ЦИКЛЫ В ПАСКАЛЕ

ЦИКЛЫ В ПАСКАЛЕ

- При решении задач может возникнуть необходимость ***повторить одни и те же действия несколько или множество раз***. В программировании блоки кода, которые требуется повторять не единожды, оборачиваются в специальные конструкции – циклы. У циклов выделяют заголовок и тело. Заголовок определяет, до каких пор или сколько раз тело цикла будет выполняться. Тело содержит выражения, которые выполняются, если в заголовке цикла выражение вернуло логическую истину (True, не ноль). После того как достигнута последняя инструкция тела, поток выполнения снова возвращается к заголовку цикла. Снова проверяется условие выполнения цикла. В зависимости от результата тело цикла либо повторяется, либо поток выполнения переходит к следующему выражению после всего цикла.

В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ ПАСКАЛЬ СУЩЕСТВУЕТ ТРИ ВИДА ЦИКЛИЧЕСКИХ КОНСТРУКЦИЙ.

- Цикл с предусловием
  - Способ описания
  - Пример
- Цикл с постусловием
  - Способ описания
  - Пример
- Цикл со счетчиком
  - Способ описания
  - Пример

Разветвляющийся алгоритм If

ЦИКЛ С ПРЕДУСЛОВИЕМ (ЦИКЛ While)

- Цикл while является циклом с предусловием. В заголовке цикла находится логическое выражение. Если оно возвращает true, то тело цикла выполняется, если false – то нет.
- Когда тело цикла было выполнено, то ход программы снова возвращается в заголовок цикла. Условие выполнения тела снова проверяется (находится значение логического выражения). Тело цикла выполнится столько раз, сколько раз логическое выражение вернет true. Поэтому очень важно в теле цикла предусмотреть изменение переменной, фигурирующей в заголовке цикла, таким образом, чтобы когда-нибудь обязательно наступала ситуация false. Иначе произойдет так называемое зацикливание, одна из самых неприятных ошибок в программировании.

К оглавлению

ЦИКЛ С ПРЕДУСЛОВИЕМ СПОСОБ ОПИСАНИЯ

- While <логическое выражение> do

```
begin
 <оператор 1>;
 <оператор 2>;
 <оператор 3>;
 Inc( i )
end;
```

Процедура **Inc** увеличивает значение переменной i на значение Increment. Если Increment не указан, то i увеличивается на единицу.

