент имеет фокус. Присваивать значение True этому свойству можно только для одной кнопки в форме. Следует учитывать, что в этом случае нажатие клавиши [Enter] перехватывается и передается этой кнопке. Обычно кнопкой по умолчанию является кнопка OK. Свойство Cancel используется подобно Default. Оно обеспечивает перехват клавиши [Esc] и вызов события Click для соответствующей кнопки. Обычно это свойство имеют кнопки Cancel (Отмена).

1.1.3.2 Надпись (Label)

Надпись (Label) предназначена для отображения текста, который пользователь не может изменить с клавиатуры. Хотя некоторые события

этого элемента управления можно обрабатывать, обычно эта возможность не используется.

Самым важным свойством надписи является Caption, содержащее отображаемый текст. Воспользовавшись свойством BorderStyle, можно установить способ отображения текста – с рамкой или без нее. Оформлять текст можно, используя все возможности форматирования текста, доступные в окне свойств, – от вида и размера шрифта до цвета символов. Если текст длиннее, чем поле надписи, то оставшаяся часть текста просто не отображается (усекается). Этого можно избежать, если присвоить значение True свойству AutoSize, что приводит размер надписи в соответствие с длиной текста. Таким же образом можно корректировать размер надписи и по вертикали. Для этого одновременно со свойством AutoSize нужно установить свойство Wordwrap. Тогда слова, не помещающиеся в строке, автоматически будут переноситься в следующую строку. Установка в тексте надписи перед любой буквой символа амперсанд (&) позволяет определить для выбора объекта клавишу быстрого доступа. Так как надпись не может получать фокус, она передает его следующему элементу управления. Если амперсанд просто должен появляться в тексте без дальнейшего действия, следует отключить свойство UseMnemonic.

1.1.3.3 Текстовое поле (TextBox)

Текстовое поле (TextBox) является основным элементом управления, предназначенным для ввода данных. При использовании текстового поля представляют интерес несколько событий. Прежде всего, это событие Change, которое вызывается при изменении содержимого текстового поля. Это событие происходит каждый раз при вводе, удалении или изменении символа. Например, при вводе в текстовое поле слова "Hello" событие Change вызывается пять раз – по одному разу для каждой буквы.

Для анализа, введенного в поле текста, лучше всего подходит событие LostFocus. Это событие вызывается после того, как текстовое поле становится неактивным (после передачи фокуса другому элементу, т.е. когда пользователь закончит ввод). Чтобы удалить или инициализировать содержимое текстового окна, используется событие GotFocus. Оно вызывается, когда пользователь "входит" в текстовое окно.

Самым важным является свойство Text. Это свойство содержит отображаемый в поле текст

```
Private Sub txtInput_LostFocus()
strText$ = txtInput.Text
End Sub
```

Элементы управления, которые разрешают ввод символов, имеют свойство Text, а элементы, предназначенные только для отображения текста, – свойство Caption. Текстовое поле подобно маленькому текстовому редактору. Чтобы использовать его в таком качестве, достаточно установить свойство MultiLine. Это дает возможность вводить в поле несколько строк. В многострочном поле для перехода на новую строку можно ис-

пользовать клавишу [Enter]. Но при этом следует помнить, что для некоторой кнопки, возможно, установлено свойство Default. Поэтому нажатие клавиши [Enter] вызовет срабатывание этой кнопки. В таком случае для перехода на новую строку надежнее использовать комбинацию клавиш [Ctrl + Enter] или [Shift + Enter].

Если форма не содержит многострочное текстовое поле и кнопку по умолчанию, то нажатие клавиши [Enter] в текстовом поле, имеющем фокус, вызовет звуковой сигнал. Для предотвращения этого напишите следующую процедуру обработки события Keypress текстового поля:

```
Private Sub Textl_Keypress (KeyAscii As Integer)
If KeyAscii =13 Then KeyAscii =0
End Sub
```

При этом клавиша [Enter] просто игнорируется.

При работе с многострочным текстовым полем можно столкнуться со следующей проблемой: если вводимый текст больше, чем может поместиться в текстовом поле, текст хотя и не усекается, но и не отображается полностью. Эта проблем решается установкой свойства ScrollBars. С его помощью можно определить, какую полосу прокрутки будет иметь текстовое поле: горизонтальную, вертикальную или обе. При этом полосы прокрутки функционируют самостоятельно, т.е. вам не нужно писать дополнительный код.

В текстовом поле, как это обычно делается в среде Windows, можно также выделять текст. Для этого Visual Basic предоставляет следующие три свойства текстового окна. Свойство SelStart определяет начальную позицию выделенного текста в символах. Свойство SelLength содержит количество выделенных символов. И, наконец, с помощью свойств SelText можно прочитать или изменить выделенный текст.

1.1.3.4 Флажок (CheckBox)

Флажки – это элементы управления, которые можно отмечать (ставить "галочку"), выбирая из ряда опций одну или несколько. CheckBox может иметь два различных состояния – отмеченное и не отмеченное. Собственно, он может иметь и третье состояние. В этом случае элемент управления отображается как отмеченный, но недоступный. Установить такое состояние элемента управления можно только программно.

Важнейшим для флажка, как и для кнопки, является событие Click.

Единственным важным свойством элемента управления CheckBox является его значение (Value). В зависимости от того, отмечен флажок или нет Value, может принимать значения, приведенные в таблице 1.5.

Таблица 1.5 - Значения флажка

Величина	Значение
0	Не отмечен
1	Отмечен
2	Отмечен, но недоступен

1.1.3.5 Переключатель (OptionButton)

Этот элемент управления, представляющий собой кружок с точкой или без, предназначен для установки только одной опции из группы. Обычно все переключатели формы объединены в одну группу. Если вы желаете сформировать новую группу переключателей, то нужно поместить их в отдельный элемент контейнер, например Frame. Работа с элементами контейнерами будет рассмотрена далее. Так же, как и для элемента управления CheckBox, для переключателей важно только одно событие – Click. Важнейшим свойством переключателей является свойство Value. С его помощью можно определить состояние переключателя. Это свойство может принимать значения True и False.

1.1.3.6 Список (ListBox)

Список – ListBox – позволяет пользователю выбирать из списка один или несколько элементов. В любое время в список можно добавлять новые элементы или удалять существующие. Если не все элементы могут одновременно отобразиться в поле списка, то в нем автоматически отображаются полосы прокрутки. Основное событие списка – Click. Это событие вызывается, если пользователь с помощью мыши или клавиш управления курсором выбирает элемент в списке.

Окно списка – это первый из рассмотренных нами элементов управления, для которых важную роль играют методы. Методы списка необходимы для обработки элементов списка – добавления или удаления. Для добавления новых элементов используется метод Addltem:

```
ListBox.Addltem Элемент[Индекс]
```

Параметр Элемент задает добавляемый элемент списка. С помощью параметра Индекс указывается место вставки в список нового элемента. Данный метод должен вызываться при вставке каждого элемента. Как правило, заполнение списка выполняется при загрузке формы:

Для удаления элемента из списка используется метод RemoveItem, которому в качестве параметра передается индекс удаляемого элемента. Индексация элементов списка начинается с 0.

```
Listbox.Removeltem Индекс_элемента
IstBox.Removeltem 2
```

Для удаления всех элементов списка используется метод Clear:

```
ListBox.Clear
```

Использование свойства списка Text – самая простая возможность получить текст выбранного элемента списка. В любой момент времени значение этого свойства содержит текст выбранного элемента списка или пустую строку, если ни один элемент не выбран. Для определения текста выбранного элемента существуют и другие возможности. Однако следует помнить, что и в памяти все элементы списка сохраняются в виде списка. При этом первый элемент списка имеет индекс 0. Зная это, можно воспользоваться свойством списка List(), которое позволяет определить текст элемента списка по его индексу:

```
thirdltem = IstBox.List(2)
```

Приведенный ниже оператор списка Listlndex содержит индекс выбранного элемента.

```
IstNumber = IstBox.Listlndex
```

Комбинируя свойства List ()и Listlndex, можно получить выбранный элемент списка:

```
Istltem = IstBox.List(lstBox.Listlndex)
```

Если в списке не выбран ни один элемент, значение свойства Listlndex равно 1. Текущее количество элементов в списке сохраняется в свойстве ListCount. Элементы поля списка по умолчанию отображаются в одном столбце. Во время проектирования, при необходимости, их число можно изменить с помощью свойства Columns. Заполнение столбцов в этом случае осуществляется последовательно – сначала заполняется первый, затем второй и т.д.

Свойство Sorted определяет способ расположения элементов в списке. Если установить это свойство, то все элементы будут сортироваться по алфавиту, даже если они были добавлены с указанием индекса. Индекс последнего добавленного элемента содержит свойство Newlndex.

Это свойство связано с другим весьма интересным свойством списка – ItemData(), с помощью которого каждому элементу списка можно поставить в соответствие число типа Long. Используя это свойство, вы можете составить, например, список сотрудников, сохранив их индивидуальные номера в свойстве ItemData. При добавлении в список нового элемента следует позаботиться о присвоении (при необходимости) требуемого значения свойству ItemData, так как оно изначально не инициализировано соответствующим значением ранее добавленного элемента.

Пользователь может выбирать одновременно несколько элементов списка. Для этого следует присвоить свойству Multiselect одно из значений, приведенных в таблице 1.6.

При множественном выборе свойство Text содержит текст последнего выбранного элемента списка. Значение свойства Selected() элемента списка показывает, выделен данный элемент списка или нет. Если свойство равно True, то данный элемент выбран.

1.1.3.7 Полосы прокрутки (ScrollBar)

Элемент управления ScrollBar – это полосы прокрутки окна. Некоторые элементы управления (например, TextBox, ListBox) используют такие полосы прокрутки, причем от разработчика не требуется написание программного кода для выполнения прокрутки. Однако полоса прокрутки как элемент управления Visual Basic хотя и предназначена для выполнения аналогичных функций, но не выполняет автоматически каких-либо действий, т.е. ее поведение необходимо программировать. Существует два вида полос прокрутки: горизонтальная и вертикальная.

Полосы прокрутки имеют два интересных события: Change, которое возникает после изменения позиции бегунка или после программного

изменения значения свойств Value, и Scroll, происходящее во время прокрутки (когда пользователь захватил и передвигает бегунок).

Перед тем как использовать полосу прокрутки, необходимо установить для нее диапазон прокрутки, который показывает количество шагов прокрутки между крайними позициями бегунка. Текущее положение бегунка определяется значением свойства Value.

Диапазон прокрутки определяется свойствами Min и Max полосы прокрутки. При этом значение Min всегда соответствует верхнему концу полосы, Max – нижнему (для вертикальной полосы прокрутки), и при прокрутке содержимого окна сверху вниз значение свойства Value увеличивается. Чтобы изменить направление изменения свойства Value, достаточно поменять местами значения свойств Min и Max.

Щелчок на одной из двух кнопок со стрелками на полосе изменяет значение свойства Value на величину, определяемую свойством SmallChange. Если пользователь щелкнет в области между бегунком и какой-либо из кнопок, то значение свойства Value полосы прокрутки и соответственно положение бегунка изменяется на величину, определяемую свойством LargeChange.

1.1.3.8 Таймер (Timer)

Использование таймера является хорошим способом управления вашей программой. С помощью таймера вы можете запускать или завершать процессы приложения в определенные моменты времени. Таймер может быть полезным и в том случае, если приложение выполняется в фоновом режиме. Во время проектирования таймер отображается в форме, но во время выполнения программы он является невидимым.

Таймер имеет единственное событие – Timer, которое вызывается по истечении установленного временного интервала.

Для установки интервала времени служит свойство Interval, значение которого устанавливается в миллисекундах. Например, задание значения 250 вызывает событие Timer через каждые 250 миллисекунд независимо от того, какое приложение активно. Для отключения таймера следует присвоить свойству Interval значение 0 или свойству Enabled значение False.

Максимально допустимый интервал составляет 64757 миллисекунд. Но следует помнить, что операционная систем может обрабатывать только 18,2 прерывания таймера в секунду, поэтому точность задания интервала составляет максимум одну восемнадцатую секунды. Необходимо также учесть, что при большой загрузке системы (поддержка сети, печать и т.п.) прерывания могут обрабатываться еще реже. В Windows вы можете использовать не более 32 таймеров. Поскольку для работы системы также нужен таймер, то для всех приложений остается максимум 31.

Если обработка события Timer длится дольше, чем задано значением Interval, то новое событие Timer не вызывается, пока Visual Basic не обработает это событие.

