Основные понятия программирования

Компьютерная программа— последовательность инструкций, предназначенных для исполнения устройством управления компьютера (процессором).

Программа — данные, предназначенные для управления конкретными компонентами системы обработки информации в целях реализации определённого алгоритма. (ГОСТ 19781—90. ЕСПД. Термины и определения).

Алгоритм — набор инструкций, описывающих порядок действий исполнителя для достижения поставленной цели за конечное число шагов.

Свойства алгоритмов

- **Дискретность** - алгоритм должен представлять процесс решения задачи как последовательное выполнение некоторых простых шагов. При этом для выполнения каждого шага алгоритма требуется конечный отрезок времени, то есть преобразование исходных данных в результат осуществляется во времени дискретно.
- **Детерминированность** (определённость) - в каждый момент времени следующий шаг работы однозначно определяется состоянием системы. Многократное применение одного алгоритма к одному и тому же набору исходных данных должно всегда давать один и тот же результат.

Свойства алгоритмов (продолжение)

- **Завершаемость** (конечность) — при корректно заданных исходных данных алгоритм должен завершать работу и выдавать результат за конечное число шагов.
- **Понятность** — алгоритм должен включать только те команды, которые доступны исполнителю и входят в его систему команд.
- **Массовость** (универсальность). Алгоритм должен быть применим к разным наборам исходных данных.
- **Результативность** — завершение алгоритма определёнными результатами. Алгоритм не содержит ошибок, если он даёт правильные результаты для любых допустимых исходных данных, в противном случае алгоритм содержит ошибки.

Формы записи алгоритмов

- словесная
- псевдокод (формальные алгоритмические языки);
- схематические:
  - графическая (блок-схемы);
  - структурограммы (диаграммы Насси-Шнейдермана).

Обычно сначала (на уровне идеи) алгоритм описывается словами, но по мере приближения к реализации он обретает всё более формальные очертания и запись на языке, понятном исполнителю (например, машинный код).

Поколения языков программирования

Язык программирования — формальная знаковая система, предназначенная для записи компьютерных программ.

- I (конец 40х гг. 20 в.) – машинный язык (двоичные коды) и язык ассемблера (50-е гг.) (система обозначений, используемая для представления в удобочитаемой форме программ, записанных в машинном коде). Ориентированы на конкретный компьютер.
- II (конец 50х гг. 20 в.) - символьный ассемблер, в котором появилось понятие переменной. Основная отличительная особенность: ориентирование на абстрактный компьютер с такой же системой команд.

Поколения языков программирования

III (60-е г.г. 20 в.) - языки программирования высокого уровня.

Отличительные особенности:

- относительная простота;
- независимость от конкретного компьютера;
- возможность использования мощных синтаксических конструкций.

Основная отличительная особенность языков третьего поколения - ориентирование на алгоритм (алгоритмические языки).

Примеры: FORTRAN, BASIC, PL/1, C PASCAL и др.

Поколения языков программирования

IV (70-е г.г. 20 в.) – языки сверхвысокого уровня, предназначенные для реализации крупных проектов. Проблемно-ориентированные языки, оперирующие конкретными понятиями узкой области. Как правило, в такие языки встраивают мощные операторы, позволяющие одной строкой описывать функции, для описания которых в языках младших поколений потребовалось бы сотни или даже тысячи строк исходного кода.

Часто относят: SQL, SGML (HTML, XML), Prolog, и др. узкоспециализированные декларативные языки.

Основная отличительная особенность языка четвертого поколения: приближение к человеческой речи (декларативные языки).

Тенденции развития языков программирования

Специализация – свой язык для отдельной области (базы данных, Web, графика и т.д.). Как результат программа становится проще и эффективнее. Недостаток – программы становятся менее универсальными.

Конструирование программ вместо традиционного программирования. Программа создается быстрее, простые программы можно создавать, не изучая язык программирования. Недостаток – программы получаются менее эффективными по использованию памяти, быстродействию и размеру.

Многоплатформенность – исполняемая программа может выполняться на разных платформах (процессор + ОС) без перекомпиляции.

Недостаток – снижение скорости выполнения.

Этапы разработки программ

1. Постановка задачи (формулирование сути задачи, определение исходных данных, требований к результату и работе программы в целом)
2. Разработка алгоритма
3. Составление текста программы на языке программирования
4. Трансляция программы (перевод на язык машинных команд)
   - Компиляция – перевод всей программы на машинный язык
   - Интерпретация – перевод каждой инструкции программы на машинный язык в процессе выполнения
5. Отладка – устранение ошибок в программе
6. Тестирование – проверка работоспособности программы при различных исходных данных

Объектно-ориентированное программирование

Традиционное (**процедурное**) программирование предполагает описание каждого шага в процессе решения задачи. При этом данные и подпрограммы (процедуры, функции) их обработки формально не связаны.

ООП - это подход к разработке программного обеспечения, основанный на объектах, а не на процедурах.

Объект – это модель реальной сущности в программной системе.

Объект состоит из структуры данных и связанных с ней процедур (называемых методами), которые работают с данными, записанными в экземплярах структуры данных.

Однотипные объекты объединяются в **классы.**

Фактически класс описывает устройство схожих объектов

Объектно-ориентированное программирование

Индивидуальные объекты называются ***экземплярами класса***, а класс – это шаблон, по которому строятся объекты. Класс определяет общие для объектов **методы и свойства**.

- ***Методы*** – это программные процедуры, определяющие взаимодействие объекта с внешней средой.
- ***Свойства*** - это характеристики объектов (видимость на экране, размер, положение и т.п.).

События - ситуации, в которых объект оказывается и на которые может ответить заранее определенными для таких ситуаций действиями (описанными как правило в обработчиках событий). К событиям можно отнести следующее:

- физические действия пользователя программы, например щелчок кнопкой мыши, перемещение курсора и т. д.;
- ситуации, в которые попадает объект в ходе выполнения программы.

Особенности ООП

- ***Инкапсуляция -*** это свойство системы, позволяющее объединить данные и методы, работающие с ними в классе, и скрыть детали реализации от пользователя. Доступ к объекту возможен только через обращение к его методам и свойствам. Внутренняя структура объекта скрыта от пользователя.
- ***Наследование -*** это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником или производным классом.
- ***Полиморфизм -*** единообразная обработка разнотипных данных. То есть возможно использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Visual Basic for Applications (VBA)

VBA – это подмножество визуального языка программирования Visual Basic (VB), которое включает почти все средства создания приложений VB.

VBA отличается от языка программирования VB тем, что система VBA предназначена для непосредственной работы с объектами MS Office, в ней нельзя создавать проект независимо от приложений MS Office.

Таким образом, в VBA языком программирования является VB, а инструментальная среда программирования реализована в виде редактора VB, который может активизироваться из любого приложения MS Office.

Основы синтаксиса VBA

Программа состоит из операторов (statements).

В каждой строке, как правило располагается один оператор.

Если нужно продолжить оператор в следующей строке, то текущая должна заканчиваться пробелом и подчеркиванием.

Регистр символов не учитывается.

Комментарии (текст, который игнорируется транслятором и не влияет на ход выполнения программы), начинаются с апострофа и продолжаются до конца строки.

Лишние пробелы в тексте программы игнорируются.

Имена, задаваемые пользователем должны:

- начинаться с букв;
- состоять не более чем из 255 символов;
- не совпадать с ключевыми словами VBA;
- не содержать в себе точек, пробелов, а также символов !, @, #, &, % и $

Основы синтаксиса VBA (для MS Excel)

Синтаксис установки значения свойства объекта:

Объект. Свойство = Выражение

Основным свойством объектов **Cells** (ячейки) и **Range** (диапазон), является **Value** (значение), которое можно не указывать. Например:

Range(“A5:A10”). Value = 0 или

Range(“A5:A10”) = 0 – в диапазон ячеек A5:A10 заносится значение 0.

Cells(2, 4). Value = n или

Cells(2, 4) = n – в ячейку, находящуюся на пересечении 2-й строки и 4-го столбца (ячейка с адресом “D2”), заносится значение переменной n (заданное когда-то ранее).

Основы синтаксиса VBA

Синтаксис чтения свойств объекта:

Переменная = Объект. Свойство

Например:

Xn = Cells(1, 2).Value или

Xn = Range(“B1”).Value – переменной Xn присваивается значение из ячейки B1 текущего рабочего листа.

Синтаксис применения метода к объекту:

Объект. Метод

Например:

Sheets(2).Activate – сделать активным лист с №2.

Sheets(“Диаграмма”).Delete – удалить лист с именем “Диаграмма”.

Основы синтаксиса VBA

В MS Excel имеются объекты, которые содержат другие объекты. Точка после имени объекта может использоваться для перехода от одного объекта к другому. Например,

Workbooks("Отчет").Worksheets("Май").Rows(2).Delete

очищает вторую строку рабочего листа **Май** в рабочей книге **Отчет.**

Объектом самого высокого уровня является **Application** (приложение).

Если вы изменяете его свойства или вызываете его методы, то результат применяется к текущей работе MS Excel. Например:

Application.Quit - завершение работы с Excel.

Типы данных VBA

Все объекты, которыми оперирует язык программирования VBA, относятся к определенному типу.

Тип данных определяет:

- область возможных значений переменной;
- структуру организации данных;
- операции, определенные над данными этого типа.

Типы данных подразделяются на простые (скалярные) и сложные (структурированные).

У простых типов данных возможные значения данных едины и неделимы.

Сложные типы имеют структуру, в которую входят различные простые типы данных.

Простые типы данных VBA

Имя типа	Название	Возможные значения
Boolean	Логический	True, False
Byte	Байт	Целые 0…255
Integer	Целое	Целые-32768…+32767
Long	Длинное целое	Целые-2147483648…+2147483647
Single	Число с плавающей точкой	Точность 7-8 значащих цифр
Double	Число с плавающей точкой двойной точности	Точность 14-15 значащих цифр
Currency	Денежный	Возможны 15 цифр до запятой и 4 после.
String	Строковый	Данные записываются в кавычках
Date	Дата	Даты от 1.01.100г. до 31.12.9999г.
Object	Объект	Ссылка на объект
Variant	Вариант	Значением могут быть данные любого из перечисленных выше типов, объекты, значения NULL и значения ошибок ERROR.

Описание переменных в VBA

Переменные в программе можно описывать или не описывать. В последнем случае им будет присвоен тип**Variant**.

Явно описывать переменную можно как в начале блока, так и в любом месте, где возникла необходимость использовать новую переменную.

Лучше все переменные описывать явно и, как правило, в начале блока.

