АЛГОРИТМЫ И ПРОГРАММЫ

Доцент кафедры «Бизнес-информатика и информационные технологии», к.пед. н. Валентина Игоревна Кузнецова

ПЛАН:

1. Понятие алгоритма.
2. Основные алгоритмические конструкции.
   a) последовательность;
   b) присвоение;
   c) условный оператор (ветвление);
   d) оператор цикла;
3. Операторы ввода-вывода.
5. Программы и программные единицы.
6. Сборка программ.

1. Понятие алгоритма

Алгоритм – это конечная последовательность инструкций (действий, предписаний), предназначенных для решения поставленной задачи.

Каждый *алгоритм задает функцию*, относящую каждому элементу области применимости соответствующий результат, т.е. область применимости совпадает с областью определения этой функции. Говорят тогда, что *алгоритм вычисляет эту функцию*. Функция, которая вычисляется некоторым алгоритмом, называется *вычислимой*.

Средства записи алгоритмов. Записать алгоритм можно на естественном языке, в виде схемы, на каком-либо (специальном) языке программирования, к числу которых относятся и языки машинных команд, ассемблеры, автокоды.

2. ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ

А) ПОСЛЕДОВАТЕЛЬНОСТЬ

Последовательность операторов означает последовательное их исполнение друг за другом. На блок-схемах эта конструкция изображается стрелкой  .

Б) ПРИСВАИВАНИЕ

Обычный синтаксис оператора присваивания:

<переменная> <знак присваивания> <выражение>,

где <знак присваивания> может иметь вид «:=» или **«=» (как, например, в VBA)**.

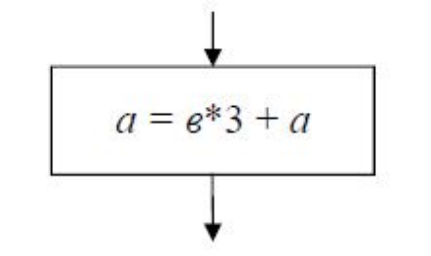
ПРИСВОЕНИЕ

Например, последовательность операторов присваивания (в языке VBA):

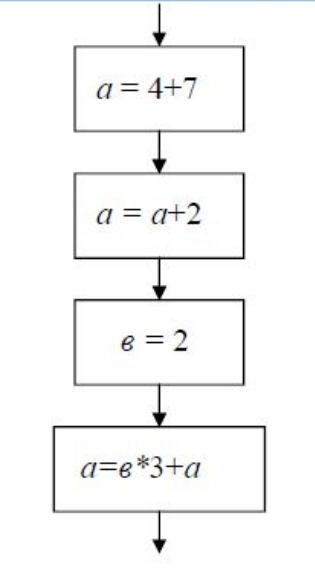
```
a = 4 + 7
a = a + 2
в = 2
a = в*3 + a
```

ПОСЛЕДОВАТЕЛЬНОСТЬ И ПРИСВОЕНИЕ

Например, $a = в*3+a$ выглядит так:
Вся последовательность предыдущего примера изображается так:



Вся последовательность предыдущего примера изображается так:



2. ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ
В) УСЛОВНЫЙ ОПЕРАТОР (ВЕТВЛЕНИЕ)

Эта алгоритмическая структура представляет разветвление алгоритма в зависимости от значения (истинности или ложности) некоторого условия.

В общем виде конструкция выглядит так:

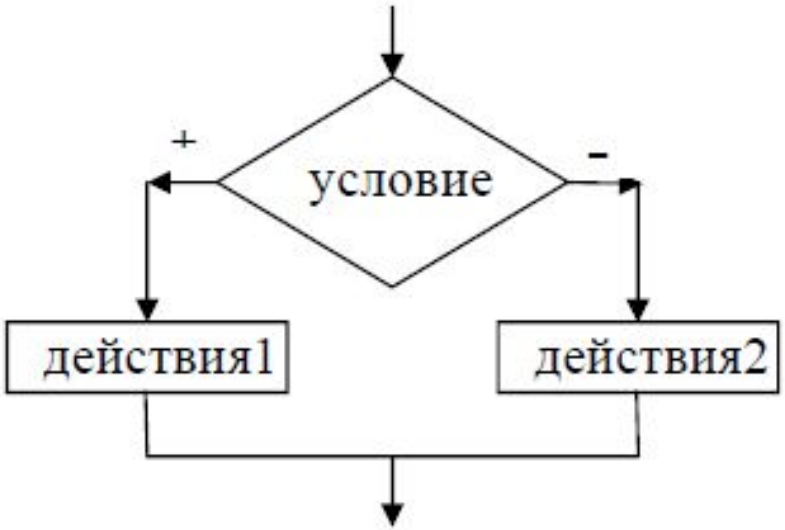
<**если**> <условие> <**то**> <действия1> <**иначе**> <действия2>

