О.А. Карасева

Информатика и программирование

Екатеринбург
2012

МИНОБРНАУКИ РОССИИ

ФГБОУ ВПО «УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ЛЕСОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра информационных технологий и моделирования

О.А. Карасева

Информатика и программирование

Методические указания
(для проведения лабораторно-практических занятий для студентов в среде VB6)
по направлению 230700.62 «Прикладная информатика»,
080500.62 «Бизнес-информатика»
для студентов дневной, очно-заочной и заочной форм обучения
часть 1

Екатеринбург
2012

Печатается по рекомендации методической комиссии ФЭУ.

Протокол № 32 от 30.09.2011 г.

Рецензент – доцент кафедры ИТМ А.И. Монтиле

Редактор К.В. Корнева
Компьютерная верстка Е.В. Карпова

Подписано в печать		Поз. 113
Плоская печать	Формат 60x84 $^{1}/_{16}$	Тираж 30 экз.
Заказ №	Печ. л. 2,56	Цена 12 руб. 96 коп.

Редакционно-издательский отдел УГЛТУ
Отдел оперативной полиграфии УГЛТУ

Введение

Данные методические указания (МУ) предназначены для знакомства студентов с основами языка Visual Basic. Проекты, предложенные для разработки, порядок их следования, а также последовательность самих работ обеспечивают постепенное введение в среду программирования Visual Basic.

Лабораторная работа 1

Интегрированная среда разработки. Объекты и их свойства

Задание

1. Запустите программу Visual Basic. После ее запуска на экран будет выведено диалоговое окно *New Project* (рис. 1.1). Окно предоставляет возможность выбора дальнейших действий и содержит три вкладки:

New - начало создания нового проекта;

Existing - выбор приложений из существующих проектов;

Resent - поиск самых последних проектов.

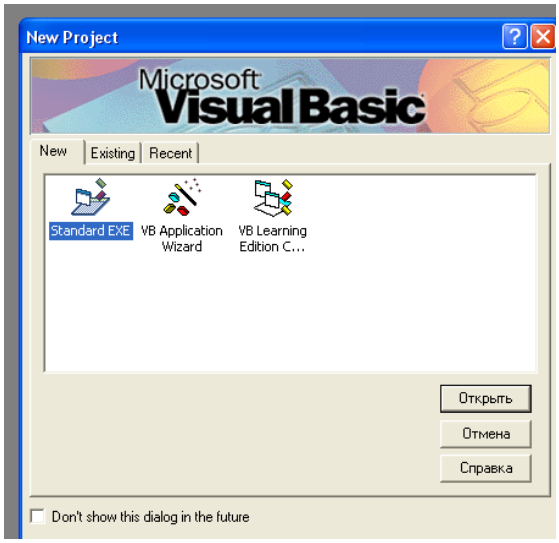

Рис. 1.1. Диалоговое окно нового проекта

2. Щелкните по кнопке *Открыть* уже выделенного программного проекта *Standart.EXE* на вкладке *New*, чтобы войти в интегрированную разработку проектов (Integrated Development Environment - IDE) (рис. 1.2).

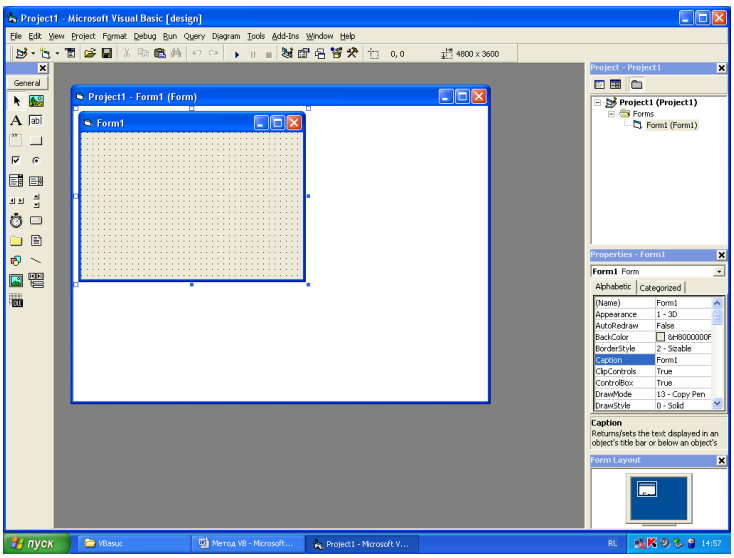

Рис. 1.2. Среда разработки приложения

3. Рассмотрите вид экрана. Если вид экрана отличается от представленного на рисунке 2, можно произвести соответствующую настройку:
   - если вы не видите *Окна конструктора форм*, выберите команду меню *View, Object;*
   - если отсутствует *Окно свойств Properties*, выберите команду меню *View, Properties Windows*;
   - если нет *Окна Tollbox*, выберите команду меню *View Tollbox*;
   - для того, чтобы представить, как форма будет располагаться на экране, удобно использовать окно расположения формы *Form Layout* (*View, Form Layout, Windows*);
   - если отсутствует *Окно проводника проекта Project Explorer (Project)*, выберите команду меню *View, Object Explorer*. *Окно проводника проекта* представляет дерево или список файлов, которые входят в проект. Это один или несколько файлов формы (файлы с расширением frm), файл самого проекта (файлы с расширением vbp), также могут входить файлы других типов (например, файлы программных модулей с расширением bas).

На панели инструментов *Окна Project* есть две кнопки: *View Code и View Project* (рис. 1.3). Первая позволяет открыть *Окно программного кода*, вторая – *Окно конструктора форм*.

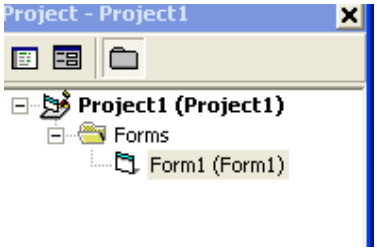

Рис. 1.3. Панель инструментов проекта

При создании нового проекта открывается новая форма. В *Окне свойств Properties* представлены свойства выделенного объекта.

4. Измените значения свойств с помощью *Окна свойств Properties:*
- в окне свойств *Caption* назначьте новый заголовок формы, для этого наберите фразу Первый_проект;
- измените цвет фона формы - свойство *Back Color*.
5. Запустите программу на выполнение, выбрав команду меню *Run, Start* (или клавиша F5).
6. Прервите выполнение программы (*Run, End*).
7. Дважды щелкните по форме. Появится окно программного кода с заголовком процедуры (рис. 1.4). Вы видите два окна списков: *Список объектов* и *Список событий*. По умолчанию для формы установлено событие Load (загрузка).

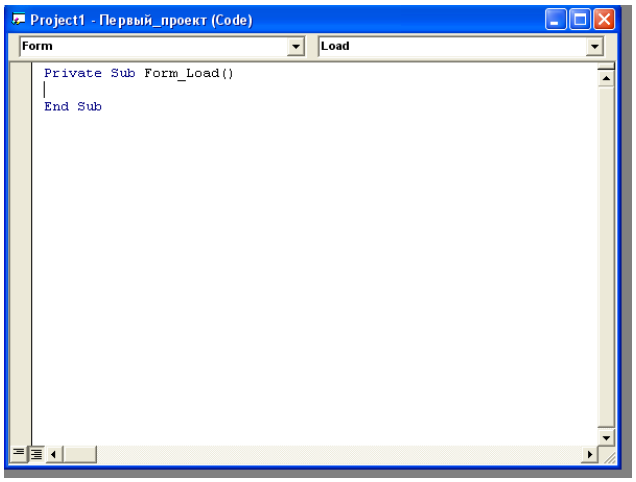

Рис. 1.4. Окно программного кода

8. Напишите процедуру, которая при щелчке мышью по форме изменит заголовок и разместит на форме картинку. Для этого:
   - убедитесь, что объект *Form* уже выбран в списке объектов;
   - в списке событий выберите *Click*;
   - наберите текст процедуры, как показано на рисунке 1.5, путь к картинке укажите свой.

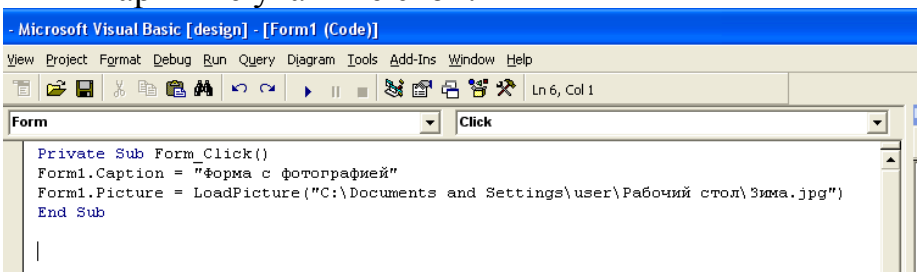

Рис. 1.5. Окно с кодом

9. Запустите программу на выполнение. Щелкните по окну формы и убедитесь, что картинка загрузилась.

10. Сохраните проект. Для этого создайте папку *Лаб1* (расположение папки определите сами) сохранив в нее форму и проект (рис. 1.6).

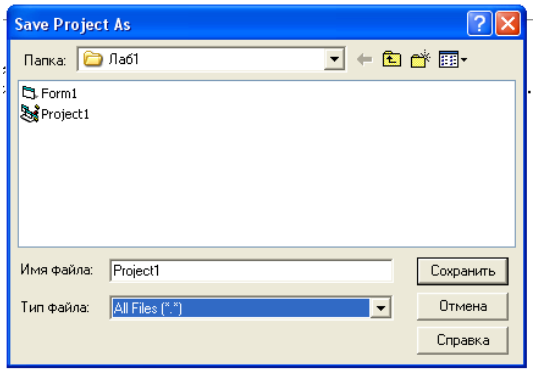

Рис. 1.6. Процесс сохранения проекта

11. В окне инструментов *Tool Box* выберите кнопку и разместите на форме. С помощью свойства *Caption* поместите на ней текст Выход, выберите шрифт для надписи с помощью свойства *Font*.
12. В верху формы разместите метку (*Label*) с текстом Первый проект. Выравнивание – по центру (*Alignment=Center*) (рис. 1.7).

