Е.А. Быкадорова, О.Н. Синявская

Основы программирования информационного контента

Учебное пособие
для студентов специальности
09.02.05 Прикладная информатика (по отраслям)
и преподавателей профессиональных образовательных учреждений

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
РОСТОВСКОЙ ОБЛАСТИ
«КАМЕНСКИЙ ПЕДАГОГИЧЕСКИЙ КОЛЛЕДЖ»

Основы программирования информационного контента

Учебное пособие
для студентов специальности
09.02.05 Прикладная информатика (по отраслям)
и преподавателей профессиональных образовательных учреждений

Каменск-Шахтинский
2016 г.

ББК 32.973-018
Б-95

Рецензенты: Назаров С.А., к.п.н., доцент кафедры естественно-научных и общеинженерных дисциплин Каменского института (филиала) ФГБОУ ВПО «ЮРГТУ (НПИ)»
Звездунова Г.В., к.п.н., доцент, заместитель директора по учебно-методической работе ГБПОУ РО «Каменский педагогический колледж»

Быкадорова Е.А., Синявская О.Н. Основы программирования информационного контента: Учебное пособие для студентов специальности 09.02.05 Прикладная информатика (по отраслям) и преподавателей профессиональных образовательных учреждений. – Ростов н/Д, 2016. – 72 с.

Учебное пособие составлено в соответствии с ФГОС СПО и предназначено для подготовки выпускников по специальности 09.02.05 Прикладная информатика (по отраслям), одобрено предметной (цикловой) комиссией преподавателей информатики и математики ГБПОУ РО «Каменский педагогический колледж».

Пособие содержит теоретический материал по основным разделам программы МДК «Основы программирования информационного контента» на языке Visual Basic, может использоваться для самостоятельной аудиторной и внеаудиторной работы студентов.

ОГЛАВЛЕНИЕ

Стр.

ВВЕДЕНИЕ

Данное пособие по тематике, уровню сложности и применяемым методическим подходам соответствует ФГОС специальности «Прикладная информатика (по отраслям)» и рабочей программе по МДК «Основы программирования информационного контента».

Основной задачей учебного пособия является обучение студентов процессу программирования на языке Visual Basic и развитие у них общих и профессиональных компетенций в области информационных вычислений.

Содержание учебного пособия предусматривает систематическое раскрытие взаимосвязи теоретических и прикладных аспектов алгоритмизации, роли и значения программирования и использования ПК при решении конкретных задач.

Пособие охватывает следующие разделы:

1. Языки и методы программирования
2. Основы языка программирования Visual Basic.

Пособие предназначено для реализации профессиональных компетенций в сфере деятельности, где основы алгоритмизации и программирования является одним из профилирующих направлений. Выполнение работ в объеме данного пособия позволит закрепить полученные знания в области программирования, связанной с прикладным решением задач. Тематика заданий определяется логикой изучения основных алгоритмических структур.

В учебном пособии также включены задания для самостоятельного выполнения, что позволяет студентам проявить индивидуальный подход к разработке проектов.

Материалы, используемые в учебном пособии, прошли апробацию в учебном процессе Каменского педагогического колледжа и могут быть рекомендованы для подготовки специалистов по специальности 09.02.05 Прикладная информатика (по отраслям).

РАЗДЕЛ 1. Языки и методы программирования

Тема 1.1. Классификация языков программирования

Языки программирования развивались одновременно с развитием ЭВМ. С начала 50-х годов это были низкоуровневые языки (машинные и ассемблеры). В 1956 году появился язык Фортран, а в 1960 – Алгол-60. Это языки компилирующего типа, существенно уменьшившие трудоемкость программирования. Языки ориентированы на выполнение математических вычислений. В дальнейшем возникло большое количество различных языков, претендовавших на универсальность (PL/1) или для решения конкретных задач (COBOL – для деловых задач, ЛОГО – для обучения, Пролог – для разработки систем искусственного интеллекта). С середины 60-х до начала 80-х разработаны и получили распространение языки Pascal, Basic, Си, Ада и другие.

Принципиально новым этапом в развитии языков программирования стало появление методологии непроцедурного (ООП) программирования. Основные достоинства ООП – быстрота разработки интерфейса программного приложения, возможность наследования свойств программных объектов.

Язык программирования — формальная знаковая система, предназначенная для описания алгоритмов в форме, которая удобна для исполнителя (например, компьютера).

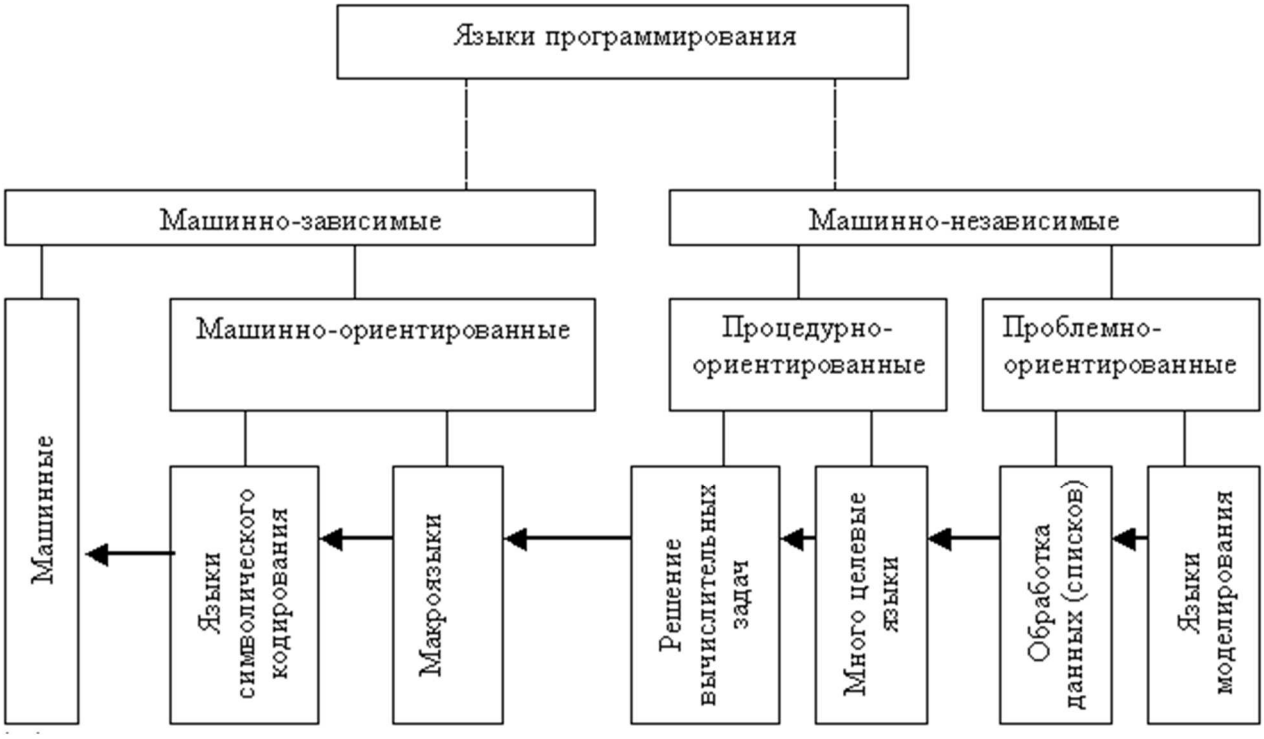


Рис. 1. Классификация языков программирования

Язык программирования определяет набор лексических, синтаксических и семантических правил, используемых при составлении компьютерной программы. Он позволяет программисту точно определить то, на какие события будет реагировать компьютер, как будут храниться и передаваться данные, а

также какие именно действия следует выполнять над этими данными при различных обстоятельствах.

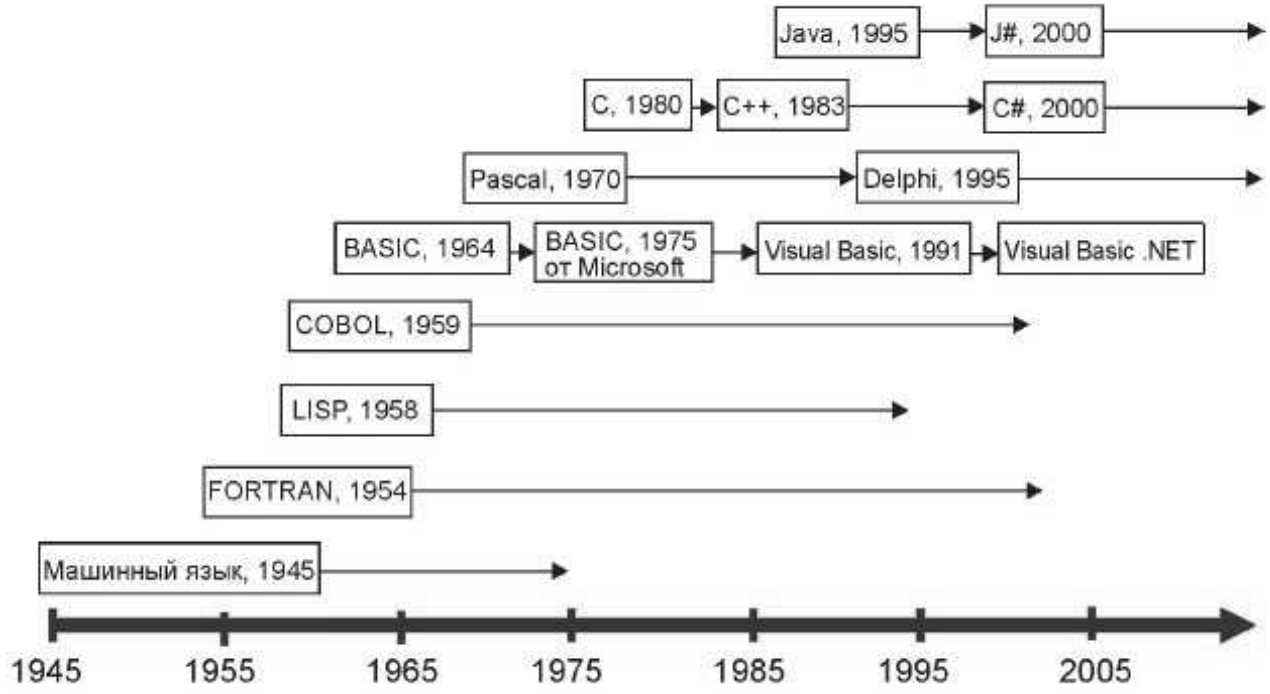


Рис. 2. История развития языков программирования

Среди программистов, пишущих программы для персональных компьютеров, наибольшей популярностью пользуются языки Си, Си++, Паскаль и Бейсик.

История изобретения языков программирования

Язык Си был изобретен в 1972 году Денисом Ричи и Кеном Томпсоном для использования в написании операционной системы Unix. Си соединяет свойства языка высокого уровня с возможностью эффективного использования ресурсов компьютера, которое обычно обеспечивается только при программировании на языке Ассемблера. Язык Си не очень прост в обучении и требует тщательности в программировании, но позволяет писать сложные и весьма высокоэффективные программы.

Язык **Паскаль** был разработан в 1970 году Никлаусом Виртом как язык для обучения студентов программированию. Паскаль позволяет писать программы, легко читаемые даже новичком, и содержит в себе все элементы, необходимые для соблюдения хорошего строгого стиля программирования (называемого структурным программированием), упрощающего разработку сложных программ.

Язык **Бейсик** был создан в 1964 году Томасом Куртом и Джоном Кемени как язык для начинающих, облегчающий написание простых программ. Существует много различных версий Бейсика. Этот язык очень широко распространен на микрокомпьютерах. На IBM PC широко используются Quick Basic и Visual Basic фирмы Microsoft и Turbo Basic фирмы Borland. Основная

идея авторов языка Бейсик – снабдить простым языком программирования непрофессиональных программистов.

Понятие программы, ее виды

Программа — это детальное и законченное описание алгоритма средствами языка программирования. Исполнителем программы является компьютер. Для выполнения компьютером программа должна быть представлена в машинном коде – последовательности чисел, понимаемых процессором. Написать программу в машинных кодах вручную достаточно сложно. Поэтому сегодня практически все программы создаются с помощью языков программирования, которые по своему синтаксису и семантике приближены к естественному человеческому языку. Это снижает трудоемкость программирования.

Текст программы, записанный с помощью языка программирования, должен быть преобразован в машинный код. Эта операция выполняется автоматически с помощью специальной служебной программы, называемой *транслятором*.

Трансляторы делятся на два типа: *интерпретаторы* и *компиляторы*.

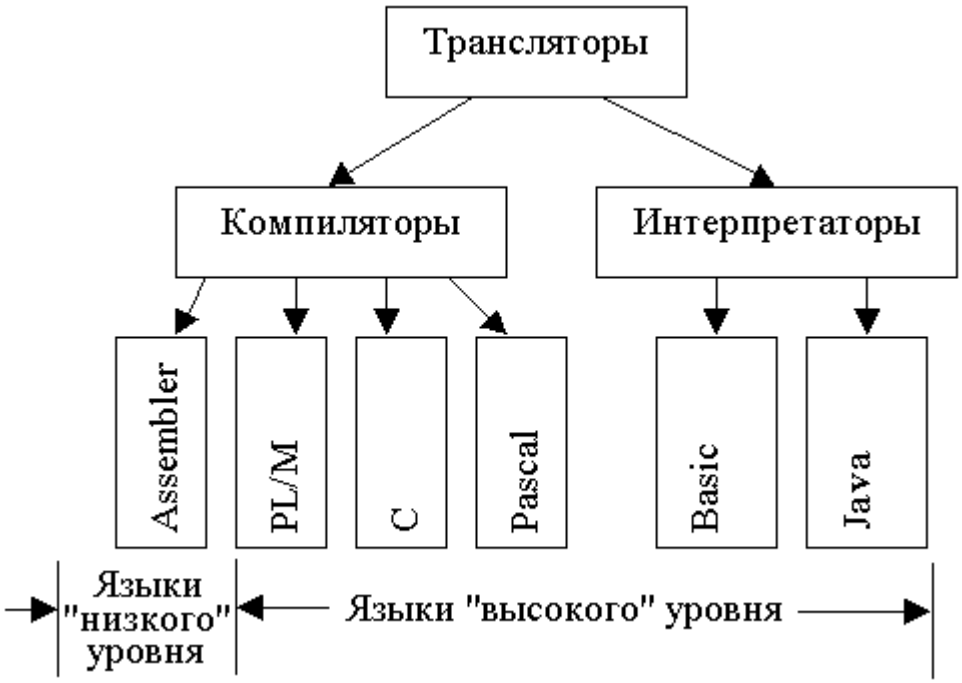


Рис. 3. Классификация трансляторов языков программирования

Интерпретатор переводит в машинный код и выполняет очередной оператор (команду) программы. Если команда повторяется, то интерпретатор рассматривает ее как встреченную впервые. Достоинство интерпретаторов – их компактность, возможность остановить в любой момент выполнение программы, выполнить различные преобразования данных и продолжить работу программы. Примерами служебных программ – интерпретаторов являются VB Basic, Лого, школьный алгоритмический язык, многие языки программирования баз данных.

Компилятор переводит в машинный код исходный текст программы целиком. Главное достоинство компиляторов – быстродействие и автономность получаемых программ. Компиляторами являются Turbo Pascal, C++, Delphi.

Средства создания программ

В общем случае для создания программ нужно иметь следующие компоненты:

- текстовый редактор – для набора исходного текста программы;
- компилятор – для перевода текста программы в машинный код;
- редактор связей – для сборки нескольких откомпилированных модулей в одну программу;
- библиотеки функций – для подключения стандартных функций к программе.

Современные системы программирования включают в себя все указанные компоненты и называются *интегрированными системами*.

По способу разработки программ можно выделить два подхода:

- ***процедурное программирование*** — это такой подход к программированию, при котором выполнение команд программы определяется их последовательностью, командами перехода, цикла или обращениями к процедурам;
- ***объектно-ориентированное программирование*** – такой подход к программированию, при котором формируются программные объекты, имеющие набор свойств, обладающие набором методов и способные реагировать на события, возникающие как во внешней среде, так и в самом объекте (нажатие мыши, срабатывание таймера, превышение числовой границы и т.д.). Таким образом, выполнение той или иной части программы зависит от событий в программной системе.

Объектно-ориентированное программирование (ООП) не исключает, а охватывает технологию процедурного программирования.

Тема 1.2 Алгоритм и его свойства. Основные алгоритмические структуры

Алгоритм — это предписание некоторому исполнителю выполнить конечную последовательность действий, приводящую к некоторому результату.

В программировании алгоритм является фундаментом программы, а основным *исполнителем* – компьютер. На стадии тестирования алгоритма исполнителем может быть сам программист.

Основными свойствами алгоритма являются:

- ***дискретность*** — представление алгоритма в виде последовательности шагов;
- ***массовость*** — применимость алгоритма к некоторому множеству исходных данных;

- ***результативность*** — способность получать требуемый результат за конечное число шагов;
- ***детерминированность*** — способность получать один и тот же результат на каждом шаге для одного и того же набора исходных данных;
- ***определенность*** — каждое действие алгоритма должно быть понятно исполнителю.

Из перечисленных свойств лишь дискретность является обязательным свойством алгоритма. Можно привести примеры, когда невыполнение свойств массовости, определенности и однозначности не позволяет говорить об отсутствии алгоритма.

Для изображения алгоритмов в наглядной форме используется блок-схема (рис. 4).

Рис. 4. Типовые блоки описания алгоритма

Основные алгоритмические структуры

В теории алгоритмов доказано, что любой, сколь угодно сложный алгоритм может быть составлен из трех основных алгоритмических структур: линейной, ветвления и цикла, показанных, соответственно на Рис. 5.

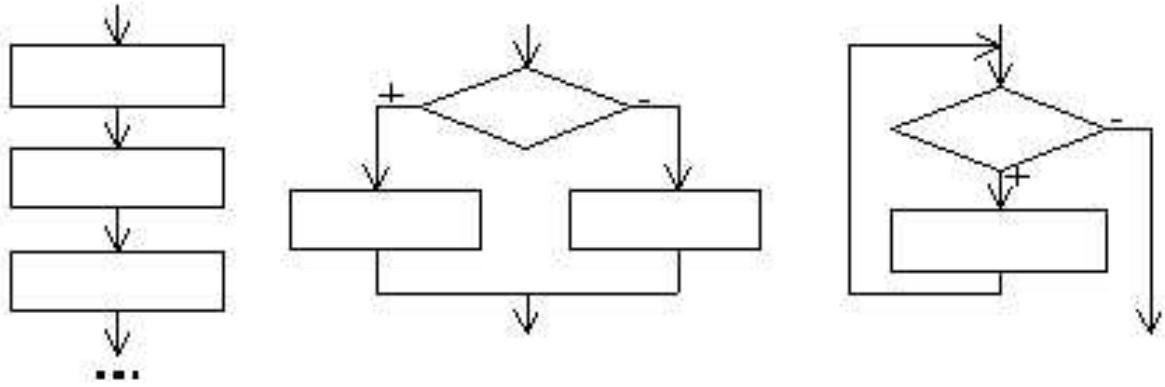


Рис. 5. Основные алгоритмические структуры (слева–направо): линейная структура, ветвление, циклический алгоритм.

Линейная структура предполагает последовательное выполнение действий, без их повторения или пропуска некоторых действий. Обычно программисты стремятся к тому, чтобы алгоритм имел линейную структуру.

Структура «ветвление» предполагает выполнение одной из двух групп действий в зависимости от выполнения условия в блоке ветвления. На рис. 5 знаками «+» показано выполнение условия, а знаками «–» обозначена ситуация его невыполнения. Часто используется неполная команда ветвления, когда один из блоков действия отсутствует.

Структура «цикл» имеет несколько разновидностей.

На Рис. 4 показан цикл типа «пока» с предусловием. Действия внутри этого цикла повторяются, пока выполняется условие в блоке ветвления, причем сначала проверяется условие, а затем выполняется действие. Достаточно часто используются другие типы цикла, показанные на рис. 5 и 6.