Для запрета использования переменных, которые не были описаны явно, в начало программы необходимо вставить оператор

Option Explicit.

Описание простых переменных в VBA

Описание простых переменных имеет следующий синтаксис:

Dim *ИМЯ_ПЕРЕМЕННОЙ* **As** *ИМЯ_ТИПА*

Одним оператором **Dim** можно описать произвольное число переменных, но конструкция **As** должна быть указана для каждой из них, иначе переменным без **As** будет присвоен тип **Variant**.

Например.

Dim X As Byte, Z As Integer, C, Слово As String

Здесь переменная *X* - это переменная байтового типа, переменная *Z* - целого типа, переменная *C* - типа вариант (по умолчанию), переменная *Слово* - строкового типа.

Выражения в VBA

Выражения устанавливают порядок выполнения действий над элементами данных. Выражения состоят из операндов и знаков операций. Операндами являются константы, переменные, указатели функций, выражения, взятые в скобки.

Примеры выражений:

P=(a+b+c)/2

Площадь=Sqr(p*(p-a)*(p-b)*(p-c))

Операции в выражениях VBA

арифметические операции:

^ возведение в степень,
* умножение,
/ деление,
\ деление нацело (остаток отбрасывается),
mod остаток от деления,
+ плюс,
- минус;

операции отношения:

< меньше,
> больше,
<= меньше или равно,
>= больше или равно,
= равно,
<> не равно;

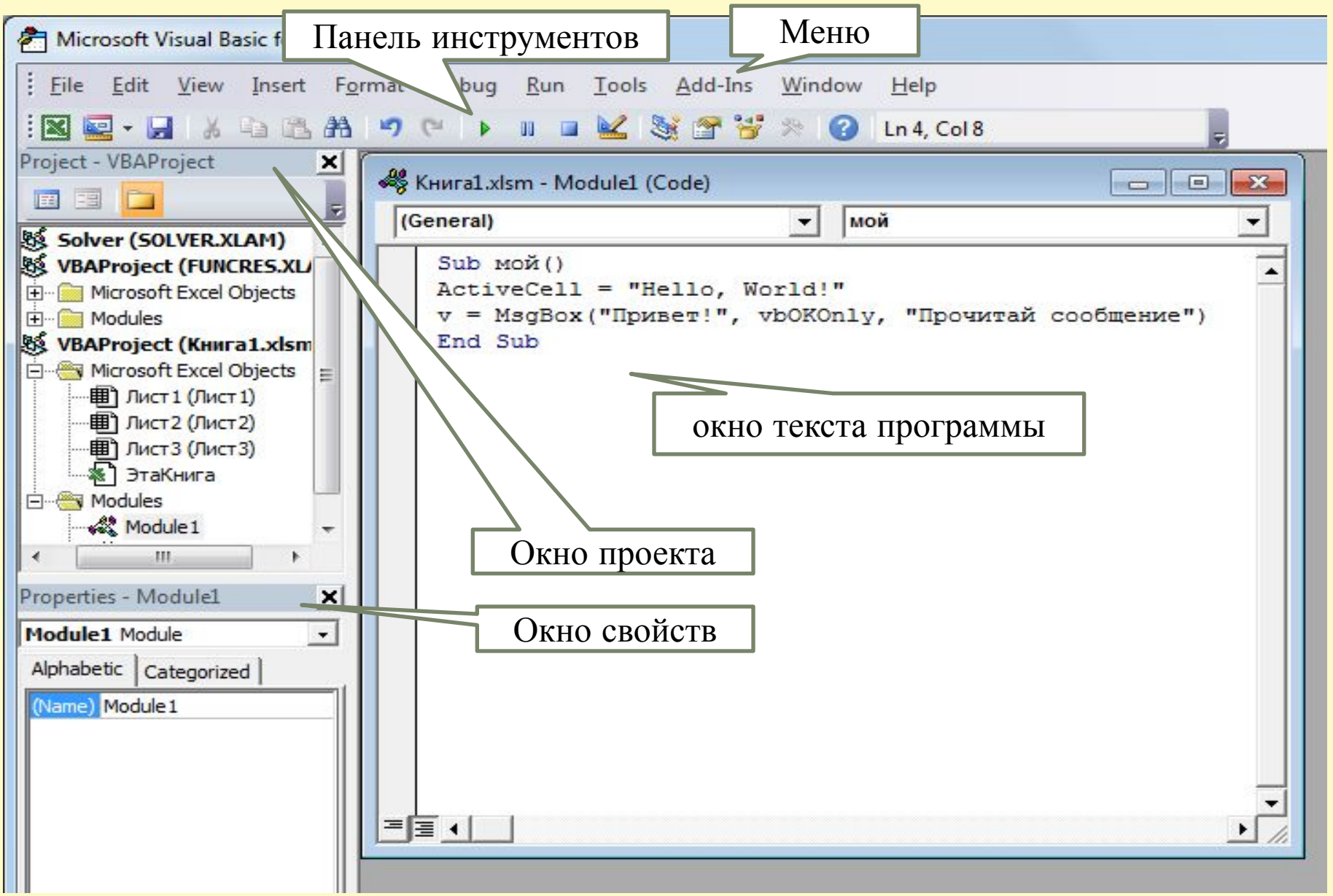
логические операции:

Not логическое отрицание,
And логическое "И",
Or логическое "ИЛИ".

Стандартные математические функции VBA

Функция	Описание
Abs(число)	Возвращает значение, тип которого совпадает с типом переданного аргумента, равное абсолютной величине указанного числа.
Atn(число)	Возвращает значение типа Double, содержащее арктангенс числа.
Cos(число)	Возвращает значение типа Double, содержащее косинус угла
Int(число)	Возвращает значение типа, совпадающего с типом аргумента, которое содержит целую часть числа.
Log(число)	Возвращает значение типа Double, содержащее натуральный логарифм числа.
Exp(число)	Возвращает значение типа Double, содержащее результат возведения числа ***e*** (основание натуральных логарифмов) в указанную степень
Sgn(число)	Возвращает значение типа Variant (Integer), соответствующее знаку указанного числа. Возможные значения -1, 0 или 1.
Sin(число)	Возвращает значение типа Double, содержащее синус угла.
Sqr(число)	Возвращает значение типа Double, содержащее квадратный корень указанного числа.
Tan(число)	Возвращает значение типа Double, содержащее тангенс угла.

Окно редактора VBA



Процедуры в VBA

Стандартные модули могут содержать:

- процедуры общего типа,
- процедуры-функции, разработанные пользователем,
- процедуры, записанные макрорекордером.

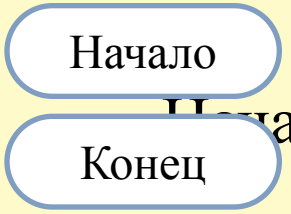
Процедура - это последовательность команд (операторов языка), начинающаяся с оператора **Sub** и заканчивающаяся оператором **End Sub**.

Все операторы, которые заключены между этими двумя операторами, составляют **тело процедуры**.

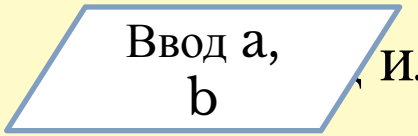
Если программа создается безотносительно к формам или их элементам, следует создать свой модуль, а в нем – свою процедуру, последовательно выполнив команды:

- **Вставка – Модуль (Insert – Module)**
- **Вставка – Процедура (Insert – Procedure)**

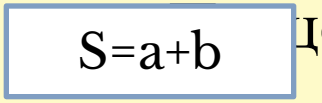
Элементы блок-схем



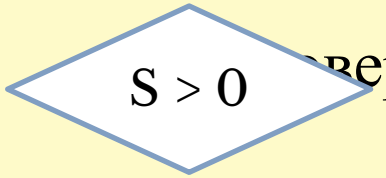
ало или конец программы



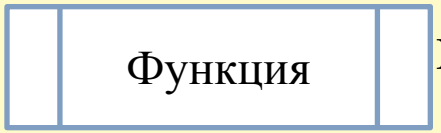
или вывод



цесс (операция)