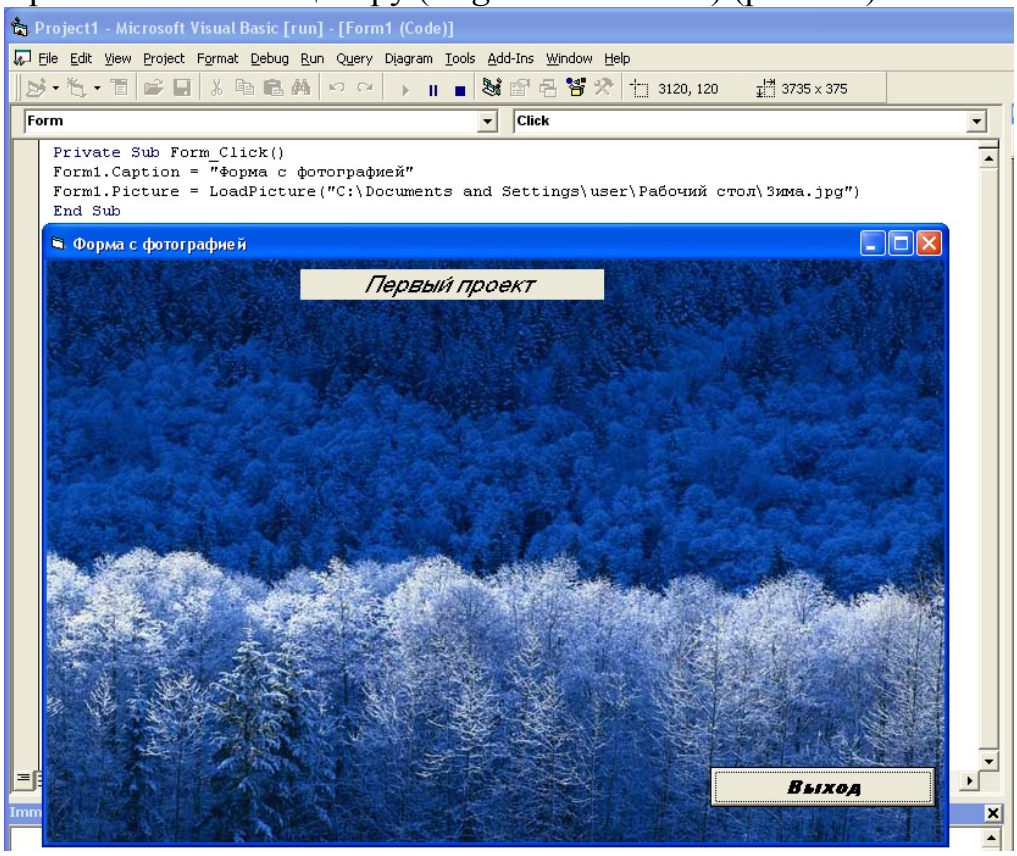

Рис. 1.7. Вид первого проекта

13. Запустите программу. Обратите внимание, что кнопка *Выход* не работает.
14. Для создания процедуры для кнопки *Выход* выберите из списка объект *Command1,* событие *Click* или дважды щелкните по кнопке *Выход* на форме в режиме конструктора. Далее наберите код:

```
Private Sub Command1_Click()
```

```
End
End Sub
```

15. Запустите программу на выполнение и проверьте работу. Кнопка *Выход* должна завершить работу.
16. Сохраните проект под тем же именем, а также сделайте копию под именем *Project1_2*. Далее работаем с проектом *Project1_2.*
17. Установите на форме специальную кнопку, при щелчке по которой на форме будет напечатан некоторый текст. Разместите на кнопке надпись и пиктограмму. Для этого:
    - свойству Caption кнопки установите значение Стихотворение;
    - установите значение свойства Style=1 – Graphical;
    - с помощью свойства *Picture* загрузите рисунок, какой вы хотели бы видеть как пиктограмму (с расширением bmp).
18. Наберите текст процедуры Command2_Click():

```
Private Sub Command2_Click()
Label1.Visible = False
Form1.Print
Form1.Print
Form1.Print
Form1.Print Tab(10); " У меня сегодня много дела: "
Form1.Print Tab(10); "Надо память до конца убить,"
Form1.Print Tab(10); "Надо, чтоб душа одервенела,"
Form1.Print Tab(10); "Надо снова научиться жить."
Form1.Print
Form1.Print
Form1.Print Tab(10); "Анна Ахматова"
End Sub
```

19. Запустите программу на выполнение. Щелкните по кнопке *Текст*. Разберитесь, как работает метод Print. Если сначала щелкнуть по форме, а потом по кнопке *Текст*, текст будет печататься по картинке и может быть плохо виден.
20. Сделайте так, чтобы при щелчке по кнопке картинка исчезала. Для этого в начало процедуры Command2_Click вставьте строку:

```
Form1.Picture = LoadPicture("")
```

21. Самостоятельно добавьте в эту же процедуру строку, которая сменит заголовок окна на слово Стихотворение.
22. Если вы щелкните на кнопку *Command2* несколько раз, текст на форме будет повторяться. Чтобы это не происходило, в начало процедуры Command2_Click добавьте еще одну строку:

```
Form1.Cls
```

23. Можно табулировать текст, сдвигая от края с помощью функций Tab(n) и Spc(n), где n – количество пробелов. Замените последнюю строку на текст с именем автора, поместив фамилию автора справа на форме.

24. Запустите программу и проверьте корректность выполнения.
25. Сохраните и покажите преподавателю оба проекта.

Лабораторная работа 2

Перемещение объектов. Анимация

2.1. Перемещение объектов с помощью полос прокрутки

Горизонтальные (*HScrollBar*) и вертикальные (*VScrollBar*) полосы прокрутки действуют одинаково. К специфическим свойствам полос прокрутки относятся свойства:

- Value – целое число, отражающее текущую позицию ползунка на полосе прокрутки;
- Max – значение свойства Value, соответствующее крайнему правому (нижнему) положению ползунка;
- Min - значение свойства Value , соответствующее крайнему левому (верхнему) положению ползунка;
- Large Change - целое число, равное шагу изменения свойства Value, при щелчке внутри полосы прокрутки;
- Small Change - целое число, равное шагу изменения свойства Value, при щелчке по стрелке полосы прокрутки;

К наиболее важным событиям, связанным с полосами прокрутки, относятся Scroll (прокрутка) и Change (изменение).

Проект Полосы прокрутки

Постановка задачи

Создать проект, в котором управление объектом осуществляется с помощью горизонтальной и вертикальной полос прокрутки. Вид работающего приложения показан на рисунке 2.1.

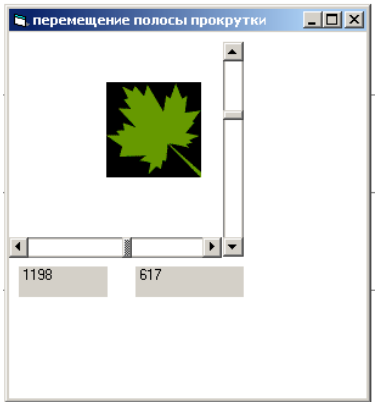

Рис. 2.1. Вид приложения

Порядок действий

1. Расположите на форме объекты: окно изображения *Image*, вертикальную и горизонтальную полосы прокрутки, две метки под горизонтальной полосой.
2. Установите значения свойств в соответствии с таблицей 2.1.

Таблица 2.1

Объекты проекта и их свойства

Объект	Свойство	Значение свойства
Форма	Name	Frm1
	ScaleMode	1-Twip
	BackColor	Белый
	Width	2970
	Heigth	3950
Горизонтальная полоса прокрутки	Name	hsb
	Heigth	255
	Width	2625
	Max	2100
	Min	0
	SmallChange	10
	LargeChange	100
Вертикальная полоса прокрутки	Name	vsb
	Heigth	2645
	Width	255
	Max	2100
	Min	0
	SmallChange	10
	LargeChange	100
Окно изображения	Picture	Подберите любой рисунок из коллекции VB
	Stretch	True
Метка	Name	lblH
Метка	Name	lblV

3. Наберите программный код:

```
Private Sub hsb_Change()
Image1.Left = hsb.Value
lblH.Caption = Str(hsb.Value)
End Sub

Private Sub hsb_Scroll()
Image1.Left = hsb.Value
lblH.Caption = Str(hsb.Value)
```

```
End Sub

Private Sub vsb_Change()
Image1.Top= vsb.Value
lblV.Caption = Str(vsb.Value)
End Sub

Private Sub vsb_Scroll()
Image1.Top = vsb.Value
lblV.Caption = Str(vsb.Value)
End Sub
```

4. Закомментируйте процедуру Private Sub hsb_Scroll() . Проанализируйте, что изменилось при горизонтальном перемещении объекта.
5. Снимите комментарии.
6. Сохраните проект под именем *Полосы прокрутки.*

2. Перемещение объектов с помощью мыши

Для выполнения ряда действий в языке VB предусмотрена операция *Drag and Drop* (Перетащить и Остановить).

Метод *Move* обеспечивает перемещение объекта в новое положение с заданными координатами верхнего левого угла объекта.

Пример.

Объект.Move Left+dx, Top+dy

Эта команда приводит к перемещению объекта в окне формы на dx вниз и на dy вправо.

Перемещаемый объект будем называть объектом-источником, а объект, над которым освобождается объект-источник, - объектом-целью.

Для того, чтобы объект можно было перетащить на новое место, следует задать его свойству DragMode (режим перетаскивания) значение 1-Automatic, а свойству Enable – значение True.

Вид указателя мыши при перетаскивании может быть задан с помощью свойства DragIcon (значок при перетаскивании).

Рассмотрим заголовок процедуры обработки события DragDrop:

```
Private Sub ОбъектЦель_DragDrop([Index As Integer,] Source As Control, X As Single,
Y As Single)
```

Здесь: ОбъктЦель – имя объекта, с которым связано событие DragDrop;

Index - переменная, которая используется только в том случае, если объект-цель является элементом массива объектов;

Source - переменная, которая равна объекту-источнику, «захваченному» мышью;

X, Y – координаты курсора мыши в момент наступления события DragDrop относительно верхнего левого угла объекта-цели. Если объектом

целью является форма или графическое окно, то координаты задаются в тех единицах, которые заданы свойством ScaleMode.

