Министерство образования и науки Российской Федерации

федеральное государственное бюджетное образовательное учреждение
высшего образования
"САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПРОМЫШЛЕННЫХ ТЕХНОЛОГИЙ И ДИЗАЙНА"

Кафедра информационных систем и компьютерного дизайна

ИНФОРМАТИКА И ПРОГРАММИРОВАНИЕ.

Часть 2

Методические указания к выполнению лабораторных работ
для студентов направления подготовки бакалавриата
09.03.03 – Прикладная информатика
всех форм обучения

Составитель
С. В. Лебедева

Санкт-Петербург
2017

Утверждено

на заседании кафедры

протокол № 1от 01.09.2016 г.

Рецензент Н. Р. Туркина

Методические указания представляют собой практическое руководство к выполнению лабораторных работ по дисциплине "Информатика и программирование" 2-я часть. В них содержится необходимый теоретический и практический материал для изучения объектно-ориентированного программирования на языке VBA в MS Excel, описание базовых алгоритмических структур и характерных приемов программирования в синтаксисе языка VBA.

Методический материал для приобретения практических навыков предваряет задания на самостоятельную работу студентов. В нем пошагово расписывается решения аналогичных задач с подробной иллюстрацией графическим материалом. Методические указания позволят закрепить теоретические знания по программированию для решения инженерно-технических задач

Учебное электронное издание сетевого распространения
Издано в авторской редакции

Системные требования:
электронное устройство с программным обеспечением для воспроизведения файлов формата PDF

Режим доступа: http://publish.sutd.ru/tp_get_file.php?id=201760 по паролю. – Загл. с экрана.
Дата подписания к использованию 18.01.2017 г. Рег. № 60/17

ФГБОУВО «СПбГУПТД»
Юридический и почтовый адрес: 191186, Санкт-Петербург, ул. Большая Морская, 18.
http://sutd.ru

Содержание

Введение

Процессы решения современных задач требуют значительных интеллектуальных затрат и переработки больших объемов информации в ограниченные сроки. С целью сократить расходы на решение задач и были созданы электронно-вычислительные машины (ЭВМ), а затем и персональные компьютеры (ПК).

Однако у современных ПК отсутствует способность мышления. Чтобы иметь возможность перепоручить машине решение задачи, необходимо точно описать каждый шаг в процессе поиска ее решения. Для этого надо знать *метод решения* задачи, выбор которого зависит от построения формализованного описания задачи, ее *математической модели*. Однако для получения конкретных результатов недостаточно создание лишь метода, пригодного для решения задачи. Важно уметь применять его для решения заданных практических задач в определенных условиях.

Важным этапом освоения технологии решения задач на компьютере является развитие навыков разработки алгоритмов, их правильного представления в соответствии с общепринятыми стандартами, знание базовых алгоритмических структур. Алгоритмическое мышление помогает сформировать следующие основные навыки решения задач:

- умение правильно планировать структуру предстоящих действий для достижения заданной цели при помощи стандартного набора средств;
- строить информационные структуры для описания объектов и процессов в конкретной предметной области;
- правильно организовывать поиск информации, необходимой для решения задачи;
- четко и однозначно формулировать способ решения задачи в общепринятой форме и правильно понимать способ решения, предложенный другим разработчиком;
- формировать навыки анализа имеющейся информации, умения представлять ее в структурированном виде.

Формализованное алгоритмическое описание задачи является основой для ее кодированной формы записи, или иначе, программной. Для этого необходимо ввести обрабатываемые данные, указать, как их обрабатывать, задать способ вывода полученных результатов. С этой целью необходимо знать:

- как ввести информацию в память (ввод);
- как хранить информацию в памяти (данные);
- как указать правильные команды для обработки данных (операции);
- как передать обратно данные из программы пользователю (вывод);

– как упорядочить команды таким образом, чтобы:

o некоторые из них выполнялись только в том случае, если соблюдается определенное условие, или ряд условий (условное выполнение);

o некоторые выполнялись повторно некоторое число раз (циклы);

o некоторые выделялись в отдельные части, которые могут быть неоднократно выполнены в разных местах программы (подпрограммы).

Таким образом, чтобы начать программировать, необходимо умело использовать семь основных элементов программирования:

– ввод – считывание значений, поступающих с клавиатуры или файла на диске;

– данные – константы, переменные, структуры, содержащие числа, текст или адреса;

– операции – осуществляют присваивание значений, их комбинирование (сложение, деление и т.д.) и сравнение значений;

– условное выполнение – выполнение определенного набора команд в зависимости от выполнения условия;

– циклы – организация повторного выполнения некоторого набора команд либо фиксированное число раз, либо в зависимости от ложности или истинности некоторого условия;

– подпрограммы – набор команд, который имеет собственное имя, и может быть вызван любое число раз из произвольного места основной программы;

– вывод – вывод результатов задачи в форме, доступной для восприятия пользователю.

Целью программирования является описание процессов обработки данных. Процесс можно определить и описать как последовательность сменяющих друг друга состояний некоторой информационной среды. Набор данных, содержащихся в какой-либо момент в информационной среде, определяет состояние этой информационной среды.

Для того чтобы по заданному описанию требуемый процесс порождался автоматически на каком-либо компьютере, необходимо, чтобы это описание было формализованным. Такое описание называется *программой*. Любая программа должна быть понятной как компьютеру, так и человеку, так как и при разработке программ и при их использовании часто приходится выяснять, какой именно процесс она порождает. Поэтому программа составляется на удобном для человека формализованном языке программирования.

Основы программирования на *VBA*

Visual Basic For Application (VBA) – визуальный объектно-ориентированный язык программирования приложений.

Ключевой идеей объектно-ориентированного программирования является объединение данных и оперирующих с ними функций в один объект.

Язык *VBA* не существует вне какого-либо приложения. Он встроен в такие приложения как редактор электронных таблиц *Excel, СУБД MS Access*, текстовый редактор *WORD*. В каждом из этих приложений существуют свои объекты, которые могут строиться на основе более мелких объектов - элементов управления и объединяются в более крупные объекты – семейства. Объект представляет собой элемент приложения, такой как, например:

- лист (*Worksheet*), ячейка (*Cells*), диапазон (*Range*) в *MS Excel*;
- форма (*Form*) или отчет (*Report*) в *СУБД MS Access*;
- документ (*Document*), абзац (*Paragraph*), стиль (*Style*) *в MS Word*.

Структура программы в VBA

В *VBA* поддерживается следующая структура программы. На высшем уровне иерархии стоит приложение, далее идут проекты, связанные с фактическими документами этого приложения, на третьем уровне находятся модули (модули приложения, модули класса, стандартные модули, модули форм и модули ссылок). И на последнем уровне находятся процедуры и функции этих модулей. Схематически данная иерархия отображена на *рис. 1*.

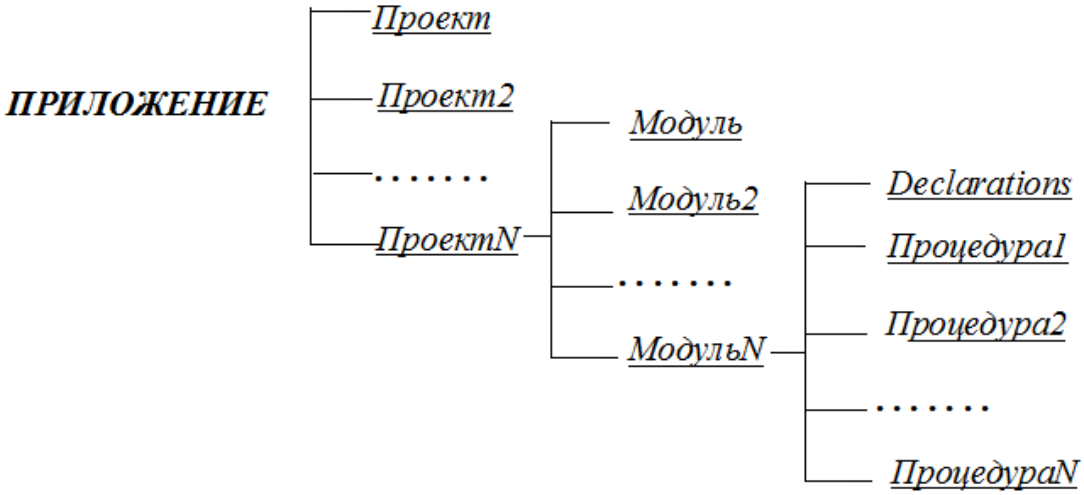


Рис. 1. Структура программы на языке *VBA*

Проектом *называется совокупность модулей, связанных с основным документом приложения.*

Всю совокупность используемых модулей можно классифицировать следующим образом:

модуль объекта приложения. Модули этого типа всегда связаны с объектами, реагирующими на события. Главное назначение подобных модулей состоит в том, что они содержат заготовки процедур реакций на события.

– ***модуль класса*** - модуль, содержащий описание объекта, в том числе описания его свойств и методов;

– ***модуль пользовательской формы*** содержит процедуры обработки событий объектов этих форм;

– ***стандартный модуль.*** В стандартных модулях содержатся общие процедуры, не связанные ни с каким объектом, а также часто используемые процедуры, которые могут быть запущены из любого другого модуля проекта. Вызов этих процедур может осуществляться разными способами – из процедур обработки событий, процедур других стандартных модулей, макросов и просто выражений.

Стандартный модуль включает только общую область, которая состоит из раздела описаний (*Declaration*), процедур и функций. Раздел описания *Declaration* содержит определение переменных, констант и данных специального типа уровня модуля.

Описания в стандартном модуле по умолчанию являются общими (*Public*).

Для исключения возможных ошибок при работе с переменными в раздел описания должна быть включена инструкция ***Option Explicit***, которая требует явного описания всех переменных, используемых в процедурах.

Процедуры и функции

В языке *VBA* замкнутыми программными единицами являются процедуры.

Процедура *– это независимая именованная часть программы, которую можно вызвать по имени в любом месте основной программы для выполнения определенных действий.*

Обращение к процедуре производится по имени процедуры. При необходимости за именем процедуры в круглых скобках указываются список параметров, значения которых передаются в процедуру. Упоминание этого имени в тексте любой программы называется вызовом процедуры.

Процедура не возвращает значения, и не может выступать как операнд в выражении.

Процедуры стандартного модуля могут быть ***общими*** *(Public)* и *личными* ***(Private)***. Личные процедуры могут использоваться только внутри модуля, в котором они определены.

Общие процедуры доступны для всех модулей проекта.

Процедура содержит ***раздел операторов*** (*исполняемых инструкций*), может содержать или не содержать раздел описания.

Процедуры имеют стандартное оформление:

[Private | Public] [Static] Sub ИмяПроцедуры ([список параметров])

раздел описания переменных

тело процедуры

[Exit Sub]

End Sub

- ***Public*** - указывает, что процедура *Sub* доступна для всех других процедур во всех модулях.
- ***Private*** - указывает, что процедура Sub доступна для других процедур только того модуля, в котором она описана.
- ***Static*** - указывает, что локальные переменные процедуры Sub сохраняются в промежутках времени между вызовами этой процедуры.

Раздел описания переменных *применяется для описания переменных, используемых в программе*.

Объявление переменной начинается со служебного слова ***Dim***. Инструкция *Dim* указывает компилятору на необходимость отведения блока оперативной памяти для хранения значений переменной заданного типа.

Синтаксис: ***Dim ИмяПеременной As ТипДанных***

ИмяПеременной - *это значимое название переменной, на которое можно ссылаться в программе.*

ТипДанных - существующий тип данных, подходящий для размещения информации определенного рода.

Тело процедуры может содержать от одной и более инструкций, написанных в соответствии с правилами программирования на данном языке.

Для того чтобы вызвать любую процедуру из тела другой процедуры используется инструкция ***Call***. Синтаксис вызова следующий:

Sub ПРИМЕР1 ()

……

Call ИмяВызываемойПроцедуры *([необязательный список передаваемых параметров])*

……

End Sub

Процедуры существуют в двух видах:

– ***процедура-подпрограмма Sub***, которая может получать аргументы, такие как, константы, переменные, или выражения, которые позволяют передать в процедуру требуемые значения.

Например:

Sub Name (age as string)
Dim s as string
s=InputBox(“Введите Ваше имя”)
MsgBox ”Ваше имя –“ & s & ”Ваш возраст - ” & age
End Sub

В этом примере в процедуру в качестве аргумента передается значение переменной *age*, которое затем используется в инструкции вывода *MsgBox*.

– ***процедура-функция (Function,*** *часто такие процедуры называют просто* ***функциями***). Функция как и процедура - независимая именованная часть программы, но имеются два отличия:

- *функция* ***возвращает скалярное значение*** *(передает в точку вызова), полученное в результате расчетов. Это значение в дальнейшем можно использовать в различных выражениях*;
- ***имя функции может входить в выражение как операнд.***

Процедура-функция *Function* также может получать аргументы в качестве параметров, которые затем используются далее при расчетах.

Синтаксис процедуры-функции:

Function ИмяФункции ([необязательный список параметров]) [As Тип]
раздел описания переменных
тело функции
ИмяФункции=ВозвращаемоеЗначение *- инструкция возврата значения*
[Exit Function]
End Function

Необязательный параметр [As Тип] позволит явно задать тип данных, который возвращает функция. Если он опущен, то по умолчанию возвращается значение типа *Variant.* Например:

Function ДлинаОкружности (Радиус As double) As double
Const Pi=3.14159
ДлинаОкружности=2 Pi* Радиус*
End Function

Обращение к функции можно использовать в выражениях наряду с переменными, например,

x=sin2(x)+ 3*cos(2*x)+ИмяПользовательскойФункции

Все процедуры и функции делятся на две группы: ***встроенные (стандартные)*** и ***определенные пользователем.***

Стандартные процедуры и функции входят в состав языка и вызываются по строго фиксированному имени. Например, математические функции – sin(), cos(), tan(), exp() и др.