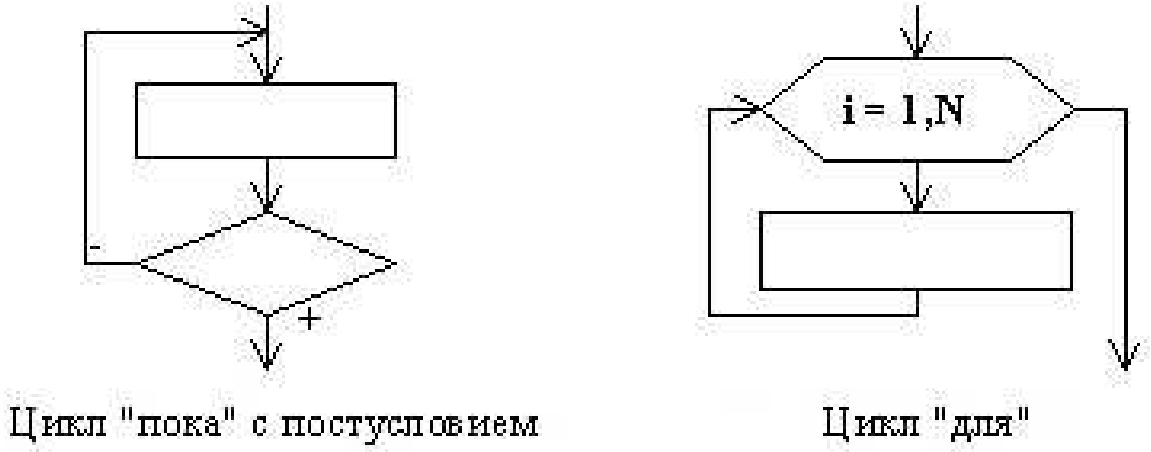


Рис. 6. Структуры циклов

В цикле с постусловием проверка условия выхода из цикла выполняется после очередного действия.

Цикл «для» является модификацией цикла «пока» для ситуации, когда заранее известно количество повторений некоторых действий.

В языках программирования имеются команды, реализующие показанные выше структуры.

При разработке блок-схемы допускается делать любые записи внутри блоков, однако эти записи должны содержать достаточно информации для выполнения очередных действий.

Формализация и моделирование задачи

Разработке алгоритма предшествуют такие этапы, как формализация и моделирование задачи.

Формализация предполагает замену словесной формулировки решаемой задачи краткими символьными обозначениями, близкими к обозначениям в языках программирования или к математическим.

Моделирование задачи является важнейшим этапом, целью которого является поиск общей концепции решения. Обычно моделирование

выполняется путем выдвижения гипотез решения задачи и их проверке любым рациональным способом (прикидочные расчеты, физическое моделирование и т.д.). Результатом каждой проверки является либо принятие гипотезы, либо отказ от нее и разработка новой.

При разработке алгоритма используют следующие основные принципы.

Принцип поэтапной детализации алгоритма (другое название – «проектирование сверху-вниз»). Этот принцип предполагает первоначальную разработку алгоритма в виде укрупненных блоков (разбиение задачи на подзадачи) и их постепенную детализацию.

Принцип «от главного к второстепенному», предполагающий составление алгоритма, начиная с главной конструкции. При этом, часто, приходится «достраивать» алгоритм в обратную сторону, например, от середины к началу.

Принцип структурирования, т.е. использования только типовых алгоритмических структур при построении алгоритма. Нетиповой структурой считается, например, циклическая конструкция, содержащая в теле цикла дополнительные выходы из цикла. В программировании нетиповые структуры появляются в результате злоупотребления командой безусловного перехода (GoTo). При этом программа хуже читается и труднее отлаживается.

Основные этапы компьютерного решения задач

1. **Постановка задачи.** Основное требование к постановке задачи – достаточное количество информации для решения задачи. Очень часто постановка задачи выполняется не программистом, а некоторым Заказчиком. Программист является Исполнителем заказа. От него требуется добиться от Заказчика полной информации о решаемой задаче.
2. **Моделирование и формализация задачи.** Цели этого этапа уже обсуждались выше в разделе методики разработки алгоритма. При моделировании важно иметь опыт программирования, знать возможности компьютера и языка программирования и выдвигать гипотезы с учетом этих возможностей. К разработке алгоритма следует приступать только после принятия гипотезы решения задачи. К категории «Дано» обычно относятся данные, вводимые в начале работы программы и обеспечивающие массовость алгоритма. К категории «Найти» относятся данные, получаемые в результате работы программы.
3. **Разработка алгоритма.** Этот этап представляет собой реализацию идеи решения задачи.
4. **Тестирование алгоритма.** Этап предполагает проверку алгоритма вручную с использованием подготовленных ранее контрольных примеров. Для сложных задач этот этап может оказаться весьма трудоемким, поэтому опытные программисты пропускают его и тестируют программу.
5. **Программирование алгоритма.** Программирование является формальной записью алгоритма средствами языка программирования.

6. **Тестирование программы.** Тестирование выполняется путем вывода промежуточных результатов работы программы и сравнения их с контрольным примером. Для этого либо используют специальные средства отладки программ, имеющиеся в интегрированной среде языка программирования, либо временно добавляют в программу команды вывода промежуточных значений. Уменьшить трудоемкость поиска ошибок в программе можно более тщательным проектированием алгоритма и планированием процесса тестирования на ранних стадиях разработки программы.

7. **Эксплуатация программы и интерпретация результатов.** В сложных программах может быть недостаточно тестирования для устранения всех ошибок. Очень часто они обнаруживаются на стадии эксплуатации Заказчиком. Успех в разработке программы зависит от двух основных факторов: соблюдения описанной выше методики и от опыта программирования.

Задания для самостоятельной работы

1. Заполните таблицу «История развития языков программирования».
2. Составьте схему «Классификация языков программирования».
3. Заполните таблицу «Блок-схема алгоритма и ее элементы».
4. Составьте алгоритм преобразования слова «информатика» в слово «форма».
5. Составьте и зафиксируйте в форме блок-схемы алгоритм вычисления площади и гипотенузы прямоугольного треугольника, если известны его катеты.
6. Составьте и зафиксируйте в форме блок-схемы алгоритм выбора большего из двух чисел.
7. Составьте и зафиксируйте в форме блок-схемы алгоритм нахождения суммы натуральных чисел.

РАЗДЕЛ 2. Объектно-ориентированное программирование на языке Visual Basic

Тема 2.1. Основы объектно-ориентированного визуального программирования

Интегрированная среда разработки языка Visual Basic

Visual Basic – система программирования, так как позволяет кодировать алгоритмы на этом языке, и среда проектирования, так как позволяет осуществлять визуальное конструирование графического интерфейса.

Результатом процесса программирования и проектирования является проект (Project), который объединяет в себе программный код и графический интерфейс. VB содержит программу – транслятор, поэтому проекты могут выполнятся в системе программирования, а также могут быть преобразованы в приложения, выполняемые в ОС Windows.

Запуск программы осуществляется следующим образом – значок VB 5.0 на рабочем столе или Пуск → Программы → Microsoft Visual Basic 5.0. После запуска программы появится приглашение (рис. 7), в котором необходимо выбрать «Standard EXE» и нажать кнопку «Открыть». Данным действием выбирается тип проекта, который будет создаваться.

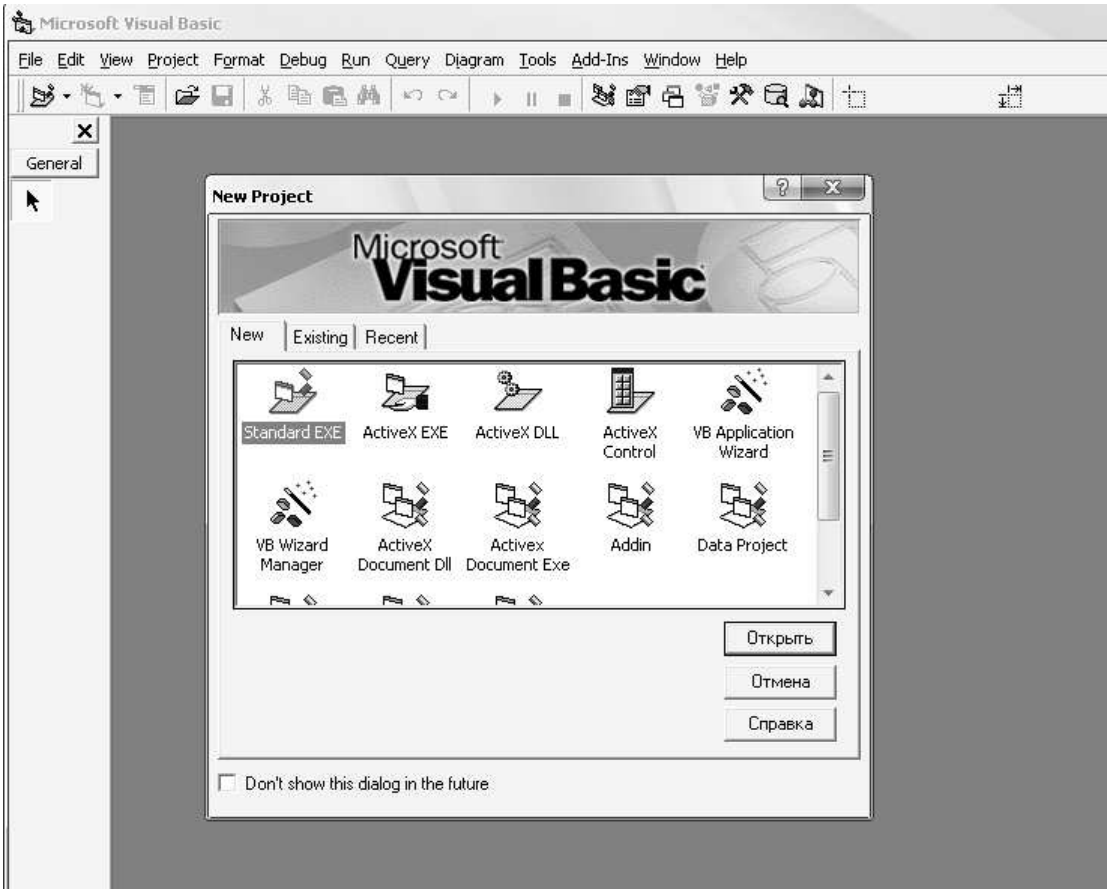


Рис. 7. Окно приглашения

Общий вид Интегрированной Среды Разработчика – ИСР (англ. IDE) – подготовленной для создания нового проекта, показан на рис.8.

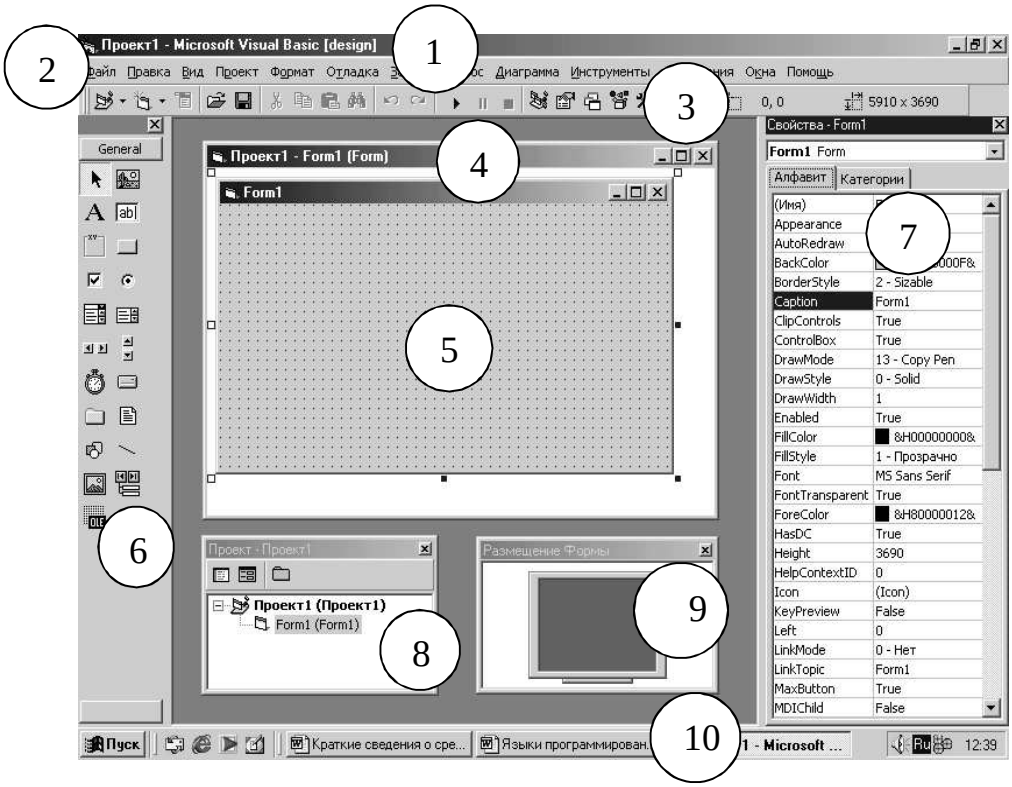


Рис. 8. Общий вид Интегрированной Среды Разработчика

1 – заголовок, 2 – главное меню, 3 – панель инструментов, 4 – окно проектирования формы, 5 – форма, 6 – панель элементов управления, 7 – окно свойств, 8 – окно проводника проекта, 9 – окно размещения формы, 10 – панель задач.

Заголовок состоит из названия системы программирования Microsoft Visual Basic, левее этих слов расположено название проекта – Project1. Это название Visual Basic присвоил автоматически, его нужно будет заменить каким-либо более осмысленным словосочетанием. В правой части заголовка обозначено design – дизайн – конструирование, разработка, это слово отражает этап работы над проектом. Другие возможные этапы работы: run – выполнение и break – прерывание.

Строка меню и панель инструментов во многом совпадают с меню и панелью Windows, однако, в них имеются меню и инструменты, которые обеспечивают доступ к специальным средствам Visual Basic.

Панель **ToolBox** – панель объектов или палитра объектов. Она содержит набор специальных инструментов - графических объектов, которые можно комбинировать, размещая их в специальном окне - окне форм (Form).

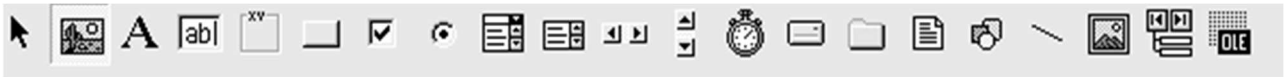


Рис. 9. Стандартный набор инструментов среды Visual Basic

Таблица 1

Элементы управления среды Visual Basic

Элемент управления	Название	Назначение
	Метка (Label)	Для вывода информации на форму, создания надписей
	Текстовое окно (TextBox)	Для ввода информации с клавиатуры и для вывода информации на форму
	Командная кнопка (CommandButton)	Для написания процедуры события
	Графическое окно (PictureBox)	Для вывода на форму графического файла
	Флажок (CheckBox)	Для выбора нескольких вариантов
	Кнопка – переключатель (OptionButton)	Для выбора альтернативного варианта
	Окно списка (ListBox)	Для ввода и вывода массивов информации
	Окно комбинированного списка (ComboBox)	Для ввода и вывода массивов информации
	Список устройств (Drive ListBox)	Для выбора устройства
	Список каталогов (Directory List)	Для выбора каталога (папки)
	Список файлов (FileListBox)	Для выбора файла
	Общий диалог (CommonDialog)	Для создания стандартных диалоговых окон

В серединной части экрана расположено **окно проектов**, озаглавленное **Project1- Form1(Form)**. Внутри него размещено окно дизайнера (конструктора) форм, чаще его называют просто окном форм. Его название Form1, автоматически присваивается Visual Basic, и должно быть впоследствии изменено.

Окно **Project Explorer** – ***Проводник Проекта*** или окно проекта позволяет анализировать структуру проекта и его состав.

Рис. 10. Окно проводника проекта с представлением содержимого проекта в виде дерева и в виде списка

Окно проводника проекта содержит свою маленькую линейку инструментов с кнопками, которые предназначены для различных целей:

- **View Code** – предназначена для открытия программного кода и представлена такой кнопкой  ;
- **View Object** – предназначена для открытия окна экранной формы и представлена такой кнопкой  ;
- **Toggle Folders** – для переключения папок, имеет такой графический вид:  .

Окно **Properties** – ***окно свойств*** – очень удобный справочник по свойствам объектов.

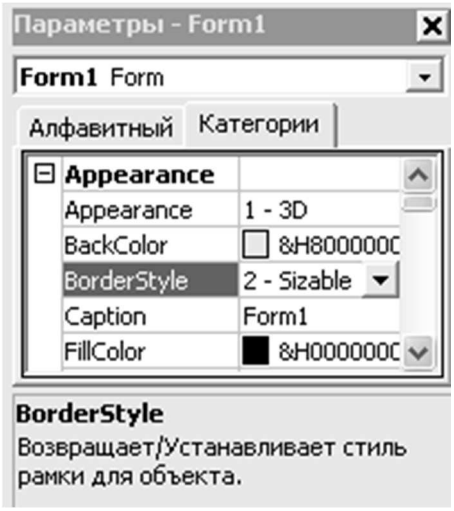


Рис. 11. Окно свойств объекта Form1

Окно **Form Layout** – ***Макет Формы*** позволяет выбрать на экране место, в котором будет выполняться программа после окончания всех этапов разработки.

Еще одно окно – **Code** – ***окно кодов***, предназначенное для создания и редактирования кода (текста) программы в исходном состоянии ИСР, не видно.

Основные термины и определения

Программный объект – основная единица в объектно-ориентированном программировании.

Программные объекты имеют *имя* (существительные, *Что?*) обладают *свойствами* (прилагательные, *Какой?*), могут использовать *методы* (глаголы, *Что делает*?) и реагируют на *события*.

Объектом называется фрагмент кода, обладающий свойствами и методами. Объекты создаются на базе так называемых классов и могут моделировать правила обработки данных, различные ситуации и даже физические предметы.

Объект, созданный по «шаблону» класса объектов, является *экземпляром класса* и наследует весь набор свойств, методов и событий данного класса. Каждый экземпляр класса объектов имеет уникальное для данного класса имя. *Классы объектов* являются «шаблонами», определяющими наборы *свойств, методов и событий.* По шаблонам создаются объекты. Основными классами объектов являются объекты, реализующие графический интерфейс.

В первую очередь, в Visual Basic выделяют объекты, привлекаемые к проектированию приложений и разделяемые на категории:

1. **Глобальные объекты** – некоторые компоненты программно-аппаратной среды – буфер обмена, экран монитора, средства отображения результатов отладки и т.д.
2. **Группа объектов интерфейса** – состоит из общей (Form) и специальной форм и большого набора элемнтов управления.

Каждый объект обладает определенным набором свойств (Properties).

Свойство (Property) – это параметр, значение которого определяет внешний вид, функциональную характеристику или другую особенность объекта в среде Visual Basic. Каждый тип объектов имеет собственное свойство, а каждое свойство соответствующее значение. Тип значения зависит от типа свойства, поэтому значения могут быть числовыми, строковыми, логическими или каким либо другим (например: значениям свойства Color – является определённый цвет). Значения одних свойств указывается в полях ввода, значения других – выбирается из определённого списка или устанавливается при помощи специальных диалоговых окон. Все свойства любого объекта имеет значение по умолчанию, следовательно, определить значения всех свойств объекта вам не придётся.

В процессе интерактивного взаимодействия со своим приложением пользователь имеет возможность управлять отдельными свойствами объектов – перемщать их по экрану, изменять размеры и т.д. С каждым объектом связан набор событий.

События (Event) – определяет реакцию программы на какое-либо действие пользователя – щелчок мыши, нажатие клавиши на клавиатуре либо на события, происходящие в системе, например прерывание системного таймера. Именно события являются первичным фактором, инициирующим дальнейшую обработку – обращение к методам и свойствам объектов, смену

форм на экране и т.д. Написание программ обработки событий полностью возлагается на программиста.

Например: событие Click (щелчок мыши) инициирует одну или несколько операций связанных с кодом программы – смену форм, выполнение определённых функций. Выход из программы.

Последний термин связан с методами-процедурами, действие которых разворачивается на фоне того или иного объекта.

Метод (Method) — это подпрограмма, позволяющая динамически (в ходе работы программы) получать или изменять значения свойств объекта. Для всех типов объектов определены наборы методов. Набор методов объекта пользователь изменить не может, то есть он не может добавить или удалить какой-либо метод. Полный перечень свойств и методов для каждого объекта содержится в соответствующем разделе файле помощи и в руководстве пользователя.

