

# Программирование на языке Паскаль

- § 54. Алгоритм и его свойства
- § 55. Простейшие программы
- § 56. Вычисления
- § 57. Ветвления
- § 58. Циклические алгоритмы
- § 59. Процедуры
- § 60. Функции
- § 61. Рекурсия

# Программирование на языке Паскаль

## **§ 54. Алгоритм и его свойства**

# Что такое алгоритм?

**Алгоритм** — это точное описание порядка действий, которые должен выполнить исполнитель для решения задачи за конечное время.

**Исполнитель** — это устройство или одушевленное существо (человек), способное понять и выполнить команды, составляющие алгоритм.

**Формальные исполнители:** не понимают (и не могут понять) смысл команд.



Мухаммед ал-Хорезми  
(ок. 783—ок. 850 гг.)

# Свойства алгоритма

**Дискретность** — алгоритм состоит из отдельных команд, каждая из которых выполняется за конечное время.

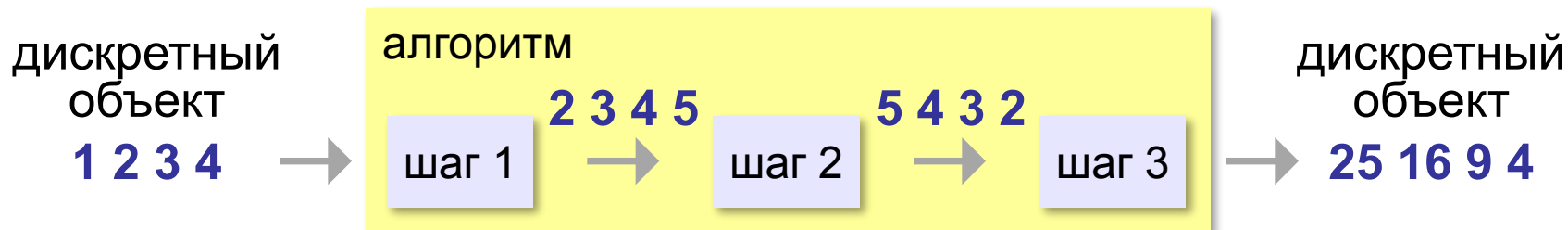
**Детерминированность** (определённость) — при каждом запуске алгоритма с одними и теми же исходными данными получается один и тот же результат.

**Понятность** — алгоритм содержит только команды, входящие в **систему команд исполнителя**.

**Конечность** (результативность) — для корректного набора данных алгоритм должен завершаться через конечное время.

**Корректность** — для допустимых исходных данных алгоритм должен приводить к правильному результату.

# Как работает алгоритм?



- получает на вход дискретный объект
- в результате строит другой дискретный объект (или выдаёт сообщение об ошибке)
- обрабатывает объект по шагам
- на каждом шаге получается новый дискретный объект

# Способы записи алгоритмов

---

- **естественный язык**

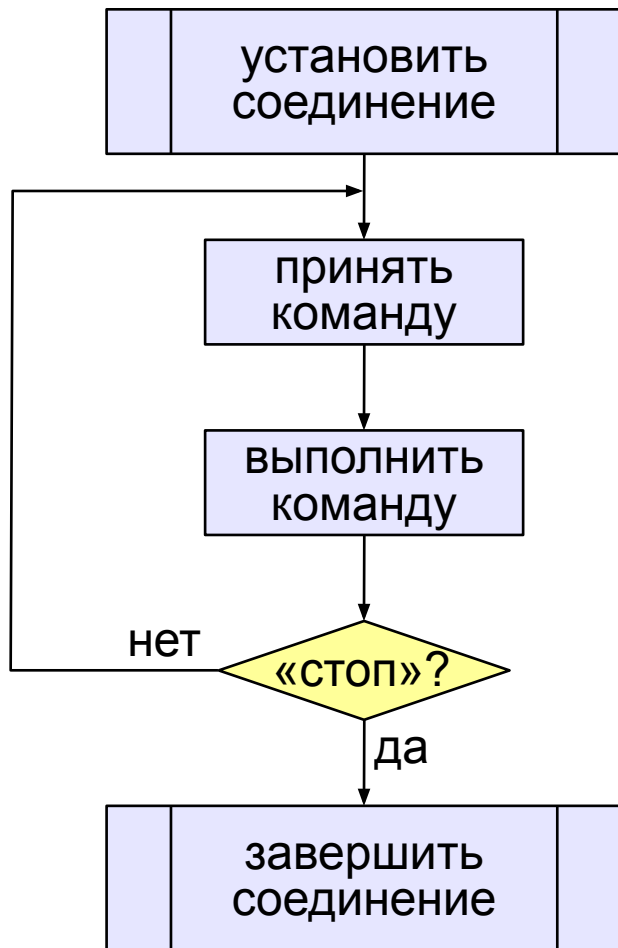
```
установить соединение  
пока не принята команда «стоп»  
    принять команду  
    выполнить команду  
завершить сеанс связи
```

