Министерство образования Республики Беларусь

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Экономика и управление на транспорте»

ПРОГРАММИРОВАНИЕ НА VBA

Лабораторный практикум

по дисциплине «Информатика»

для студентов специальностей

1-27 01 01 «Экономика и организация производства»,

1-37 01 08 «Оценочная деятельность на автомобильном транспорте»,

1-27 02 01 «Транспортная логистика»

Учебное электронное издание

Минск 2010

УДК 004.438

Составители:

И.И. Краснова, Н.В. Стефанович

Рецензенты:

А.А. Косовский, зав. кафедрой «Инновационный менеджмент» РИИТ БНТУ, кандидат экономических наук, доцент;

А.В. Сак, зав. кафедрой «Экономика» БГУИР, кандидат экономических наук, доцент

Задача данного лабораторного практикума состоит в том, чтобы познакомить студентов с современной системой проектирования Visual Basic (Beginner's All-purpose Symbolic Instruction Code).

Белорусский национальный технический университет
пр-т Независимости, 65, г. Минск, Республика Беларусь
Тел.(017) 293-91-97 факс (017) 292-91-37
Регистрационный № БНТУ/АТФ21–5.2010

СОДЕРЖАНИЕ

Лабораторная работа № 1
СИСТЕМА ПРОЕКТИРОВАНИЯ VISUAL BASIC 6.0

Цель работы. Ознакомиться с системой проектирования Visual Basic 6.0.

1.1. Запуск Visual Basic

Visual Basic (Beginner's All-purpose Symbolic Instruction Code) – система проектирования, которая позволяет легко создавать Windows-приложения.

Visual Basic запускается как любое Windows-приложение:

1. С рабочего стола двойным нажатием левой клавиши мыши на пиктограмме Visual Basic (рис. 1).

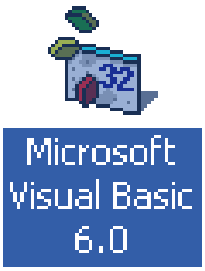

Рис. 1

2. Из подменю **Программы** в меню Windows **Пуск** (рис. 2).

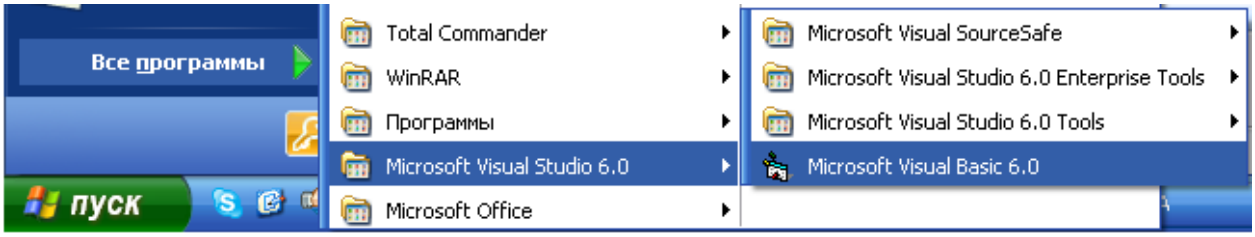

Рис. 2

После запуска на экране появляется Главная панель проекта Visual Basic (рис. 3).

Рис. 3

Вверху находится окно *New Project* – Новый проект, в окне – три закладки New, Existing и Resent (новый, существующий, недавний).

Выбрав пиктограмму *Standard EXE,* дважды щелкнув мышью, попадаем на *Главную панель проекта* (рис. 4).

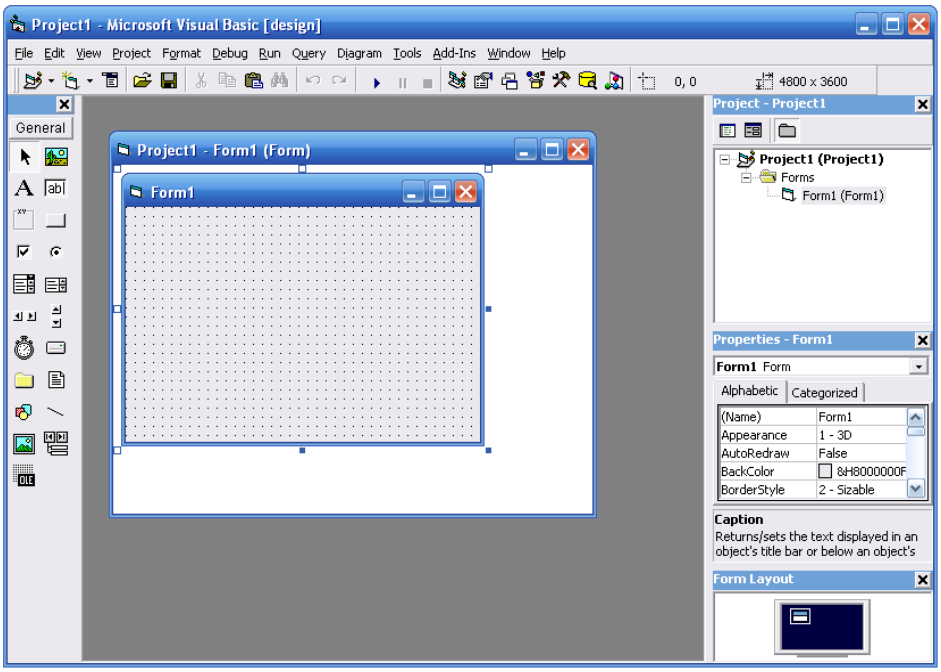

Рис. 4

В строке заголовка записано название проекта и его экранной формы. Под строкой заголовка расположены две строки – строка *Главного меню* и строка *Линейки инструментов.* Под меню и линейкой инструментов на *Главной панели проекта* располагается большое окно, которое может иметь различный вид в зависимости от установки.

1.2. Строка Главного меню

Как и на большинстве панелей систем, разработанных фирмой Microsoft, строка *Главного меню* состоит из слов, каждое из которых означает опцию (или пункт) этого меню. Многие опции могут быть знакомы по работе с другими приложениями фирмы Microsoft.

Меню **File** (Файл) содержит команды, типичные для этой опции в других приложениях, например: *Print* (печать), *Exit* (завершение работы). Есть среди команд этого меню и такие, которых нет в других приложениях: *создание* нового проекта, *открытие* уже существующего проекта, *сохранение* текущего проекта.

Меню **Edit** (Правка) содержит команды редактирования, например: *вырезка, копирование, вставка, поиск, откат, восстановление.*

Меню **View** (Вид) позволяет раскрыть на *Главной панели проекта* инструментальные *окна,* необходимые для работы.

Меню **Project** (Проект) содержит команды, позволяющие управлять проектом, например, добавить в текущий проект новый элемент *(форму, файл, модуль, элемент управления* и т.д.)

Меню **Format** (Формат) позволяет *форматировать* объекты, размещенные на экранной форме, например, изменять их размеры, выравнивать размеры выделенной группы объектов и т. д.

Меню **Debug** (Отладка) содержит команды, предназначенные для поиска ошибок в программе – отладки программы.

Меню **Run** (Выполнить) содержит команды, с помощью которых можно управлять режимом выполнения программы (запустить программу, прервать ее работу, завершить выполнение программы).

Меню **Tools** (Инструменты) содержит команды создания новых *процедур* и *функций,* открытия *Окна редактора меню,* изменения параметров настройки среды проектирования.

Меню **Add-lns** (Добавление утилит) позволяет расширять возможности среды проектирования включением в нее дополнительных возможностей.

Меню **Window** (Окно) позволяет изменять расположение окон на *Главной панели* по усмотрению.

Меню **Help** (Справка) позволяет обращаться к *Справочной системе* Visual Basic.

1.3. Главная панель, наборы инструментов

Кнопки *Линейки инструментов* выполняют те же действия, которые выполняют команды некоторых меню. Если установить указатель мыши на пиктограмме и задержаться, то под кнопкой появится маленькое окошко, которое называется *окном указателя.* В нем можно прочитать название команды.

К стандартной линейке инструментов, которая входит в состав исходной конфигурации Главной панели проекта, можно добавить дополнительные наборы инструментов (рис. 5).

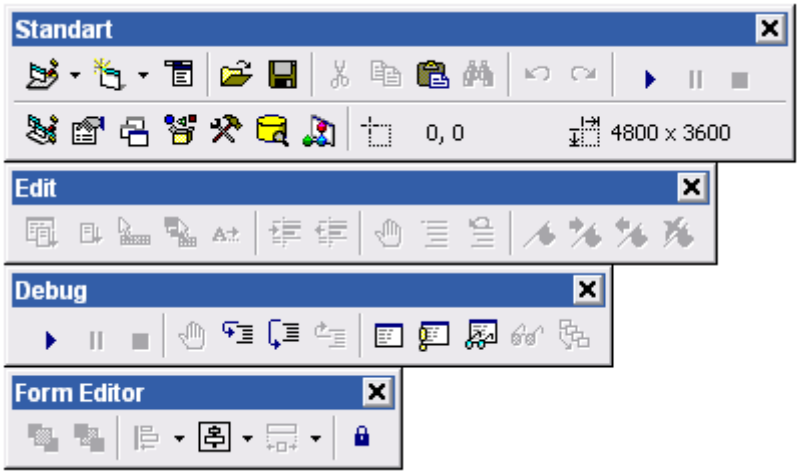

Рис. 5

Рассмотрим основные окна среды проектирования: Окно инструментов (Tollbox), Окно формы (Object), Окно проекта (Project Explorer), Окно кода (Code), Окно свойств (Properties), Окно просмотра характеристик (Object Browser) и Окно расположения формы (Form Layout).

Кроме перечисленных окон в среде проектирования Visual Basic есть и другие окна. В частности, Окно Редактирования формы (Form Editor), Окно Редактирования кода (Edit), Окно Отладчика программ (Debug).

В процессе запуска и отладки программы появляются: Окно немедленного выполнения (Immediate), Окно слежения (Watch), Окно локальных переменных (Locals).

Окно инструментов Tollbox представлено на рис. 6. На рисунке показан минимальный комплект инструментов, но его можно значительно расширить и изменить. Дополняется комплект инструментов из окна *Components {Дополнительные элементы)* из меню *Project/Components* или путем выделения правой кнопкой мыши поля элементов управления в нижней части *Toolbox (Панели элементов).* В дополнительном комплекте есть инструменты для создания объектов самого разного назначения. С помощью инструментов панели создаются объекты управления на экранной форме проекта.

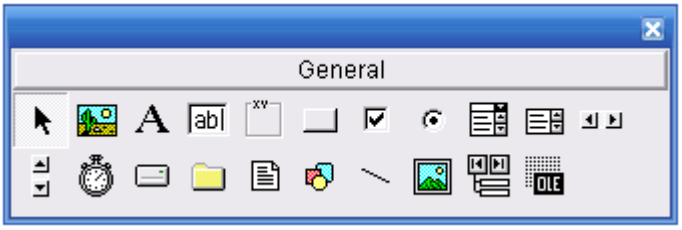

Рис. 6

1.4. Создание объектов управления

Элемент управления – это визуализированное средство для создания объектов управления на экранной форме. Элементы управления, которые представляют кнопки в окне, – это не сами объекты, а только их образцы, шаблоны. Выбрав тот или иной образец, можно создать его копию на экранной форме. Эту копию называют экземпляром класса объектов управления. Таких копий можно создать сколько угодно – точнее, столько, сколько может вместить экранная форма.

Создать объект управления с помощью инструмента панели можно двумя способами.

Способ 1. Двойной щелчок мыши по кнопке инструмента на панели вызывает появление в центре экранной формы объекта управления данного класса.

Способ 2. Одиночный щелчок мыши по кнопке выбранного инструмента на панели вызывает его выделение. После этого указатель мыши помещается в то место формы, куда предполагается поместить левый верхний угол объекта. Теперь с помощью мыши с нажатой левой клавишей можно "растянуть" размеры объекта, переместить его по полю.

1.5. Экранная форма

Окно формы Object показано на рис. 7. Заготовка экранной формы появляется на *Главной панели проекта* в особом окне, которое называется *Окном экранной формы (Form)*. Открыть *Окно экранной формы* можно либо с помощью кнопки *Объект (View Object)* на линейке инструментов, либо с помощью команды

View/Object (Вид/Объект). Вначале экранная форма пуста, а список ее свойств содержит лишь те значения, которые установлены системой по умолчанию. По умолчанию Visual Basic устанавливает экранной форме имя Form1 в качестве значения свойства (Name). А всему проекту по умолчанию присваивается имя Project1.

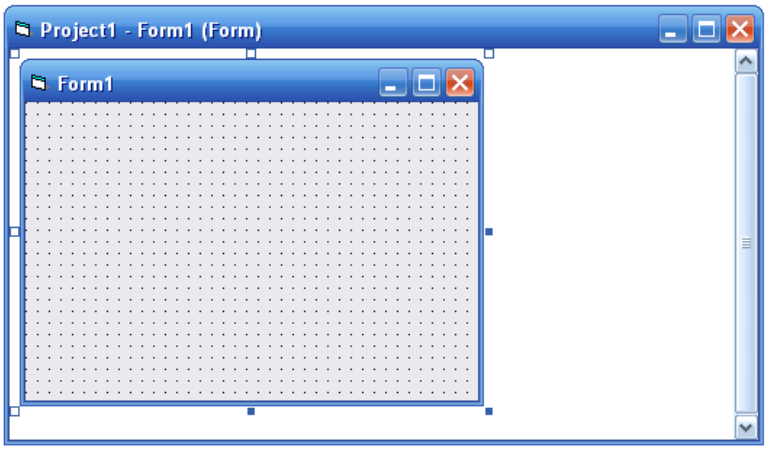

Рис. 7

Далее выполняется заполнение экранной формы, и, прежде всего, устанавливаются размеры экранной формы. Существует несколько способов:

1. Можно "ухватить" мышью правую или нижнюю сторону экранной формы и "на глазок" растянуть или сжать форму по горизонтали или вертикали.

2. Можно сделать это более точно, установив в окне *Properties* (Свойства) нужные значения *Width* (Ширина) и *Height* (Высота). Эти величины измеряются в особых единицах, которые называются твипами. Чтобы экранная форма, превратившись в окно работающего приложения, располагалась в нужном месте экрана монитора, нужно назначить установки значений свойств *Left* (Левый край) и Top (Верхний край).

Форму на время конструирования можно покрыть сетью точек и задать шаг перемещения, тогда всякое перемещение и растягивание объектов на форме происходит не плавно, а рывками – от точки к точке. Этим достигается аккуратное построение различных комбинаций на форме.

Установка и подключение сетки выполняется из окна *Options* (Параметры) (рис. 8).

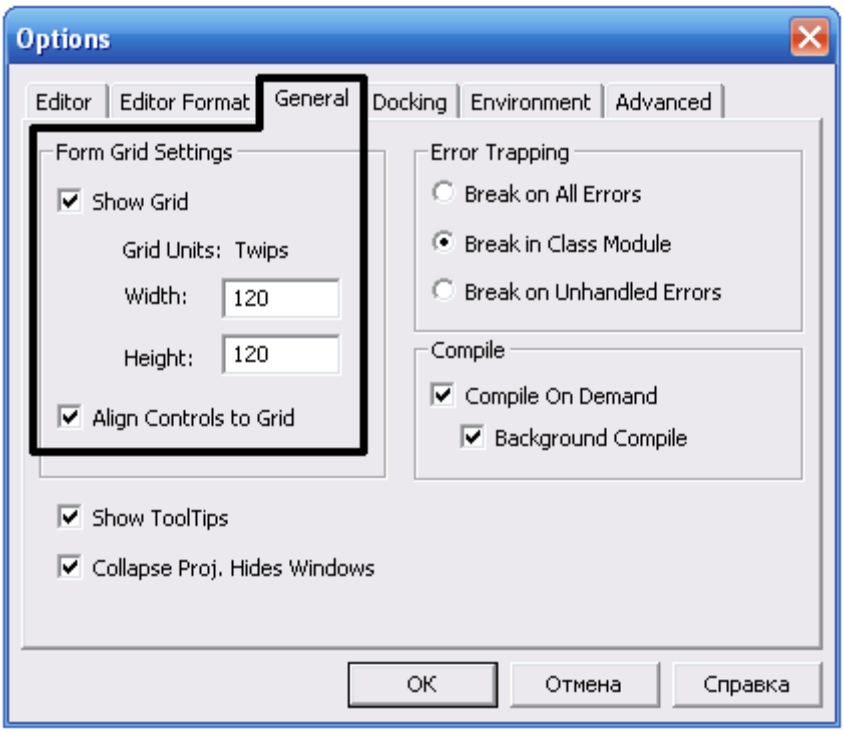

Рис. 8

Выход в окно выполняется по команде Tools/Options/General (*Сервис/Параметры/ Общие).*

1.6. Основные окна проекта

Окно проекта Project Explorer представлено на рис. 9. *Окно проводника проекта (Project Explorer)* содержит графическое представление содержимого проекта. Это может быть *дерево* или *список* всех файлов, которые входят в проект. Прежде всего, это файл экранной формы или несколько таких файлов, файл самого проекта, файлы программных модулей и файлы других видов. *Окно проводника проекта* раскрывается либо с помощью команды *View Project Explorer (Вид/Окно проекта),* либо через кнопку *Project Explorer (Окно проекта)* линейки инструментов *Главной панели (Standard).*

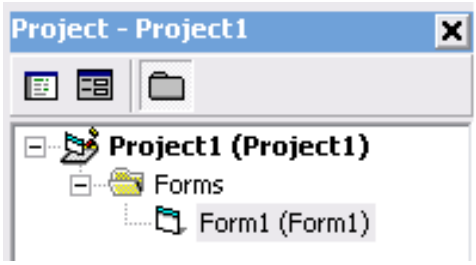

Рис. 9

Окно кода Code показано на рис. 10. *Окно программного кода* проекта раскрывается либо с помощью команды *View/Code (Вид/Программа),* либо через кнопку **Программа** линейки инструментов *Главной панели (Standard).*

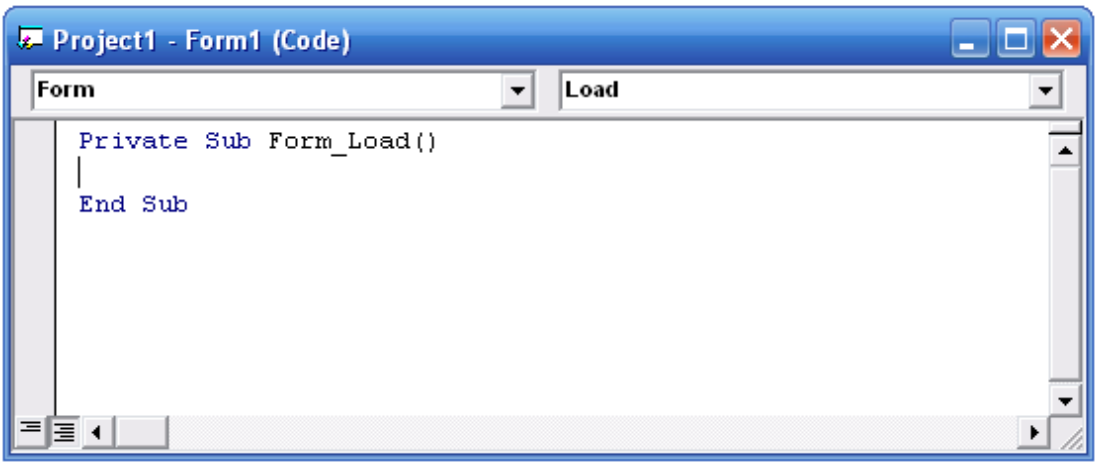

Рис. 10

Окно содержит два списка. Первый список *(General)* – это перечень *Объектов* настоящего проекта. Второй список *(Declaration)* – это перечень *Процедур* для объекта, выделенного в первом списке. Имена процедур во втором списке – это имена событий, которые могут происходить с выделенным объектом. В самом окне программного кода находится текст процедуры для обрабатываемого события. Выбрать событие можно, щелкнув мышью элемент правого списка.