верка условия

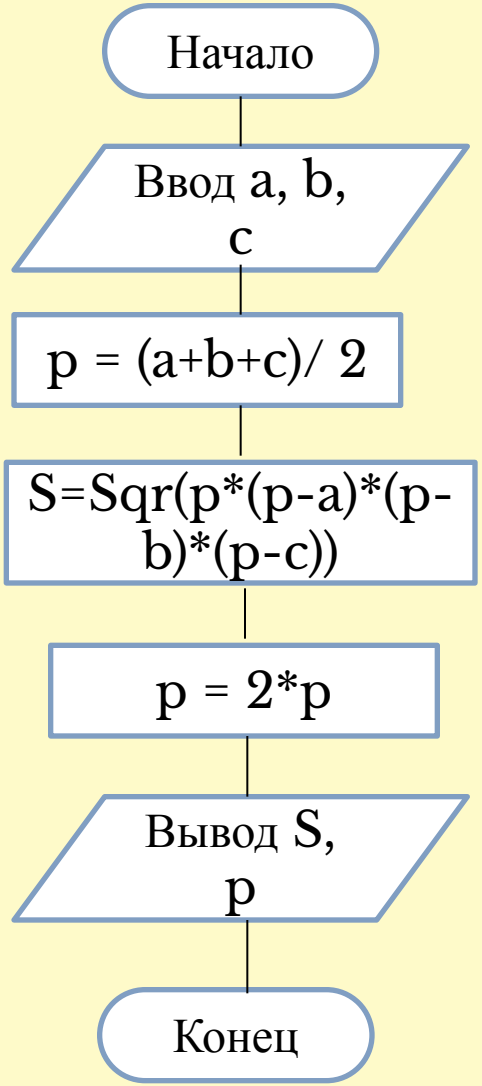


подпрограммы

икла со счетчиком

Линейный алгоритм

Все действия выполняются последовательно одно за другим



Вычисление площади и периметра треугольника по трем сторонам

Ввод сторон треугольника

Вычисление полупериметра

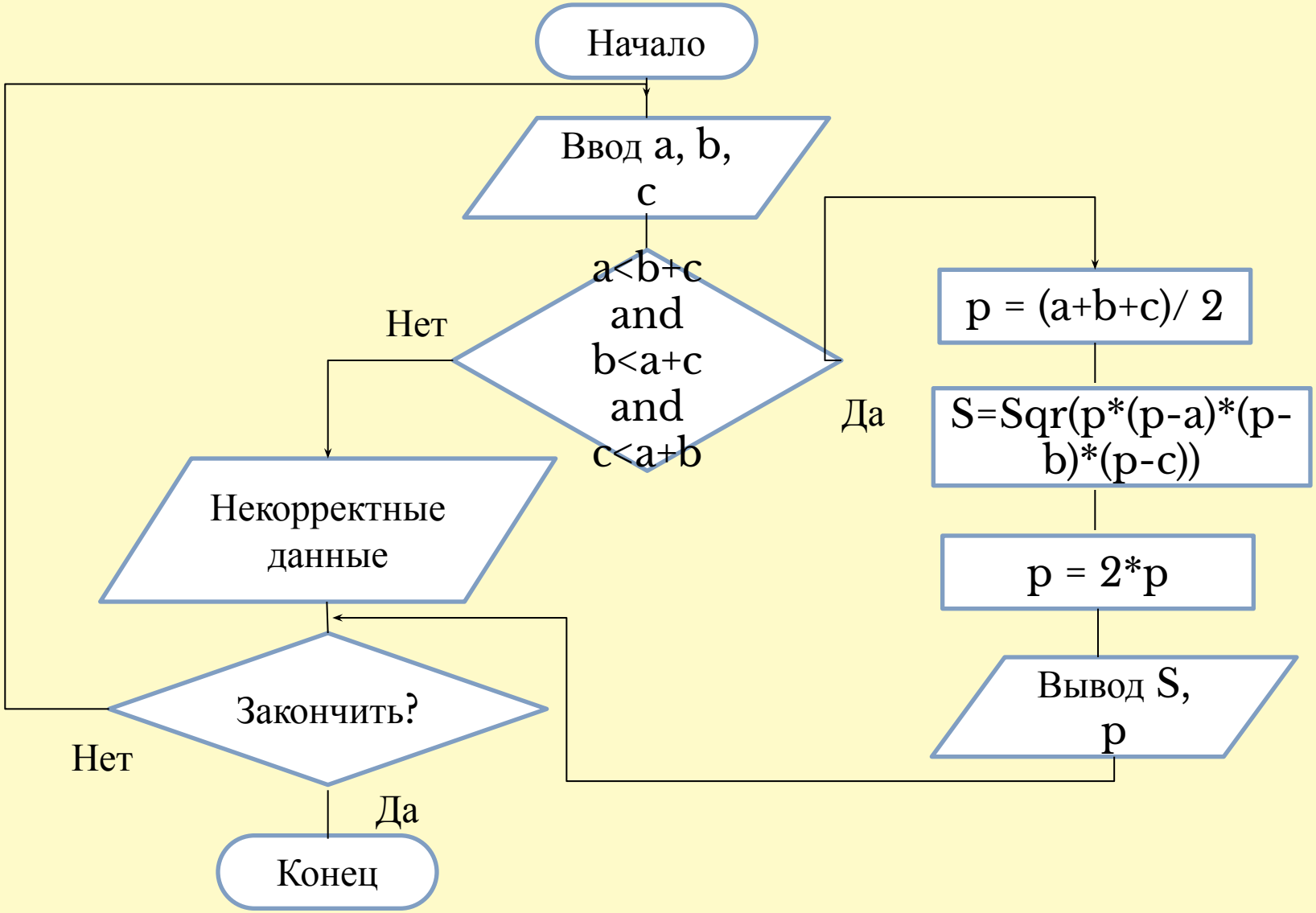
Вычисление площади

Вычисление периметра

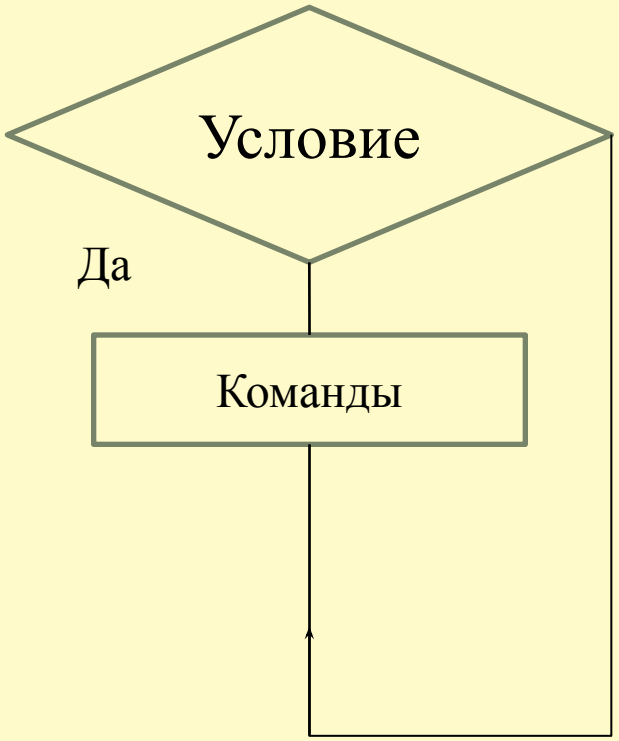
Вывод площади и периметра

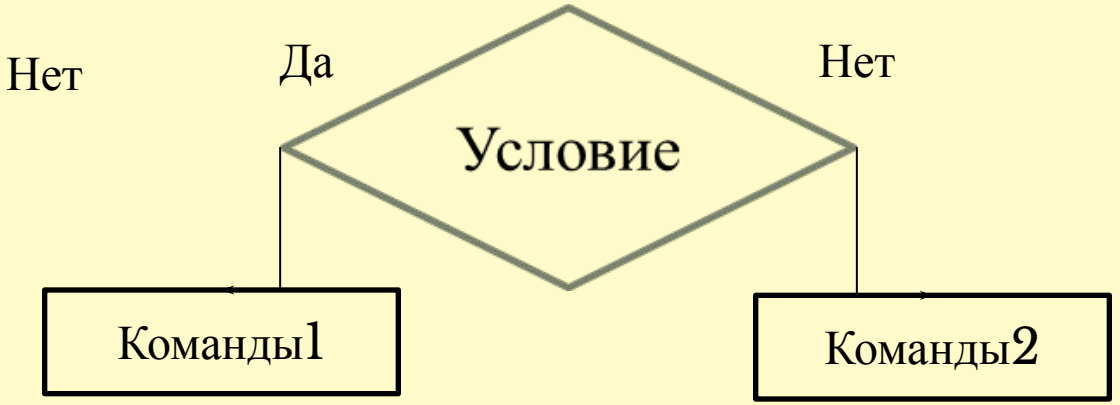
Разветвляющийся алгоритм

Ход выполнения зависит от проверки условий



Оператор условного перехода





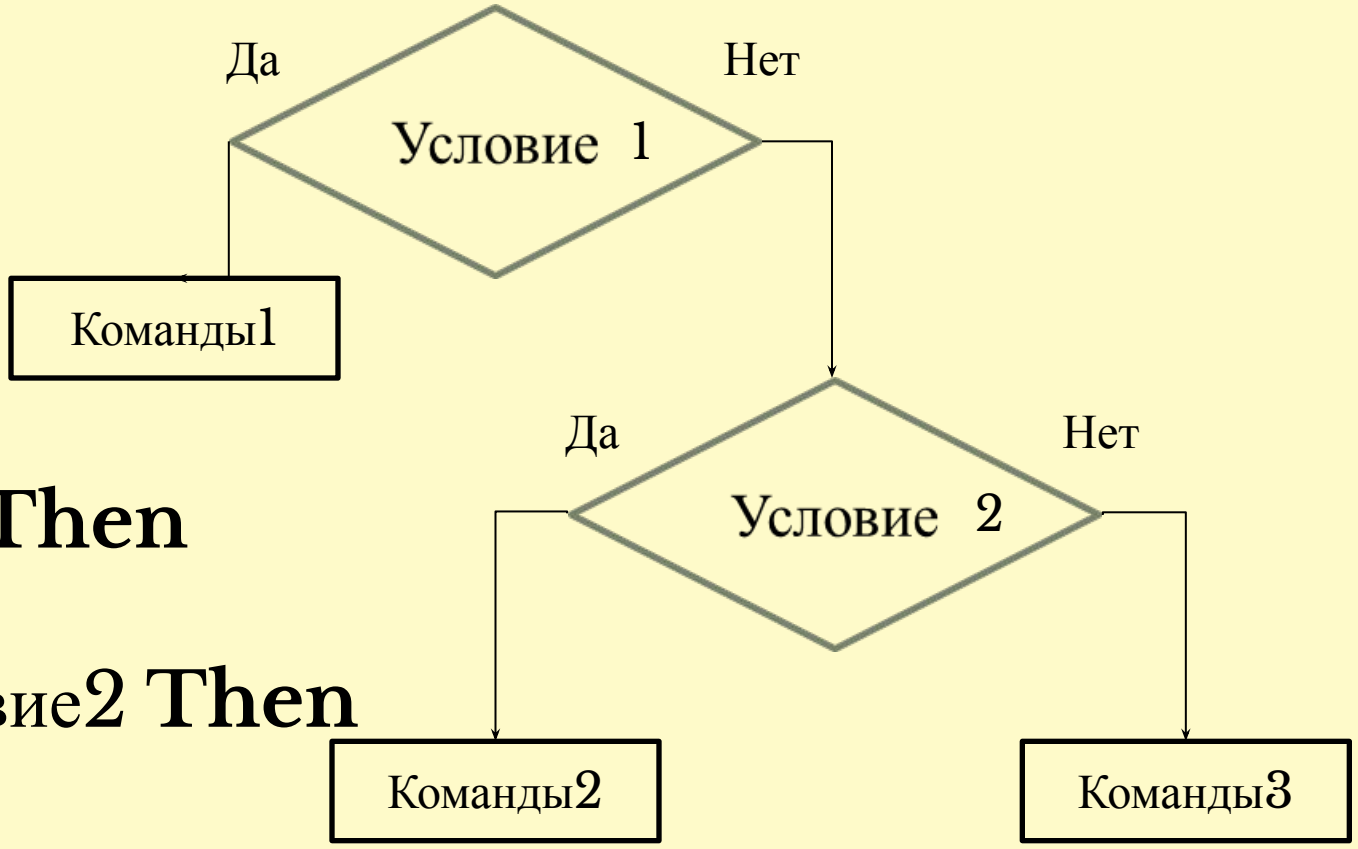
If Условие **Then**
Команды
End If

If Условие **Then**
Команды1
Else
Команды2
End If

Если команда одна, то можно в одну строку без End If

If Условие **Then** Команда или **If** Условие **Then** Команда1 **Else** Команда2