Кроме математических функций, библиотека встроенных функций может содержать статистические функции, текстовые, функции даты и времени, логические функции и т. д.

Процедуры и функции, *определенные пользователем* добавляются в библиотеку приложения и могут быть вызваны в любом модуле приложения (если они описаны с ключевым словом *Public*).

Понятие данных

Программа реализует алгоритм решения задачи. В ней записывается последовательность действий, выполняемых над данными с помощью определенных операций. Применительно к языкам программирования ***- данные - это величины, обрабатываемые программой на выбранном языке программирования.***

Для хранения данных, определенных в программе, транслятором языка программирования резервируется определенный участок оперативной памяти. Поскольку данные могут храниться только в двоичной (иначе машинной) форме, то в зависимости от типа данных, может резервироваться либо 1 байт памяти, слово (2 байта), двойное слово (4 байта) памяти.

Все формы представления данных, заложенные в язык «от рождения», называются ***базовыми типами данных.*** Это обычно целые и вещественные числовые типы, символьные и строковые типы. Помимо базовых типов можно конструировать свои формы представления данных − производные типы данных.

Имеются три основных вида данных, которые существуют в любом языке программирования это - *константы, переменные и массивы*.

Константы

Константами называют элементы данных, значения которых установлены в описательной части программы и в процессе выполнения программы не изменяются.

Константы могут использоваться в любых местах программы вместо реальных значений.

В зависимости от выбранного языка программирования варьируется и набор констант, которые могут быть определены в программе. Можно определить базовый набор констант, который существует в любом языке программирования, это:

− *числовые (целые и вещественные)* константы, например - 7.5, 12;

–*логические константы* – *true* (истина), *false* (ложь);

–*символьные константы* – одиночный символ, заключенный в одинарные кавычки. Например, *“A”, “\n”;*

– *именованные константы – константы, определенные пользователем с использованием зарезервированного слова CONST*, например,

const double E=2.72;

– *строчные константы – последовательности, состоящие из нескольких символов*, заключенных в двойные кавычки, например “*i am a student*”.

Переменные

Переменная *– это именованная область оперативной памяти, отведенная для временного хранения данных, которые могут изменяться при выполнении программы.*

Каждая переменная должна иметь имя, однозначно определяющее данную переменную внутри ее области определения (область, в которой доступна эта переменная). Переменная может иметь или не иметь указанный тип данных.

Переменная обладает следующими свойствами:

– переменная всегда хранит не более 1 значения;

– переменная способна хранить разные значения в процессе выполнения программы, но только одного и того же типа;

– переменная хранит значение до тех пор, пока в нее не поместят новое значение, при этом предыдущее содержимое переменной стирается;

– значение переменной может быть вызвано для использования сколько угодно раз без изменения оригинала.

Все переменные должны быть описаны до их использования в области описания программы.

К началу выполнения программы в некоторых языках программирования содержимое переменной считается неопределенным и ячейки памяти, отведенные под переменную путем ее описания, заполняются значениями в ходе выполнения программы с помощью операции присваивания; этим переменная отличается от константы, которой значение присваивается до выполнения основной программы.

В языке *VBA* в начале выполнения программы все переменные инициализируются. Числовая переменная получает значение 0, а строка получает значение «пустой строки» –“”.

Если значение переменной не изменяется во время выполнения программы, она сохраняет значение, полученное при инициализации, до тех пор, пока не потеряет *область определения*.

Типы данных

Тип данного *определяет множество значений, которые может принимать данное (константа или переменная).*

Каждый тип имеет свой ***размер, диапазон допустимых значений*** и ***специальное зарезервированное слово*** для описания.

Размер *– это количество байтов, которое занимает переменная данного типа в оперативной памяти.*

Диапазон *допустимых значений определяет наибольшее и наименьшее значение для переменных данного типа.*

В языке *VBA*, можно не указывать явно тип данного в программе, а просто объявить переменную в разделе описания, в этом случае, по умолчанию, ей присвоится универсальный тип данного (*variant*), предусмотренный в языке программирования. В этом случае переменная этого типа займет в оперативной памяти место, равное *16 байтам*.

В любом языке программирования присутствуют простые и структурированные типы данных. Все они различаются размером и диапазоном допустимых значений. В *табл. 1* описаны типы данных, поддерживаемые *VBA*.

В любом языке программирования присутствуют простые и структурированные типы данных. Все они различаются размером и диапазоном допустимых значений. В *табл. 1* описаны типы данных, поддерживаемые *VBA*.

Таблица 1. Типы данных *VBA*

Наименование	***Byte***	**Интервал допустимых значений**
Byte	1 байт	*От 0 до 255*
Boolean	2 байта	*True (истина) ИЛИ False (ложь)*
Currency	8 байт	*От -922337203685477.5808 до 922337203685477.5807*
Date	8 байт	*От 1 января 100 г. до 31 декабря 9999 г.*
Decimal	12 байт	*79228162514264337593543950335 без десятичной точки; либо +/-7.9 и 28 десятичных разрядов после точки -*
Double	8 байт	*От -922337203685477.5808 до 922337203685477.5807*
Integer	2 байта	*От -32768 до 32767*
Long	4 байта	*От -2147483648 до 2147483647*
Object	4 байта	*Ссылка на любой объект*
Single	4 байта	*От -3.402823E38 до -1.401298E-45 для отрицательных величин и от 1.401 298E-45 до 3.402823E38 для положительных величин*

Продолжение табл.1

String	произвольный	Длина строки — приблизительно до 2 млрд символов
Type	произвольный	Пользовательский тип данных
Variant	16 байт	Любое числовое или символьное значение

Дополнения к таблице

1) ***Логический тип*** определен таким образом, что *False< True*. Это позволяет применять к булевским операндам все операции отношения.

2) Для работы с ***символами*** в языке *VBA* используется функция *Chr(),* которая возвращает значение типа *String*, содержащее символ, соответствующий указанному коду символа.

Например,

Chr(65) – *возвращает символ A*

Chr(62) – *возвращает символ >*

Chr(97) – *возвращает символ a*

Chr(37) – *возвращает символ %*

3) ***Строковый тип данных – String*** используется для хранения:

– *строк* ***переменной*** *длины*, которые могут содержать до 2 млрд (2^31) символов. Объявление такой переменной в языке *VBA* имеет вид:

Dim name as string

– *строк* ***постоянной (фиксированной)*** *длины*, которые могут содержать от 1 до 64 Кбайт (2^16) символов. Операции с данным типом строк выполняются быстрее. Кроме того, с помощью строк фиксированной длины можно задать ограничение на количество символов в строке, например:

*Dim name as string*13*

4) ***Переменные типа Currency*** хранят числа с дробной частью до 4 цифр и целой частью до 15 цифр, что обеспечивает соответствующий диапазон представления чисел. Это точный тип данных с фиксированной точкой (*fixed-point data type*), удобный для денежных вычислений. Числа с плавающей десятичной точкой (*floating-point numbers – типы данных single и double*) имеют больший диапазон значений, но могут приводить к ошибкам округления.

5) ***Переменные типа Date*** являются числами с плавающей точкой двойной точности, и могут хранить как дату, так и время. Целая часть такого числа представляет дату, а дробная - время дня. В программном коде можно использовать литералы (буквальные константы) даты и времени. Они должны быть заключены между символами # точно так же, как строковые литералы заключаются между символами двойных кавычек – “”. Например:

Dim day as date

Day=#1-Jul-2014#

Day=#01.07.14#

Day=#1-7-2014 13:20#

В этом примере переменной ***Day*** разными способами передается значение даты "*1 июля 2014 года*".

Переменные типа *Object* сохраняются как 32-разрядные (4-байтовые) адреса, в которых содержатся ссылки на объекты.

Переменная типа Object или объектная переменная – это переменная, которой впоследствии можно присвоить (оператором Set) ссылку на любой объект, созданный в приложении.

Например:

```
Dim MyObject as Object
'присваивает ссылку на активный  файл базы данных
 Set MyObject = CurrentDB
'удаляет ссылку на объект
Set MyObject = Nothing
```

Задание для объектной переменной значения *Nothing* прекращает сопоставление этой переменной с каким-либо определенным объектом. Это предотвращает случайное изменение объекта при изменении переменной.

Общие правила записи выражений

Если выражение содержит несколько операторов, то значения компонентов выражения рассчитываются в определенном порядке. Такой порядок называют порядком *старшинства* или ***приоритетом операторов***.

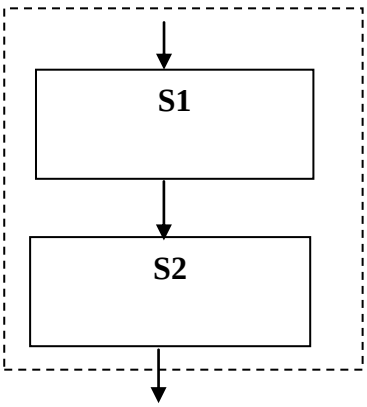
Если выражение содержит операторы разных типов, то первыми выполняются арифметические операции, следом за ними - операции сравнения, а последними - логические операции.

1. Все операторы сравнения имеют равный приоритет, т.е. выполняются в порядке их расположения в выражении слева направо.
2. Стоящие рядом в выражении операторы умножения и деления выполняются слева направо.
3. В таком же порядке выполняются стоящие рядом операторы сложения и вычитания.
4. Операторы внутри круглых скобок всегда выполняются раньше, чем операторы вне скобок.
5. Порядок выполнения операторов, стоящих внутри скобок, определяется старшинством операторов. Оператор конкатенации (слияния строк) не является арифметическим оператором, однако, по порядку старшинства он следует сразу за арифметическими операторами и перед операторами сравнения.

Лабораторная работа № 1. АЛГОРИТМ ЛИНЕЙНОЙ СТРУКТУРЫ

Алгоритм, в котором действия выполняются последовательно друг за другом, называется ***алгоритмом линейной структуры***. Такой порядок записи действий называют *естественным.*

Базовая структура, соответствующая линейному алгоритму, называется ***следованием.***



Структура *следование* означает, что два действия *S1* и *S2* должны быть выполнены последовательно одно за другим, т.е. управление передается от предыдущего функционального блока к следующему.

Алгоритмы линейной структуры применяются в большинстве расчетных задач как простых, так и сложных.

В качестве примера рассмотрим следующую задачу.

Задача. *Вычислить площадь треугольника со сторонами a, в, с по формуле Герона, которая имеет следующий вид:*

$S = \sqrt{p*(p-a)*(p-b)*(p-c)}$, где $p=(a+b+c)/2$.

Решение:

1. Вербальный алгоритм

1.1. Ввести три числа в переменные *a, в, с*;

1.2. Вычислить полупериметр треугольника по формуле

p=(a+b+c)/2;

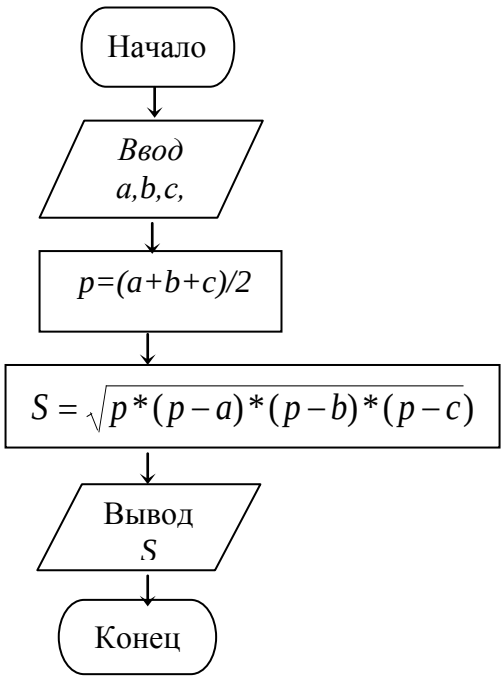
1.3. Вычислить площадь *S* по формуле

$S = \sqrt{p*(p-a)*(p-b)*(p-c)}$;

1.4. Вывести результат (значение *S*);

1.5. Закончить решение задачи.

2. Графический алгоритм



Программная реализация алгоритма

1. Начало работы

1.1. Запуск редактора VBA

Для того, чтобы начать работу в модуле, необходимо перейти на вкладку *Разработчик* на которой можно вызвать редактор *Visual Basic* и другие инструменты разработчика. Поскольку в *Office 2010* вкладка ***Разработчик*** не показана по умолчанию, необходимо вывести ее на экран, выполнив следующую процедуру.

1.1.1. На вкладке ***Файл*** выберите ***Параметры***, чтобы открыть диалоговое окно ***Параметры Excel***.

1.1.2. Щелкните ***Настройка ленты*** в левой части диалогового окна.

1.1.3. В разделе ***Выбрать команды из***, расположенном слева в окне, выберите ***Популярные команды***.

1.1.4. В разделе ***Настройка ленты***, который находится справа в диалоговом окне, выберите **Основные вкладки**, а затем установите флажок **Разработчик.**

1.1.5. Нажмите кнопку ***ОК***.