Проект Мышь

Постановка задачи

Выполнить проект, который позволит «навести порядок» на форме. Объект с недовольным выражением лица «проводить» в корзину, а остальных «пригласить в компанию для сотрудничества» (переместить в центр формы). Иконки следует взять у преподавателя.

Работающее приложение до наведения порядка представлено на рисунке 2.2.

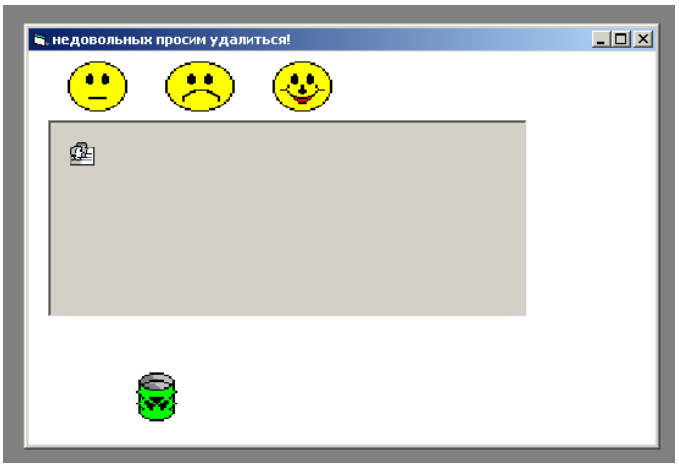

Рис. 2.2. Исходный вид работающего приложения

Порядок действий

1. Расположите на форме следующие объекты: графическое окно PictureBox и пять объектов типа Image (рис. 2.3). Верхний ряд объектов типа Image образуют массив объектов.

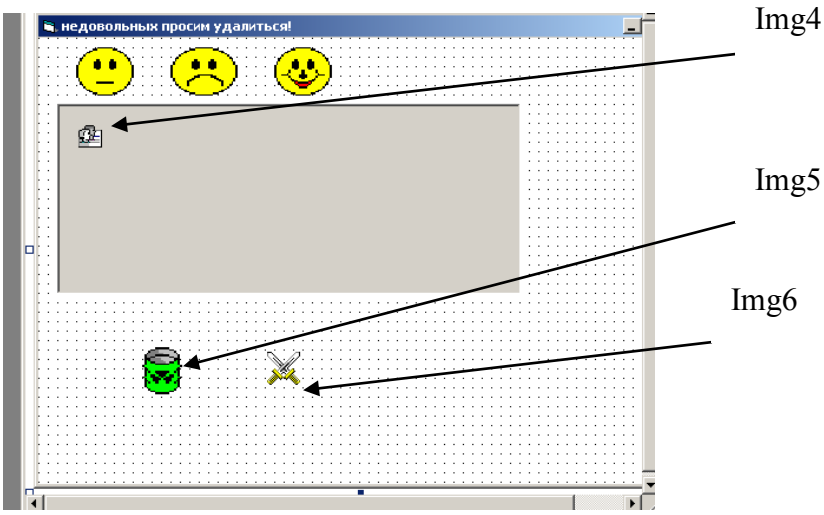

Рис. 2.3. Вид приложения в конструкторе

2. Установите значения свойств объектов в соответствии с таблицей 2.2. По окончании установки значений свойств объектов форма должна выглядеть примерно так, как показано на рисунке 2.1.

Таблица 2.2

Объекты проекта и их свойства

Объект	Имя объекта по умолчанию (значение свойства Name)	Свойство	Значение свойства
Форма	Form1	Name	Frm1
		Caption	Недовольных просим удалиться!
		BackColor	Белый
Графическое окно	Picture1	Name	PicTable
		BorderStyle	1
		BackColor	Серый
		Caption	-
Окно изображений	Image (три в верхнем ряду)	Name	img
		Index	1 (2 или 3 соответственно)
		Stretch	True
		Picture	Укажите имена иконок из папки «Иконки» для каждого окна соответственно
		DragMode (режим перетаскивания)	1-Automatic
		DragIcon (значок при перетаскивании)	Можете повторить эти же иконки соответственно
		Enabled	True
Окно изображений	Image (в нижнем ряду)	Name	Img5
		Picture	Укажите имя иконки из папки «Иконки» для корзины
Окно изображений	Image (в центре, в графическом окне)	Name	Img4
		Picture	Укажите имя иконки из папки «Иконки»
Окно изображений	Image (в нижнем ряду)	Name	Img6
		Picture	Укажите имя иконки из папки «Иконки»
		Visible	False

3. Двойным щелчком по объекту img5 (корзина) откройте окно кода.
4. Откройте список событий и выберите событие *DragDrop*. Наберите программный код процедуры:

```
Private Sub Img5_DragDrop(Source As Control, X As Single, Y As Single)
```

```
Source.Visible = False
Img5.Picture = Img6.Picture
End Sub
```

Здесь первая строка делает невидимым перетаскиваемый объект при его «сбросе» на объект img5. Вторая строка обеспечивает замену изображения урны на «символ расправы».

5. Сохраните проект под именем *Мышь* в папке *Перемещение*.
6. Запустите приложение. Убедитесь, что при перетаскивании объекта в урну его изображение исчезает с формы, а изображение корзины заменяется другим изображением. Обратите внимание на то, что при перетаскивании появляются соответствующие значки.
7. Наберите код процедуры:

```
Private Sub picTable_DragDrop(Source As Control, X As Single, Y As Single)
If Source = img(1) Or Source = img(2) Or Source = img(3) Then
Set Source.Container = picTable
Source.Move X, Y
End If
End Sub
```

Здесь, во второй строке, используется свойство Container . Если его не применить, перетаскиваемые объекты Image окажутся под объектом PictureBox, а не в нем. В качестве контейнера могут использоваться объекты Form, PictureBox, Frame.

Формат использования свойства:

Объект1. Container= Объект2

Здесь Объект1 перемещается в Объект2, как в контейнер.

8. Сохраните изменения в проекте.
9. Запустите приложение. Проанализируйте, как происходит перетаскивание объектов в форме. Независимо от того, где находился курсор в момент захвата объекта-источника, при его освобождении верхний левый угол объекта перемещается в точку с координатами курсора мыши. Объект как бы «отскакивает». Для устранения этого недостатка можно использовать событие *MouseDown*. Оно позволяет для захваченного объекта вычислить поправку к координатам. Если объект имеет свойство *DragMode=1*, то он не получает ни одного события, кроме DragDrop. Нужно сменить это свойство на 0 (*Manual*). Чтобы объект начал двигаться, к нему нужно применить метод *Drag 1.*
10. Смените значения свойств DragMode объектов *img1-3* на 0 (Manual).
11. Наберите программный код процедуры для массива img (трех верхних объектов), предварительно описав в разделе общих объявлений новые переменные.

```
Dim dragx As Single, dragy As Single

Private Sub img_MouseDown(Index As Integer, Button As Integer, Shift As Integer,
X As Single, Y As Single)
```

```
dragx = X
dragy = Y
img(Index).Drag 1
End Sub
```

12. Измените код процедуры picTable:

строку
```
Source.Move X, Y
```

Замените на строку

```
Source.Move (X - dragx), (Y - dragy)
```

13. Запустите приложение и проверьте, происходит ли перемещение объектов нормально.
14. Измените проект следующим образом: после того, как объекты с формы будут перемещены, в урну и в центр формы, заголовок должен смениться на «Порядок наведен!». Для этого введите новую целочисленную глобальную переменную C, означающую количество объектов на форме, которое нужно переместить. Сразу после загрузки формы она должна быть равна 3:

```
Private Sub Form_Load()
c = 3
End Sub
```

15. Далее в процедуры Img5_DragDrop и picTable_DragDrop введите строки:

```
c = c - 1
If c = 0 Then
frm1.Caption = "Порядок наведен!"
End If
```

16. После перемещения всех объектов в соответствующем направлении окно приложения будет иметь вид, представленный на рисунке 2.4.
17. Проверьте, как работает приложение.
18. Сохраните изменения в проекте.

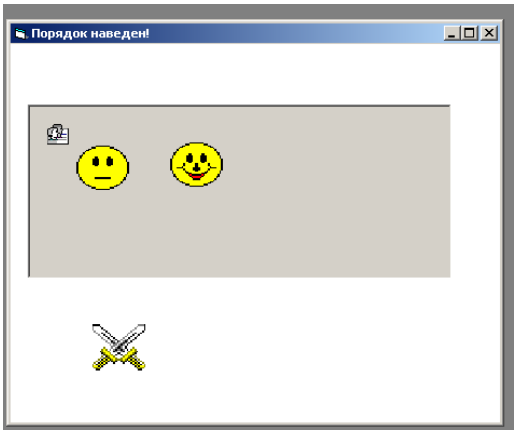

Рис. 2.4. Конечный вид работающего приложения

3. Анимация с использованием таймера и метода *Move*

Дополнение к проекту Мышь

Постановка задачи

Доработать проект так, чтобы после «наведения порядка на форме» началось перемещение Земного шара из правого нижнего угла в левый верхний. При этом шар должен увеличиваться в размерах и не должен выходить за пределы формы (рис. 2.5, 2.6).

