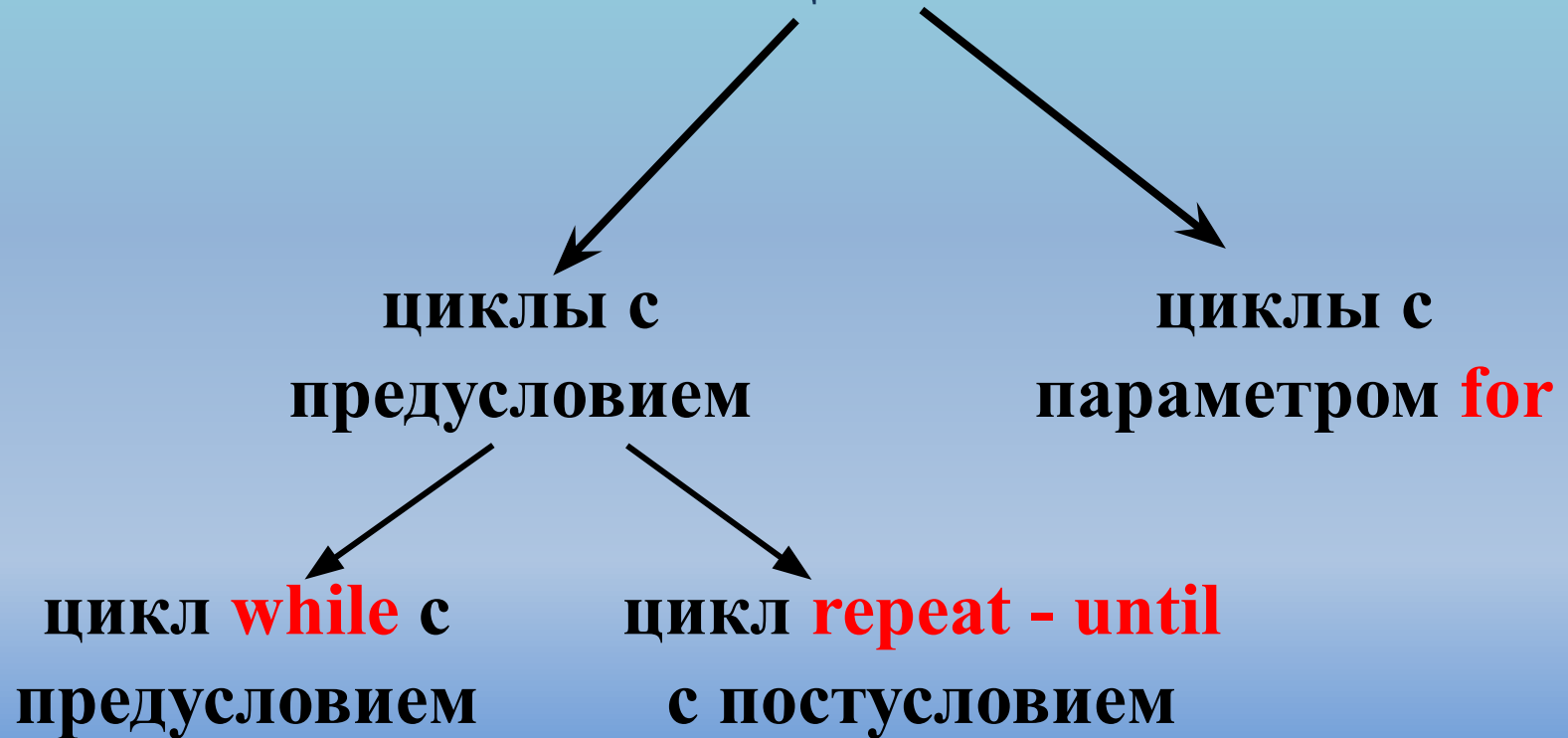


# ЦИКЛЫ В Pascal

**Автор:** учитель информатики  
Волкова Екатерина Сергеевна

# Типы циклов



# Цикл с предусловием в Паскале - WHILE

*Оператор цикла с предусловием выполняет действия заранее неизвестное число раз. Выход из цикла осуществляется, если некоторое логическое выражение или его результат окажется ложным.*