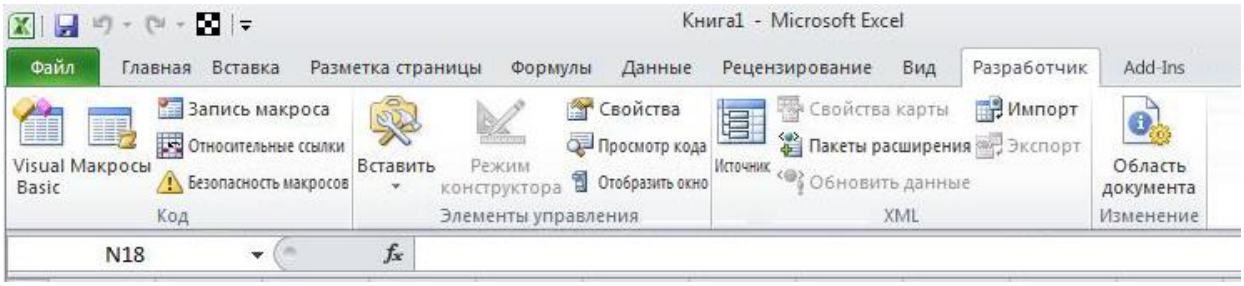


Рис.1. Вывод вкладки *Разработчик* на ленту

1.2. Нажмите кнопку *Visual Basic*

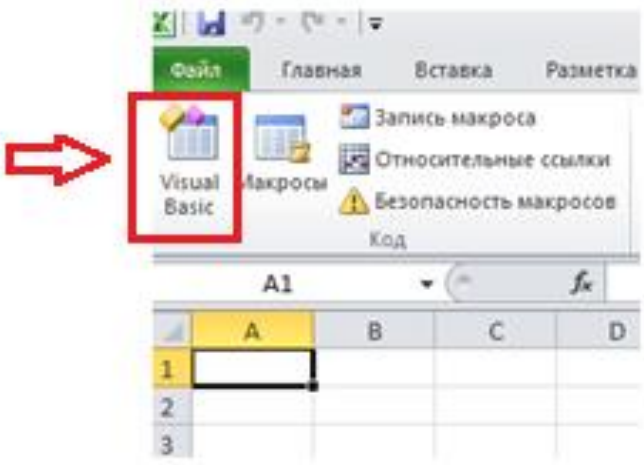


Рис. 2. Начало работы с редактором *VBA*

1.3. Запустится окно встроенного редактора *VBA* (*рис. 3*).

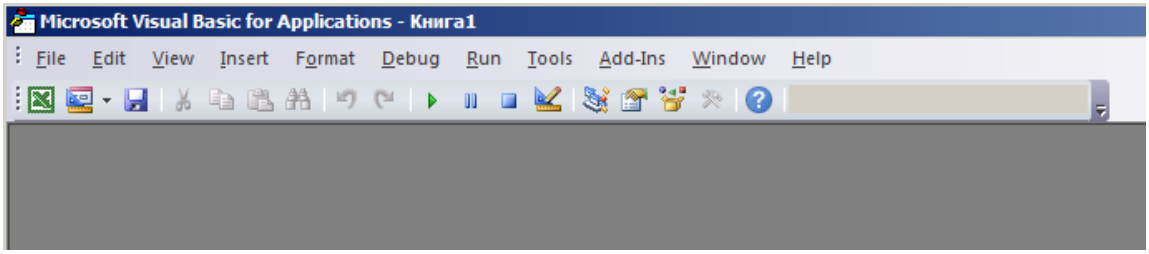


Рис. 3. Окно приложения *VBA*

Все программы приложения создаются в модулях – редакторах языка. Для того чтобы начать работу с модулем его необходимо создать, для этого – меню **Insert,** команда ***Module*** (*рис. 4*)/

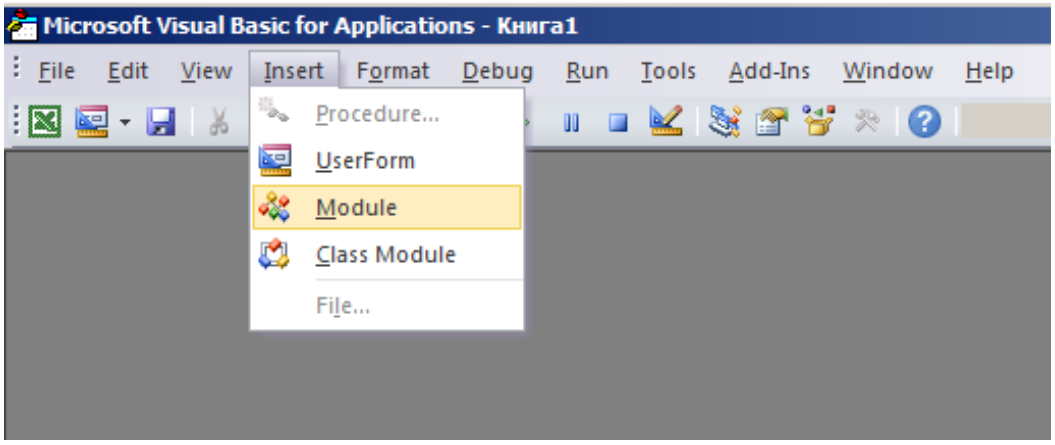


Рис. 4. Начало работы с редактором

1.4. После вставки модуля, нажмите кнопу ***Project Explorer*** (*рис. 5*), для того чтобы вывести слева окно проекта со всеми объектами файла рабочей книги *Excel.*

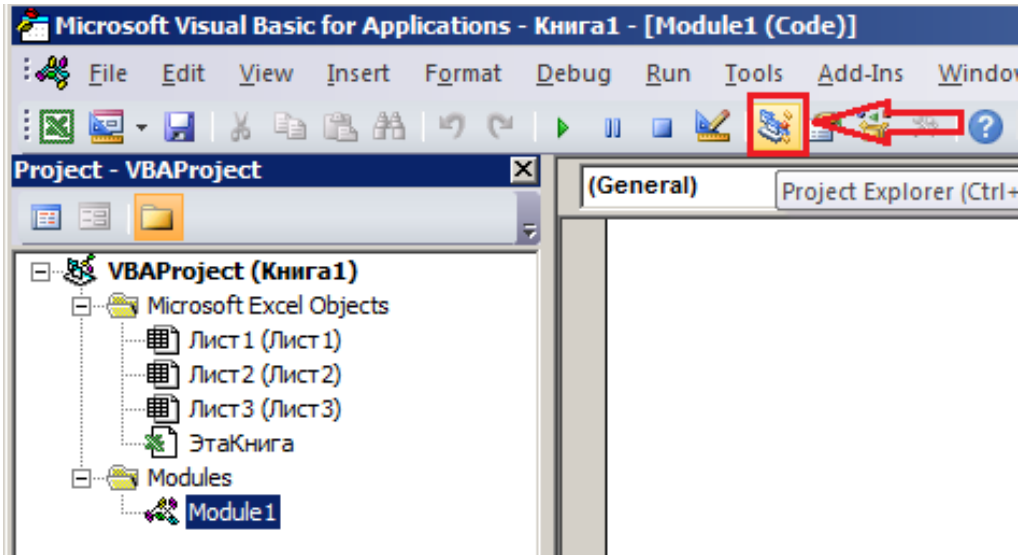


Рис. 5. Проводник *Project Explorer*

Модули можно создавать как для каждого рабочего листа, так и для всего проекта (файла рабочей книги) в целом. Будем работать в модуле проекта – *Module 1*.

Перед тем, как создать первую самостоятельную программу, в разделе описания модуля введите следующую инструкцию – *Option Explicit*. Эта инструкция предназначена для отслеживания используемых переменных в *программах-процедурах* или *программах-функциях*. Тело процедуры может содержать от одной и более инструкций, написанных в соответствии с правилами программирования на данном языке. Если пользователь использует переменные в программе, предварительно не описав их в разделе описания процедуры или функции, то появится следующее сообщение об ошибке (*рис. 6*).

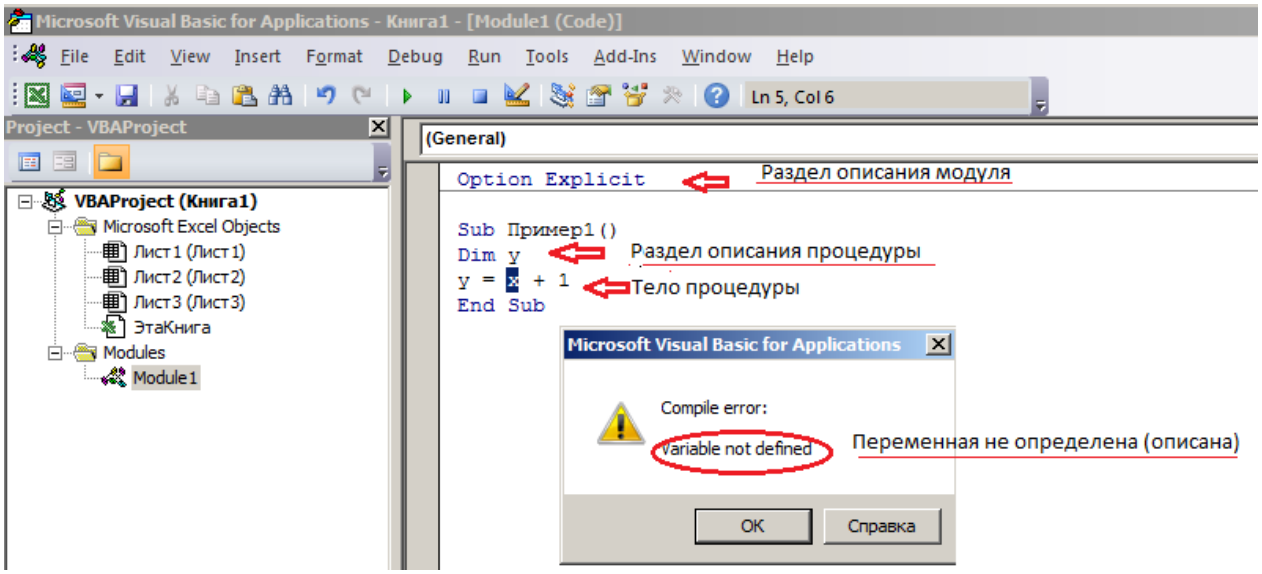


Рис. 6. Ошибка использования переменной в программе

Структуру простейшей программы можно определить как совокупность двух основных блоков или разделов – это *раздел описания переменных* и *исполнительный раздел* (иногда называемый *телом программы*), т.е. раздел, в котором непосредственно происходит выполнение основных операторов программы. Операторы располагаются в теле программы непосредственно друг под другом, причем на одной строке для лучшего понимания логики программы нужно располагать *не более одного оператора (инструкции)*.

Раздел описания переменных применяется для описания переменных, используемых в программе. Объявление переменной начинается со служебного слова *Dim*. Инструкция *Dim* указывает компилятору на необходимость отведения блока оперативной памяти для хранения значений переменной заданного типа.

Синтаксис:

Dim ИмяПеременной As ТипДанных

ИмяПеременной - это значимое название переменной, на которое можно ссылаться в программе.

ТипДанных - существующий тип данных, подходящий для размещения информации определенного рода.

Рассмотрим программную форму записи линейного алгоритма на примере решения следующей задачи: «Найти длину окружности по заданному радиусу».

Решение:

1. Для решения задачи используем процедуру, присвоив ей имя *ДлинаОкружности.*

2. В разделе описания введем две переменные *Длина* с типом данных *integer* и *Радиус* с типом данных *byte*.

3. Используем служебное зарезервированное слово Const для определения числовой константы *Pi*.

4. Для ввода значения в переменную *Радиус* используем функцию *InputBox*. Функция *InputBox* осуществляет следующие действия:

- выводит на экран диалоговое окно, содержащее заголовок, зону сообщения, поле ввода, значение по умолчанию ;
- устанавливает режим ожидания ввода текста пользователем или нажатия кнопки;
- возвращает значение типа *String*, содержащее текст, введенный в поле.

Формат записи функции:

Inputbox **(***сообщение[, заголовок окна][,поле ввода][,значение по умолчанию][,…])* , где

– *сообщение* – строковое выражение, отображаемое как текст сообщения в окне диалога.

– *заголовок окна* – строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот аргумент опущен, в строку заголовка помещается имя приложения.

– *поле ввода* – вводимое числовое или строковое выражение (строка знаков).

Замечание: аргументы, заключенные в квадратные скобки, являются необязательными.