Порядок действий

1. Откройте проект *Мышь*.
2. Расположите в левом нижнем углу формы таймер (Timer), а в правом нижнем - объект типа Image (шар).
3. Установите для объекта Image свойство Name – ImgSun,
   Visible=False,
   Stretch=True
4. Установите для объекта Timer свойство Enabled = False
   Interval=60
5. Внесите изменения в процедуры Img5_DragDrop и picTable_DragDrop:
   Добавьте после строки
   ```
   frm1.Caption = "Порядок наведен!"
   ```
   строку
   ```
   Timer1.Enabled = True
   ```

6. Введите программный код процедуры Timer1_Timer():

```
Private Sub Timer1_Timer()
ImgSun.Visible = True
If ImgSun.Top > 0 Then
  If ImgSun.Left > 0 Then
  ImgSun.Move ImgSun.Left - 50, ImgSun.Top - 50
  ImgSun.Width = ImgSun.Width + 5
  ImgSun.Height = ImgSun.Height + 5
End If
Else
  Timer1.Enabled = False
End If
End Sub
```

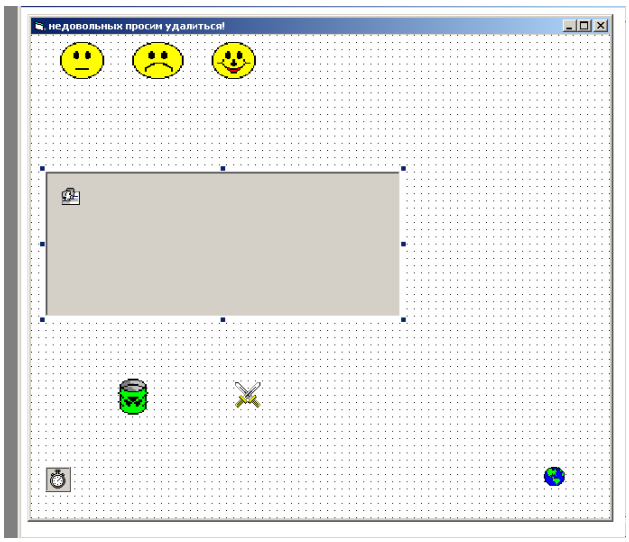

Рис. 2.5. Расположение объектов на форме

Код

```
Timer1.Enabled = True
```

запускает таймер, применяемый для имитации движения шара.

После включения таймер запускает эту процедуру через каждые 60 мс. Каждое выполнение процедуры *Move* приводит к смещению объекта на 50 твипов вверх и на 50 твипов влево. Это движение продолжается до тех пор, пока шар не выйдет за пределы формы. В этом случае таймер выключается (Timer1.Enabled = False) (см. рис. 2.5).

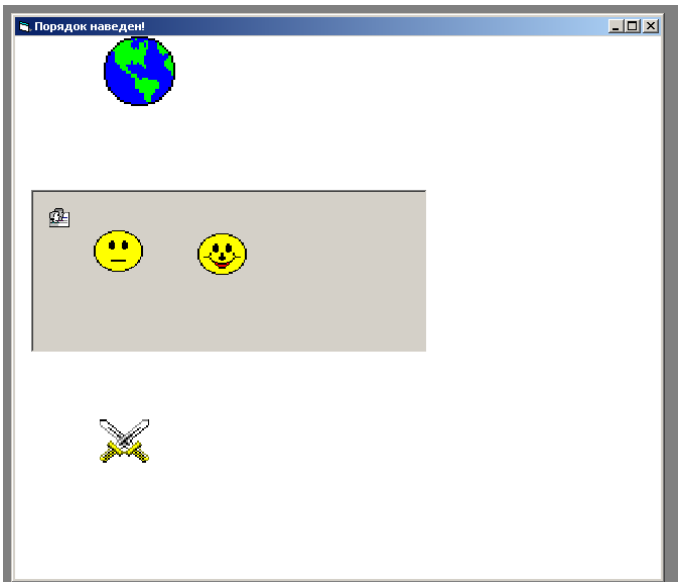

Рис. 2.6. Вид формы с движущимся объектом

7. Проверьте правильность работы проекта.
8. Сохраните изменения в программе.

Лабораторная работа 3
Основные этапы разработки Windows-приложений

Процесс создания Windows-приложения в среде Visual Basic состоит из следующих 8 этапов.

1. **Содержательная постановка задачи**

Это четкая формулировка, выделение исходных данных и точные указания относительно того, какие результаты и в каком виде должны быть получены.

2. **Математическая постановка задачи. Выбор метода решения**

На этом этапе строится математическая модель – система математических отношений – формул, уравнений, неравенств и т.д., отражающих существенные свойства объекта или явления.

При построении математической модели не всегда удается найти формулы, явно выражающие искомые величины через данные. В таких случаях используют математические методы, позволяющие дать ответы с той или иной степенью точности.

Могут встретиться задачи, которые не допускают или не требуют математической постановки (например, задачи обработки текста). Таким образом, данный этап не является обязательным.

3. **Разработка пользовательского интерфейса**

Интерфейс – это внешняя оболочка приложения, позволяющая работать с информацией, хранящейся на компьютере или за его пределами, а также вводимой пользователем. Интерфейс должен обеспечить максимальное удобство и эффективность работы.

4. **Программирование**

На этом этапе определяют события, которые будут происходить в процессе работы приложения, составляют алгоритмы процедур для этих событий и пишут программы (программные коды этих процедур).

5. **Отладка**

Это процесс испытания работы программы и исправление этих ошибок. VB располагает эффективным инструментом для поиска источников ошибок.

6. Анализ результатов

На этом этапе уже разработанная программа применяется для получения искомых результатов; производится анализ результатов и в случае необходимости – уточнение математической модели (с последующей корректировкой алгоритма и программы).

7. Компиляция

Этот термин означает превращение проекта в исполняемое приложение, способное работать самостоятельно за пределами среды проектирования (этап не является обязательным).

8. Создание инсталляционного пакета

В среде VB подход к переносу программы с компьютера на компьютер следующий: не копировать приложение, а инсталлировать его (этап выполняется при необходимости).

Все эти этапы продемонстрированы на конкретном примере.

Проект Треугольник

1. Содержательная постановка задачи

Определить периметр и площадь треугольника по трем его сторонам.

Дано: A, B, C – значения сторон треугольника.

Требуется определить: P - периметр треугольника, S - площадь треугольника.

Ограничения на значения исходных данных и их соотношения:

A>0, B>0, C>0

A+B>C, A+C>B, B+C>A одновременно (условие существования треугольника).

2. Математическая постановка задачи

Для решения задачи существуют готовые формулы:

P=A+B+C;

$S=\sqrt{Pp(Pp-A)(Pp-B)(Pp-C)}$ (формула Герона),

где $Pp=\frac{P}{2}$ - полупериметр.

3. Разработка пользовательского интерфейса

При разработке интерфейса необходимо учитывать следующее:

- строка заголовка должна содержать название приложения – Периметр и площадь треугольника;
- для наглядности приложение должно включать чертеж;
- для ввода исходных данных необходимо использовать текстовые поля; их должно быть 3;
- вычисления должны выполнятся при нажантии на кнопку *Вычислить*;
- для вывода результатов следует использовать текстовые поля (2 поля);
- для завершения работы должна быть предусмотрена кнопка *Выход*;
- для удобства пользователя приложение должно содержать поясняющие записи. Для этой цели будем использовать метки;
- зоны ввода и вывода данных должны быть визуально разделены (будем использовать линию).

Принимая во внимание все изложенное выше, окно работающего приложения может иметь вид, представленный на рисунке 3.1.

Порядок действий

1. Подготовьте рисунок треугольника и сохраните его в папке *Треугольник* под именем *Треугольник.bmp*.
2. Расположите на форме объекты, как показано на рисунке 3.1.

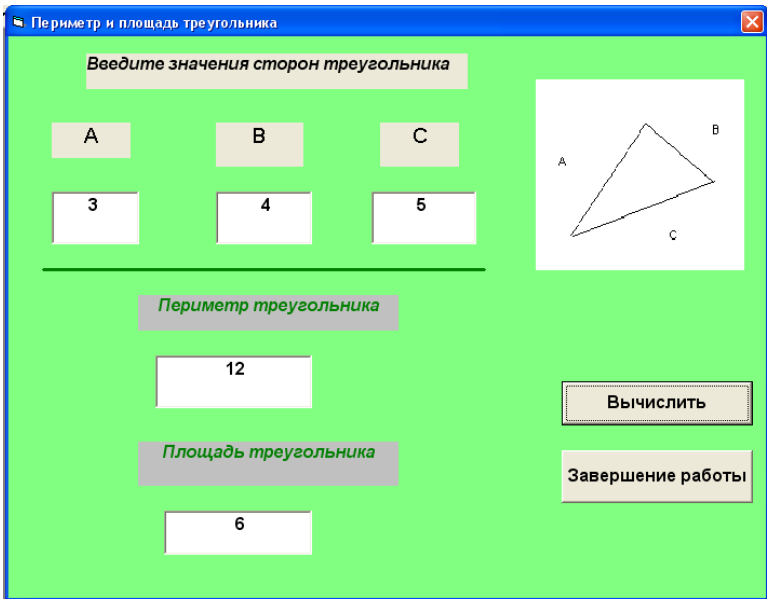

Рис. 3.1. Вид работающего приложения

3. Установите значения свойств объектов в соответствии с таблицей 3.1. Задавая какое-либо свойство объекта, наблюдайте за тем,

что изменяется, и вы поймете, что это за свойство. Префиксы в именах объясняются следующими соглашениями:

frm	-	форма;	pic	-	рисунок;
lbl	-	метка;	cmd	-	командная кнопка.
txt	-	текстовое поле;			
img	-	изображение;			

4. Сохраните данный проект в папку *Треугольник*. Имя файла формы – *Треугольник.frm*, имя файла проекта – *Треугольник.vpb*.