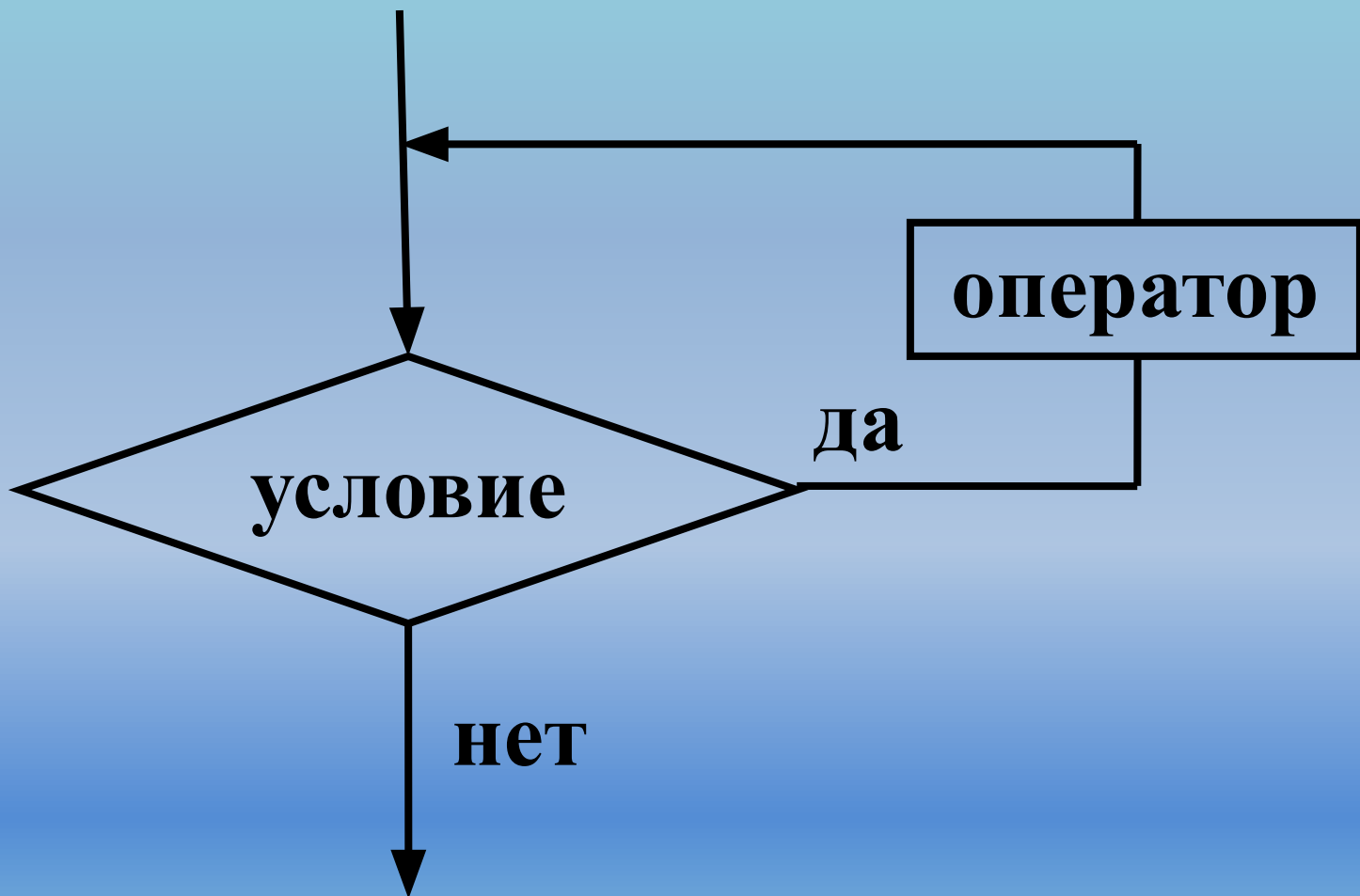
*Так как верность логического выражения проверяется в начале, тело цикла может не выполниться ни одного раза.*

# Структура цикла **WHILE**

```
WHILE <условие> DO  
  begin  
    <тело цикла>;  
  end;
```

# Блок – схема цикла

## WHILE



# Пример

**Задача: Написать программу, которая вычисляет сумму всех четных чисел до 50.**

```
var
  sum: integer;
  n: integer;
begin
  sum:=0;
  n:=2;
  while n <= 50 do
    begin
      sum:= sum + n;
      n:= n + 2;
    end;
  writeln ('Сумма равна: ',sum);
end.
```

# Задача

Написать программу, которая ищет  $n!$ .

# Цикл с постусловием в Паскале – REPEAT-UNTIL

*Этот оператор аналогичен оператору цикла с предусловием, но отличается от него тем, что проверка условия производится после выполнения тела (действий) цикла. Это обеспечивает его выполнение хотя бы один раз в отличие от ранее разобранных циклов.*

*Обратите внимание на то, что данный оператор цикла предполагает наличие нескольких операторов в теле цикла, то есть можно выполнять несколько действий, поэтому служебные слова **Begin** и **End** не нужны.*