Поскольку функция всегда возвращает значение в точку вызова, она используется только вместе с переменной. В этом случае строка программного кода будет выглядеть следующим образом:

Радиус=InputBox («Введите число, отличное от 0») (рис.7)

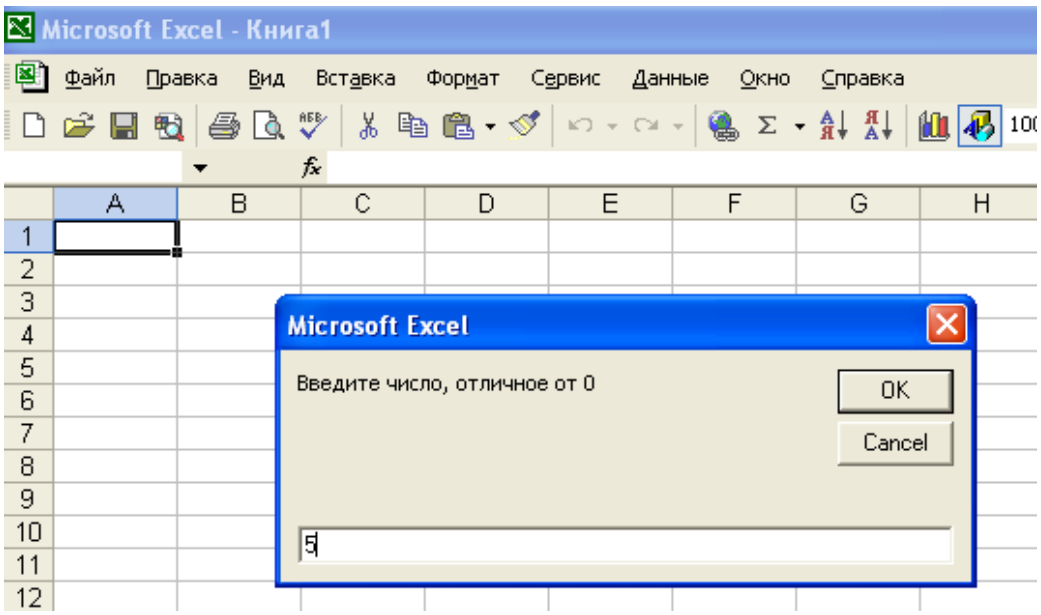


Рис. 7. Диалоговое окно функции *InputBox*

5. Далее, вычисляем длину окружности, так как все необходимые значения для вычисления получены –

Длина=2 Pi* Радиус*

Для вывода результата на экран, используем инструкцию *MsgBox* (может использоваться и как функция). Инструкция требует наличия обязательного текстового аргумента (*любой текст, заключенный в кавычки*). Поэтому заключительная инструкция будет выглядеть так –

MsgBox «Длина окружности =» & Длина

Знак & (оператор конкатенации (сцепления строк) здесь применяется для того, чтобы к выводимому тексту присоединить значение переменной *Длина*) (*Рис. 8).*

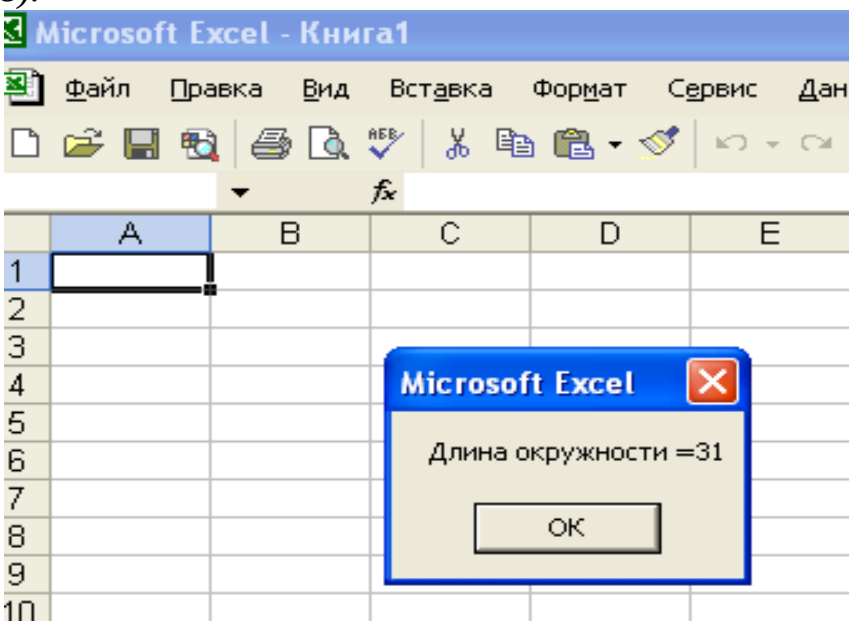


Рис. 8. Диалоговое окно инструкции *MsgBox*

Полностью текст программы будет выглядеть так (*рис. 9*):

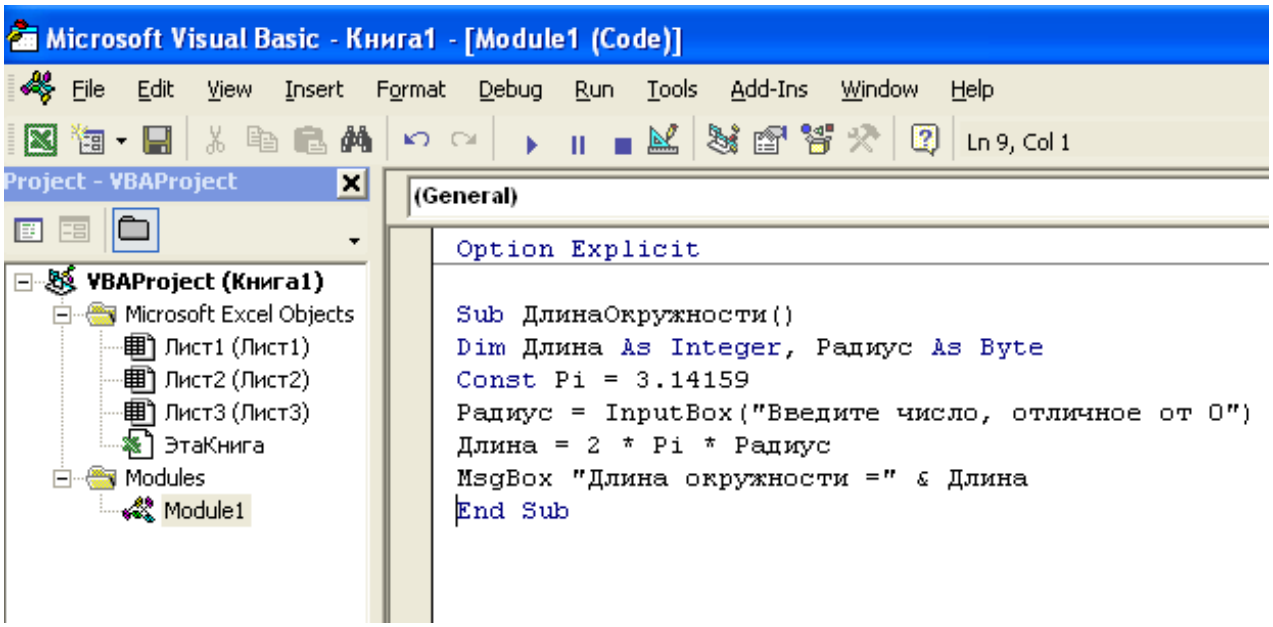


Рис. 9. Программная форма записи алгоритма для вычисления длины окружности

Программная форма записи решения задачи на вычисление площади треугольника со сторонами а, в, с по формуле Герона, будет выглядеть следующим образом:

sub geron()
dim a,b,c,p,S
a=val(inputbox("Введите a"))
b=val(inputbox("Введите b"))
c=val(inputbox("Введите c"))
p=(a+b+c)/2
S=(p(p-a)*(p-в)*(p-c))^(1/2)*
msgbox S
end sub

ЗАДАЧИ НА САМОСТОЯТЕЛЬНУЮ РАБОТУ

Решить линейным алгоритмом следующие задачи:

1. Задать длину ребра куба. Найти объем куба и площадь его поверхности.

2. Задать три действительных положительных числа. Найти среднее арифметическое и среднее геометрическое этих чисел.

3. Число А составляет 25% от числа В, которое в свою очередь составляет 18% от числа С, а С=369. Найти А и В.

4. Для изготовления заготовки ботинок рабочий тратит 2 мин. Сколько заготовок рабочий изготовит за 7 часов?

5. При производстве пряжи из каждого килограмма сырья получают 0,93 кг пряжи, отходы составляют 0,06 кг, потери – 0,01 кг. Сколько пряжи, отходов и потерь получим из 12 т сырья?

6. Сколько заготовок круглой формы можно получить из куска материи длиной 12 и шириной 1,4 м., если радиус заготовки =15 см? Определить площадь отходов. Центры заготовок должны располагаться на одной линии.

7. Вычислите расстояние между двумя точками с координатами X_1, Y_1 и X_2, Y_2.

8. Сколько процентов от (А + В – С) приходится на А, В и С?

9. Производительность формовочной машины – 7 тарелок в 1 мин. Сколько тарелок выпустят три машины за 6 час?

Лабораторная работа № 2-3
АЛГОРИТМ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ

Алгоритм, в котором вычислительный процесс содержит проверку условия, от выполнения которого выбирается одно из двух возможных направлений, называется ***алгоритмом разветвляющейся структуры.***

Базовая структура, соответствующая этому типу алгоритма, называется ***ветвление.*** Структура начинается с блока проверки условия и заканчивается пересечением дуг. В зависимости от результата проверки условия обеспечивается выбор одного из альтернативных путей работы алгоритма.

Структура *ветвление* существует в четырех основных вариантах:

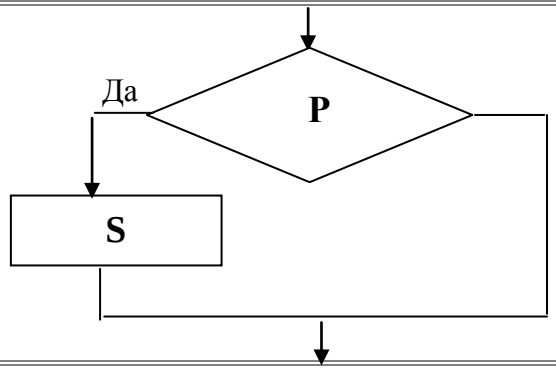
1). ***ЕСЛИ-ТО;***
2). ***ЕСЛИ-ТО-ИНАЧЕ;***
3). ***ВЫБОР;***
4). ***ВЫБОР-ИНАЧЕ***.

ПОЛНАЯ И НЕПОЛНАЯ РАЗВИЛКИ

Структура **ЕСЛИ-ТО (*неполная развилка*)**

Если условие **P** истинно, то выполняется действие **S**, иначе выполняется оператор, следующий после структуры.

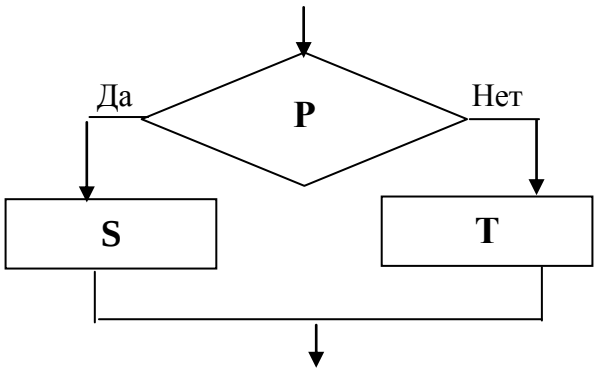
Краткая запись: если **P**, то **S**.



Структура **ЕСЛИ-ТО-ИНАЧЕ (*полная развилка*)**

Если условие **P** истинно, то выполняется действие **S**, иначе выполняется действие **T**.

Краткая запись: если **P**, то **S**, иначе **T**.



В качестве примера рассмотрим решение следующей задачи.

Задача. *Найти максимум из двух чисел, используя для решения базовые структуры полной и неполной развилок.*

Решение:

1.1. Вербальный алгоритм методом полной развилки

1.1.1. Ввести два числа в переменные ***a*** и ***b***;

1.1.2. Задать буферную (промежуточную) переменную ***max*** для хранения максимального числа;

1.1.3. Сравнить значения переменных ***a*** и ***b***. Если ***a*** больше ***b***, то присвоить переменной ***max*** значение переменной ***a***, иначе значение переменной ***b***;

1.1.4. Вывести максимальное число – значение переменной ***max***;

1.1.5. Закончить решение задачи.

1.2. Вербальный алгоритм методом неполной развилки

1.2.1. Ввести два числа в переменные ***a*** и ***b***;

1.2.2. Задать буферную (промежуточную) переменную ***max*** для хранения максимального числа;

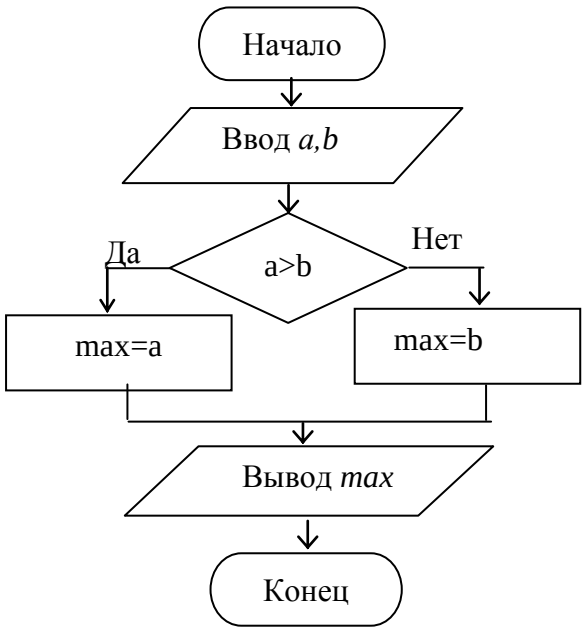
1.2.3. Допустить, что значение переменной ***a*** является максимальным числом и сохранить это значение в буферной переменной ***max***;

1.2.4. Сравнить значение переменной ***b*** с максимумом – значением переменной ***max***; если ***b*** больше максимума, то заменить прежнее значение переменной ***max*** на новое – значение переменной ***b***;

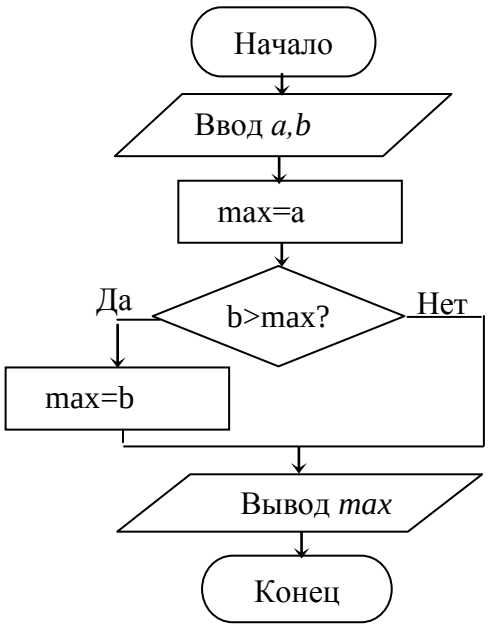
1.2.5. Вывести максимальное число – значение переменной ***max***;

1.2.6. Закончить решение задачи.