Оператор условного перехода

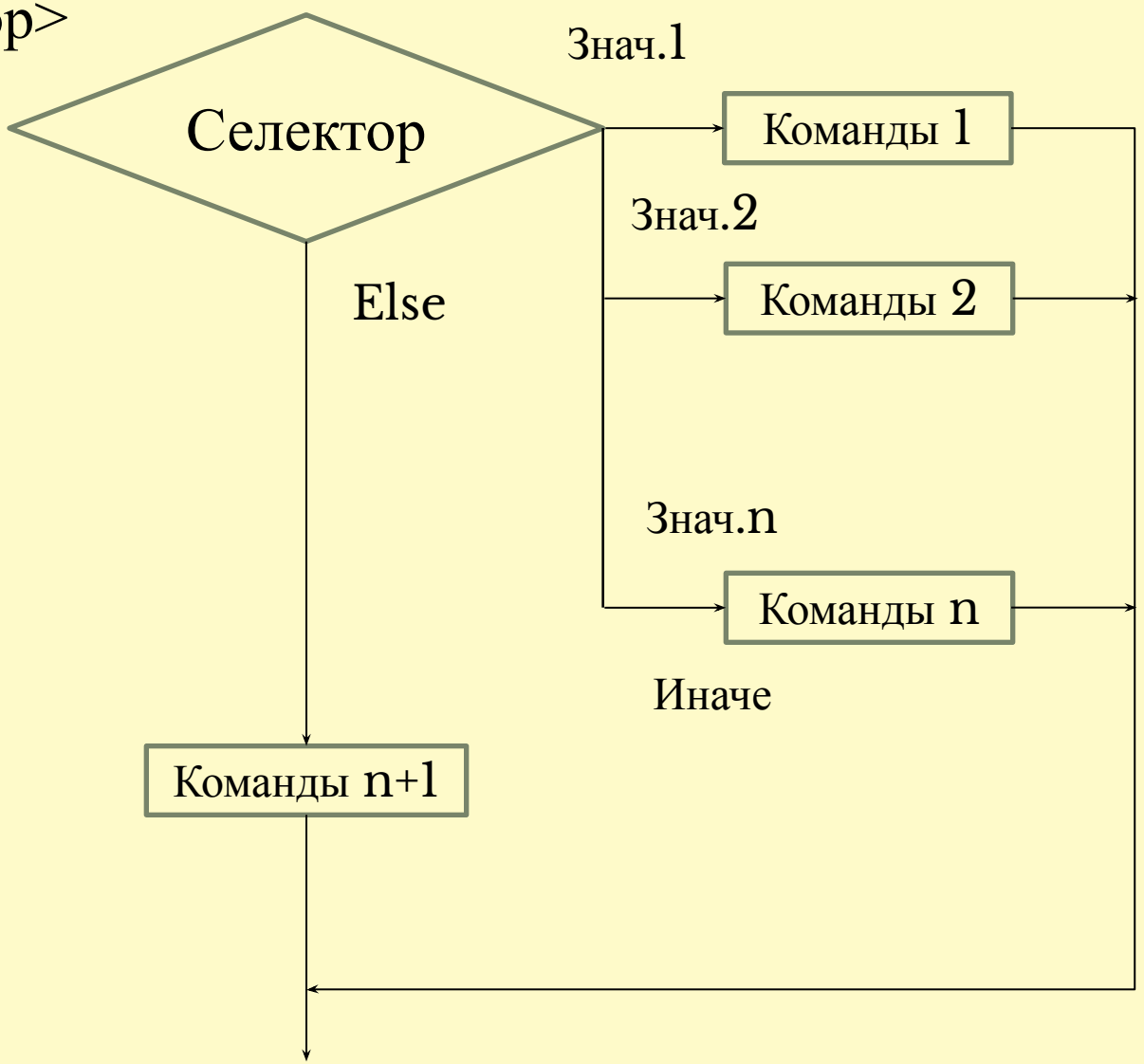


If Условие1 **Then**
    Команды1
ElseIf Условие2 **Then**
    Команды2
Else
    Команды3
End If

Ветвей **ElseIf** может быть несколько

Оператор выбора

```
Select Case <селектор>
Case Значение_1
     Команды1
Case Значение_2
     Команды2
……….
Case Значение_n
     Команды n
Case Else
     Команды n+1
End Select
```

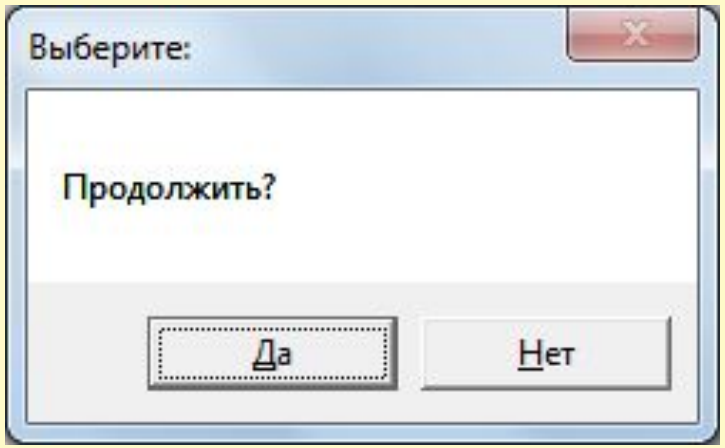


Селектор – это переменная обычно целочисленного или строкового типа

Вывод данных через диалоговое окно

Для вывода данных можно использовать функцию MsgBox:

Результат=MsgBox(“Продолжить?”, vbYesNo, “Выберите:”)



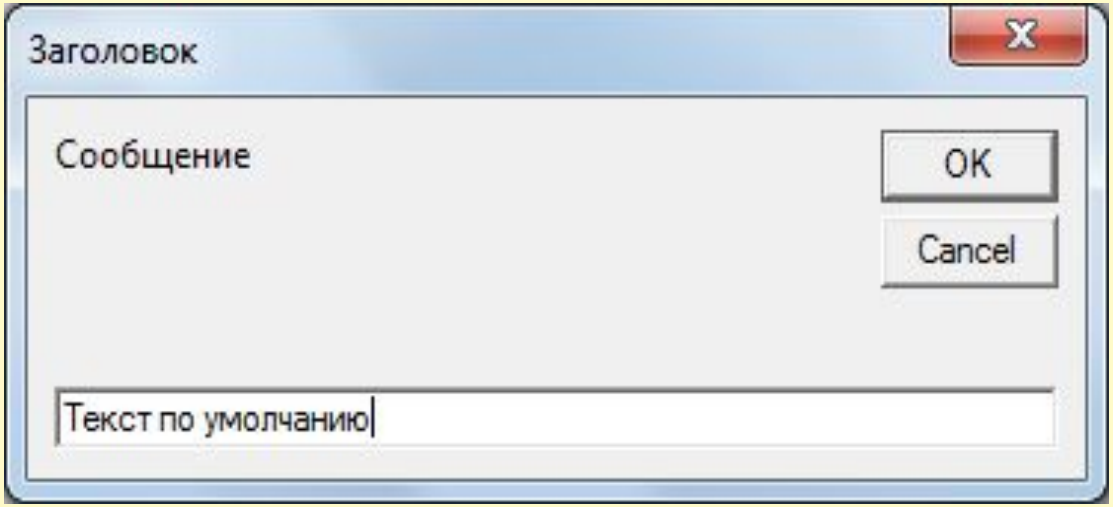
Функция выводит на экран диалоговое окно, содержащее сообщение, устанавливая режим ожидания нажатия кнопки пользователем и возвращает значение типа *Integer*, указывающее, какая кнопка была нажата. Второй и третий аргументы функции могут отсутствовать.

Выражение	Кнопки	Значение
VbOkOnly или не указано	Ок	1
VbYesNo	Да, Нет	6, 7
VbCancel	Прервать, Повтор, Пропустить	3, 4, 5
VbYesNoCancel	Да, Нет, Отмена	6, 7, 2
VbOkCancel	Ок, Отмена	1, 2

Ввод данных через диалоговое окно

Для ввода данных можно использовать функцию InputBox:

Данные=InputBox(“Сообщение”, “Заголовок”, “Текст по умолчанию”)



Функция возвращает в качестве результата данные типа String, даже если вводились только цифры.

Второй и третий аргументы функции могут отсутствовать.

В качестве аргументов можно использовать переменные строкового типа или ссылки на ячейки, например:

Данные=InputBox(“Новые данные”, , Range(“B2”))

Преобразование типа данных

Для конвертации типов данных используются функции, имя которых выглядит как C (от слова Convert) + сокращенное имя типа данных: **CBool(), CByte(), CCur(), CDate(), CDbl(), CDec(), CInt(), CLng(), CSng(), CStr(), CVar().** В качестве аргумента в скобках указывается преобразуемая переменная, константа, ячейка и т.п., например:

Число1=CSng(“123,56”)

Также можно использовать функции:

- *Str(число)* — позволяет перевести числовое значение в строковое. Делает почти то же самое, что и *CStr()*, но при этом вставляет пробел впереди для положительных чисел.
- *Val(строка)* — преобразует переданную строку по возможности в число. При этом функция читает данные слева направо и останавливается на первом нечисловом значении (допускается единственное нечисловое значение — запятая, отделяющая целую часть от дробной).

Обращение к объектам приложения

Диапазоны ячеек, листы, книги и пр. являются объектами. Обращаться к ним можно напрямую или через переменные типа Object. Для присвоения значения таким переменным используется оператор Set.

```
Dim L1 as Object, D1 as Object
Set L1 = WorkSheets(1)
Set D1 = Range(“A1:C3”)
L1.Name=“Первый лист”
D1.Value=1
Range("A4") = WorksheetFunction.Sum(D1)
```

В последней строке вызывается функция MS Excel СУММ (A1:C3).

Цикл с предусловием

While условие

команды

Wend

‘Вычисление 10!