Таблица 3.1

Объекты проекта и их свойства

Объект	Имя объекта по умолчанию (значение свойства Name)	Свойство	Значение свойства
Форма	Form1	Name	frmТреугольник
		Caption	Периметр и площадь треугольника
		BackColor (вкладка Palette)	Светло-зеленый
Метка	Label1	Caption	Введите значения сторон треугольника
Метка	Label2	Caption	A
Метка	Label3	Caption	B
Метка	Label4	Caption	C
Выделить метки Label2-Label4 (клавиша Shift+мышь) и установить одинаковые для всех меток свойства		Alighment	2-Center
		Back Style	0-Transparent
		Font	Arial, жирный, 12
		Fore Color	Черный
Метка	Label5	Caption	Периметр треугольника
Метка	Label6	Caption	Площадь треугольника
Выделить метки Label1, Label5, Label6 и установить одинаковые для всех меток свойства		Alighment	2-Center
		BackStyle	Светло-серый
		Font	Arial, жирный курсив, 12
		Fore Color	Темно-зеленый
Текстовое поле	Text1	Name	txtA
		Объект	Имя объекта по умолчанию (значение свойства Name)

Окончание табл. 3.1

Объект	Имя объекта по умолчанию (значение свойства Name)	Свойство	Значение свойства
		Text	Пусто
Текстовое поле	Text3	Name	txtB
		Text	Пусто
Текстовое поле	Text4	Name	txtC
		Text	Пусто
Текстовое поле	Text5	Name	txtP
		Text	Пусто
Выделить метки Text1 - Text5 и установить одинаковые для всех меток свойства		Alighment	2-Center
		Font	Arial, обычный, 12
Командная кнопка	Command1	Name	cmdStart
		Caption	ВЫЧИСЛИТЬ
Командная кнопка	Command2	Name	cmdEnd
		Caption	ЗАВЕРШЕНИЕ РАБОТЫ
Выделить кнопки Command1-Command2 и установить одинаковые для всех меток свойства		Font	Arial, жирный, 12
Окно изображения	Image1	Stretch	True (картинка подгоняется под размер объекта управления)
		BorderWidth	1-Fixed Single
Линия	Line1	BorderWidth	3
		BorderColor	Темно-зеленый

4. Программирование

Прежде чем приступить к программированию, необходимо определить те события, для которых необходимо разработать алгоритмы и описать их на языке программирования. В примере есть следующие события:

- щелчок по кнопке *Вычислить*;
- щелчок по кнопке *Завершение работы*;
- загрузить чертеж при загрузке формы.

Алгоритм вычисления прост:

- ввести исходные данные;
- вычислить периметр;
- вычислить полупериметр;
- вычислить площадь;
- вывести результат P – периметр;
- вывести результат S – площадь.

Порядок действий

1. Самостоятельно напишите процедуру обработки события для кнопки *Завершение работы*.
2. Напишите процедуру, которая будет выполняться при загрузке формы:

```
Private Sub Form_Load()
Image1.Picture = LoadPicture("……. ")
End Sub
```

3. Напишите процедуру, которая будет выполняться при нажатии на кнопку *Вычислить*:

```
Private Sub cmdStart_Click()
B = txtB.Text
A = txtA.Text
C = txtC.Text
P = A + B + C
Pp = P / 2
S = Sqr(Pp * (Pp - A) * (Pp - B) * (Pp - C))
txtP.Text = P
txtS.Text = S
End Sub
```

5. Отладка

На этом этапе выполняется правильность работы программы.

Порядок действий

1. Запустите программу на выполнение и введите значения сторон, как показано на рисунке 3.2.

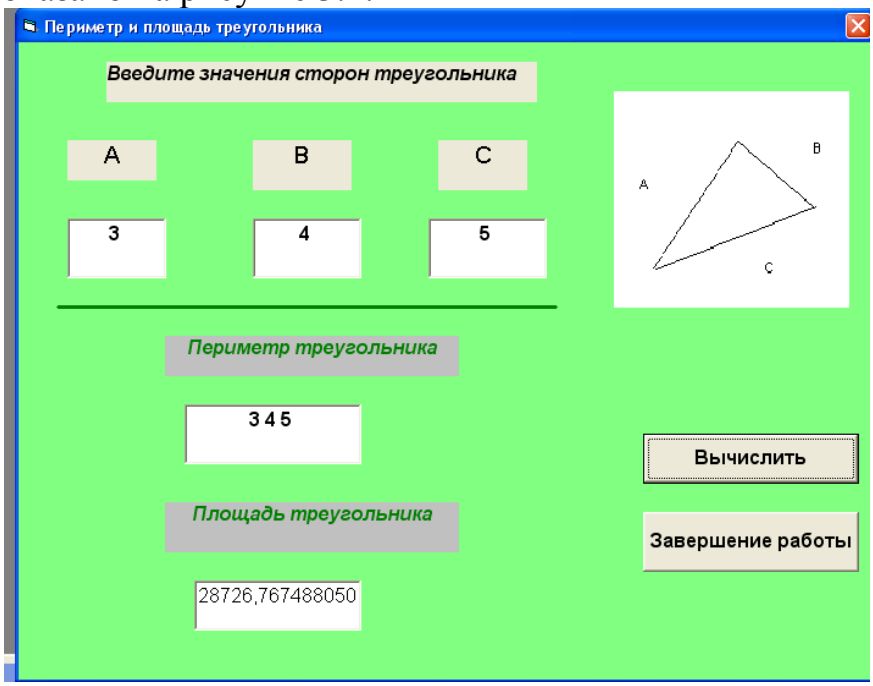

Рис. 3.2. Вид работающего приложения

2. Проанализируйте результаты. Периметр должен быть равен 12. Объясним результат.

VB считает содержимое текстового поля текстом. При работе с текстом знак «+» интерпретируется как соединение (конкатенация). Отсюда и результат. Т.к. при вычислении полупериметра и площади используются различные арифметические операции, VB понимает, что эти действия выполняются с цифрами.

3. Измените программный код:

```
Private Sub cmdStart_Click()
B = Val(txtB.Text)
A = Val(txtA.Text)
C = Val(txtC.Text)
P = A + B + C
Pp = P / 2
S = Sqr(Pp * (Pp - A) * (Pp - B) * (Pp - C))
txtP.Text = Str(P)
txtS.Text = Str(S)
End Sub
```

Функция **Val** преобразует текстовый аргумент в числовое значение. Функция **Str** выполняет обратное преобразование.

4. Запустите программу на выполнение. В результате окно работающего приложения должно выглядеть так, как показано на рисунке 3.2.
5. Запустите программу на выполнение, введя значения исходных данных:

A=10; B=2; C=1.

Система выдаст сообщение об ошибке.

6. Щелкните по кнопке *Debug* (Отладка). Ошибка произошла в связи с тем, что вы ввели данные, при которых треугольник не может существовать: подкоренное выражение отрицательно. В следующей лабораторной работе нужно будет устранить этот недостаток.
7. Проверьте процедуру обработки события *Завершение работы*.
8. Сохраните проект и форму с индексом 2.
9. Измените процедуру обработки события кнопки *Вычислить*, добавив 5 строк в начало процедуры:

```
txtP.BackColor = QBColor(14)
txtS.BackColor = QBColor(14)
txtP.ForeColor = QBColor(2)
txtS.ForeColor = QBColor(2)
frmТреугольник.BackColor = QBColor(15)
```

Функция QBColor – функция, которая позволяет изменять цвета в зависимости от аргумента.

10. Проанализируйте, какие свойства изменились по сравнению с первым вариантом программы.
11. Сохраните изменения в проекте и покажите оба варианта формы.

В этом проекте используется конструкция условного оператора:

```
If условие  Then
    Оператор1
Else
    Оператор2
End If
```

и функция MsgBox для вывода текста в две строки благодаря используемой внутри функции Chr(13).

Проект Теугольник_3

Постановка задачи

Необходимо предусмотреть в проекте проверку данных. Программа должна выдавать предупреждающее сообщение, если вводятся значения сторон треугольника, при которых треугольник не существует (рис. 3.3).

Порядок действий

1. Откройте проект ***Треугольник***_2
2. Внесите изменения в процедуру cmdStart_Click():

после строк

```
B = Val(txtB.Text)
A = Val(txtA.Text)
C = Val(txtC.Text)
```

введите строки, позволяющие проверить корректность исходных данных.

```
If (A + B) > C And (B + C) > A And (C = A) > B Then
   P = A + B + C
   Pp = P / 2
   S = Sqr(Pp * (Pp - A) * (Pp - B) * (Pp - C))
   txtP.Text = Str(P)
   txtS.Text = Str(S)
Else
   MsgBox "Ошибка!" + Chr(13) + "Сумма двух сторон треугольника должна быть
_ больше третьей стороны", vbCritical + vbOKOnly, " Ошибка!!!"
   txtA.Text = ""
   txtBText = ""
   txtC.Text = ""
   txtA.SetFocus
End If
```

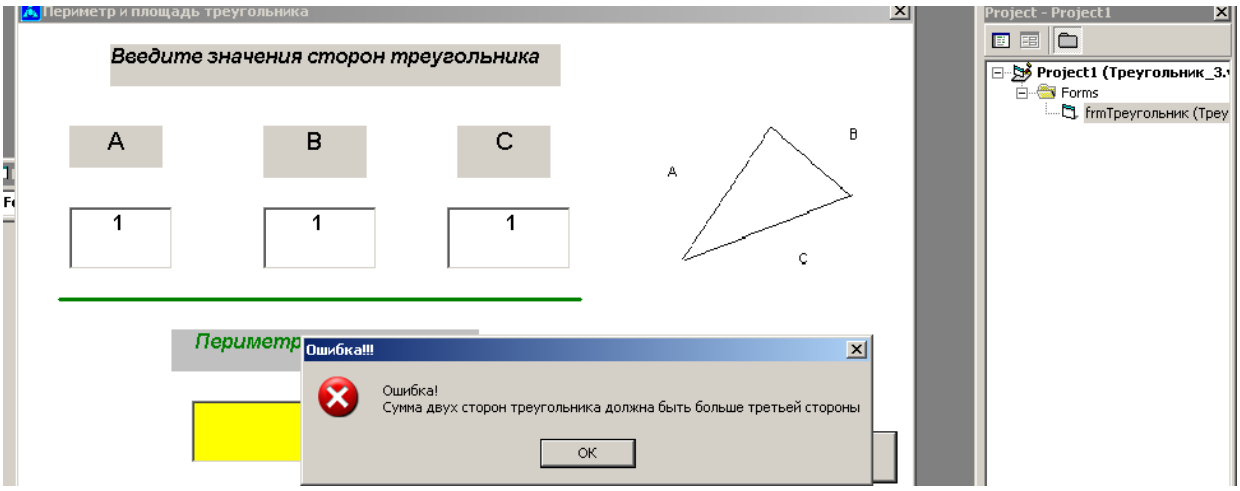

Рис. 3.3. Вид работающего приложения при некорректном вводе исходных данных

3. Проверьте, как работает приложение при вводе значений сторон, при которых треугольник не существует.
4. Сохраните изменения в проекте.

6. Анализ результатов

Приложение практически готово. Можно вводить новые данные и анализировать результаты.