2.1. Графический алгоритм методом полной развилки



2.2. Графический алгоритм методом неполной развилки



Программная реализация алгоритма

Оператор условия в языке *VBA* используется для описания алгоритмов полной и неполной развилок и имеет несколько вариантов синтаксиса. В общем случае оператор условия имеет вид

If условие ***Then*** оператор1 ***Else*** оператор2

Параметр ***условие*** – логическое выражение, которое возвращает одно из двух возможных значений:

True – если условие верно (истинно);

False – если условие не выполняется (ложно).

Если условие истинно, то выполняется оператор или группа операторов, находящиеся после ключевого слова *Then*, а операторы, стоящие после ключевого слова *Else,* игнорируются. И наоборот, если условие ложно, то выполняются операторы, находящиеся после ключевого слова *Else*, а остальные операторы игнорируются.

Логическое выражение, используемое в качестве условия, может быть простым и сложным. При записи простых условий могут использоваться все возможные операции отношения. Сложные условия образуются из простых путем применения логических операций (***and, or, not***) и круглых скобок.

Для описания алгоритма неполной развилки (*если условие **P** истинно, то выполняется действие **S***) применяется следующий синтаксис:

If условие ***Then*** оператор

В этом варианте синтаксиса, называемом ***однострочным***, ключевое слово ***Else*** отсутствует. Если логическое условие *ложно*, то оператор, находящийся после ключевого слова ***Then*** не выполнится, и управление будет передано следующей строке программы.

Рассмотрим применение этого синтаксиса при решении следующей задачи.

Задача. *Пусть даны два числа. Если первое больше второго по абсолютной величине, то необходимо уменьшить первое в пять раз. Иначе оставить числа без изменений.*

Решение (язык *VBA*):

```
Sub primer()
Dim x,y
x=InputBox("x=")
y=InputBox("y=")
  If abs(x) > abs(y) Then x=x/5
MsgBox x
End Sub
```

В условном операторе допустимо использование блока операторов вместо одного. В этом случае *однострочный синтаксис неполной развилки* заменяется на ***многострочный*** и имеет следующий вид:

If *условие* ***Then***
Инструкция 1
Инструкция 2
. . . . . .
Инструкция N
End If

Ключевое слово ***Else*** также отсутствует, но появляется инструкция ***end if,*** которая завершает оператор условия. Синтаксические правила при описании неполной развилки достаточно строги, и, если, допустим, опустить завершающую инструкцию при применении многострочного синтаксиса, появится следующая ошибка, приведенная на *рис. 10.*

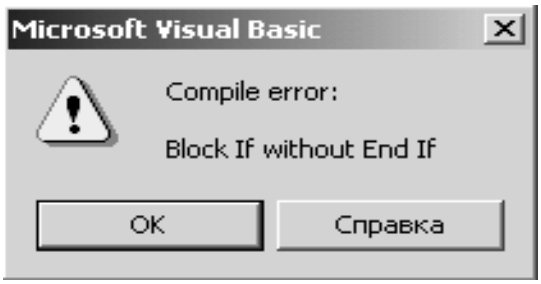


Рис. 10. Синтаксическая ошибка при записи оператора условия

Рассмотрим решение задачи о максимуме из двух чисел, объявленной выше, программным алгоритмом неполной развилки, используя синтаксис:

If условие ***Then*** оператор

Решение.

```
Option Explicit
Sub max_3()
Dim a, b, max
a = Val(InputBox("Введите A"))
b = Val(InputBox("Введите  B"))
max = a
If b > max Then max = b
MsgBox max
End Sub
```

Программная реализация *алгоритма* ***полной развилки*** тоже имеет два вида синтаксиса – *однострочный и многострочный.*

1. Однострочный синтаксис имеет вид:

If *условие* ***Then*** *инструкция1* ***Else*** *инструкция2*

2) ***многострочный синтаксис*** (применяется в основном тогда, когда после ключевых слов *Then* и *Else* используется более одной инструкции):

If условие ***Then***
Инструкция 1
Инструкция 1n
Else
Инструкция 2
Инструкция 2n
End if

При использовании этого вида синтаксиса решение задачи о максимуме из двух чисел будет выглядеть следующим образом:

Программа (полная развилка) (*однострочный синтаксис*)	**Программа (полная развилка)** *(многострочный синтаксис)*
.... *if a>b then max= a else max=b* *msgbox "максимум=" & max*	.... *if a>b then* *max= a* *else* *max=b* *end if* *msgbox "максимум=" & max* ....

Имеется расширенный вариант многострочного синтаксиса, который применяется тогда, когда нужно проверить на *истинность* выполнение нескольких условий. В этом случай многострочный синтаксис принимает следующую форму:

If *условие1* ***then***
Блок операторов1
Elseif *условие2* ***then***
Блок операторов2
. . . . . .
Elseif *условиеN* ***then***
Блок операторовN
Else
Блок операторов N+1
End if

Применяя этот синтаксис оператора условия, нельзя располагать блок операторов на той же строке, что и ключевое слово ***Then***, так как возникнет синтаксическая ошибка на стадии компиляции программы (*рис. 11*).

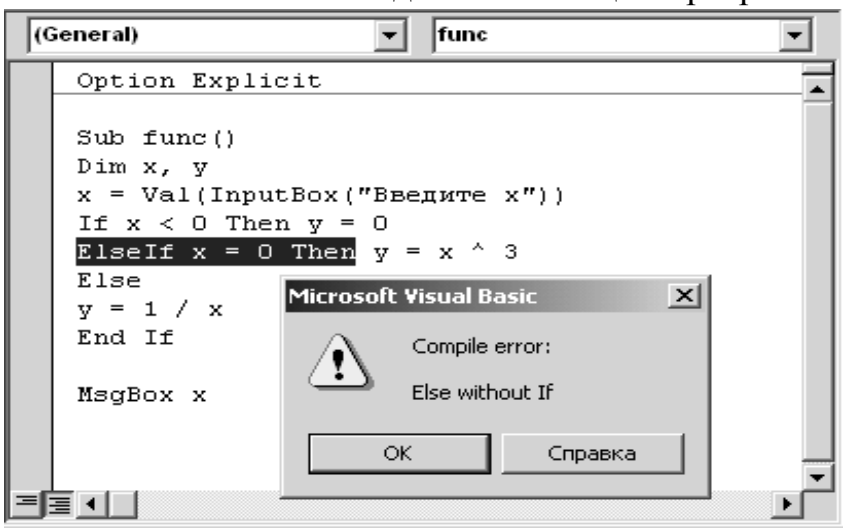


Рис. 11. Ошибка компиляции программы при неправильном написании многострочного синтаксиса оператора условия

При выполнении блока ***If*** проверяется первое условие. Если условие выполняется (*условие=истина)*, тогда выполняются инструкции, следующие за ключевым словом ***Then***. Если условие1 не выполняется (имеет значение ***false*** *(условие=ложь))*, то по очереди оценивается каждое условие ***ElseIf***. При обнаружении условия со значением ***true*** *(условие=истина)* выполняются инструкции, непосредственно следующие за соответствующим предложением ***Then***. Если ни одно из условий ***ElseIf*** не выполняется (или предложения ***ElseIf*** отсутствуют), тогда выполняются инструкции, следующие за инструкцией ***Else***.

После выполнения операторов, следующих за ключевыми словами ***Then*** или ***Else***, выполнение программы продолжается с инструкции, следующей за инструкцией ***End If***.

ЗАДАЧИ НА САМОСТОЯТЕЛЬНУЮ РАБОТУ

1. Даны два числа A и B. Если A>B, то A присвоить значение B, а B - значение A. Вывести прежние и полученные значения A и B.

2. Даны две пары чисел A, B и C, Д. Если A*B>C*Д, то вывести среднее арифметическое этих чисел, в противном случае среднее геометрическое.

3. Даны три числа: a, b, c. Определить, можно ли построить треугольник, если интерпретировать a, b, c как длины сторон треугольника

4. Найти максимумы из 3, 4, 5 чисел. Числа для сравнения вводить как с клавиатуры, так и с помощью функции RND (*функция для получения случайного числа*).

Функция RND возвращает случайное число в диапазоне от 0 до 1 с точностью 12 знаков после запятой. Для того, чтобы комфортно можно было использовать случайные числа в программах, следует умножать функцию на 10, 100, 1000 и т.д. Для отбрасывания дробной части используйте функции INT или FIX.

*Например: x=Int(Rnd*100)*

5. Даны три числа A, B, C. Найти и вывести наименьшую по абсолютной величине разность из трех возможных.

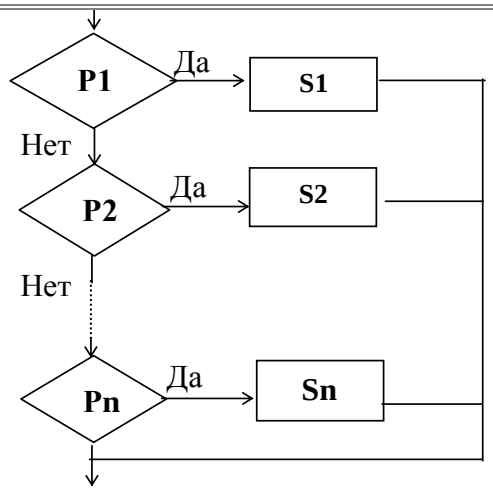
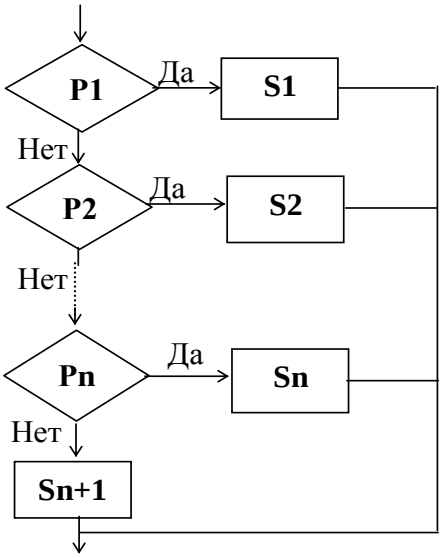
6. Написать программу, которая вычисляет по требованию периметр и площадь одной из шести фигур (треугольник, прямоугольник, квадрат, круг, трапеция, ромб).

7. Предлагаются две, три, четыре, пять точек. Написать программу, которая по количеству точек определяет возможные фигуры, которые получаются при их соединении. Для решения задачи использовать случайное число и функцию MOD (*остаток от деления*).

8. Определить высоту треугольника, если его площадь равна S, а основание больше высоты на величину A.

Базовая структура ВЫБОР

Базовая алгоритмическая структура существует в двух вариантах: структура ***выбор*** и структура ***выбор-иначе***. Последняя подразумевает альтернативное действие, если выбор по условию не совпадает ни с одним из предложенных действий.

<table>
<tr><td colspan="2">Структура ВЫБОР</td></tr>
<tr><td>Если истинно условие P1, то выполняется действие S1;
Если истинно условие P2, то выполняется действие S2;
……
Если истинно условие Pn, то выполняется действие Sn;
Если не выполняется ни одно из условий, управление передается оператору, следующему за структурой.</td><td> 
</td></tr>
<tr><td colspan="2">Структура ВЫБОР-ИНАЧЕ</td></tr>
<tr><td>Если истинно условие P1, то выполняется действие S1;
Если истинно условие P2, то выполняется действие S2;
……
Если истинно условие Pn, то выполняется действие Sn;
Если не выполняется ни одно из условий, то выполняется Sn+1 действие.</td><td> 
</td></tr>
</table>

Программная реализация алгоритма

Оператор выбора служит альтернативой многострочному синтаксису условной инструкции *If..Then ..Else* при оценке одного условия, которое может иметь несколько возможных значений. В то время как в условном операторе для каждой инструкции *ElseIf* оцениваются разные выражения, оператор

выбора оценивает выражение только один раз, в начале управляющей структуры.

Синтаксис оператора выбора:

Select case *ПроверяемоеВыражение*

Case *список выражений 1*

Блок операторов1

Case *список выражений 2*

Блок операторов2

....

Case *список выражений N*

Блок операторовN

Case else

Альтернативная инструкция

End Select

ПроверяемоеВыражение – любое строковое или числовое выражение. Вместо выполнения логического условия (как в случае с условной инструкцией *If*) выполняется сравнение значений проверяемого выражения с каждой из величин, заданных параметром *список выражений 1..N*, находящихся после ключевого слова ***Case***. Если значение проверяемого выражения совпадает с одним из этих значений, то управление передается на соответствующую инструкцию ***Case***. В этом случае выполняются все инструкции, находящиеся в этом блоке.

Список выражений, используемый для сравнения в случае числового выражения, может иметь одну из следующих форм синтаксиса:

Case 1,3 (либо значение 1, либо значение 3)

Case 5 ***to*** 10 (список значений от 5 до 10)

Case 12 ***is*** >=15 (или значение 12, или все значения большие либо равные 15.

В инструкции может быть любое число проверяемых блоков. Если ни одна из величин, заданная параметром «список выражений» не является истиной для сравниваемого проверяемого выражения, то выполняется альтернативное выражение инструкции Case Else (как правило, это сообщение пользователю о допущенной ошибке ввода).