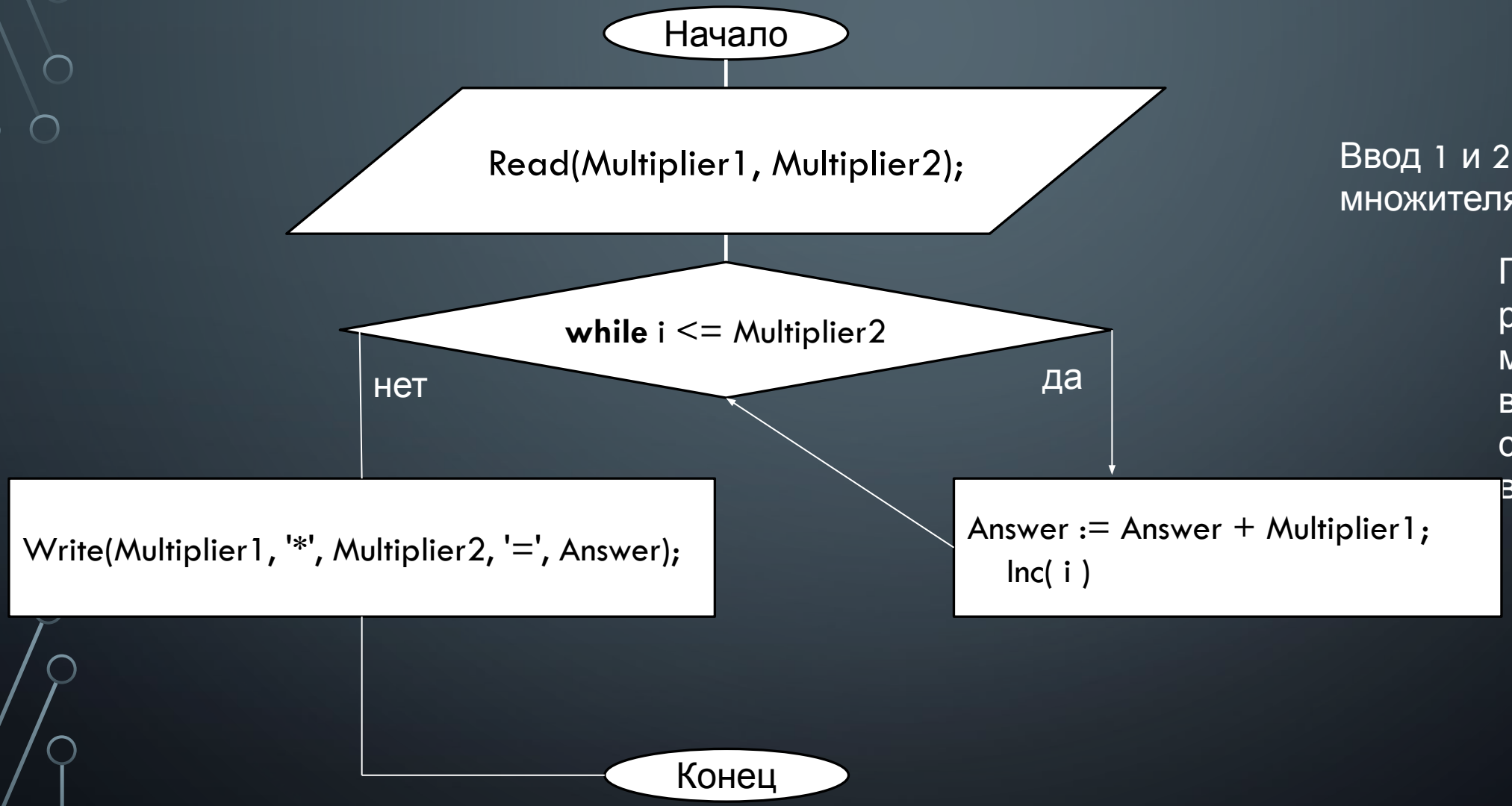
ЦИКЛ С ПРЕДУСЛОВИЕМ (ПРИМЕР)

- Пример: Выполните произведение двух чисел без операции умножения.

```
var
  i, Multiplier1, Multiplier2, Answer: integer;

begin
  Write('Веедите первый множитель ');
  Readln(Multiplier1);
  Write('Введите второй множитель ');
  Readln(Multiplier2);
  i := 1;
  while i <= Multiplier2 do
  begin
    Answer := Answer + Multiplier1;
    inc(i)
  end;
  Write(Multiplier1, '*', Multiplier2, '=', Answer);
end.
```

ЦИКЛ С ПРЕДУСЛОВИЕМ (ПРИМЕР)



Ввод 1 и 2 множителя

Пока i меньше или равно 2 множителя будут выполнятся следующие выражения

ЦИКЛ С ПРЕДУСЛОВИЕМ (ПРИМЕР)

- Допустим, что Multiplier1 := 5 Multiplier2 := 4 => цикл будет выполняться 4 раза (т.к. второй множитель равен 4).