Окно свойств Properties представлено на рис. 11. *Окно свойств объекта* можно раскрыть двумя способами: либо с помощью команды *View/Properties*

Windows (Вид/ Свойство) меню *Главной панели,* либо щелкнув кнопку *Properties Windows (Свойства)* линейки инструментов *Standard Главной панели.*

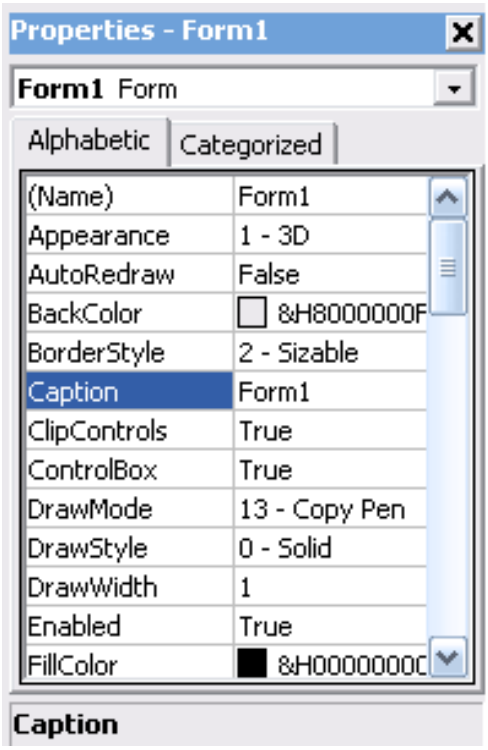

Рис. 11

В этом окне два списка. У верхнего открыта только одна строка, но с помощью кнопки прокрутки можно просмотреть этот список и выбрать нужную строку. Этот список содержит имена всех объектов, которые помещены на экранную форму (сама экранная форма тоже рассматривается как объект). Нижний список содержит перечень свойств, которым обладает объект, и их установки. Некоторые свойства выбираются из перечисляемого списка. Каждый объект имеет свой набор свойств, и их количество может доходить до полусотни.

Окно просмотра характеристик Object Browser показано на рис. 12. *Окно* можно раскрыть двумя способами: либо с помощью команды *View/Object Browser (Вид/Просмотр объектов)* меню *Главной панели,* либо щелкнув кнопку **Просмотр объектов** линейки инструментов *Главной панели.*

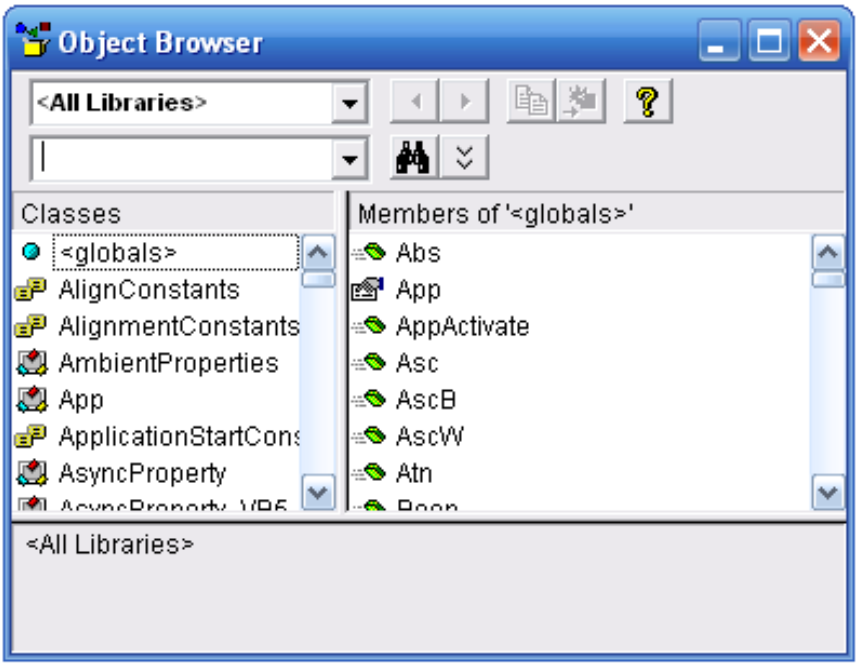

Рис. 12

В этом окне два списка:

1. Список Классов (Classes).

2. Список Характеристик выбранного Класса (Members of Class).

Выделив класс, сразу получаем доступ ко второму списку – *Характеристикам выделенного класса.* Сюда входят, например, *методы, свойства* и *события,* относящиеся к выбранному *Классу.*

В нижней части окна можно видеть текст со справочной информацией о характеристике, которая выделена в правом списке, краткая аннотация, в которой дается описание того, что собой представляет это событие.

Окно расположения формы Form Layout показано на рис. 13. *Окно* позволяет визуально контролировать расположение и размеры проектируемой формы на экране монитора.

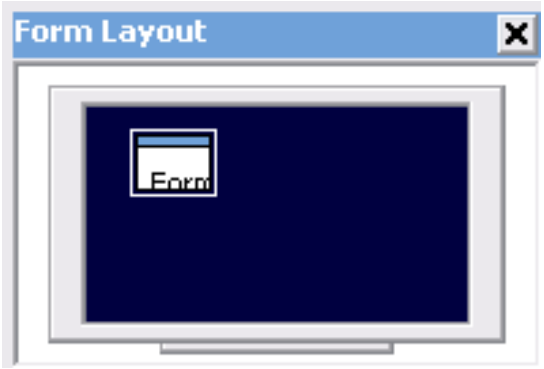

Рис. 13

1.7. Сохранение проекта, завершение работы

Важным моментом работы является сохранение проекта, его файлов с исходными текстами и конфигурацией.

Сохранение проекта с переименованием или без выполняется соответствующей командой из группы выпадающего меню **File** (рис. 4).

По умолчанию проект сохраняется в двух файлах. В файле **Form1.frm** сохраняется текст программы, макет формы, свойства объектов, в файле **Project1.vbp** сохраняется настройка среды программирования. В проекте могут находиться и другие файлы, поэтому лучше все файлы проекта помещать в отдельную папку.

Завершающим этапом проекта является компиляция – перевод файлов проекта в машинный код. При этом создается исполняемый файл с расширением *.exe. Компиляция выполняется по команде **Make Project1.exe...** из группы команд **File.**

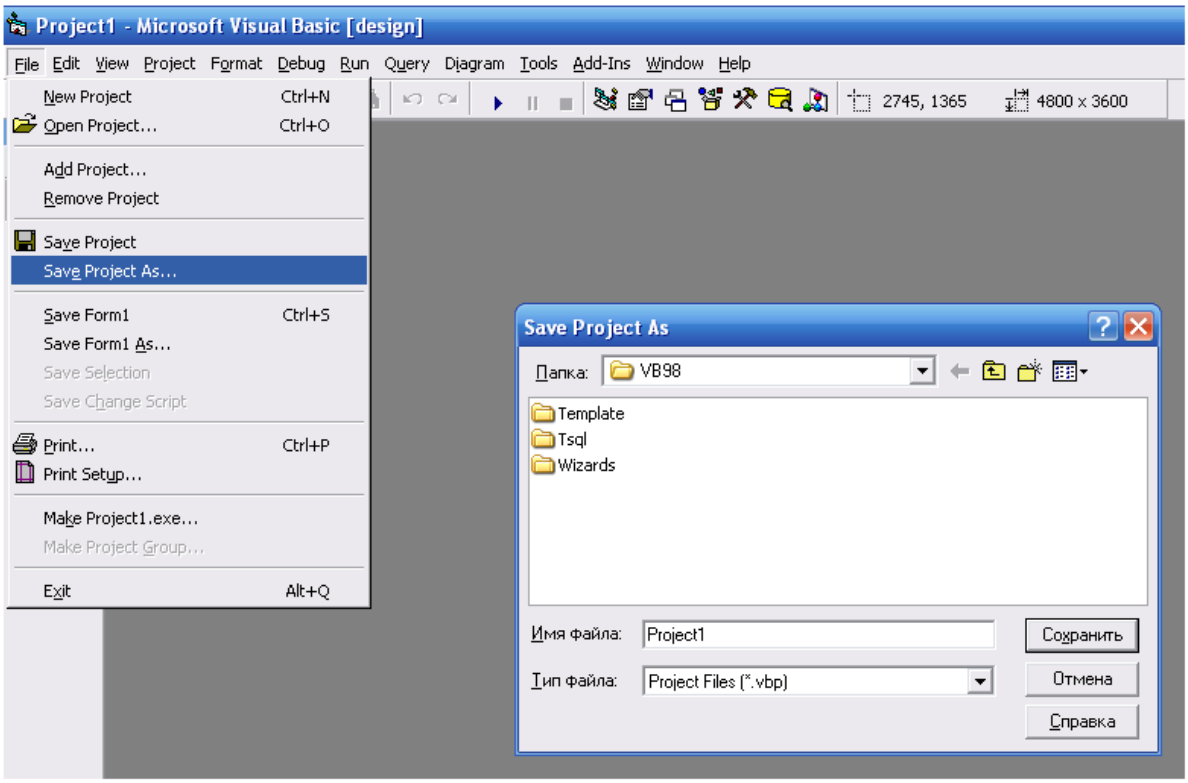

Рис. 14

Упражнение 1

- Загрузить среду проектирования Windows-приложений Visual Basic 6.0.
- Выйти на главную панель проекта.
- Просмотреть Главное меню, найти строку линейки инструментов.
- Научиться перемещать и видоизменять окна на главной панели инструментов.
- Изучить окно инструментов Toolbox, окно формы Object, окно проекта Project Explorer, окно свойств Properties.
- Сохранить файлы открытого проекта во вновь созданной папке "Первый_проект".

Лабораторная работа № 2
СОЗДАНИЕ ПЕРВОГО ПРИЛОЖЕНИЯ

Цель работы. Ознакомиться с технологией создания приложения в системе проектирования Visual Basic 6.0.

2.1. Этапы создания приложения

Процесс создания Windows-приложения состоит из пяти этапов:

1. Постановка задачи – составление по возможности точного и понятного словесного описания того, как должно работать будущее приложение. Это описание должно объяснить, как будет выглядеть форма (окно) этого приложения, определить порядок и формат ввода и вывода информации.

2. Разработка интерфейса – создание экранной формы со всеми находящимися на этой форме объектами и свойствами этих объектов.

3. Программирование – определение того, какие события будут происходить в процессе работы приложения, составление алгоритмов процедур для этих событий и написание программы (программных кодов) этих процедур.

4. Отладка программы – устранение логических ошибок и выполнение тестовых задач.

5. Сохранение проекта и, если необходимо, компиляция – превращение проекта в исполняемое приложение, способное работать самостоятельно за пределами среды проектирования.

2.2. Постановка задачи

В качестве примера создадим приложение для расчета площади стен комнаты. Исходными данными будут три размера комнаты: ширина, глубина, высота. После ввода исходных данных и нажатия кнопки "РАСЧЕТ" результат должен появиться в

текстовом поле "Площадь стен". Форма этого Windows-приложения показана на рис. 15.

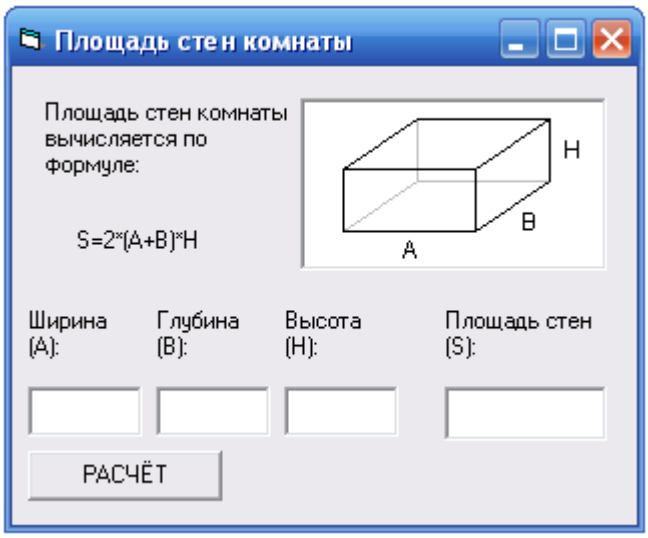

Рис. 15

2.3. Разработка интерфейса

Разработка интерфейса состоит из таких шагов:

- создание эскиза экранной формы;
- вход в среду проектирования Windows-приложений Visual Basic;
- создание экранной формы и установка значений свойств этой формы;
- создание на форме объектов управления и установка значений свойств этих объектов.

Как следует из постановки задачи, необходимо создать окно Windows-приложение, на котором будет 4 текстовых поля: 3 из них для ввода ширины, глубины и высоты комнаты и одно для вывода результата, т. е. площади стен этой комнаты. Кроме текстовых полей на экранной форме должны быть пояснения в виде заголовка, надписей, расчетной формулы и небольшого чертежа. Сигналом для расчета по формуле будет нажатие командной кнопки. Эскиз будущей формы показан на рис. 16.

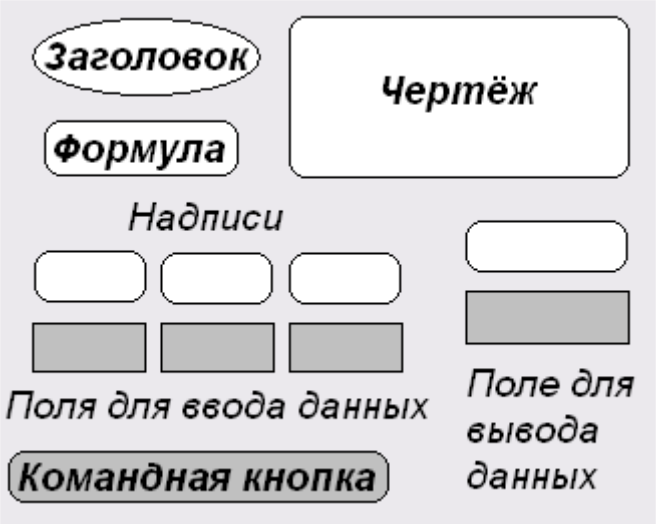

Рис. 16

Далее запускается система проектирования Visual Basic. На мониторе появляется *Главная панель проекта,* на ней нужно открыть окна для создания экранной формы (рис. 17): *TollBox (Окно инструментов), Form (Окно экранной формы), Properties (Окно свойств этой формы).*

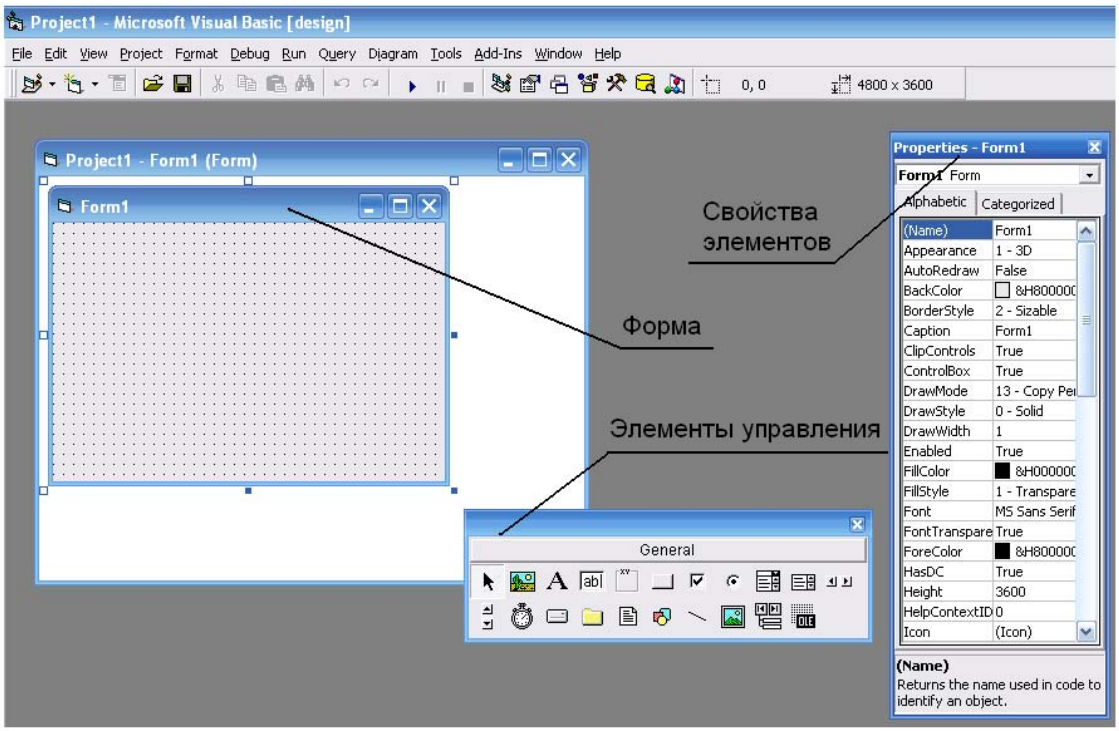

Рис. 17

Эти, а также некоторые другие окна могут появиться на Главной панели и без нашего участия, поэтому возможна корректировка окон.

Создание экранной формы. Если на *Главной панели проекта* нет *Окна экранной формы,* то открыть его можно, выбрав команду **Object** в меню **View** (рис.18).

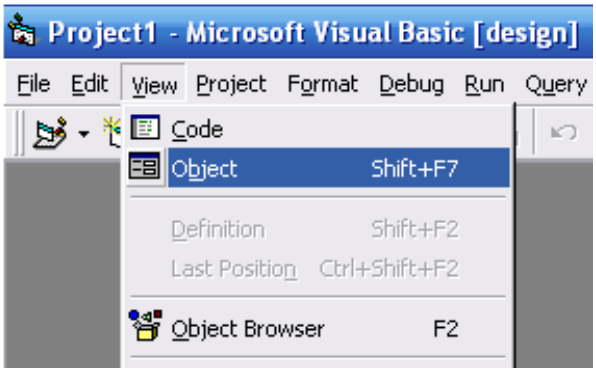

Рис. 18

Окно свойств Properties открывается командой **Properties Windows** *(Окно свойств* F4) в меню **View** (рис. 19).