Если эта инструкция отсутствует, то выполняется следующий оператор, находящийся после ключевой инструкции ***End Select***.

В качестве иллюстрации, рассмотрим действие оператора выбора ***Select ..Case*** на примере программы (*рис. 12*), которая предлагает пользователю ввести код города и, в зависимости от введенного значения, пользователю

выдается либо название города (если код совпадает с одним из значений, указанных в инструкции ***Case***), либо сообщение об ошибке, если код города набран неправильно.

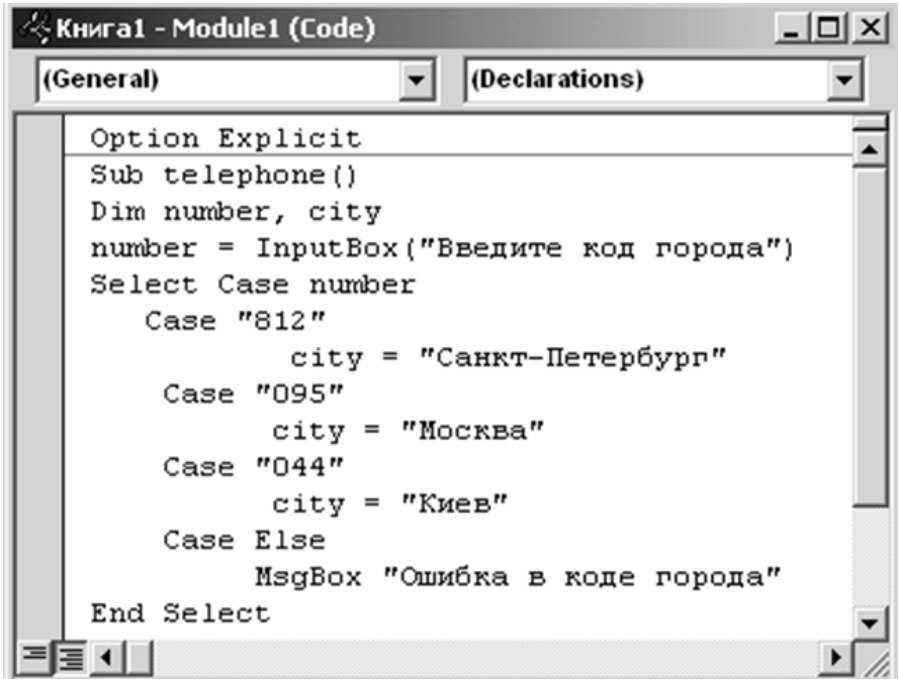


Рис. 12. Демонстрация программы на языке *VBA*

ЗАДАНИЯ НА САМОСТОЯТЕЛЬНУЮ РАБОТУ

1. Решить следующие уравнения, используя многострочный синтаксис оператора выбора

1	$x \in [0;1]$ $\begin{cases} 1+\ln(1+x), & x<0.2 \\ \dfrac{1+x^{\frac{1}{2}}}{1+x}, & x\in[0.2;0.8] \\ 2e^{-2x}, & x>0.8 \end{cases}$	2	$x \in [-2;2]$ $\begin{cases} \dfrac{1+\lvert x\rvert}{\sqrt[3]{1+x+x^2}}, & x\le -1 \\ 2\ln(1+x^2)+\dfrac{1+\cos^4(x)}{2+x}, & x\in(-1;0) \\ (1+x)^{\frac{3}{5}}, & x\ge 0 \end{cases}$
3	$x \in [-2;2]$ $\begin{cases} \dfrac{1+x}{\sqrt[3]{1+x^2}}, & x\le 0 \\ -x+2e^{-2x}, & x\in(0;1) \\ \lvert 2-x\rvert^{\frac{1}{3}}, & x\ge 1 \end{cases}$	4	$x \in [-2;2]$ $\begin{cases} 3x+\sqrt{1+x^2}, & x<0 \\ 2\cos(x)e^{-2x}, & x\in[0;1] \\ 2\sin(3x), & x>1 \end{cases}$

5	$x \in [-1.7;1.5]$ $\begin{cases} \frac{1+x+x^2}{1+x^3}, & x<0 \\ \sqrt{1+\frac{2x}{1+x^2}}, & x \in [0;1[\\ 2\lvert 0.5+\sin(x)\rvert, & x \geq 1 \end{cases}$	6	$x \in [-1.5;1.8]$ $\begin{cases} 1+\frac{3+x}{1+x^2}, & x<0 \\ \sqrt{1+(1-x)^2}, & x \in [0;1[\\ \frac{1+x}{1+\cos^2(x)}, & x \geq 1 \end{cases}$
7	$x \in [-1.4;1.9]$ $\begin{cases} \frac{1+2x}{1+x^2}, & x<0 \\ \sin^2(x)\sqrt{1+x}, & x \in [0;1[\\ \sin^2(x)e^{0.2x}, & x \geq 1 \end{cases}$	8	$x \in [-1.4;1.4]$ $\begin{cases} \frac{\lvert x\rvert}{1+x}e^{-2x}, & x<0 \\ \sqrt{1+x}, & x \in [0;1[\\ \frac{1+\sin(x)}{1+x}+3x, & x \geq 1 \end{cases}$

2. Написать программу, которая будет выводить слово, обозначающее разрядность числа, например, двузначное, трехзначное и т.д.(использовать не менее 5 проверок).

3. Решить задачи 6 и 7 (*лаб.работа №2*) многострочным синтаксисом.

4. Определить кратность числа (*использовать оператор MOD*). Следует предусмотреть, что число может быть кратно нескольким цифрам, для этого следует накопить все варианты кратности в строковой переменной. Вывести на экран полный результат – само число и варианты его кратности.

5. Написать программу, которая вычисляет по требованию периметр и площадь одной из шести фигур (треугольник, прямоугольник, квадрат, круг, трапеция, ромб).

6. Написать программу определения сотовых операторов.

Лабораторная работа № 4.
АЛГОРИТМ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

Часто при решении задач приходится многократно вычислять значения по одним и тем же зависимостям.

Многократно вычисляемые участки вычислительного процесса называются ***циклами.***

Алгоритм, содержащий цикл, называют циклическим.

Различают циклы с заданным и незаданным числом повторений.

Алгоритм с заданным числом повторений – цикл со счетчиком.

Алгоритм с неизвестным числом повторений – цикл с предусловием, цикл с постусловием.

Циклические процессы описывает базовая структура ***повторение*** или иначе***, цикл.***

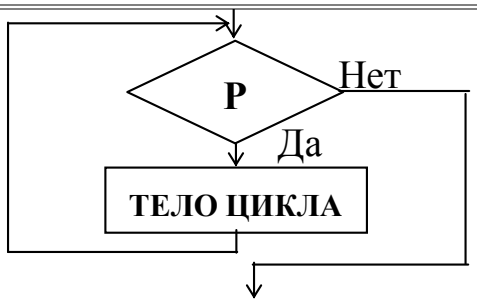
Алгоритм цикла с предусловием

Совокупность команд, называемая ***телом цикла,*** будет выполняться до тех пор, ***пока выполняется условие*** (*условие=истина* (*true*)).

Когда условие становится ложным(*false*) цикл завершает работу и выполняется следующая за циклом команда.

Следует отметить, что поскольку число повторений тела цикла заранее не определено, и, если при первом входе в цикл условие уже не выполняется, то тело цикла не будет выполнено ни разу.

Базовая структура ***цикл ПОКА.***

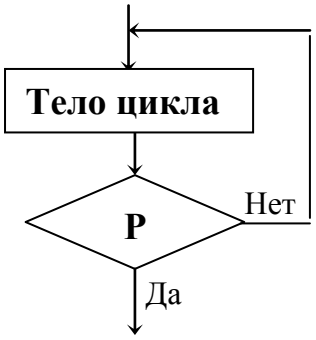
Если условие **Р** истинно, то выполняется **тело цикла**, иначе выполняется оператор, следующий за циклом. Краткая запись: если **Р** истинно, выполнять **тело цикла.**	

Алгоритм цикла с постусловием

Тело цикла будет выполняться до тех пор, ***пока не будет выполнено условие*** (*условие=ложь* (*false*)).

В случае выполнения условия цикл завершит работу и выполнится следующая за циклом команда. Так как число выполнения тела цикла заранее не определено и если при первом входе в цикл условие уже выполняется, то тело цикла будет выполнено хотя бы один раз.

Базовая структура ***цикл ДО.***

Если **Р** ложно, то выполняется **тело цикла,** иначе выполняется оператор, следующий после структуры. Краткая запись: если **Р** ложно, выполнять **тело цикла.**	

Цикл с постусловием отличается от *цикла с предусловием* тем, что, во-первых, тело цикла выполняется *пока условие ложно* и, во-вторых, независимо от того, истинно условие или ложно, *тело цикла будет выполнено хотя бы один раз.*

Схема выполнения такого цикла следующая:

1) вычисляется выражение (простое с применением операторов отношения либо сложное с применением логических операторов);

2) если выражение истинно (true), то выполнение оператора цикла заканчивается и выполняется следующая за оператором инструкция. Если выражение ложно (false), то выполняется тело оператора цикла;

3) процесс повторяется с пункта 1.

В качестве примера рассмотрим решение следующих задач.

Задача. *Найти сумму N натуральных чисел.*

Графическая интерпретация решения задачи представлена на *рис.13*:

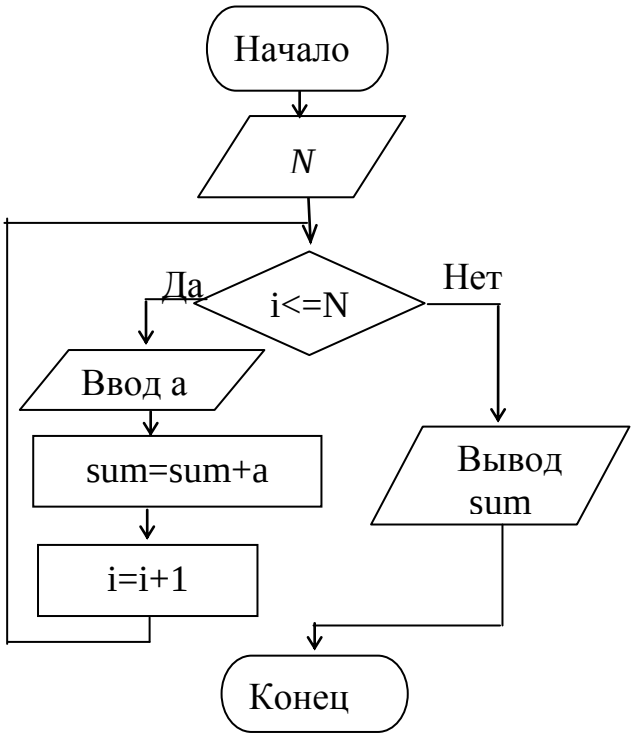


Рис.13. Графический алгоритм нахождения суммы циклом ***ПОКА***

Задача. *Найти максимум из N натуральных чисел.*

Графическая интерпретация решения задачи представлена на *рис.14*:

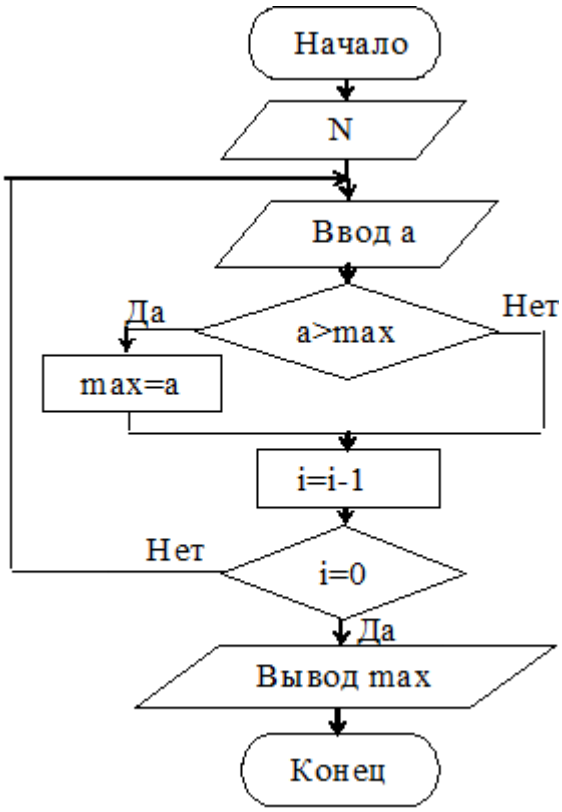


Рис.14. Графический алгоритм нахождения максимума из N натуральных чисел циклом ***ДО***

Программная реализация алгоритма

Цикл ***DO..LOOP*** может использоваться в двух формах. Первая из них предполагает проверку условия в ***начале цикла***, а другая — ***в конце*** него.

Циклы, условие повторения которых вычисляется каждый раз в начале цикла, называются ***циклами с предусловием.***

Схема выполнения такого цикла следующая:

1) вычисляется выражение (простое с применением операторов отношения либо сложное с применением логических операторов);

2) если выражение ложно (*false*), то выполнение оператора цикла заканчивается и выполняется следующая за оператором инструкция. Если выражение истинно (t*rue*), то выполняется тело оператора цикла;

3) процесс повторяется с пункта 1.

Цикл с постусловием аналогичен циклу с предусловием, только тело цикла будет выполнено хотя бы один раз, пока не произойдет проверка условия.