7. Создание исполняемого приложения

Для того, чтобы Windows-приложение могло работать независимо от среды проектирования (автономно), нужно произвести трансляцию проекта в исполняемый *exe-файл*. Для этого:

- выполните команду меню *File, Make ИмяВашегоФайла.exe* (например, *File*, *Make*, *Треугольник.exe*). По умолчанию файл создается в той же папке, что и проект, и будет иметь значок, который вы выбрали;
- закройте среду разработки программ VB;
- запустите на выполнение исполняемый файл *Треугольник.exe* и убедитесь, что он работает автономно.

Если вы хотите, чтобы исполняемый файл вашего проекта имел собственный значок в Windows, задайте значение свойству Icon формы. Щелкните по кнопке построителя (троеточие) и в окне *Load Icon* найдите и откройте нужный файл. Вы можете найти изображение иконки на вашем компьютере по образцу *.ico и положить в свою папку.

8. Создание инсталляционного пакета

Этот этап будет рассмотрен дальше.

Задачи для самостоятельного решения (ветвления)

1. Даны два угла треугольника (в градусах). Определить, существует ли такой треугольник, и если да, то будет ли он прямоугольным.
2. Даны действительные числа x и y, не равные друг другу. Меньшее из этих чисел заменить половиной их суммы, а большее – их удвоенным произведением.
3. Подсчитать количество целых чисел среди a, b, c.
4. Услуги телефонной сети оплачиваются по следующему правилу: за разговоры до A минут в месяц – B руб., а за разговоры сверх установленной нормы оплачиваются из расчета C руб. за минуту. Написать программу, вычисляющую плату за пользование телефоном для введенного времени разговоров в месяц.

Лабораторная работа 4.
Данные в VB

В работе будут использованы:
функция:
Date (текущая системная дата);
свойства:
FontName (тип шрифта);
FontSize (размер шрифта);
FontItalic (начертание шрифта – курсив);
FontBold (начертание шрифта - полужирный);
событие:
Load (загрузка);
операторы:
With / end With (изменение нескольких свойств одного объекта);
Dim (определение переменной);
Def (объявление типа);
Option Explict (обязательное явное объявление всех переменных в программе);
типы данных:
Integer (целое число);
Single (десятичное число одинарной точности);
Double (десятичное число двойной точности);
String (строка символов);
Date (дата).

Формат оператора определения переменной:
Public/Private/Dim ИмяПеременной [As ТипПеременной]

Проект Треугольник2 (доработка проекта *Треугольник*)

1. Откройте проект *Треугольник.vpb*. Сохраните его под именем *Треугольник2.vpb*.
2. Закройте среду VB и откройте снова, выбрав только что созданный проект.
3. Измените программный код, добавив описание переменных:

```
Dim A  As Single
Dim B As Single
Dim C As Single
Dim P As Double
Dim S As Double
```

4. Теперь вы можете вводить числа не только целые. Запустите приложение (рис. 4.1).

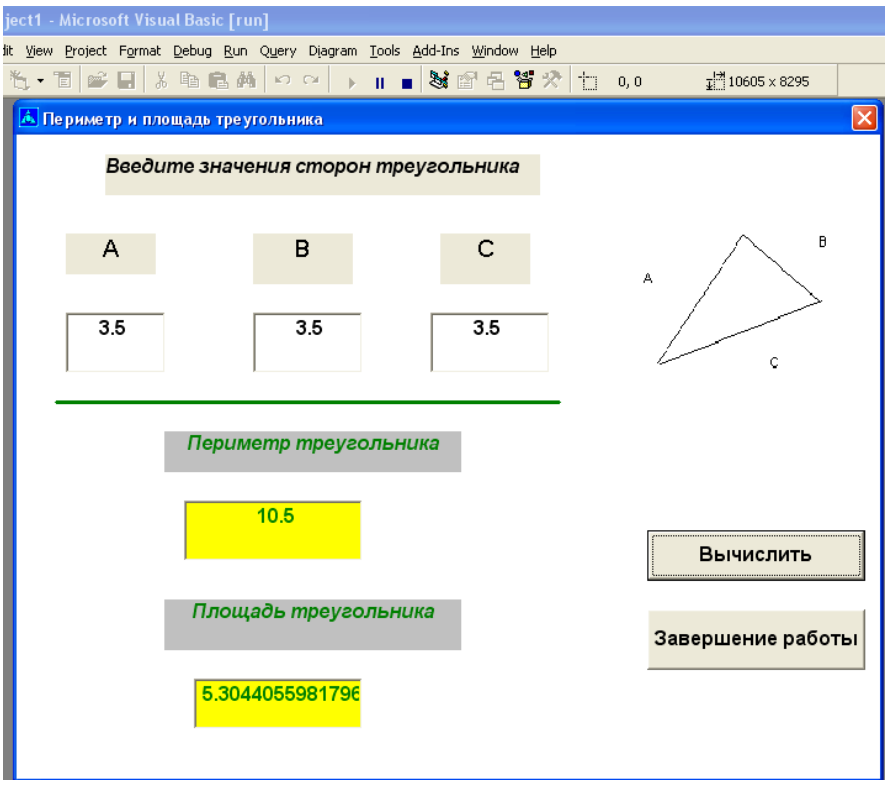

Рис. 4.1. Вид приложения с доработкой

5. Сохраните изменения в проекте.

Проект Переменные-Дата

Постановка задачи

Создать проект, позволяющий определить, сколько дней, часов и минут вы прожили со дня своего рождения до сегодняшнего дня, а также определить, какая дата будет через определенное количество дней.

Окно работающего приложения может иметь вид, представленный на рисунок 4.2.

Порядок действий

1. Расположите на форме объекты в соответствии с рисунком 4.2 и таблицей 4.2. Значения свойств, определяющих внешний вид, установите сами.

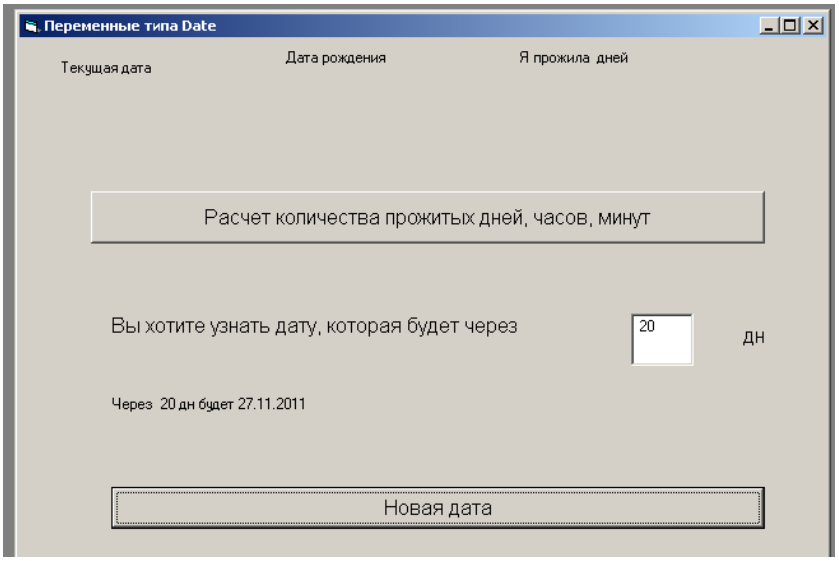

Рис. 4.2. Внешний вид приложения

Таблица 4.2

Объекты проекта и их свойства

Объект	Свойство	Значение свойства
Форма	Name	frmПеременные
	Caption	Переменные типа Date
Метка	Caption	Вы хотите узнать дату, которая будет через
Метка	Caption	дней
Метка	Name	lblNewDate
	Caption	Пусто
Текстовое поле	Name	txtKD
	Caption	Пусто
Командная кнопка	Name	cmdПуск1
	Caption	Расчет количества прожитых дней, часов, минут
Командная кнопка	Name	cmdПуск2
	Caption	Новая дата

2. Составьте программный код процедуры, позволяющий рассчитать новую дату. В процедуре используются следующие переменные:

TD – текущая дата;
K – количество дней
TB – новая дата.

```
Private Sub cmdПуск2_Click()
TD = Date
```

```
K = Val(txtKD.Text)
ND = TD + K
lblNewDate.Caption = "Через " + Str(K) + " дн будет " + Str(ND)
End Sub
```

3. Запустите приложение, введите количество дней и щелкните по кнопке.
4. Сохраните проект в папке ***Переменные-Дата*** с именем *Дата.vpb.*
5. Самостоятельно напишите код процедуры, которая выполняется при щелчке по кнопке *Расчет количества прожитых дней*.

Введите дополнительную переменную для расчета количества дней. Дату рождения введите в формате D=”24/12/19..”. Для вывода результатов можно использовать метку, текстовое поле или метод Print.

Лабораторная работа 5.

Циклы

Существует два основных типа циклов:

- циклы со счетчиком (с известным числом повторений);
- циклы с условием, в которых действия повторяются до тех пор, пока выполняется определенное условие

В VB организации циклов с определенным количеством повторений используется оператор For ….Next.
Формат оператора For ….Next:

```
For Счетчик=Начало  TO Конец [Step Шаг ]
[операторы цикла]
[Exit For]
Next [Счетчик]
```

Проект Печать

Постановка задачи

Напечатать фразу «Изучаем циклы!» 15 раз с использованием оператора For ….Next.

Порядок действий

1. Расположите на форме одну командную кнопку.
2. Установите значения свойства Name для формы и кнопки такие, которые используются ниже в программном коде.
3. Установите значения свойства Caption для формы и кнопки соответственно *Циклы* и *Пуск.*
4. Составьте программный код для проекта:

```
Private Sub cmdПуск_Click()
Dim i As Integer
```

```
Cls
For i=1 To 15 Step 1
frmЦиклы.FontSize=9 + i
Print i; “Изучаем циклы”
Next i
End Sub
```

5. Запустите проект на выполнение. Проанализуйте результат, ответив на вопросы: «Почему перед текстом печатаются числа; почему изменяется размер шрифта?»
6. Сохраните проект под именем *Печать.vpb* в папке *Циклы*.

Проект Табулирование

Постановка задачи

Вычислить значения функции $y=x^2e^{-x}$ для всех x на интервале [a, b] с шагом h. Напечатать таблицу значений на форме. Ввод исходных данных осуществлять через окно *InputBox*.