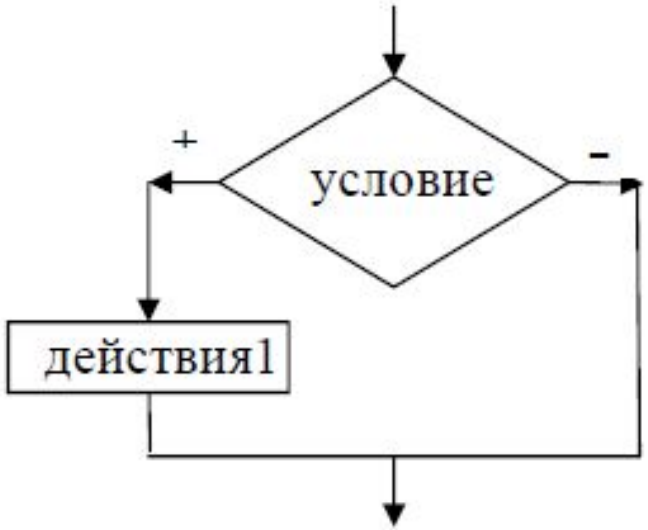
В VBA синтаксис условного оператора:

If <условие> **Then** <действия1> **Else** <действия2> **End If**

Условный оператор может быть неполным, без ветки <иначе> <действия2>.

2. ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ
В) УСЛОВНЫЙ ОПЕРАТОР (ВЕТВЛЕНИЕ)

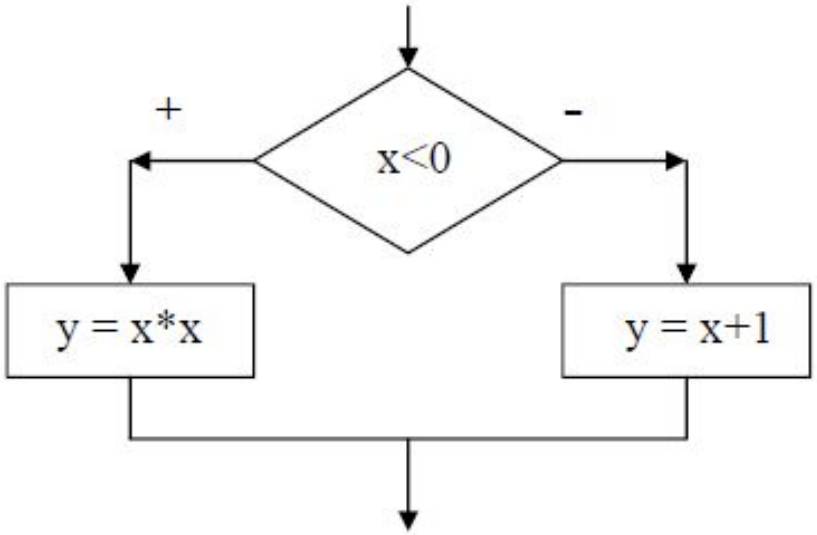




2. ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ
В) УСЛОВНЫЙ ОПЕРАТОР (ВЕТВЛЕНИЕ)

$$y = \begin{cases} x^2 & \text{при } x < 0, \\ x+1, & \text{при } x \geq 0 \end{cases}$$

можно представить следующей блок-схемой:



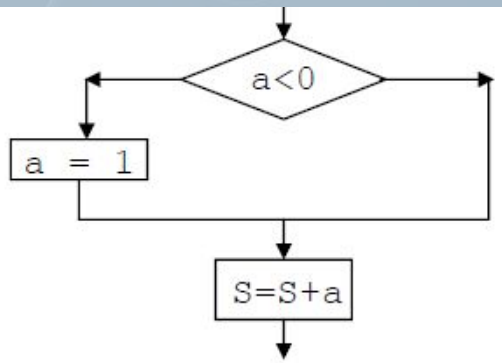
На языке программирования VBA эта конструкция выглядит так:

```
If   x<0   Then
y=x*x
Else
y=x+1
End If
```

2. ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ УСЛОВНЫЙ ОПЕРАТОР (ВЕТВЛЕНИЕ)

Пример использования неполного условного оператора: суммируются числа, вводимые с клавиатуры; если число отрицательное, оно прежде заменяется единицей.