N=1

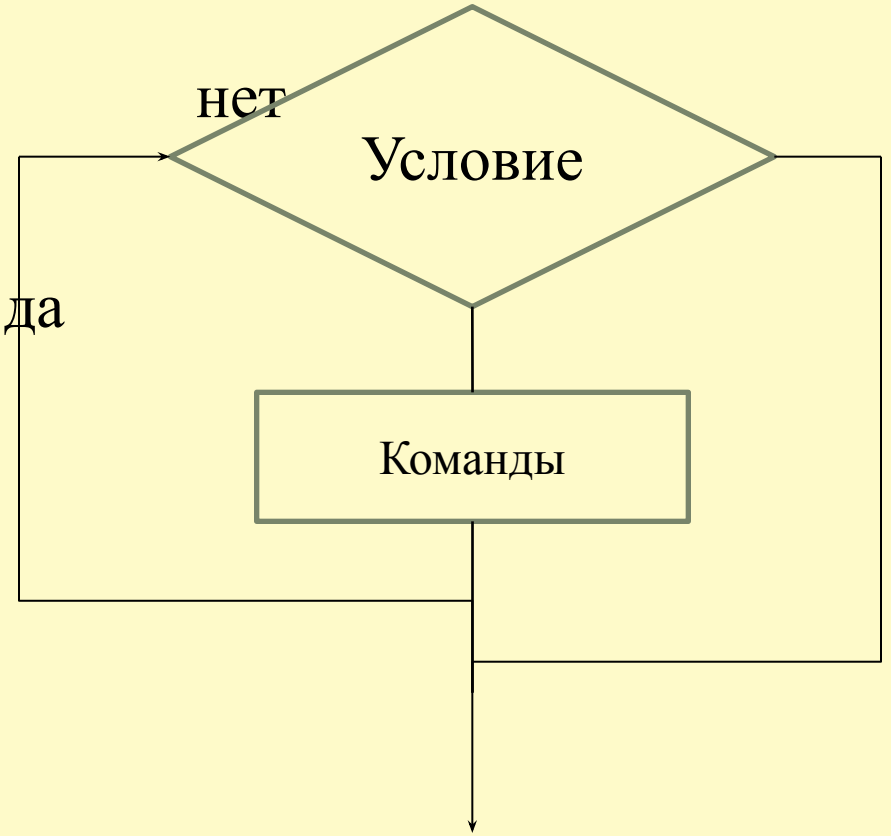
Factor=1

While N<=10

Factor=Factor*N

N=N+1

Wend



Выход из такого цикла возможен только при невыполнении условия после слова While. Прервать выполнение цикла нельзя.

Цикл с предусловием (второй вариант)

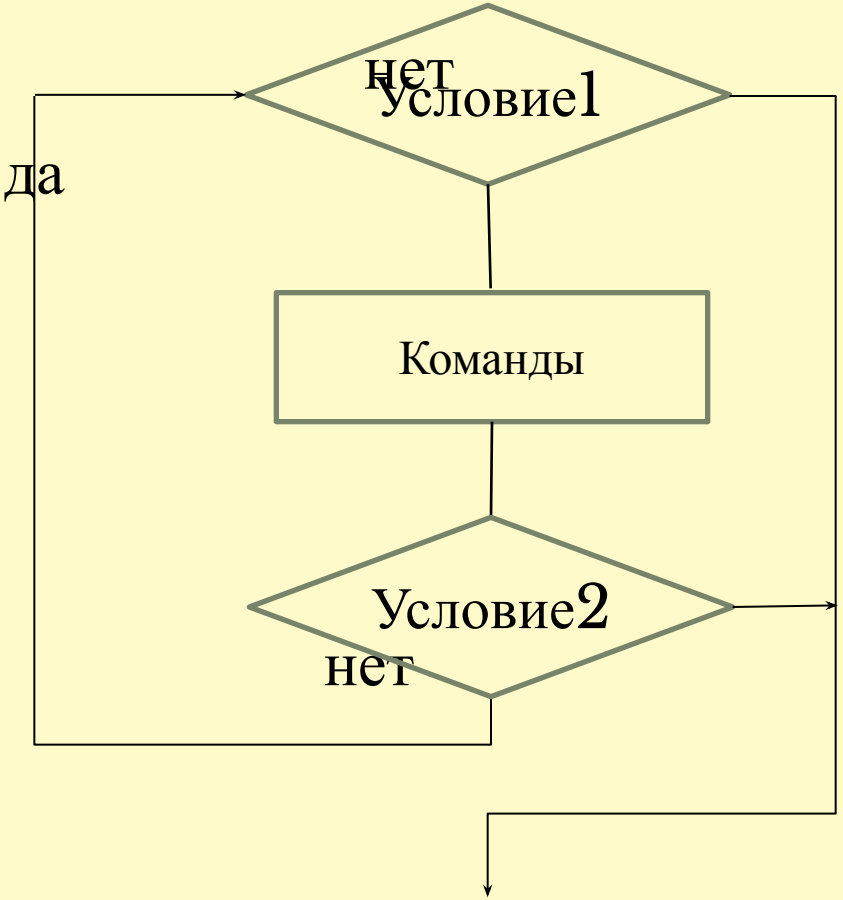
Выход из такого цикла возможен как при невыполнении условия в начале цикла, так и изнутри тела цикла. Условие при входе в цикл можно не задавать.

Do While условие

команды

Loop

```
N=1
Factor=1
Do  While N<100
    Factor=Factor*N
    If Factor>32767 Then
     Exit Do
    EndIf
    N=N+1
Loop
```



Цикл с постусловием

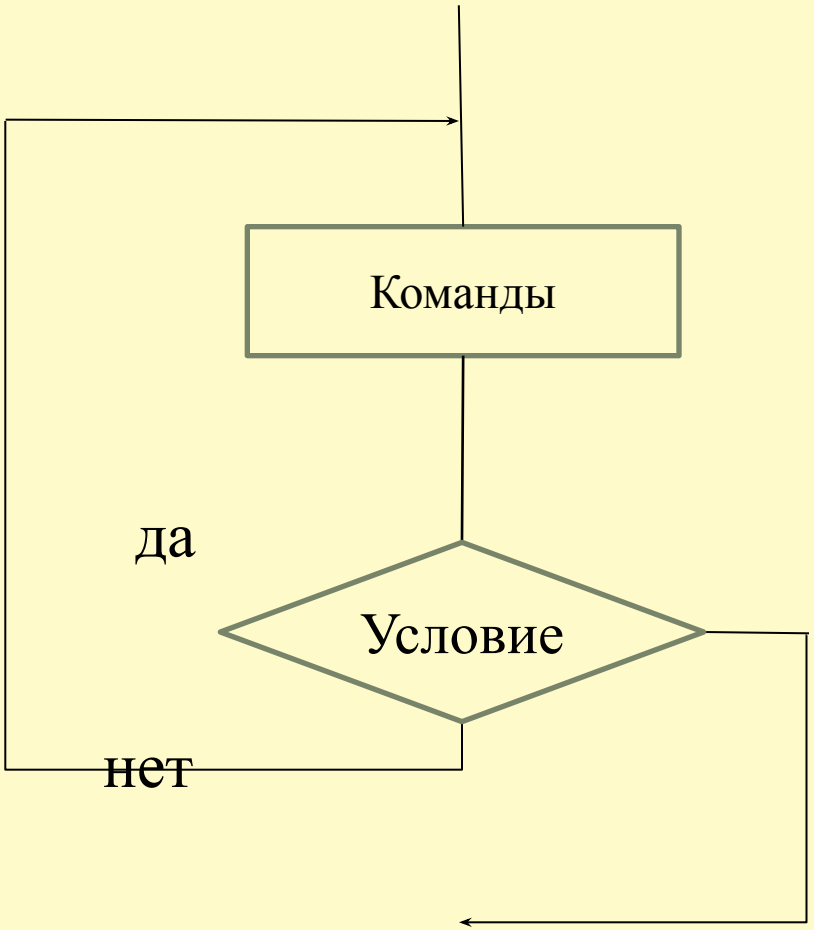
Такой цикл выполнится хотя бы раз. Условие в конце – это условие выхода из цикла.

Do

команды

Loop Until условие

```
N=1
Factor=1
Do
    Factor=Factor*N
    N=N+1
Loop Until N>10
```

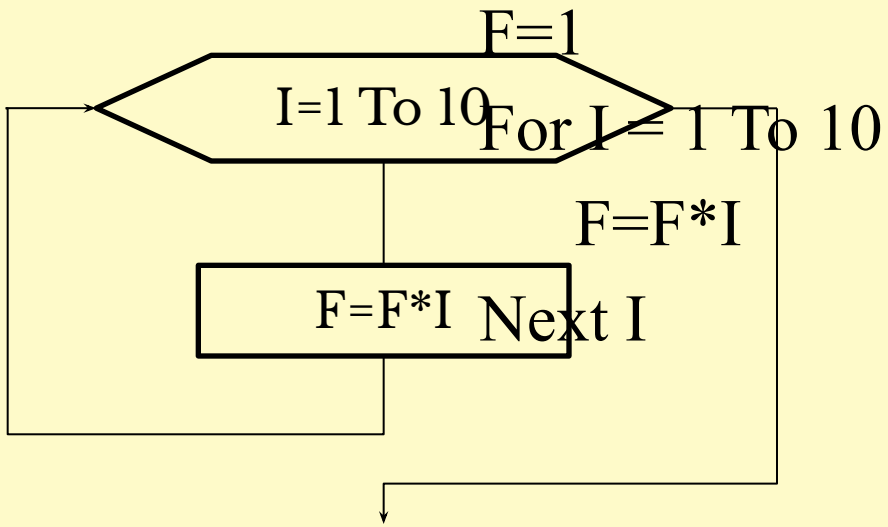


Цикл со счетчиком

For счетчик = начало **To** конец **Step** шаг

команды

Next счетчик



Счетчик может быть как целым, так и действительным числом.

Шаг, равный 1, можно не указывать.

Шаг может быть как положительным, так и отрицательным, как целым, так и действительным числом.

Массивы

Массив – это упорядоченный набор однотипных данных, доступ к которым осуществляется по индексу (номеру).

По умолчанию нумерация элементов массивов начинается с 0.

```
Dim A(9) As Integer ‘Одномерный массив из 10 целых чисел
Dim B(1,2) As String ‘Двумерный массив из 6 строк
```

Можно явно задать диапазон чисел для нумерации:

```
Dim A1(1 To 10) As Byte ’10 элементов типа Byte
Dim B1(1 To 2, 1 To 3) As Object ‘6 ссылок на объекты
```

Обращение к элементам массива:

```
For K=0 To 9
    A(K)=Cells(1,K+1) ‘Запись данных из ячеек A1:J1
Next K
```

Фиксированные и динамические массивы

Фиксированный массив - это массив с заданным размером, который в свою очередь определяет количество элементов.

Динамический массив - это массив с переменным размером, т.е. количество элементов может изменяться во время выполнения программы.