Порядок действий

1. Расположите на форме одну командную кнопку.
2. Установите значения свойства *Name* для формы и кнопки такие, которые используются ниже в программном коде.
3. Установите значения свойства *Caption* для формы и кнопки соответственно *Функция f* и *Пуск*.
4. Составьте программный код для проекта. Самостоятельно заполните пропуски (……):

```
Private Sub cmdПуск_Click()
Dim a As Single, b As Single, h As Single
Dim x As Single, f As Single
With frmФункция
  ….. ‘ Задайте для формы следующие параметры: цвет формы-белый, шрифт-  ‘ Arial Cyr размер шрифта -12,  начертание – полужирный, цвет символов - синий
End With
a=Val(InputBox (“Введите начало диапазона”, “Начало”))
b=Val(InputBox (“Введите конец диапазона”, “Конец”))
h=Val(InputBox (“Введите размер шага”, “Шаг”))
Print “-------------------------------------------------”
Print “   x      !            y (x)          ”
Print “-------------------------------------------------”
For x=a To b Step h
F=x^2*exp(-Abs(x))
Print “ “, x; f
Next x
Print “-------------------------------------------------”
```

```
End Sub
```

5. Запустите проект на выполнение, задав в качестве исходных данных значения a=-5; b=5; h=0,5 (сначала лучше сохранить проект, т.к. при некоторых ошибках программа может зациклиться).
6. Сохраните проект под именем *Табулирование.vpb* в папке *Циклы*.

Проект Среднее

Постановка задачи

Составить программу для определения среднего арифметического введенных чисел. Количество чисел и сами числа вводятся с помощью окна *InputBox*.

Порядок действий

1. Расположите на форме одну командную кнопку, два текстовых поля и две метки (рис. 4.3).

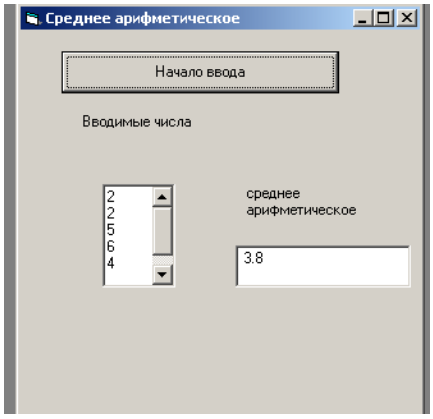

Рис. 4.3. Вид работающего приложения

2. Установите значения свойства *Name.* Для формы – *frmЦикл,* для кнопки – *cmdЦикл*, для первого тектового поля – *txtЧисла*, для второго текстового поля – *txtСреднее*.

Для первого текстового поля установите значение свойства *MultiLine-True* и для свойства *ScrollBars* – (*Vertical*).

Свойство *MultiLine*, равное *True*, для текстового поля означает, что текст разбит на строки, между которыми стоят символы перехода на новую строку и возврата к левому краю текста (коды ASCII этих символов 13 и 10).

3. Все остальные свойства, определяющие внешний вид проекта, установите по собственному усмотрению.
4. Составьте программный код для проекта.

```
Private Sub cmdПуск_Click()
Dim N As Integer, i As Integer ‘количество чисел, счетчик цикла
Dim sum As Single, sr As Single
Dim P As String, K As String
txtЧисла. Text =””: txtСреднее= “”
```

```
P=InputBox (“Сколько чисел?”, “Количество чисел”)
N=Val(P)
sum=0
For i=1To N
K=InputBox (“Введите”+Str(i)+ “число и нажмите кнопку ОК”,”Ввод очередного числа” )
txtЧисла. Text = txtЧисла. Text +K+Chr(13)+Chr(10) ‘ добавление чисел в
                                                    ‘текстовое поле
  sum= sum+Val(K)
Next i
  sr=sum/N
  txtСреднее.Text=Str(sr)
  End Sub
```

5. Сохраните проект под именем *Среднее.vpb* в папку *Циклы*.
6. Запустите проект на выполнение.
7. Измените программу так, чтобы выполнялся подсчет среднего арифметического только отрицательных чисел. Внимание: программа должна корректно работать при отсутствии отрицательных чисел.

Проект Максимум

Постановка задачи

Составить программу, которая определяет максимальное число из введенных чисел. Количество чисел и сами числа вводятся с помощью окна InputBox. Вид работающего приложения представлен на рисунке 4.4.

Порядок действий

1. Создайте папку *Максимум*, скопируйте в нее файл формы предыдущего проекта и откройте его.
2. Внесите изменения в соответствии с рисунком 4.4.
3. Измените значение свойства Name второго текстового поля на txtMax.

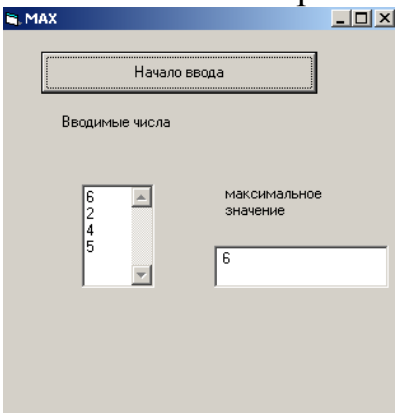

Рис. 4.4. Результаты расчетов

4. Разберите представленный ниже код. Здесь для первого сравнения в качестве начального значения переменной Max можно использовать

значение первого введенного числа, тогда сравнение целесообразно начать с i=2.

```
Private Sub cmdЦикл_Click()
Dim N As Integer, I As Integer ‘количество чисел, счетчик цикла
Dim max As Single
Dim K As String
txtЧисла.Text = “”: txtMax = “”
N = Val(InputBox(«Сколько чисел?», «Количество чисел»))
K = InputBox(«Введите 1 число», «Ввод очередного числа»)
txtЧисла.Text = txtЧисла.Text + K + Chr(13) + Chr(10) ' добавление чисел в текстовое поле
max = Val(K)
For I = 2 To N
        K = InputBox(«Введите» + Str(i) + «число и нажмите кнопку ОК», «Ввод очередного числа»)
        txtЧисла.Text = txtЧисла.Text + K + Chr(13) + Chr(10)
        If Val(K) > max Then max = Val(K)
Next i
txtMax.Text = Str(max)
End Sub
```

5. Измените код предыдущего проекта.
6. Сохраните проект под именем *Максимум.vpb* в папке *Циклы*.
7. Запустите проект на выполнение.

Вложенные циклы

Проект Таблица_Пифагора

Постановка задачи

Напечатать таблицу Пифагора (таблицу умножения). Вид работающего приложения представлен на рисунках 4.5, 4.6.

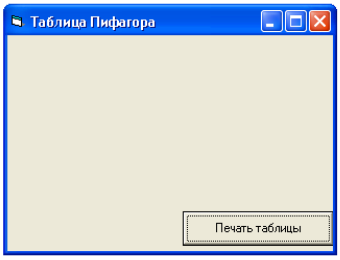

Рис. 4.5. Исходная форма

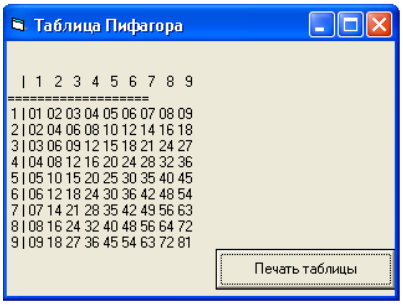

Рис. 4.6. Вид формы после нажатия кнопки «Печать таблицы»

Порядок действий

1. Расположите на форме одну командную кнопку.
2. Установите значения свойств *Name* для формы и кнопки такие, которые используются ниже в программном коде.
3. Установите значения свойства *Caption* для формы и кнопки в соответствии с рисунком 4.5 или 4.6.
4. Разберите программный код и наберите его:

```
Private Sub cmdPrint_Click()
Dim I, j As Integer
Print: Print
Print “    |   1   2   3   4   5   6   7   8   9”
Print “====================”
For I = 1 To 9
        Print I; “|”;
        For j = 1 To 9
                Print “ “;
                Print Format((I * j), “00”);
        Next j
Print
Next i
End Sub
```

5. Запустите проект на выполнение. Сравните результаты с рисунком 4.6, в случае несовпадения отредактируйте код.
6. Сохраните проект под именем *Таблица_Пифагора.vpb* в папке *Циклы*.

Циклы с условием

При программировании повторений далеко не всегда известно количество повторов. В этих случаях используют циклы с условием, которые могут быть реализованы с помощью операторов, которые мы сейчас изучим.

Можно выделить два типа циклов с условием:

- проверка условия осуществляется в начале цикла (предусловие);
- проверка условия осуществляется в конце цикла (постусловие).

Форматы операторов ***с проверкой условия в начале цикла:***

```
Do While Условие
Тело цикла
[Exit Do]
Loop
```

Тело цикла выполняется, если условие истинно, иначе осуществляется переход на оператор, расположенный после *Loop*. Если первая проверка условия даст результат «ложь», то цикл не выполнится ни разу.

Do Until Условие
Тело цикла
[Exit Do]
Loop

Тело цикла выполняется, если условие ложно, иначе осуществляется переход на оператор, расположенный после *Loop*. Если первая проверка условия даст результат «истина», то цикл не выполнится ни разу.

Форматы операторов с ***проверкой условия в конце цикла:***

Do While Условие
Тело цикла
[Exit Do]
Loop While Условие

Тело цикла выполняется до тех пор, пока Условие истинно, иначе выполнение цикла заканчивается.

Do
Тело цикла
[Exit Do]
Loop Until Условие

Тело цикла выполняется до тех пор, пока условие ложно, иначе выполнение цикла заканчивается.

Отличие этого оператора от оператора цикла с предусловием: проверка условия производится после очередного выполнения тела цикла. Это обеспечивает его выполнение хотя бы один раз.

Проект *Среднее*-2

Постановка задачи

Составить программу, которая определяет среднее арифметическое введенных чисел. Количество чисел заранее неизвестно. Числа вводятся с помощью окна *InputBox*. Ввод завершается занесением пробела в окно *InputBox* и щелчком мыши по кнопке *Cancel*. Окно должно иметь вид, представленный на рисунках 4.7, 4.8.