Вариант 1 **(Цикл с предусловием)**	**Вариант 2** **(Цикл с постусловием)**
Do while|Until условие *Инструкции|* *[Exit Do]* *Loop*	*Do* *Инструкции* *[Exit Do]* *Loop while|Until условие*

Ключевые слова ***While*** и ***Until*** по-разному обеспечивают реализацию проверки условия и выхода из цикла.

Ключевое слово ***While*** применяется, когда необходимо проверить на истинность условие, задаваемое в цикле. Другими словами, цикл выполняется ***ПОКА (While)*** условие истинно.

Ключевое слово ***Until*** применятся тогда, когда необходимо обеспечить выполнение цикла ***ДО ТЕХ ПОР ПОКА (Until)*** условие ложно.

Логическое выражение, используемое в качестве ***условия***, задает числовое или строковое выражение, при вычислении которого должно получиться значение либо "*true*", либо "*false*".

Оператор ***Exit Do*** обеспечивает досрочный выход из цикла, независимо от условия продолжения инструкций. Квадратные скобки, внутрь которых помещается инструкция, указывают на необязательность параметра.

Программная реализация вышеуказанных задач выглядит следующим образом:

Сумма N натуральных чисел *(цикл с предусловием)*	**Максимум из N натуральных чисел** *(цикл с постусловием)*
Sub summa () *Dim N, a,sum* *N=InputBox("N=")* ***Do while*** *i<N* *a=val(inputbox("Введите a"))* *sum=sum+a* *i=i+1* ***loop*** *msgbox sum* *end sub*	*Sub summa ()* *Dim i as byte, a, max* *i=inputbox(" i=")* ***Do*** *a=inputbox("Введите a")* *if a>max then max=a* *i=i -1* ***loop until*** *i=0* *msgbox max* *end sub*

Предотвращение бесконечного цикла

При работе с циклами имеется опасность входа в бесконечный цикл, если значение заданного условия не может измениться. В этом случае цикл выполняется бесконечно или до прерывания пользователем работы программы (клавиши <*Ctrl+Break*>). Например:

```
....
counter=99
Do until counter = 0
  MsgBox («Мы в цикле!»)
 counter = counter -2
loop
```

После 49 повторений значение счетчика counter станет равным 1, а затем, после следующего повторения, его значение станет равным –1. Нулевое значение пропущено и условие никогда не будет истинным.

ЗАДАНИЯ НА САМОСТОЯТЕЛЬНУЮ РАБОТУ

Решение задач представить всеми видами циклов.

1. Найти сумму ряда $1+1/2^2 + 1/3^3 ...1/n^n$.
2. Вычислить факториал числа N.
3. Задать случайным числом N (N>10) конечное число членов последовательности целых натуральных двухзначных чисел (*случайных*). Если сумма чисел кратных трем будет больше суммы чисел, кратных 4, то вычислить среднее арифметическое этих сумм, иначе – среднее геометрическое.
4. Найти максимум и минимум из N натуральных двухзначных случайных чисел. Вывести на экран наколенные числа ряда, максимальное и минимальное число.
5. Дан ряд из N натуральных двухзначных случайных чисел. Найти среднее арифметическое всех элементов ряда и наибольшее отклонение от среднего.
6. Дан ряд из N натуральных двухзначных случайных чисел. Если количество четных элементов ряда больше количества нечетных, то увеличить среднее арифметическое всех элементов ряда в 5раз, иначе, вычислить из него корень пятой степени.

Лабораторная работа № 5
АЛГОРИТМ ЦИКЛА СО СЧЕТЧИКОМ

Является частным случаем цикла с предусловием. Отличие состоит в том, что в цикле задаются границы диапазона, по которым определяется число повторений цикла. Для реализации алгоритма используется блок ***Модификация*** *(рис. 15).*

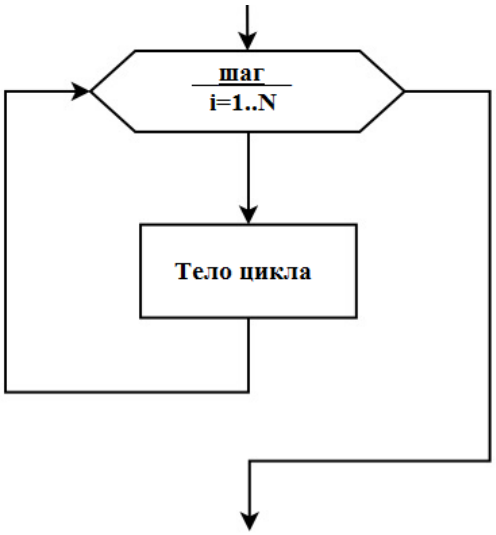


Рис. 15. Базовая алгоритмическая структура цикла со счетчиком

Тело цикла будет выполняться от начального значения, заданного счетчиком, до конечного.

Программная реализация алгоритма

В случаях, когда число повторений заранее известно, для организации циклической обработки информации применяется оператор повтора For (в обоих языках программирования). Часто этот оператор называют оператором цикла с параметром, так как число повторений задается переменной, называемой параметром цикла или управляющей переменной цикла.

Схема выполнения оператора повтора следующая:

1) задаются нижняя и верхняя границы изменения параметра цикла;
2) тело цикла будет выполняться до тех пор, пока не будут перебраны все значения параметра цикла от начального до конечного;
3) после этого управление передается на оператор, следующий за циклом.

Синтаксис языка VBA

For *счетчик* = *начало* ***To*** *конец* [***Step*** шаг]
  Инструкции
  [***Exit For***]
  Инструкции
Next [счетчик]

Счетчик – обязательный параметр цикла; числовая переменная, используемая в качестве счетчика, может быть только целого типа.

Начало *и* ***конец*** – обязательные параметры цикла; определяют начальное и конечное значения переменной *счетчик*.

Шаг – необязательный параметр; значение, на которое может изменяться счетчик при каждом выполнении тела цикла. Если это значение не задано, то по умолчанию значение шага принимается равным 1.

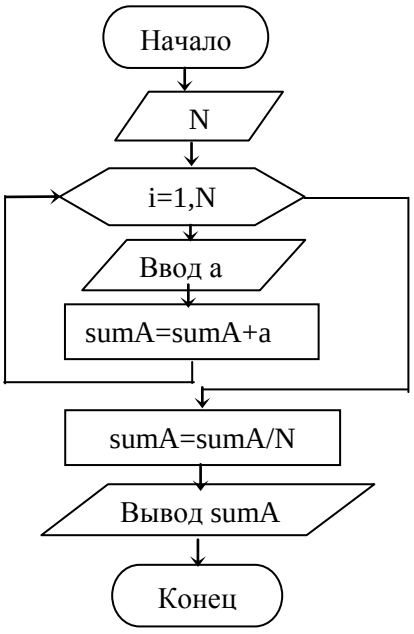
Exit For – альтернативный выход из цикла. При его использовании выполнение тела цикла немедленно прекращается, и управление передается оператору, находящемуся после зарезервированного слова ***Next.***

В качестве примера рассмотрим решение следующей задачи.

Задача*. Найти среднее арифметическое из N введенных чисел.*

Решение:

Графический алгоритм



Программа (*фрагмент*)

```
....
N=val(inputbox("Введите N"))
For i=1 to N
    a=val(inputbox("Введите a"))
    sumA=sumA+a
next i
    sumA=sumA/N
    msgbox sumA
    ....
```

ВЛОЖЕННЫЕ ОПЕРАТОРЫ ЦИКЛА

Если телом цикла является циклическая структура, то такие циклы называют ***вложенными***. Цикл, содержащий в себе другой цикл, называют ***внешним***, а цикл, содержащийся в теле другого цикла, - ***внутренним***. При программировании вложенных циклов необходимо соблюдать следующее дополнительное условие: *все операторы внутреннего цикла должны полностью располагаться в теле внешнего цикла (рис. 15).*

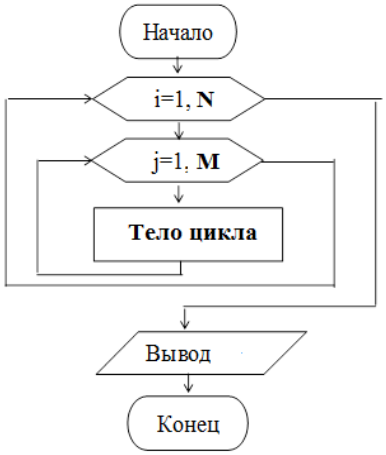


Рис. 15. Вложенный цикл

Глубина вложения циклов (количество вложенных друг в друга циклов) может быть различной.

В качестве иллюстрации рассмотрим решение следующей задачи.

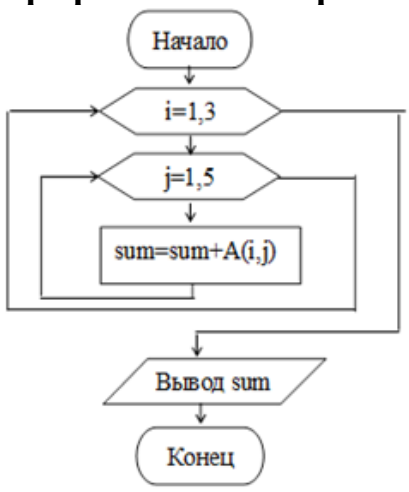
Задача. *Вычислить сумму элементов заданной матрицы* ***A(3,5).***

Матрица А

	1	**2**	**3**	**4**	**5**
1	10	15	0	3	99
2	1	7	82	15	0
3	17	98	1	0	5

Решение:

Графический алгоритм суммы



Программа (*фрагмент*)

```
....
For i=1 to 3
  For j=1 to 5
   sum=sum+A(i,j)
  next j
next i
....
```

ЗАДАНИЯ НА САМОСТОЯТЕЛЬНУЮ РАБОТУ

1. Вывести в диалоговое окно таблицу умножения.

2. Заполнить двумерную матрицу (N строк, M столбцов) случайными числами, подсчитать количество нулевых значений.

3. Заполнить двумерную матрицу (N строк, M столбцов) случайными числами, найти максимальный и минимальный элемент матрицы.

Лабораторная работа № 6-7. МАССИВЫ

Массив *– это структурированный тип данных, состоящий из фиксированного числа элементов, имеющих один и тот же тип данных.*

Массив представляет собой одну переменную с множеством ячеек памяти для хранения значений, тогда как обычная переменная имеет только одну ячейку памяти, в которой может храниться только одно значение. Название ***регулярный тип*** (или ***ряды***) массивы получили за то, что в них объединены *логически однородные* (однотипные) элементы, упорядоченные по ***индексам***. Каждый *индекс,* иначе, (***номер***), однозначно определяет местоположение значения элемента массива в оперативной памяти.

Количество индексов, используемых для определения элементов массива, определяет ***размерность*** *массива.*

Различают ***одномерный*** *(вектор; строка) массив,* ***двумерный*** *(матрица; таблица) массив и в общем случае* ***- n-мерный*** *массив.*

Массив с заданным размером называется массивом ***фиксированного размера.***

Массив с переменным размером называется ***динамическим массивом.***

Описание массива фиксированного размера

Для описания массива фиксированного размера используется следующий синтаксис:

Dim ИмяПеременной[([индексы])] [As тип]

(Квадратные скобки указывают на наличие необязательного параметра).

Для задания аргумента ***индексы*** может использоваться следующий синтаксис:

[нижний индекс ***To****] верхний индекс[, [нижний индекс* ***To****] верхний индекс] ...*

Если нижний индекс не задан явно, нижняя граница массива равняется нулю. Верхняя граница определяет максимальное значение количества элементов в массиве. ***To*** *– зарезервированное слово.* Тип элементов массива можно не указывать явно, поскольку по умолчанию тип данных – *variant.*

В описании массива в языке VBA размерность задается числом граничных пар или максимальных значений. Рассмотрим несколько способов объявления одного и того же двумерного массива вещественного типа.

Dim mass(8,3) as double *– 'объявление массива заданием его максимальных значений;*

Dim mass(0 To 8, 0 To 3) as double *– 'объявление массива заданием его граничных пар.*

Индексация массива по умолчанию начинается с 0. В языке *VBa* допускается индексация и с 1. Для этого необходимо использовать инструкцию ***Option Base 1***, которая должна находиться в разделе описания модуля.

Описание массива переменного размера

Если массив описан как динамический, то можно изменять его размер во время программы.

Синтаксис:

Dim ИмяПеременной () [As Type]

В этом случае при описании динамического массива с помощью инструкции *Dim* размер массива не указывается, для описания используются пустые круглые скобки. Далее в программе следует воспользоваться инструкцией ***ReDim*** для изменения числа *размерностей* и определения *числа элементов* в массиве.

Например:

Redim mass(25) – одномерный массив из 26 элементов;

Redim mass(10,10)- двумерный массив из 121 элемента.

Инструкцию ***Redim*** в программе можно применять столько раз, сколько потребуется. Однако при каждом применении ***Redim*** данные, содержащиеся в массиве, теряются.