- **псевдокод**

```
установить соединение  
нц  
    принять команду  
    выполнить команду  
кц_при команда = 'stop'  
завершить сеанс связи
```

# Способы записи алгоритмов

- блок-схема



- программа

```
setConnection;  
repeat  
    cmd := getCommand;  
    executeCommand(cmd);  
until cmd = "stop";  
closeConnection;
```

# Программирование на языке Паскаль

## **§ 55. Простейшие программы**



# Простейшая программа

---

название алгоритма

```
program qq;  
begin { начало программы }  
      { тело программы }  
end.  { конец программы }
```

комментарии в скобках {}  
не обрабатываются



Что делает эта программа?

# Вывод на экран

---

```
program qq;  
begin  
▶ write('2+');  
▶ writeln('2=?'); { на новую строку}  
▶ writeln('Ответ: 4');  
end.
```

## Протокол:

2+

Ответ: 4

# Задания

---

«В»: Вывести на экран текст «лесенкой»

Вася

пошел

гулять

«С»: Вывести на экран рисунок из букв

```
  Ж
  ЖЖЖ
 ЖЖЖЖЖ
ЖЖЖЖЖЖЖ
  НН  НН
  ZZZZZ
```

# Сложение чисел

---

Задача. Ввести с клавиатуры два числа и найти их сумму.

Протокол:

Введите два целых числа

25 30

25+30=55

компьютер

пользователь

компьютер считает сам!

?

1. Как ввести числа в память?
2. Где хранить введенные числа?
3. Как вычислить?
4. Как вывести результат?

## Сумма: псевдокод

```
програм qq;  
begin  
  { ввести два числа }  
  { вычислить их сумму }  
  { вывести сумму на экран }  
end.
```

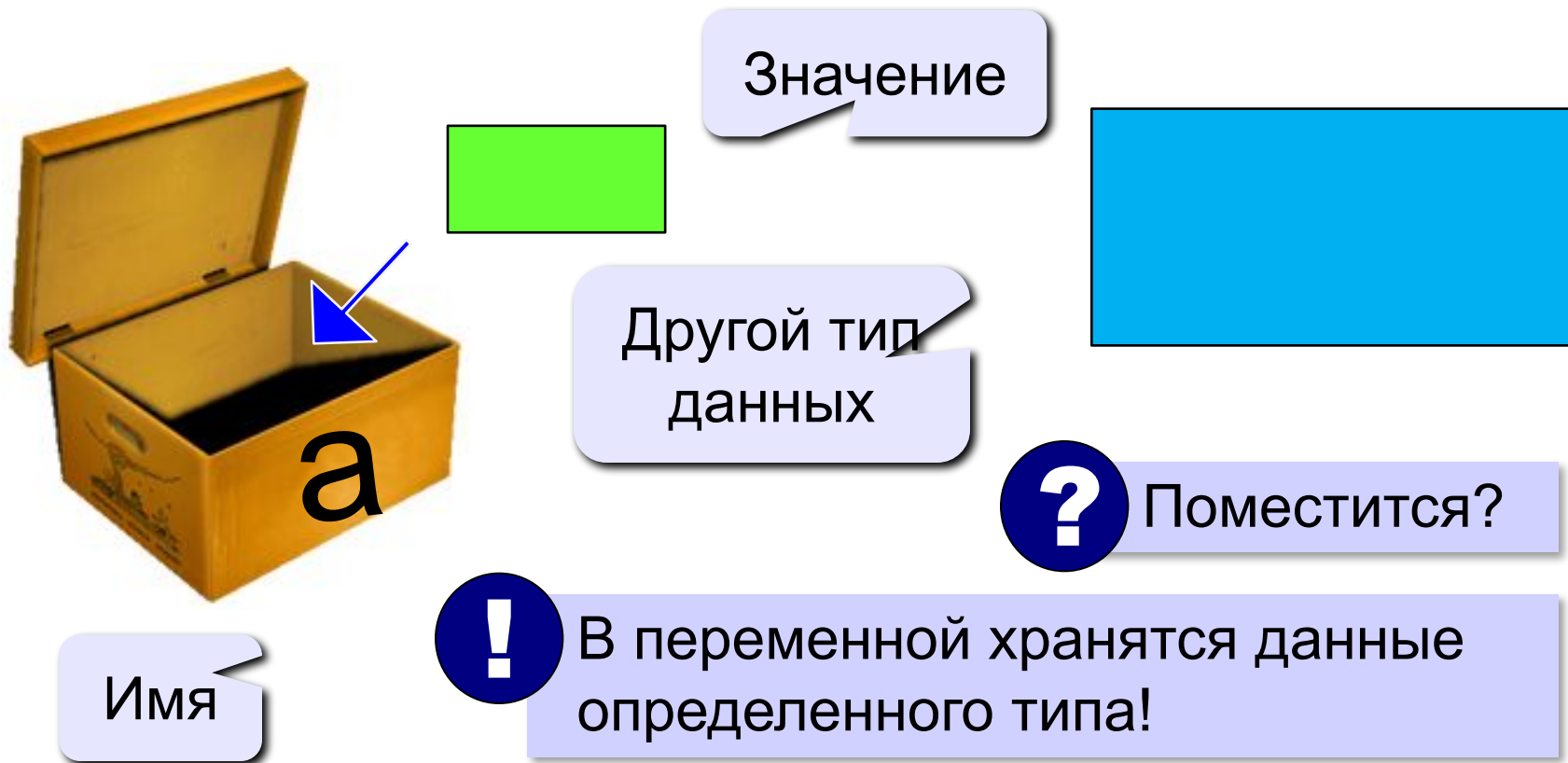
**Псевдокод:** алгоритм на русском языке с элементами Паскаля.