- Заполним трассировочную таблицу к этой задаче:

№ шага	i <= Multiplier2	i	Multiplier1	Multiplier2	Answer
0		1	5	4	0
1	Да	1	5	4	5
2	Да	2	5	4	10
3	Да	3	5	4	15
4	Да	4	5	4	20
5	Нет	5	Конец цикла		

Ответ: 5 * 4 = 20

ЦИКЛ С ПОСТУСЛОВИЕМ (ЦИКЛ Repeat)

- Цикл **while** может не выполниться ни разу, если логическое выражение в заголовке сразу вернуло **false**. Однако такая ситуация не всегда может быть приемлемой. ***Бывает, что тело цикла должно выполниться хотя бы один раз, не зависимо оттого, что вернет логическое выражение***. В таком случае используется цикл repeat – цикл с постусловием.
- В цикле **repeat** логическое выражение стоит после тела цикла. Причем, в отличие от цикла **while**, здесь всё наоборот: в случае **true** происходит выход из цикла, в случае **false** – его повторение.

К оглавлению

ЦИКЛ С ПОСТУСЛОВИЕМ СПОСОБ ОПИСАНИЯ

- Repeat

```
begin
    <оператор 1>;
    <оператор 2>;
    <оператор 3>;
    Inc( i )
end
Until <логическое выражение>
```

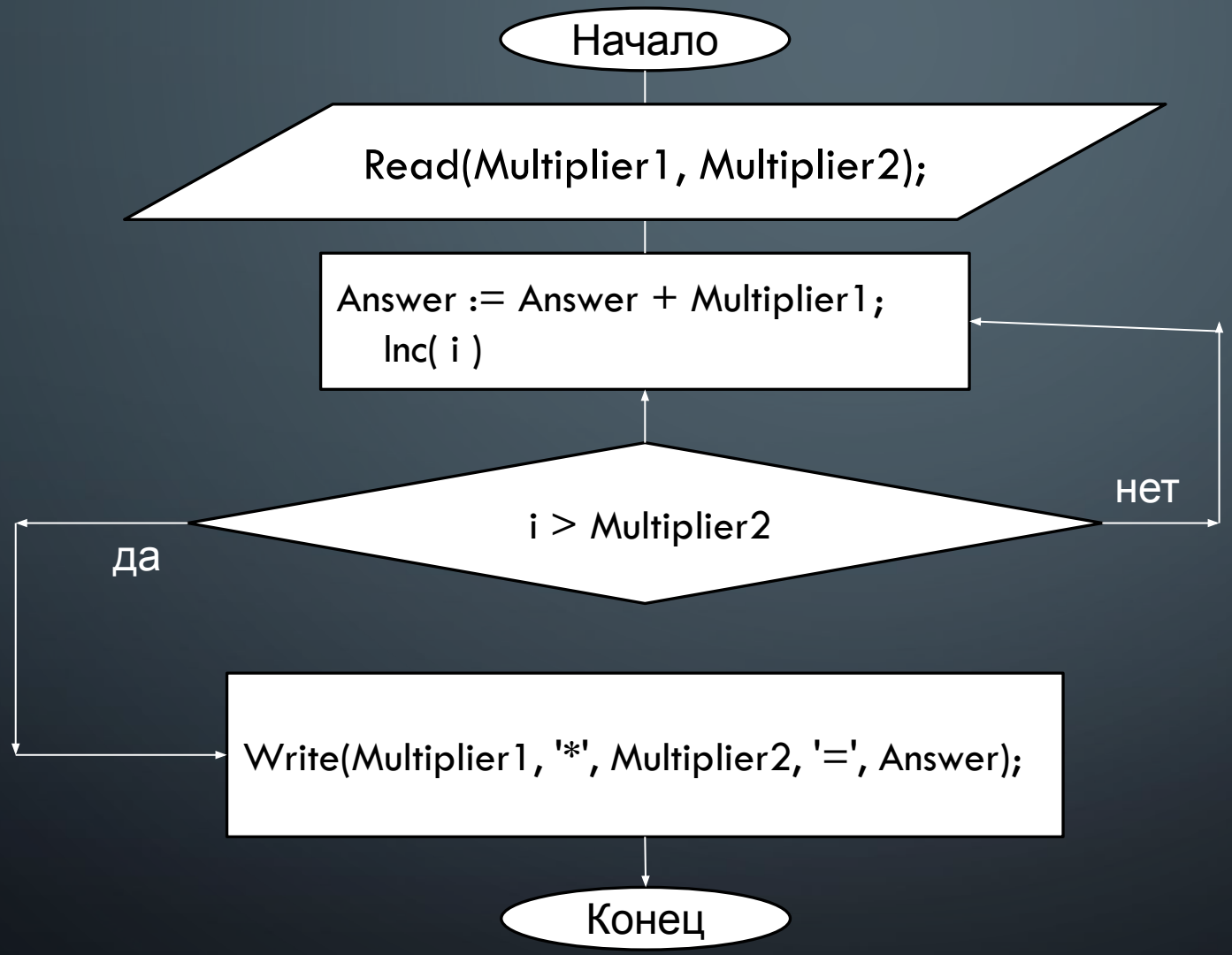
ЦИКЛ С ПОСТУСЛОВИЕМ (ПРИМЕР)