1.1.3.9 Рамка (Frame)

Рамка (Frame) – это один из элементов контейнеров. Его назначение – объединить в группу несколько элементов управления. Объекты, объединенные с помощью рамки, можно как единое целое перемещать, активизировать и деактивизировать, делать видимыми или невидимыми. Некоторые элементы сами нуждаются в контейнере – например, все переключатели в форме всегда объединяются в одну группу. Чтобы создать вторую группу опций, нужно требуемые переключатели объединить в контейнере.

Для объединения объектов в группу нужно сначала создать элемент контейнер, затем добавить в него нужные элементы управления. Если требуемые элементы управления уже находятся в форме, их достаточно переместить в элемент контейнер. Чтобы проверить, действительно ли элемент принадлежит контейнеру, достаточно переместить контейнер. Элемент управления, принадлежащий контейнеру, будет перемещаться вместе с ним.

Рамка – это элемент управления, который не имеет особых свойств, присущих только ей. События рамки обычно не анализируются, так как чаще всего проектировщик работает только с элементами управления, принадлежащими рамке.

1.1.3.10 Окно с рисунком (PictureBox)

Как следует из самого названия, элемент PictureBox предназначен для отображения рисунков и других графических объектов. Этот элемент управления также является контейнером, поэтому его можно использовать для объединения других элементов.

Как и события рамки, события PictureBox обычно не обрабатываются, хотя при необходимости это можно сделать.

Change - событие возникает при изменении содержимого PictureBox. Paint - событие Paint возникает при перемещении или изменении размеров PictureBox. События Click, DblClick, Gotfocus, LostFocus, события клавиатуры, мыши возникают и могут обрабатываться так же, как и для других элементов управления.

Положение PictureBox в форме задается свойством Align, которое определяет, будет ли PictureBox закрепляться у одного из краев формы или сохранит положение, заданное разработчиком. Если элемент управления закрепляется у одного из краев формы, то его размер (ширина или высота) всегда устанавливается в соответствии с размером формы.

Свойство AutoSize определяет, будут ли автоматически изменяться размеры элемента управления для отображения рисунков различного размера. Свойство AutoRedraw, равное True, автоматически перерисовывает содержимое PictureBox, если объект изменял размеры, перемещался или был скрыт и появился вновь после удаления (перемещения) закрывавшего его другого объекта. Свойства, возвращающие текущее значение координат по горизонтали (CurrentX) или вертикали (CurrentY) для последующе-

го вывода печати или методов PictureBox. Свойство Image – отображает содержимое PictureBox - отображаемый графический объект. Самое важное свойство PictureBox – Picture, которое содержит отображаемый графический объект. Это может быть растровое изображение (*.BMP), пиктограмма (*.ICO), метафайл (*.WMF) или расширенный метафайл (*.EMF), также GIF или JPEG файлы.

При выполнении приложения для изменения свойства используется функция LoadPicture:

```
Picture1.Picture = LoadPicture("C:\WINDOWS \AUTOS.BMP")
```

Сохранить изображение можно при помощи функции SavePicture:

```
SavePicture Picture1.Picture,"BUILD.BMP"
```

Методы вывода примитивов и печати

Методы PictureBox позволяют нарисовать точку, линию и окружность, также вывести текст (метод Print):

```
picDisplay.Print "Hello world"
picDisplay.Line (0,0)-(100,500),vbRed
picDisplay.Circle (300,300),250,vbBlue
picDisplay.PSet (300,300), vbBlue
```

Способность элемента PictureBox отображать рисунки различных форматов можно использовать для преобразования пиктограммы (*.ICO) в растровое изображение (*.BMP). Для этого нужно загрузить пиктограмму и сохранить ее с расширением BMP. Однако растровое изображение преобразовать в пиктограмму нельзя.

Метод Scale определяет систему координат для PictureBox:

```
picDisplay.Scale (x1, y1) - (x2, y2)
```

x1, y1 as Single – определяют координаты верхнего левого угла PictureBox; x2, y2 as Single – определяют координаты нижнего правого угла PictureBox;

1.1.3.11 Меню

Большинство приложений Windows обладает меню. Вы также можете оснастить им свое приложение, поскольку меню это тоже элемент управления. Прежде чем проектировать меню, следует ознакомиться с некоторыми правилами его создания.

Обычно меню состоят из нескольких уровней. Верхний уровень – это строка меню, в которой находятся элементы главного меню. Они обозначают главные группы команд, например File или Edit. Лучше придерживаться общепринятой структуры меню – это облегчает пользователю изучение программы и работу с ней.

При выборе элемента меню первого уровня автоматически открывается меню второго уровня. Элементы второго уровня "выпадают" в виде списка. В большинстве приложений меню заканчивается на этом уровне. Но можно создать и больше уровней – максимум шесть.

Для обеспечения простого и удобного обращения с меню следует учитывать следующие правила:

• обычно элементы строки меню позволяют открывать меню следующего уровня. Если же требуется, чтобы после выбора команды меню выполнялся код, то текст этого элемента должен заканчиваться восклицательным знаком, например Info!;

• ограничивайтесь двумя или тремя уровнями – так пользователю легче находить элементы меню;

• элементы меню, выполнение которых вызывает появление диалогового окна, отмечаются тремя точками после имени, например Open....

Создание меню в Visual Basic осуществляется с помощью специального инструмента создания меню – редактора меню. Окно редактора меню открывается одним из трех способов: путем нажатия комбинации клавиш [Ctrl+E],нажатием соответствующей кнопки на панели инструментов или после выбора команды меню Tools \Menu Editor....

Элементы меню создаются аналогично элементам управления и имеют такие свойства, как Caption и Enabled. Меню строится иерархически, и его структура выглядит примерно так:

```
Элемент1 Строка_заголовка (Уровень_1)
Уровень_2
Уровень_3
Уровень_4
Уровень_5
Уровень_6
Элемент2 Строка_заголовка
Уровень_2
```

Проще всего создавать меню при помощи встроенного редактора меню Visual Basic. Для этого сначала в поле Caption окна редактора вводится текст, который будет виден в строке меню. Для быстрого открытия меню или вызова команды используются горячие клавиши. Для определения горячих клавиш достаточно в поле Caption перед соответствующей буквой поставить амперсанд (&). Например, для быстрого открытия меню File в поле Caption диалогового окна редактора меню необходимо ввести "&File". Если же в качестве горячих клавиш нужно определить [Alt+i],то в поле Caption следует ввести "F&ile".

Второй шаг – назначение имени элементу меню (так же, как любому элементу управления). Учтите, что Visual Basic не задает имя меню по умолчанию, как для других элементов управления.

При назначении имен элементам меню также нужно соблюдать правила. Имя должно состоять из следующих частей: прежде всего, префикса, указывающего, что это меню, т.е. mnu; затем следует: для пунктов главного меню – имя пункта, для подменю – сокращенные имена родительских пунктов, а затем собственное имя меню. В таблице 1.8 приводятся некоторые примеры.

Последняя задача при создании меню – определение уровней. Для этого воспользуйтесь кнопками со стрелками. Кнопка со стрелкой вправо смещает элемент меню на уровень ниже, а со стрелкой влево – на уровень

выше. Кнопки с вертикальными стрелками изменяют позицию отдельных элементов меню, т.е. перемещают их в списке вверх или вниз.

Таблица 1.8 - Имена меню

Команда меню	Имя
File	mnuFile
File\Qpen...	mnuFOpen
File\Send\Eax	mnuFSFax

Поле Shortcut позволяет определить комбинации клавиш для быстрого доступа к отдельным элементам меню. Среди наиболее часто используемых приложениями Windows комбинаций клавиш можно отметить следующие: [Ctrl+X] (Cut), [Ctrl+C] (Copy), [Ctrl+V] (Paste).

Свойства элементов меню можно изменять и во время выполнения. При этом синтаксис такой же, как и для других элементов управления.

Для создания процедуры выполнения команды меню следует во время проектирования выбрать соответствующий пункт в форме. При этом создастся процедура обработки события Click. Другие элементы меню можно найти и в списке объектов окна кода.

1.2 Порядок выполнения работы

1.2.1 Задания для выполнения работы

1. Включить компьютер.
2. Запустить на выполнение программу «Visual Basic 6.0».
3. Открыть проект Standard.EXE.
4. Разработать формы, содержащие необходимое количество элементов управления.
5. Написать коды обработки событий, связанных с элементами управления.
6. Написать коды обработки информации.
7. Задание для каждого студента приведено ниже.
8. Запустить проект на выполнение.
9. При необходимости исправить ошибки в проекте.

1.2.2 Порядок выполнения

1. Разработать проект для реализации алгоритмов, разработанных для реализации заданий.
2. Проект должен содержать формы и программный модуль.
3. Каждая форма должна содержать необходимые элементы управления для реализации соответствующего алгоритма, а также строку состояния, меню и панель инструментов для облегчения работы пользователя с программой.

4. Для каждого элемента управления используйте всплывающую подсказку (Заполните свойство ToolTipText для каждого элемента управления). На каждом шаге работы проекта в строку состояния должна выводиться подсказка о дальнейших действиях пользователя.

5. При построении графика по точкам запрограммируйте ProgressBar, который должен отражать ход выполнения вычислительного процесса при выводе графика в PictureBox.

6. Программный модуль должен содержать процедуры (функции), выполняющие необходимые вычисления, обусловленные вариантом задания.

7. При старте проекта нужно ввести необходимые данные (параметры кривых, координаты отдельных точек) и провести их обработку.

8. Исходные данные (параметры кривых, координаты отдельных точек) и результаты обработки (построенные изображения графиков и плоских фигур сохранить в файле) необходимо занести (вставить изображения из файла) в отчет.

1.2.3 Варианты заданий

1.2.3.1 Построить график функции

Общее задание для всех вариантов:

При старте проекта (в процедуре обработки события Form_Load) установите фокус на соответствующий элемент управления для ввода параметра(ов). При вводе нескольких параметров переход между соответствующими окнами организуйте через нажатие клавиши Tab.

В PictureBox создайте согласно с полученным вариантом задания изображение геометрической фигуры. С помощью мыши выберите отдельный его элемент и измените для него толщины линий и цвет. В закрашенных областях измените цвет заливки изображения. Для этого используйте соответствующие пункты меню и/или кнопки на панели инструментов.

Перед выводом изображения проведите проверку значений введенных параметров, и при необходимости организуйте очистку соответствующих окон и предложение с подсказкой о вводе неверных параметров. Вычисленные параметры должны быть перед построением вычислены и выведены на форму в соответствующие окна с метками, указывающими на то, какой параметр выводится в близлежащем окне.

Вариант № 1. Построить циссоиду Диоклеса: $y^2 \cdot (a - x) = x^3$ или $\rho = a(\cos^{-1}(\varphi) - \cos(\varphi))$. Вводимый параметр a. Для построения используйте полярные координаты.

Вариант № 2. Построить лемнискату Бернулли: $(x^2 + y^2)^2 - a^2 \cdot (x^2 - y^2) = 0$ или $\rho^2 - a^2 \cdot \cos(2\varphi) = 0$. Вводимый параметр a. Для построения используйте полярные координаты.

Вариант № 3. Построить крест: $x^2 \cdot y^2 = a^2 \cdot (x^2 + y^2)$ или $\rho = 2a / \sin(2\varphi)$. Вводимый параметр a. Для построения используйте полярные координаты.

Вариант № 4. Построить кардиоиду: $(x^2 + y^2 – a·x)^2 = a^2·(x^2 + y^2)$ или $\rho = a·(1 + \cos(\varphi))$. Вводимый параметр a. Для построения используйте полярные координаты.

Вариант № 5. Построить трисектрису: $y^2 = x^3·(3a – x) / (a + x)$ или $\rho = a·(4\cos(\varphi) – \cos^{-1}(\varphi))$. Вводимый параметр a. Для построения используйте полярные координаты.

Вариант № 6. Построить астроиду: $x^{2/3} + y^{2/3} = a^{2/3}$ или $x = a·\cos^3(\varphi)$, $y = a·\sin^3(\varphi)$. Вводимый параметр a. Для построения используйте полярные координаты.

Вариант № 7. Построить улитку Паскаля: $\rho = b – a·\cos(\varphi)$. Вводимые параметры a и b. Для построения используйте полярные координаты.

Вариант № 8. Построить эволюту эллипса: $a^{2/3}·x^{2/3} + b^{2/3}y^{2/3} = (a^2 – b^2)^{2/3}$ или $x = (a^2 – b^2)/a·\cos^3(t)$, $y = (a^2 – b^2)/b·\sin^3(t)$. Вводимые параметры a и b. Используйте полярные координаты.