Программный модуль языка Visual Basic представляет собой текстовый ASCII файл содержащий подпрограммы, функции, переменные и или константы.

У объектов, как и у любых компонентов Visual Basic есть свойство: методы и события, но они отличаются от процедур тем, что поддерживают три важных принципа:

1. Наследование.
2. Инкапсуляцию.
3. Полиморфизм.

Наследование — это способность объекта сохранять атрибуты класса родителя.

Инкапсуляция — возможность защищать одни фрагменты программы от доступа, другие предоставить в распоряжение программы, обеспечивается инкапсуляцией. Это достигается с помощью двух ключевых слов:

- *Public* – открыта;
- *Private* – закрыта.

Полиморфизм — это способность объекта принимать различные формы, объекты могут быть производными от других объектов. Новый объект наследует методы и свойства объекта родителя. Полиморфизм позволяет добавлять, видоизменять или удалять некоторые особенности поведения производного объекта.

Этапы разработки проекта

1. Создание графического интерфейса проекта
2. Установка значений свойств объектов графического интерфейса
3. Создание и редактирование программного кода
4. Сохранение проекта

Форма и размещение на ней управляющих элементов

Графический интерфейс проекта представляет собой форму, на которой размещены управляющие элементы.

Форма выполняет роль основы, на которой собираются все элементы управления, обеспечивающие интерфейс приложения с пользователем и другими источниками информации.

Виды форм:

1. Форма (Form) – обычная форма, используемая в несложных программах.
2. Основная форма (MDI Form) – это форма, которая может содержать дочерние (вложенные) формы. В приложении может быть только одна такая форма.
3. Дочерняя форма (Child) – содержится только внутри основной формы. Таких форм в приложении может быть несколько.
4. Форма диалога (Dialog) – появляется на экране на короткое время, служит для ввода или вывода информации, не изменяется в размерах и находится поверх других окон.

Общее число характеристик формы превышает 50. Некоторые из них недоступны во время конструирования приложения, поскольку имеют смысл только во время выполнения программы. К их числу относятся такие параметры как:

- количество загруженных форм (Count),
- координаты текущей точки для графических построений и т.д.

Перечень свойств формы, которыми можно управлять на стадии создания приложения, насчитывает 43 наименования. Далеко не все из них нуждаются в индивидуальной перенастройке. Параметры формы, значения которых придется изменить представлены в таблице 2.

Таблица 2

Свойства формы

Свойство	Назначение
Caption	Название, которое будет написано в заголовке формы или объекта управления при исполнении программы. Заголовок окна (Caption) присутствует только в верхней строке окна приложения. Он может включать русские и латинские буквы, состоять из нескольких слов, разделенных пробелом. Максимальная длина заголовка окна – 255
Name	Имя, под которым форма или объект управления могут упоминаться в программе. При включении в проект форм или размещении объектов управления на формах транслятор автоматически присваивает им имена, однако при желании можно присвоить этому свойству любое другое имя.

Picture	Позволяет заполнить поверхность формы или объекта управления рисунком, имеющим расширение .bmp. Для этого при проектировании программы нужно щелкнуть по соответствующей строке в окне свойств и, пользуясь открывшимся окном, указать путь к соответствующему файлу.

Управляющие элементы — объекты, являющиеся элементами графического интерфейса приложения и реагирующие на события, произведенные пользователем или программными объектами.

Форма и управляющие элементы обладают определенными наборами свойств, методов и событий.

Визуальное проектирование графического интерфейса состоит в том, что на форму с помощью мыши перемещаются и «рисуются» управляющие элементы.

Классы управляющих элементов (Controls):

- для ввода/вывода данных*: Текстовые поля* (TextBox), *метки* (Label), *списки* (ListBox).
- для вывода графики Графические окна (PictureBox).
- для организации диалога *командные кнопки* (CommandButton), *переключатели* (OptionsButton).

Таблица 3

Некоторые классы объектов, их свойства, методы и события

Класс объектов	**Свойства**	**Методы**	**События**
Form (форма) User Form	Name (Имя) Caption (Надпись) Font (Шрифт) Height (Высота) Width (Ширина)	Show (Показать)	Load (Загрузка)
Command Button (командная кнопка)	Name (Имя) Caption (Надпись) Font (Шрифт) Height (Высота) Width (Ширина)	Move (Переместить)	
Text Box (текстовое поле)	Name (Имя) Text(Текст) Font (Шрифт) Height (Высота) Width (Ширина)		DblClick (Двойной щелчок)

Для каждого объекта можно запрограммировать *отклик* (событийную процедуру) – реакцию объекта на произошедшее событие, например:

- Click (щелчок по объекту мышью),
- Resize (изменение размера объекта),
- Load (загрузка объекта) и т.д.

Событийная процедура — подпрограмма, которая начинает выполняться после реализации определенного события.

Задания для самостоятельной работы

Создайте проект «Вывод сообщения», в котором на форму выводится текстовое сообщение «Первое задание выполнено!» с помощью метки и текстового поля, а выход из программы реализуется щелчком по кнопке «Выход». Продумать графический интерфейс проекта.

Тема 2.2. Введение в язык Visual Basic

Структура языка программирования Visual Basic. Алфавит и основные конструкции

Структура любого языка программирования высокого уровня состоит из следующих элементов: алфавит; данные; выражения и операции; операторы и функции.

Алфавит — это полный набор букв, цифр и символов, принятых в языке для обозначения данных и действий над ними.

Алфавит языка Visual Basic включает следующий набор символов:

- прописные (A – Z) и строчные (a – z) буквы латинского алфавита;
- цифры от 0 до 9;
- знаки арифметических операций (в порядке возрастания приоритета): +, -, *, /, |, ^;
- знаки операций отношения: =, <, >;
- знаки препинания и разделители: , . : ; ( ).

В алфавит языка входят также зарезервированные слова, которые не могут быть использованы в качестве имен переменных или процедур. Примеры зарезервированных слов: Dim, Sub, Integer и т.д. По умолчанию для выделения ключевых слов в окне редактирования кода Visual Basic используют шрифт синего цвета.

Данные — это возможные структуры языка, над которыми выполняются разрешенные действия (операции): константы, переменные и массивы. По способности к изменению все данные делятся на переменные и константы.

Переменная — это величина, которая может меняться при выполнении программы.

Константа — это величина, не меняющаяся в процессе работы. Примером константы может быть число π.

Тип, имя и значение переменной

Переменные используются для хранения и обработки данных в программах.

Переменная представлена *именем* и служит для обращения к данным определенного *типа*, конкретные *значения* которых хранятся в ячейках оперативной памяти. *Имя переменной (идентификатор)* состоит из латинских букв, начинается с буквы и не должно включать знак «.». Длина имени ≤ 255 символов. Тип переменных определяется типом данных, которые могут быть значением переменных.

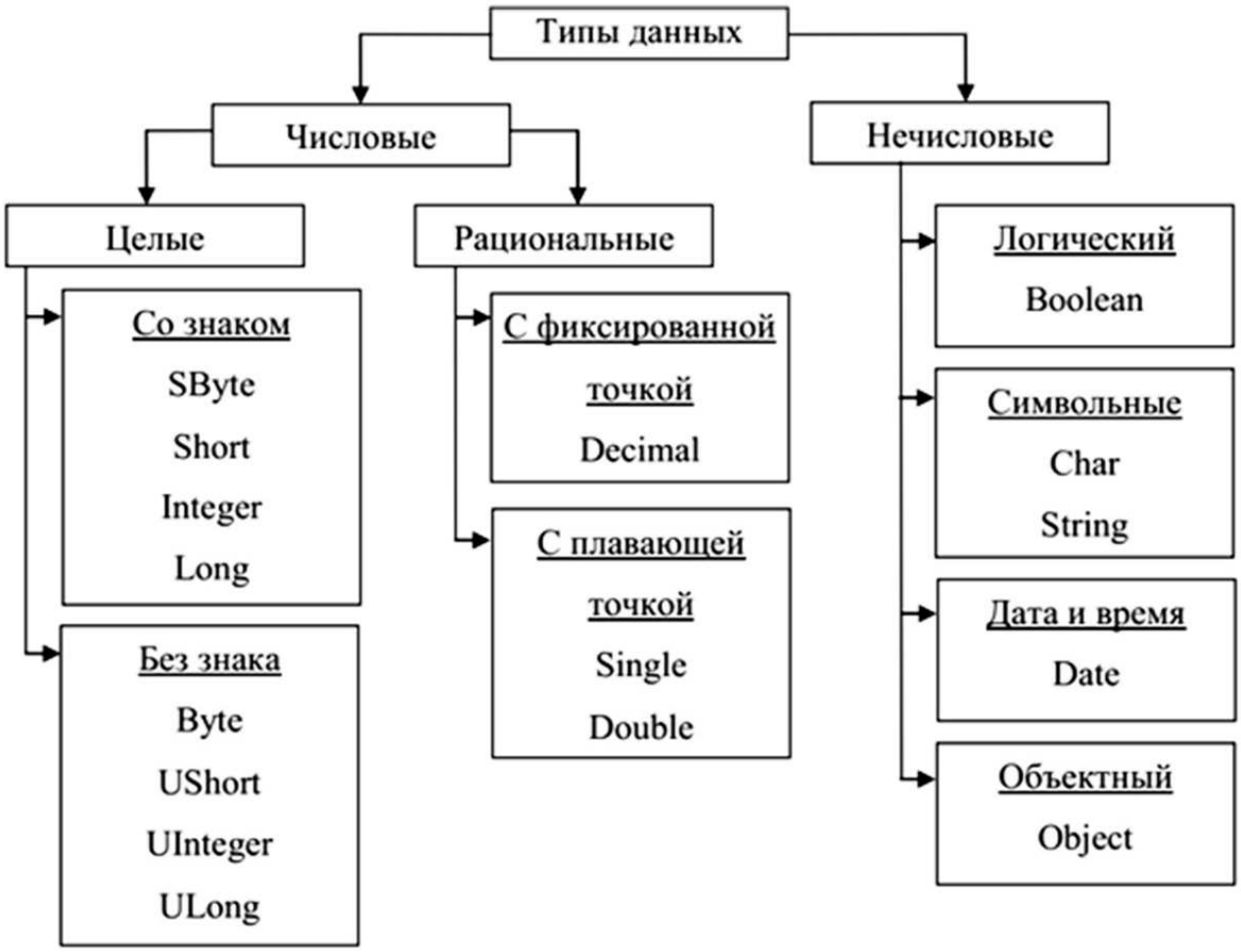


Рис. 12. Типы данных

Таблица 4

Типы данных

Тип переменной	Возможные значения	Объем памяти	Приставка к имени
Byte	целые числа от 0 до 255	1 байт	byt
Integer	целые числа [-32 768; +32 767]	2 байта	int

Long	целые числа от [-2 147 483 648; +2 147 483 647]	4 байта	lng
Single	Десятичные числа одинарной точности (7 знач. цифр) [$-1{,}4\cdot10^{-45}$; $+3{,}4\cdot10^{38}$]	4 байта	sng
Double	Десятичные числа двойной точности (15 знач. цифр) [$-5\cdot10^{-324}$; $1{,}7\cdot10^{308}$]	8 байтов	dbl
Boolean	Логические значения – True и False	2 байта	bln
String	Строка символов	1 символ - 1 байт	str
Currency	Число в денежном формате	8 байтов	cur
Date	дата 1 января 100 г. до 31 декабря 9999 года	8 байтов	dtm
Object	Ссылка на любой объект	4 байта	obj
Variant	Любые значения	>16 байтов	vnt

Объявление типа переменных

Объявление переменной означает указание имени и типа переменной перед ее использованием. Это можно сделать с помощью четырех различных операторов:

Dim *Имя Переменной* **As** *Тип Данных*
Private *Имя Переменной* **As** *Тип Данных*
Public *Имя Переменной* **As** *Тип Данных*
Static *Имя Переменной* **As** *Тип Данных*

Переменная, объявленная с помощью оператора Dim, называется локальной. Она доступна только в пределах области, содержащей оператор Dim. Как только выполнение программы выходит за границы этой области, объявленная переменная перестает существовать.

Объявление переменной с помощью оператора Public означает, что данная переменная является глобальной или, другими словами, общедоступной. Она доступна во всех областях, подпрограммах и модулях, входящих в состав проекта. Оператор Public нельзя использовать внутри подпрограмм.

Оператор Private используется для объявления переменной, доступной только в пределах того модуля, в котором она объявлена. Такая переменная является глобальной в пределах своего модуля, но по отношению ко всему

проекту она будет локальной, то есть недоступной из других его частей. Оператор Private нельзя использовать внутри подпрограмм.

Переменная, описанная как Static, является локальной, то есть доступной только в пределах той подпрограммы, где она объявлена. Но в отличие от Dim-переменной, она сохраняет свое значение и после завершения подпрограммы, хотя и недоступна из других частей проекта. Оператор Static можно использовать только внутри подпрограммы.

Если одновременно требуется объявить несколько переменных одного и того же типа, то их имена перечисляются через запятую. В конце перечисления запятую ставить не надо.

Dim a, b, c As Single

В пределах одного оператора можно описывать несколько переменных различного типа.

Dim s as String, d as Char

Таблица 5

Арифметические операции

Операция	Пояснение	Пример	Приоритет операций
+	Сложение	2+3	^
-	Вычитание	5–2	/
*	Умножение	2*3	\
/	Деление	7/2 (результат 3.5)	MOD
\	Деление нацело	7\2 (результат 3)	*
MOD	Остаток от деления	7 MOD 2 (остаток 1)	+
^	Возведение в степень	2^3 (результат 8)	-

В Visual Basic выделяют три вида выражений: арифметические, логические и строковые.

Арифметическое выражение — это последовательность чисел, констант, переменных, функций и других арифметических выражений, заключенных в круглые скобки, которые соединены между собой знаками арифметических операций.

Переменные, входящие в состав арифметического выражения должны иметь числовой тип. Функции, входящие в состав арифметического выражения должны возвращать числовое значение.

При вычислении значения арифметического выражения сначала выполняются действия в круглых скобках, затем все остальные

арифметические операции в соответствии со своими приоритетами. При равенстве приоритетов операции выполняются слева направо.

Строковые выражения включают переменные строкового типа, строки (последовательности символов, строковые функции).

Конкатенация (+ или &) — операция сложения строк или строковых переменных.

Логические выражения включают логические переменные (blnA As Bolean), логические значения (True, False), результаты операций сравнения чисел и строк, а также логических операций (AND, OR, NOT).

Таблица 6

Операции сравнения и логические операции

Операции сравнения	Логические операции
< меньше	**And** – логическое умножение
> больше	**Eqv** – эквивалентность
<= меньше или равно	**Imp** – импликация
>= больше или равно	**Not** – логическое отрицание
= равно	**Or** – логическое сложение
<> не равно	**Xor** –логическое исключающее сложение

Функции в языке Visual Basic

(Формат:Имя Функции (Список Аргументов)

I. Функции преобразования типов данных

1. Val (Строка) – превращает строку в число. Value – величина

Пример:

```
Dim V
V = Val("2457") ' Возвратит 2457.
V = Val(" 2 45 7") ' Возвратит 2457.
V = Val("24 and 57") ' Возвратит 24.
V = Val("") ' Возвратит 0.
V = Val("laja") ' Возвратит 0.
```

Применяется для преобразования строкового значения свойства Text текстовых полей в число, которое затем используется в арифметических выражениях.

```
Private Sub CmndMinus_Click()
Txt3.Text = Val(Txt1.Text) - Val(Txt2.Text)
End Sub
```

2. **Str (Число)** – преобразует десятичное число в строку

Пример:

```
Dim S
S = Str(459) ' Возвратит "459".
S = Str(-459.65) ' Возвратит "-459.65".
```

```
S = Str(459.001) ' Возвратит "459.001".
```

3. **Hex(Число)** – преобразует десятичное число в шестнадцатеричное в строковой форме.

Пример:

```
Dim H
H = Hex(5) ' Возвратит 5.
H = Hex(10) ' Возвратит A.
H = Hex(459) ' Возвратит 1CB.
```

4. **Oct(Число)** – преобразует десятичное число в восьмеричное в строковой форме.

```
Dim O
O = Oct(4) ' Возвратит 4.
O = Oct(8) ' Возвратит 10.
O = Oct(459) ' Возвратит 713.
```

5. **Asc (Строка)** – преобразует строку в числовой код первого символа.

Пример:

```
Dim N
N = Asc("A") ' Возвратит 65.
N = Asc("a") ' Возвратит 97.
N = Asc("Apple") ' Возвратит 65.
```

6. **Chr (ЧислоКод)** – возвращает символ, соответствующий определённому коду. Эта функция является обратной Asc.

Пример:

```
Dim C
C = Chr(65) ' Возвратит A.
C = Chr(97) ' Возвратит a.
C = Chr(62) ' Возвратит >.
C = Chr(37) ' Возвратит %.
```

II. Математические функции

1. **Sin(Число)** – вычисляет синус числа;
2. **Cos(Число)** – вычисляет косинус числа;
3. **Tan(Число)** – вычисляет тангенс числа;
4. **Atn (Число)** – вычисляет арктангенс числа.

Пример:

```
Dim A, C, S, D, pi
A = 1.3 ' Определяет угол в радианах
C = 1 / Sin(A) ' Вычисляет косинус
S = 1 / Cos(A) ' Вычисляет синус
C = 1 / Tan(A) ' Вычисляет котангенс
pi = 4 * Atn(1) ' Вычисляет значение числа pi.
```

5. **Sqr(Число)** – Возвращает корень числа

Пример:

```
Dim S
S = Sqr(4) 'Возвратит 2.
S = Sqr(23) 'Возвратит 4.79583152331272.
S = Sqr(0) 'Возвратит 0.
S = Sqr(-4) 'Генерирует ошибку (корень из отрицательного числа).
```

6. **Log(Число)** – вычисляет натуральный логарифм числа (по основанию e). (Возвращает тип Double) **e=2,71**

Для того, чтобы получить логарифм по основанию n нужно произвести следующее вычисление:

Logn(x) = Log(x) / Log(n)

7. **Exp (Число)** – Возвращает экспоненту числа.

Пример:

```
Form1.Caption = Exp(1) 'Отобразит на Caption число e (т.е. e в степени 1)
```

8. **Rnd[(Число)]** – Генерирует случайное число от 0 до 1.

Для генерации случайного числа X в интервале [A,B] используют формулу:

X=RND*(B-A) +A или **X=RND*(B-A+1) +A** (включает крайние знач. интервала [A,B]).

Каждый раз при запуске программы, если не переустанавливается база генератора случайных чисел, формируется одна и та же последовательность чисел.

RANDOMIZE (база) – переустанавливает базу генератора случайных чисел.

Пример:

```
Dim V
RANDOMIZE TIMER
V = Int((6 * Rnd) + 1) ' Генерирует случайное число от 1 до 6.
```

III. Строковые функции

1. **LEN(Строка)** – определяет длину строки;
2. **Left(Строка, n)** – вырезает n символов, начиная с первого символа до указанного номера;
3. **Right(Строка, n)** – вырезает n символов из строки, начиная справа;
4. **Mid(Строка, n, k)** – вырезка из строки с n-ой позиции k символов.

Пример:

```
Dim strA, strL, strR, strS As String, intN As Integer
strA = «Школа» ' ОпределяеТ строку
intN=Len(strA) ' ОпределяеТ длину строки
strL = Left(«Школа», 1) ' Возвратит «Ш»
strL = Left(strA, 3) ' Возвратит «Шко»
```

str**L** = **Lef**t(«Школа», 20) ' Возвратит «Школа»
str**R**= **Right**(strA, 1) ' Возвратит «а»
str**R** = **Right**(«Школа», 3) ' Возвратит «ола»
str**S**=**Mid**(«Школа», 2, 3) ' Возвратит «кол»

IV. Функции даты и времени

1. **Date()** – функция не имеет аргументов и возвращает текущую дату.

2. **Time()** – функция также не имеет аргументов и возвращает текущее время.

3. **DateAdd()** – возможность добавить к дате указанное количество лет, кварталов, месяцев и так далее — вплоть до секунд.

4. **DateDiff()** – возможность получить разницу между датами (в единицах от лет до секунд).