Пусть переменная *a* хранит значение введенного числа, а переменная S хранит сумму введенных чисел. Фрагмент блок-схемы решения такой задачи:

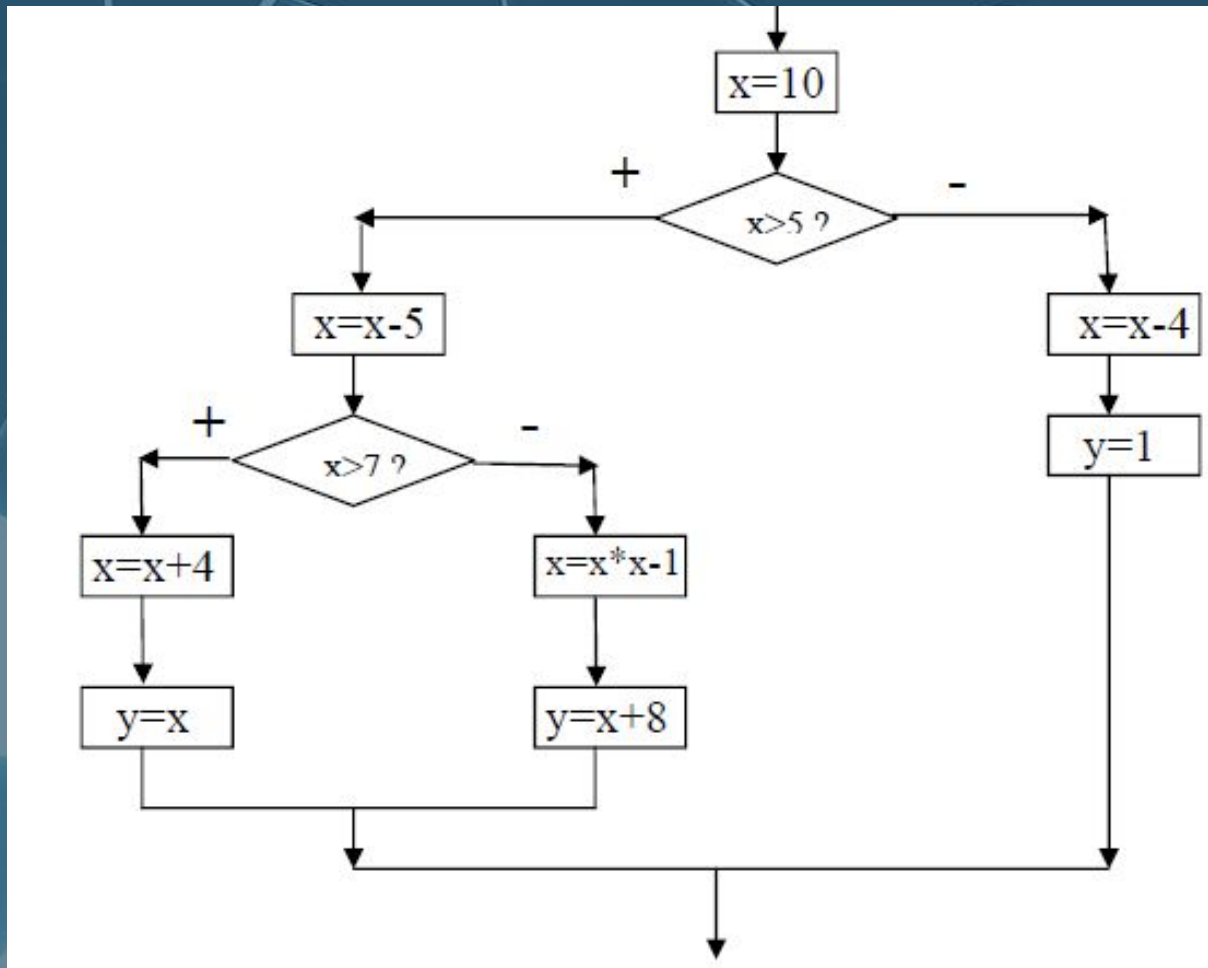


На языке программирования VBA эта конструкция выглядит так:

```
If a<0 Then
a=1
End If
S=S+a
```

2. ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ
В) УСЛОВНЫЙ ОПЕРАТОР (ВЕТВЛЕНИЕ)

```
x=10
If  x>5  Then
    x=x-5
    If  x>7 Then
    x=x+4
    y=x
    Else
    x=x*x-1
    y=x+8
    End If
Else
x=x-4
y=1
End If
```



2. ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ
В) УСЛОВНЫЙ ОПЕРАТОР (ВЕТВЛЕНИЕ)

Какие операции можно включать в условие?