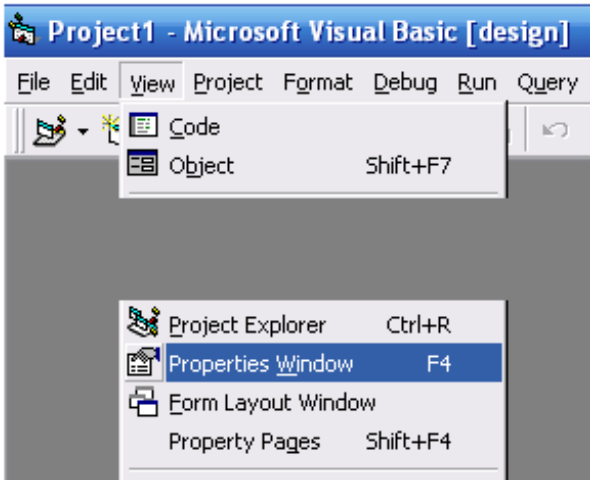

Рис. 19

После открытия этих окон можно начинать назначать свойства формы. Прежде всего, устанавливаются значения ее размеров – свойства Width (Ширина) и Height (Высота). Эти свойства можно установить с помощью окна ***Properties,*** но можно это

сделать, "растягивая" мышкой стороны экранной формы. При этом значения в таблице окна ***Properties*** будут изменяться автоматически. Положение формы на экране определяется свойствами Left (Левый край) и Top (Верхний край). Определим новое имя формы, новую надпись в строке заголовка формы и цвет фона формы. Этими свойствами являются соответственно: Name, Caption, BackColor. Значение свойства BackColor выбирается с помощью раскрывающейся панели с двумя закладками.

Далее на экранной форме создаются объекты управления и устанавливаются их свойства.

Окно Панель инструментов открывается командой **TollBox** *(Панель Элементов)* в меню **View** (рис. 20).

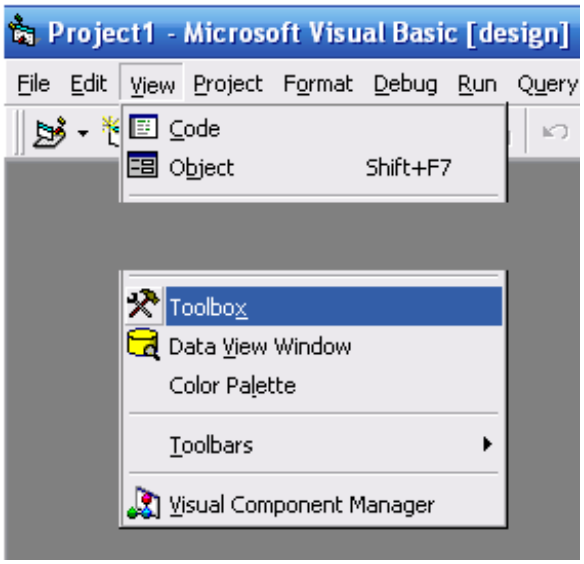

Рис. 20

Заполнение экранной формы начинаем с установки объектов *Label (Метка)*. Все надписи, заголовок и формула размещаются в нескольких таких объектах. В *окне Панель элементов* выбирается инструмент Label.

Для переноса на форму объекта Label щелкаем пиктограмму *Label (Метка)* в окне ToolBox *(Панель элементов)*. Помещаем указатель мыши в то место экранной формы, где будет находиться левый верхний угол будущего объекта. При нажатой левой клавише мыши "протащим" указатель в то место, где будет находиться правый

нижний угол будущего объекта. Это процесс будет сопровождаться растяжением пунктирной рамки – границы создаваемого объекта.

Размеры и положение появившегося прямоугольника можно подкорректировать. Делается это так. Прямоугольник окружен восемью маленькими квадратиками – маркерами "Ухватив" какой-либо из этих маркеров мышью, можно изменить размеры объекта. Установив курсор внутри прямоугольника, с помощью мыши можно переместить объект в пределах экранной формы, не меняя его размеров. Создаем еще пять таких объектов (рис. 21).

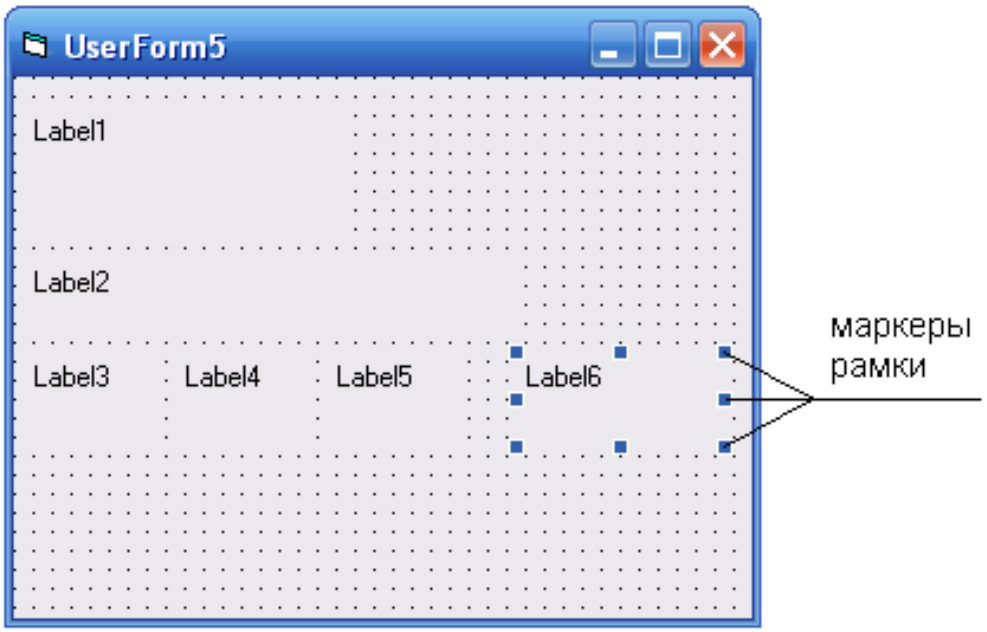

Рис. 21

Далее переносим на форму объекты *Поле*. Объект *Поле* на *Панели элементов* имеет пиктограмму, показанную на рис. 22.

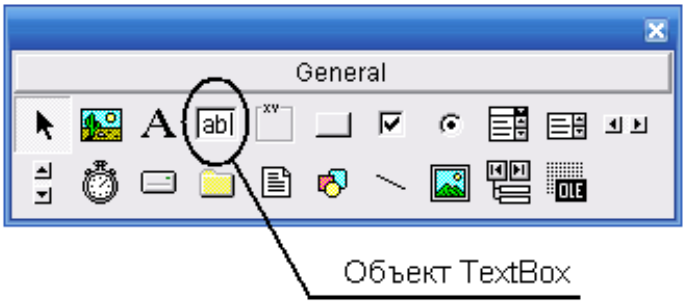

Рис. 22

Форма приобретает вид, представленный на рис. 23.

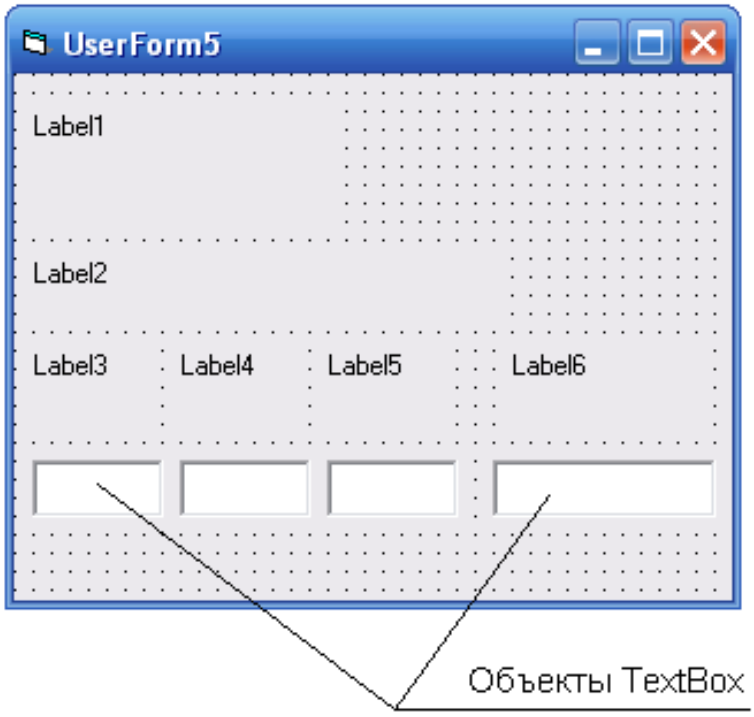

Рис. 23

Таким же образом помещаем на экранной форме два оставшихся объекта (рис. 24): *CommandButton (Кнопка)* и *PictureBox (Рисунок)*.

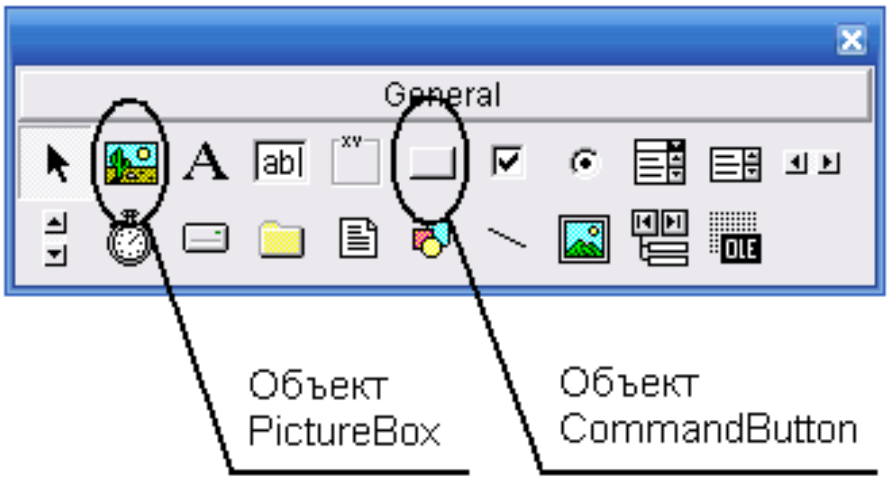

Рис. 24

Форма приобретает вид, представленный на рис. 25.

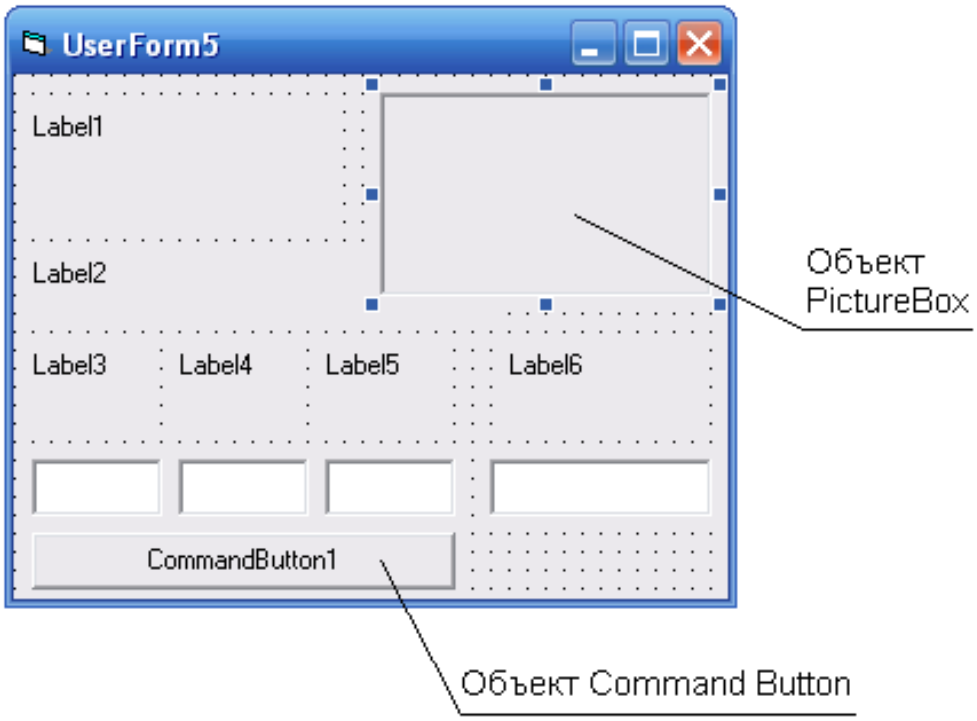

Рис. 25

Теперь наполним созданные объекты конкретным содержанием. Это содержание определяется установкой значений свойств перенесенных объектов.

2.4. Установка свойств объектов

В Properties *(Окне свойств)* на *Главной панели* видим список свойств активного объекта. Стоит щелкнуть мышью по другому объекту экранной формы, как к этому объекту переходит вся активность и мгновенно изменяется содержимое *Окна* Properties.

На примере объекта *Метка Label2* рассмотрим изменение свойств в *Окне* Properties (рис. 26).

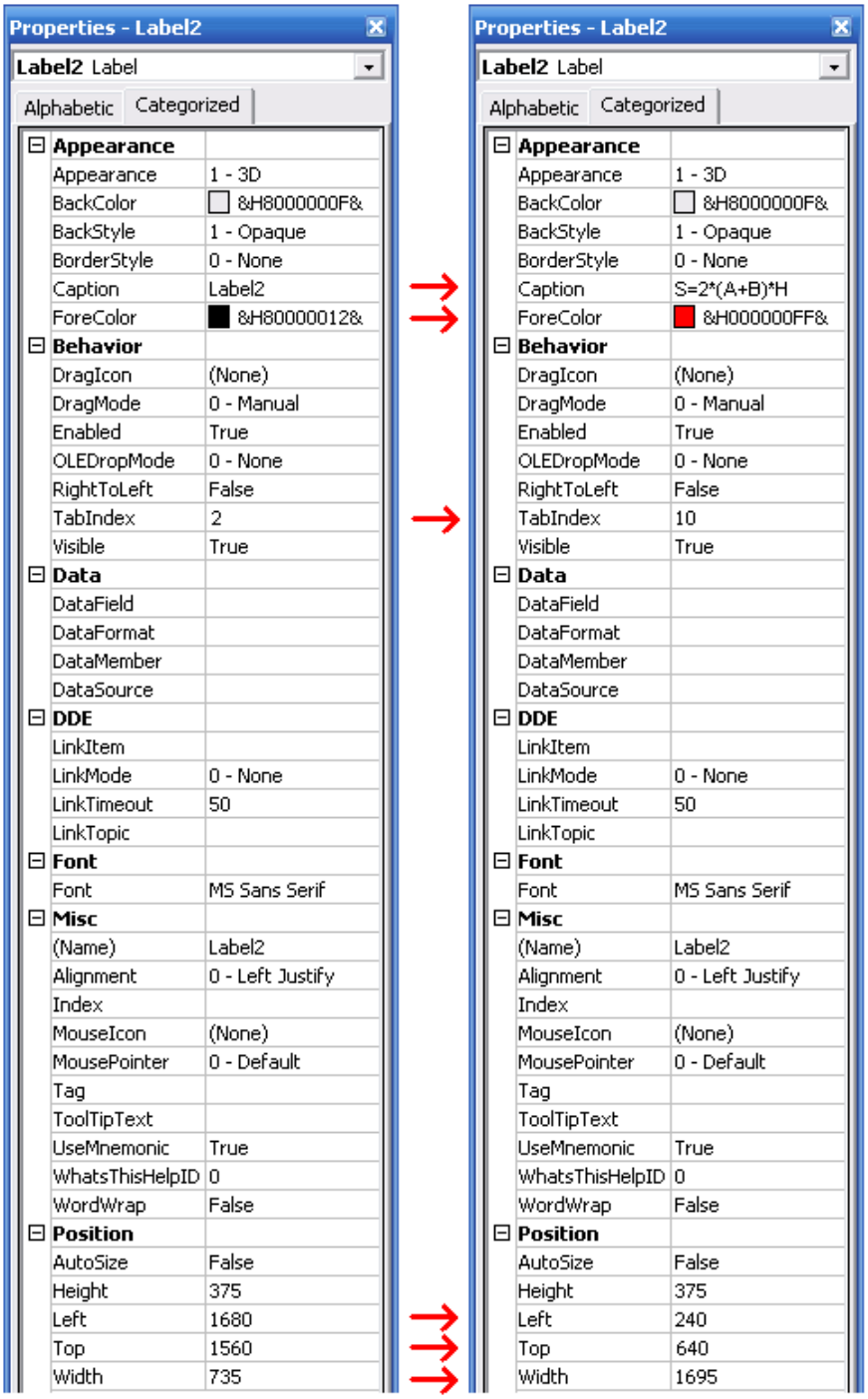

Рис. 26

Теперь устанавливаются значения четырех текстовых полей. У текстовых полей много свойств, аналогичных свойствам метки. Свойству Name даем такие значения: "Длина", "Ширина", "Высота", "Площадь". Вместо свойства Caption у текстового поля есть свойство Text. В текстовое поле можно вводить как очень большой объем информации, так и одно число.

Главной характеристикой объекта *CommandButton (Командная кнопка)* является не какое-нибудь свойство, а событие. Оно заключается в щелчке мышью по этой кнопке. Свойству (Caption) присваивается значение в виде слова "РАСЧЕТ".

В заключение устанавливается значение всего одного свойства объекта *PictureBox (Рисунок)* – свойство Picture. Этим значением должен быть графический файл с рисунком, находящийся на компьютере.

После установки значений свойств объектов экранная форма приобретет вид, заданный в начале примера создания приложения (см. рис. 15).

2.5. Программирование

Составление алгоритма и написание программы – это второй и главный этап проектирования приложения в среде Visual Basic. В составляемом приложении есть только одно событие: щелчок мышью по командной кнопке. Именно это событие должно запустить программу вычисления площади стен комнаты.

Алгоритм решения задачи вычисления площади стен комнаты следующий:

1. Ввести три числа: A, B, H – длину, глубину и высоту.
2. Найти площадь одной стены: S1=A*H.
3. Найти площадь другой стены: S2=B*H.
4. Удвоить сумму этих площадей: S=2*(S1+S2).
5. Вывести результат: число S – площадь всех 4 стен.

Пункты 2, 3 и 4 алгоритма можно, очевидно, объединить:

$$S=2*(A+B)*H.$$

Для написания программного кода и привязки его к событию *Нажатие кнопки* необходимо раскрыть *Окно программного кода* Code (рис. 27), которое открывается командой **Code** *(Программа)* в меню **View** (рис. 28).