Вариант № 9. Построить циклоиду: $x = a·(t – \sin(t))$, $y = a·(1 – \cos(t))$. Кривая задана параметрически. Параметр t изменяется в интервале [c, d]. Вводимые параметры a, c и d.

Вариант № 10. Построить лист Декарта: $x^3 + y^3 – 3a·x·y = 0$ или $x = 3a·t/(1 + t^3)$, $y = 3a·t^2/(1 + t^3)$.)). Кривая задана параметрически. Параметр t изменяется в интервале [c, d]. Вводимые параметры a, c и d.

Вариант № 11. Построить трохоиду: $x = a·t – h·\sin(t)$, $y = a – h·\cos(t)$ (при $a > h$). Кривая задана параметрически. Параметр t изменяется в интервале [c, d]. Вводимые параметры a, h, c и d.

Вариант № 12. Построить эпициклоиду: $x = (a + b)·\cos(t) – a·\cos((1 + b/a)·t)$, $y = (a + b)·\sin(t) – a·\sin((1 + b/a)·t)$ при $a / b = p / q$, где p и q – целые числа. Кривая задана параметрически. Параметр t изменяется в интервале [c, d]. Вводимые параметры a, b, p, q, c и d.

Вариант № 13. Построить трохоиду: $x = a·t – h·\sin(t)$, $y = a – h·\cos(t)$ (при $a < h$). Кривая задана параметрически. Параметр t изменяется в интервале [c, d]. Вводимые параметры a, h, c и d.

Вариант № 14. Построить гипоциклоиду: $x = (b – a)·\cos(t) + a·\cos((b/a – 1)·t)$, $y = (b – a)·\sin(t) – a·\sin((b/a – 1)·t)$, если a / b – иррациональное число. Кривая задана параметрически. Параметр t изменяется в интервале [c, d]. Вводимые параметры a, b, c и d.

Вариант № 15. Построить спираль Архимеда: $\rho = a·\varphi$ ($a > 0$). Для построения используйте полярные координаты. Азимут φ изменяется в интервале [c, d]. Вводимые параметры a, c и d.

1.2.3.1 Построить плоские фигуры

Общее задание для всех вариантов:

Построить плоские фигуры, используя методы PictureBox (Line, Circle) после вычисления координат соответствующих точек и других параметров.

Вариант № 1. Равносторонний треугольник, вписанный в окружность. Вводимые параметры: радиус окружности, координаты ее центра, координаты одной из вершин треугольника. Вычисляемые параметры и координаты: сторона треугольника, координаты вершин треугольника.

Вариант № 2. Произвольный треугольник, вписанный в окружность. Вводимые параметры: радиус окружности, координаты ее центра, координаты одной из вершин треугольника. Вычисляемые параметры и координаты: координаты вершин треугольника, стороны треугольника.

Вариант № 3. Окружность, вписанная в равносторонний треугольник. Вводимые параметры: сторона треугольника, координаты одной из вершин треугольника. Вычисляемые параметры и координаты: координаты вершин треугольника, радиус окружности, координаты ее центра.

Вариант № 4. Окружность, вписанная в произвольный треугольник. Вводимые параметры: стороны треугольника, координаты одной из вершин треугольника. Вычисляемые параметры и координаты: координаты вершин треугольника, радиус окружности, координаты ее центра.

Вариант № 5. Квадрат, вписанный в окружность. Вводимые параметры: радиус окружности, координаты ее центра, координаты одной из вершин квадрата. Вычисляемые параметры и координаты: координаты вершин квадрата, сторона квадрата.

Вариант № 6. Окружность, вписанная в квадрат. Вводимые параметры: радиус окружности, координаты ее центра, координаты одной из вершин квадрата. Вычисляемые параметры и координаты: координаты вершин квадрата, сторона квадрата.

Вариант № 7. Прямоугольный треугольник, вписанный в окружность и опирающийся на ее диаметр. Вводимые параметры: радиус окружности, координаты ее центра, координаты одной из вершин треугольника. Вычисляемые параметры и координаты: координаты вершин треугольника, стороны треугольника.

Вариант № 8. Окружность и ромб, меньшая диагональ которого является диаметром окружности. Вводимые параметры: диагонали ромба, координаты его центра. Вычисляемые параметры и координаты: радиус окружности, координаты ее центра.

Вариант № 9. Ромб и окружность, диаметром которой является большая диагональ ромба. Вводимые параметры: диагонали ромба, координаты его центра. Вычисляемые параметры и координаты: радиус окружности, координаты ее центра.

Вариант № 10. Правильный шестиугольник, вписанный в окружность. Вводимые параметры: радиус окружности, координаты ее центра. Вычисляемые параметры и координаты: сторона шестиугольника, координаты его вершин.

Вариант № 11. Окружность, вписанная в правильный шестиугольник. Вводимые параметры: сторона шестиугольника, координаты его вершин. Вычисляемые параметры и координаты: радиус окружности, координаты ее центра.

Вариант № 12. Окружность, диаметром которой является основание равнобедренного треугольника, высота которого в два раза больше основания. Вводимые параметры: основание треугольника, координаты одной из его вершин. Вычисляемые параметры и координаты: высота треугольника, координаты остальных его вершин, радиус окружности, координаты ее центра.

Вариант № 13. Прямоугольную трапецию, вписанную в окружность. Вводимые параметры: радиус окружности, координаты ее центра, высота трапеции, координаты одной из её вершин. Вычисляемые параметры и координаты: координаты остальных вершин трапеции, длины её сторон.

Вариант № 14. Прямоугольную трапецию на меньшей диагонали, которой построена как на диаметре окружность. Вводимые параметры: высота и основание и противоположная ему сторона трапеции, координаты одной из её вершин. Вычисляемые параметры и координаты: координаты остальных вершин трапеции, длины её сторон, радиус окружности, координаты ее центра.

Вариант № 15. Окружность и вписанный в нее прямоугольник, стороны которого относятся как 1 : 2. Вводимые параметры: две не противоположные стороны прямоугольника и, координаты одной из его вершин. Вычисляемые параметры и координаты: координаты остальных вершин прямоугольника, радиус окружности, координаты ее центра.

1.2.4 Содержание отчета

В отчете должны быть представлены:

1. Цель работы.
2. Основные сведения о теме работы.
3. Алгоритм разработанного проекта.
4. Перечень форм и элементов управления.
5. Процедуры обработки данных.
6. Исходные данные и результаты их обработки.

1.3 Контрольные вопросы

1. Каково назначение элементов управления (Controls)?
2. Как изменять свойства элементов управления (Controls) во время выполнения приложения?
3. Перечислите основные свойства элементов управления (Controls).
4. Для чего используют свойства – Enabled и Visible?

5. Перечислите основные события, которые могут обрабатываться большинством элементов управления.
6. Что означает в Windows понятие Фокус?
7. Какие события связаны с понятием Фокус?
8. Перечислите специфические свойства и события формы.
9. Перечислите основные элементы управления.
10. Как вводится (выводится в файл) из файла изображение в PictureBox?
11. Как создается меню?

ЛАБОРАТОРНАЯ РАБОТА 2

ЭЛЕМЕНТЫ УПРАВЛЕНИЯ WINDOWS В VISUAL BASIC. МАССИВЫ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ

Цель работы – Научиться работать с различными элементами управления Windows. Научиться создавать массивы элементов управления в Visual Basic 6. Создать проект на Visual Basic 6, содержащий формы, массивы элементов управления, процедуры и функции.

2.1 Общие сведения

2.1.1 Элементы управления Windows

Специально для работы с Windows в Visual Basic были включены элементы управления Microsoft Windows Common Controls, перечисленные в таблице 2.1.

2.1.1.1 ImageList – набор зображений

ImageList – это элемент управления, который никогда не используется самостоятельно. Он предоставляет другим элементам управления список графических образов. ImageList невидим в форме во время выполнения программы. Он только предоставляет информацию для других элементов управления и не имеет собственных событий. Отдельные изображения ImageList просто используются другими элементами управления.

Важнейшее свойство ImageList – это Index, с помощью которого другие элементы управления выбирают изображения из ImageList. При этом во вкладке свойств General нужно установить размер изображения. Для определенных целей, например при просмотре с использованием элемента управления ListView, предполагается неизменный размер изображения.

В Visual Basic этот элемент управления поддерживает форматы *.GIF, *.JPG и *.CUR.

Таблица 2.1 - Элементы управления Windows

Элемент управления	Назначение
Animation	Элемент отображения анимации.
CoolBar	Используется для создания конфигурируемых пользователем панелей элементов (аналогично Microsoft Internet Explorer).
DTPicker	Представляет собой комбинацию текстового поля и календаря, с помощью которого пользователь может легко вводить дату и время
FlatScrollBar	Плоская версия полосы прокрутки.
ImageCombo	Подобен элементу управления ComboBox с одним преимуществом: вы можете добавлять графические изображения в список элементов.
ImageList	Элемент хранения списков графических изображений для других элементов управления.
ListView	Элемент для отображения списка каких-либо элементов.
MonthView	Элемент управления, позволяющий пользователю выбирать дату или последовательность дат с помощью графического представления календаря.
ProgressBar	Элемент индикации процесса выполнения длительных операций.
Slider	Скользящий указатель (слайдер) или элемент установки значения из диапазона.
StatusBar	Панель состояния.
TabStrip	Элемент для отображения многостраничных вкладок.
ToolBar	Панель инструментов.
TreeView	Элемент отображения иерархических структур.
UpDown	Элемент изменения значений.

2.1.1.2 ToolBar – панель инструментов

Панель инструментов (ToolBar) есть в каждом приложении Windows. Visual Basic предоставляет элемент управления для создания панели инструментов – ToolBar. ToolBar – это панель с различными кнопками, свойства которых определяются разработчиком.

Важнейшее значение при этом имеет свойство Key, используемое для определения выбранной пользователем кнопки. Для анализа выбора кнопки обрабатывается событие ButtonClick. Процедуре обработки события передается объект типа Button, идентифицирующий нажатую кнопку. Затем с помощью свойств Key определяется, какая кнопка нажата.

```
Private Sub Toolbar1_ButtonClick(ByVal Button As Button)
```

```
If Button.Key ="tbLoad" Then
MsgBox "Теперь загружается"
End If
End Sub
```

Кнопки панели инструментов можно оформить графическими изображениями, для чего используется элемент управления ImageList.

Возможности элемента управления ImageList расширены добавлением нового семейства ButtonMenus. Это семейство содержит набор объектов ButtonMenu, каждый из которых представляет собой элемент меню, появляющегося при нажатии на кнопку (если такое меню было добавлено при разработке). Однако в этом случае значение свойства Style такой кнопки должно быть равным tbrDropdown.

DisabledlmageList содержит ссылку на элемент управления ImageList, – набор графических изображений, используемых в случае, если кнопка становится недоступной (Disabled). Во-вторых, свойство HotImageList, которое подобно свойству DisabledlmageList, но дополнительно содержит ссылку на список графических изображений, отображаемых при открытии меню. Еще одно свойство – Style, определяющее стиль панели инструментов. И, наконец, свойство TextAlignment, с помощью которого можно определить или задать позицию вывода текста – под рисунком (tbrTextAlignBottom) или справа от него (tbrTextAlignRight).

Событие ButtonDropDown происходит после щелчка на стрелке (dropdown arrow) элемента управления Button. Такая стрелка присутствует у элемента управления только тогда, когда значение его свойства Style равно tbrDropdown. Событие ButtonDropDown происходит до события ButtonMenuClick и может быть использовано для определения набор пунктов открывающегося меню (их добавления, удаления или редактирования). Событие ButtonMenuClick происходит после щелчка мышью на объекте ButtonMenu.

2.1.1.3 StatusBar – панель состояния

Панель состояния (StatusBar)– это тоже важный элемент, который должен быть в каждом приложении Windows. Речь идет об узкой полосе внизу окна, отображающей различную информацию, например дату, время и многое другое. Для создания такой панели состояния в семействе Windows Common Controls имеется элемент управления StatusBar.

Панель состояния может содержать до 16 отдельных панелей, в которых отображается различная информация: время, дата, число страниц и пр. С помощью свойства Style можно установить вид панели состояния: она может быть простой или состоящей из нескольких частей (табл. 2.2).

Отдельные части или поля панели состояния описывают определяемые пользователем свойства. С их помощью можно добавлять новые

поля или изменять их содержимое. Отдельные части панели состояния обслуживаются семейством объектов Panels. Во время выполнения программы можно изменять ее содержимое, параметры выравнивания текста и графики, оформление и т.д.