Условие – это логическая формула, значение которой может быть вычислено.

Простейшее условие – это отношение: больше (>), меньше (<), больше или равно (>=), меньше или равно (<=), не равно (<>).

Более сложные условия составляются из простых с помощью операций конъюнкции (в VBA это **And**), дизъюнкции (**Or**), отрицания (**Not**), импликации (**Imp**).

Например, условие x меньше пяти, но больше или равно двум, записывается как (x<5) **And** (x>=2).

2. ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ
Г) ОПЕРАТОР ЦИКЛА

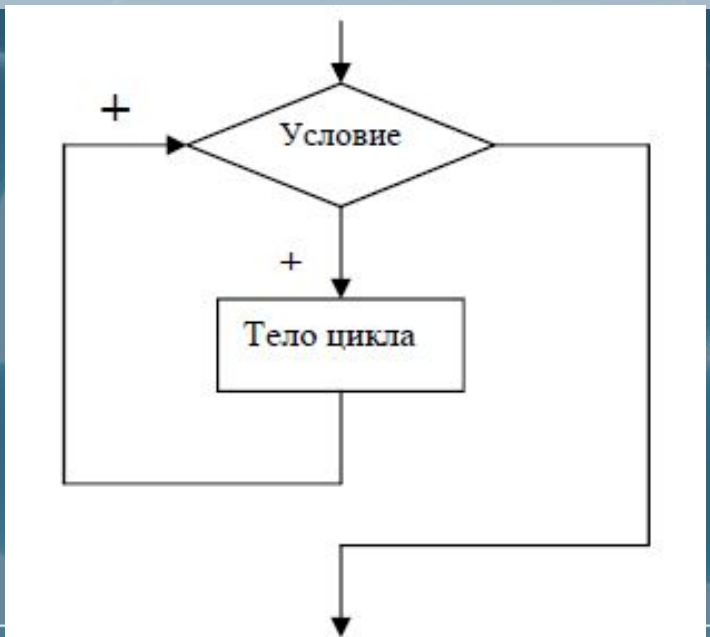
Цикл означает повторяющийся набор действий.

Для обозначения повторений в записи алгоритмов используют конструкции, называемые циклами: *цикл с параметром* (счетчиком*), цикл с предусловием* и *цикл с постусловием*. Цикл с предусловием является наиболее общей конструкцией: этого оператора достаточно, чтобы записать любые циклические действия алгоритмов.

Повторяющаяся группа действий называется **телом цикла**, однократное выполнение этой группы – **шагом цикла**. Часть конструкции, в которой определяется, продолжать выполнение цикла или нет, называют **заголовком цикла**.

Г) ОПЕРАТОР ЦИКЛА
ЦИКЛ С ПРЕДУСЛОВИЕМ

Заголовок этого цикла содержит условие, которое проверяется всякий раз ***перед*** очередным исполнением тела цикла. Если условие истинно, тело исполняется, если ложно, управление передается следующему за циклом оператору в алгоритме. Таким образом, тело цикла исполняется столько раз, сколько раз истинно условие цикла. Блок-схема этого оператора:



Г) ОПЕРАТОР ЦИКЛА
ЦИКЛ С ПРЕДУСЛОВИЕМ

Циклы с предусловием называют обычно циклами типа *While* или циклами **ПОКА** (работает, пока условие выполняется). \

Синтаксис этих циклов в VBA:

While <Условие> <Тело цикла> **Wend**

или

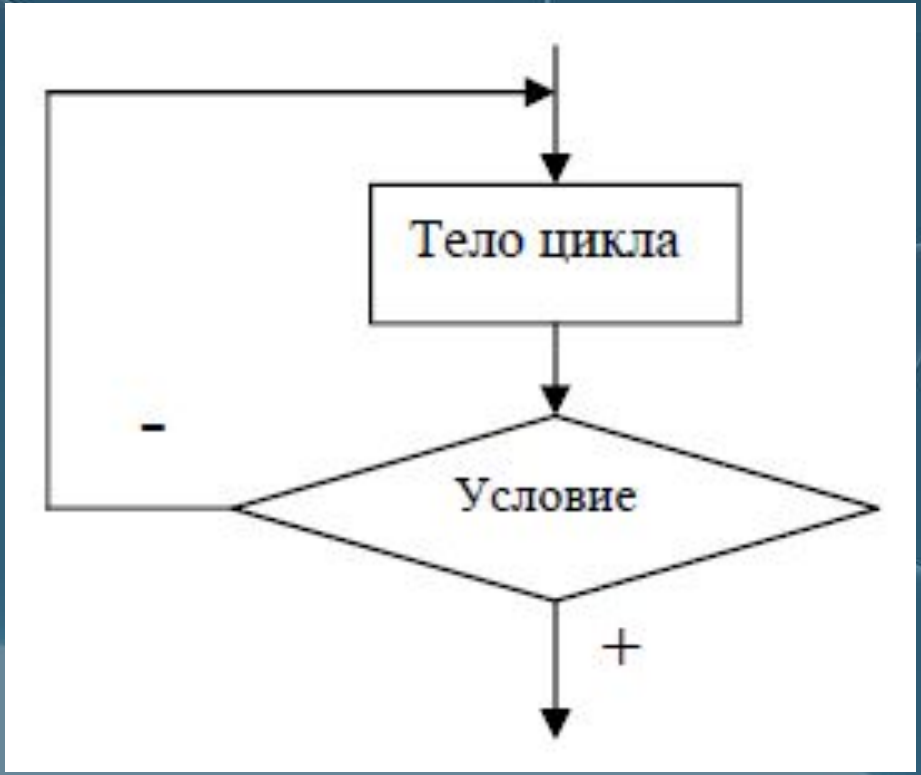
Do While <Условие> <Тело цикла> **Loop**

Например, исполнение фрагмента программы:

```
a = 7
While a > 0
a = a - 1
Wend
```

Г) ОПЕРАТОР ЦИКЛА
ЦИКЛ С ПОСТУСЛОВИЕМ

В таких циклах условие проверяется **после** того, как операторы тела цикла хотя бы раз отработают. Блок схема этого цикла такова:



Г) ОПЕРАТОР ЦИКЛА ЦИКЛ С ПОСТУСЛОВИЕМ

Циклы с постусловием называют **Until**-циклами, циклами **ПОКА НЕ** или циклами **ДО** (работают до того, как условие выполнится).

Синтаксис этих циклов в VBA:

Do <Тело цикла> **Loop Until** <Условие>

Программа предыдущего примера, но с Until-циклом:

```
a = 7
Do
a = a - 1
Loop Until  a < 0
```

Г) ОПЕРАТОР ЦИКЛА
ЦИКЛ С ПАРАМЕТРОМ

Эти циклы используются тогда, когда число повторений известно заранее – количество шагов задано, например, 20, 100, N, или может быть вычислено как результат какого-либо выражения до исполнения цикла.

Параметром в цикле ***является счетчик*** шагов.

Счетчик (это значение специально выделенной переменной) может изменяться на единицу с каждым шагом или получать некоторое заданное приращение, *например*, 0,15. Цикл тогда исполняется до тех пор, пока значение счетчика не достигнет указанного в заголовке цикла значения. Циклы с параметром называют часто *циклами типа For.*

Г) ОПЕРАТОР ЦИКЛА
ЦИКЛ С ПАРАМЕТРОМ

Для VBA синтаксис этого цикла:

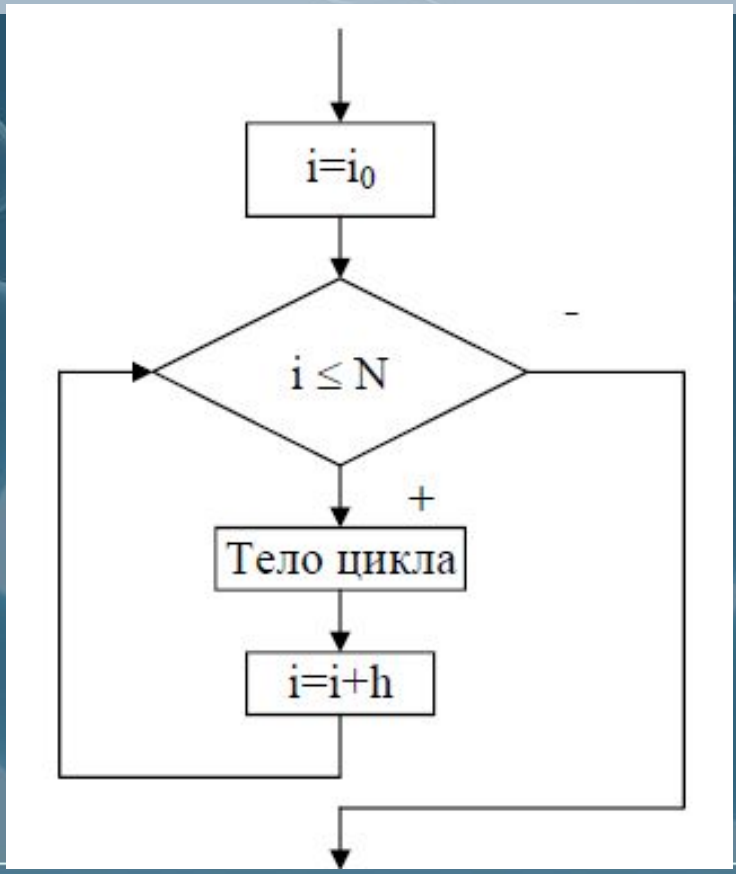
For <переменная-счетчик> = <начальное значение> **To** <конечное значение>
<Тело цикла>
Next