Типовые алгоритмы обработки массивов

1. Обнуление всех элементов массива:

```
For i=0 To N
  mass(i)=0
Next i
```

2. Заполнение элементов массива значениями с клавиатуры

```
For i=0 To N
  mass(i)= InputBox("Ввод числа")
Next i
```

3. Заполнение элементов массива случайными целыми значениями

```
Randomize
  For i=0 To N
    mass(i)=Int(Rnd*100+1)
  Next i
```

4. Заполнение и вывод на экран элементов массива

```
Dim s as string
  For i=0 To N
    mass(i)= Int(Rnd*100+1)
    s=s & mass(i) & ”-” ( накапливание в строке значений элементов массива)
  Next i
```

5. Формирование одного массива из элементов другого массива, удовлетворяющих заданному условию.

```
For i=0 To N
 massA(i)=Int(Rnd*100+1)
    If massA(i)>50 then
      massB(j)=massA(i)
      j=j+1
   End if
Next i
```

6. Поиск максимального и минимального элементов массива

```
max=mass(0)
min=mass(0)
 For i=1 To n
   If mass(i)>max Then max=mass(i)
   If mass(i)<min Then min=mass (i)
Next i
```

7. Обмен местами максимального и минимального элементов массива

```
max=mass(0)
min=mass(0)
For i=1 To n
  If mass(i)>max Then
   max=mass(i)
   imax=i
  End if
  If mass(i)<min Then
    min=mass (i)
   imin=i
 End if
Next i
x = mass(imax)
mass(imax) = mass(imin)
mass(imin) = x
```

ЗАДАНИЯ НА САМОСТОЯТЕЛЬНУЮ РАБОТУ

1. Дан массив A(15). Заполнить массив случайными значениями от 0 до 25. Если в массиве есть хотя бы один элемент, равный 2, то все четные элементы массива заменить нулями и подсчитать их количество, иначе найти среднее арифметическое всех нечетных элементов. Вывести начальный и конечный состав массива.

2. Дан массив M(15). Если разность между средним арифметическим и средним геометрическим кратно 2, то элементы с четными индексами заменить нулями, иначе найти сумму квадратов всех элементов массива. Вывести полученный вектор.

3.Дан массив C (15). Найти среднее арифметическое всех элементов массива и наибольшее отклонение от среднего, т.е. максимум из (C(i) – Ср.арифм.) .

4. Дан массив A(12). Заполнить массив случайными значениями. Найти среднее арифметическое всех элементов массива. Заменить значение элемента массива нулем, если оно меньше среднего арифметического и единицей – если больше. Вывести начальный и конечный состав массива.

5. Даны массивы C(8) и B(8). Заполнить массивы случайными значениями. Преобразовать значения элементов массива B по правилу: если C(i) делится на 5 , то B(i) увеличить в пять раз, иначе B(i) заменить нулем. Вывести начальный и конечный составы массивов.

6. Дан массив M(15). Заполнить его случайными значениями от 0 до 15. Если количество элементов, значения которых лежат в пределах от 3 до 11 меньше 4, то найти сумму всех элементов массива кратных этому количеству, иначе возвести в квадрат те элементы массива, чей остаток от деления на количество не превышает 2. Вывести начальный и конечный состав массива.

7. Дан массив C (15). Найти среднее геометрическое всех элементов массива и наибольшее отклонение от среднего, т.е. максимум из (C(i) – Ср.арифм.). Если полученное наибольшее кратно 2, то заменить этим значением каждый третий элемент массива, иначе каждый пятый возвести в квадрат.

Лабораторная работа № 8-9
Объектное программирование в VBA

Visual Basic For *Application* – визуальный объектно-ориентированный язык программирования приложений. Ключевой идеей объектно-ориентированного программирования является объединение данных и оперирующих с ними функций в один объект. Язык *VBA не* существует вне какого-либо приложения. Он встроен в такие приложения как редактор электронных таблиц *Excel, СУБД MS Access*, текстовый редактор *Word*. В каждом из этих приложений существуют свои объекты, которые могут строиться на основе более мелких объектов - ***элементов управления*** и объединяются в более крупные объекты – ***семейства.***

Объект представляет собой элемент приложения, такой как, лист (*Worksheet*), ячейка (*Cells*), диапазон (*Range)* в *Excel*.

Семейство (объект *Collection*) представляет собой объект, содержащий несколько других объектов, как правило, одного и того же типа. Например, в *Microsoft Excel объект Workbooks* (рабочие книги) содержит все открытые объекты *Workbook* (рабочая книга).

Элемент семейства может быть идентифицирован по номеру или имени. Например, в следующей процедуре *Workbooks(1)* обозначает первый закрытый объект Workbook.

```
Sub CloseFirst()
 Workbooks(1).Close
End Sub
```

Все объекты имеют сохраняемый набор свойств, изменяя которые можно управлять объектом.

Свойство – это качественная или количественная характеристика объекта.

Свойства определяют такие характеристики объекта, как размер, цвет, положение на экране или состояние объекта, например, включенное или отключенное. Изменяя свойства, можно изменять характеристики объекта или набора объектов. Установка значения свойств – это один из способов управления объектами.

Для установки свойства необходимо ввести имя объекта, а затем через точку имя свойства объекта. Далее должны следовать знак равенства и значение свойства. Синтаксис установки значения свойства объекта выглядит следующим образом:

Объект.Свойство=Выражение

Например:

Range(“A2”).Formula=”=СУММ(A1:C1)”

В этом примере в ячейку с относительным адресом A2 (для адресации используется объект *Range*, который позволяет обратиться как к диапазону ячеек, так и к одной ячейке) активного рабочего листа *Excel* вставляется формула *=СУММ(A1:C1)* путем изменения свойства *Formula*.

Некоторые свойства являются неизменяемыми, т.е. ***доступными только для чтения***. В этом случае синтаксис - *Объект.Свойство=Выражение* применять нельзя. Существует другой синтаксис, позволяющий читать свойства объекта, который выглядит следующим образом:

Переменная=Объект.Свойство

Например:

ТекущийПроцент=Cells(1,1).Value

В этом примере переменной *ТекущийПроцент* присваивается значение (свойство *Value)* ячейки A1 (первая строка, первый столбец - *Cells(1,1)*) текущего рабочего листа Excel.

С каждым объектом также связывают ряд методов, иначе, команд, применяемых к объекту.

Метод представляет собой действие, выполняемое над объектом.

Синтаксис вызова метода объекта имеет следующий вид:

Объект.метод

В приложении *Microsoft Excel* у объекта диапазон ячеек имеется метод *Clear*, позволяющий очистить содержимое диапазона. Если предварительно задать имя диапазона, то вызов метода очистки имеет следующий синтаксис:

Range (“Личные сведения”).Clear

А в примере:

Range (“A10:C12”).Select - выбирается диапазон ячеек A10:C12.

Инструкция With..End With

Инструкция *With..End With*, как правило, применяется тогда, когда нужно задать или изменить несколько свойств одного и того же объекта. Эта инструкция не является управляющей, так как она не изменяет порядка выполнения команд, не вызывает циклического выполнения группы команд или безусловный переход. Однако данная инструкция часто используется в циклах. Она выполняет последовательность инструкций над одиночным объектом или определенным пользователем типом данных, не повторяя задания имени объекта.

Синтаксис:

With объект
[инструкции]
End With

Например:

```
Sub Цвет()
Dim a As Object
  For Each a In Selection
    If a.Value = 0 Then
         With a.Font ‘ – шрифт диапазона ячеек
              .Bold = True ‘ – жирный шрифт
             .ColorIndex = 3 ‘ – цвет шрифта ячейки (3 - красный)
            .Size = 20 ‘ – размер шрифта ячейки
        End With
     End If
   Next
End Sub
```

В этом примере в зависимости от содержимого ячейки выделенной области, которая возвращается методом ***Selection***, меняется размер шрифта, цвет и тип шрифта каждой ячейки. В данной процедуре Font – шрифт диапазона ячеек – является объектом, который имеет такие свойства, как цвет, размер и тип шрифта. Воспользовавшись инструкцией *With..End With* можно, не указывая каждый раз имени объекта, изменять эти свойства.

Инструкция For Each...Next

Инструкция For Each...Next повторяет набор инструкций для всех объектов семейства или для всех элементов массива.

Синтаксис:

For Each элемент In группа
инструкции
[Exit For]
инструкции
Next [элемент]

Элемент – управляющая объектная переменная цикла; ***группа*** - имя семейства объектов***; Exit For*** – альтернативный выход из цикла.

Вход в блок *For...Each* выполняется только в том случае, если группа содержит хотя бы один элемент. После входа в цикл все инструкции цикла

выполняются для первого элемента группы. Затем, если группа содержит другие элементы, инструкции цикла выполняются для каждого элемента группы. После обработки всех элементов цикл завершается, а выполнение продолжается с инструкции, следующей за инструкцией *Next*.

Следующий пример выполняет цикл по диапазону *A1:D10* на рабочем листе “Лист1” и присваивает любому числу, имеющему значение меньше 10, значение 0.

```
Sub List()
Dim c as object
  For Each c In Sheets(“Лист1”).Range(“A1:D10”)
      If c.Value < 10 Then c.Value = 0
   Next c
End Sub
```

В следующем примере на рабочем листе диапазону выделенных ячеек предварительно присваивается имя “*МойДиапазон*”, которое затем и участвует в программе:

```
Sub Диапазон()
 Dim x as object, count
   For Each x In Range("МойДиапазон")
      If x.Value = "" Then count = count + 1
    Next x
 MsgBox "Всего " & count & " пустых ячеек"
End Sub
```

В этой программе в диапазоне ячеек рабочего листа с именем “*МойДиапазон*” проверяется содержимое ячеек. Если ячейка ничего не содержит (“”- значение пустой строки), тогда счетчик - переменная *count* - увеличивается на 1. Далее, инструкция *MsgBox* выводит на экран значение этого счетчика, т. е. количество пустых ячеек в указанном диапазоне.

Создание диалоговых форм

Для создания диалоговых окон приложений в *VBA* используются формы. Форма – это окно пользовательского интерфейса. В *VBA* это *UserForm* со стандартной сеткой для удобного размещения визуальных элементов на ней. Шаг сетки может быть изменен через команду ***Параметры.*** Работа с UserForm выполняется в редакторе форм.

Визуальные компоненты - ***элементы*** находятся на панели элементов и называются ***элементами управления***. Это кнопки, поля, переключатели

надписи, списки и т. д. Все эти элементы могут быть добавлены в форму для создания и оформления *пользовательского интерфейса (рис. 16)*.

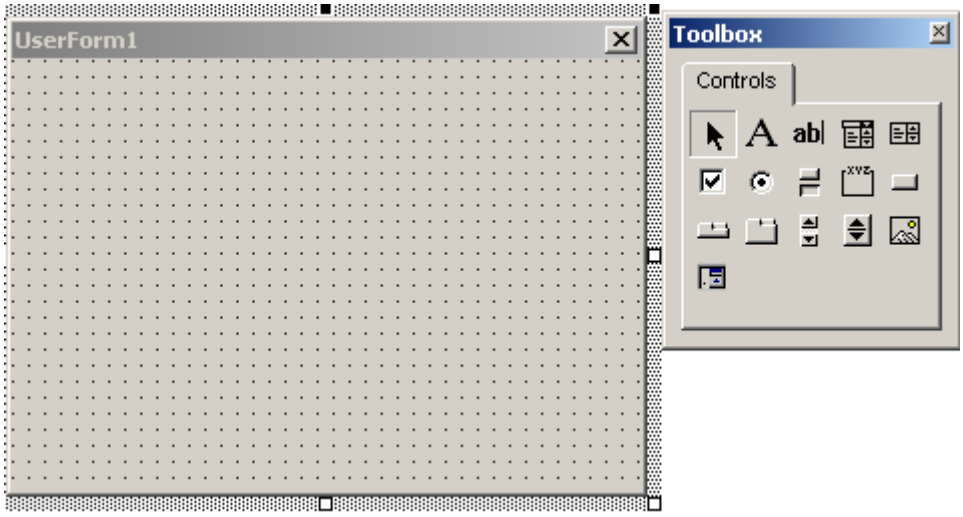


Рис.16. Пользовательская форма в Excel

Элементы управления размещаются в окне формы путем перетаскивания их с панели элементов, при этом для распознавания элементов достаточно установить на него указатель мыши.

Объект (элемент управления) может реагировать на определенные события, происходящие в процессе работы приложения. Совокупность событий, на которые объект способен реагировать, определяется создателем класса, экземпляром которого является данный объект. Например, набор событий, которые определены для формы можно посмотреть на вкладке События (диалогового окна *Properties*). Реакцией объекта на произошедшее событие может быть выполнение объектом некоторой специальной процедуры, которая называется ***процедурой обработки события***. Любому событию объекта может быть назначена некоторая процедура его обработки.

Обычно события инициируются действиями пользователя. В зависимости от производимых пользователем действий события можно разделить на несколько типов: события данных, события фокуса, события клавиатуры, события мыши, события печати, события фильтра, события окна, события ошибок и событие таймера.

ЗАДАНИЯ НА САМОСТОЯТЕЛЬНУЮ РАБОТУ

1. Создать рабочую книгу с именем "Объектное программирование в Excell";

2. Зайти в модуль книги и создать процедуру заполнения ячеек рабочего листа с адресом «A1:E10» случайными двухзначными значениями.

3. Найти среднее арифметическое, среднее геометрическое заполненного диапазона.

4. Поменять размер, начертание и цвет тех ячеек диапазона, которые кратны 5.

5. Поменять местами максимальный и минимальный элемент диапазона.

6. Создать пользовательскую форму с элементами ***группа*** (переключатели красный, синий, зеленый), ***прямоугольник*** и кнопка ***Закрыть.*** Написать процедуры обработки события на изменение цвета прямоугольника и закрытия формы.

7. Создать пользовательскую форму с полями ***X*** и ***Y*, Результат**, списком ***Операция***, полем со списком ***Действие*** и кнопкой ***Закрыть***. В список и поле со списком поместить значения, соответствующие арифметическим операциям (не менее 5). Создать соответствующие процедуры обработки событий на выполнение арифметических действий и помещения полученного значения в поле ***Результат***. Предусмотреть обработку ошибки от деления на 0.