Таблица 2.2 - Константы стиля панели состояния

Константа	Значение	Вид панели
SbrNormal	0	Составная
SbrSimple	1	Простая текстовая

События панели состояния имеют второстепенное значение, т. к. как этот элемент обычно используется только для отображения информации. Но он, конечно, может обрабатывать события Click и DblClick.

2.1.1.4 ProgressBar – отображение процесса выполнения операцій

Индикатор или ProgressBar можно найти во многих приложениях Windows. Этот элемент управления отображает, насколько продвинулся процесс копирования, перемещения, загрузки или сохранения файлов.

Важнейшими свойствами являются Min (нижняя граница), Max (верхняя граница) и Value (текущее значение). Если необходимо отображать процесс от 0 до 100%, то устанавливается Min = 0, Max = 100 и затем для Value – значение в данный момент. Определение значения свойств Value предоставляется разработчику, т. к. как сам элемент управления не имеет возможности отслеживания продвижения процесса:

```
' Пример для загрузки нескольких файлов
Sub LoadFiles()
Progress1.Min = 0: Progress1.Max = nFiles
For i =1 To nFiles
Call LoadFile(i)
Progress1.Value = i
Next i
End Sub
```

Так могло бы выглядеть отображение прогресса при загрузке нескольких файлов, изображений и т.п. Свойство Orientation позволяет задать ориентацию этого элемента управления – вертикальную или горизонтальную, а свойство Scrolling определяет способ отображения на экране процесса – непрерывный или сегментарный. Метод Refresh позволяет принудительно вызвать перерисовку элемента управления.

2.1.1.5 ListView – просмотр списков

С элементом управления TreeView часто используется элемент управления ListView, тоже предназначенный для просмотра списков, но без иерархической структуры. Элемент управления ListView может отображать элементы в виде пиктограмм (маленьких и больших), списка или

таблицы. Различные виды отображения можно увидеть на примере Windows Explorer.

Каждый элемент списка представляет собой объект типа ListItem и хранится в семействе ListItems. Добавление элементов в семейство осуществляется с помощью метода Add (табл. 2.4):

```
ListItems.Add ([Index], [Key], Text, [Icon], [SmallIcon])
```

Каждый объект ListItem обладает свойством SubItems, в котором хранятся дополнительные данные, но отображаются они, только если установлен режим отображения (свойство View) "Таблица". В этом случае при просмотре информация представляется в нескольких столбцах, которые хранятся в семействе ColumnHeaders. Добавление столбцов происходит с помощью метода Add (табл. 2.5):

```
ColumnHeaders.Add ([Index], [Key], Text, [Width], [Alignment])
```

Таблица 2.4 - Параметры метода Add

Параметр	Значение
Index	Индекс добавляемого элемента
Key	Уникальная строка идентификатор
Text	Текст элемента списка
Icon	Индекс крупных пиктограмм в ImageList
SmallIcon	Индекс мелких пиктограмм в ImageList

Таблица 2.5 - Параметры метода Add

Параметр	Значение
Index	Индекс столбца в семействе
Key	Уникальная строка идентификатор
Text	Текст заголовка столбца
Width	Ширина столбца
Alignment	Выравнивание текста в столбце

В качестве примера можно привести следующий код:

```
Private Sub Form_Load()
Dim Col As ColumnHeader
Set Col = ListView1.ColumnHeaders.Add( , ,"Caption", ListView1.Width /3)
Set Col = ListView1.ColumnHeaders.Add( , ,"Name", ListView1.Width /3)
Set Col = ListView1.ColumnHeaders.Add( , ,"Visible", ListView1.Width /3)
ListView1.Icons = ImageList1
Dim f As Form
Dim Insert As ListItem
For Each f In Forms
Set Insert =ListView1.ListItems.Add ( , ,f.Caption, 1)
```

```
Insert.SubItems (l) = "f. Name»
Next f
End Sub
```

В результате выполнения данной процедуры отображается список всех загруженных форм с указанием их заглавий, имен и указанием, является ли форма видимой (свойства Caption, Name и Visible).

Редактирование надписей обрабатывается так же, как и для элемента управления TreeView – здесь тоже есть свойство LabelEdit и события BeforeLabelEdit и AfterLabelEdit.

Для обработки событий щелчка мыши, кроме события Click, можно обрабатывать и события ItemClick и ColumnClick, наступающие при щелчке соответственно на отдельном элементе или столбце.

Семейство ListSubItems содержит объекты ListSubItem, которые представляют собой подэлементы (subitem) элемента управления ListView и заменяют массив строк SubItems, используемый в предыдущих версиях этого элемента управления.

С помощью свойств AllowColumnReorder разработчик может разрешать или запрещать пользователю изменять порядок колонок в списке с помощью мыши. Чтобы такая возможность существовала, необходимо установить значение свойства равным True.

Свойство ColumnHeaderIcons содержит ссылку на элемент управления ImageList, который обеспечивает рисунками семейство ColumnHeaders. Свойство FlatScrollBar определяет внешний вид полос прокрутки – стандартный (False, по умолчанию) или плоский (True).

Свойство GridLines определяет, будут ли отображаться линии сетки, если элемент управления ListView отображается в виде отчета (Report View). По умолчанию присваивается значение False, и линии сетки не отображаются.

Свойство HoverSelection позволяет определить поведение объекта ListItem при перемещении по нему курсора мыши. Если значение свойства равно True, то при перемещении курсора мыши по объекту ListView соответствующий объект ListItem будет автоматически активизироваться.

Свойство Picture содержит графическое изображение, которое будет появляться в элементе управления. Загрузить рисунок во время работы приложения можно с помощью функции LoadPicture. Позиция рисунка в объекте задается значением свойства PictureAlignment.

Порядок следования объектов ColumnHeader можно задавать с помощью свойства Position этого объекта. Его значение может быть любым целым числом в диапазоне от 1 до n, где n – количество объектов ColumnHeader.

Фон текста для объекта ListItem задается с помощью свойства TextBackGround. Если его значение равно IvwTransparent, то фон будет

прозрачным, если IvwOpaque, то цвет фона будет определяться значением свойства BackColor.

Событие ItemCheck, подобно событию NodeCheck элемента управления TreeView, происходит при изменении состояния элемента (отмечен/не отмечен), если элементы отображаются в виде флажков.

Элемент управления ListView имеет также новые свойства CausesValidation, Checkboxes, FullRowSelect, HotTracking и новое событие Validate, которые были описаны ранее.

2.1.1.6 Slider – элемент установки значения из диапазона

Slider – элемент управления, часто встречающийся в приложениях Windows. Он позволяет выбрать дискретное значение или набор значений из определенного диапазона.

В принципе, этот элемент функционирует аналогично ScrollBar. Он также имеет свойства Min, Max и Value, которые устанавливают границы области значений и текущее значение. Параметры изменения значения при перемещении в области значений определяют свойства SmallChange и LargeChange. В отличие от ScrollBar, для этого элемента можно определить не только одно значение, но и некоторый диапазон значений. Для этого следует воспользоваться свойствами SelStart и SelLength. Но само выделение диапазона должно выполняться программно.

Свойство Text позволяет задавать текст надписи, который будет отображаться при перемещении ползунка. Позиция отображения этой надписи определяется значением свойства TextPosition. Можно использовать событие Validate.

2.1.1.7 Элемент управления UpDown

Элемент управления UpDown, присутствующий во многих приложениях Windows, используется для установки различных значений. Важнейшее преимущество этого элемента управления заключается в том, что он может использоваться без написания кода – достаточно правильно определить его свойства AutoBuddy, SyncBuddy, BuddyControl и BuddyProperty. Buddy – это элемент управления, значение которого меняет UpDown. Buddy можно установить с помощью свойства BuddyControl или свойства AutoBuddy, при этом автоматически будет использоваться элемент управления со свойством TabIndex на единицу меньшим, чем у UpDown. Свойство BuddyProperty указывает, какое свойство элемента Buddy должно синхронизироваться со свойством Value элемента UpDown. Для этого дополнительно устанавливается свойство SyncBuddy. Область значений устанавливается с помощью свойств Min, Max и Value.

2.1.1.8 Элемент управления Animation

Элемент управления Animation предназначен, прежде всего, для отображения небольшой анимации в диалоговом окне, как, например, в Windows при копировании файлов. При этом могут воспроизводиться

только несжатые AVI файлы без звука либо сжатые с использованием технологии Run Length Encoding (RLE). Некоторые из файлов инсталлируются в каталоге графики Visual Basic.

Основные методы – Open и Play, с помощью которых можно загружать и воспроизводить видеофайлы.

2.1.1.9 Элементы управления для работы с датами и временем

Для обработки значений дат и времени имеются два элемента управления – DateTimePicker и MonthView.

Элемент управления MonthView позволяет пользователю выбирать дату или последовательность дат с помощью графического представления Календаря.

Свойства Day, Month и Year позволяют задавать или считывать соответственно день, месяц и год. С помощью свойств MinDate и MaxDate разработчик может ограничить диапазон дат, с которыми может работать пользователь. Свойство MultiSelect определяет возможность выбора непрерывной последовательности дат, а свойство MaxSelCount – длину этой последовательности в днях. Для считывания или установки начальной и конечной даты последовательности предназначены свойства SelStart и SelEnd. Свойства MonthColumns и MonthRows позволяют указать количество месяцев, которые будут отображаться на экране (по горизонтали и по вертикали). Это позволяет просматривать одновременно на экране календарь на несколько месяцев (но не более 12). Этот элемент управления можно также связать с определенным полем базы данных, для чего тоже предусмотрен набор свойств.

Среди событий этого элемента управления выделим DateClick и DateDblclick, которые происходят после соответственно одинарного или двойного щелчка на дате. Обрабатывая эти события и используя свойство Value, разработчик имеет возможность определить значение даты, выбранной пользователем.

Элемент управления DateTimePicker выглядит, как поле со списком, и позволяет отображать дату и/или время, а также вводить дату при помощи элемента управления MonthView. Этот элемент управления может функционировать в двух режимах:

- режим ниспадающего календаря (по умолчанию) – позволяет пользователю выбирать требуемую дату в календаре;
- режим форматирования времени – позволяет пользователю выбирать поле в отображаемой дате (день, месяц, год, и т.п.) и изменять его значение, нажимая кнопки на элементе управления UpDown, отображаемого в правой части поля со списком.

Среди свойств этого элемента управления имеются, кроме свойств Day, Month и Year, также свойства Hour, Minute и Second, содержащие значения часов, минут и секунд.

Важными являются также свойства Format и CustomFormat, определяющие формат отображения значений даты и времени. Свойство Format может принимать одно из следующих значений: dtpLongDate (дата отображается в длинном формате), dtpShortDate (дата отображается в кратком формате), dtpTime (формат времени) или dtpCustom (пользовательский формат). Если значение свойства Format равно dtpCustom, то при отображении Даты/времени используется пользовательский формат, определяемый свойством CustomFormat.

Среди событий этого элемента управления отметим, во-первых, событие CloseUp, происходящее после закрытия ниспадающего календаря. Во-вторых, событие Change, происходящее при изменении содержимого элемента управления. И, наконец, событие Validate, обрабатывая которое, можно предотвратить потерю элементом управления фокуса при вводе некорректного значения пользователем.

2.1.2 Массивы элементов управления

Массив элементов управления представляет собой группу элементов управления одного типа, которые идентифицируются по одному и тому же имени и индексу. Индекс действует аналогично обычному массиву, как, например, в массиве чисел. Каждый из элементов массива характеризуется свойством Name, имеющим одно и то же значение для всех элементов. Все они относятся к одному типу. Идентифицируются элементы такого массива с помощью свойства Index. При написании кодов программы для обращения к элементу массива элементов управления необходимо указывать как его имя Name, так и индекс Index:

```
Имя_Control(Index).Property = …
```

2.1.2.1 Создание массива элементов управления на этапе разработки

Массив элементов управления может создаваться при проектировании приложения. Для создания массива существуют следующие основные способы:

- созданный элемент управления дублируется на форме или в контейнере при помощи команд Copy [Ctrl + C] (Копировать) и Paste [Ctrl + V] (Вставить) меню Edit (Правка);
- создаются элементы управления одного типа по отдельности, а затем им присваивается одно и то же имя в свойстве Name и соответствующее значение индекса в свойстве Index.