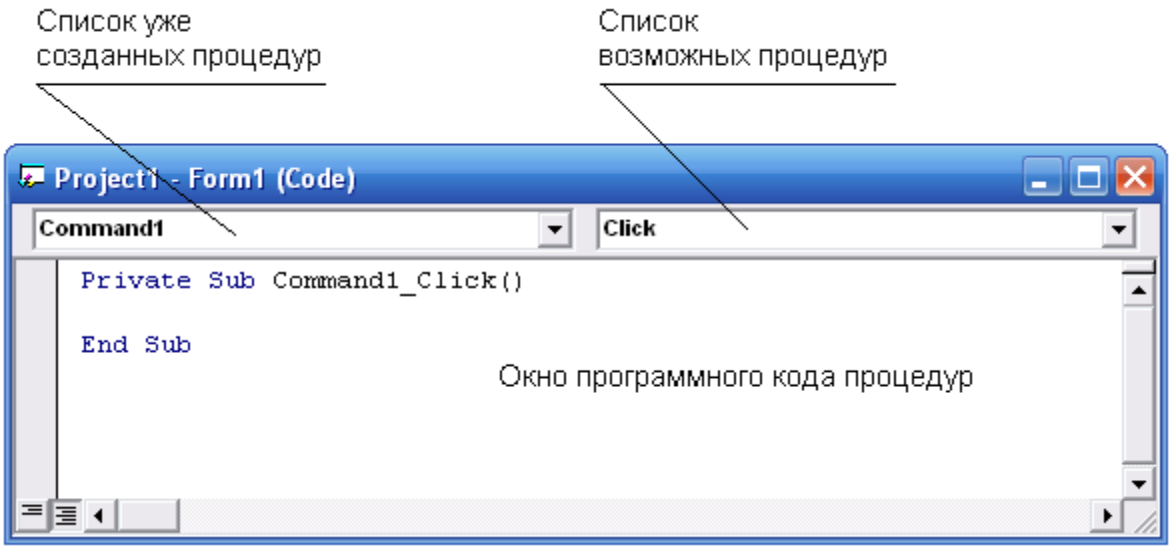

Рис. 27

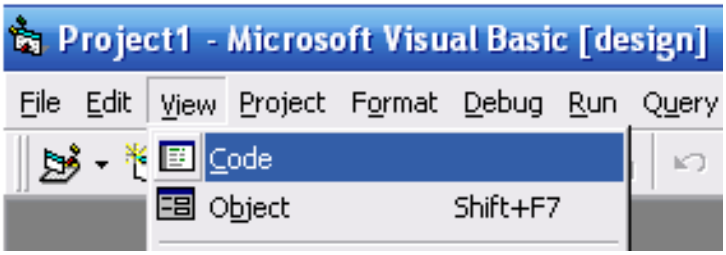

Рис. 28

Процедура – это фрагмент программного кода, с помощью которого решается какая-то локальная задача. Часто **(но не всегда!)** процедура вызывается событием. В рассматриваемом примере вычисление по формуле начинается после нажатия кнопки "РАСЧЕТ".

Из правого списка выбираем событие Click, из левого – объект Command-Button1. В *Окне программного кода* появляется заготовка процедуры, программы реакции на нажатие кнопки "РАСЧЕТ". Для завершения оформления процедуры необходимо ввести недостающие операторы (рис. 29).

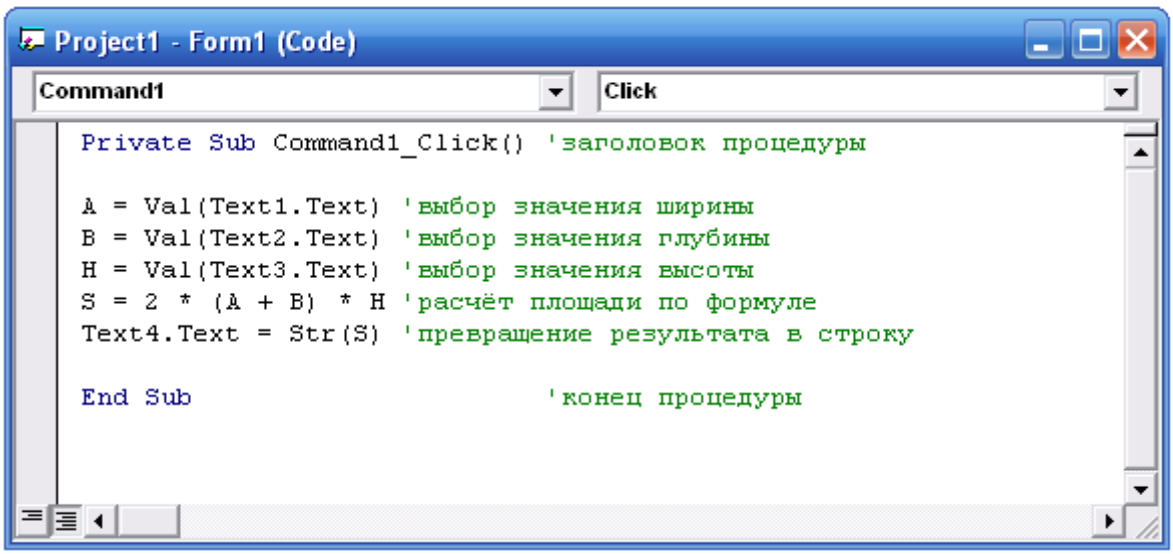

Рис. 29

Первая строка программы начинается со слов **Private Sub,** а заканчивается программа словами **End Sub;** это служебные слова языка. Последовательность строк кода соответствует последовательным шагам алгоритма решения данной задачи. Знак "=" обозначает присваивание переменной определенного значения. Знаки "*" и "+" обозначают операции умножения и сложения. Выражение Text1 Text обозначает значение свойства Text объекта Text1. Запись Val(X) означает, что значение переменной X преобразуется из *строки символов* в *число,* а запись Str (X) означает, что значение переменной X преобразуется из *числа* в *строку символов.*

Запускать программу можно:

- с помощью опции Run и команды **Start** *Главной панели проекта* (см. рис. 30);
- с помощью кнопки **Start** линейки инструментов *Главной панели проекта*;
- с помощью клавиши F5 клавиатуры.

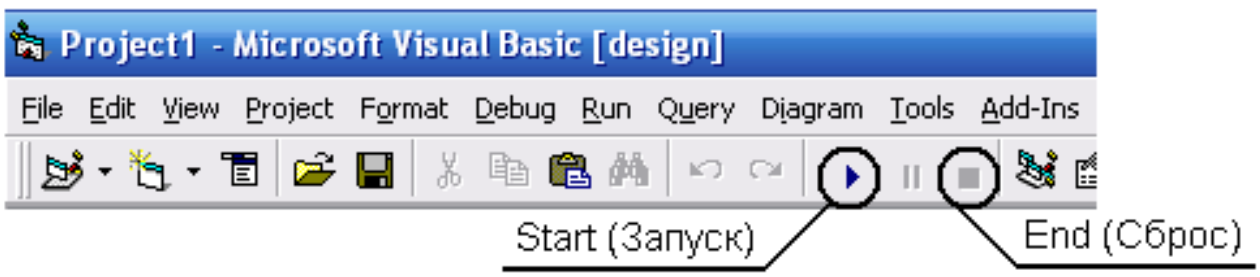

Рис. 30

Завершить работу программы можно тоже по-разному, например:

- с помощью кнопки **End** на линейке инструментов (см. рис. 30);
- с помощью стандартного элемента Windows – системной кнопки *закрытия окна* в правом верхнем углу окна приложения.

Отладка программы. Первая попытка запустить программу не всегда бывает успешной. Часто попытка запуска приводит к появлению сообщений системы Visual Basic об ошибках. В этом случае их нужно исправить – для этого Visual Basic предоставляет разнообразные средства отладки.

Сохранение экранной формы проекта в виде файлов. Когда программа отлажена, проверена, когда доведен интерфейс, выполняется заключительный этап – компилирование. Файлы проекта собираются и переводятся на машинный язык командой **Make Project1 .exe** из группы меню **File.** Таким образом, образуется исполняемый файл с расширением *.exe. После этого все файлы проекта сохраняются для дальнейшего использования.

Упражнение 2

- Загрузить среду программирования Visual Basic 6.0.
- Организовать на главной панели проекта окно экранной формы Form1, вынести панель инструментов ToolBox, окно свойств Properties. Разместить окна на панели проекта удобным образом.
- Перенести объекты с панели инструментов на форму и разместить их согласно схеме, изображенной на рис. 25.
- Используя окно свойств объектов, просмотреть свойства вынесенных на форму объектов, откорректировать свойства и придать форме вид, максимально приближенный к виду, изображенному на рис. 15.
- Открыть окно программного кода, создать процедуру Command1_Click согласно алгоритму задачи.
- Запустить приложение, проверить работу, правильность счета.

- Откомпилировать приложение под именем "Расчет площади".
- Сохранить файлы проекта во вновь созданной папке "Комната".

Задание для самостоятельной работы

Разработать программный интерфейс, реализованный посредством различных элементов управления и пользовательских диалоговых окон.

Лабораторная работа № 3
ПЕРЕМЕННАЯ И ЕЕ ЗНАЧЕНИЕ

Цель работы. Ознакомиться с типами переменных, их описанием в программе.

3.1. Имя и значение переменной

Необходимость использования переменных в программировании возникает по той же причине, что и в математике, - для обозначения величин, которые часто меняют свое значение.

Понятие переменных является, пожалуй, самым главным понятием в каждом языке программирования. У переменной есть две характеристики: имя и значение. Имя переменной уникально и неизменно, а значение может меняться в процессе выполнения алгоритма. Переменные позволяют с помощью небольшого числа инструкций предусмотреть большое число шагов исполнителя. Кроме этого, одна программа может решать задачу для разных значений исходных величин.

Имя переменной – это строка символов, которая идентифицирует переменную в программе. Переменные создаются программистом при написании программного кода. Он же дает им имена. Имена переменных создаются по определенным правилам:

1. Первым символом имени должна быть буква.

2. Остальные символы – буквы и цифры (прописные и строчные буквы различаются).

3. В имени можно использовать знак "_", но нельзя использовать знак "." (точка). Точка в языке используется в синтаксических конструкциях.

4. Число символов в имени может достигать 255, но лучше работать с короткими именами.

5. Имя не должно быть ключевым словом Visual Basic, в противном случае фиксируется ошибка и на экране не отображается подсказка.

Значение переменной – это данные, которые хранятся и обрабатываются компьютером. Это выполняется по-разному и зависит от того, к какому типу принадлежат данные. Типом данных называется способ хранения и представления данных в компьютере. В языке Visual Basic переменная может принадлежать к одному из более чем десяти типов.

Вот некоторые из них, наиболее употребляемые:

- тип **Byte.** Короткое неотрицательное целое число, оно занимает 1 байт памяти, его значение меняется в пределах от 0 до 255; тип Integer. Целое число, оно занимает 2 байта памяти, его значение меняется в пределах от -32768 до 32767;
- тип **Long.** Длинное целое число. Значение переменной этого типа занимает 4 байта памяти и меняется в пределах от -2147483648 до 2147483647;
- тип **Single.** Десятичное число обычной точности, оно занимает 4 байта памяти, его значение меняется в пределах от 1.401298E-45 до 3.402823E+38 (по модулю);
- тип **Double.** Десятичное число двойной точности, занимает 8 байт памяти и меняется в пределах от 4.94065645842147E-324 до 1.79769313486232E+308 (по модулю);
- тип **String.** Переменная строкового типа и текстовая. Значением переменной этого типа является строка любых символов, длина которой может достигать двух миллиардов. Слева и справа строка обрамляется кавычками;

- тип **Variant.** Произвольное значение. Переменная этого типа может иметь любой размер. Но за это надо платить дорогую цену - объем памяти, занимаемой значением переменной этого типа, бывает разным, но не менее 16 байт!

Тип переменной может в программе и не объявляться. В этом случае по умолчанию он будет установлен самой системой Visual Basic и будет соответствовать типу **Variant.** Но хорошим тоном в программировании считается обязательное объявление типа каждой переменной.

Объявлять тип созданной переменной можно несколькими способами, самый распространенный - с *использованием оператора определения переменной.*

Оператор определения переменной записывается с помощью строки программного кода, которая устанавливается в начале текста программного кода и имеет следующий синтаксис:

```
Dim Имя Переменной [As Тип Переменной]
```

Здесь **Dim** и **As** – ключевые слова языка, с помощью которых записывается данный оператор. Назначение этого оператора – объявить переменную, то есть задать ее имя и тип.

Пример.

```
Dim MyName As String
Dim Nmb As Integer
Dim AAA, BBB, CCC As Double
Dim Str_ As String* 12
Dim My_book
Dim X As Singlr, NmbX As Integer, XX As Double
```

3.2. Оператор языка

Для присвоения переменной некоторого значения используется оператор присваивания. **Оператор** – это такая синтаксическая единица языка

программирования, которая используется в программе для выполнения отдельного предписания. Операторы делятся на две категории. К первой относятся алгоритмические операторы, ко второй – функциональные

Алгоритмические операторы – это такие операторы, которые используются для организации последовательности выполняемых исполнителем действий. Важнейшие из них – операторы безусловных переходов, условные операторы, операторы циклов.

Функциональные операторы – это встроенные в язык функции и процедуры, с помощью которых производятся важные и распространенные действия, такие, как ввод данных, действия над числами. Любая программа состоит из последовательности операторов, которые записываются в соответствии со **строгими синтаксическими правилами:** компьютер не воспринимает программы, написанные с ошибками.

Оператор присваивания – один из самых распространенных. Синтаксическое правило для этого оператора выглядит так:

```
[Let] Имя Переменной = Значение Переменной
```

Прямоугольными скобками в правилах синтаксиса обрамляются такие конструкции, которые могут опускаться и отсутствовать. В подавляющем большинстве случаев ключевое слово **Let** перед именем переменной в операторе присваивания опускается.

При выполнении оператора присваивания переменная, имя которой указано слева от знака равенства, получает значение, равное значению выражения, находящегося справа от знака равенства.

Пример.

```
Dim Var_1 As Integer, Var_2 As Long
Dim Str_1,Str_2 As String* 12


Var1 =32000
Var2= -20000000
```

```
Str_1=’Проверка___’
Str_2=Str_1
```

Если необходимо разместить несколько операторов на одной строке, то они должны быть разделены двоеточием – *разделителем.* Если строку нужно разбить на несколько, то используется *символ переноса* – знак подчеркивания.

3.3. Пример Windows-приложения

Рассмотрим пример построения Windows-приложения (рис. 31).

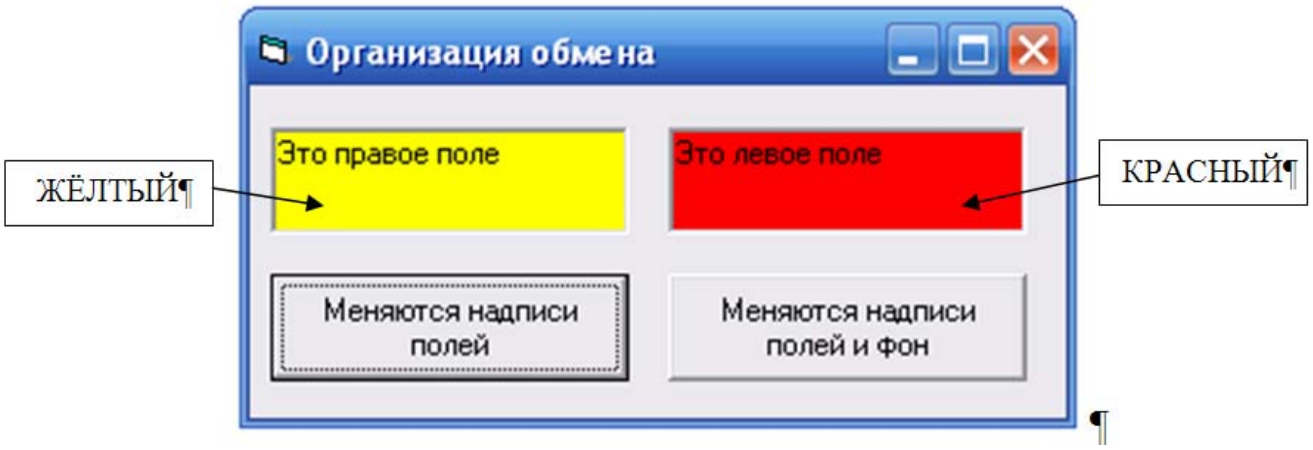

Рис. 31

В приложении программируются два события: нажатие левой кнопки "Меняются надписи полей" и нажатие правой кнопки "Меняются надписи полей и фон".

Левая кнопка меняет местами содержание окон без изменения цвета фона, правая кнопка меняет местами содержание окон с изменением цвета фона.

Процедура, срабатывающая при нажатии левой кнопки, имеет вид:

```
Private Sub Command 1_Click() 'программирование левой кнопки
Dim Str_1 As String, Str_2 As String 'описание переменных
```

```
'организация обмена
Str_1 = Text1 Text
Str_2 = Text2.Text
Text2.Text = Str_1
Text1.Text = Str_2
End Sub
```

Процедура, срабатывающая при нажатии правой кнопки, имеет вид:

```
Private Sub Command2_Click() 'программирование левой кнопки
Dim Str_1, Str_2 As String, Color_F, Color_L As Long
Str_1 = Text1 .Text
Str_2 - Text2.Text
Color_F = Text1 .BackColor
Text 1.Text = Str_2
Text2.Text = Str_1
Text1 .BackColor = Text2.BackColor
Text2.BackColor = Color_F
End Sub
```

Упражнение 3

• Сделать приложение по рис. 31. Код приложения должен иметь описание использованных переменных.

• Запустить приложение, проверить работу, правильность обмена.

• Дополнить приложение третьей кнопкой, по нажатию на которую выполнялся бы обмен содержаний окон, цвета фона окон и меток "Красный", "Желтый".

• Откомпилировать приложение под именем "Обмен".

• Сохранить файлы проекта в папке "Организация_обмена".