или, если используется счетчик с приращением значения, отличным от 1:

For <переменная-счетчик> = <начальное значение> **To** <конечное значение>
Step <приращение>
<Тело цикла>
Next

Г) ОПЕРАТОР ЦИКЛА
ЦИКЛ С ПАРАМЕТРОМ

Обозначим начальное значение счетчика i как i0, конечное значение – N, приращение – h. Тогда блок-схема этого оператора имеет вид:



Г) ОПЕРАТОР ЦИКЛА
ЦИКЛ С ПАРАМЕТРОМ

Пример использования цикла For в программе VBA:

```
k = 2
m = 4
For i = k + 1 To m * 2 + 1 Step 0.5
k = k + i
Next
```

Обратите внимание, что заголовок цикла For: начальное, конечное значения счетчика и шаг - вычисляется один раз, до первого выполнения тела цикла, и не перевычисляется, несмотря на возможные изменения переменных, входящих в выражения для этих значений.

Г) ОПЕРАТОР ЦИКЛА ДОСРОЧНЫЙ ВЫХОД ИЗ ЦИКЛА

При использовании циклических конструкций может возникнуть необходимость досрочного выхода из цикла.

Например, получен искомый результат, а условие цикла еще истинно и позволяет продолжить исполнение этого оператора. В языке VBA оператором досрочного выхода является **Exit**, причем в циклах **For** он имеет вид **Exit For**, а в циклах, начинающихся с **Do**, он имеет вид **Exit Do.**

Например, программа:

```
a = 7
Do
   a = a – 1
   If a = 5 Then
      Exit Do
   End If
Loop Until a < 0
```

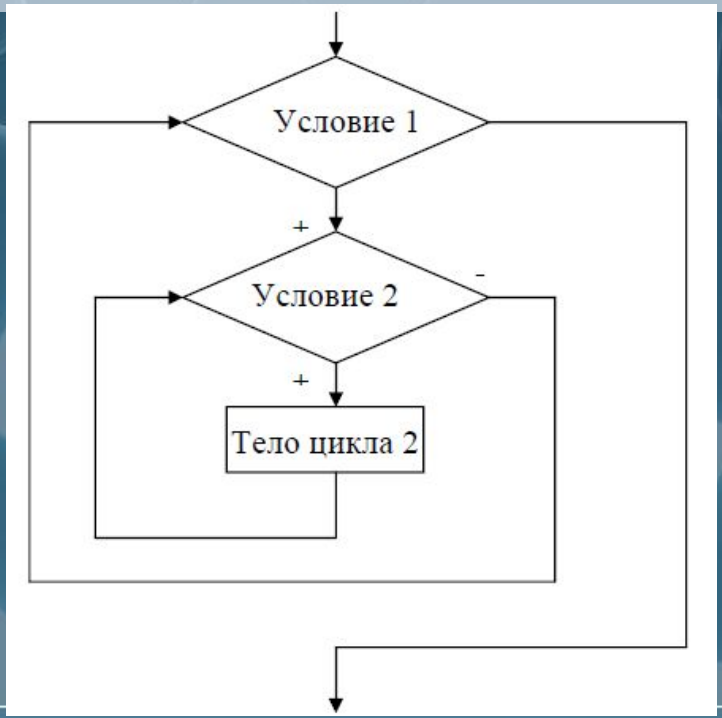
Г) ОПЕРАТОР ЦИКЛА ДОСРОЧНЫЙ ВЫХОД ИЗ ЦИКЛА