2.1.2.2 Создание массивов элементов управления в режиме выполнения проекта

При подготовке проекта к созданию массива элементов управления во время работы приложения необходимо на форме объявить массив элементов управления (на этапе проектирования):

```
Private lblObject() As Control
```

При работе приложения после того, как станет известно количество N (N и M для двумерного массива) элементов управления, необходимо переобъявить соответствующий массив, указав его размерность и количество элементов как для обычного массива.

```
Redim lblObject(N)
```

Затем в цикле по всем создаваемым элементам управления необходимо задать значение строки с именем типа элемента управления и значение строки с уникальным именем соответствующего элемента управления. Пример создания массива меток приведен ниже с соответствующими комментариями.

```
For i = 0 To N
strID = "VB.Label"
‘ строка с именем типа элемента управления
strW = "lblA" & Trim(i)
‘ строка с уникальным именем соответствующего элемента управления, например: lblA2.
frmLab2DT.Controls.Add strID, strW
' Создание метки на форме
Set lblObject(i) = frmLab2DT.Controls.Item(strW)
' Новую метку добавить в массив меток (Коллекцию элементов управления формы).
Next i
```

Не забудьте программно задать такие свойства элемента массива, как Top, Left и т.п. Аналогичным образом можно создавать текстовые окна (VB.TextBox) с соответствующими метками (VB.Label) для ввода одно- и двумерных массивов данных.

2.2 Порядок выполнения работы

2.2.1 Задания для выполнения работы

1. Включить компьютер.
2. Запустить на выполнение программу «Visual Basic 6.0».
3. Открыть проект Standard.EXE.
4. Разработать формы, содержащие необходимое количество элементов управления.
5. Написать коды обработки событий, связанных с формой и элементами управления.
6. Написать коды обработки информации.
7. Задание для каждого студента приведено ниже.
8. Запустить проект на выполнение. При необходимости отладить работу проекта.

2.2.2 Порядок выполнения

1. Разработать проект для реализации алгоритма, приведенного в вариантах заданий.

2. Проект должен содержать как минимум одну форму и программный модуль.

3. Форма должна содержать необходимые элементы управления для реализации алгоритма.

4. Программный модуль должен содержать процедуры и функции, выполняющие необходимые действия, обусловленные вариантом задания.

5. При старте проекта ввести необходимые данные и провести их обработку.

6. Исходные данные и результаты обработки необходимо занести в отчет.

2.2.3 Варианты заданий

Общее задание для всех:

Размерности одно- и двумерных массивов вводить на форме в специальное окно. После этого по величине размерности массива создать в ходе работы проекта необходимое количество меток и текстовых окон для ввода (вывода) значений элементов соответствующих числовых массивов. Допускается заполнение текстовых окон при их создании числами из соответствующего диапазона [a, b] с помощью генератора случайных чисел. Диапазон [a, b], приведенный в соответствующем варианте задания, должен вводиться на форме в соответствующие окна.

Вариант № 1. Ввести число n (Integer) (n ≥ 5). Создать массивы случайных чисел (Single): X(n) из диапазона [1.0, 6.1]; Y(n) из диапазона [-24, 81]; A(n, n) из диапазона [-24, 11]. Вычислить максимальный элемент M массива X(n) и билинейную форму $B = M\sum_{i=1}^{n} x_i \sum_{k=1}^{n} A_{ik} y_k$.

Вариант № 2. Ввести число n (Integer) (n ≥ 6). Положить m = n – 1; k = n + 1. Создать массивы случайных чисел (Single): X(m) из диапазона [0.4, 6.9]; A(k) из диапазона [-0.6, 5.2]. Вычислить полином $P_n(x_j)$ по схеме Горнера (т.е. $P_n(x) = ((a_1x + a_2)x + a_3)x + a_4)x + \dots a_n)x + a_{n+1})$.

Вариант № 3. Ввести число n (Integer) (n ≥ 8). Положить m = n – 2. Создать массивы случайных целых чисел (Integer): X(n) из диапазона [0, 16]; Y(m) из диапазона [-4, 10]. Вычислить $Z = \prod_{k=1}^{m}\left(\left(\sum_{j=1}^{n} x_j\right) y_k + \sum_{j=1}^{n}\left(x_j + y_k\right)\right)$.

Вариант № 4. Ввести число n (Integer) (n ≥ 6). Создать массивы случайных чисел (Single): X(n – 2) из диапазона [0.7, 5.5]; A(n) из диапазона [-4.9, 5.2]. Первая производная полинома $P_n(x) = \sum_{i=0}^{n} a_i x^i$ представляет

собой полином $P_{n-1}(x) = \sum_{i=0}^{n-1} b_i x^i$, где $b_i = (i + 1)a_{i+1}$, i Є [0, n – 1]. Вычислить первую производную $P_{n-1}(x_k)$ для всех $\{x_k\}$.

Вариант № 5. Ввести число n (Integer) (n ≥ 8). Создать массив случайных чисел (Single) упорядоченных по возрастанию X(n) из диапазона [-6.1, 31.1]. Вычислить значение аппроксимирующего многочлена P(x) = $a_1 + a_2 \cdot x$ в точках $\{x_i\}$, если для вычисления коэффициентов a_1, a_2·необходимо решить систему: $\begin{cases} S_1 a_1 + S_2 a_2 = t_1 \\ S_2 a_1 + S_3 a_2 = t_2 \end{cases}$, где $S_k = \sum_{i=1}^{n} x_i^{k-1}$, k = 1, 2, 3; $t_j = \sum_{i=1}^{n} x_i^{j-1}$, j = 1, 2.

Вариант № 6. Ввести число n (Integer) (n ≥ 6). Создать массивы случайных чисел (Single): X(n) из диапазона [–1.6, 4.9]; Y(n) из диапазона [-2.8, 8.6]. Вычислить расстояние между векторами X и Y, если известно, что расстояние между векторами A и B размером n определяется по формуле $d = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2}$.

Вариант № 7. Ввести числа n и k (Integer) (n ≥ 4; k < n). Создать массив случайных чисел (Single) упорядоченных по возрастанию A(n, n) из диапазона [-6.3, 4.2]. Вычислить вектор Z = $\{z_i\}$, являющийся произведением матрицы A = $\{a_{ij}\}$ на вектор X = $\{x_j\}$, элементы которого вычисляются по формуле $x_j = \begin{cases} k \cdot \sin j, & j \le k, \\ \cos j, & k < j \le n, \end{cases}$ j = 1, …, n. Каждую компоненту вектора Z определяют по формуле $z_i = \sum_{j=1}^{n} a_{ij} x_i$, i = 1,…,n.

Вариант № 8. Ввести число n (Integer) (n ≥ 8). Создать массив случайных чисел (Single): X(n) из диапазона [–5.3, 6.2]. Вычислить $M = \frac{1}{n} \sum_{i=1}^{n} x_i$ – математическое ожидание и дисперсию массива случайных величин $D = \frac{1}{n} \sum_{i=1}^{n} (x_i - M)^2$.

Вариант № 9. Ввести числа n (Integer) и q (Single) (n ≥ 7; q < n). Создать массив случайных чисел (Single) P(n) из диапазона [-2.9, 5.7]. Выделить из вектора P(n) вектор R(m) (m ≤ n) по правилу: компонента вектора P является компонентой вектора R, если квадратное уравнение $x^2 - 2P_i x + q = 0$ имеет вещественные корни.

Вариант № 10. Ввести числа n (Integer), a, x_0 и x_n (Single) (n ≥ 20; -0.3 ≤ a ≤ 0; 0 ≤ x_0 < x_n ≤ 5). Вычислить $J = h \sum_{i=1}^{n} y_i$, где h = (x_n – x_0)/n; y(x) =

$e^{-ax}\cos^2 x$; y_i – значение y(x) в точке $x_i = x_{i-1} + h$, i = 1,…,n.

Вариант № 11. Ввести число n (Integer) (n ≥ 6). Создать массив случайных чисел (Single): X(n) из диапазона [0, 2.9]. Вычислить $S = \sum_{k=2}^{n+1} a_k$, где $a_k = \frac{1}{(2k)!}\sum_{i=1}^{k-1} x_i^2$, k = 2,…, n + 1.

Вариант № 12. Ввести число n (Integer) (n ≥ 6). Создать массивы (Single): X(n) – равномерно распределенных чисел из диапазона [0, 2.5]; Y(n) – функция от x, причем $y_i = x_i^3$ при i = 1, …, n. Составить процедуру нахождения вектора P(n), элементы которого равны производным функции Y(x) в точках x_1, x_2, …, x_n, вычисленным по формулам численного дифференцирования: $P_1 = y'(x)_{x_1} = y'_1 = \frac{-3y_1 + 4y_2 - y_3}{2(x_2 - x_1)}$;

$P_n = y'(x)_{x_n} = y'_n = \frac{y_{n-2} - 4y_{n-1} + y_n}{2(x_n - x_{n-1})}$; $P_i = y'(x)_{x_i} = y'_i = \frac{y_{i+1} - y_{i-1}}{2(x_{i+1} - x_i)}$; i = 2, …, n - 1. Функция Y(x) задана таблично, т.е. значениям аргумента x_1, x_2, …, x_n соответствуют значения функции y_1, y_2, …, y_n.

Вариант № 13. Ввести число n (Integer) (n ≥ 6). Создать массивы случайных чисел (Single): A(n) из диапазона [–6.4, 4.1]; B(n) из диапазона [-42.1, 80.1]. Вычислить угол φ между векторами A = {a_i} и B = {b_i}, i = 1, …, n по формуле $\cos\varphi = \sum_{i=1}^{n} a_i b_i \Big/ \sqrt{\sum_{i=1}^{n} a_i^2 \sum_{i=1}^{n} b_i^2}$.

Вариант № 14. Ввести целое число n (Integer) (n = 3) и малое число $\varepsilon \le 10^{-5}$. Создать массив случайных чисел (Single): A(n) из диапазона [–0.9, 0.9]. Вычислить $Y = f(a_1) + f(a_2) - 2\cdot f^2(a_3)$. Значение f(x) вычислять с точностью ε по формуле $f(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots + (-1)^{n+1}\frac{x^{2n+1}}{(2n+1)!}$.

Вариант № 15. Ввести число n (Integer) (n ≥ 10). Создать массив случайных чисел (Single): X(n) из диапазона [–0.9, 3.2]. Составить процедуру для вычисления вектора Y(n), компоненты которого $Y_i = 0.26\cdot\ln(1 + i^{1/2})$; i = 1, …, n, и присвоения переменной Z значения TRUE, если $S = (X_1Y_n)^2 + (X_2Y_{n-1})^2 + \ldots + (X_nY_1)^2$ принадлежит отрезку [0, 1] и значение FALSE – в противном случае.

2.2.4 Содержание отчета

В отчете должны быть представлены:

1. Цель работы.
2. Основные сведения о теме работы.
3. Алгоритм разработанного проекта.

4. Перечень форм и элементов управления.
5. Процедуры обработки данных.
6. Исходные данные и результаты их обработки.

2.3 Контрольные вопросы

1. Свойства элементов управления и формы определяют…
2. События элементов управления и формы это…
3. Методы элементов управления и формы это…
4. Перечислите параметры событий связанных с мышкой.
5. Перечислите параметры событий связанных с клавиатурой.
6. Как создать массив элементов управления на этапе разработки проекта?
7. Опишите последовательность операций при создании массивов элементов управления во время выполнения проекта.

ЛАБОРАТОРНАЯ РОБОТА 3

РАБОТА С ГРАФИКОЙ В VISUAL BASIC. ЭКРАН, КЛАВИАТУРА И МЫШЬ

Цель работы – Научиться работать с клавиатурой и мышью в Visual Basic 6.0. Создать проект на Visual Basic 6.0, содержащий формы, различные элементы управления, процедуры и функции. Научиться обрабатывать графические, текстовые и математические данные.

3.1 Общие сведения

В этой лабораторной работе изучаются различные возможности ввода и вывода информации в Visual Basic. В частности, рассматриваются вопросы работы с мышью и клавиатурой, ввод и вывод информации на экран и принтер. Рассмотрим системный объект, представляющий экран Windows, а также диалоговые окна для ввода и вывода.

3.1.1 Объект Screen

В Visual Basic объект Screen представляет собой весь экран Windows. В Visual Basic, наряду с объектом Screen существует и свойство Screen объекта Global.

Поскольку имеется только единственный экран Windows, переменные типа Screen обычно не объявляются, а используется системный объект Screen. Однако если такая необходимость возникла, то переменная

типа Screen перед использованием должна не только объявляться, но и содержать действительную ссылку на реальный объект. Например:

```
Dim scr As Screen
Set scr = Screen
```

Основные свойства обеъкта Screen приведены в таблице 3.1.