- Пример: Выполните произведение двух чисел без операции умножения.

```
var
  i, Multiplier1, Multiplier2, Answer: integer;

begin
  Write('Веедите первый множитель ');
  Readln(Multiplier1);
  Write('Введите второй множитель ');
  Readln(Multiplier2);
  i := 1;
  repeat
    begin
      Answer := Answer + Multiplier1;
      inc(i)
    end;
  until i > Multiplier2;
  Write(Multiplier1, '*', Multiplier2, '=', Answer);
end.
```

ЦИКЛ С ПОСТУСЛОВИЕМ (ПРИМЕР)



Ввод 1 и 2 множителя

Пока i больше 2 множителя будут выполнятся следующие выражения

ЦИКЛ С ПОСТУСЛОВИЕМ (ПРИМЕР)

- Допустим, что Multiplier1 := 5 Multiplier2 := 4 => цикл будет выполняться 4 раза (т.к. второй множитель равен 4).
- Заполним трассировочную таблицу к этой задаче:

№ шага	i > Multiplier2	i	Multiplier1	Multiplier2	Answer
0		1	5	4	0
1	Нет	1	5	4	5
2	Нет	2	5	4	10
3	Нет	3	5	4	15
4	Нет	4	5	4	20
5	Да	5	Конец цикла		

Ответ: 5 * 4 = 20

ЦИКЛ СО СЧЕТЧИКОМ (ЦИКЛ For)

- Этот цикл используется, когда число повторений ***не связано с тем, что происходит в теле цикла***. Т.е. количество повторений может быть вычислено заранее (хотя оно не вычисляется).
- В заголовке цикла указываются два значения. Первое значение присваивается так называемой переменной-счетчику, от этого значения начинается отсчет количества итераций (повторений). Отсчет идет всегда с шагом равным единице. Второе значение указывает, при каком значении счетчика цикл должен остановиться. Другими словами, количество итераций цикла определяется разностью между вторым и первым значением плюс единица. В Pascal тело цикла не должно содержать выражений, изменяющих счетчик.

К оглавлению

ЦИКЛ СО СЧЕТЧИКОМ СПОСОБ ОПИСАНИЯ

Цикл for существует в двух формах:

- For <счетчик:=значение> To <конечное_значение> Do

  begin

  <оператор1>;

  <оператор2>;

  end;

  Тело цикла

- For <счетчик:=значение> Downto <конечное_значение> Do

  <тело_цикла>;

Счетчик – это переменная любого из перечисляемых типов (целого, булевого, символьного, диапазонного, перечисления). Начальные и конечные значения могут быть представлены не только значениями, но и выражениями, возвращающими совместимые с типом счетчика типы данных. Если между начальным и конечным выражением указано служебное слово to, то на каждом шаге цикла значение параметра будет увеличиваться на единицу. Если же указано downto, то значение параметра будет уменьшаться на единицу.

Количество итераций цикла for известно именно до его выполнения, но не до выполнения всей программы. К началу выполнения цикла, уже точно известно, сколько раз выполниться цикл.