Компьютер не может исполнить псевдокод!

# Переменные

**Переменная** – это величина, имеющая имя, тип и значение. Значение переменной можно изменять во время работы программы.



# Имена переменных

**МОЖНО** использовать

- латинские буквы (A-Z)

заглавные и строчные буквы **НЕ различаются**

- цифры

имя не может начинаться с цифры

- знак подчеркивания \_

**НЕЛЬЗЯ** использовать

- ~~русские буквы~~
- ~~пробелы~~
- ~~скобки, знаки +, =, !, ? и др.~~

Какие имена правильные?

**AXby R&B 4Wheel Вася "PesBarbos"**  
**TU154 [QuQu] \_ABBA A+B**

# Объявление переменных

---

## Типы переменных:

- **integer** { целая }
- **real** { вещественная }
- и другие...

выделение  
места в памяти

## Объявление переменных:

*variable* – переменная

тип – целые

```
var a, b, c: integer;
```

СПИСОК ИМЕН  
переменных



# Тип переменной

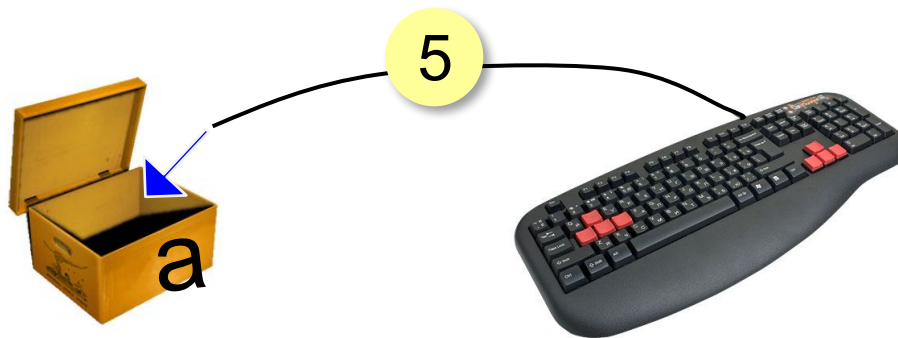
---

- область допустимых значений
- допустимые операции
- объём памяти
- формат хранения данных
- для предотвращения случайных ошибок

# Ввод значения в переменную

оператор  
ввода

```
read ( a );
```



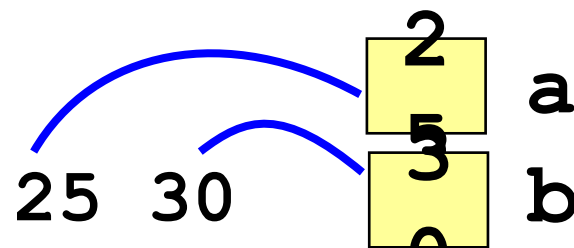
1. Программа ждет, пока пользователь введет значение и нажмет *Enter*.
2. Введенное значение записывается в переменную **a**.