Задания для самостоятельной работы

1. По заданному радиусу *R* определить длину окружности *l*, ее диаметр *d* и площадь круга *S*.

$$l = 2\pi R; S = \pi R^2; d = 2R.$$

2. По заданному диаметру d и углу α определить радиус окружности R, длину дуги l и площадь сектора S.

$$R = \frac{d}{2}; l = \frac{\pi R\alpha}{180}; S = \frac{\pi R^2\alpha}{360}.$$

3. По заданным трем сторонам прямоугольного параллелепипеда a, b, c определить площадь его боковой поверхности $S_{бок}$, площадь полной поверхности S и объем V.

$$S_{бок} = 2(a + b)c; S = 2(ab + bc + ac); V = abc.$$

4. По заданному радиусу R определить диаметр шара d, площадь его поверхности S и объем V.

$$d = 2R; S = 4\pi R^2; V = \frac{4}{3}\pi R^3.$$

5. По заданным радиусу основания R и высоте цилиндра H определить площадь его боковой поверхности $S_{бок}$, площадь полной поверхности S и объем V.

$$S_{бок} = 2\pi RH; S = 2\pi RH + 2\pi R^2; V = \pi R^2 H.$$

6. По заданному радиусу R и высоте шарового сегмента H определить площадь сегментной поверхности S, объем шарового сегмента V и объем шарового сектора $V_{сек}$.

$$S = 2\pi RH; V = \frac{1}{3}\pi H^2(3R - H); V_{сек} = \frac{2}{3}\pi R^2 H.$$

7. По заданным радиусу основания R, высоте H и образующей L определить площадь боковой поверхности конуса $S_{бок}$, площадь его полной поверхности S и объем V.

$$S_{бок} = \pi RL; S = \pi RL + \pi R^2; V = \frac{1}{3}\pi R^2 H.$$

8. По заданным радиусам оснований R, r, высоте H и образующей L определить площадь боковой поверхности усеченного конуса $S_{бок}$, площадь его полной поверхности S и объем V.

$$S_{бок} = \pi(R + H)L; S = \pi(R + r)L + \pi R^2 + \pi r^2; V = \frac{1}{3}\pi(R^2 + Rr + r^2)H.$$

9. По заданным катетам прямоугольного треугольника a, b определить его гипотенузу c, периметр p и площадь S.

$$c = \sqrt{a^2 + b^2};\ p = a + b + c;\ S = \frac{1}{2}ab.$$

10. По заданным сторонам прямоугольника a, b определить квадрат его диагонали d^2, периметр p и площадь S.

$$d^2 = a^2 + b^2; p = 2(a + b); S = ab.$$

11. По заданному радиусу R описанной вокруг квадрата окружности определить его сторону a, периметр p и площадь S.

$$a = R\sqrt{2};\ p = 4a;\ S = 2R^2.$$

12. По заданному радиусу R описанной вокруг правильного треугольника окружности определить его сторону a, периметр p, площадь S.

$$a = R\sqrt{3}; p = 3a; S = \frac{3\sqrt{3}}{4}R^2.$$

Лабораторная работа № 4
ВЫРАЖЕНИЯ И ФУНКЦИИ

Цель работы. Изучить правила построения выражений. Ознакомиться с использованием функций в приложении.

4.1. Выражения

В операторе присваивания справа от знака "=" может быть расположено не только конкретное значение, но и *выражение*. При выполнении оператора присваивания во время работы программы это выражение вычисляется. Это означает, что по определенным правилам рассчитывается значение этого выражения, а затем это значение присваивается переменной. В состав выражений могут входить конкретные *числа, переменные, строки, функции.*

Чаще всего в операторе присваивания справа от знака "=" находится так называемое арифметическое выражение. *Арифметическое выражение* – это последовательность чисел, констант, переменных, функций и арифметических выражений, заключенных в круглые скобки, которые соединены между собой знаками арифметических операций. Значения арифметических выражений вычисляются по правилам, которые являются общеизвестными. Ниже приведена лишь таблица арифметических операций, используемых в языке Visual Basic.

Операция	Описание операции
A^B	Возведение A в степень B
-A	Перемена знака A
A*B	Умножение A на B
A/B	Деление A на B
A\B	Целочисленное деление A на B
A mod B	Деление по модулю A на B

A+B	Сложение A с B
A-B	Вычитание B из A

Переменные, входящие в выражение, должны иметь численные значения. *Функции* также должны иметь численные значения. Говорят, что функции *возвращают* определенные численные значения.

Константы – это величины, значения которых не могут меняться. Как и переменные, константы объявляются в начале текста программного кода. Синтаксис объявления константы:

```
Const Имя Константы [As Тип] = Значение Константы
```

Кроме объявляемых констант в программе могут использоваться *системные,* встроенные константы, например значение цвета: **vbRed** – значение красного цвета.

Пример фрагмента программы, состоящего из операторов объявления переменных, константы и нескольких операторов присваивания:

```
Dim R As Single, S As Single 'объявление переменных
Const Pi=3.1415  'объявление константы
R=10 'присвоение значения переменной R
S=Pi*R^2 'вычисление площади круга
RR=2*R 'присвоение значения переменной
S=Pi*R^2 'вычисление площади круга
```

В приведенном фрагменте присутствуют *комментарии* – произвольные строки, находящиеся правее символа "апостроф". Комментарии не влияют на ход выполнения программы, а используются для пояснения текста программы.

4.2. Функции в языке VBasic

Понятие функции в языке близко понятию функции в математике. Функция – это правило, которое ставит в соответствие одному набору значений аргументов из области их допустимых значений ровно одно значение самой функции.

Синтаксис функции такой:

```
Имя Функции (Список Аргументов Функции)
```

Имя Функции – это имя уже имеющейся в языке функции либо функции, написанной программистом.

Аргумент Функции – это либо число, либо переменная, либо выражение. Аргументы в *Списке Аргументов Функции* отделяются друг от друга запятыми.

4.3. Встроенные функции

Это функции языка, которые прилагаются со средой программирования. Встроенные функции группируются по виду.

Математические функции. В Visual Basic есть набор встроенных математических функций. Вот некоторые из них, наиболее распространенные:

Abs(x) – абсолютная величина числа x

Cint(x) – целое число, ближайшее к числу x

Cos(x) – косинус числа x

Fix(x) – целое число, равное числу x без дробной части

Int(x) – наибольшее целое число, не превышающее x

Sin(x) – синус числа x

Sqr(x) – квадратный корень из числа x

Пример использования функции **Int.**

В результате деления получается число 5=225.333333333..., требуется округлить его до второго знака после запятой. Для этого используем такой прием:

SS = **Int**(S*100)/100,

где S – неокругленный результат;

SS – результат с указанным округлением.

Финансовые функции. Таких функций в Visual Basic более десяти. Одна из них – функция, которая решает задачу о банковском кредите. Она имеет следующий синтаксис:

```
Pmt(ПроцСтавка, ЧислоПлатежей, СуммаКредита)
```

Эта функция возвращает размер разового платежа (со знаком минус), если известны *Процентная Ставка, Число Платежей и Сумма Кредита.*

Системные функции. К системным функциям относятся функции, действие которых напрямую зависит от работы системы Windows. К таким функциям относятся две:

- функция **InputBox** – для ввода данных пользователем через системное окно;
- функция **MsgBox** – для выдачи сообщений пользователю через системное окно.

Функция **InputBox** имеет следующий синтаксис:

```
InputBox (Приглашние[,Заголовок][,НачЗначение])
```

Приглашение – это любой текст, который должен, по замыслу программиста, находиться в *Окне ввода.* Его назначение – подсказать пользователю, какую информацию он должен ввести в специальное *поле ввода,* находящееся в этом окне.

Необязательный аргумент *Заголовок* – это надпись в строке заголовка *Окна ввода.*

НачЗначение – это значение, которое будет введено автоматически, если пользователь будет с этим согласен.

Возвращаемым значением данной функции является информация, вводимая пользователем. Visual Basic автоматически приписывает этой информации тип **String.**

Функция **MsgBox** имеет следующий синтаксис:

```
MsgBox( Текст [, Опция] [, Заголовок])
```

Это основная форма синтаксиса. Функция возвращает значение, которое затем как-то используется (например, присваивается переменной).

Есть вторая форма синтаксиса, когда функция не возвращает никакого значения, а действует просто как оператор – выдает информацию в *Окне сообщения.* В этом случае в записи функции отсутствуют скобки:

```
MsgBox Текст[,Опция][,Заголовок]
```

Текст – это строка сообщения, ради получения которой данная функция и применяется. Текст может содержать до 1024 символов. Кроме сообщения пользователю *Окно сообщения* может содержать и дополнительную информацию. Она задается значением аргумента *Опция.* Этим аргументом является целое число, которое может быть представлено как сумма двух слагаемых: Op = Op1 + Op2.

Значение *Op1* определяет вид сообщения и пиктограмму, которая помещается в *Окно сообщения.*

Значение Op1	Вид сообщения
16	Критическое сообщение
32	Вопрос
48	Предупреждение

64	Информация

Значение *Op2* определяет набор кнопок в *Окне сообщения.*

Значение Op2	Набор кнопок
0	ОК
1	ОК, Отмена
2	Стоп, Повтор, Пропустить
3	Да, Нет, Отмена
4	Да, Нет
5	Повтор, Отмена

Легко можно убедиться, что для любой комбинации Op1 и Op2 их сумма будет уникальной. Другими словами, с помощью одного числа можно установить и определенную пиктограмму, и определенную комбинацию кнопок.

Действие функции **MsgBox** таково: когда доходит очередь до ее выполнения, на экране появляется *Окно сообщения.* Если используется бесскобочная форма синтаксиса, нажатие одной из кнопок на этом окне просто завершает работу функции. А если используется форма со скобками, то значение функции присваивается какой-нибудь переменной.

Возвращаемое значение – это целое число от 1 до 7. Оно зависит от того, какая из кнопок *Окна сообщения* нажата.

Возвращаемое значение	Кнопка
1	ОК
2	Отмена
3	Стоп
4	Повтор

5	Пропустить
6	Да
7	Нет

4.4. Пример Windows-приложения

Рассмотрим пример построения Windows-приложения (рис. 32), в котором используются функции **InputBox** и **MsgBox.**

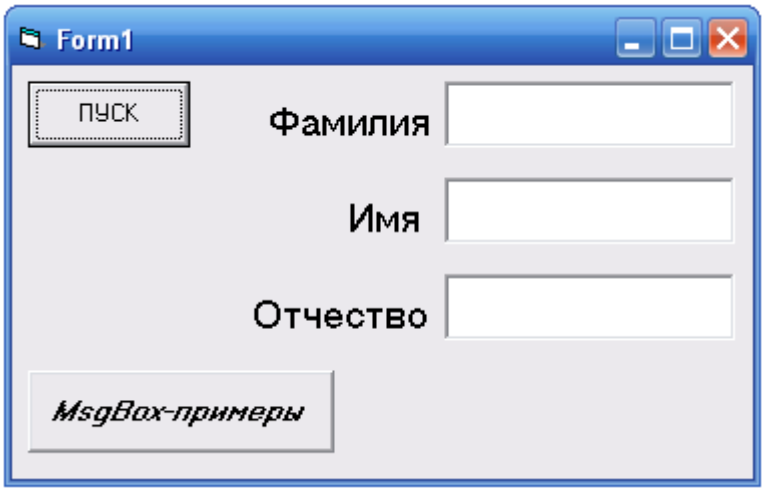

Рис. 32

Программа состоит из двух частей. В первой части кнопка "ПУСК" через стандартные окошки ввода функции вводит фамилию, имя, отчество и отображает их в текстовых полях. Во второй части кнопка "MsgBox-примеры" последовательно выводит ряд сообщений разного вида.

Процедура, срабатывающая при нажатии кнопки "MsgBox-примеры", имеет вид:

```
Private Sub Command1_Click() 'программирование кнопки "ПУСК"
Фамилия = InputBox("Введите Вашу фамилию:", _ "Ввод фамилии")
Имя = InputBox ("Введите Ваше имя:", _ "Ввод имени")
Отчество = InputBox ("Введите Ваше отчество:", _ "Ввод отчества")
Text1 = Фамилия: Text2 = Имя: Text3 = Отчество
```

```
End Sub
```

Процедура, срабатывающая при нажатии кнопки "MsgBox-примеры", имеет вид:

```
Private Sub Command2_Click() 'программирование кнопки "MsgBox-примеры"
    MsgBox(!!! !!! !!!")
    MsgBox ("  ??? ??? ???")
    MsgBox "MsgBox", 0, "Проверка вывода сообщений"
    MsgBox "Ошибка! Работа программы прерывается!", 16, "Критическое сообщение"
    MsgBox "Вычисления продолжать?", 32 + vbYesNo, "Вопрос"
    MsgBox "Ошибка в программе! Необходима коррекция!", 48 + 3, "Предупреждение"
    MsgBox "Рабочий день оканчивается в 19.30"
    MsgBox "Необходимо отключить оборудование, закрыть форточки, сдать помещение под охрану", 64, "Информация"
    End Sub
```

4.5. Определяемые функции

Рассмотренные раньше функции являются встроенными функциями языка, но каждый программист может определить и свои собственные (определяемые) функции.

Упражнение 4

- Сделать приложение по рис. 32.
- Запустить приложение, проверить работу.
- Дополнить приложение процедурой, срабатывающей, если щелчок левой кнопки мыши будет приходиться не по кнопкам формы. При этом на экране должно появляться сообщение-предупреждение "Ох! Да Вы промазали по кнопке!" (рис.33).

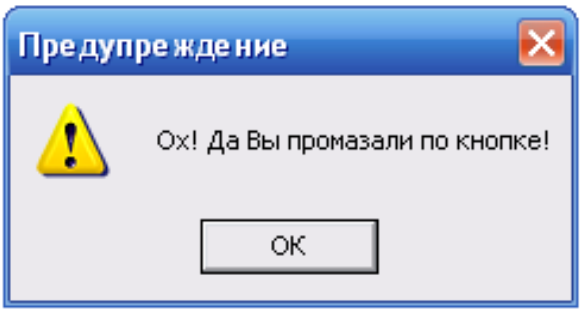

Рис. 33

- Откомпилировать приложение под именем "Предупреждение".
- Сохранить файлы проекта в папке "Ввод_данных".

Задания для самостоятельной работы

1. Ввести с клавиатуры произвольное строковое выражение и поместить его в выбранную ячейку текущего рабочего листа.

2. Отобразить содержимое любой ячейки рабочего листа в диалоговом окне MsgBox.

3. Разместить несколько вводимых с клавиатуры чисел в разных ячейках на одной строке (в одном столбце) текущего рабочего листа.

4. Ввести с клавиатуры элементы квадратной матрицы заданного размера. Расположить их в соответствующих ячейках текущего рабочего листа.

5. Скопировать содержимое некоторых ячеек с одного рабочего листа на другой.

6. Вывести значение некоторого числового выражения в различные ячейки текущего рабочего листа с использованием произвольных атрибутов форматирования (цвет текста и фона, числовые форматы данных, гарнитура шрифта и т. д.).

Лабораторная работа № 5
ФУНКЦИИ РАБОТЫ СО СТРОКАМИ. ФИНАНСОВЫЕ ФУНКЦИИ

Цель работы. Ознакомиться с функциями обработки строк, основами программирования финансовых функций.

5.1. Функции обработки строк

Строка – это либо упорядоченная последовательность символов, либо пустая строка. Для обозначения строки используются кавычки:

“” – пример пустой строки;

“Программирование**”** – пример непустой строки.

Число символов строки называется длиной строки. Длина пустой строки равна нулю. Каждый символ строки имеет свою позицию – порядковый номер при счете слева направо. В VBasic используется понятие подстроки – это вырезанный кусок из строки.

Две строки можно соединить в одну, такое действие называется конкатенацией или сложением строк:

ОбъединениеСтрок = Строка1 + Строка2 + Строка3.

Можно применить знак конкатенации & (амперсант). С его помощью можно соединить не только строки, но и числа. При этом числа будут сначала преобразованы в строки, и результат тоже будет строкой. Например:

```
Dim Строка1 As String, Строка2 As String,
Результат As String
Строка1 = "Объем комнаты"
Строка2 = "куб. метров"
Результат = Строка1 &2,5*3*5& Строка2
```

После выполнения этого кода результатом будет строка:

"Объем комнаты 37,5 куб. метров"

Существует несколько **функций обработки строк**, которые позволяют модифицировать, обрабатывать строки, выбирать информацию.

Функция определения длины строки.

```
Len(Строка)
```

В результате возвращается длина строки.

Функция выделения подстроки:

```
Mid(Строка, Позиция [,Длина])
```

В *Строке* выделяется и возвращается подстрока, начиная с заданной *Позиции. Длину* выделяемой подстроки можно не указывать - тогда будет возвращена подстрока от данной *Позиции* до конца *Строки*.

Функция выделения подстроки.

```
Left(Строка, Длина)
```

В *Строке* выделяется левая подстрока указанной *Длины*. Она и будет возвращаемым значением.

Функция выделения строки.

```
Right(Строка, Длина)
```

В *Строке* выделяется правая подстрока указанной *Длины.*

Функция поиска подстроки.

```
lnStr([Старт,] Строка, Подстрока)
```

В *Строке* ищется то место, где находится *Подстрока.* В результате возвращается позиция первого символа *Подстроки.* Если подстрока не найдена, возвращается 0.

Функции преобразования *имеют следующие назначения.*

```
Val(Строка)
```

Эта функция преобразует *Строку* в число.

```
Str(Число)
```

Эта функция преобразует *Число* любого типа в строку.

```
Asc(Строка)
```

Эта функция преобразует *Строку* в код ASCII первого символа этой строки.

```
Chr(Код)
```

Эта функция преобразует *Код ASCII* в строку из одного символа.

```
Ucase(Строка)
```

Эта функция возвращает исходную *Строку,* преобразуя все буквы в прописные.

```
Lcase(Строка)
```

Эта функция возвращает исходную *Строку,* преобразуя все буквы в строчные.

5.2. Использование Финансовых функций

В языке Visual Basic есть больше десяти встроенных финансово-математических функций. Рассмотрим, например, использование трех функций: **Pmt, PV, Rate.**

Функция **Pmt(rate,nper,pv)** возвращает в банк величину периодического взноса, который должен выплачивать клиент, чтобы в определенный срок вернуть банку взятые у него деньги (кредит). В функции:

rate – банковская процентная ставка. Если клиент выплачивает взнос ежемесячно, а банк берет 10 % годовых, то значение rate равно 0,01;

nper – общее число периодов выплат. Если кредит взят на два года, то значение rate равно 24;

pv – величина кредита, т. е. сумма, которую клиент взял в банке.

Функция **PV(rate,nper,pmt)** возвращает величину приведенной стоимости (кредита) при заданной величине периодической выплаты pmt.

Функция **Rate(nper,pmt,pv)** возвращает величину банковской процентной ставки, относящейся к периоду выплаты.