5. **DatePart()** – очень важная функция, которая возвращает указанную вами часть даты (например, только год, только месяц или только день недели).

6. **DateSerial()** – возможность создать значение даты на основе передаваемых символьных значений. То же самое делает **DateValue()**, отличия – в формате принимаемых значений. Аналогичным образом (для времени) работают TimeSerial() и TimeValue().

7. **Day()** (а также **Year(),** Month**(), Weekday(), Hour(), Minute(), Second())** – специализированные заменители функции DatePart(), которые возвращают нужную вам часть даты.

8. **MonthName()** – возвращает имя месяца словами по его номеру. Возвращаемое значение зависит от региональных настроек. Если они русские, то вернется русское название месяца.

9. **Timer()** – возвращает количество секунд, прошедших с полуночи.

Задания для самостоятельной работы

1. Записать арифметические выражения по правилам языка программирования VB:

№1. $\dfrac{b+\sqrt{b^2+4ac}}{2a}$

№2. $\dfrac{Sin\,x+Cos\,y}{Cos\,x-Sin\,y}\cdot tg\,xy$

№3. $\dfrac{1+\sin^2(x+y)}{2+\left|x-\dfrac{2x}{1+x^2y^2}\right|}$

№4. $\dfrac{x^2+7x+10}{x^2-8x+12}$

№5. $x-10\sin x+\left|x^4-x^5\right|$

№6. $\left|x^2-x^3\right|-\dfrac{7x}{x^3-15x}$

№7. $\sin\sqrt{x+1}-\sin\sqrt{x-1}$

№8. $e^x-x-2+(1+x)^x$

№9. $2(3x)-\dfrac{1}{12x^2+7x-5}$

2. Запишите алгебраические выражения, соответствующие следующим записям на языке VB:

а) (a + b) / c; б) a + b / c; в) a / b / c;
г) a / (b * c); д) (a + b) / (d + c); е) a + b / (d + c);
ё) a + b / d + c; ж) (a + b) / d + c.

3. Разработать проект, позволяющий вычислить гипотенузу и площадь прямоугольного треугольника, если известны его катеты.

4. Разработать проект «Мультисистемный калькулятор», который позволяет производить арифметические операции над целыми числами в десятичной, восьмеричной и шестнадцатеричной системах счисления.

5. Разработать проект «Коды символов», который выводит числовой код введенного символа, а также распечатывает символы по числовым кодам (выводит кодировочную таблицу символов).

6. Создать проект, позволяющий печатать текущую дату и количество дней, прошедших с начала третьего тысячелетия.

7. Разработать проект, который выводит в текстовые поля текущие дату и время.

Тема 2.3. Линейные программы

Функции ввода и вывода

1. Функция **InputBox** (Окно ввода) применяется для ввода чисел или текста. Эта функция отображает диалоговое окно ввода, содержащее поле ввода и поясняющий текст:

Переменная = InputBox (Приглашение [, Заголовок] [, НачЗначение])

где:

Переменная – строковая переменная, в которой хранится значение, возвращаемое функцией InputBox();

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

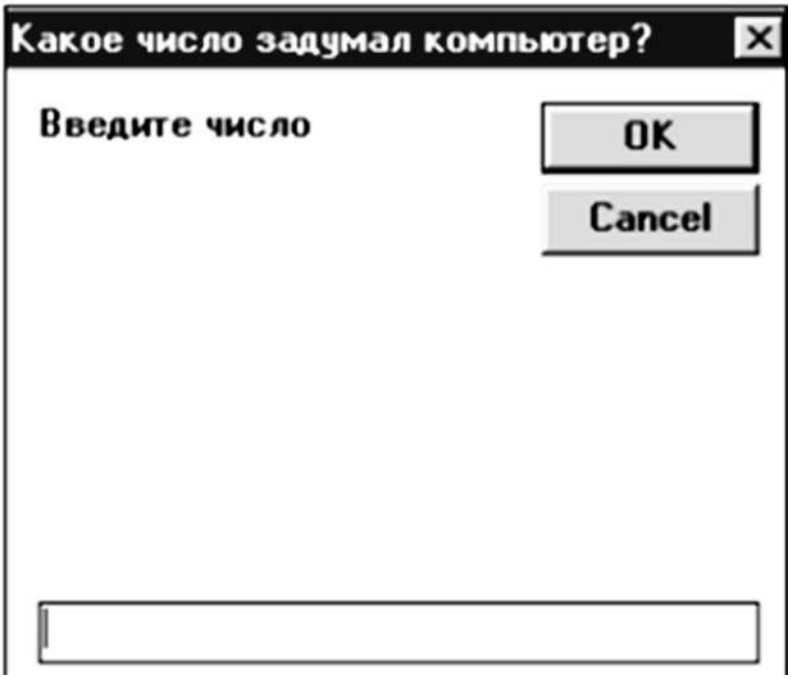
Заголовок – это надпись в строке заголовка Окна ввода.

НачЗначение – это значение, которое будет отображаться (по умолчанию) в поле ввода, пока пользователь не введет свое значение. Если этот аргумент опустить, то поле ввода отображается пустым.

Возвращаемым значением данной функции является информация, вводимая пользователем. Возвращаемое значение можно использовать в окнах сообщений, применить в вычислениях и т.д. VB автоматически приписывает этой информации тип Variant.

Например:

X = InputBox («Введите число», «Какое число задумал компьютер?»)



2. Функция **MsgBox** (Окно сообщений) служит для организации диалоговых окон, содержащих какие-либо сообщения.

После появления на экране, окно сообщений ждет, пока пользователь щелкнет на одной из кнопок, присутствующих в окне. Синтаксис этой функции следующий:

MsgBox (Текст [, Кнопки] [, Заголовок])

где: **Текст** – единственный обязательный аргумент этой функции. Значением этого аргумента является строка текста, которая появляется как сообщений в диалоговом окне. Эта строка текста должна быть заключена в кавычки. Текст может содержать до 1024 символов. Использование круглых скобок в синтаксисе MsgBox указывает на то, что в данном случае MsgBox является функцией, возвращающее какое-либо значение. Если скобки опущены, то для VB это признак того, что данное выражение значение не возвращает.

Кнопки – это аргумент, который является целым числом и может быть представлен как сумма двух слагаемых: Кнопки = Опция1 + Опция2. Если не указан аргумент Кнопки, то VB предполагает, что в диалоговом окне сообщения присутствует только кнопка ОК. Аргумент Кнопки позволяет управлять следующими параметрами окна сообщения:

- Количество кнопок в окне;
- Типы кнопок и их размещение в окне;
- Пиктограмма, отображаемая в окне.

Значение Опция1 устанавливает число и тип кнопок в Окне сообщения. Значение Опция2 определяет вид сообщения и пиктограмму, которая помещается в Окно сообщения.

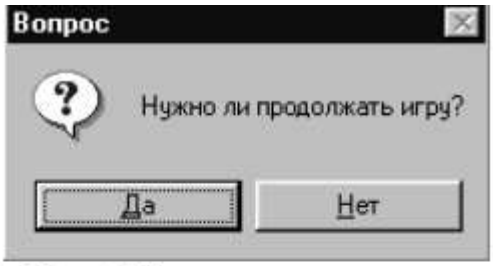
Значения Опция1	Набор кнопок
0	ОК
1	ОК, Отмена
2	Стоп, Повтор, Пропустить
3	Да, Нет, Отмена
4	Да, Нет
5	Повтор, Отмена

Значения Опция2	Вид сообщения	Пиктограмма
16	Критическое сообщение	
32	Вопрос	
48	Предупреждение	
64	Информация	

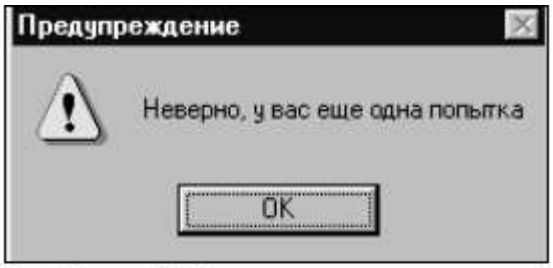
Для создания конечного значения аргумента Кнопка можно использовать только одно значение из каждой опции, сложив их значения.

Заголовок позволяет задать текст, помещаемый в строке заголовка диалогового окна сообщения.

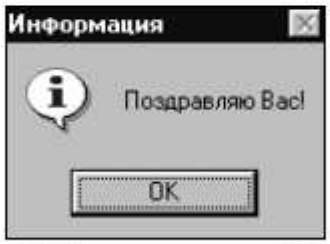
Примеры Окон сообщений представлены на следующем рисунке.



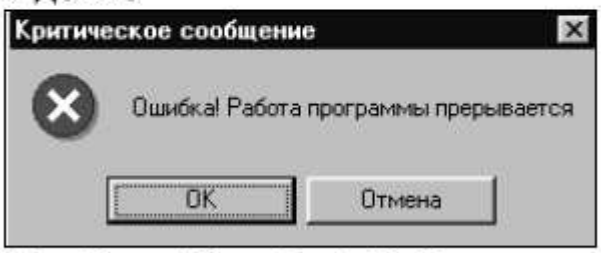
MsgBox "Нужно ли продолжать игру?", 36, "Вопрос"



MsgBox "Неверно, у вас еще одна попытка", 48, "Предупреждение"



MsgBox "Поздравляю Вас!", 64, "Информация"



MsgBox "Ошибка! Работа программы прерывается", 17, "Критическое сообщение"

Мы рассматриваем MsgBox как функцию, это подразумевает, что она должна возвращать значение. В таблице представлен список значений, возвращаемых функцией MsgBox. Возвращаемое значение зависит от того, на какой кнопке щелкнул пользователь в окне сообщения.

Значение	Нажатая кнопка
1	ОК
2	Отмена
3	Стоп
4	Повтор

Значение	Нажатая кнопка
5	Пропустить
6	Да
7	Нет

Для отображения выходных данных программы в форме может быть использована команда Print. Эта команда имеет следующий синтаксис:

Print выражение

где выражение может быть свойством, текстовым или числовым значением в процедуре, списком переменных.

В последнем случае в качестве разделителей элементов в списке можно использовать запятую или точку с запятой. При использовании точки с запятой элементы располагаются вплотную друг к другу, а при использовании запятой отделяются длиной поля табуляции. Текстовые переменные заключаются в кавычки.

Переменная может получить или изменить значение с помощью *оператора присваивания.*

Оператор присваивания является простейшим функциональным оператором. Он позволяет записать значение в переменную. Так же говорят, что оператор присваивания позволяет задать значение переменной. Оператор присваивания обозначается знаком равно (=). Слева от оператора указывается имя переменной, а справа – значение.

[Let] Имя переменной = Значение переменной

Значение переменной может быть задано явно, в виде выражения или с помощью другой переменной. При явном задании значения оператор присваивания имеет следующий вид:

```
a = 5.78
```

Если значение задается выражением, то справа от оператора присваивания записывается выражение, тип которого должен соответствовать типу переменной, стоящей слева от оператора присваивания. При выполнении такого оператора сначала вычисляется значение выражения, а потом вычисленное значение записывается в переменную:

```
b = 2 + a / 3
```

Если значение переменной задается с помощью другой переменной, то оператор присваивания выглядит так:

```
c = a
```

При этом происходит копирование значения переменной a в переменную c. И никогда наоборот. В таких случаях говорят, что оператор присваивания работает справа налево.

Это означает, что значение переписывается из переменной, стоящей справа от знака равенства, в переменную стоящую слева от знака равенства.

Переменные, значения которых не меняются в процессе выполнения программы, называются ***константами.***

Значение константы определяется один раз. Оно остается неизменным в течение всего времени работы программы.

Объявление константы похоже на объявление переменной. Константы могут быть глобальными и локальными. Глобальные константы доступны из всех частей модуля, в котором они описаны. Локальные константы доступны только в той подпрограмме, где они были объявлены. Для описания константы используется следующая конструкция:

Const Имя константы As Тип данных = Значение

Например:

```
Const n As Integer= 20
Const FileName As String = "D:\text.txt"
```

Программа на VB – это последовательность операторов. Для того чтобы сделать программу легко читаемой, используют комментарии (пояснения к тексту программы). Существует два способа ввода комментариев: применение апострофа ('), который можно поставить в любом месте строки, и зарезервированное слово Rem вместо апострофа.

Например:

1) С помощью оператора REM:

```
REM произвольный_текст
```

2) С помощью апострофа «’»:

```
’ произвольный_текст
```

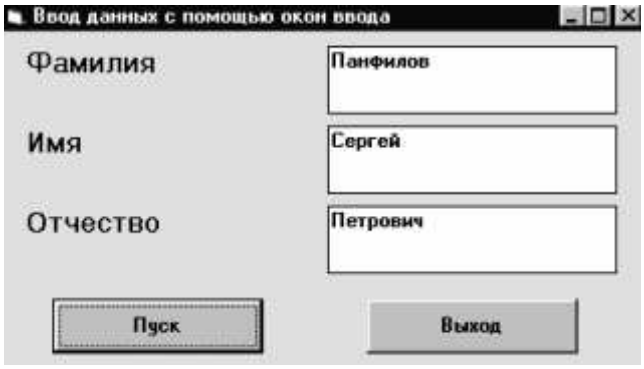
Иногда оператор получается такой длинный, что не умещается на экране. Перенос длинного оператора на другую строку осуществляется комбинацией пробела и подчеркивания. Например:

```
Результат.Text = Val(Число1.Text) _
+Val(Число2.Text)
```

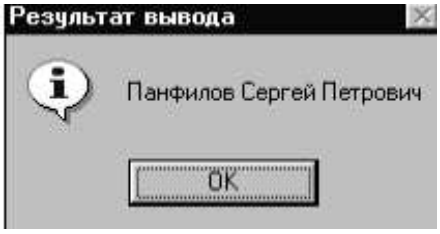
Задания для самостоятельной работы

1. Пусть экранная форма вашего приложения содержит три текстовых поля и командную кнопку «Пуск». После нажатия на эту кнопку последовательно появляются Окна для ввода вашей фамилии, имени и отчества. После ввода данных все три текстовых поля пользовательской формы должны быть заполнены.

После ввода данных все три текстовых поля будут заполнены. Экранная форма примет вид:



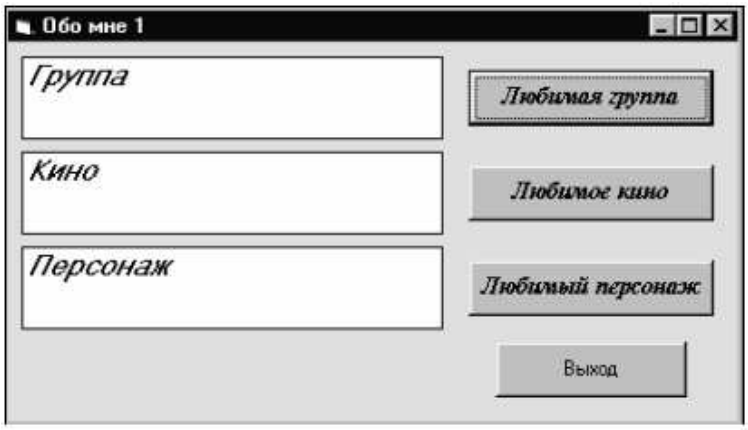
2. Дополните программу таким образом, чтобы введенная информация отображалась как в экранной форме, так и в окне вывода.



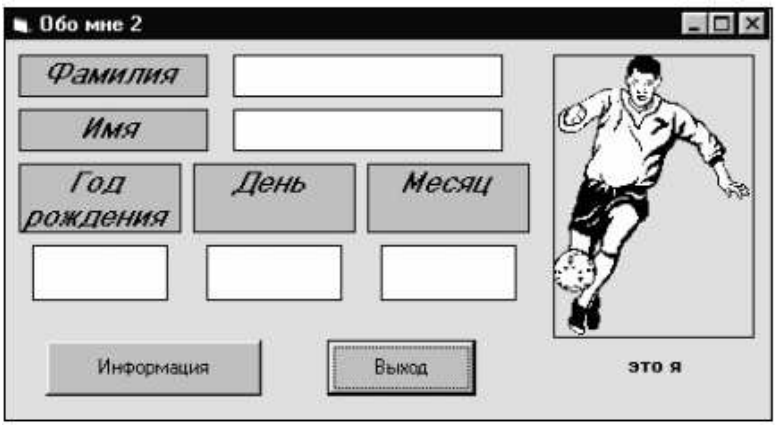
3. Разработать приложение «Обо мне» внешний вид диалогового окна показан на рисунке. В форме должны выводиться ваши любимые группа, кино

и персонаж по нажатию на соответствующие кнопки в форме. По нажатию на кнопку «Выход» форма скрывается с экрана.

4. Измените, диалоговое окно таким образом, чтобы каждый ответ выводился в отдельном окне. Один из вариантов представлен на рисунке.



5. Разработайте приложение «Обо мне 2», внешний вид диалогового окна показан на рисунке. В форме должны выводиться ваши фамилия, имя и дата вашего рождения, по нажатию на кнопку «Информация», для которой сделайте всплывающую подсказку. По нажатию на кнопку «Выход» форма закрывается. Для вывода рисунка в форме используйте элемент PictureBox.



6. Присвоить переменной X значение 20 и вывести
X = _____ квадрат X = _____
(вместо _____ надо вывести соответствующее значение).

7. Присвоить трем переменным целые нечетные числа, вычислить их сумму и вывести:

Вот некоторые нечетные числа:

_____ _____

А вот их сумма: _____

8. Присвоить переменным A, B, C любые положительные значения. Вычислить значение переменной $D = \frac{(3{,}5 \cdot A + B) \cdot 2}{4 \cdot C + 5 \cdot B}$ и вывести

При A = ____; B = _____; C = _____

D = _____

9. Присвоить переменной A – ваше имя, B – вашу фамилию, X – год рождения. Вывести следующую информацию:

Фамилия: _____

Имя: _____

Год рождения: _____

Мне _____ лет.

10. Присвоить переменным X и Y два числа. Вывести:

Для чисел _____ и _____

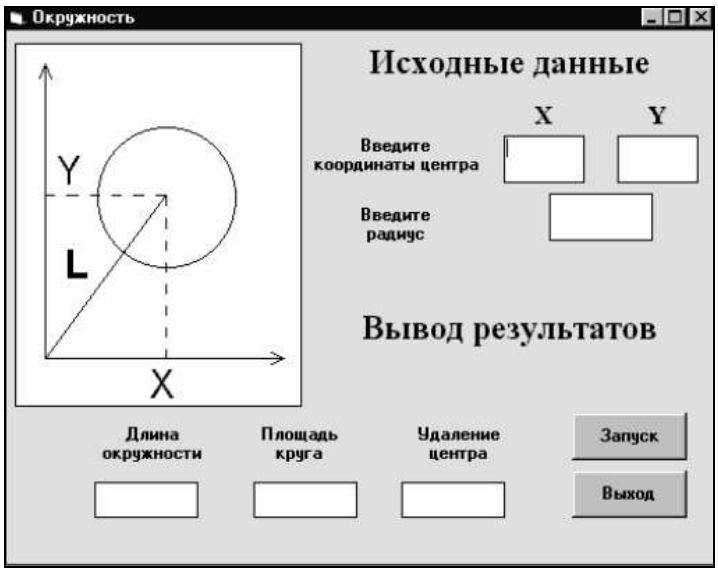
Сумма = _____

Произведение = _____

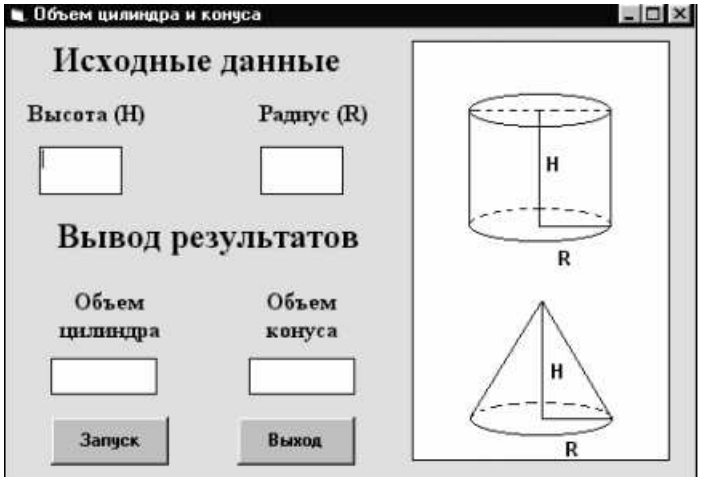
Разность = _____

Частное = _____

11. Определите длину окружности C и площадь S круга, ограниченного этой окружностью, если радиус равен R. Определите удаление L центра окружности от начала координат O. Координаты центра окружности равны X и Y. Разработайте диалоговое окно для ввода исходных данных и вывода результатов. В диалоговом окне расположите рисунок задания. Рисунок подготовьте в графическом редакторе MS Paint и сохраните в своей папке. При подготовке рисунка установите следующие размеры: высота – 250 точек, ширина – 200 точек.