# Ввод значений переменной

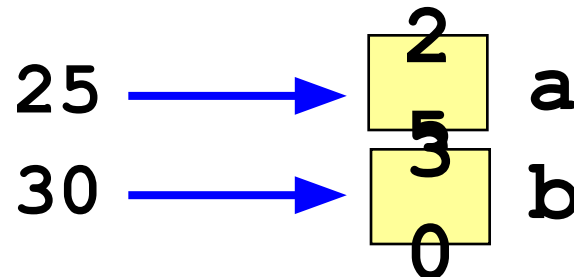
```
read ( a , b ) ;
```

Ввод значений двух переменных (через пробел или *Enter*).

через пробел:



через *Enter*:



# Изменение значений переменной

```
var a, b: integer;
```

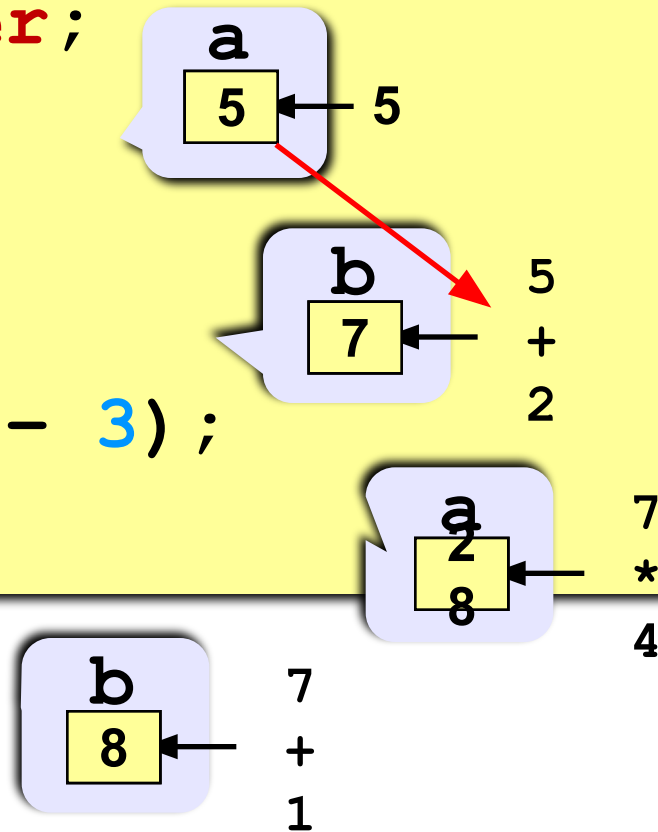
```
...
```

```
a := 5;
```

```
b := a + 2;
```

```
a := (a + 2) * (b - 3);
```

```
b := b + 1;
```



# Вывод данных

---

```
write( a ); { вывод значения  
              переменной a }
```

```
writeln( a ); { вывод значения  
                переменной a и переход  
                на новую строку }
```

```
writeln( 'Привет!' ); { вывод текста }
```

```
writeln( 'Ответ: ', c );
```

{вывод текста и значения переменной c}

```
writeln ( a, '+', b, '=', c );
```

# Сложение чисел: простое решение

---

```
program Sum;  
var a, b, c: integer;  
begin  
    read ( a, b );  
    c := a + b;  
    writeln ( c );  
end.
```



Что плохо?

# Сложение чисел: полное решение

```
program Sum;  
var a, b, c: integer;  
begin  
  writeln('Введите два целых числа');  
  read ( a, b );  
  c := a + b;  
  writeln ( a, '+', b, '=', c );  
end.
```

Протокол:

КОМПЬЮТЕР

Введите два целых числа

25 30

ПОЛЬЗОВАТЕЛЬ

25+30=55

# Снова про оператор вывода

---

Вычисление выражений:

```
writeln ( a, '+', b, '=', a+b );
```

Форматный вывод:

```
a := 123;  
write( a:5 );
```

\_\_123  
5 знаков




# Программирование на языке Паскаль

## **§ 56. Вычисления**

# Типы данных

---

- `byte` { целые 0..255 }
- `shortint` { целые -128..128 }
- `word` { целые 0..65535 }
- `longint` { целые -2147483648..2147483647 }
- `single` {  Сколько байт в памяти? 4 байта }
- `real` { вещественная, 6 байта }
- `double` { вещественная, 8 байтов }
- `extended` { вещественная, 10 байтов }
- `boolean` { логическая, 1 байт }
- `char` { символ, 1 байт }
- `string` { символьная строка }