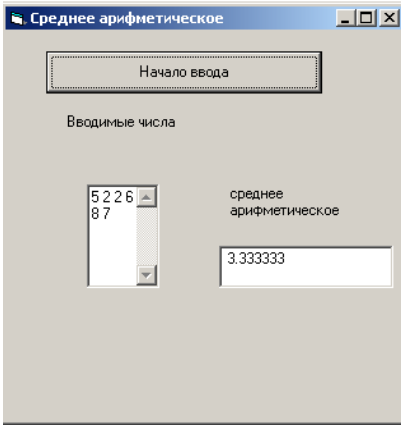

Рис. 4.7. Вид работающего приложения в случае корректного ввода исходных данных

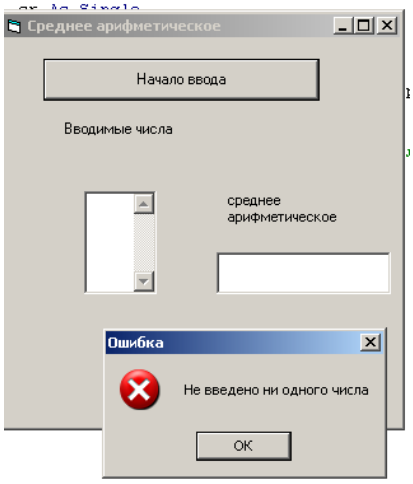

Рис. 4.8. Вид работающего приложения в случае некорректного ввода исходных данных

Порядок действий

1. Создайте папку *Среднее-2* и скопируйте в нее проект *Среднее-2.vpb*.
2. Откройте скопированный проект.
3. Измените программный код:

```
Private Sub cmdПуск_Click()
Dim N As Integer, I As Integer ‘количество чисел, счетчик цикла
Dim sum As Single, sr As Single
Dim P As String, K As String
txtЧисла. Text =””: txtСреднее= “”
sum=0: N=0
P=InputBox (“Введите число и нажмите кнопку ОК. Для завершения ввода
                                    ‘ щелкните по кнопке Cancel”)
Do Until P=”“
  N=N+1
  txtЧисла. Text = txtЧисла. Text +P+” “ ‘ добавление чисел в
                                                    ‘текстовое поле
  sum= sum+Val(P)
  P=InputBox(“Введите число и щелкните по кнопке ОК”)
Loop
If N=0 Then
  MsgBox “Не введено ни одного числа ”, 0 + 16,”Ошибка”
Else
  sr=sum/N
  txtСреднее.Text=Str(sr)
End If
  End Sub
```

4. Запустите проект на выполнение.
5. Сохраните проект.

Задачи для самостоятельного решения

1. Найти сумму всех двузначных чисел, кратных 3.
2. Одноклеточная амеба каждые 3 часа делится на 2 клетки. Определить, сколько клеток будет через 3, 6, 9, …, 24 часа.
3. Напечатать таблицу значений функции

$Y=x^2+2x-7$

Для всех значений x, изменяющихся с шагом h на интервале $[a,b]$, а также определить суммарное значение функции в данных точках.

4. Клиент поместил в банк S руб. За год банк увеличивает сумму помещенных в него денег на P %. Написать программу, которая определит, какой станет сумма вклада через N лет.
5. По очереди вводятся координаты N точек. Определить координаты точки, максимально удаленной от начала координат.

Лабораторная работа 6.
Модульное программирование. Процедуры и функции

При составлении сложных программ целесообразно разбивать их на несколько небольших программ (процедур).

Все процедуры делятся на два типа: процедуры типа Sub (подпрограммы) и процедуры типа Function (функции).

Общие процедуры, в отличии от процедур обработки событий, начинают работать не в ответ на какое-либо событие, а после их явного вызова из какого-нибудь места программы. После выполнения такой процедуры происходит автоматический возврат в то место программы, откуда процедура была вызвана.

Общая процедура может входить в состав модуля формы (в файл .frm) или в состав универсального модуля (в файл .bas).

Чтобы создать процедуру, достаточно в Окне программного кода для данной формы выполнить команду *Tools, Add Procedure***.**

В то место, откуда должен быть сделан вызов процедуры, помещается оператор вызова.

```
Call  ИмяПроцедуры([СписокПараметровВызова])
                    или
ИмяПроцедуры([СписокПараметровВызова])
```

Проект *Процедура*

Постановка задачи

Определить максимальное число из трех чисел, используя общую процедуру определения максимального и числа из двух чисел (рис. 6.1).

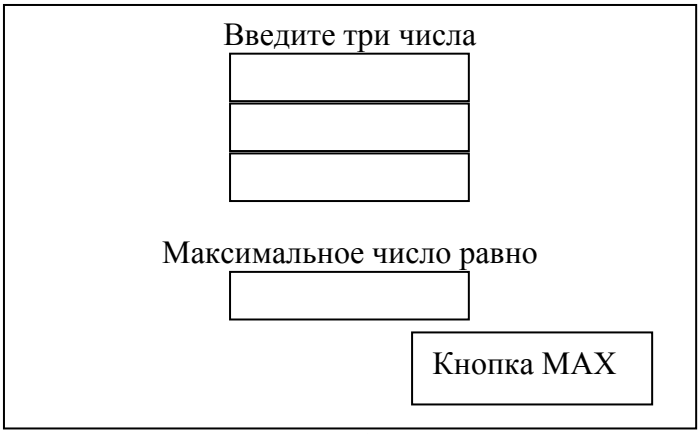

Рис. 6.1. Проект приложения

1. Определить максимальное число из трех чисел, используя общую процедуру определения максимального и числа из двух чисел. Общую процедуру разместить в программном модуле формы.
2. Расположите на форме объекты.
3. Установите значения свойства Name для текстовых полей – txtX, txtY, txtZ, txtMAX.
4. Установите значения свойства Name для кнопки cmdMax.
5. Установите значения свойства Caption для формы, кнопки и меток в соответствии с рисунком 6.1.
6. Создайте общую процедуру. Для размещения ее в программном модуле формы используйте один из способов, описанных выше:

```
Private Sub max2 (a As Single, b As Single, Max As Single)
If  a>b Then Max=a Else Max=b
End Sub
```

Здесь a, b – входные параметры, а Max – выходной.

7. Напишите процедуру, которая будет выполняться при щелчке по кнопке:

```
Privat Sub cmdMax_Click()
Dim  ……
X=Val(txtX.Text)
Y=Val(txtY.Text)
Z=Val(txtZ.Text)
Call Max2(x,y,m)
Call Max2(m,z,m)
txtMAX.Text=Str(m)
End Sub
```

8. Проверьте результат проекта и сохраните проект.

Функции

Результатом выполнения функции является *значение* какой-нибудь величины. В функции, в отличии от процедуры, нет выходных параметров; ее входные параметры называют *аргументами*.

Функция включается в состав выражения, значение которого используется в том или ином месте программы. *Функция возвращает свое значение*.

Формат определения функции:

```
[Private|Public] [Static] Function ИмяФункции ([СписокАргументов]) As Type
[Блок операторов]
ИмяФункции=Выражение
[Блок операторов]
[Exit Sub]
[Блок операторов]
End Function
```

Здесь Private – необязательное ключевое слово, которое означает, что данная функция может быть вызвана только из того модуля или формы, в которых она описана.

Public – необязательное ключевое слово, которое означает, что данная функция может быть вызвана из любого модуля или формы.

Static свидетельствует о статусе локальных переменных, т. е. переменных, объявленных внутри этой функции. При наличии этого слова локальные переменные будут сохранять свои значения между последовательными применениями данной функции, при отсутствии – не будут.

Аргументы перечисляются через запятую.

Type – это тип значения, возвращаемого функцией.

Проект *Функции*

Постановка задачи

Определить максимальное число из трех чисел, используя функцию определения максимального числа из двух. Функцию разместить в программном модуле формы.

Порядок действий

1. Откройте предыдущий проект и скопируйте файл формы в папку *Функции*.
2. Измените программный код. Создайте функцию определения максимального числа из двух. Для размещения ее в программном модуле формы используйте описанный выше способ.

```
Function max2(a As Single, b As Single ) As Single
If a>b Then max2=a else max2=b
End Function
```

Здесь max 2- имя функции, a, b - аргументы.

3. Измените процедуру, которая будет выполняться при щелчке по кнопке:

```
Private Sub cmdMax_Click()
Dim x As  Single, y As Single, z As Single,max As Single
x=Val(txtX.Text)
y=Val(txtY.Text)
z=Val(txtZ.Text)
max=max2(x,y)
max=max2(max,z)
txtMAX.Text=str(max)
End Sub
```

4. Проверьте правильность работы проекта. Сохраните изменения.

Задачи для самостоятельного решения (процедуры с параметрами)

1. Два треугольника заданы своими сторонами. Определить площадь большего треугольника по формуле Герона и найти максимальное из двух значений, задав соответствующие процедуры.
2. На плоскости заданы координатами пять точек. Найти расстояние до самой удаленной от начала координат точки.

Задачи для самостоятельного решения (функции)

1. Даны две дроби:

$$\frac{A}{B} \text{ и } \frac{C}{Д},$$

где (A, B, C, $Д$ – натуральные числа).

Написать код процедуры для деления дроби на дробь. Написать код функции для деления дроби на дробь.

Написать код процедуры для сложения дроби с дробью. Написать код функции для сложения дроби с дробью.

Написать код процедуры для вычитания дроби с дробью. Написать код функции для сложения дроби с дробью.

2. Написать программу для нахождения суммы большего и меньшего из трех чисел.
3. Вычислить площадь правильного шестиугольника со стороной a, используя подпрограмму вычисления площади треугольника.
4. На плоскости заданы своими координатами n точек. Составить программу, определяющую, между какими из пар точек самое большое расстояние. Указание. Координаты точек занести в массив.

Список литературы

1. Гусева, О.Л. Практикум по Visual Basic. – М.: Финансы и статистика, 2007.
2. Якушева, Н.М. Введение в программирование на языке Visual Basic.Net: учеб. пособие. – М.: Финансы и статистика, 2006.

ОГЛАВЛЕНИЕ