5.3. Пример Windows-приложения

На рис. 34 приведен пример программы расчета выплаты с использованием финансовой функции **-Pmt(r, n, v).**

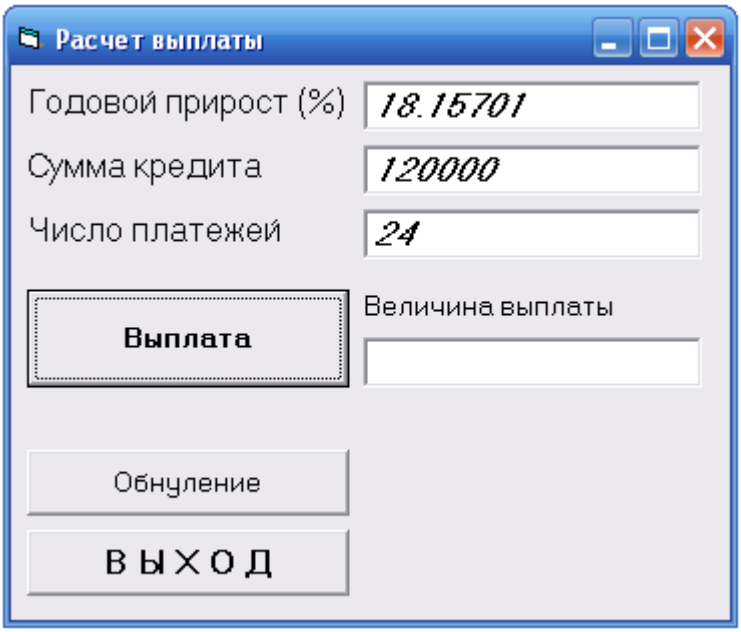

Рис. 34

Программный код приложения:

```
Private Sub Command1_Click() 'Выплата
r = Val(Text1 .Text)
n = Val(Text2.Text)
v = Val(Text3.Text)
r = r/1200
p = -Pmt(r, n, v)
Text4.Text = Str(p)
End Sub

Private Sub Command2_Click() 'Очистка
Textl .Text =""
Text2.Text =""
Text3.Text =""
Text4.Text =""
End Sub

Private Sub Command3_Click() 'Кнопка "ВЫХОД"
End
End Sub
```

Упражнение 5

- Сделать приложение по рис. 34.
- Запустить приложение, проверить работу.
- Используя функции обработки строк, форму и код приложения (см. рис. 32), дополнить его кнопкой и процедурой, позволяющей подсчитывать число символов в фамилии, имени, отчестве и выводить в первое отдельное поле первые буквы (инициалы), а во второе отдельное поле – фамилию, имя, отчество в виде одной строки, используя конкатенацию. Примерный вид дополнения формы изображен на рис. 35.

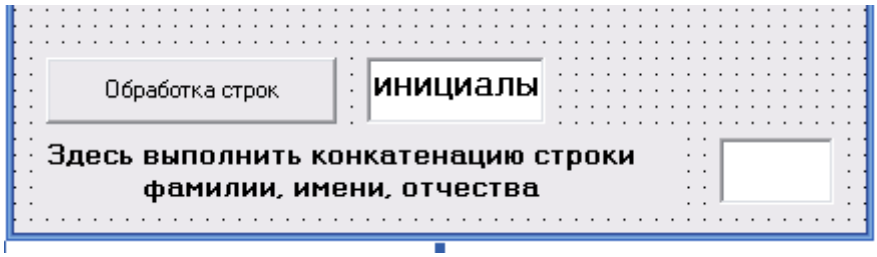

Рис. 35

• По аналогии с приложением на рис. 34 сделать программу с использованием финансовых функций PV(rate,nper,pmt) или Rate(nper,pmt,pv).

Задания для самостоятельной работы

1. Подсчитать, сколько раз каждый символ русского алфавита встречается в заданной строке. Прописные и строчные символы считать одинаковыми.

2. Для заданной строки символов строчные буквы латинского алфавита преобразовать в прописные, а прописные – в строчные.

3. Вывести на экран заданную строку символов в обратном порядке.

4. Для заданного предложения вывести каждое слово в отдельную ячейку. Подсчитать количество слов в предложении.

5. Удалить из произвольного текстового выражения все пробелы. Подсчитать количество пробелов.

6. Для заданных фамилии, имени и отчества студента вывести на экран только фамилию и инициалы.

7. Написать заданное слово вразрядку (буквы отделены друг от друга пробелом). Определить количество букв в слове.

8. Ввести с клавиатуры 2 строки: фамилия, имя, отчество и номер группы. Получить строку вида: ФИО – студент группы 101xxx. Определить длину полученной строки.

9. Ввести с клавиатуры 2 строки. Определить, входит ли вторая строка в состав первой. Если да, то с какой позиции.

10. Преобразовать заданное число в строку. Сформировать строку вида: xxx рублей. Определить длину полученной строки.

11. Определить, какая из двух заданных строк длиннее. Результаты вывести в виде: первая строка (текст) длиннее второй (текст). Определить длину результирующей строки.

12. Подсчитать количество гласных и согласных букв в заданном слове.

Лабораторная работа № 6
ПРОГРАММИРОВАНИЕ ВЕТВЛЕНИЙ

Цель работы. Ознакомиться с условным оператором IF, оператором перехода **Select Case.**

6.1. Условные выражения

При решении большинства задач часто приходится выбирать, по какому из нескольких путей нужно идти к решению. Для реализации условия выбора в языке существует вид выражений – условные выражения.

Простое условие – это два выражения, между которыми помещается знак сравнения. Выражениями могут выступать числа, числовые переменные, функции, арифметические выражения, строки. Операции сравнения и их знаки приведены в таблице:

Операция	Описание операции
>	Больше чем
>=	Больше или равно
<	Меньше чем
<=	Меньше или равно
=	Равно
< >	Не равно

Простое условие, в зависимости от того, выполняется оно или нет, имеет значение **True** или **False** – *Истина* или *Ложь*. Примеры простых условий и их значений приведены в таблице.

2.9990<2.9991	имеет значение True
3.14<=3.14	имеет значение True
-Y^2>Abs(Y)	имеет значение False
"abc"="abc"	имеет значение True
"-abc"="abc"	имеет значение False

Сложное условие – это последовательность простых условий или других выражений, заключенных в круглые скобки, которые соединены между собой знаками логических операций: **AND** – *логического умножения,* OR – *логического сложения,* **NOT** – *логического отрицания.* Каждое условное выражение вычисляется, а результатом является одно из двух значений: **True** или **False** – *Истина* или *Ложь.*

Правила вычисления значений логических выражений нужно знать так же, как таблицу умножения.

A	B	A AND B
True	True	True
True	False	False
False	True	False
False	False	False

A	B	A OR B
True	True	True
True	False	True
False	True	True
False	False	False

A	NOT A
True False	False True

Примеры сложных условий и их значений.

(X"2>0)**OR**(X^2=0) имеет значение True
(Y^2>=0)**AND**(Abs(Z)>=0)3.14<=3.14 имеет значение True
NOT(Len("abcd")>0) имеет значение False

6.2. Условный оператор IF

Простые и сложные условия являются элементами условного оператора, позволяющего в программном коде выполнять ветвления. Условный оператор имеет две формы: однострочную и многострочную.

Синтаксис однострочной формы:

```
If Условное Выражение Then Оператор 1 [Else Оператор2]
```

Синтаксис многострочной формы:

```
If Условное Выражение Then
Группа
операторов
[Else Группа операторов]
End If
```

Оператор условного перехода используется в приложении, показанном на рис. 36. В этой программе в текстовое окошко Text вносится число от 0 до 24, а в графическом окошке высвечивается фотография утреннего неба, если вводится число из интервала [5 ... 8], дневного неба, если вводится число из интервала [8 ... 18], вечернего или ночного неба, если вводится число из интервалов соответственно [18 ... 21] и [21 ... 5].

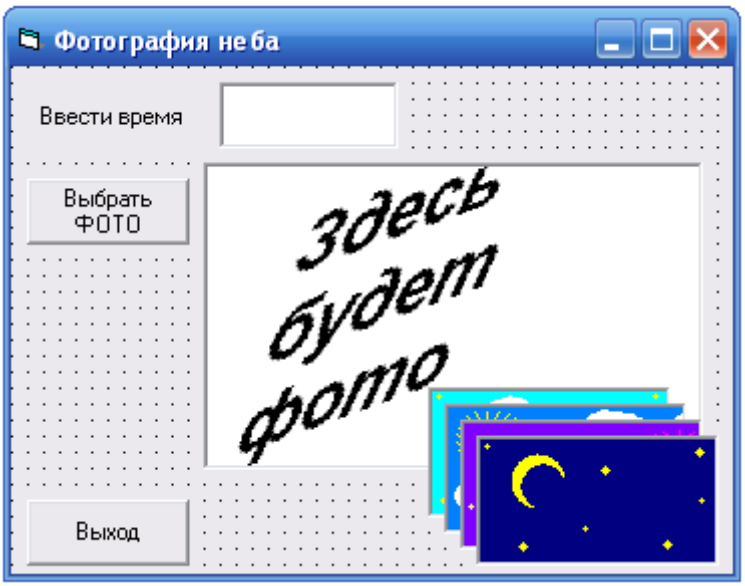

Рис. 36

В проекте использованы объекты: Label, Text, CommandBatton, PictureBox, Image.

Код проекта:

```
Private Sub Command1_Click() 'Выбрать ФОТО
If ((Val(Text1 .Text) > 5) And (Val(Text1 .Text)<=8))
Then Imagel .Picture =Picture1 (0).Picture
If ((Val(Text1 Text) > 8) And (Val(Text1 .Text) <= 18))
Then Image 1.Picture = Picture1(1).Picture
If ((Val(Text1 Text) > 18) And (Val(Text1 .Text) <= 21))
Then Imagel .Picture=Picture1 (2).Picture
If (((Val(Text1 Text) > 21) And (Val(Text1 Text) <= 24))
Or ((Val(Text1 Text) > 0) And (Val(Text1 Text) <= 5))) _
                Then Imagel .Picture = Picturel (3).Picture
End Sub
Private Sub Command2_Click() 'Выход
End
End Sub
```

6.3. Оператор перехода CASE

Для альтернативы, или выбора варианта, существует оператор перехода **Select Case.** Его синтаксис:

```
Select Case Переменная Case Значение1
   Если Переменная=Значение1, то выполняется эта группа операторов Case Case Значение2
   Если Переменная=Значение2, то выполняется эта группа операторов Case Case Значение3
   Если Переменная=Значение3, то выполняется эта группа операторов
[Case Else
   Если Переменная не равна ни одному из употребляемых значений]
```

```
End Select
```

Пример использования оператора перехода **Select Case** и двух новых инструментов в наборе **Tollbox - OptionButton** *(Кнопка-переключатель)* и **Frame** *(Рамка)* (рис. 37) показан в приложении, форма которого представлена на рис. 38.

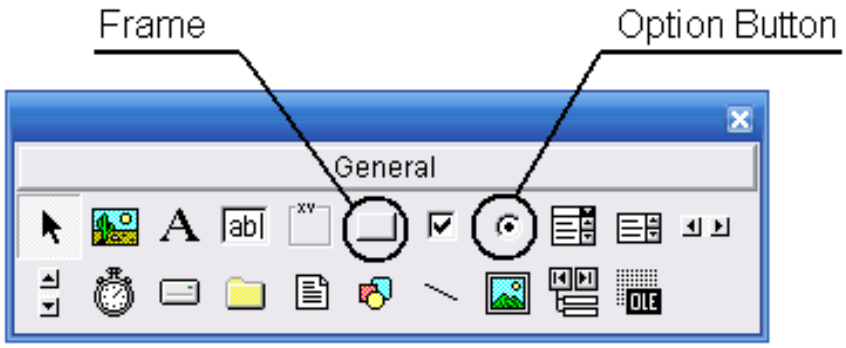

Рис. 37

Рис. 38

В приложении в зависимости от выбора кнопки переключателя выводится соответствующий рисунок.

Код программы:

```
Dim AA As Byte    'описание глобальной переменной
Private Sub F_Case()    'функция выбора варианта рисунка
```

```
по значению AA Select Case AA
Case 1
Image1 .Picture = Picture1 (0).Picture
Case 2
Image1.Picture = Picture1 (1). Picture
Case3
Image1.Picture = Picture1 (2).Picture
Case 4
Image 1. Picture = Picture 1(3). Picture
End Select
End Sub

Private Sub Option_1 Click()
AA= 1
F_Case
End Sub
Private Sub Option2_Click()
AA = 2
F_Case
End Sub

Private Sub Option3_Click()
AA = 3
F_Case
End Sub

Private Sub Option4_Click()
AA = 4
F_Case
End Sub

Private Sub Command5_Click() 'кнопка "Выход" End
End Sub
```

6.4. Оператор перехода GoTo

В Visual Basic есть еще один, хотя и не очень популярный, оператор перехода **GoTo,** который называется *оператором безусловного перехода.* Его синтаксис:

```
Метка 11:

IF a>b then GoTo Метка22
GoTo Метка11
…
Метка22:
```

Упражнение 6

- Сделать приложение по рис. 36.
- Запустить приложение, проверить работу.
- Используя в качестве основы форму и код приложения, приведенного на рис. 36, сделать приложение по рис. 38.

Задания для самостоятельной работы

Каждое задание состоит из двух частей: а) и б).

а) Применение операторов `Goto` ***и*** `If`

1. Найти корни квадратного уравнения $ax^2 + bx + c = 0$.

2. Вычислить по формуле

$$y = \begin{cases} a^2 + b^2 + 4\sqrt{a+b} - \dfrac{ab+2}{|a-2b|}, & \text{если } a > 2b \\ a^b + \sin|a| - \cos\sqrt{b}, & \text{если } a = 2b \\ \lg\dfrac{a}{b} + \operatorname{tg}\dfrac{b}{a}, & \text{если } a < 2b \end{cases}$$

3. Найти наибольшее и наименьшее из трех чисел A, B, C.

4. Ввести с клавиатуры число, месяц, год, день недели. Вывести на экран дату и день недели для следующего дня.

5. Рассчитать величину подоходного налога следующим образом:

Если доход ≤ 240 минимальных заработных плат (мзп), то налог равен 9 % от величины дохода. От 240 мзп + 1 руб. до 600 мзп – 21,6 мзп + 15 % от суммы, превышающей 240 мзп. От 600 мзп + 1 руб. до 840 мзп – 75,6 мзп + 20 % от суммы, превышающей 600 мзп. От 840 мзп + 1 руб. до 1080 мзп – 123,6 мзп + 25 % от суммы, превышающей 840 мзп. Свыше 1080 мзп –183,6 мзп + 30 % от суммы, превышающей 1080 мзп.

6. Задать три стороны прямоугольного параллелепипеда. Определить, является ли данная фигура кубом.

7. Рассчитать модуль введенного с клавиатуры числа, не используя функцию Abs.

8. С клавиатуры ввести два числа. Разделить большее на меньшее.

9. По названию месяца определить количество дней в нем.

10. По введенному с клавиатуры году определить, является ли он високосным.

11. Определить принадлежность человека определенному знаку зодиака по дате его рождения.

12. Компьютер выдает на экран 5 вопросов и по 2 варианта ответов к каждому. Правильный ответ оценивается в 1 балл. Оценить уровень знаний тестируемого. Вопросами и ответами задаться самостоятельно.

б) Применение оператора `Select Case`

1. Для заданного числа из диапазона от 1 до 10 выдать его словесное (символьное) представление.

2. По номеру месяца выдать его название и количество дней в нем.

3. Ввести два числа и знак арифметической операции между ними ("+", "*", "/"). Вычислить значение арифметического выражения согласно введенному варианту.

4. Ввести с клавиатуры первые две цифры штрих-кода товара. По введенному значению определить страну-производителя.

5. Организовать телефонный справочник известных аварийных и справочных служб.

6. По номеру группы определить количество студентов в ней и год поступления.

7. По номеру группы определить название специальности и курс.

8. По порядковому номеру выдать на экран фамилию студента вашей группы.

9. Вывести на экран список цветов (5-7 элементов). Выдать на экран свою фамилию выбранным цветом. Выбор цвета реализовать по его порядковому номеру или названию.

10. По номеру дня недели или его названию выдать на экран расписание занятий.

11. Для введенного с клавиатуры символа определить, является ли этот символ буквой русского или латинского алфавита, прописной или строчной буквой.

12. Для введенной с клавиатуры цифры вывести на экран ее графическое представление (аналогично индексу на почтовых конвертах) с помощью заливки фона текущей ячейки.

Лабораторная работа № 7
ПРОГРАММИРОВАНИЕ ПОВТОРЕНИЙ

Цель работы. Ознакомиться с операторами повторения.

Повторение – это выполнение одного или нескольких операторов программы более одного раза. За счет повторений сокращается размер программного кода. Реализуются повторения многострочными операторами цикла двух видов: со счетчиком и с условием.

7.1. Цикл со счетчиком

Синтаксис оператора повторения для цикла со счетчиком:

```
For  Имя=Значение1 To Значение2 [Step Значение3]
   Повторяющиеся операторы (операторы цикла)
Next [Имя]
```

где Имя – имя переменной, которую называют счетчиком (индексом цикла);

Значение1 – начальное значение счетчика;

Значение2 – конечное значение счетчика;

Значение3 – величина, на которую изменяются значения счетчика при одном повторении.

Если нет противоречий в значениях, то операторы цикла выполняются при начальном значении счетчика. Далее значение счетчика меняется на величину шага и выполняется проверка со Значением2. Если значение счетчика меньше, то начинается повторное выполнение операторов цикла с новым значением счетчика, если больше, то цикл завершается и начинают отрабатываться следующие операторы программы.

Пример блока простейшей программы, использующей цикл со счетчиком:

```
For  I=0To 25 Step 5
Print I
Next I
```

Результатом работы этой программы является печать столбиком чисел: О, 5, 10, 15, 20, 25 при нажатии на кнопку "Печать чисел" (рис. 39). Программа дополнена кнопкой "Очистка поля", что позволяет очистить поле вывода, и кнопкой "Печать приветов", что позволяет напечатать указанное число приветов. В приложении

использованы объекты: Form, CommandButton, Label, TextBox, PictureBox. При печати приветов меняются свойства шрифта, поэтому каждое последующее слово "Привет" напечатано большими по размеру буквами.