# Структура цикла REPEAT-UNTIL

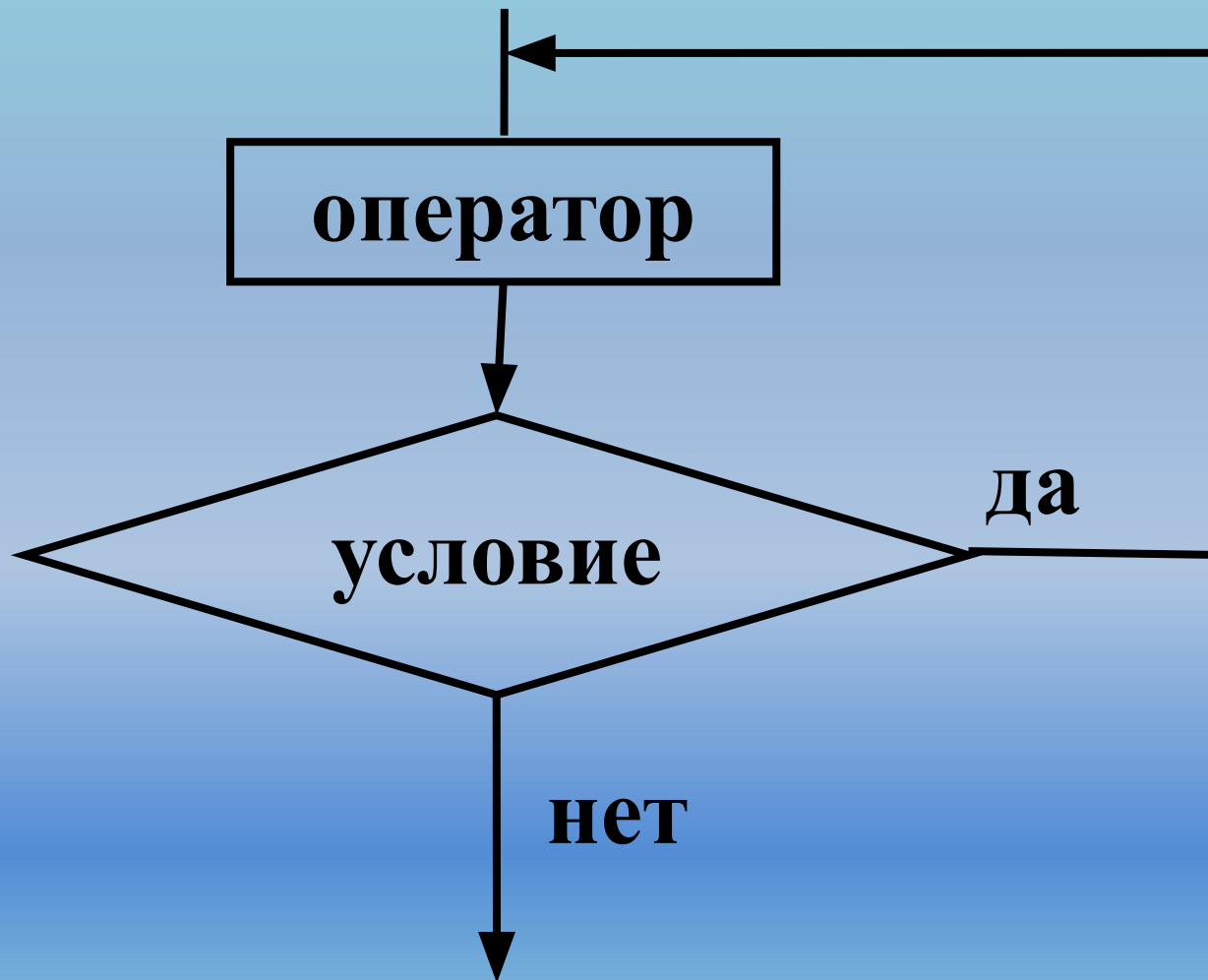
REPEAT

<тело цикла>;

UNTIL

<условие>;

# Блок – схема цикла REPEAT-UNTIL



# Пример

**Задача:** Написать программу, которая определяет сумму первой и последней цифр в числе.

```
var
  a,b,c,d:integer;
begin
  writeln('введите число');
  readln(a);
  d:=a mod 10;
  repeat
    b:=a mod 10;
    a:=a div 10;
  until a=0;
  c:=d+b;
  writeln('Сумма первой и последней цифры равна: 'c);
end.
```

# Задача

Написать программу, которая определяет является ли число простым.

# Цикл с параметром в Паскале - FOR

Цикл **FOR** задаёт условие по которому программа будет работать до его выполнения, допустим нужно *n* раз зациклить программу, то это легко сделать с помощью данного цикла.

У цикла **FOR** есть характерная черта - счетчик который обычно обозначается буквой *i* или *j*.

В цикле счетчик можно задавать как в прямом (служебное слово **to**), так и в обратном порядке (служебное слово **downto**).