Того же результата можно достичь и в программе с циклом **For**:

```
k = 2
m = 4
For i = k + 1 To m * 2 + 1 Step 0.5
    k = k + i
    If  k = 5 Then
        Exit For
    End If
Next
```

Г) ОПЕРАТОР ЦИКЛА ВЛОЖЕННЫЕ ЦИКЛЫ

Тело любого оператора цикла может содержать другие циклы. Такие конструкции называют ***вложенными циклами***. Вложенные циклы (цикл в цикле) применяют обычно в задачах, когда требуется связать или сравнить ***каждый*** элемент одного множества ***с каждым*** элементом другого множества. Блок схема вложенных циклов для, например, циклов с предусловием выглядит так:



Г) ОПЕРАТОР ЦИКЛА ВЛОЖЕННЫЕ ЦИКЛЫ

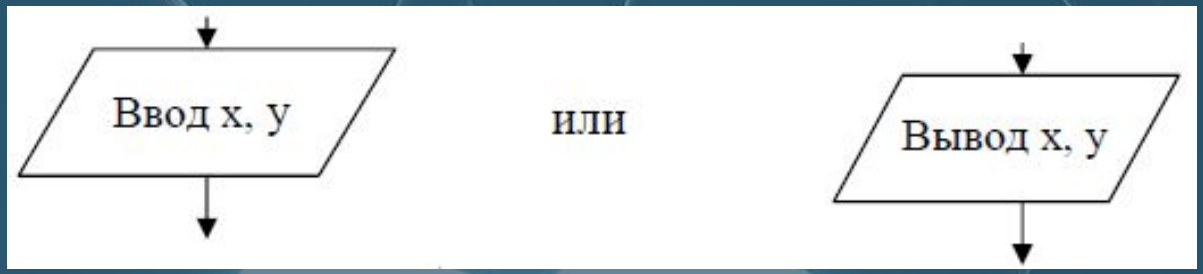
Пример программы с вложенными циклами в VBA:

```
For i = 1 To 7
   For j = 1 To 5
       If  i <= j  Then
       Cells(i, j) = 1
       End If
   Next
Next
```

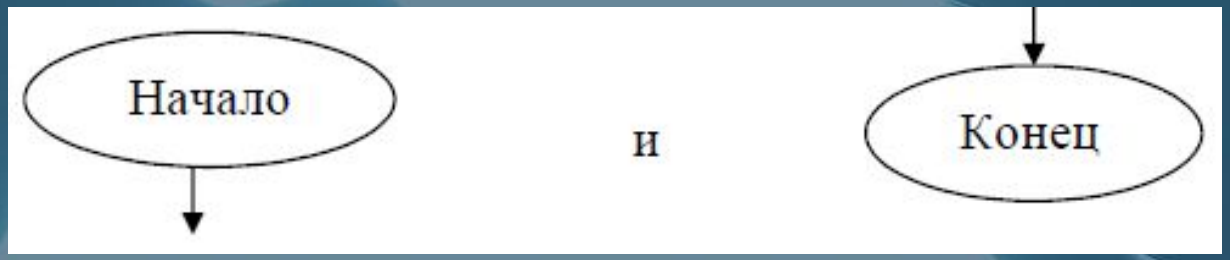
2. ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ
Д) ОПЕРАТОРЫ ВВОДА-ВЫВОДА

Эти действия выполняют ввод и вывод информации (значений переменных) в ходе работы программ.

Для более точного представления алгоритмов в блок-схемах эти операторы имеют специальное изображение, например,



Кроме того, для обозначения начала и конца алгоритма используют фигуры:



2. ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ
Д) ОПЕРАТОРЫ ВВОДА-ВЫВОДА

В языке VBA **ввод** значения переменной *x* можно осуществить с помощью функции **InputBox**, например:

x = **InputBox**(“Введи значение x”)

Вывод в специальное окно экрана можно реализовать оператором **MsgBox**, например:

MsgBox (x)

в итоге на экран выведется только значение переменной *x*, например, 5.

Оператор **MsgBox** “x =”&x
выведет на экран фразу: x = 5.