12. Разработайте приложение вычисления объемов цилиндра и конуса, которые имеют одинаковую высоту H и одинаковый радиус основания R. Разработайте диалоговое окно для ввода исходных данных и вывода результатов. В диалоговом окне расположите рисунок задания. Объем цилиндра вычисляется по формуле V = πR2H, а объем конуса – по формуле V = 1/3πR2H, где π= 3,14.



13. Подошло время провести ремонт вашей комнаты – оклеить стены новыми обоями. Необходимо вычислить площадь стен комнаты и посчитать нужное количество необходимых для ремонта материалов – рулонов обоев. Разработайте приложение для ввода исходных данных и вывода результатов. В диалоговом окне расположите рисунок задания. Напишите программу для кнопки «Выход». По нажатию на эту кнопку форма должна скрываться с экрана.

14. Создайте приложение для нахождения суммы цифр заданного трехзначного целого числа k.

15. Создайте приложение для нахождения произведения цифр заданного четырехзначного числа n.

Тема 2.4. Разветвляющиеся программы. Условные конструкции

Алгоритм, в котором последовательность выполнения некоторых предписаний зависит от выполнения проверяемых исполнителем условий, называется нелинейным. Ветвление представляет собой выбор пути решения задачи в соответствии с выполнением или невыполнением некоторого условия выбора.

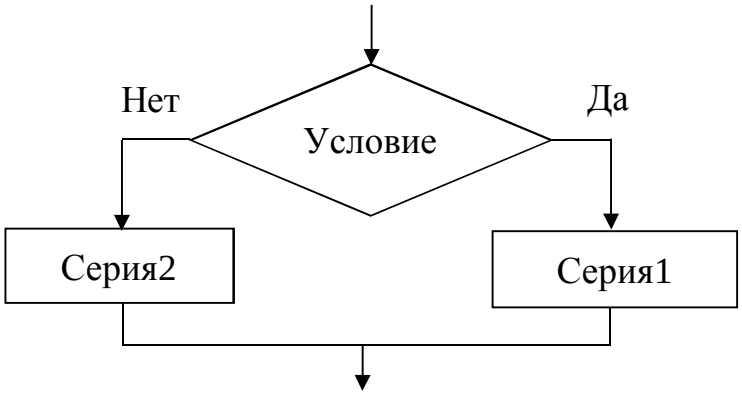


Рис. 13. Блок-схема условной конструкции

Условие – логическое выражение.

Серия1, Серия2 – набор команд.

Логическое (условное) выражение может быть представлено в четырех видах: логическая константа, логическая перемен ная, простое условие или сложное условие. Любое логическое выражение может иметь только одно из двух значений: Истина (True) или Ложь (False).

Простое условие – это два выражения, между которыми стоит знак операции сравнения. В роли выражений могут выступать числа, числовые переменные, математические функции, арифметические выражения, строки, строковые переменные, строковые функции и строковые выражения. Оба выражения, участвующие в сравнении, обязательно должны принадлежать к одному и тому же типу. Если простое условие выполняется, оно имеет значение Истина (True). В противном случае оно имеет значение Ложь (False).

Сложное условие – это последовательность простых условий, логических переменных и логических констант, которые соединены между собой знаками логических операций. Традиционно каждую составную часть сложного условия берут в круглые скобки.

Условный оператор в Visual Basic предназначен для организации ветвлений. Существует две формы синтаксиса условного оператора: однострочный оператор и многострочный оператор.

Однострочный оператор всегда записывается в одну строку и используется в тех случаях, когда ветви алгоритма содержат небольшое количество действий, чаще всего одно. Однострочный условный оператор имеет следующий синтаксис.

If *Условное Выражение* **Then** *Оператор1* **Else** *Оператор2*

Многострочный условный оператор записывается в несколько строк. Причем распределение ключевых слов по строкам является обязательным и не может быть изменено. Рассмотрим синтаксис многострочного условного оператора.

If *Условное Выражение* **Then**
    Группа Операторов1
Else
    Группа Операторов2
End If

В отличие от однострочного условного оператора, ветви многострочного оператора могут содержать не одно, а несколько действий. Поэтому данная форма условного оператора применяется гораздо чаще однострочной формы.

Оператор выбора Select Case

Оператор выбора применяется для программирования выбора одной из нескольких взаимоисключающих возможностей (альтернатив), при этом условием выбора ветви алгоритма является значение некоторого выражения.

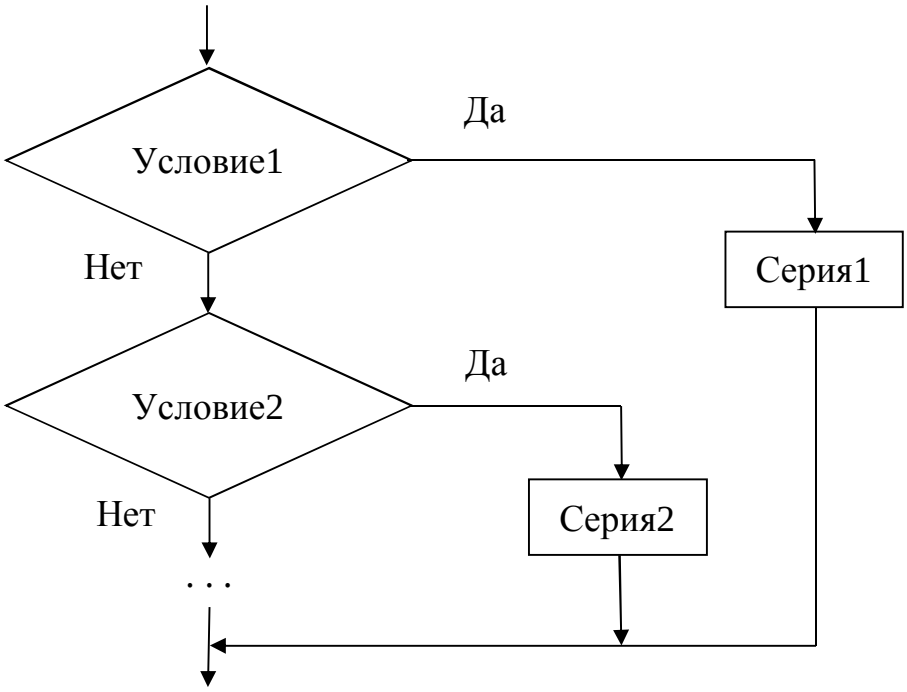


Рис. 14. Блок-схема оператора выбора

Условие1, **Условие2** – логические выражения
Серия1, Серия2 – наборы команд
Оператор выбора имеет следующий синтаксис.
Select Case *Выражение*
    Case *Список Значений1*
        Группа операторов1
    Case *Список Значений2*
        Группа операторов 2

. . .
Case *Список ЗначенийN*
Группа операторов N
Case Else
Группа операторов Else
End Select

Список значений — это последовательность выражений, разделенных запятым. Выражение может быть представлено в виде константы, числа, строки, переменной, арифметического, логического или строкового выражения. Также Список Значений может включать выражения специального вида: диапазон и луч.

Выражение – диапазон позволяет задать диапазон возможных значений. Оно имеет следующий синтаксис.

Выражение **To** *Выражение*

Например, диапазон 1 To 5определяет отрезок от 1 до 5, включая границы.

Выражение-луч используется, когда надо определить полуинтервал возможных значений. Оно имеет следующий синтаксис.

Is Знак операции сравнения **Выражение**

Например, луч Is > 10 задает множество чисел, превышающих 10.

Принцип работы Select Case тот же, что и многострочного оператора. В процессе исполнения программа сравнивает значение переменной, стоящей после Select Case, по очереди со всеми значениями, стоящими после Case. Наткнувшись на совпадающее значение, она выполнит операторы, стоящие в этом варианте. Если совпадающего значения не нашлось, то выполняются операторы, стоящие после Else.

Например:

```
…
Select Case Otmetka
        Case 1, 2
                Print "Кошмар!"
        Case 3
                Print "Слабовато!"
        Case 4
                Print "Неплохо!"
        Case 5
                Print "Молодец!"
        Case Else
                Print "Таких оценок не бывает!"
End Select
```

Оператор безусловного перехода GoTo

Операторы передачи управления применяются в программе для реализации безусловных алгоритмических конструкций. Они выполняют

переход с одного участка программы на другой без проверки какого-либо условия. Оператор безусловного перехода имеет следующий вид:

GoTo Метка

GoTo – ключевое слово Visual Basic.

Метка – это последовательность символов, заканчивающаяся двоеточием. Метка может начинаться как с буквы, так и с цифры. Она может содержать символы латинского и русского алфавитов, цифры и знак подчеркивания. В метке нельзя использовать скобки, пробелы, знаки пунктуации и арифметических операций. Традиционно рекомендуется составлять метки, пользуясь правилом имен. Метка всегда ставится на отдельной строке перед каким-либо оператором программного кода. Помеченный оператор будет выполняться сразу после оператора GoTo. Такой способ передачи управления называется безусловным переходом, так как он выполняется без проверки каких-либо условий. Передавать управление таким способом можно как вперед, так и назад по тексту программы.

Например:

```
m1: Print "Это ";
Print "тело ";
Print "цикла";
Print "   ";
GoTo m1
```

Оператор GoTo и метку можно ставить в любом месте процедуры, заставляя, таким образом, заново повторять выполненные операторы, или, наоборот, пропускать выполнение каких-либо действий.

Задания для самостоятельной работы

1. Используя встроенные диалоговые окна (InputBox и MsgBox) напишите программу, реализующую диалог компьютера с вами:

Компьютер: «Как Вас зовут?»

Пользователь: Name (вводит свое имя)

Компьютер: «Привет, Name» (введенное имя) «Сколько Вам лет?»

Пользователь: ГГ (вводит число лет)

Если ГГ (число лет) < 20, то компьютер выдает сообщение «Уже немало, Name», в противном случае компьютер сообщает: «Вы прекрасно выглядите для своих ГГ лет».

2. Разработать проект «Проверка знаний», который позволит контролировать знания. Алгоритм контроля должен последовательно реализовывать следующие операции: задать (напечатать) вопрос; запросить ответ и запомнить введенное с клавиатуры значение; полученный ответ сравнить с правильным и, в зависимости от выполнения или невыполнения этого условия, реализовать различные действия.

3. Найти наименьшее из трех чисел.

4. Заданы три числа x, y, z. Если $x < 0$, то P задать как максимальное из y, z; если $x \geq 0$, то P задать как минимальное из y, z.

5. Создайте приложение для нахождения корней линейного уравнения ax = b.

6. Создайте приложение для нахождения корней квадратного уравнения ax2 + bx + c = 0.

7. Определить, равна ли сумма двух первых цифр заданного четырехзначного числа сумме двух его последних цифр.

8. Определить, равен ли квадрат заданного числа кубу суммы цифр этого числа.

9. Заданы три числа a, b, c. Программа должна определить, можно ли построить треугольник из сторон длиной a, b, c.

10. Заданы три числа a, b, c, предположительно являющиеся длинами сторон треугольника. Определить, будет ли полученный треугольник равносторонним.

11. В школу танцев принимаются юноши и девушки, имеющие рост не ниже 168 см. и не выше 178 см. Их вес должен соотноситься с ростом по формуле: значение веса меньше чем значение роста минус 115. Определите, будет ли поступающий принят в школу.

12. Во время летних каникул вы устроились работать на предприятие быстрого питания. По договору вы должны работать 5 дней в неделю. Количество рабочих часов в день не фиксировано, а оговаривается с администрацией предприятия. Разработайте форму и напишите программу, которая позволяет вводить для пяти дней недели количество часов, отработанных в эти дни (для каждого рабочего дня количество часов вводится с помощью окна ввода). Программа должна выводить в окно формы сообщение с суммарным количеством рабочих часов за неделю. Программа должна рассчитать недельную зарплату при условии почасовой оплаты. Предположим, что минимальная оплата рабочего часа составляет 6 €. В форме предусмотрите окно для ввода оплаты рабочего часа и окно для вывода вашей недельной зарплаты. В программе предусмотрите пересчет вашей недельной зарплаты из евро в рубли, с учетом текущего курса евро. Для этого в форме подготовьте окна для ввода курса евро и вывода пересчитанной зарплаты.

13. Разработайте форму, через которую осуществляется ввод: названия покупаемого товара; количество покупаемого товара; цена товара за единицу. Рассчитайте и выведите в этой же форме стоимость товара без скидки. Предположим, в магазине действует 5% скидка на товар, стоимость которого превышает 1000 рублей. В форме должен быть предусмотрен вывод скидки в рублях и вывод стоимости покупки с учетом скидки. В форме подготовьте кнопку для очистки всех полей с целью ввода новой информации.

14. Составить алгоритм и программу начисления зарплаты согласно следующему правилу: если стаж работы сотрудника менее 5 лет; то зарплата 130 у.е., при стаже работы от 5 до 15 лет – 180 у.е., при стаже свыше 15 лет зарплата повышается с каждым годом на 10 у.е.

15. Составить программу ввода величины времени суток t ($0 \le t \le 24$) и выдачи текста:

«Вы уже проснулись?» – если t < 10;
«Не пора ли обедать?» – если t = 12;
«Еще не вечер!» – если t >= 18;
«Как работается?» – в остальных случаях.

16. Группу детей, приехавшую в пионерский лагерь, распределяют по отрядам по принципу: с 6 до 7 лет – 5 отряд, с 7 до 9 лет – 4 отряд, с 9 до 11 лет – 3 отряд, с 11 до 13 лет – 2 отряд, с 13 до 15 лет (включительно) – 1 отряд. Составьте программу, которая позволила бы каждому приезжающему самому определить свой отряд. В лагере имеется персональная ЭВМ.

17. Составить программу, позволяющую получить словесное описание школьных отметок (1 – плохо, 2 – неудовлетворительно, 3 – удовлетворительно, 4 – хорошо, 5 – отлично).

18. Написать программу, которая бы по введенному номеру единицы измерения (1 – дециметр, 2 – километр, 3 – метр, 4 – миллиметр, 5 – сантиметр) и длине отрезка L выдавала бы соответствующее значение длины отрезка в метрах.

19. Дано натуральное число N. Если оно делится на 4, вывести на экран ответ N=4k (где k – соответствующее частное); если остаток от деления на 4 равен N = 4k+1; если остаток от деления на 4 равен 2, N = 4k+2; если остаток от деления на 4 равен 3, N = 4k+3. Например, 12 = 43, 22 = 45+2.

20. Написать программу, которая бы по введенному номеру времени года (1 – зима, 2 – весна, 3 – лето, 4 – осень) выдавала соответствующие этому времени года месяцы, количество дней в каждом из месяцев.

21. Для целого числа К от 1 до 32000 напечатать «У меня К рублей», учитывая при этом, что при некоторых значениях К слово «рублей» надо заменить на слово «рубль» или «рубля». Например, 11 рублей, 22 рубля, 51 рубль.

22. Составьте программу-меню, которая при выборе фамилии поэта выводит текст его стихов.

23. «Исторический тренажер». Программа предлагает историческое событие и меню с выбором дат его свершения под номерами. Проверяется правильность ответов и выставляется оценка.

Тема 2.5. Циклические программы. Циклические конструкции

Повторение – это многократное выполнение одного или нескольких действий алгоритма. Повторение – это еще одно проявление нелинейности алгоритма. Оно может быть реализовано явно (с помощью операторов цикла) или неявно (с помощью оператора безусловного перехода).

Цикл – это оператор или группа операторов, которые программа выполняет многократно.

Циклы делятся на две категории: циклы с известным числом повторений (циклы со счетчиком) и циклы с неизвестным числом повторений (циклы с условием).

В первом случае для организации цикла используется специальная переменная (счетчик), значение которой меняется в заданном диапазоне с некоторым шагом. Во втором случае счетчика нет. Цикл продолжается пока выполняется некоторое условие. Его еще называют условием цикла.

Классификация циклов

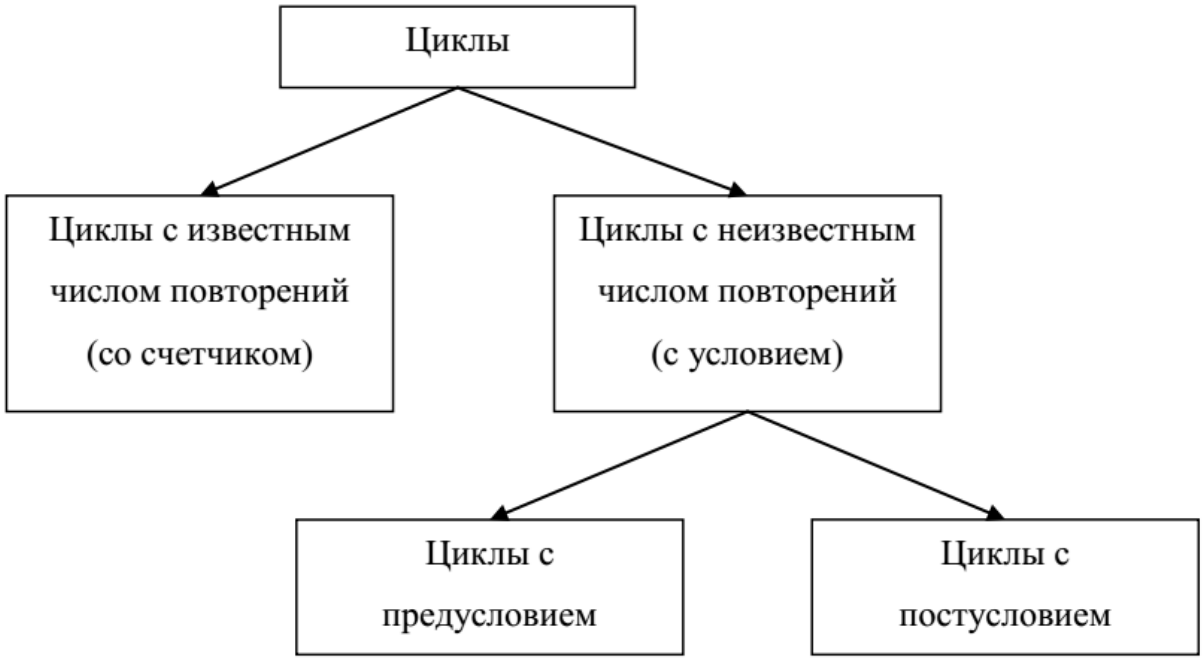


Рис. 15. Классификация циклов

1. Цикл со счетчиком

Циклы со счетчиком (с известным числом повторений) в Visual Basic еще называют циклами For или циклами For ... Next. Так они называются потому, что начало и конец этих циклов определяются операторами For и Next. Синтаксис цикла For … Next таков:

For *Счетчик*=*НачЗначение* **To** *КонечноеЗначение* **[Step** *Шаг***]**
Операторы
[Exit For]
Next [Счетчик]

где:

For – ключевое слово Visual Basic, обозначающее начало цикла;

Счетчик – переменная, определенная в качестве счетчика цикла;

НачальноеЗначение – число, задающее начальное значение счетчика;

To – ключевое слово Visual Basic, разделяющее НачЗначение и КонечноеЗначение;

КонечноеЗначение – число, задающее значение счетчика, при котором цикл завершается;

Step – ключевое слово Visual Basic, используемое для задания шага цикла, необязательный аргумент;

Шаг – число, задающее шаг цикла, т.е. значение, на которое увеличивается (или уменьшается) значение счетчика на каждом шаге. Это число может быть отрицательным;

Exit For – оператор досрочного выхода из цикла (необязательный);

Next – ключевое слово Visual Basic, обозначающее конец цикла.