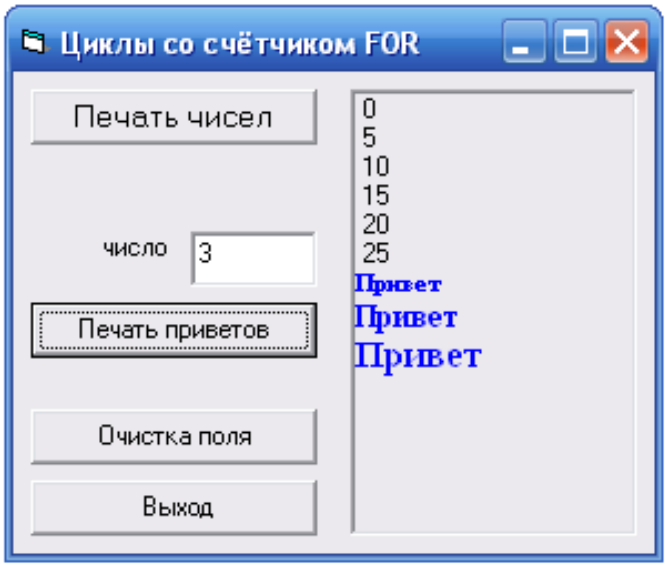

Рис. 39

Код приложения:

```
Private Sub Command1_Click() 'вывод шести чисел
For I = 0 To 25 Step 5
Picture 1. Print I
Next I
End Sub

Private Sub Command2_Click() 'вывод приветов
'изменение шрифта вывода
Picture1. Font.Bold = True
Picture1. Font.Name = "Times New Roman"
Picture1. Font.Size = 8
Picture1. ForeColor = vbBlue
'печать приветов
For I = 1 To Val(Text1 Text)
Picture1. Print "Привет"
Picture1. Font.Size = Picture1 .Font.Size + 1*2
```

```
Next I
'восстановление шрифта и размера
Picture1 .Font.Bold = False
Picture1 .Font.Name = "Ms Sans Serif"
Picture1 .Font.Size = 8
Picture1 .ForeColor = vbBlack
End Sub

Private Sub Command3_Click() 'очистка поля
Picture1.Cls
End Sub

Private Sub Command4 Chck() кнопка Выход
End
End Sub
```

7.2. Цикл с условием

Если при программировании повторяющихся действий неизвестно число повторений, то используется оператор цикла в форме **Do Until... Loop** или **Do While ... Loop.**

Синтаксис этого оператора имеет две формы:

```
Do Until Условие
Повторяющиеся операторы (операторы цикла)
Loop
```

или

```
Do Условие

Повторяющиеся операторы (операторы цикла)

Loop Условие
```

В качестве условия могут использоваться ключевые слова **While** или **Until.** В

первом случае *Повторяющиеся операторы* выполняются, если значение условного выражения равно **True,** во втором случае *Повторяющиеся операторы* выполняются, если значение условного выражения равно **False.**

Если Условие стоит в начале цикла, то сначала оно проверяется, а далее выполняются операторы цикла. Если условие стоит в конце цикла, то проверка и выход из цикла выполняются после выполнения *Повторяющихся операторов.*

Оператор, который прерывает выполнение цикла, называется *оператором прерывания цикла.* Его синтаксис:

```
Exit DO
```

Этот оператор может быть помещен в любое место в пределах цикла, после его выполнения произойдет переход к операторам, стоящим за циклом.

Операторы цикла допускают использование внутри циклов других циклов, так называемых вложенных циклов. Их конструкция имеет вид:

```
For  Имя1 =Значение1 ToЗначение2 [Step Значение3]
Повторяющиеся операторы (операторы цикла)
For  Имя2 = Значение1 ToЗначение2 [Step Значение3]
Повторяющиеся операторы (операторы цикла)
Next [Имя1]
Next [Имя2]
```

Жестким правилом при использовании вложенных циклов является их непересекаемость.

Цикл с условием реализован при расчете среднего значения вводимых чисел в приложении на рис. 40. Эта программа рассчитывает среднее значение любого количества введенных чисел. Ввод чисел выполняется при помощи встроенной функции **InputBox.** При этом ведется счет выполненных функций, а отметкой завершения ввода чисел является ввод пустой строки, т.е. нажатие кнопки "Enter" при пустом окне InputBox.

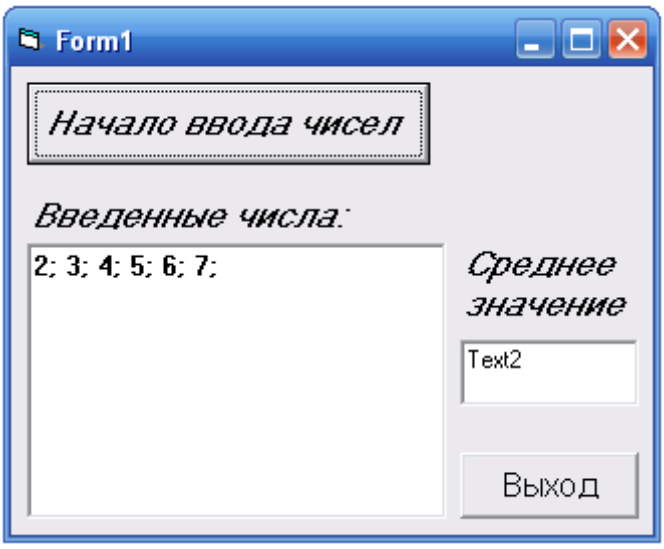

Рис. 40

Текст программы:

```
Private Sub Command1_Click() 'начало ввода чисел
Dim ВводимаяСтрока As String, НомерЧисла As Integer
Dim ВводимоеЧисло, СуммаЧисел As Single
Dim Заголовок, Сообщение As String

Text1 .Text="" 'очистка экрана

Заголовок = "Ввод очередного числа"
Сообщение = "Введите очередное число" &_
"и щелкните кнопку ОК" & _
"Для завершения ввода щелкните кнопку Cancel"
НомерЧисла = 0
СуммаЧисел = 0
ВводимаяСтрока = InputBox(Сообщение, Заголовок)

Do Until ВводимаяСтрока = ""
ВводимоеЧисло = Val(ВводимаяСтрока)
НомерЧисла = НомерЧисла + 1
СуммаЧисел = СуммаЧисел + ВводимоеЧисло
```

```
Text1 = Text1 &Str(ВводимоеЧисло) &"; "
Text2 = Str(СуммаЧисел / НомерЧисла)
ВводимаяСтрока = InputBox (Сообщение, Заголовок)
Loop
End Sub
Private Sub Command2_Click()
End
End Sub
```

Упражнение 7

- Сделать приложение по рис. 39.
- Запустить приложение, проверить работу.
- Сделать приложение по рис. 40. Обратить внимание на значения свойств объектов:

```
Form Forml
Caption = "Циклический ввод чисел"
StartUpPosition = 2 'CenterScreen
TextBox.Text1 = Введенные числа
Locked = -1 'True
MultiLine = -1 'True
TextBox.Text2 = Среднее значение
Alignment = 2 'Center
Locked = -1 'True
```

- Запустить приложение, проверить работу.
- Для наглядности дополнить форму приложения выводом количества введенных чисел и их суммы.

Задания для самостоятельной работы

Каждое задание состоит из двух частей: а) и б).

а) Применение оператора `For`

1. Вычислить число размещений из n по m ($n \geq m$)

$$A_n^m = \frac{n!}{(n-m)!}$$

2. Вычислить число сочетаний из n по m ($m \geq n$)

$$C_n^m = \frac{n!}{m!\,(n-m)!}$$

3. Вывести на экран таблицу умножения для введенного с клавиатуры числа n в виде:

$n \times 1 = n$

$n \times 2 = 2n$

...

$n \times 10 = 10n$

4. Вычислить число e с помощью ряда

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{n!} + \cdots$$

Число членов ряда – 10. Вычислить точное значение e с помощью функции *Exp*. Определить относительную погрешность вычислений.

5. Вычислить значение функции $\ln(1+x)$ с помощью ряда

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} + \cdots + (-1)^{n-1}\frac{x^n}{n} + \cdots$$

6. Вычислить значение функции (протабулировать функцию)

$$y(x) = \begin{cases} |\sqrt{x}|, & \text{если } x \leq 0 \\ x^3 + x^2, & \text{если } x \in]0,2] \\ \dfrac{144}{6x}, & \text{если } x > 2 \end{cases}$$

на интервале [-5,5] с шагом измерения аргумента, равным 1.

7. Используя заливку ячее, нарисовать на экране закрашенный квадрат размерностью $n \times n$ вида:

***
***
***

8. Используя заливку ячеек, нарисовать на экране незакрашенный квадрат размерностью $n \times n$ вида:

```
***
* *
***
```

9. Используя заливку ячеек, нарисовать на экране закрашенный прямоугольный треугольник с катетами длиной *n* вида:

```
*
**
***
```

10. Для заданного целого числа вывести на экран список чисел, на которые оно делится без остатка.

11. Компьютер генерирует N чисел (-5 < N < 5) с помощью функции `Rnd`. Перед использованием `Rnd` инициализируйте генератор случайных чисел с помощью `Randomise`. Определить количество положительных, отрицательных, нулевых значений. Величину *N* задать с клавиатуры.

12. С клавиатуры вводится название месяца и день недели, приходящийся на первое число. Вывести календарь на указанный месяц в виде, представленном на рис. 41.

Июнь

Рис. 41

б) Применение операторов `While…Wend` ***и*** `Do…Loop`

1. Протабулировать функцию

$$f(x)=\begin{cases} x^2\sin 3x-|\ln x^4|, & \text{если } x<0 \\ \sqrt[3]{x^3-x^2}, & \text{если } x\geq 0 \end{cases}$$

на отрезке [-2; 2] с шагом изменения аргумента 0,5.

2. Какой процент годовых *P* (*P*=100*I*) должен быть, чтобы за *N* лет ежегодные вклады по S_1 привели к накопленной сумме *S*? Задача сводится к решению неравенства

$$S_1 \frac{(1+I)^N - 1}{I} \geq S.$$

Исходные данные: S=1500 руб., S_1=200 руб., N=5 лет, начальное значение P=1%, шаг изменения ΔP=0,5 %.

3. Рассчитать траекторию полета тела, брошенного под углом α к горизонту со скоростью V. Критерий окончания расчетов – падение тела на землю. Уравнение движения тела имеет вид

$$y = x \operatorname{tg} \alpha - \frac{gx^2}{2V^2 \cos^2 \alpha}.$$

Исходные данные α = 30°, V= 10 м/с, Δx = 1 м, g = 9,81 м/с^2.

4. Тело брошено вертикально вверх со скоростью V. Рассчитать изменение вертикальной координаты тела в течение 10 секунд полета с шагом изменения времени Δt = 5 с. Уравнение движения тела имеет вид

$$h = Vt - \frac{gt^2}{2}.$$

5. Рассчитать изменение величины атмосферного давления при изменении высоты местности. Динамику изменения давления отражает формула

$$P = 101{,}3\left(1 - \frac{6{,}5h}{288}\right)^{5{,}125}.$$

Шаг изменения высоты местности – 0,2 км. Диапазон изменения высоты – от 0 до 2 км.

6. Используя заливку ячеек нарисовать на экране фигуру вида

```
***
 **
  *
  **
  ***
```

Число n (длину стороны) ввести с клавиатуры.

7. Компьютер генерирует случайные числа от 0 до 10 с помощью функции Rnd до тех пор, пока не выдаст значение 9. Перед использованием Rnd инициализируйте генератор случайных чисел с помощью Randomise. Подсчитать количество сгенерированных чисел.

8. Компьютер генерирует 0 или 1 случайным образом с помощью функции Rnd (число, меньшее 0,5, полагаем равным 0, большее – 1). Перед использованием Rnd инициализируйте генератор случайных чисел с помощью Randomise. Вы пытаетесь угадать задуманные числа. Процесс повторяется до тех пор, пока вы не угадаете. Определить, сколько попыток было сделано.

9. С клавиатуры вводятся числа с помощью функции InputBox до тех пор, пока не будет введено значение 0. Определить количество введенных чисел, их сумму, наибольшее и наименьшее значения.

10. Вывести на экран обеденное меню. Выбор блюд – по их порядковому номеру с помощью функции InputBox. Окончание выбора – число 0. На экран выдается общая стоимость заказанных блюд.

11. Подсчитать значение определенного интеграла методом левых прямоугольников (рис. 42) с шагом интегрирования $\Delta x = 0{,}1$.

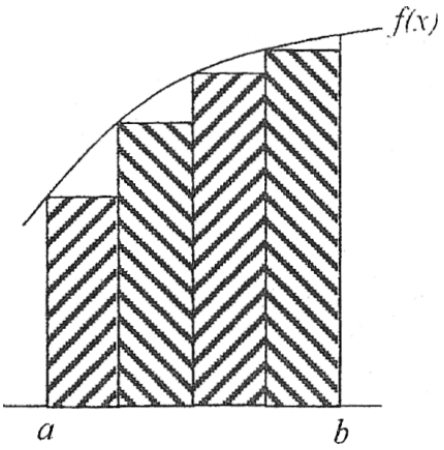

Рис. 42

$$\int_a^b f(x)dx - \int_0^1 \sin x\,dx$$

12. С клавиатуры с помощью функции InputBox вводятся координаты узловых точек ломаной линии. Признак окончания ввода – задание точки (0,0). Вычислить длину ломаной линии.

Лабораторная работа № 8
МАССИВЫ

Цель работы. Изучить понятие "массив". Научиться использовать массивы в приложениях VBasic.

8.1. Одномерный массив

Дальнейшим развитием понятия "переменная" является понятие "массив". Массив – это объединение переменных одного типа. У них одно имя, а отличаются они друг от друга своим номером – значением так называемого индекса. У переменной массива могут быть два, три или даже больше индексов – это **многомерные массивы.** Организация данных в виде массивов экономит место и упрощает алгоритмы.

Рассмотрим применение одномерного массива при расчете средней температуры нескольких дней месяца. Описывается массив двумя способами:

```
Dim Температура( 1 to 31) As Single
```

или

```
Dim Температура(31) As Single
```

В первом случае под массив резервируется тридцать ячеек, начиная с ячейки под номером 1, а во втором случае резервируется тридцать одна ячейка, начиная с нулевой. Допускается использование отрицательных значений индексов.

Пусть индексу массива соответствует день месяца, тогда, чтобы вызвать значение температуры в день **n**, необходимо обратиться по форме:

```
Температура(n)
```

8.2. Пример Windows-приложения

Пример кода, использующего массив:

```
Dim M(0 To 30) As Single    'описание массива
Dim Nmb As Integer, Nmbl  As Integer
Dim NmbStr As String
Dim Flag As Byte
Private Sub Form_Load()    'срабатывает при выводе формы
Nmbl = 1  'начальное значение счетчика
Text2.Text = Str(Nmbl)
End Sub
Private Sub Text1_Click()   'ввод значений массива
M(Nmbl) = Val(Text1 .Text)
Nmbl = Nmbl + 1   'счетчик номера записи
Text2.Text = Str(Nmbl)
Textl .Text =""
M(0) = Nmbl-1  'число введенных значений
End Sub


Private Sub Command 1_Click()  'вывод массива на просмотр
Label2.Caption = "Всего дней"
Label2.ForeColor = vbBlue
Text2.ForeColor = vbBlue
Text2.Text = Str(M(0))
For Nmb = 1 To Nmbl - 1
Text3.Text = Text3.Text & Str(M(Nmb)) &";"
Next
End Sub
Private Sub Command2_Click() 'вычисление среднего значения
Dim A As Single
For Nmb =1 To Nmbl- 1
A = A+M(Nmb)
Next
Text4.Text = Str(A / (Nmbl -1))
```

```
End Sub


Private Sub Command3_Click()'минимальное значение
Dim Min As Single, I As Integer
Min = 999999
For Nmb=1ToClnt(M(0))
If M(Nmb) < Min Then
Min = M(Nmb)
End If
Next
Text5.Text = Str(Min)
End Sub

Private Sub Command4 Click()    'максимальное значение
Dim Max As Single
Max = -999999
For Nmb = 1 To Clnt(M(0))
If (M(Nmb) > Max) Then Max = M(Nmb)
Next
Text6.Text = Str(Max)
End Sub

Private Sub Command5_Click() 'Очистка
Text1 .Text =
Text2.Text =
Text3.Text ='"
Text4.Text =
Text5.Text =
Text6.Text =
Nmbl = 0
ForNmb=1ToM(0)
M(Nmb) = 0
Next
M(0) = 0
Label2. Caption = "Индекс(день)"
```

```
Label2.ForeColor = vbBlack
Text2.ForeColor = vbBlack
End Sub

Private Sub Command6_Click() 'выход
End
End Sub
```

В этом примере (рис. 43) рассчитывается среднее значение температуры нескольких дней и выбираются наименьшее и наибольшее значения.

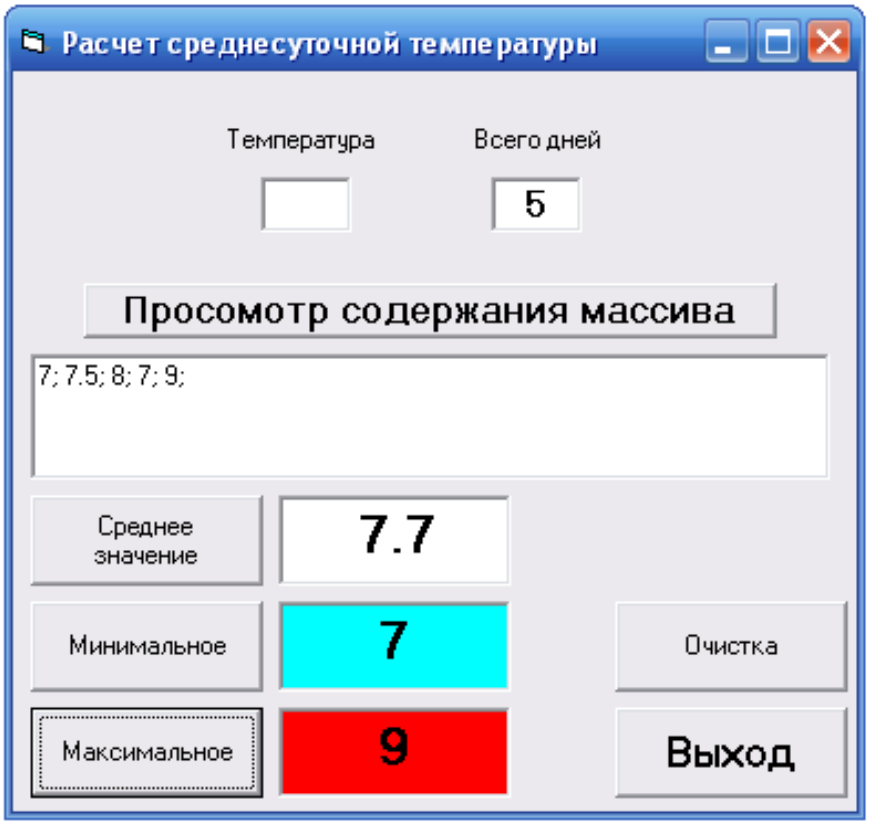

Рис. 43

8.3. Массив объектов

Массивы могут состоять не только из чисел, но и из строк, объектов. Следующая программа (рис. 44) демонстрирует работу с массивом объектов, картинок. Форма приложения является библиотекой объектов OLE-рисунков

PaintBrush.

Рис. 44

На форме устанавливаются объекты управления, в которых помещаются изображения. Объектами выбираются контейнеры OLE из стандартного набора Tollbox.

Пиктограмма этого инструмента показана на рис. 45. После размещения контейнера OLE на экранной форме появляется системное окно Windows. Для того чтобы картинка целиком умещалась в границах объекта OLE на экранной форме, следует установить значение свойства SizeMode этого объекта равным Stretch или Autosize.