При объявлении (описании) динамического массива его размер не указывается. В процессе выполнения программы его размер может изменяться, причём неоднократно. Поэтому динамический массив применяют, если предполагается, что размер массива не будет постоянным.

```
Dim M1() As Integer ’Объявление динамического массива
```

Размерность динамических массивов

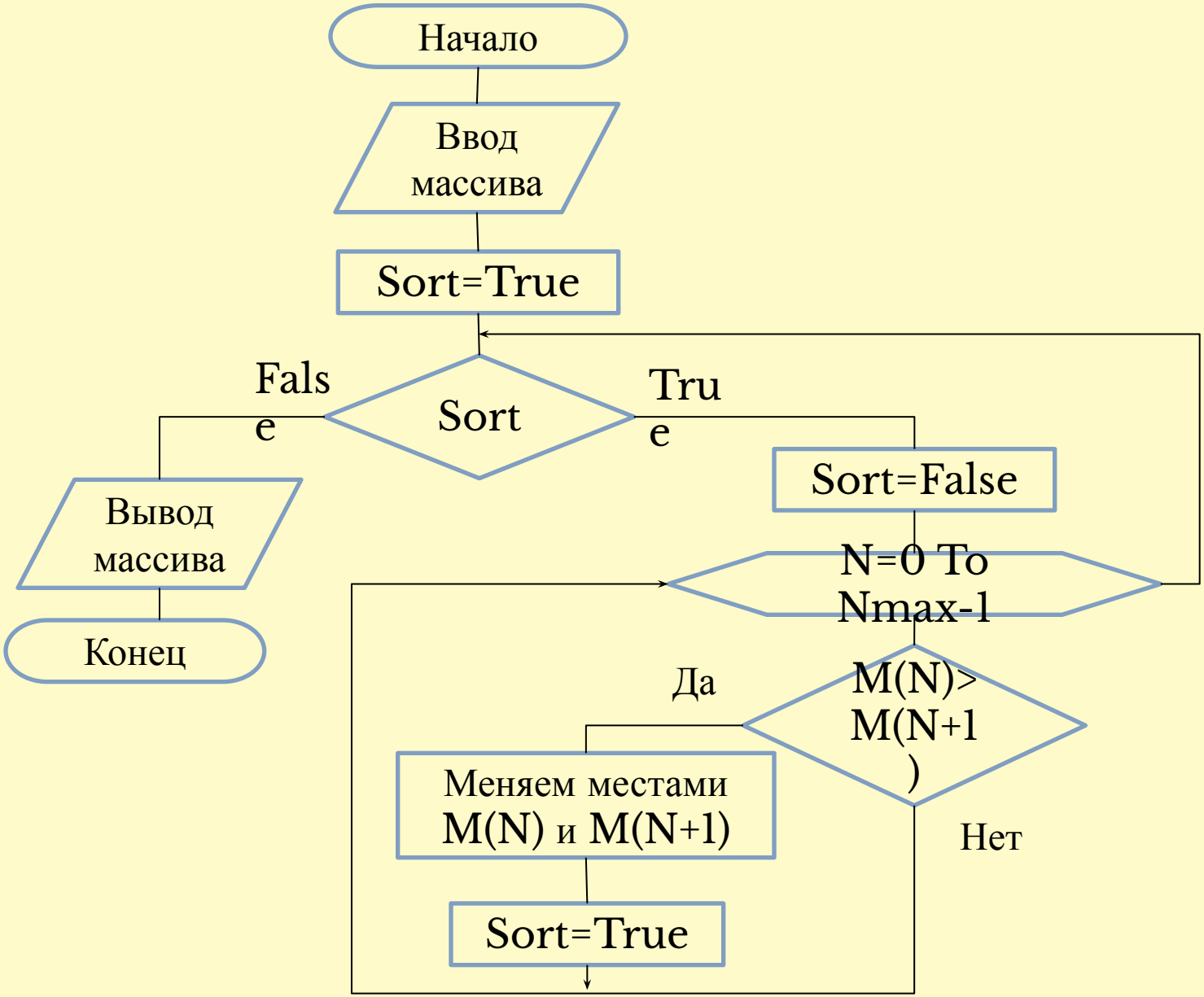
Перед использованием динамического массива его размерность должна быть определена.

При использовании инструкции **ReDim** создается массив указанного размера, при этом имевшиеся ранее в элементах значения не сохраняются:

```
ReDim M1(5) ’Определение  размера массива
For N=0 To 5  ‘ Начало цикла заполнения массива
M1(N)=2*N+1 ‘Очередной элемент
Next N    ‘Конец цикла
ReDim M1(10) ‘Массив расширяется, данные теряются
```

ReDim Preserve M1(10) ‘Массив изменяется с ‘сохранением имеющихся в нем данных.

Сортировка массива методом «пузырька»



Подпрограммы

Процедурой называется фрагмент текста на языке VBA (программный код), заключенный между операторами Sub и End Sub.

Sub имя_процедуры (арг_1, арг_2, ... арг_n)
<инструкция VBA>
…
End Sub

Список аргументов может быть пустым, например:

Sub Сорт().

Вызов такой процедуры в программе:

Call Сорт

или

Сорт

Подпрограммы

Описание процедуры с параметрами:

Sub Trk(r, c) ‘ Без описания типов данных

Sub Trk(r as Single, c as Single) ‘ С указанием типов данных

Вызов в программе:

Call Trk(a,h)’Переменные в качестве параметров

Call Trk(10.2,12.3)’Значения в качестве параметров

Массив в качестве параметра:

Sub Copr1(M() As Integer) ‘Описание процедуры

Вызов в программе:

Call Copr1(Massiv) ‘Massiv – имя массива в программе

Подпрограммы

Функцией называется фрагмент текста на языке VBA (программный код), заключенный между операторами Function и End Function.

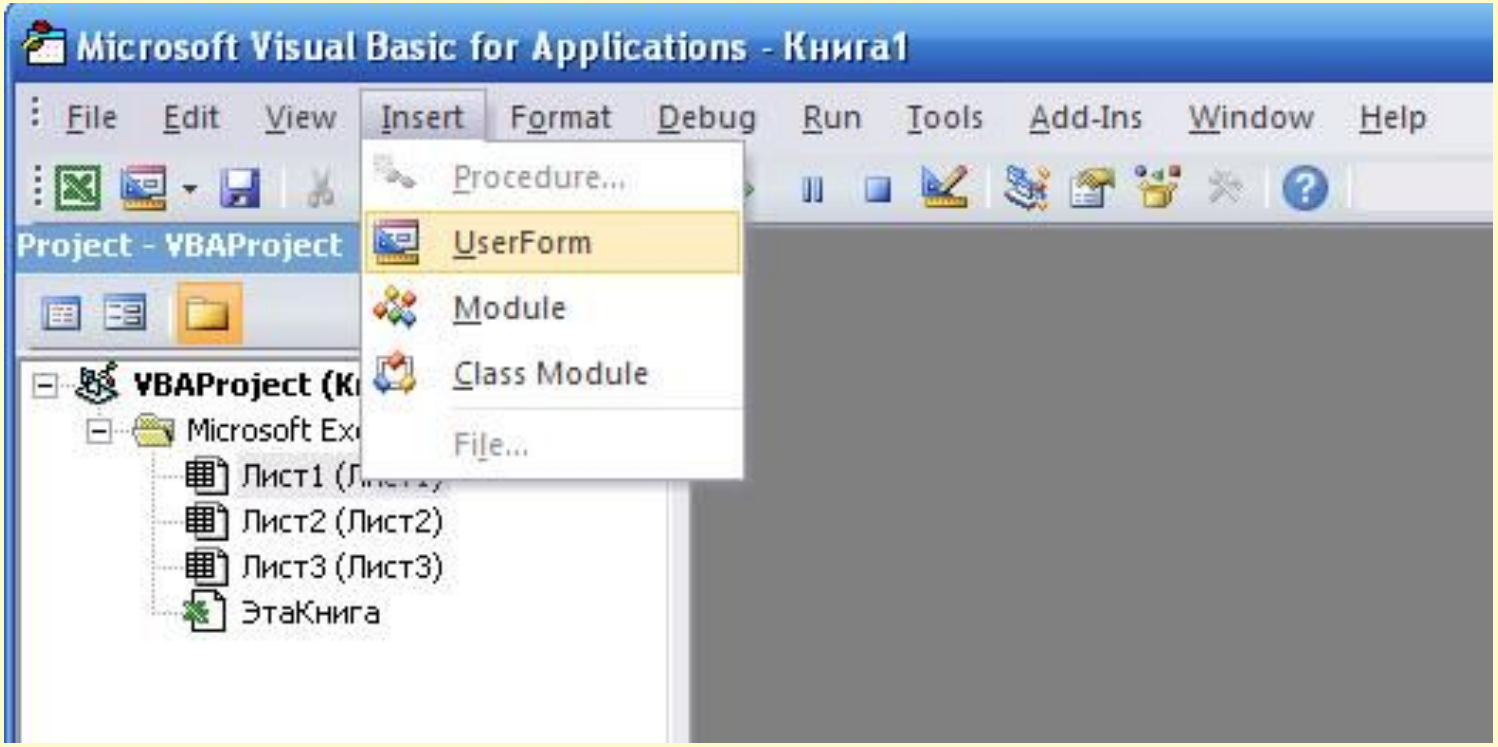
```
Function  имя_функции  (арг_1, ... арг_n)  As
тип_данных
<инструкция  VBA>
…
имя_функции  =  вычисленное_значение
…
End  Function

Function  Fact(N as Integer)  As Integer ‘Описание
функции
```