В начале цикла For ... Next определяется переменная счетчик, а также начальное и конечное значения этой переменной. В самом начале выполнения цикла переменная-счетчик устанавливается в начальное значение. Каждый раз, когда программа, пройдя через цикл, возвращается к его началу, значение счетчика увеличивается. Если используется ключевое слово Step, то переменная-счетчик изменяется в соответствии с числом, указанным после этого слова.

Пример:

```
For I = 0 To 10 Step 2 'Значение I будет увеличиваться на 2
```

Если ключевое слово Step отсутствует, то значение шага равно единице.

Каждый раз, когда значение переменной-счетчика изменяется, оно сравнивается с заданным конечным значением счетчика.

Если значение счетчика превышает конечное значение, программа сразу выходит из цикла и переходит к той строке кода, которая следует за циклом.

Цикл For ... Next может быть прерван досрочно, например, при достижении какого-либо условия. Для этого в нужном месте цикла нужно поместить оператор Exit For.

Например: подсчитаем сумму четных чисел в промежутке от 1 до 50 с помощью оператора FOR … NEXT.

```
S = 0 ‘ начальное значение суммы
FOR i = 2 TO 50 STEP 2
S = S + i
NEXT
PRINT “Сумма четных чисел S =”; S
```

После завершения цикла управление передается команде, следующей за NEXT.

С помощью FOR…NEXT можно организовать вложенные циклы – каждый со своим FOR, NEXT и счетчиком:

```
FOR i = …
  FOR j =…
    …
  NEXT j
NEXT i
```

2. Цикл с условием

Цикл с условием используется в тех случаях, когда число повторений цикла заранее неизвестно. Главной особенностью циклов с условием является условие, которое может быть любым выражением, принимающим значение

True или False. Например, при обработке совокупности чисел, ввод которой прекращается при появлении первого нуля.

Цикл с условием – это многострочный оператор Visual Basic, первая строка которого начинается со слова Do, а последняя строка начинается со слова Loop.

Выделяют две разновидности цикла с условием: цикл с предусловием и цикл с постусловием.

В Visual Basic есть два основных цикла Do ... Loop - с условием, вводимым ключевым словом While, и с условием, вводимым ключевым словом Until.

Оба они могут быть с предусловием или с постусловием.

Циклы Do While | Until имеют следующий синтаксис:

Цикл с предусловием	*Цикл с постусловием*

Do While|Until Выражение
  Операторы
Loop
	Do
	  Операторы
	Loop While|Until Выражение

где:

Do – ключевое слово;

While | Until – ключевые слова, указывающие тип цикла;

Выражение – выражение условия, принимающее значение True или False;

Loop – ключевое слово, указывающее на окончание цикла.

Цикл **Do While** выполняется до тех пор, пока выражение условия имеет значение True.

Цикл **Do Until** выполняется до тех пор, пока выражение условия имеет значение False.

Например: подсчитаем сумму четных чисел в промежутке от 1 до 50 с помощью цикла DO … LOOP.

```
1) S = 0 : i = 2                    2) S = 0 : i = 2
   DO WHILE  i < = 50                  DO UNTIL  i > 50
      S = S + i                           S = S + i
      i = i + 2                           i = i + 2
   LOOP                                LOOP
   PRINT  "S ="; S                     PRINT  "S ="; S
```

```
3) S = 0 : i = 2                    4) S = 0 : i = 2
   DO                                  DO
      S = S + i                           S = S + i
      i = i + 2                           i = i + 2
   LOOP WHILE  i < = 50                LOOP UNTIL  i > 50
   PRINT  "S ="; S                     PRINT  "S ="; S
```

Иногда бывает необходимо прервать цикл Do ... Loop, если выполняется какое-либо дополнительное условие. Это может быть сделано с помощью оператора Exit Do.

Меняя Условные выражения, каждый вид цикла можно заменить на любой другой без потери работоспособности программы. Если Условие цикла

сформулировано с ошибкой, то программа может попасть в бесконечный цикл. Бесконечный цикл – это цикл, в котором количество повторов ничем не ограничено. В таких случаях говорят, что программа «зациклилась» или «зависла».

Для создания бесконечного цикла условное выражение должно быть таким, которое никогда не выполнится или выполняется всегда. Выйти из такого бесконечного цикла и прервать работу программы можно, нажав комбинацию клавиш [Ctrl + Break].

Задания для самостоятельной работы

1. Напечатать таблицу соответствия между весом в фунтах и весом в килограммах для значений 1, 2, …, 10 фунтов (1 фунт = 453 г.)
2. Напечатать таблицу перевода 1, 2, …, 20 долларов США в рубли по текущему курсу (значение курса вводится с клавиатуры).
3. Одноклеточная амеба каждые 3 часа делится на 2 клетки. Определить, сколько клеток будет через 3, 6, 9, …, 24 часа, если первоначально была одна амеба.
4. Вычислить сумму:
   a) первых пятидесяти натуральных чисел;
   b) всех целых чисел от –10 до B (значение B вводится с клавиатуры).
5. Вычислить произведение:
   a) всех целых чисел от A до 20 (значение A вводится с клавиатуры, $1 \le A \le 20$);
   b) нечетных чисел от 1 до 21 включительно.
6. Вычислить среднее арифметическое:
   a) всех целых чисел от 1 до 1000;
   b) всех целых чисел от A до B (значения A и B вводятся с клавиатуры, $A \le 200$).
7. Ввести n чисел. Определить, сколько среди них положительных и отрицательных.
8. Ввести n положительных чисел. Найти a) наибольшее из них; b) наименьшее из них; c) наибольшее и наименьшее из них.
9. Ввести n положительных чисел. Найти a) Номер наибольшего из них. Если таких чисел несколько, то должен быть найден номер последнего из них. b) Номер наименьшего из них. Если таких чисел несколько, то должен быть найден номер первого из них. c) Номер наибольшего и номер наименьшего из них. Если таких чисел несколько, то должны быть найдены номера последних из них.
10. Ввести n чисел. Определить, сколько среди них превосходит первое число.
11. Начав тренировки, спортсмен в 1-ый день пробежал 10 км. Каждый следующий день он увеличивал дневную норму на 10 % от нормы предыдущего дня. a) Определить пробег лыжника за второй, третий, …, десятый день

тренировок. b) Какой суммарный путь пробежал спортсмен за первые 7 дней? c) Через сколько дней спортсмен будет пробегать в день больше 20 км.? d) Через сколько дней спортсмен пробежит суммарный путь в 100 км.?

12. Известны данные о температуре воздуха в течение месяца. Определить, сколько было дней за месяц с самой низкой температурой.

13. В кинотеатре 30 рядов кресел. В первом ряду 20 кресел, в каждом последующем на 2 кресла больше, чем в предыдущем. Сколько мест в зрительном зале?

14. Последовательно вводятся №целых чисел. Найти количество отрицательных среди них.

15. Составьте программу вывода на экран таблицы умножения.

Тема 2.6. Обработка текстовой информации

Строка – это некоторый набор символов, в том числе и пустой. Для обозначения строки используются кавычки ("). *Пустая строка* обозначается парой кавычек, между которыми нет ни одного символа, в том числе и пробела(s =""").

Строки описываются с помощью типа **String**. Каждая строка может содержать до двух миллиардов символов в формате Unicode. Объем памяти, занимаемой строковой переменной, зависит от количества символов в этой строке. Чем больше символов, тем больше памяти требуется для хранения этой строки.

Число символов в строке называется **длиной строки**. Длина пустой строки равна нулю. Все символы в строке последовательно пронумерованы. Нумерация идет слева направо и начинается с единицы. Номер символа в строке также называют **позицией символа**.

Подстрока – это любой фрагмент строки, состоящий хотя бы из одного символа исходной строки. Например, в строке «электростанция» содержится подстрока «рост». *Левой подстрокой* называется такая подстрока, которая начинается с первого символа исходной строки. В нашем случае левой подстрокой будет «элек». *Правая подстрока* – это подстрока, которая заканчивается последним символом исходной строки. В нашем примере это будет подстрока «станция».

Конкатенацией двух строк s1и s2называется строка s, для которой s1 является левой подстрокой, s2 – правой подстрокой, а длина строки s равна сумме длин строк s1и s2.Часто конкатенацию называют сложением или склейкой строк. В Visual Basic она обозначается знаком плюс.

```
s = s1 + s2
```

Также допускается следующая запись конкатенации.

```
s= s1 & s2
```

Основные функции обработки строк

- **Len** (Строка) – возвращает количество символов в строке, то есть ее длину. Например, Len ("Окно") вернет значение 4.

- **Left** (Строка, Длина) – выделяет левую подстроку указанной длины из заданной строки. Например, Left ("Пароход", 3) вернет строку «Пар».
- **Right** (Строка, Длина) – выделяет правую подстроку указанной длины из заданной строки. Например, Right ("Пароход", 3) вернет строку «ход».
- **Mid** (Строка, Позиция, Длина) – выделяет подстроку заданной длины из исходной строки, начиная с указанной позиции. Например, Mid ("Пароход", 2, 4) вернет строку «арох».

Параметр длина может отсутствовать. В этом случае функция Mid возвращает правую подстроку, которая начинается с заданной позиции. Например, Mid ("Пароход", 4) вернет строку «оход».

- **LTrim** (Строка) – удаляет все пробелы, стоящие в начале строки, до первого символа, который не является пробелом.
- **RTrim** (Строка) – удаляет все пробелы, стоящие в конце строки, до ближайшего символа, который не является пробелом.
- **Trim** (Строка) – удаляет все пробелы, стоящие в начале и в конце строки. Пробелы, стоящие в середине строки, остаются без изменений.
- **Space** (Количество) – формирует строку, состоящую из заданного количества пробелов.
- **StrDup** (Количество, Символ) – формирует строку, состоящую из заданного количества повторений указанного символа.
- **Val** (Строка) – функция пытается преобразовать указанную строку в число. Преобразование прерывается при первой же ошибке. Если исходная строка начинается не с цифры, а с символа, то функция возвращает значение 0.
- **Str** (Число) – преобразует число в строку. Перед положительными числами функция ставит один пробел.
- **Asc** (Строка) – возвращает ASCII код первого символа строки.
- **Chr** (Код) – возвращает символ, соответствующий заданному ASCII коду.

Задания для самостоятельной работы

1. С помощью операций вырезания и склеивания составить из частей слова «электроника» слова «контролер», «лирика».
2. Определить, какое из 2-х исходных слов длиннее и на сколько. Полученные результаты вывести на форму.
3. Заменить в тексте букву «а» на «о» и посчитать количество замен.
4. С помощью операций вырезания и склеивания составить из частей слова «электроника» слова «кролик», «лектор».
5. Написать программу, сравнивающую количество букв «а» с количеством букв «о» в заданном предложении и выводящую соответствующее сообщение на экран.
6. В произвольной строке символов заменить прописные буквы на строчные, строчные на прописные.

7. Написать программу, которая подсчитывает, сколько русских гласных букв содержится во введенной фразе.

8. Поменять местами символы, стоящие на четных и нечетных местах в произвольной строке.

9. Удалить из строки все сдвоенные буквы А.

10. В произвольной строке переставить символы в обратном порядке.

11. Каждые пять символов во введенном тексте отделите знаком «*».

12. Напечатать заданный текст, преобразовав его следующим образом:
    а) заменить все восклицательные знаки точками;
    б) заменить каждую точку многоточием (т.е. тремя точками);
    в) каждую из групп стоящих рядом точек одной точкой;
    г) каждую из групп стоящих рядом точек многоточием (т.е. тремя точками).

13. Дан текст. Найти в нем
    а) номер первой запятой;
    б) номер последней запятой.

14. Дан текст.
    а) определить количество пробелов;
    б) выяснить, есть ли в нем буква «ю»;
    в) выяснить, есть ли в нем все буквы, входящие в слово «шина»;
    г) выяснить, имеется ли пара соседних букв «но» или «он»;
    д) выяснить, имеется ли пара соседствующих одинаковых символов.

15. Дана непустая последовательность непустых слов из латинских букв; соседние слова отделены друг от друга запятой, за последним словом – точка. Определить количество слов, которые:
    а) начинаются с буквы «a»;
    б) оканчиваются буквой «w»;
    в) начинаются и заканчиваются одной и той же буквой;
    г) содержат хотя бы одну букву «d»;
    д) содержат ровно три буквы «l»;
    е) найти длину самого короткого слова.

Тема 2.7. Графические возможности языка Visual Basic

Графикой у компьютерщиков принято называть все то, что связано с рисунками и изображениями.

В VB есть три графических объекта, которые позволяют работать с графикой, это:

- форма (**Form**);
- управляющий элемент графическое поле (**PictureBox**);
- рисунок (**Image**).

Форма и графическое поле – это два объекта-контейнера, которые способны содержать в себе точечный рисунок из графического файла; обладают графическими методами и позволяют с помощью графических

методов рисовать на своей поверхности; способные содержать в себе другие управляющие элементы.

Объект Image (рисунок) может только содержать в себе точечный рисунок, не обладает графическими методами и не не может включать управляющих элементов, т.е. не является контейнером.

Система координат

Как форма, так и графическое поле обладают системой координат (см. рис.16), которая применяется при любом выводе на поверхности формы или графического поля текста, графики и рисунков.

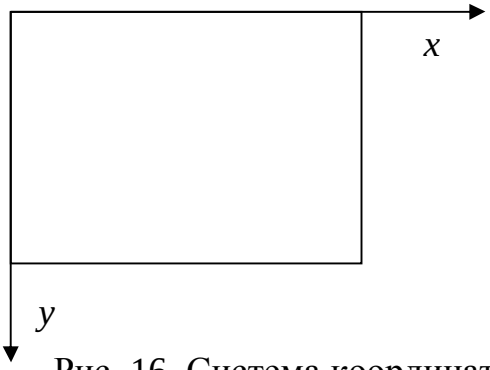


Рис. 16. Система координат

По умолчанию начало отсчета находится в левом верхнем углу объекта. Ось ***X*** направлена вправо, ось ***Y*** – вниз.Системой координат можно управлять как на этапе разработки так и при выполнении программы.

Свойство ScaleMode

Единицу измерения координат можно выбрать на этапе проектирования. За это отвечает свойство ***ScaleMode***. По умолчанию в качестве единицы измерения выбран твип. Эта единица соответствует 1/1440 дюйма или 0,0176 миллиметра. В одном сантиметре примерно 567 твипов. Можно также выбрать в качестве единицы измерения пиксели (размер точки на рисунке), пункты или символы (применяется для текста), дюймы, сантиметры или миллиметры.

Свойства ScaleLeft и ScaleTop

Значения этих свойств определяют числовые значения координат левого верхнего угла объекта. Значения этих свойств не влияют на положение контейнера.

Если для контейнера (формы или поля рисунка) задать значение свойства ScaleTop = 100, то помещенный в этом контейнере объект управления, значение свойства ***Top*** (значение координаты Y объекта) которого равно 100 будет помещен под верхнюю кромку контейнера.

Свойства ScaleWidth и ScaleHeigt

Эти свойства задают масштаб при существующей высоте и ширине объекта имеются в виду внутренние размеры. Например:

ScaleWidth = 1000
ScaleHeigt = 100

Эти инструкции определяют единицу горизонтальной оси как 1/1000 внутренней ширины объекта, и единицу вертикальной оси как 1/100 текущей внутренней высоты объекта. Если размеры формы при выполнении программы изменяются, единицы остаются теми же.

Объекты Line и Shape

Создавать собственные рисунки в VB можно с помощью графических средств управления Line (отрезок) и Shape (фигура), которые находятся на панели элементов управления.

Средство управления **Line** позволяет создавать в форме прямые линии и оформлять их стиль путем установки значений для свойств этого объекта.

Средство управления **Shape** позволяет создавать в форме прямоугольники, квадраты, окружности и овалы в зависимости от значений свойства Shape (одинаковые названия у объекта и его свойства) и также оформлять их стиль путем установки значений для свойств этого объекта.

Наиболее важные свойства объектов Line и Shape представлены в таблице 7.

Таблица 7

Свойства объектов Line и Shape

Свойство	Описание	Значение	Line	Shape
BackColor	цвет фона		–	+
BackStyle	стиль фона	Целое число в диапазоне 0-1	–	+
BorderColor	цвет контура		+	+
BorderStyle	стиль линии – точечная, сплошная или пунктирная	Целое число в диапазоне 0-6	+	+
BorderWidth	толщина линии в пикселах	Целое число в диапазоне 0-32767	+	+
FillColor	цвет заливки		–	+
FillStyle	тип заливки (текстура узора)	Целое число в диапазоне 0-7	–	+
Height	высота объекта в твипах	Целое число	–	+
Left	положение левой границы объекта в твипах	Целое число	–	+
Top	положение верхней границы объекта в твипах	Целое число	–	+
Visible	видимость		+	+
Width	ширина объекта в твипах	Целое число	–	+
X1, Y1, X2, Y2	координаты крайних точек объекта	Целое число	+	-

Графические методы рисования

Методами для рисования различных геометрических фигур обладают два объекта: форма и PictureBox. Кстати, те, же, что обладают и методом Print, который с полным основанием тоже называют графическим.

Свойства объектов, влияющие на графические методы:

DrawWidth – толщина линии.

ForeColor – цвет линии.

DrawStyle – стиль линии (сплошная, штриховая и т.п.).

FillStyle – стиль (узор) заливки и будет ли заливка.

FillColor – цвет заливки.

AutoRedraw – определяет, будет ли автоматически восстанавливаться графика и напечатанный текст, случайно стертые из-за того, что объект скрылся из виду.

DrawMode – способ наложения краски. По умолчанию значение равно 13, когда краска плотно накладывается и предыдущая картинка через нее не просвечивает. При других значениях новая краска меняет свой цвет.

Метод **Scale –** задает систему координат и масштаб для формы или графического окна:

object.Scale (x1, y1)-(x2, y2)

где:

Object – имя объекта, к которому применяется метод. Если имя объекта опущено, то метод будет применен непосредственно к форме;

(x1, y1) – координаты левого верхнего угла (начала координат) системы координат;

(x2, y2) – координаты правого нижнего угла системы координат.

Обе пары координат могут быть опущены. Тогда будет принята система координат по умолчанию с единицей измерения – твип.

Например:

```
PictureBox.Scale (-10, 2)-(10, -2)
```

Метод **PSet** устанавливает цвет отдельного пиксела на экране:

Object.Pset [Step] (x,y), [Цвет]

где:

Object – имя объекта, к которому применяется метод. Если имя объекта опущено, то метод будет применен непосредственно к форме;

Step – ключевое слово, определяющее, что координата вычисляется от текущей позиции;

(x,y) – выражения или константы, определяющие координаты для установки цвета;

Цвет – длинное целое число, определяющее цвет отметки. Если опущено, то используется текущая установка свойства ForeColor.

Например:

Pset (1000, 2000) – рисуется точка цветом, определенным свойством ForeColor.

Pset (1000, 2000), vbRed – рисуется красная точка.

Метод **Line** служит для создания отрезков, прямоугольников, замкнутых многоугольников в объекте:

Object.Line [Step] (x1,y1), [Step] (x2,y2), [Цвет], [B] [F]

где:

(x1,y1) – координаты начальной точки изображаемой линии;

(x2,y2) – координаты конечной точки изображаемой линии;

Цвет – значение, устанавливающее цвет изображаемой линии или прямоугольника;

B – указывает, что изображается прямоугольник с заданными координатами верхнего левого и правого нижнего угла;

F – указывает, что прямоугольник заполняется цветом изображаемой линии. Использовать без B нельзя.

Например:

Line (2000, 1000)-(5000,3000), vbBlack – отрезок черного цвета

Line (2000, 1000)-(5000,3000), vbGreen, B – прямоугольник зеленого цвета

Line (2000, 1000)-(5000,3000), vbYellow, BF – прямоугольник желтого цвета, залитый этим же цветом.

Line (2000, 1000)-(5000,3000), , B – прямоугольник, цвет которого определяется свойством ForeColor

Метод Circle отображает на объекте окружность, эллипс или дугу:

Object.Circle [Step] (x,y), Радиус, [Цвет, Начало, Конец, Сжатие]

где:

(x,y) – координаты центра окружности, эллипса или дуги;

Сжатие – отношение размера по оси Y к размеру по оси X.