Таблица 3.1 - Свойства объекта Screen

Свойство	Описание
ActiveControl	Активный элемент управления
ActiveForm	Активная форма
FontCount	Количество доступных шрифтов
Fonts()	Возвращает имена всех доступных шрифтов
Height	Высота экрана
MouseIcon	Позволяет установить пользовательскую пиктограмму для курсора
MousePointer	Тип указателя мыши
TwipsPerPixelX	Количество твипов (twips) в пикселе (разрешение по горизонтали)
TwipsPerPixelY	Количество твипов (twips) в пикселе (разрешение по вертикали)
Width	Ширина экрана

3.1.1.1 MouseIcon/MousePointer

Свойство MousePointer определяет внешний вид курсора мыши. Можно использовать либо стандартные курсоры Windows, либо создавать собственные.

Если для свойства MousePointer установлено значение vbCustom, то для отображения курсора мыши будет использоваться пиктограмма, определяемая свойством MouseIcon:

```
Screen.MouseIcon.LoadPicture ("c:\Pointer.CUR")
Screen.MousePointer = vbCustom
```

При этом допускается использовать файлы пиктограмм (*.ICO) и файлы курсоров (*.CUR). Анимационный курсор (*.ANI) свойством MousePointer не поддерживается.

Свойства MouseIcon и MousePointer используются для информирования пользователя об определенном состоянии системы. Так, в начале продолжительной процедуры можно установить значение свойства MousePointer равным vbHourglass ("Песочные часы"). Это информирует пользователя о выполнении длительной операции и о том, что ему следует дождаться ее окончания. Однако в конце процедуры свойство должно быть изменено, о чем не следует забывать.

Свойство MousePointer имеет не только объект Screen. Есть оно и у формы. Если процесс затрагивает операционную систему, то соответ-

ствующий указатель мыши должен устанавливаться для объекта Screen. Если же операция не выходит за рамки приложения, то курсор изменяется в форме.

В некоторых случаях с помощью этих двух свойств почти для каждого элемента управления можно предусмотреть собственный курсор мыши.

3.1.1.2 Height/Width

Свойства Height и Width используются для определения размера объекта Screen, т.е. размера экрана. В качестве единицы измерения используется твип. Эта единица измерения, общепринятая в Windows, связана с выводом информации на печать и подробно описывается ниже:

```
HeightInTwips = Screen.Height
WidthInTwips = Screen.Width
HeightInPixel = Screen.Height / Screen.TwipsPerPixelX
WidthInPixel = Screen.Width / Screen.TwipsPerPixelY
```

Для того чтобы пересчитать размер в твипах в пиксели, используются свойства TwipsPerPixelX и TwipsPerPixelY.

3.1.1.3 ActiveControl

При написании кода, который может быть использован повторно, не следует указывать явные имена элементов управления. Если при этом важно определить, для какого элемента управления установлен фокус, то для этого используют свойство ActiveControl объекта Screen.

Например, в форме находится два текстовых поля и с помощью кнопки нужно удалить выделенный текст только в активном поле:

```
Private Sub cmdDelete_Click()
Text1.SelText = ""
Text2.SelText = ""
End Sub
```

Приведенное в примере решение некорректно, так как удаляется выделенный текст в обоих текстовых полях. Для корректного решения поставленной задачи следует заменить конкретное имя на ActiveControl:

```
Private Sub cmdDelete_Click()
Screen.ActiveControl.SelText = ""
End Sub
```

С помощью этой процедуры удаляется выделенный текст только в активном элементе управления.

Могут возникнуть проблемы и при использовании элементов управления различных типов. Необходимо обеспечить, чтобы используемые свойства для ActiveControl фактически имелись в активном элементе управления. Например, процедура cmdDelete_Click() для кнопки не будет функционировать, так как при щелчке на кнопке фокус установится именно для этой кнопки, а поскольку она не имеет свойства SelText, то последует сообщение об ошибке: "Объект не поддерживает это свойство или

метод". Если же осуществить вызов процедуры из меню, то проблема будет решена, так как команды меню не могут принимать фокус:

```
Private Sub mnuDelete_Click()
Screen.ActiveControl.SelText = ""
End Sub
```

При разработке приложения следует иметь в виду, что в форме могут быть элементы управления различных типов, например TextBox и PictureBox. Поэтому если при нахождении курсора в текстовом поле попытаться изменить свойство ActiveControl.Picture, возникает ошибка.

3.1.1.4 TypeOf

Использование ключевого слова TypeOf позволяет проверить тип активного элемента управления. Обычно его используют вместе с оператором If...Then:

```
Private Sub mnuDelete_Click()
If TypeOf Screen.ActiveControl Is TextBox Then
Screen.ActiveControl.Seltext = ""
ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
Screen.ActiveControl.Picture = LoadPicture()
End If
End Sub
```

В этом примере проверяется, какой тип имеет активный элемент управления — TextBox или PictureBox, и в зависимости от результатов проверки удаляется его содержимое или загружается изображение.

Типы элементов управления указываются также в окне свойств рядом с именем, элемента управления.

3.1.1.5 ActiveForm

Свойство ActiveForm действует аналогично ActiveControl, но содержит ссылку на активную форму:

```
Screen.ActiveForm.ActiveControl.SelText = ""
```

3.1.1.6 Окно InputBox

Текстовое поле требуется для ввода разнообразной информации. Но иногда требуется ввести только краткую информацию, например значения даты или времени. Создавать для этого отдельное текстовое поле или форму нерационально. Для ввода небольших фрагментов текста Visual Basic предлагает функцию InputBox.

Окно InputBox состоит из четырех элементов:

- строка заголовка;
- приглашение к вводу (Prompt);
- поле ввода со значением, предлагаемым по умолчанию;
- две кнопки (OK и Cancel).

Функция вызова окна InputBox имеет следующий синтаксис с соответствующими именованными аргументами:

```
Возвращаемое_значение = InputBox(Prompt [, title] [, default] [, xpos] [, ypos] [, helpfile, context])
```

Параметр Prompt определяет текст, отображающийся в диалоговом окне как приглашение. Title отвечает за надпись заголовка; если этот параметр не указан, то отображается название приложения. Параметр Default определяет значение по умолчанию, отображаемое в строке ввода. Параметры xpos и ypos указывают координаты верхнего левого угла окна. По умолчанию окно отображается посредине экрана. Параметры xpos и ypos нужно использовать совместно:

```
StrReturn = InputBox("Вопрос", "Заголов.", "Зад. значение")
If StrReturn = "" Then Exit Sub
```

Функция InputBox возвращает строку, введенную пользователем. При нажатии кнопки Cancel возвращается пустая строка.

Функция InputBox имеет еще два необязательных параметра — HelpFile и Context, которые позволяют открывать определенные файлы справочной системы.

Это диалоговое окно почти не встречается в стандартных приложениях Windows, так как выглядит не слишком привлекательно. Вам также следует отказаться от использования InputBox в готовом приложении. Однако это окно имеет смысл использовать на этапе проектирования, если вам необходимо временно ввести информацию с помощью InputBox. Большим преимуществом является то, что вызов этой функции легко удалить из программы, так как он содержит только одну строку. Но переменная, содержащая возвращаемое значение, и проверка этого значения нужны и в окончательной редакции программы.

3.1.1.7 Окно MessageBox

Для вывода различных сообщений из работающего приложения имеется окно, подобное InputBox, – MessageBox. Почти все приложения Windows используют MessageBox, так как этот компонент входит в состав Windows, а Visual Basic только предоставляет возможность его вызова.

Вид окна MessageBox может быть различным, но в его состав всегда входят:

- текст сообщения;
- заголовок;
- пиктограмма;
- набор кнопок.

MessageBox можно вызывать как процедуру и как функцию.

```
MsgBox Prompt [,Buttons ] [,Title ] [, Helpfile, Context]
Возвращаемое_значение = MsgBox (Prompt[,Buttons ] [,Title ]
[ ,Helpfile, Context])
```

Примеры:

```
MsgBox "Здравствуй, пользователь!", vbExclamation,
"Приветствие"
ret = MsgBox ("Закончить?" ,vbCritical, "End")
```

Параметры Prompt и Title не требуют пояснения. Параметр Buttons определяет внешний вид MessageBox. Значение параметра формируется из нескольких частей, которые можно складывать:

```
Buttons = Button + Icon + Default + Modal + Extras + Extras
```

Для категорий параметра Button, Icon, Default и Modal можно использовать только одну из допустимых констант. А для категории Extras допускается применение комбинации значений (табл. 3.1).

Например, вы хотите, чтобы в окне MessageBox отображались вопросительный знак, кнопки «Да» и «Нет», и при этом кнопка «Нет» была кнопкой по умолчанию. Кроме того, окно должно быть модальным окном приложения, т.е. выполнение приложения продолжается только после закрытия окна. Для этого следует просуммировать соответствующие значения. Так как всем символьным константам Visual Basic соответствуют числовые значения, код, реализующий перечисленные требования, мог бы выглядеть следующим образом:

```
TYPE =4+32+256+0
MsgBox "Сообщение", TYPE, "Заглавие"
```

Однако такой вариант вызова, использующий числовые значения, без комментариев понять трудно, поэтому лучше применять соответствующие константы:

```
TYPE = vbYesNo Or vbQuestion Or vbDefaultButton2 Or
vbApplicationModal
MsgBox "Сообщение", TYPE, "Title"
```

Обратите внимание, что корректно суммирование значений констант выполняется оператором Or. Однако можно применить и операцию арифметического суммирования. В этом случае следует использовать только одну константу из категории Button, Icon, Default и Modal.

После нажатия кнопки пользователем следует проанализировать возвращаемое функцией значение:

```
ret = MsgBox("Хотите?", vbYesNo Or vbQuestion, strUserName)
```

Возвращаемое функцией значение позволяет определить, какую кнопку нажал пользователь (табл. 3.2).

Таблица 3.2 - Значения, возвращаемые функцией MsgBox

Константа	Значение	Нажата кнопка
vbOK	1	OK
vbCancel	2	Отмена
vbAbort	3	Стоп
vbRetry	4	Повторить
vbIgnore	5	Пропустить
vbYes	6	Да
vbNo	7	Нет

Таблица 3.1 - Константы функции MsgBox

Константа	Значение	Описание
Категория: Button		
VbOKOnly	0	Только кнопка OK
VbOKCancel	1	Кнопки OK и Отмена
VbAbortRetryIgnore	2	Кнопки Стоп, Повторить и Пропустить
VbYesNoCancel	3	Кнопки Да, Нет и Отмена
VbYesNo	4	Кнопки Да и Нет
VbRetryCancel	5	Кнопки Повторить и Отмена
Категория: Icon		
VbCritical	16	Отображает пиктограмму Critical Message
VbQuestion	32	Отображает пиктограмму Warning Query
VbExclamation	48	Отображает пиктограмму Warning Message
VbInformation	64	Отображает пиктограмму Information Message
Категория: Default		
VbDefaultButton1	0	По умолчанию активна первая кнопка
vbDefaultButton2	256	По умолчанию активна вторая кнопка
vbDefaultButton3	512	По умолчанию активна третья кнопка
vbDefaultButton4	768	По умолчанию активна четвертая кнопка
Категория: Modal		
VbApplicationModal	0	Модальное диалоговое окно приложения
VbSystemModal	4096	Мод. диал. окно системы
Категория: Extras		
VbMsgBoxHelpButton	16384	Дополнительная кнопка для справки
VbMsgBoxSetForeground	65536	Отображение диалогового окна в фоновом режиме
VbMsgBoxRight	524288	Текст выровнен по правому краю
VbMsgBoxRtlReading	1048576	Текст отображается справа налево (еврейский, арабский)

Теперь становится понятен различный синтаксис вызовов функции.

Синтаксис процедуры предназначен для окна сообщения (MessageBox) с одной кнопкой, так как в этом случае возвращаемое зна-

чение не столь важно. Однако при запросах требуется вызывать MsgBox как функцию, так как возвращаемое значение используется для определения кнопки, нажатой пользователем:

```
   Select Case MsgBox ("Вопрос", vbAbortRetryIgnore Or
vbQuestion, "Заглавие")
   Case vbAbort
   Call Abort
   Case vbRetry
   Call Retry
   Case vbIgnore
   Call Ignore
   End Select
```

Кнопка Default (кнопка, которая является активной по умолчанию) должна определяться таким образом, чтобы при случайном нажатии клавиши [Enter] действия, реализуемые в процедуре обработки нажатия кнопки по умолчанию, не могли нанести большого вреда. Например, не следует в окне с вопросом "Удалить все внесенные изменения?" (Да/Нет) назначать кнопку Да кнопкой по умолчанию.