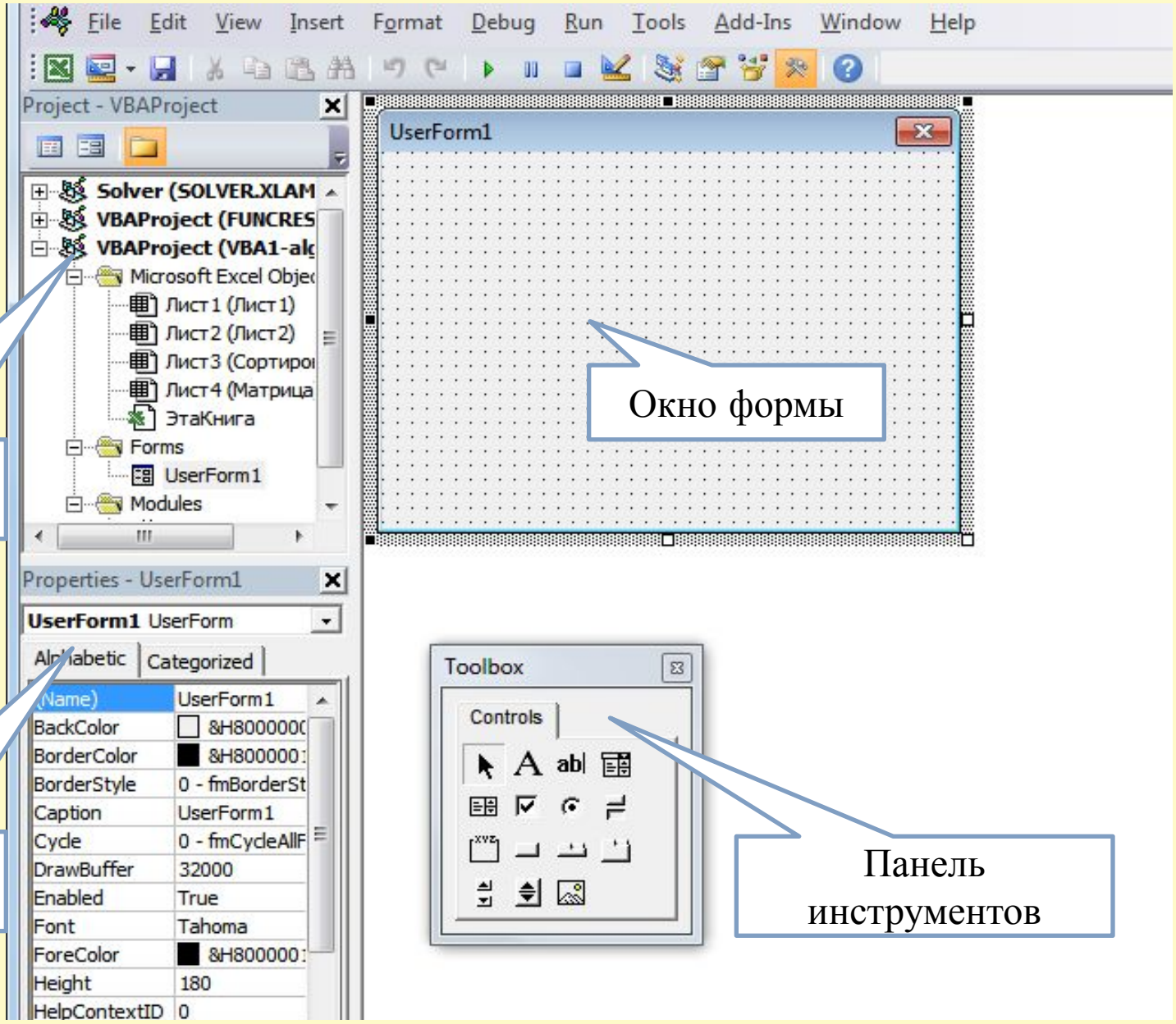
Работа с формой

Форма пользователя– это окно, в котором нужным образом размещаются различные элементы управления и данные.

Создание формы: меню **Insert - UserForm**

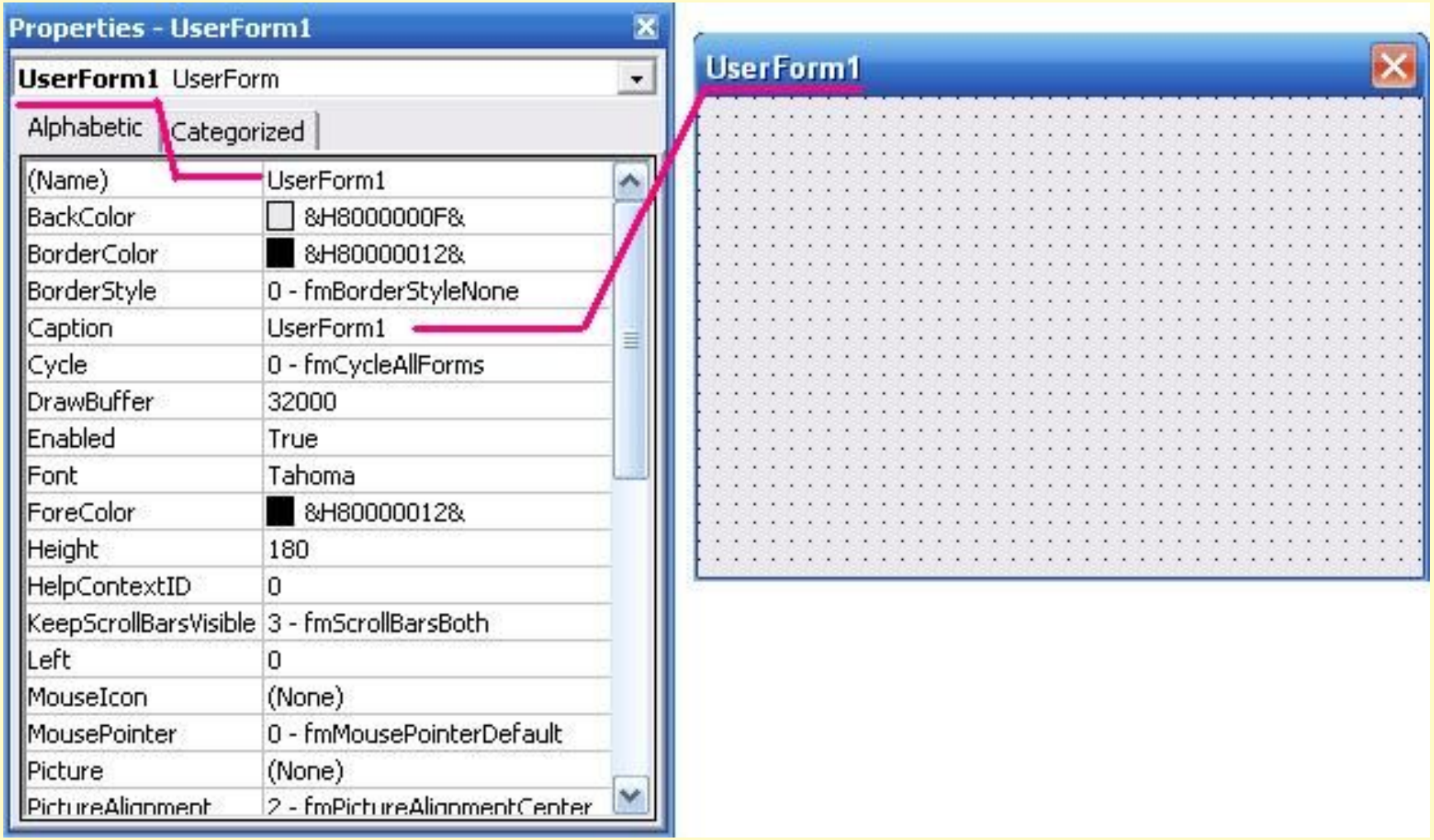


Работа с формой



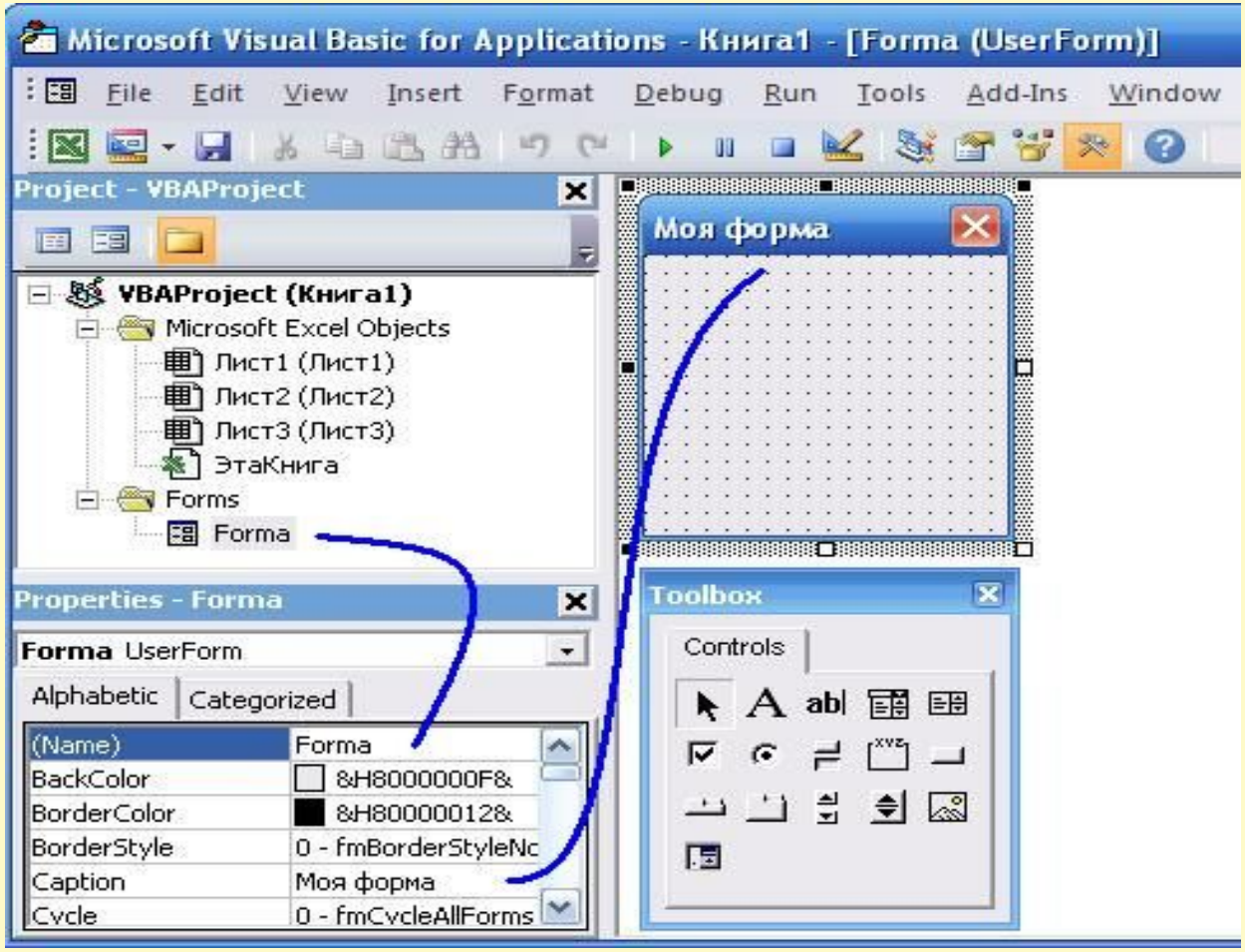
Работа с формой

В окне свойств формы можно указать различные свойства как самой формы, так и любых элементов, размещенных на ней.