# Структура цикла FOR

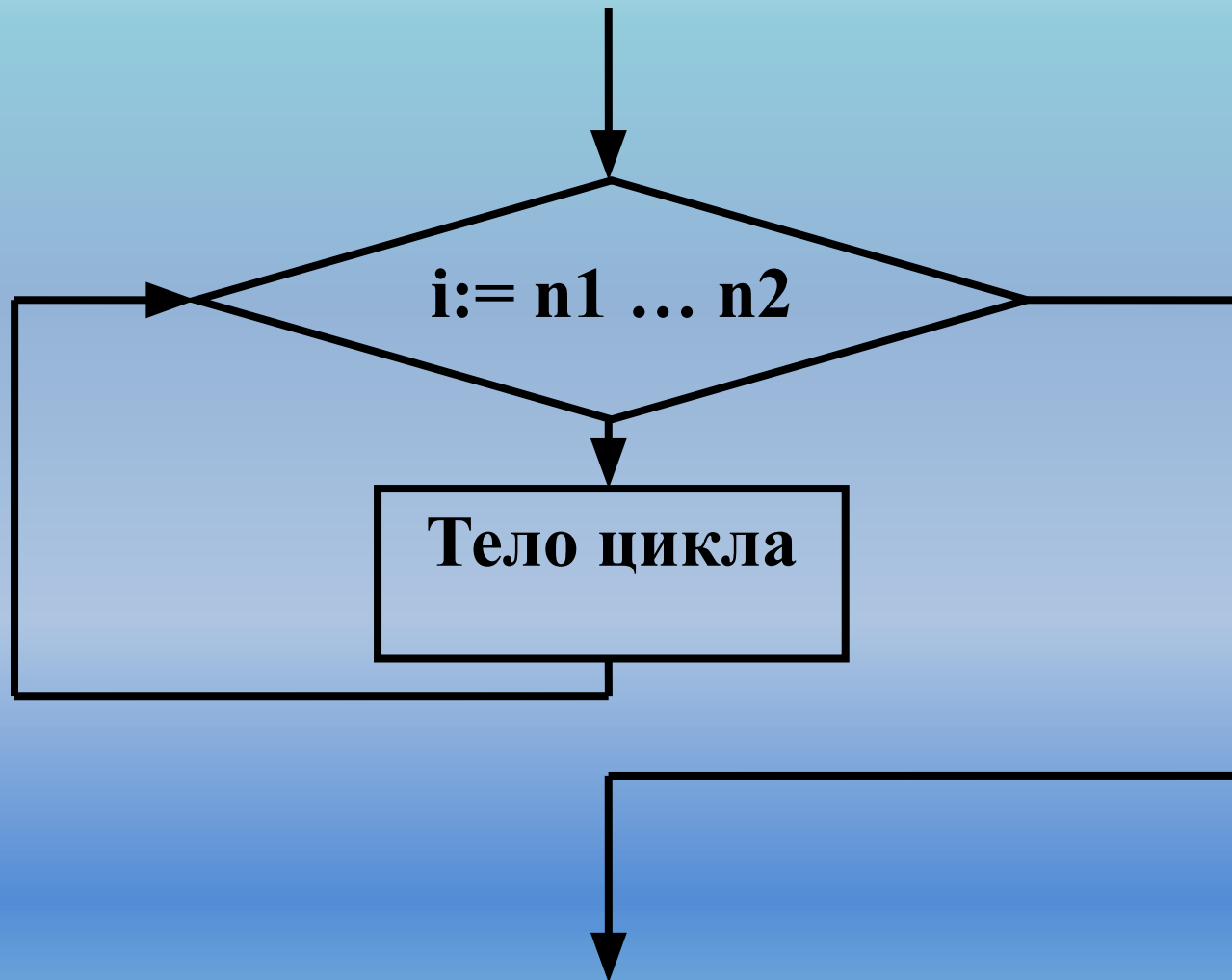
```
FOR i:= n1 TO n2 DO  
  begin  
    <тело цикла>;  
  end;
```

1 – ая форма  
записи

2 – ая форма  
записи

```
FOR i:= n2 DOWNTO n1 DO  
  begin  
    <тело цикла>;  
  end;
```

# Блок – схема цикла **FOR**



# Пример

**Задача: Написать программу, которая вычисляет n-ую степень заданного числа.**

```
var
  a, n, i, pr: integer;
begin
  writeln ('Введите число');
  readln (a);
  writeln ('Введите степень числа');
  readln (n);
  pr:= 1;
  for i:= 1 to n do
    begin
      pr:= pr * a;
    end;
  writeln ('Степень числа равна',pr);
end.
```



# Задача

Написать программу, которая находит число  $P = (1-1/2)(1-1/3) \dots (1-1/n)$ .

$N$  вводится с клавиатуры.

**СПАСИБО ЗА  
ВНИМАНИЕ!**