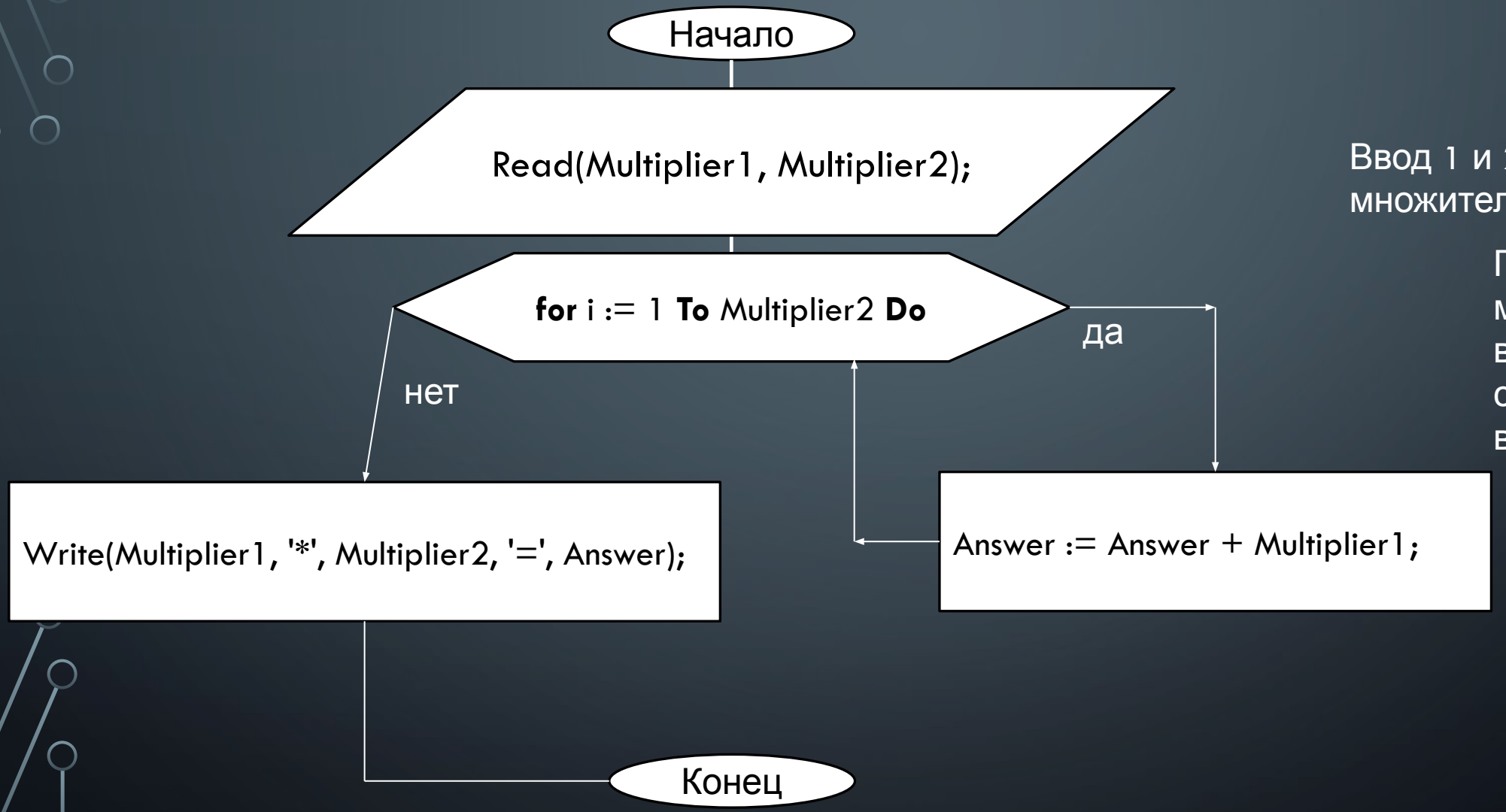
ЦИКЛ СО СЧЕТЧИКОМ (ПРИМЕР)