Окно MessageBox используется также для того, чтобы сообщить пользователю о результатах выполнения программой определенных действий, например о том, что "Регистрация в баз данных не удалась". В этом случае следует выводить только те сообщения, которые действительно важны (например, при невыполнении важной операции).

3.1.2 Клавиатура

Основными устройствами, с помощью которых пользователь взаимодействует с приложением и операционной системой, являются клавиатура и мышь. Клавиатура, как правило, используется для ввода данных, а мышь — для управления рабочей средой.

Основной проблемой при использовании клавиатуры является то, что для обслуживания множества элементов управления имеется только одна клавиатура. Поэтому система должна определить, какому элементу управления передается вводимая с клавиатуры информация. В связи с этим в Windows используется понятие "фокус". Если говорят, что элемент управления имеет фокус, это означает, что Ввод информации с клавиатуры относится к этому элементу.

Элемент, имеющий фокус, можно распознать по различным визуальным признакам. Чаще всего это мигающий курсор или пунктирная рамка.

Фокус устанавливается в результате щелчка на элементе управления, после нажатия клавиши [Tab], или в результате выполнения программного кода (…SetFocus). После этого элемент управления может принимать информацию от клавиатуры.

Несмотря на то, что Windows ориентирована на работу с мышью, диалоговые окна и меню следует разрабатывать с учетом возможности использования клавиатуры. Рабочий процесс можно ускорить, если при длительном вводе с клавиатуры предоставить возможность перемещать фокус также с помощью клавиатуры, без использования мыши.

В Windows почти во всех надписях на элементах управления есть подчеркнутые символы. С помощью клавиши [Alt] и этого символа можно переместить фокус на этот элемент управления или выполнять соответствующее действие. Например, посредством нажатия клавиш [Alt+F] можно вызвать команду меню File, не пользуясь мышью.

В Visual Basic при помощи свойства Caption можно задать горячую клавишу путем установки перед нужной буквой символа амперсанд (коммерческое И — &), воспользовавшись окном свойств:

```
mnuFile.Caption = "&File"
```

Для того чтобы использовать амперсанд в надписи, нужно поставить два таких знака.

```
IblDragDrop.Caption = "Drags&&Drop"
```

Можно также отобразить символ & в текст Caption, не прибегая к указанию двух знаков амперсанта. Для этого надо установить значение свойства UseMnemonic равным False.

Перемещать фокус можно также с помощью клавиши [Tab]. При нажатии этой клавиши происходит переход к следующему элементу управления формы, имеющему на единицу большее значение свойства TabIndex, при нажатии [Shift+Tab] - к предыдущему. Порядок перехода между элементами управления в Visual Basic по умолчанию совпадает с порядком добавления элементов в форму, но при необходимости его можно изменить.

Все элементы управления, для которых возможно установить фокус, обладают свойством TabIndex. Элемент управления, у которого свойство TabIndex равно 0, получает фокус сразу после загрузки формы. После каждого нажатия клавиши [Tab] фокус переходит к элементу со следующим значением TabIndex. Последовательность переходов при нажатии клавиши [Tab] следует создавать таким образом, чтобы пользователь мог проследить ее логически, а не искать активный элемент управления. Для этого следует корректно присваивать значения свойству TabIndex, помня, что Visual Basic препятствует заданию одинакового индекса TabIndex для двух элементов управления.

Элемент управления Надпись (Label) также имеет свойство TabIndex, хотя и не может принимать фокус. Благодаря этому свойству, можно назначать горячие клавиши другим элементам управления, не имеющим собственного свойства Caption (например, текстовое окно). Для

этого значение свойства TabIndex надписи должно быть на единицу меньше, чем значение такого же свойства самого элемента управления.

Для удаления элемента управления из последовательности переходов табуляции нужно установить равным False значение его свойства TabStop. В этом случае фокус элементу управления можно установить только с помощью мыши.

Ввод с клавиатуры может обрабатываться не только Windows, но и непосредственно элементами управления. Для этого необходимо обрабатывать события: KeyDown, Keypress и KeyUp. При нажатии клавиши для активного элемента управления генерируются соответствующие события.

В качестве параметра процедуре обработки события Keypress передается переменная KeyAscII, содержащая ANSI код нажатой клавиши. Такое имя (KeyAscII) немного вводит в заблуждение, но различие между ASCII и ANSI кодом клавиши важно только для символов с кодом больше 127.

```
Private Sub Text1_KeyPress (KeyAscII As Integer)
MsgBox KeyAscII
End Sub
```

В примере код нажатой клавиши выводится в окне сообщений, если фокус принадлежит элементу управления с именем Text1. Если нажата, например, клавиша [A], то в окне сообщений выводится значение 65. Обратите внимание, что KeyAscII — это не свойство, а переменная. Следовательно, выражение вида Text1.KeyAscII не может использоваться в коде.

Значение переменной KeyAscII можно не только считывать, но и устанавливать. Поэтому возможна замена введенного с клавиатуры символа другим:

```
KeyAscII = Asc(UCase(Chr(KeyAscII)))
```

В этом примере функция Chr преобразует ANSI код клавиши в соответствующий текстовый символ. С помощью функции UCase этот символ преобразуется в прописной, а затем функция Asc преобразует его в ANSI код. Это значение опять присваивается переменной KeyAscII, и соответствующий символ появляется в элемент управления. Таким образом, все вводимые с клавиатуры символы преобразуются в прописные.

Событие KeyPress вызывается только при нажатии клавиш, имеющих ANSI код. А нажатие, например, клавиш управления курсором или функциональных это событие не вызывает. Для обработки нажатия этих клавиш следует использовать события KeyUp и KeyDown.

```
Private Sub Text1_KeyDown(KeyCode As Integer, Shift As
Integer)
MsgBox KeyCode
End Sub
```

Процедурам обработки этих событий передаются две переменные: KeyCode и Shift. Параметр KeyCode содержит клавиатурный код нажатой

клавиши, а Shift позволяет определить состояние клавиш [Shift], [Ctrl] и [Alt] (табл. 3.3).

Таблица 3.3 - Константы параметра Shift

Константа	Значение	Описание
VbShiftMask	1	Нажата клавиша [Shift]
VbCtrlMask	2	Нажата клавиша [Ctrl]
VbAltMask	4	Нажата клавиша [Alt]

Значения кодов отдельных клавиш можно добавить в программу как константы, воспользовавшись каталогом объектов. В качестве констант для аргумента Shift используются вышеуказанные значения.

При работе с клавиатурой события происходят в такой последовательности: KeyDown, KeyPress, KeyUp.

Обычно события клавиатуры генерируются для активного элемента управления. Но нажатие некоторых клавиш, например функциональных, должна обрабатывать форма, а не активный элемент управления. Поэтому форма также может обрабатывать события KeyDown, KeyPress и KeyUp, но для этого необходимо установить равным True значение свойства формы KeyPreview. В этом случае все события клавиатуры сначала будет обрабатывать форма, а лишь затем активный элемент управления.

```
Private Sub Form1_KeyDown(KeyCode As Integer, Shift As
Integer)
If KeyCode = vbKeyF4 Then
MsgBox "Нажата F4"
Elself KeyCode = vbKeyF4 And Shift = VbShiftMask Then
MsgBox "Нажаты F4 и Shift"
End If
End Sub
```

В данном примере значение свойства KeyPreview формы равно True, поэтому все нажатия клавиш сначала будет обрабатывать форма. Если нажать клавишу [F4] или [Shift+F4], то процедура Form1_KeyDown проанализирует нажатие клавиши и выведет соответствующее сообщение независимо от того, для какого элемента управления установлен фокус.

Visual Basic позволяет не только обрабатывать реальные нажатия клавиш клавиатуры, но и имитировать такие нажатия. С помощью оператора SendKeys код, имитирующий нажатие клавиши, записывается непосредственно в буфер клавиатуры. Система при этом не отличает такой ввод от "настоящего" ввода.

```
SendKeys Ctrl [, Wait]
```

С помощью именованного параметра Wait можно определить режим ожидания обработки имитации нажатия клавиши. Если значение параметра равно False (по умолчанию), то управление возвращается проце-

дуре немедленно после посылки сообщения о нажатии клавиш. Если же его значение равно True, сообщение должно быть обработано, прежде чем управление будет передано процедуре, пославшей код клавиш.

Реальный ввод с клавиатуры либо имитация нажатия клавиш с помощью оператора SendKeys всегда передается активному элементу управления. Для того чтобы вводимые символы передавались нужному приложению, необходимо сначала передать ему фокус.

```
SendKeys "+1F1"
```

Эта строка имитирует нажатие клавиш [Shift+Fl]. В Visual Basic для передачи фокуса приложению существует оператор AppActivate:

```
AppActivate Title [, Wait]
```

Параметр Title оператора — это текст строки заголовка (значение свойства Caption) приложения, которое нужно активировать. Текст должен слово в слово совпадать со значением свойства Caption. При этом не имеет значения вид написания — прописными буквами или строчными.

```
Private Sub Command1_Click()
ret = Shell ("calc.exe", vbNormalFocus)
AppActivate "Калькулятор", False
SendKeys "1{+}2 = ^C % {F4}", True
Text1.Text = Clipboard.GetText()
End Sub
```

В данном примере запускается стандартная программа — калькулятор Windows. Затем суммируются числа 1 и 2, результат вычисления копируется в буфер обмена, и калькулятор закрывается.

3.1.3 Мышь

Вторым важным устройством ввода в Windows является мышь. Все действия, выполняемые мышью (щелчки, перемещения) также можно анализировать, для чего генерируется ряд событий (табл. 3.4).

Таблица 3.4 - События мыши

Событие	Описание
Click	Одинарный щелчок мыши
DblClick	Двойной щелчок
MouseDown	Нажатие кнопки мыши
MouseUp	Отпускание кнопки мыши
MouseMove	Перемещение указателя мыши
DragDrop	Отпускание кнопки мыши в режиме Drag&Drop (перетаскивание)
DragOver	Перемещение мыши в режиме Drag&Drop (перетаскивание)

Самым важным событием мыши является Click. Оно наступает, когда пользователь щелкает мышью на элементе управления. Однако для

Windows Click состоит из двух событий: нажатия и освобождения кнопки мыши. Событие Click может наступить и при изменении значения определенного свойства элемента управления. Например, для ListBox событие Click вызывается и тогда, когда с помощью клавиш перемещения курсора выбирается новый элемент списка. Многие элементы управления могут обрабатывать событие Click: Form, CommandButton, Label, PictureBox, OptionButton, ListBox, ComboBox и т.д.

```
Private Sub Comniand1_Click ()
```

Событие Click можно также вызывать программно — для этого следует изменить свойство Value. Событие DblClick вызывается, если выполняются два щелчка за промежуток времени, определяемый значением параметра Скорость двойного нажатия, который устанавливается на вкладке Кнопки мыши диалогового окна Свойства: Мышь. Если в этом промежутке времени не регистрируются два щелчка, генерируется не событие DblClick, а два последовательных события Click. С помощью двух последовательных нажатий кнопки мыши, вызывающих события Click, можно выделить элемент списка, однако только после двойного щелчка, вызвавшего событие DblClick, будет выполнена соответствующая операция, например открытие файла или запуск программы. Если требуется более детальный анализ нажатия кнопки мыши, то следует обрабатывать отдельно нажатие и отпускание кнопки мыши. После нажатия кнопки мыши на активном элементе управления генерируется событие MouseDown. При этом процедуре обработки этого события передаются четыре параметра: Button, Shift, X и Y. Событие MouseUp функционирует аналогично MouseDown, но наступает при отпускании кнопки мыши.

```
Private Sub Form1_MouseDown (Button As Integer, Shift
As Integer, X As Single, _ Y As Single)
Select Case Button
Case vbLeftButton
MsgBox "Нажата левая кнопка"
Case vbRightButton
MsgBox "Нажата правая кнопка"
Case vbMiddleButton
MsgBox "Нажата средняя кнопка"
End Select
End Sub
```

Параметр Button определяет состояние кнопки мыши. Он может принимать одно из следующих значений: vbLeftButton, vbRightButton или, если имеется третья кнопка мыши, vbMiddleButton. Параметр Shift позволяет определить, были ли нажаты клавиши [Shift] (vbShiftMask), [Ctrl] (vbCtrlMask) и [Alt] (vbAltMask) при нажатии или отпускании кнопки мыши. Переменные X и Y содержат значения координат текущей позиции курсора мыши на элементе управления.

При перемещении мыши наступает множество событий MouseMove. Процедуре обработки этого события передаются такие же переменные, как и при обработке событий MouseUp и MouseDown. Аргумент Button позволяет определить, какая кнопка мыши была нажата не только при обработке событий MouseUp и MouseDown, но и при обработке события MouseMove. Таким образом, можно узнать была ли нажата кнопка во время перемещения мыши.