Например:

Circle (4000, 2000), 1000, vbBlue – синяя окружность

Circle (4000, 2000), 1000, , 1, 3 – дуга окружности, начинающаяся от угла в 1 радиан и кончающаяся углом в 3 рад. (угол отмеряется против часовой стрелки)

Circle (4000, 2000), 1000, , -1, -3 – сектор круга, начинающийся от угла в 1 радиан и кончающаяся углом в 3 рад.

Circle (4000, 2000), 1000, , , 2 – эллипс, полученный из окружности радиусом 1000 горизонтальным сжатием в 2 раза.

Circle (4000, 2000), 1000, , , 1/3 – эллипс, полученный из окружности радиусом 1000 вертикальным сжатием в 3 раза.

Circle (4000, 2000), 1000, , 1, 3, 2 – дуга эллипса.

Circle (4000, 2000), 1000, , -1, -3, 2 – сектор эллипса.

Метод **Cls** просто стирает все нарисованное и напечатанное.

Метод **Point (x,y)** сообщает цвет любой точки на объекте.

Метод **PaintPicture** копирует с одного объекта на другой прямоугольный кусок изображения.

Цвет в Visual Basic

Как до сих пор мы придавали цвет какому-нибудь элементу VB?

- В режиме проектирования мы устанавливали соответствующее свойство в окне свойств. Для этого в закладке Palette выбирали один из нескольких десятков цветов.
- В режиме работы пользовались операторами типа Form1.BackColor = vbRed.

Имеется всего 8 цветов, представленных этим способом:

vbBlack – черный

vbRed – красный

vbGreen – зеленый

vbYellow – желтый

vbBlue – синий

vbMagenta – фиолетовый

vbCyan – голубой

vbWhite – белый

Но в VB существует 16 млн. цветов с лишним (точнее, 16777216)!

Так, цвет можно указывать просто числом от 0 до 16777215:

Form1.BackColor = 12456743. Недостаток этого способа – по числу трудно угадать, что за цвет.

Вспомните, что любую краску можно получить, смешав в определенной пропорции красную (Red), зеленую (Green) и синюю (Blue) краски. В VB каждой краски в смесь можно положить от 0 до 255 единиц. Смешивает краски специальная функция RGB (название – по первым трем буквам цветов).

Результатом работы функции RGB является число, обозначающее цвет. Пусть мы хотим покрасить форму краской, в которую мы положили 100 единиц красной, 200 единиц зеленой и 50 единиц синей краски. Для этого пишем такой оператор:

Form1.BackColor = RGB(100, 200, 50).

Если каждой краски положить поровну, получится серый цвет. Чем меньше каждой краски мы положим, тем темнее будет цвет, чем больше – тем светлее:

RGB(70, 90, 88) – темный цвет;

RGB(210, 250, 202) – светлый цвет;

RGB(0, 0, 0) – черный цвет;

RGB(255, 255, 255) – белый цвет.

В режиме работы мы научились задавать цвета. А в режиме проектирования? Мы видим, что цвета в окне свойств закодированы строкой из каких-то непонятных значков, например:

&H0080C0FF&.

Попробуем разобраться, что означают значки этого кода:

&H00	80	C0	FF	&
-	Сколько в цвете синей краски	Сколько в цвете зеле-ной краски	Сколько в цвете крас-ной краски	-

Количество каждой краски закодировано в шестнадцатеричной системе счисления.

Анимация

Для украшения приложения могут быть применены простые приемы анимации, для реализации которых достаточно применить имеющиеся в VB средства, например, метод **Move**, обеспечивающий перемещение и изменение размеров управляющего элемента.

Метод Move

Этот метод обеспечивает перемещение объекта (например, надписи или поля рисунка) в новое положение с заданными координатами верхнего левого угла объекта. Обращение к методу Move выглядит так:

[Объект].Move Left [, Top [, Width [, Height]]]

где Left (слева), и Top (сверху) – это координаты верхнего левого угла объекта после выполнения метода **Move**,

Width и Height – это ширина и высота объекта после выполнения метода **Move**.

Значения текущих координат объекта являются значениями свойств ***Left*** и ***Top*** объекта.

Значения текущей ширины и высоты объекта являются значениями свойств ***Width*** и ***Height*** объекта.

Для задания относительного перемещения объекта можно воспользоваться этими свойствами, например, так:

Объект.**Move** Left+δx, Top+δy

Эта команда приводит к перемещению объекта в окне формы на δy вниз и на δx вправо.

Операция перетащить и оставить

Выполнение большинства команд в приложениях для Windows часто осуществляется щелчком на кнопке. Для выполнения ряда действий в VB также предусмотрена еще одна операция – «Перетащить и оставить» (drag-and-drop). Для ее осуществления пользователь должен поместить фокус мыши на объекте, нажать левую клавишу мыши и, удерживая ее нажатой, перетащить объект на новое место, а затем отпустить кнопку мыши.

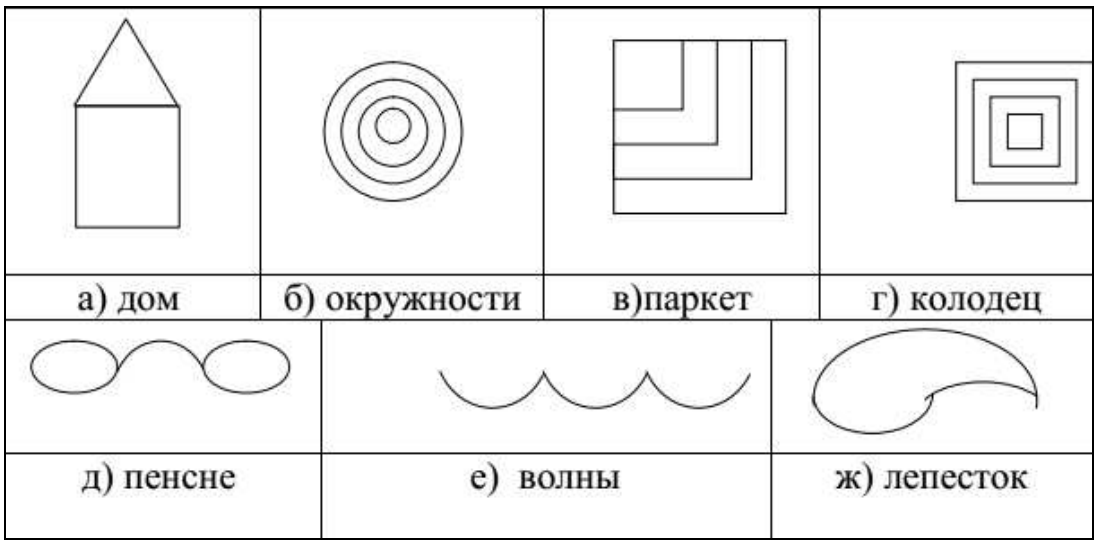
Для того, чтобы в режиме выполнения приложения объект можно было перетаскивать на новое место, следует задать его свойству ***DragMode*** (Режим перетаскивания) значение «1».

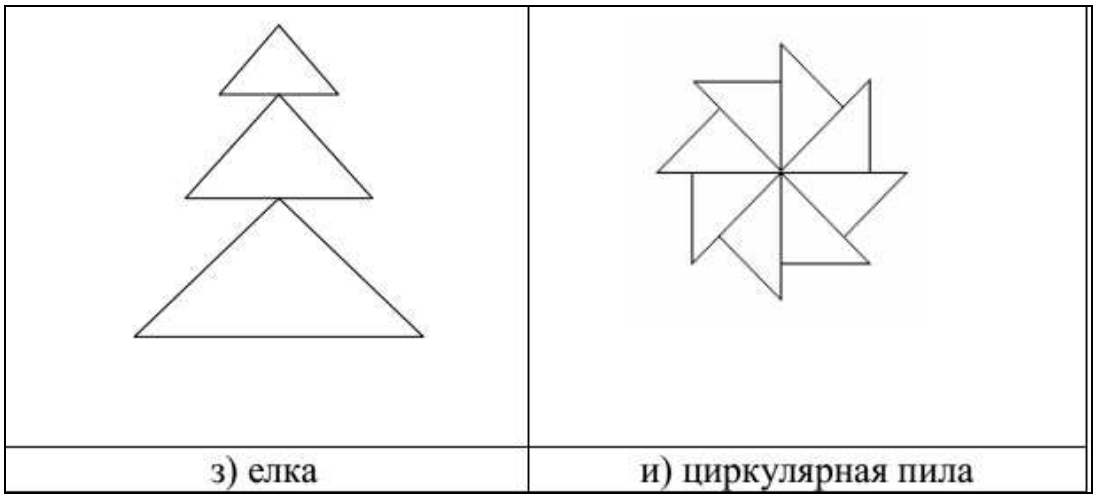
Вид указателя мыши при перетаскивании может быть задан с помощью свойства ***DragIcon*** (Значок при перетаскивании). При оставлении перетаскиваемого объекта (источника) на другом объекте (мишени) с последним объектом происходит событие ***DragDrop***.

При перетаскивании объекта (источника) через другой объект (мишень) с последним происходит событие ***DragOver***.

Задания для самостоятельной работы

1. Нарисуйте ряд точек по направлению из левого нижнего угла в правый верхний.
2. Нарисуйте «звезды в окне» – звездное небо в пределах прямоугольника.
3. Нарисуйте «дождь в луже» – заполните форму окружностями или эллипсами радиуса 200 в случайных местах.
4. Нарисуйте «мыльные пузыри» – заполните форму окружностями или эллипсами случайного радиуса и цвета в случайных местах.
5. Нарисуйте «стог сена» – множество случайных разноцветных отрезков прямых преимущественно желтоватых оттенков, причем один конец любого отрезка находится в случайной точке левой трети стога, другой – в случайной точке правой. Размер стога – 6000 на 6000. Используйте функцию RGB со случайными аргументами.
6. Нарисуйте шахматную доску.
7. Вывести на экран компьютера треугольник, положение вершин которого определяется парами чисел (200, 10), (300, 100) и (400, 10).
8. Вывести на экран компьютера закрашенный прямоугольник со сторонами, параллельными осям координат. Положение вершин одной из его диагоналей определяется парами чисел (10, 70) и (350, 200).
9. Изобразить часы и закрасить отдельные элементы.
10. Вывести на экран компьютера окружность, центр которой определяется парой чисел (200, 100), а радиус – числом 90. Отметить центр этой окружности. Закрасить внутреннюю область полученной окружности.
11. Изобразить на экране российский флаг.
12. Изобразить машину и закрасить отдельные элементы различными цветами.
13. Изобразить движение стрелки по горизонтали. Даны начальные координаты стрелки X=10, У=150.
14. Изобразить движение маятника. Даны координаты верхней точки маятника X=300, У=200, длина маятника A=60.
15. Изобразить вращение против часовой стрелки отрезка длиной A=60 вокруг точки с координатами X=300,У=200.
16. Создайте следующие рисунки:





Тема 2.8. Процедуры и функции

Понятие подпрограммы

Подпрограмма (англ. subroutine) — поименованная или иным образом идентифицированная часть компьютерной программы, содержащая описание определённого набора действий. Подпрограмма может быть многократно вызвана из разных частей программы. Главное назначение подпрограмм – структуризация программы с целью удобства её понимания и сопровождения.

Подпрограммы предоставляют следующие преимущества:

- позволяют связать часто используемую группу операторов программы со знакомым именем;
- устраняют повторяющиеся строки;
- делают программы более читаемыми;
- упрощают разработку программы;
- могут повторно использоваться в других проектах и решениях.

Параметры подпрограмм

Подпрограммы часто используются для многократного выполнения стереотипных действий над различными данными.

Параметры описываются при описании подпрограммы (в её заголовке) и могут использоваться внутри подпрограммы аналогично переменным, описанным в ней. Такие параметры принято называть *формальными параметрами*.

При вызове подпрограммы значения каждого из параметров указываются в команде вызова (обычно после имени вызываемой подпрограммы), часто называют *фактическими параметрами*.

При вызове подпрограммы фактические параметры, указанные в команде вызова, становятся значениями соответствующих формальных параметров, чем и обеспечивается передача данных в подпрограмму.

Существует несколько способов передачи параметров в подпрограмму, основными из которых являются:

- Передача параметров по значению,
- Передача параметров по ссылке,
- Передача параметров по имени.

Виды подпрограмм

В Visual Basic существует два главных типа подпрограмм: процедуры (Sub) и функции (Function).

Процедура

Процедура — самодостаточный блок кода, который может быть запущен из других блоков кода. В целом *каждая процедура содержит код, необходимый для выполнения одной задачи.*

Например, функция MsgBox является встроенной процедурой, выполняющей вывод диалогового окна.

Хотя Visual Basic имеет многие встроенные процедуры для выполнения общих действий, всегда есть случаи, когда программе требуется выполнять действия, которые встроенные процедуры не обрабатывают. Например, функция MsgBox не может отобразить диалоговое окно с рисунком. В этом случае необходимо написать собственную процедуру для выполнения этой задачи.

Процедура запускается или выполняется путем ее вызова в коде. Можно вызывать сколько угодно процедур. Процедуры выполняются в том порядке, в котором они вызываются.

Процедура (Sub) не возвращает значения, связанного с ее именем. Процедуры обычно используются для получения ввода от пользователя, отображения или печати информации, или манипуляций с несколькими свойствами, связанными с каким-либо условием. Процедуры Sub также используются для обработки и возврата одновременно нескольких переменных.

Процедуру можно помещать в стандартные модули, модули классов и форм.

Описание процедуры имеет следующий синтаксис:

Уровень Доступности **Sub** *имя Процедуры (аргументы)*

Операторы

End Sub

С помощью параметра уровень Доступности указывается, доступна ли процедура другим частям программы.

Между ключевыми словами Sub и End Sub в процедуре располагаются выполняемые при ее вызове операторы программного кода. Параметр аргументы используется для объявления передаваемых в процедуру переменных.

Важными являются следующие синтаксические элементы:

- имя Процедуры – это имя создаваемой вами процедуры Sub;
- аргументы – это необязательный список аргументов (разделенных запятыми, если их больше одного), используемых в процедуре Sub. Каждый аргумент должен быть объявлен с указанием конкретного типа данных;
- операторы – это блок операторов, который выполняет работу процедуры.

Уровень доступности (видимости) может принимать следующие значения:

- Public – процедура общедоступна в проекте, в котором определена;
- Private – процедура доступна только в том классе или модуле, в котором она определена (т.е. процедура является локальной);
- Protected – защищенные процедуры доступны внутри класса, в котором они объявлены, а также в производных от данного классах.
- Friend – дружественные процедуры доступны только внутри той сборки, в которой объявлены. Сборка – полностью самостоятельная единица приложения, которая обычно соответствует всей программе, поэтому данный модификатор можно воспринимать как указание видимости в пределах программы;
- Protected Friend – доступность процедуры расширяется на сборку и производные классы.

По умолчанию все процедуры VB (за исключением процедур обработки событий) определяются как открытые (Public). Это значит, что их можно вызвать из любой части программы – из модуля, где он создан, из другого модуля, из другого проекта.

Написание процедуры начинается с объявления процедуры. Объявление процедуры выполняет несколько задач. Оно указывает, является ли подпрограмма процедурой или функцией, присваивает процедуре имя и подробно описывает параметры, которые может иметь процедура. Ниже приведен пример простого объявления процедуры.

```
Sub MyFirstSub()
End Sub
```

Объявить процедуру можно вручную, например, впечатав в коде строку и нажав Enter.

Private Sub Farewell ()

(редактор кода автоматически добавит строку End Sub и разделитель).

Процедуры Sub подразделяются на общие процедуры и процедуры событий.

Общие процедуры

Общие процедуры — это набор операторов и выражений Visual Basic, заключенных между ключевыми словами Sub и End Sub. Каждый раз при вызове процедуры эти операторы выполняются, начиная с первого исполняемого оператора после ключевого слова Sub и заканчивая первым встреченным утверждением End Sub, Exit Sub. Процедуры sub выполняют определенные действия, но не возвращают значения. Они могут принимать аргументы, такие как переменные, константы, выражения.

Процедуры событий

Процедуры обработки событий связаны с объектами, размещенными в формах Visual Basic, или с самой формой и выполняются при возникновении события, с которым они связаны, например, щелчок мышью на какой-либо кнопке окна. Для события, связанного с формой, процедура обработки событий Sub имеет следующий синтаксис:

Private Sub имя Формы_имя События(аргументы)

Операторы

End Sub

Рис. 18. Редактирование кода процедуры в Visual Basic

Как видно из синтаксиса, наименование процедуры обработки события для формы содержит имя формы, заданное в свойстве Name, затем размещаются символ подчеркивания «_» и имя события. Например, имя процедуры, выполняемой при загрузке формы с именем Form1, будет Form1_Load, а процедуры, выполняемой при щелчке мыши на форме, – Form1_Click.

Для события, связанного с элементом управления формы, процедура обработки Sub имеет следующий синтаксис:

Private Sub *имя ЭлементаУправления_имяСобытия(аргументы)*
Операторы
End Sub

Подпрограммы Function

Подпрограмма типа Function – это группа операторов, расположенных между оператором Function и оператором End Function.

Функция — это подпрограмма специального вида, которая, кроме получения параметров, выполнения действий и передачи результатов работы через параметры имеет ещё одну возможность – она может возвращать результат.

Базовый синтаксис функции имеет следующий вид:

УровеньДоступности **Function** название_функции (аргументы) **As** тип
Операторы_функции
End Function

Важными являются следующие синтаксические элементы:

- название_функции – это имя создаваемой вами функции.
- **As** тип – это пара ключевых слов, которые определяют возвращаемый тип данных функции.
- аргументы – это список аргументов (разделенных запятыми), используемых в данной функции. Каждый аргумент должен быть объявлен с указанием конкретного типа данных.
- операторы_функции – это блок операторов, который выполняет работу функции.

В качестве уровня доступности может быть указано Public, Protected, Friend, Protected Friend или Private. Подпрограммы Function, как и переменные, имеют тип, задаваемый с помощью ключевого слова As. Если тип функции не задан, по умолчанию ей присваивается тип Object.

Тип функции определяет в свою очередь тип возвращаемого ею значения. *Возвращаемым значением* называется значение, которое функция передает обратно в вызвавшую ее программу.

Объявление функций выглядит, как и объявление процедур, однако, необходимо добавить тип возвращаемого значения (например, Integer, String и т. д.). Например, функция, которая возвращает значение типа Integer, может выглядеть следующим образом.

Function MyFirstFunction() As Integer
End Function

Ключевое слово As Integer указывает, что эта функция возвращает значение типа Integer. Внутри тела функции ей хотя бы раз должно быть присвоено какое-нибудь значение. Именно это значение функция и будет возвращать.

Вызов функции является, с точки зрения языка программирования,

выражением, он может использоваться в других выражениях или в качестве правой части присваивания. Чтобы вызвать функцию, надо в оператор программы вставить имя этой функции и все требуемые для нее аргументы.

Функции всегда возвращают значение в вызывающий код через свое имя функции. По этой причине последний оператор функции часто является выражением присвоения, которое помещает окончательный результат вычислений функции в имя функции.

Параметры функций и процедур

Передача параметров

Переменные, передаваемые подпрограмме, называют *параметрами подпрограммы*.

Параметры очень похожи на переменные. Они имеют тип и имя, а также хранят сведения, как переменные. Их можно использовать так же, как переменные в подпрограмме. Ниже приведены два основных различия между параметрами и переменными:

- Параметры объявляются в объявлении подпрограммы (процедуры или функции), а не в отдельных строках кода.
- Параметры могут использоваться только в подпрограмме, в которой они объявлены.

Параметры объявляются при объявлении подпрограммы в скобках, следом за именем подпрограммы. Ключевое слово As используется для объявления типа, каждому параметру обычно предшествует ключевое слово, показывающее способ передачи параметров.

Передача параметров в процедуру может осуществляться несколькими способами. Чаще всего применяемыми являются способы:

- *по значению* (by value – ByVal);
- *по ссылке* (by reference – ByRef).

В первом случае в подпрограмму в качестве переменной передается не сама переменная, а ее копия.

Оператор End

Оператор **End** заставляет VB завершить работу не только подпрограммы, а всего проекта и проект переходит в режим проектирования.

Оператор End можно поместить в любое место подпрограммы, чтобы заставить все приложение остановиться. End закрывает все файлы, открытые оператором Open и очищает все переменные приложения.