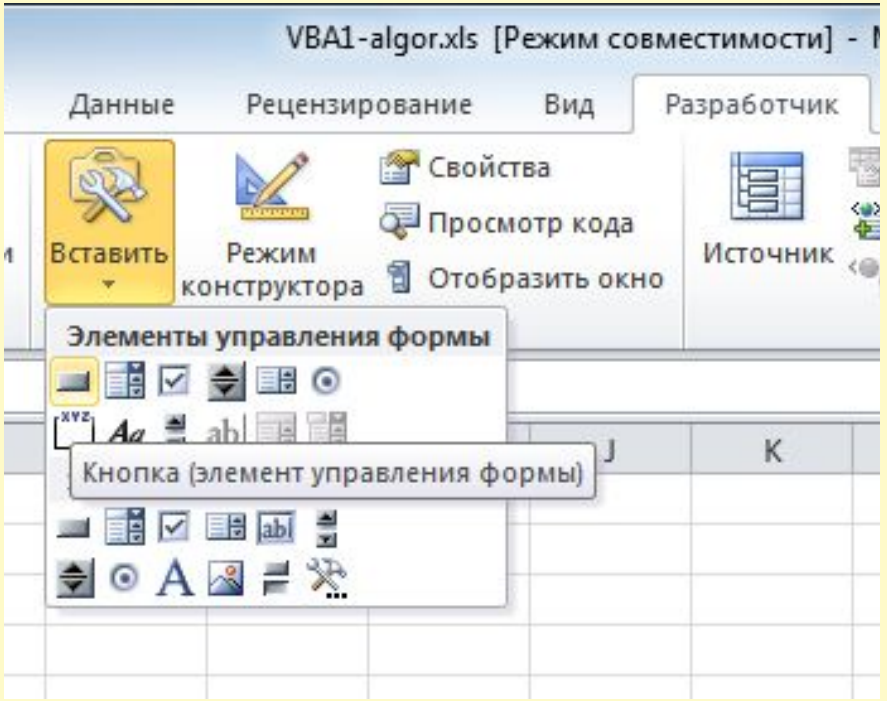
Работа с формой

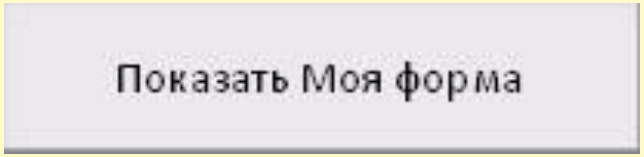
Имя задаётся в окне свойств в строке **Name** - это то имя, к которому необходимо обращаться при работе с формой. Зададим свойство **Name - Forma,** а в строке **Caption** напишем заголовок окна формы **"Моя форма"**.



Работа с формой

Поместим на Лист1 кнопку при помощи, которой будем вызывать форму. Пусть это будет кнопка **"Показать Моя Форма"**. Для этого на вкладке **Разработчик** нажимаем кнопку **Вставить,** выбираем элемент управления **Кнопка** и растягиваем ее на листе. Пишем текст на кнопке:





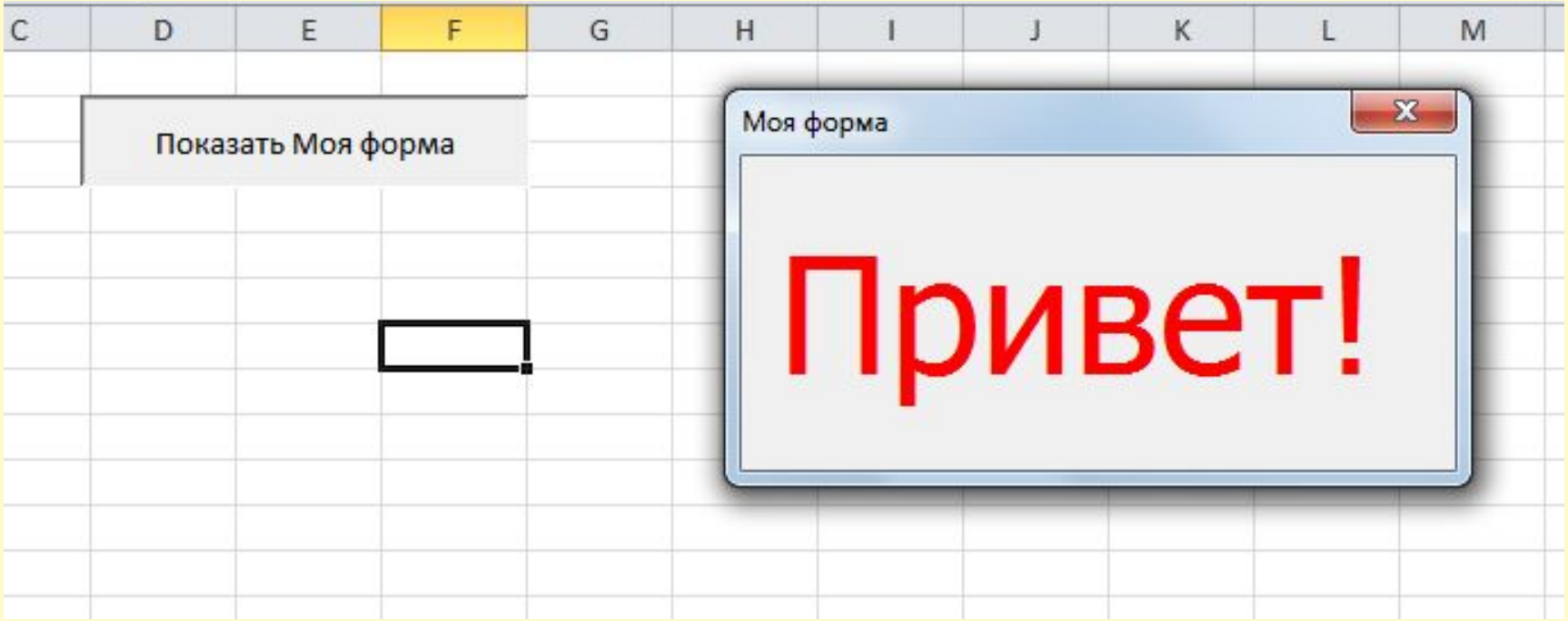
В коде этой кнопки пропишем следующее:

```
Private Sub CommandButton1_Click()
Forma.Show
End Sub
```

Работа с формой

Поместим на форму приветственный текст посредством элемента Label. Выбрав кнопку A, растягиваем область для текста на форме. В качестве свойства Caption этого элемента пишем нужный текст. Настраиваем свойство Font для указания размера и типа шрифта, свойство ForeColor для указания цвета.

При щелчке по кнопке на листе получаем результат:



ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

Функция	Описание	Пример
Len(str)	Определяет длину строки	a=len("Персонажи") =9
Left (<строка>, <длина>)	Выделяет из аргумента <строка> указанное количество символов слева	Left(" 1234string", 4) ="1234"
Right(<строка>, <длина>)	Выделяет из аргумента <строка> указанное количество символов справа	Right("1234string",6) ="string"
Mid(<строка>, <старт> [, <длина>])	Выделяет из аргумента <строка> подстроку с указанным числом символов, начиная с позиции <старт>	Mid ("12345678",4,3) ="456"
Mid(<строка>, <старт>)	Выделяется подстрока от позиции <старт> до конца строки	Mid ("12345678", 4) ="45678"
LTrim (<строка>)	Удаляет пробелы в начале строки	LTrim(" печать") =" печать"
RTrim (<строка>)	Удаляет пробелы в конце строки	RTrim("печать ") =" печать"
Trim (<строка>)	Удаляет пробелы в начале и в конце строки	Trim(“ печать ") =" печать"

ФУНКЦИИ РАБОТЫ СО СТРОКАМИ (поиск и замена)

Функция	Описание	Пример
InStr([<старт>,] < строка 1>, <строка 2>)	Производит поиск подстроки в строке. Возвращает позицию первого вхождения строки <строка 2> в строку <строка 1>, <старт> - позиция, с которой начинается поиск. Если этот аргумент пропущен, поиск начинается с начала строки	Instr("C:\Temp\test.mdb", "Test")=9 Если искомая строка не находится в указанной строке, функция возвращает 0
InStrRev ([<старт>,] <строка 1>, <строка 2>)	Ищет подстроку в строке, но начинает поиск с конца строки и возвращает позицию последнего вхождения подстроки.	
Replace (<строка>, <строка Поиск>, <строка Замена>)	Позволяет заменить в строке одну подстроку другой. Эта функция ищет все вхождения аргумента <строка Поиск> в аргументе <строка> и заменяет их на <строка Замена>	

ФУНКЦИИ РАБОТЫ СО СТРОКАМИ (массивы строк)

Функция	Описание	Пример
Split (<строка> [, <разделитель>])	Преобразует строку в массив подстрок. По умолчанию в качестве разделителя используется пробел. Данную функцию удобно использовать для разбиения предложения на слова.	M=Split(“Это строка из пяти слов”) В массиве М будет создано 5 элементов.
Join (<массив Строк> [, <разделитель>])	Преобразует массив строк в одну строку с указанным разделителем.	S=Join(M,”_”) =“Это_строка_из_пяти_слов”
Filter (<массив Строк>, <строка Поиск>[, <включение>])	Просматривает массив строковых значений и ищет в нем все подстроки, совпадающие с заданной строкой. <включение> - параметр (boolean значение), который указывает, будут ли возвращаемые строки включать искомую подстроку или, наоборот, возвращаться будут только те строки массива, которые не содержат искомой строки в качестве подстроки	M1=Filter(M, ”из”, false) В массиве М1 будет создано 4 элемента (без “из”).

Преобразование и создание строк

Функция	Описание	Пример
LCase (<строка>)	Преобразует все символы строки к нижнему регистру	a=Lcase(“Студент”) =“студент”
UCase (<строка>)	Преобразует все символы строки к верхнему регистру	b=Ucase(“Петров”) =“ПЕТРОВ”
Space (<число>)	Создает строку, состоящую из указанного числа пробелов	
String (<число>, <символ>)	Создает строку, состоящую из указанного в первом аргументе числа символов. Сам символ указывается во втором аргументе.	c=String(5,”+”) =“+++++”

Для сравнения строковых значений можно применять оператор Like, который позволяет обнаруживать неточное совпадение, например выражение “Входной сигнал” Like “Вход*” будет иметь значение True. Другие символы, которые обрабатываются оператором Like в сравниваемой строке:

- ? - любой символ (один);
- #- одна цифра (0-9);
- [<список>] - символ, совпадающий с одним из символов списка;
- [!<список>] - символ, не совпадающий ни с одним из символов списка.