3. ОПЕРАТОРЫ ВВОДА-ВЫВОДА

Можно осуществить ввод и вывод, используя лист Excel. Ячейки листа имеют имена **Cells(i,j)**, где i – номер строки, j – номер столбца. Например, **ввод значения переменной x из ячейки A1** выглядит в программе так:

x = Cells(1,1)

а **вывод результата – значения x в ячейку, например, C2 (третий столбец)** реализуется так:

Cells(2,3) = x

4. ПРОГРАММЫ И ПРОГРАММНЫЕ ЕДИНИЦЫ

Структура программы: ↓		*Операционная система + компилятор* ↓
Описание переменных – указание их имен и типов данных	⇒	Отводится память ($\mathcal{R}$) под переменные
Ввод данных и начальные присваивания	⇒	Эта память частично заполняется значениями
Операторы, представляющие алгоритм	⇒	Машинные команды, взаимодействуют с памятью $\mathcal{R}$
Вывод результата и/или метка конца программы	⇒	Визуализация данных из $\mathcal{R}$ (печать, экран, специальные носители)

4. ПРОГРАММЫ И ПРОГРАММНЫЕ ЕДИНИЦЫ

При написании программы стоит помнить некоторые ***технологические правила***:

1. При составлении алгоритма сразу определите его вход и выход: исходный набор данных и тот, который требуется получить.
2. Если задача сложная, попытайтесь разбить ее на несколько более простых задач. Для каждой части также определите ее вход и выход.
3. При обнаружении в алгоритме циклических конструкций четко определите условие выхода из цикла и набор повторяющихся операторов – тело цикла. Проверьте, не приводит ли условие к зацикливанию. Проверьте также, не приводит ли приращение счетчика к выходу за пределы массивов, если они используются в цикле.

4. ПРОГРАММЫ И ПРОГРАММНЫЕ ЕДИНИЦЫ

4. Все переменные, используемые в программе, следует описать в начале программы, *до* их использования. Тогда компилятор будет следить за соответствием типов.
5. Имена для переменных лучше выбирать так, чтобы они «говорили» о назначении этих переменных или, хотя бы, о типе их значений. Удобнее читать программу (и исправлять в ней ошибки), в которой вместо «всеядных» x, y, z используются информативные имена «цена», «среднее», «температура» или их понятные сокращения.
6. Переменные, используемые в *правых* частях операторов присваивания, в частности, переменные для накопления значений - счетчики, суммы, произведения - должны предварительно получить какое-то начальное значение.

4. ПРОГРАММЫ И ПРОГРАММНЫЕ ЕДИНИЦЫ

7. Типы левой и правой частей любого оператора присваивания должны быть совместимы. Следя за этим самостоятельно, вы застрахуетесь от «изуродованных» значений, полученных при автоматическом, «по умолчанию», приведении типов.
8. Проверьте «выходы за пределы»: если в задаче предусматриваются какие-либо предельные, граничные значения счетчиков, индексов, интервалов, отдельных величин, проанализируйте (самостоятельно, с помощью оператора If или дополнительной печати промежуточного результата), не превзойдены ли эти границы. Ошибки «пограничных ситуаций» наиболее типичны и трудно уловимы при просмотре текста программы.

4. ПРОГРАММЫ И ПРОГРАММНЫЕ ЕДИНИЦЫ

9. Проверяйте ситуации «дурака»: всегда может найтись пользователь вашей программы, не выполнивший «очевидных» действий. Например, вы просите ввести ненулевое число и надеетесь поэтому, что ввели не ноль. Все надежные программы имеют «защиту от дурака» - операции проверки правильности ввода, предупреждения о возможном неверном ходе выполнения и т.п.

5. СБОРКА ПРОГРАММ

История программирования показывает все большее стремление к модульности в технологиях создания программ. Языки программирования и поддерживающие их системы включают средства разбиения программ на **части (модули)**, отдельной компиляции этих частей и последующей сборки. При сборке программы могут сообщить друг другу свои результаты, и эти результаты могут стать входом других программ. Такую *связь по входу и выходу* обеспечивает механизм входных и выходных параметров, которые можно указать в заголовке программы.

Так, если программа P работает для входных данных X и выдает результат – данные Y, а программа G использует эти данные, вычисляя результат Z, который в свою очередь принимает на вход программа R, то образуется цепочка $P(X)\cdot Y \Rightarrow G(Y)\cdot Z \Rightarrow R(Z)\cdot \ldots$

5. СБОРКА ПРОГРАММ

Очевидно, чтобы такие последовательности вызовов программ работали правильно, необходимо, чтобы типы входных и выходных данных соответствовали друг другу, т.е. были совместимы.

Поэтому сборщику программ для каждой включаемой программы необходимо знать имя программы, тип ее входных данных (говорят, *входных*, input- *параметров*) и тип результата (говорят, *выходных*, output- *параметров*).

Такого рода рассуждения приводят к пониманию программы как данного особого типа, который характеризуется типами входных параметров и типом результата программы.