Задания для самостоятельной работы

1. Даны три пары целых переменных. Поменять местами их значения (попарно).
2. Даны 6 переменных A, B, C, H, P, K. Найти наибольшее, используя две процедуры: нахождения наибольшего из двух значений и нахождения наибольшего из трех значений.

3. Даны координаты трех вершин треугольника. Найти длины всех его сторон и площадь, если треугольник существует. Процедура нахождения стороны.

4. Даны четыре числа. Для каждого числа найти все его делители и подсчитать их количество.

5. Найти сумму наименьших (наибольших) цифр пяти натуральных чисел.

6. Написать функцию, подсчитывающую количество цифр натурального числа. Используя ее определить, в каком из двух заданных чисел больше цифр.

7. Найти сумму цифр трех чисел.

Тема 2.9. Работа с массивами

Массив — это множество однотипных данных, имеющих одинаковое имя и отличающихся друг от друга только порядковым номером, который называется индексом.

На практике массивы применяются для хранения и обработки больших объемов однотипных данных. Но эти данные доступны только во время выполнения программы. После завершения работы программы данные, хранящиеся в массивах, теряются.

Для обработки массивов используются специальные алгоритмы. Наиболее распространенными из них являются сортировка массива, поиск максимального и минимального элементов массива, формирование нового массива из элементов исходного массива, поиск одного или нескольких элементов, удовлетворяющих некоторому заранее заданному условию. Мы подробно рассмотрим большинство основных алгоритмов, но сначала сформулируем основное правило обработки одномерных массивов.

Массивы всегда обрабатываются в цикле.

Причем счетчик цикла используется в качестве индекса массива. В зависимости от количества индексов массивы делятся на одномерные (с одним индексом) и многомерные (с двумя и более индексами).

Одномерный массив – это средство языка программирования, которое позволяет обращаться к любому элементу пронумерованного множества значений. При этом все элементы множества должны иметь одинаковый тип данных, а их количество не должно превосходить некоторого заранее заданного числа, которое называется размером массива.

1. Объявление одномерного массива

Объявление массива выполняется аналогично объявлению переменной. Для этого используются уже знакомые нам операторы Dim, Static, Public и Private.

По способу описания массивы делятся на две основные группы:

- массив с заранее известным числом элементов;
- массив, число элементов которого заранее неизвестно.

При описании массива указывается его имя, затем ставятся круглые скобки, показывающие Visual Basic, что мы организуем массив, и задается тип данных, к которому будут принадлежать все элементы массива.

Например:

```
Dim a() As Integer
Dim b(10) As Single
```

Обратите внимание, что нумерация элементов массива всегда начинается с нуля независимо от способа объявления массива. Размер этого массива тоже можно будет изменять в процессе выполнения программы с помощью оператора ReDim.

Оператор **ReDim** предназначен для изменения размера массива в процессе выполнения программы. Причем размер массива можно как уменьшать, так и увеличивать. Но он не позволяет изменить тип элементов массива и размерность массива (количество используемых индексов). При изменении размера массива данные, хранящиеся в нем, могут теряться. Чтобы этого не происходило после слова ReDim необходимо поставить ключевое слово Preserve. В общем виде оператор ReDim записывается следующим образом:

ReDim Имя массива(Номер последнего элемента) или

ReDim Preserve Имя массива(Номер последнего элемента)

Объем памяти, необходимой для хранения массива, определяется как произведение количества элементов массива и объема памяти, занимаемого одной переменной указанного типа данных.

Для того чтобы обратиться к некоторому элементу массива, необходимо указать имя массива и в круглых скобках номер нужного элемента. Например:

```
a(3) = 5
b(5) = (a(0) + a(4)) / b(1)
```

2. Ввод массива

Ввести массив – значит задать его размер и указать значения всех его элементов. Существует два основных способа заполнения массива. В первом случае значения элементов последовательно вводятся с клавиатуры. Во втором случае они задаются с помощью генератора случайных чисел.

Задание значений элементов массива с клавиатуры – это самый распространенный способ ввода массива. Он состоит из двух этапов. На первом шаге указывается количество элементов в массиве и соответствующим образом переопределяется размер массива. На втором шаге организуется цикл, на каждом шаге которого вводится значение одного элемента.

Второй способ ввода массива – это заполнение случайными числами. В Visual Basic есть специальная функция, которая по определенным правилам генерирует рациональные случайные числа в диапазоне [0;1). Она называется Rnd(). Используя эту функцию, можно заполнить массив случайными числами из любого диапазона.

Функция Rnd() возвращает рациональное случайное число в диапазоне [0; 1). Умножив это значение на разность между концом и началом желаемого диапазона, получим диапазон [0; fin -start). Так как в общем случае начало диапазона отличается от нуля, то полученный диапазон случайных чисел надо сдвинуть в требуемую точку. Для этого добавим к нашему числу величину равную началу диапазона. Получим [0 + start; fin –start + start) или [start; fin).Так как заполняемый массив является целочисленным, то полученное значение необходимо округлить. Описанный процесс реализуется следующим арифметическим выражением.

a(i) = Round(start + (fin -start) * Rnd())

Двумерный массив можно представить себе в виде таблицы, в которой все строки и столбцы пронумерованы.

Каждый элемент такого массива имеет два индекса:

Первый индекс – это номер строки;

Второй индекс – номер столбца.

A[1,1]	A[1,2]	A[1,3]	A[1,4]	A[1,5]
A[2,1]	A[2,2]	A[2,3]	A[2,4]	A[2,5]
A[3,1]	A[3,2]	A[3,3]	A[3,4]	A[3,5]
A[4,1]	A[4,2]	A[4,3]	A[4,4]	A[4,5]

3. Объявление двумерного массива

Описание двумерного массива практически не отличается от описания одномерного массива. Сначала указывается имя матрицы, затем – круглые скобки, между которыми ставится одна запятая. Она показывает Visual Basic, что мы организуем массив, который будет иметь два индекса. Последним задается тип данных, к которому будут принадлежать все элементы матрицы.

Существует два способа объявления матрицы.

Первый способ

Dim a(,) As Integer

Эта конструкция описывает двумерный целочисленный массив a(,)типа Integer, размеры которого заранее неизвестны. Изначально в нем нет ни одного элемента. Размеры этой матрицы будут определены позднее с помощью оператора ReDim.

Второй способ

Dim b(10, 18) As Single

Такая запись определяет двумерный массив b(,) типа Single, состоящий из 11 строк и 19 столбцов. Строки матрицы будут пронумерованы от 0 до 10, а столбцы – от 0 до 18.Обратите внимание, что при описании матрицы всегда сначала указывается номер последней строки, а потом –номер последнего столбца. Обратите внимание, что нумерация строк и столбцов матрицы всегда начинается с нуля независимо от способа ее объявления. Размеры этого массива

тоже можно будет изменять в процессе выполнения программы с помощью оператора ReDim.

Объем памяти, занимаемой матрицей, вычисляется как произведение числа строк, числа столбцов и объема памяти, который занимает одна переменная указанного типа данных.

При обращении к отдельному элементу двумерного массива необходимо в круглых скобках указать оба индекса, разделив их запятой. Обратите внимание, что всегда сначала указывается номер строки, а затем –номер столбца. Нумерация всегда начинается с нуля.

Для обработки двумерных массивов применяется специальная конструкция из двух циклов, один из которых находится в теле другого. Такая конструкция называется вложенные циклы. Она записывается следующим образом.

```
For i = 0 to 10
For j = 0 to 18
A(i, j) = i + j
Next
Next
```

4. Ввод прямоугольной матрицы

Ввести матрицу – значит, задать ее размеры и указать значения всех ее элементов. Как и при обработке одномерного массива, значения элементов матрицы можно задавать двумя способами: вводя значения с клавиатуры или пользуясь генератором случайных чисел.

Задания для самостоятельной работы

Одномерные массивы

1. Найти сумму положительных элементов массива.
2. Найти максимальный элемент массива и его номер при условии, что все элементы массива различны.
3. Найти минимальный элемент массива.
4. Найти номера отрицательных элементов (вывести их на экран), если таких нет, то сообщить об этом.
5. Сколько элементов массива превосходят по модулю заданное число.
6. Найти количество нечетных элементов.
7. Найти количество отрицательных элементов массива.
8. При выводе на экран элементов массива:

а) после каждого элемента, кроме последнего, поставить точку;

б) пропустить все отрицательные;

в) удалить все отрицательные элементы.

9. Упорядочить массив A(N) по возрастанию.
10. Расположить элементы целочисленного массива по убыванию.
11. Имеется массив X. Определить индексы отрицательных элементов данного массива. Массив X содержит m элементов.

12. Вычислить сумму элементов массива C, стоящих на нечетных местах. Массив C содержит 10 элементов.

13. Для целочисленного массива A, содержащего 10 элементов, определить, кратна ли сумма его элементов 7.

14. Элементы массива A(K) получить по формуле: y=2*x^2-5*x+6. Найти сумму элементов, имеющих номера индексов кратных 3.

15. Вычислить среднее арифметическое и среднее геометрическое элементов массива C(n).

16. Для целочисленного массива Y вычислить среднее геометрическое элементов, кратных трем. Массив Y содержит K элементов.

17. Дан массив B из десяти элементов. Организовать новый массив, элементы которого расположены в обратном порядке.

18. Расположить в массиве R сначала положительные, а затем отрицательны
элементы массива Z(12).

19. Из массива X, содержащего 15 элементов, в массив Y переписать подряд отрицательные элементы.

20. Дан одномерный массив. Переставить в обратном порядке элементы массива, расположенные между минимальным и максимальными элементами.

21. Дан массив целых чисел (n=20), заполненный случайным образом числам из промежутка [-45, 95].

а) удалить из него все элементы, кратные 7 и принадлежащие промежутку [a, n] (a и b вводить с клавиатуры);

б) вставить число k между всеми соседними элементами, которые образуют пару элементов с одинаковыми знаками (k вводить с клавиатуры);

в) переставить в обратном порядке часть массива между элементами с номерами k1 и k2, включая их. Сделать проверку корректности ввода k1 и k2, если ввод неправильный, то ничего не делать.

Двумерные массивы

1. Найти наибольший и наименьший элементы матрицы X(k, n) и поменять их местами.

2. Найти максимальный элемент главной диагонали матрицы X(5, 5).

3. Дана матрица Y(m, n). Найти столбец с наибольшей и наименьшей суммой элементов.

4. Имеется матрица S(m, n). Найти максимальный из всех минимальных элементов строк. Вывести номер строки матрицы S, в которой расположено выбранное число.

5. Вычислить сумму элементов двух главных диагоналей матрицы C(5, 5).

6. Из матрицы Y(k, k) получить вектор T, элементами которого являются элементы главной диагонали матрицы.

7. Сформировать диагональную матрицу A(p, p). У диагональной матрицы все элементы равны нулю, кроме диагональных.

8. Получить матрицу K(5, 5), элементами которой являются квадраты сумм номеров строк и столбцов.

9. Из матрицы X(5, 4) получить матрицу H, поменяв местами строки и столбцы.

10. Имеется целочисленная матрица A(k, l). Проверить, есть ли в ней элементы, равные нулю. Если есть, найти номер первого из них.

11. Определить:

а) есть ли в данном массиве отрицательный элемент;

б) есть ли два одинаковых элемента;

в) есть ли данное число A среди элементов массива (массив имеет размер n×n).

12. Заполнить квадратный массив B размером n×n. Например, для n=6

1	12	13	24	25	36
2	11	14	23	26	35
3	10	15	22	27	34
4	9	16	21	28	33
5	8	17	20	29	32
6	7	18	19	30	31

Тема 2.10. Работа с файлами

Современные компьютеры имеют оперативную и внешнюю память. В оперативной памяти находятся одна или несколько открытых программ и обрабатываемые ими данные. После закрытия программы ее данные в оперативной памяти не сохраняются, так как используемые ячейки памяти выделяются для данных другой программы. Объем оперативной памяти ограничен, ее стоимость относительно высока, информация не сохраняется при выключении питания компьютера – все это не позволяет ее применять для постоянного хранения больших объемов информации. Для этой цели используется внешняя память. Это жесткие и гибкие магнитные диски, оптические диски, магнитные ленты, позволяющие хранить сотни мегабайт и гигабайты информации, стоимость которых относительно невелика.

Хранение больших объемов информации на внешних носителях с учетом того, что время доступа к данным на внешних носителях на один, два порядка выше времени доступа к данным в оперативной памяти, требует их хорошо продуманной организации. Данные на внешних носителях информации хранятся в виде файлов.

Файл – это поименованная область памяти на внешнем носителе (магнитный диск, лента и т.п.), содержащая некоторые данные или программу.

Файл состоит из записей. Запись характеризуется длиной, выраженной в байтах.

Запись состоит из данных, которые передаются между оперативной и внешней памятью за одну операцию чтения или записи данных.

Чтение данных – это передача данных из внешней памяти в оперативную память, запись данных – это передача данных из оперативной памяти во внешнюю память.

Работа с файлами включает этапы:

- получение номера свободного канала ввода-вывода, с которым будет связан файл;
- открытие файла;
- чтение или запись данных;
- закрытие файла;
- удаление файла.

Номер файла

При открытии файлу ставится в соответствие канал с определенным номером. Каждый открытый файл имеет свой канал. Функция FreeFile () возвращает номер свободного канала (номер, или дескриптор файла), который можно использовать для очередного открываемого файла.

FreeFile **[(RangeNumber)]**

Необязательный параметр RangeNumber может принимать значение 0 (по умолчанию) и 1. Если его значение равно 0, то возвращается номер канала из диапазона 1– 255, если 1, то из диапазона 256 – 511.

Пример: n = FreeFile

Типы доступа к файлам

Реализуются три типа доступа к файлам:

1. последовательный (Sequential) – для чтения и записи текстовых файлов;
2. произвольный (Random) – для чтения и записи текста или структурированных двоичных файлов с записями фиксированной длины;
3. двоичный (Binary) – для чтения и записи произвольно структурированных файлов.

Для открытия файла с последовательным доступом используется команда

Open ИМЯ_ФАЙЛА For РЕЖИМ_РАБОТЫ As #НОМЕР_ФАЙЛА,

где – ИМЯ_ФАЙЛА – имя с расширением и маршрутом (полным путем);

режимы работы:

1) Append: – файл открывается для помещения в него записей, если он уже содержит какие то записи, то новые помещаются в конец файла;

2) Input – файл открывается для чтения из него записей;

3) Output: – файл открывается для помещения в него записей; НОМЕР_ФАЙЛА – целое число между 1 и 255, предварительно определяемое с помощью функции FreeFile;

обращение к файлу из программы выполняется под этим номером.

После того, как файл обработан, его закрывают командой Close #НОМЕР_ФАЙЛА.

Доступ к файлу возможен между командами Open и Close.

Для помещения записи в файл используется команда

Print #НОМЕР_ФАЙЛА, ПЕРЕМЕННАЯ [, ПЕРЕМЕННАЯ]

Для чтения записи из файла используются следующие команды:

1) чтение одной строки:

Line Input #НОМЕР_ФАЙЛА, str_Переменная

2) чтение всего файла в строковую переменную (strText):

strText = Input$ (LOF(НОМЕР_ФАЙЛА), НОМЕР_ФАЙЛА),

где LOF() – функция определения длины файла в байтах;

3) чтение последовательности определенного количества символов:

Input #НОМЕР_ФАЙЛА, ПОЛЕ_ДАННЫХ_1 [, ПОЛЕ_ДАННЫХ_2]

Функция EOF(#НОМЕР_ФАЙЛА) возвращает логическое значение ИСТИНА (1), если достигнуто окончание файла, и значение ЛОЖЬ (0) – в противном случае.

Открытие файла для произвольного доступа осуществляется командой

Open ИМЯ_ФАЙЛА For Random [Acces ДОСТУП] As #НОМЕР_ФАЙЛА [Len = ДЛИНА_ЗАПИСИ],

где параметр Acces задает режим доступа к файлу: Read - чтение, Write - запись, Read Write - чтение и запись (без указания параметра также чтение и запись).

Для записи используется команда

Put #НОМЕР_ФАЙЛА, НОМЕР_ЗАПИСИ, ПЕРЕМЕННАЯ

для считывания команда

Get #НОМЕР_ФАЙЛА, НОМЕР_ЗАПИСИ, ПЕРЕМЕННАЯ

Открытие файла для двоичного доступа осуществляется командой

Open ИМЯ_ФАЙЛА For Binary [Acces ДОСТУП] As #НОМЕР_ФАЙЛА,

длина записи не указывается, т. к. обмен происходит побайтно; для ввода и вывода используются те же операторы Get и Put, но вместо номера записи указывается номер байта.

Контрольные вопросы

1. Что такое файл?
2. Какие этапы включает в себя работа с файлами?
3. Что называют номером файла?
4. Какие типы доступа к файлам существуют?
5. Каковы режимы работы с файлом?

Задания для самостоятельной работы

1. Дан файл F, компоненты которого являются действительными числами. Найти:

а) сумму компонент файла;

б) произведение компонент файла;

в) сумму квадратов компонент файла;

г) среднее арифметическое элементов.

2. Дан файл F, компоненты которого являются целыми числами. Найти:

а) количество четных чисел среди компонент файла;

б) каких чисел больше: четных или нечетных.

3. Дано натуральное число N. Записать в файл целые числа B1, B2, …, Bn, определенные ниже:

а) I I = 1, 2, 3, …, N

б) I**!**

4. Дан символьный файл F. Добавить в файл символы E, N, D.

5. Дан символьный файл F.

а) подсчитать число вхождений в файл сочетаний AB;

б) определить, входит ли в файл сочетание abcdef;

в) подсчитать число вхождений в файл каждой из букв a, b, c, d.

6. Определить количество слов в символьном файле.

7. Дан текстовый файл. Составить программу вывода на экран его содержимого:

а) в обратном порядке;

б) с исключением символов, отличных от букв и пробела.

8. Дан текстовый файл. Дописать в его конце следующие данные: количество строк, количество символов в каждой строке, количество элементов в каждой строке.

9. Сформируйте типизированный файл из записей, каждая из которых содержит простое число и его квадрат. Включите в создаваемый файл все простые числа, не превосходящие 1000. Выведите содержимое созданного файла на экран.

10. Дан текстовый файл *in.txt*, содержащий произвольный текст. Получить файл *out.txt*, содержащий исходный текст с выровненными краями. Выравнивание производить за счет равномерной вставки пробелов в более короткие строки.

ЛИТЕРАТУРА

1. Бойченко Л.П. Введение в Visual Basic: Метод. указания. – Ухта: УГТУ, 2006.
2. Браун С. Visual Basic 5 с самого начала. – СПб: Питер, 1998.
3. Голицына О.Л., Попов И.И. Основы алгоритмизации и программирования: Учебное пособие. – М.: ФОРУМ, 2010.
4. Жаров М.В., Палтиевич А.Р., Соколов А.В. Основы информатики: Учебное пособие. – М.: ФОРУМ, 2011.
5. Райтингер М., Муч Г. Visual Basic – для пользователя. – Киев, 1999.
6. Петрусос Е. Руководство разработчика Visual Basic. – Киев, 2000.
7. Макарова Н.В. Информатика: Учебник / Под ред. Н.В. Макаровой. – М.: Финансы и статистика, 2004.
8. Острейковский В.А. Информатика: Учебник для вузов. – М.: Высшая школа, 2001.
9. Симонович С.В. Информатика: Базовый курс. – СПб: Питер, 2003.
10. Харвореон М. Microsoft Visual Basic: Шаг за шагом: Практическое пособие / Перевод с англ. М. Харвореона. – М.: ЭКОМ, 1998.
11. Шоль Н.Р. Информатика: Учеб. пособие. – Ухта: УГТУ, 2000.

Елена Анатольевна Быкадорова
Оксана Николаевна Синявская

Основы программирования информационного контента

Учебное пособие
для студентов специальности
09.02.05 Прикладная информатика (по отраслям)
и преподавателей профессиональных образовательных учреждений

Подписано в печать 12.01.16 г. Формат 60х80/16.
Бумага офсетная. Печать цифровая. Объем 4,5 п.л.
Тираж 100 экз. Заказ № 4184
Изготовлено в «Копицентр»
г. Ростов-на-Дону, ул. Суворова, 19.