Обрабатывая события MouseUp, MouseDown и MouseMove, можно создать простейшую программу рисования. При нажатии левой кнопки мыши с помощью метода PSet ставится точка в том месте формы, в котором в данный момент находится указатель мыши. После этого курсор принимает форму перекрестия:

```
Private Sub Form1_MouseDown(Button As Integer, Shift
As Integer, X As Single, Y As Single)
If Button = vbLeftButton Then
Form1.MousePointer = vbCrosshair
Form1. PSet (X,Y)
End If
End Sub
```

При перемещении мыши наступает последовательно несколько событий MouseMove. Если при перемещении была нажата левая кнопка мыши, то точки, в которых эти события произошли, соединяются линией с помощью метода Line:

```
Private Sub Form1_MouseMove(Button As Integer, Shift
As Integer, X As Single, Y As Single)
If Button =vbLeftButton Then
Form1.Line -(X,Y)
End If
End Sub
```

При отпускании кнопки мыши курсор принимает первоначальный вид:

```
Private Sub Form1_MouseUp(Button As Integer, Shift As
Integer, X As Single, Y As Single)
Form1.MousePointer = vbDefault
End Sub
```

Этот небольшой пример демонстрирует возможности использования событий мыши.

Последовательность генерации событий мыши зависит от типа элемента управления, для которого эти события генерируются. Например, для элементов управления ListBox и CommandButton события наступают в следующем порядке: MouseDown, Click, MouseUp. Для FileListBox, Label или PictureBox события наступают в другой последовательности: MouseDown, MouseUp, Click. При двойном щелчке события происходят в последовательности: MouseDown, MouseUp, Click, DblClick, MouseUp.

3.2 Порядок выполнения работы

3.2.1 Задания для выполнения работы

1. Включить компьютер.
2. Запустить на выполнение программу «Visual Basic 6.0».
3. Открыть проект Standard.EXE.
4. Разработать форму, содержащую несколько элементов управления.
5. Написать коды обработки событий, связанных с элементами управления. Задание для каждого студента приведено ниже.
6. Запустить программу на выполнение.

3.2.2 Порядок выполнения

1. Разработать проект для реализации алгоритма, приведенного в вариантах заданий.
2. Проект должен содержать несколько форм и программный модуль.
3. Формы должны содержать необходимые элементы управления для реализации алгоритма и облегчения работы пользователя с программой. Для каждого элемента управления используйте всплывающую подсказку.
4. Формы должны содержать меню, панель инструментов и строку состояния (Для этого подключите к разрабатываемому проекту компонент Microsoft Windows Common Control 6.0 (SP…) (Проект/Компоненты)). На каждом шаге в строку состояния должна выводиться подсказка о дальнейших действиях пользователя. ProgressBar должен отражать ход выполнения вычислительного процесса и вывода графики в PictureBox.
5. При необходимости для ввода данных используйте окно InputBox, а окно MessageBox используйте для вывода сообщений и/или для выбора ветви хода вычислительного процесса.
6. Используйте события мыши для выделения и/или изменения вида объектов, над которыми перемещается указатель мыши.
7. Программный модуль должен содержать процедуры, выполняющие необходимые действия, обусловленные вариантом задания.
8. При старте проекта ввести необходимые данные и провести их обработку.
9. Исходные данные и результаты обработки необходимо занести в отчет (сохранить изображение в файле и вставить его затем в отчет).
10. Для выбора имени файла при вводе и выводе данных в файл используйте окно Common Dialog. Для этого подключите к разрабатываемому проекту компонент Microsoft Common Dialog Control 6.0. (Проект/Компоненты)

11. Алгоритм построения изображения трехмерных геометрических объектов необходимо занести в отчет.

3.2.3 Варианты заданий

Общее задание для всех:

Дать двумерные изображения трехмерных геометрических объектов, изображенных с помощью диметрии. При изображении тела в диметрии масштаб по осям z и y одинаков, а по оси x в 2 раза меньше; угол между осями y и z равен 90°; угол между осями x и y составляет 135°, как показано на рисунке.

Для построения трехмерного изображения какого-либо объекта необходимо вначале определить трехмерные координаты его характерных точек XYZ(0 To 2), затем вычислить его экранные координаты XYZS(0 To 1), и только после этого выводить соответствующие точки, линии и дуги (эллипсы) в PictureBox.

Изображения линий и плоских граней фигур должны быть окрашены в различные цвета. Предусмотреть перекрытие одних фигур другими и учесть их видимость.

Коэффициенты уравнений, описывающих линии, плоскости и поверхности, а также трехмерные координаты соответствующих точек вводить из текстовых окон. Используя уравнения линий и плоскостей, определить соответствующие точки, коэффициенты в уравнениях линий и плоскостей и построить точки и линии пересечения соответствующих объектов. Необходимые уравнения можно взять из справочника по математике. При разработке алгоритма программы необходимо получить решения соответствующих уравнений (систем уравнений) пересечения линий и/или плоскостей.

Совет: Вспомните начертательную и аналитическую (один из разделов высшей математики) геометрию перед разработкой алгоритма.

Вариант № 1. Ромб и окружность, диаметром которой является сторона ромба. Вводимые параметры: диагонали ромба, координаты его центра. Вычисляемые параметры и координаты: координаты вершин ромба, радиус окружности, координаты ее центра.

Вариант № 2. Окружность, диаметром которой является высота равнобедренного треугольника, отношение стороны которого к основанию равно 1 : 2. Вводимые параметры: стороны треугольника, координаты одной из его вершин. Вычисляемые параметры и координаты: координаты вершин и высота треугольника, радиус окружности, координаты ее центра.

Вариант № 3. Равнобедренный прямоугольный треугольник, на катете которого как на диаметре построена окружность. Вводимые параметры: стороны треугольника, координаты одной из его вершин. Вычисляе-

мые параметры и координаты: координаты вершин треугольника, радиус окружности, координаты ее центра.

Вариант № 4. Равнобедренный треугольник, вписанный в квадрат, причем основание треугольника равно половине диагонали квадрата, а его высота – ¾ диагонали квадрата. Вводимые параметры: сторона квадрата, координаты одной из его вершин. Вычисляемые параметры и координаты: координаты вершин и длины сторон треугольника.

Вариант № 5. Квадрат, вписанный в окружность. Вводимые параметры: радиус окружности, координаты ее центра, координаты одной из вершин квадрата. Вычисляемые параметры и координаты: координаты вершин квадрата, сторона квадрата.

Вариант № 6. Окружность, вписанная в квадрат. Вводимые параметры: радиус окружности, координаты ее центра, координаты одной из вершин квадрата. Вычисляемые параметры и координаты: координаты вершин квадрата, сторона квадрата.

Вариант № 7. Прямоугольный треугольник, вписанный в окружность и опирающийся на ее диаметр. Вводимые параметры: радиус окружности, координаты ее центра, координаты одной из вершин треугольника. Вычисляемые параметры и координаты: координаты вершин треугольника, стороны треугольника.

Вариант № 8. Окружность и ромб, меньшая диагональ которого является диаметром окружности. Вводимые параметры: диагонали ромба, координаты его центра. Вычисляемые параметры и координаты: радиус окружности, координаты ее центра.

Вариант № 9. Ромб и окружность, диаметром которой является большая диагональ ромба. Вводимые параметры: диагонали ромба, координаты его центра. Вычисляемые параметры и координаты: радиус окружности, координаты ее центра.

Вариант № 10. Правильный шестиугольник, вписанный в окружность. Вводимые параметры: радиус окружности, координаты ее центра. Вычисляемые параметры и координаты: сторона шестиугольника, координаты его вершин.

Вариант № 11. Окружность, вписанная в правильный шестиугольник. Вводимые параметры: сторона шестиугольника, координаты его вершин. Вычисляемые параметры и координаты: радиус окружности, координаты ее центра.

Вариант № 12. Окружность, диаметром которой является основание равнобедренного треугольника, высота которого в два раза больше основания. Вводимые параметры: основание треугольника, координаты одной из его вершин. Вычисляемые параметры и координаты: высота треугольника, координаты остальных его вершин, радиус окружности, координаты ее центра.

Вариант № 13. Прямоугольную трапецию, вписанную в окружность. Вводимые параметры: радиус окружности, координаты ее центра, высота трапеции, координаты одной из её вершин. Вычисляемые параметры и координаты: координаты остальных вершин трапеции, длины её сторон.

Вариант № 14. Прямоугольную трапецию на меньшей диагонали, которой построена как на диаметре окружность. Вводимые параметры: высота и основание и противоположная ему сторона трапеции, координаты одной из её вершин. Вычисляемые параметры и координаты: координаты остальных вершин трапеции, длины её сторон, радиус окружности, координаты ее центра.

Вариант № 15. Окружность и вписанный в нее прямоугольник, стороны которого относятся как 1 : 2. Вводимые параметры: две не противоположные стороны прямоугольника и, координаты одной из его вершин. Вычисляемые параметры и координаты: координаты остальных вершин прямоугольника, радиус окружности, координаты ее центра.

3.2.4 Содержание отчета

В отчете должны быть представлены:

1. Цель работы.
2. Основные сведения о теме работы.
3. Алгоритм разработанного проекта.
4. Перечень форм и элементов управления.
5. Процедуры обработки событий и данных.
6. Исходные данные и результаты их обработки.

3.3 Контрольные вопросы

1. Перечислить свойства объекта Screen.
2. Описать назначение свойств MouseIcon и MousePointer.
3. Описать назначение свойств Height и Width объекта Screen.
4. Описать назначение свойства ActiveControl объекта Screen.
5. Как проверить тип активного элемента управления?
6. Описать назначение свойства ActiveForm объекта Screen.
7. Описать синтаксис вызова функции InputBox.
8. Описать синтаксис вызова окна MessageBox.
9. Назначение «горячих» клавиш.
10. Клавиша табуляции и свойство TabIndex элемента управления.
11. Последовательность событий, связанных с клавиатурой.
12. Операторы SendKeys и AppActivate.
13. События мыши Click и DblClick.
14. События MouseDown и MouseUp. Событие MouseMove.
15. Последовательность событий мыши.

СПИСОК ЛИТЕРАТУРЫ

1. Сайлер Б., Споттс Дж. Использование Visual Basic 6. Классическое издание. - М.: Вильямс, 2007. – 832 с.

2. Сафронов И. Visual Basic в задачах и примерах. - СПб.: БХВ-Петербург, 2008. – 400 с.

3. Эпплман Д. Win32 API и Visual Basic. Для профессионалов. - СПб.: Питер, 2001. – 1120 с.

4. Сергеев В. Visual Basic 6.0. Наиболее полное руководство для профессиональной работы в среде Visual Basic 6.0. – СПб.: БХВ-Петербург, 2004. – 992 с.

5. Balena F. Programming Microsoft Visual Basic 6.0. – USA: Microsoft Press, 1999. – 1312 p.

6. Halvorson M. Microsoft Visual Basic Professional 6.0 Step by Step. – USA: Microsoft Press, 1998. – 672 p.

7. Holzner S. Visual Basic 6 Black Book: The Only Book You'll Need on Visual Basic. – USA: Coriolis Group Books, 1998. – 700 p.

8. Microsoft Visual Basic 6. Шаг за шагом: Практ. пособ. / Пер. с англ. – М.: Изд. ЭКОМ., Изд. 2-е, исправленное, 2003. – 432 с.

СОДЕРЖАНИЕ

Учебное издание

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторным работам
«Элементы управления среды Visual Basic.
Ввод и вывод информации»
из раздела «Программирование в среде Visual Basic.
Интегрированная среда разработки»
дисциплины «Информатика»
для студентов направления подготовки
6.050801 «Микро- и наноэлектроника»

Составители: ШКАЛЕТО Владимир Иванович
ЗАЙЦЕВ Роман Валентинович
ХРИПУНОВ Геннадий Семенович

Ответственный за выпуск М.В. Кириченко

План 2013 р.

Підписано до друку 10.06.13. Формат 60×84 1/16. Папір друк. №2.

Друк – ризографія. Гарнітура Times New Roman. Ум. друк. арк. 4,3.

Обл.-вид. 4,0. Тираж 50 прим. Зам. № _____. Ціна договірна.

Видавничий центр НТУ “ХПІ”. 61002, Харків, вул. Фрунзе, 21.
Свідоцтво про державну реєстрацію ДК № 116 від 10.07.2000 р.

Друкарня НТУ “ХПІ”. 61002, Харків, вул. Фрунзе, 21.