- Пример: Выполните произведение двух чисел без операции умножения.

```
var
  i, Multiplier1, Multiplier2, Answer: integer;

begin
  Write('Веедите первый множитель ');
  Readln(Multiplier1);
  Write('Введите второй множитель ');
  Readln(Multiplier2);
  for i := 1 To Multiplier2 Do
    Answer := Answer + Multiplier1;
  Write(Multiplier1, '*', Multiplier2, '=', Answer);
end.
```

ЦИКЛ СО СЧЕТЧИКОМ (ПРИМЕР)



Ввод 1 и 2 множителя

Пока i не больше 2 множитель будут выполнятся следующие выражения

ЦИКЛ СО СЧЕТЧИКОМ (ПРИМЕР)

- Допустим, что Multiplier1 := 5 Multiplier2 := 4 => цикл будет выполняться 4 раза (т.к. второй множитель равен 4).
- Заполним трассировочную таблицу к этой задаче:

№ шага	for i := 1 To Multiplier2 Do	i	Multiplier1	Multiplier2	Answer
0		1	5	4	0
1	Да	1	5	4	5
2	Да	2	5	4	10
3	Да	3	5	4	15
4	Да	4	5	4	20
5	Нет	5	Конец цикла		

Ответ: 5 * 4 = 20

РАЗВЕТВЛЯЮЩИЙСЯ АЛГОРИТМ IF

- *Ветвление* – это команда алгоритма, в которой делается выбор, выполнять или не выполнять какую-нибудь группу команд в зависимости от условий.

Ветвление используется в двух случаях:

- 1) Когда требуется пропустить определенную команду или группу команд.
- 2) Когда нужно записать выбор тех или иных действий в зависимости от условия.

К оглавлению

РАЗВЕТВЛЯЮЩИЙСЯ АЛГОРИТМ IF СПОСОБ ОПИСАНИЯ

```
If <условие>
 then
  <оператор 1>;
  <оператор 2>
 else
  <операторы>
```

- Логическая операция ***AND*** (И), поставленная между двумя условиями, говорит о том, что должны выполняться сразу оба эти условия (должны быть истинными). Логический смысл операции - "конъюнкция".
- Поставленный между двумя условиями, знак ***OR*** (ИЛИ) говорит о том, что достаточно, если будет выполняться хотя бы одно из них (одно из двух условий истинно). Логический смысл операции - "дизъюнкция".
- На языке Паскаль ***XOR*** - знак логической операции, имеющий смысл "строгая дизъюнкция" и указывающий на то, что необходимо, чтобы одно из двух условий выполнялось (истинно), а другое - не выполнялось (ложно).
- Логическая операция ***NOT*** перед логическим выражением или переменной имеет смысл "отрицание" или "инверсия" и указывает на то, что если данная переменная или выражение истинны, то их отрицание — ложь и наоборот.