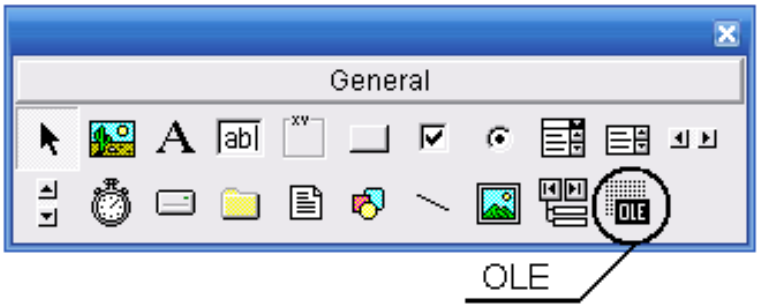

Рис. 45

Код приложения:

```
    Private Sub Command 1_Click()  'выбор из массива по 'значению
TabIndex=0
    OLE1(Command1.Tablndex).Visible = True
    End Sub

    Private Sub Command2_Click()   'выбор из массива по 'значению
TabIndex=1
    OLE1 (Command2.Tablndex).Visible = True
    End Sub

    Private Sub Command3_Click()   'выбор из массива по 'значению
TabIndex=1
    OLE1 (Command3.Tablndex).Visible = True
    End Sub
```

8.4. Многомерный массив

Массивы могут быть и многомерными. В таких массивах присутствует несколько диапазонов значений индексов, которые записываются через запятую в скобках после имени массива. Пример объявления двумерного и трехмерного массивов:

```
Dim МассивA( 1 To 100, -5 To 4)
As Double
Dim МассивB( 19,49,100) As
String* 10
```

Упражнение 8

- Сделать приложение по рис. 43.
- Запустить приложение, проверить работу.
- Сделать приложение по рис. 44. Массив объектов OLE делается копированием вынесенного на форму объекта. Выбор объекта сделать по рис. 46.

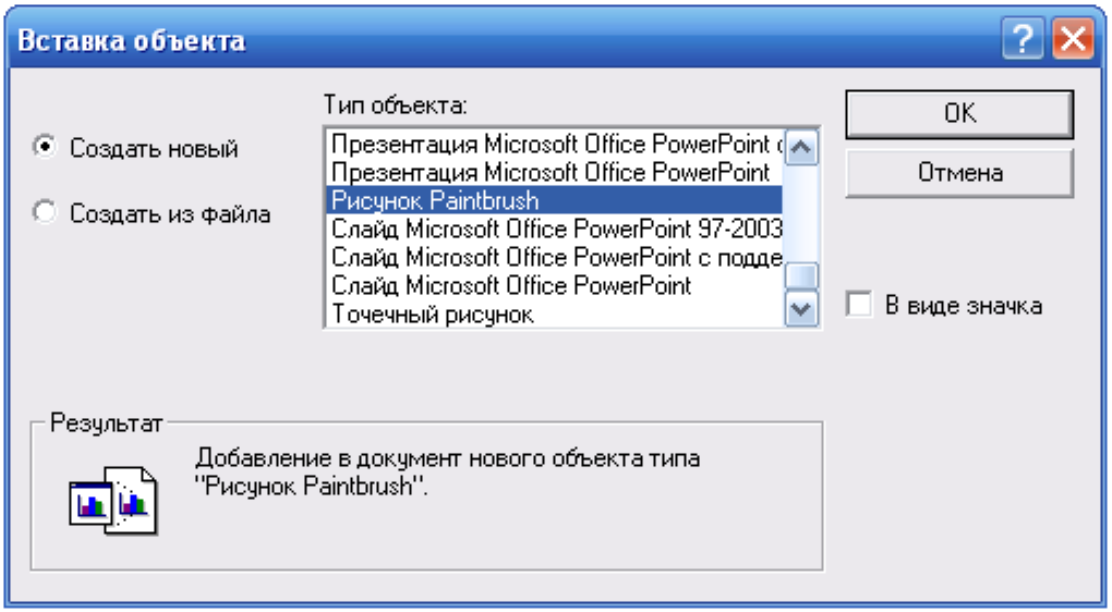

Рис. 46

• Заполнение контейнеров OLE выполняется через редактор Paint-Brush (рис. 47).

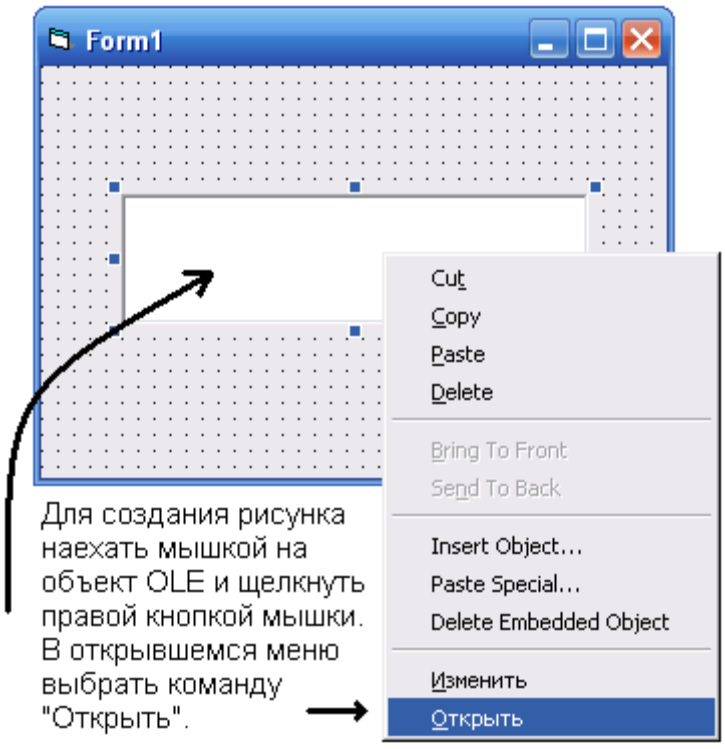

Рис. 47

• Запустить приложение, в процессе работы отредактировать рисунки.

Задания для самостоятельной работы

Каждое задание состоит из двух частей: а) и б).

а) Одномерные массивы

1. Подсчитать количество положительных, отрицательных, нулевых значений, количество целых и дробных чисел, хранящихся в одномерном массиве из 10 элементов.

2. Найти наибольший и наименьший элементы одномерного массива из 10 элементов.

3. Подсчитать отдельно сумму четных и нечетных элементов одномерного массива из 10 элементов.

4. Вычислить сумму и произведение элементов одномерного массива из 10 элементов. Разделить первую величину на вторую.

5. Дан массив целых чисел *A*. Четные значения элементов массива записать в массив *B*, нечетные – в массив *C*. Подсчитать количество тех и других.

6. Протабулировать функцию

$$f(x) = \begin{cases} x^2 + x, & \text{если } x < 0 \\ 25, & \text{если } x = 0 \\ e^x + \dfrac{1}{x+2}, & \text{если } x > 0 \end{cases}$$

на интервале [-5; 5] с шагом изменения аргумента, равным 1. Значения аргумента записать в массив *X*, результат – в массив *Y*.

7. Координаты точек ломаной линии находятся в двух одномерных массивах *X* и *Y*. Определить, в какой четверти лежит каждая точка.

8. Два одномерных массива *X* и *Y* (из 5 элементов каждый) объединить в один – *Z*. Четные элементы массива *Z* состоят из элементов массива *X*, нечетные – из элементов массива *Y*.

9. В одномерном массиве хранятся значения температуры воздуха в течение суток с интервалом в один час. Определить средние температуры дня и ночи, среднесуточную температуру.

10. Сгенерировать массив случайных чисел (от 0 до 10) из 25 элементов с помощью функции `Rnd`. Перед использованием `Rnd` инициализируйте генератор случайных чисел с помощью `Randomise`. Подсчитать, сколько раз встречается каждое число. Счетчик организовать в виде другого одномерного массива.

11. Поменять местами четные и нечетные элементы одномерного массива из 10 элементов.

12. Сформировать массив, содержащий элементы исходного массива в обратном порядке.

б) Многомерные массивы

1. Перемножить две матрицы A и B размерностью $n \times m$ и $m \times k$. Размерности матриц и значения их элементов ввести с клавиатуры. Результат вывести на экран в виде матрицы $n \times k$.

2. Сложить две матрицы A и B размерностью $n \times m$. Размерности матриц и значения их элементов ввести с клавиатуры. Результат вывести на экран в виде матрицы $n \times m$.

3. Транспонировать матрицу A размерностью $n \times m$. Размерности матрицы и значения ее элементов ввести с клавиатуры. Результат вывести на экран в виде матрицы $m \times n$.

4. Вычислить определитель матрицы A размерности 3×3.

5. Координаты точек ломаной линии хранятся в двумерном массиве. Определить, пересекает ли каждый отрезок ломаной оси абсцисс и ординат.

6. Подсчитать сумму элементов каждой строки двумерного массива размерностью $n \times m$.

7. Подсчитать среднее арифметическое каждого столбца двумерного массива размерностью $n \times m$.

8. Найти наибольший элемент каждой строки и наименьший элемент каждого столбца двумерного массива размерностью $n \times m$.

9. Из имеющегося одномерного массива из 20 элементов сформировать двумерный массив размерностью 5×4.

10. Подсчитать сумму элементов и их количество для верхней треугольной части матрицы размерностью $n \times n$.

11. Подсчитать сумму элементов главной и побочной диагоналей квадратной матрицы размерностью $n \times n$. Разделить большее число на меньшее.

12. Из двумерного массива размерностью 5×5 элементов удалить четные строки и столбцы. Результат вывести на экран в виде матрицы.

ПРИЛОЖЕНИЕ

Основные элементы управления VBA

Таблица П1

Свойства элементов управления

Свойство	*Описание*
Элемент управления `TextBox`	
`Text`	Содержимое поля
`Enable`	Можно ли вносить изменения в содержание поля (`True`) или нет (`False`)
`Multiline`	Многострочный (`True`) или однострочный (`False`) режим ввода текста
`WordWrap`	Включение/выключение (`True/False`) режима автоматического переноса
`ScrollBars`	Отображение полос прокрутки: не выводить (`fmScrollBarsNone`), только горизонтальная (`fmScrollBarsHorizontal`), только вертикальная (`fmScrollBarsVertical`), обе полосы прокрутки (`fmScrollBarsBoth`)
`SelLength`	Длина выделенного фрагмента
`SelStart`	Начало выделенного фрагмента
`SelText`	Текст выделенного фрагмента

Таблица П1

Свойства элементов управления

Свойство	*Описание*
`MaxLength`	Максимально допустимое количество вводимых символов (0 – нет ограничений)
`PasswordChar`	Определяет символ, отображаемый при вводе пароля
Элемент управления `Label`	
`Caption`	Текст надписи
`Multiline`	Многострочный (`True`) или однострочный (`False`) режим ввода текста
`WordWrap`	Включение/выключение (`True/False`) режима автоматического переноса
Элемент управления `CommandButton`	
`Caption`	Текст, отображаемый на кнопке
`Cancel`	Включение/выключение (`True/False`) отменяющего режима для кнопки (аналогичного действию при нажатии на клавишу ESC)
`Accelerator`	Определение "горячей" клавиши (клавиши, нажатие на которую одновременно с клавишей ALT приводит к выполнению действий, инициируемых нажатием управляющего элемента `CommandButton`)
`Default`	Задает кнопку по умолчанию
Элемент управления `ListBox`	
`ListIndex`	Номер текущего элемента списка (нумерация осуществляется с нуля)
`ListCount`	Число элементов списка
`TopIndex`	Элемент списка с наибольшим номером
`ColumnCount`	Число столбцов в списке
`TextColumn`	Устанавливает столбец в списке, элемент которого возвращается свойством `Text`
`Text`	Выбранный элемент
`List(row, column)`	Заданный элемент списка
`RowSource`	Диапазон, содержащий элементы списка
`ControlSource`	Диапазон (ячейка), куда возвращается выбранный элемент списка

Таблица П1

Свойства элементов управления

Свойство	*Описание*
MultiSelect	Способ выбора элементов списка: выбор только одного элемента (fmMultiSelectSingle), выбор нескольких элементов с помощью мыши или клавиши SPACE (fmMultiSelectMulti), выбор нескольких элементов с использованием клавиши SHIFT (fmMultiSelectExtended)
Selected	Выбран элемент списка (TRUE) или нет (FALSE)
ColumnWidths = "число [; число [;…]]"	Ширина столбцов списка
ColumnHeads	Вывод заголовков столбцов списка (TRUE) или нет (FALSE)
ListStyle	Пометка выделенного элемента списка цветом (fmListStylePlain) или флажком (fmListStyleOption)
MatchEntry	Режим вывода первого подходящего элемента списка при наборе его имени на клавиатуре: режим отключен (fmMatchEntryNone), вывод по первой букве (fmMatchEntryFirstLetter), вывод по полному набранному имени (fmMatchEntryComplete)
BoundColumn	Тип, возвращаемый свойством Value: индекс выбранной строки (0) или сам элемент (1)
Clear	Удаляет все элементы из списка
RemoveItem(index)	Удаляет из списка элемент с номером index
AddItem([item [, index]])	Добавляет элемент item как элемент списка с номером index
Элемент управления ComboBox	
DropButtonStyle	Вид раскрывающегося списка: без символа (fmDropButtonStylePlain), со стрелкой fmDropButtonStyleArrowDisplays), с эллипсом, (fmDropButtonStyleEllipsis), с линией (fmDropButtonStyleReduce)
ListRows	Число элементов, отображаемых в раскрывающемся списке
MatchRequired	Разрешение (True) или запрещение (False) ввода с клавиатуры значений, отличных от перечисленных в списке
MatchFound	Найден (True) или нет (False) введенный с

Таблица П1

Свойства элементов управления

Свойство	*Описание*
	клавиатуры элемент среди значений перечисленных в списке
Элемент управления `ScrollBar`	
`Value`	Текущее значение полосы прокрутки
`Min`	Минимальное значение полосы прокрутки (целое неотрицательное число)
`Max`	Максимальное значение полосы прокрутки
`SmallChange`	Шаг изменения значения при использовании стрелок полосы прокрутки
Элемент управления `SpinButton`	
`Value`	Переключатель выбран (`True`) или нет (`False`)
`Capture`	Текст, отображаемый рядом с переключателем
Элемент управления `Image`	
`Picture=Loadpic-ture(ИмяФайла)`	Задает имя отображаемого графического файла
`PictureSizeMode`	Масштабирование рисунка: обрезка не умещающихся в заданных границах частей рисунка (`fmPictureSizeModeClip`), изменение разме-ров рисунка в соответствии с границами объекта (`fmPictureSizeModeStretch`), масштабирование по границам объекта с соблюдением пропорций (`fmPictureSizeModeZoom`)
`PictureAlignment`	Расположение рисунка внутри объекта: в левом верхнем углу (`fmPictureAlignmentTopLeft`), в правом верхнем углу (`fmPictureAlignment-TopRight`), в центре (`fmPictureAlignment-Center`), в левом нижнем углу (`fmPicture-AlignmentBottomLeft`), в правом нижнем углу (`fmPictureAlignmentBottomLRight`)
`PictureTiling`	Режим расположения рисунка мозаикой (`True`)
Элемент управления `MultiPage`	
`Value`	Возвращает номер выбранной страницы
`MultiRow`	Включение/выключение (`True/False`) режима разрешения отображения ярлыков страниц в несколько строк
`SelectedItem`	Возвращает выбранную страницу
Семейство `Pages` ***элемента управления*** `MultiPage`	
`Count`	Возвращает число элементов семейства

Таблица П1

Свойства элементов управления

Свойство	*Описание*
`Set Object = object.Add ([Name [, Caption [, index]]])`	Создает новую страницу `Object` семейства `Pages`; `Name` – имя страницы, `Caption` – текст на ярлыке страницы, `index` – номер страницы (нумерация осуществляется с нуля)
`Clear`	Удаляет все страницы из семейства `Pages`
`Remove`	Удаляет страницу из семейства `Pages`
`Set Object = object.Item (coolectionindex)`	Возвращает страницу с указанным индексом
Общие свойства элементов управления	
`Name`	Имя элемента управления
`Caption`	Текст, отображаемый на элементе управления
`AutoSize`	Включение/выключение (`True/False`) режима автоматического изменения размеров элемента управления, чтобы на нем полностью отображался текст, присвоенный свойством `Caption`
`Visible`	Включение/выключение (`True/False`) режима отображения элемента управления
`Enabled`	Включение/выключение (`True/False`) режима разрешения управления объектом
`Height`	Высота объекта
`Width`	Ширина объекта
`Left, Top`	Координаты верхнего левого угла элемента управления
`ControlTipText`	Задает текст всплывающей подсказки
`BackColor`	Цвет заднего плана элемента
`ForeColor`	Цвет переднего плана элемента
`BorderColor`	Цвет границы элемента
`BackStyle`	Тип (стиль) заднего фона
`BorderStyle`	Устанавливает один тип (стиль) границы, но различных цветов: `fmBorderStyleSingle` (граница в виде контура), `fmBorderStyleNone` (невидимая граница)
`SpecialEffect`	Устанавливает несколько типов (стилей) границы, но одного цвета
`Picture`	Внедряет графическое изображение на элемент управления
`Tag`	Используется для хранения дополнительной

Таблица П1

Свойства элементов управления

Свойство	*Описание*
	информации о форме или элементе управления

Таблица П2

Некоторые символические константы VBA

Константа	*Значение*	*Цвет*
`vbBlack`	0×0	Черный
`vbRed`	0×FF	Красный
`vbGreen`	0×FF00	Зеленый
`vbYellow`	0×FFFF	Желтый
`vbBlue`	0×FF0000	Синий
`vbMagenta`	0×FF00FF	Розовый
`vbCyan`	0×FFFF00	Голубой
`vbWhite`	0×FFFFFF	Белый

Таблица П3

Основные методы и события объекта `UserForm`

Метод/событие	*Описание*
`Show`	Отображает форму на экране
`Hide`	Закрывает форму
`Move`	Изменяет положение и размер формы
`PrintForm`	Печатает изображение формы
`Initialize`	Происходит при отображении формы на экране
`Terminate`	Происходит при закрытии формы

ЛИТЕРАТУРА

1. Вычислительная техника и программирование: Учебник для техн. вузов / А.В. Петров, В.Е. Алексеев, А.С. Ваулин [и др.]; под ред. А.В. Петрова. – М.: Высшая школа, 2000. – 537 с.
2. Дубина, А.Г. Расчёты в среде Excel / А.Г. Дубина. – СПб.: BHV - Санкт-Петербург, 2006. – 416 с.
3. Информатика: Базовый курс / С.В. Симонович [и др.]. – Спб.: Питер, 2001. – 640 с.
4. Информатика: Учебник для экономических специальностей / Н.В. Макарова [и др.]. – М.: Финансы и статистика, 2003. – 767 с.
5. Карпов, Б. Microsoft Excel: справочник / Б. Карпов. – СПб.: Питер,2007. – 544 с.
6. Хэлворсон, М. MS Office. Ч.3: Пер. с англ. / М. Хэлворсон. – СПб.: Питер,2008. – 644 с.