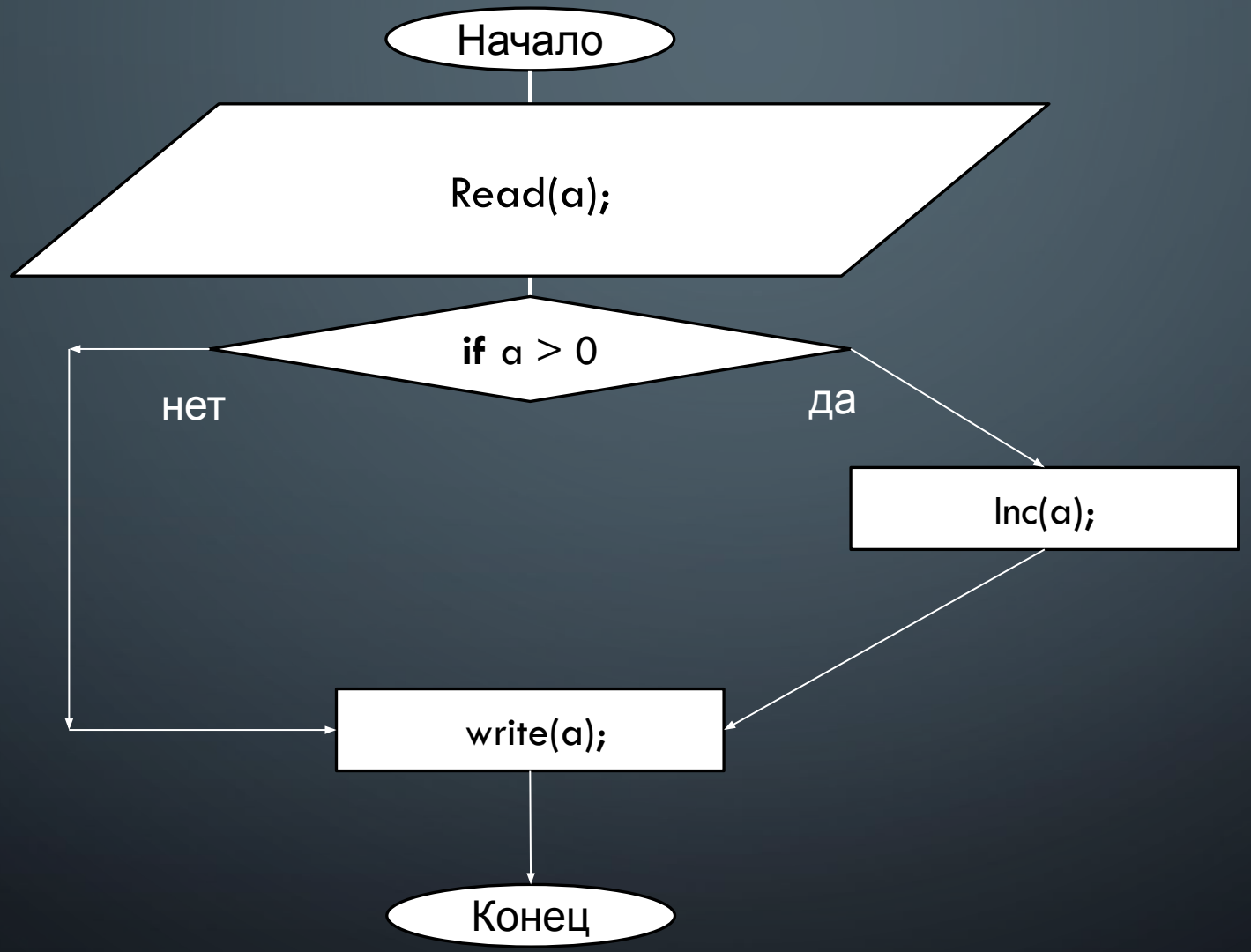
РАЗВЕТВЛЯЮЩИЙСЯ АЛГОРИТМ IF (ПРИМЕР)

- Пример: Дано целое число. Если оно является положительным, то прибавить к нему 1; в противном случае не изменять его. Вывести полученное число.

```
var
  a: integer;

begin
  readln(a);
  if a > 0
    then
    Inc(a);
  write(a);
end.
```

РАЗВЕТВЛЯЮЩИЙСЯ АЛГОРИТМ IF (ПРИМЕР)



Ввод 1 и 2 множителя

Если а положительное число то к нему прибавиться 1

ЗАДАНИЯ:

- 1) Возведите число в степень (без операции умножения и не использую exp(y*(In (x)))) при помощи всех трех циклов.
- 2) Даны два целых числа A и B (A < B). Найти сумму всех целых чисел от A до B включительно.
- 3) Даны натуральные числа от 20 до 50. Напечатать те из них, которые делятся на 3, но не делятся на 5.

К оглавлению