# Арифметические выражения

3    1 2    4            5    6

`a := (c + b*5*3 - 1) / 2 * d;`

**Приоритет** (*старшинство*):

1) скобки

2) умножение и деление

3) сложение и вычитание


$$a = \frac{c + b \cdot 5 \cdot 3 - 1}{2} \cdot d$$

## Деление, `div`, `mod`

---

Результат деления «/» – вещественное число:

```
var a: single;  
a := 2 / 3; → 0.6666...
```



`div` – деление нацело (остаток отбрасывается)

`mod` – остаток от деления

```
var a, b, d: integer;  
...  
d := 85;  
b := d div 10;  
a := d mod 10;
```

## div и mod для отрицательных чисел

```
write(-7 div 2, ', ');  
write(-7 mod 2);
```

$$\begin{array}{l} -3 \\ -1 \end{array} \quad -7 = (-3) * 2 + (-1)$$



В математике не так!

$$-7 = (-4) * 2 + 1$$

остаток  $\geq 0$

# Вещественные числа



Целая и дробная части числа разделяются точкой!

```
var x: double;
...
x := 123.456;
```

## Форматный вывод:

```
a := 1;
write( a/3 );
write( a/3:7:3 );
```

$$3,333333 \cdot 10^{-1} = 0,33333333$$

```
3.3333333E-001
  0.333
```

всего знаков

в дробной части

# Стандартные функции

---

- abs** (x) — модуль
- sqrt** (x) — квадратный корень
- sin** (x) — синус угла, заданного **в радианах**
- cos** (x) — косинус угла, заданного **в радианах**
- exp** (x) — экспонента  $e^x$
- ln** (x) — натуральный логарифм
- trunc** (x) — отсечение дробной части
- round** (x) — округление до ближайшего целого

# Случайные числа

## Случайно...

- встретить друга на улице
- разбить тарелку
- найти 10 рублей
- выиграть в лотерею

## Случайный выбор:

- жеребьевка на соревнованиях
- выигравшие номера в лотерее

## Как получить случайность?





# Случайные числа на компьютере

## Электронный генератор



- нужно специальное устройство
- нельзя воспроизвести результаты

**Псевдослучайные числа** – обладают свойствами случайных чисел, но каждое следующее число вычисляется по заданной формуле.

## Метод середины квадрата (Дж. фон Нейман)

зерно

564321

в квадрате

- малый период  
(последовательность повторяется через  $10^6$  чисел)

318458191041

209938992481

# Линейный конгруэнтный генератор

$X := (a \cdot X + b) \bmod c$  | интервал от 0 до  $c-1$

$X := (X + 3) \bmod 10$  | интервал от 0 до 9

$X := 0 \rightarrow 3 \rightarrow 6 \rightarrow 9 \rightarrow 2 \rightarrow 5 \rightarrow 8$

$8 \rightarrow 1 \rightarrow 4 \rightarrow 7 \rightarrow 0$

зерно

заикливание



Важен правильный выбор параметров  $a$ ,  $b$  и  $c$ !

**Компилятор GCC:**

$a = 1103515245$

$b = 12345$

$c = 2^{31}$

# Генератор случайных чисел

---

## Вещественные числа в интервале [0,1):

```
var X, Y: double;  
...  
X:= random; { интервал от 0 до 1 (<1) }  
Y:= random; { это уже другое число! }
```

англ. *random* – случайный

## Целые числа в интервале [0,10):

```
var K, L: integer;  
...  
K:= random( 10 ) { интервал от 0 до 9 (<10) }  
L:= random( 10 ) { это уже другое число! }
```

# Другой интервал

---

## Вещественные числа:

```
var X, a, b: double;
...
X := random*10; { расширение интервала: [0,10) }
X := random*10 + 5;
                { расширение и сдвиг: [5,15) }
X := random*(b-a) + a;
                { расширение и сдвиг: [a,b) }
```

## Целые числа:

```
var K, a, b: integer;
...
K := random(10) + 5;    { [5,14] }
X := random(b-a+1) + a; { [a,b] }
```

# Задачи

---

**«А»:** Ввести с клавиатуры три целых числа, найти их сумму, произведение и среднее арифметическое.

**Пример:**

Введите три целых числа:

5 7 8

$$5+7+8=20$$

$$5*7*8=280$$

$$(5+7+8) / 3 = 6.667$$

**«В»:** Ввести с клавиатуры координаты двух точек (А и В) на плоскости (вещественные числа). Вычислить длину отрезка АВ.

**Пример:**

Введите координаты точки А:

5.5 3.5

Введите координаты точки В:

1.5 2

Длина отрезка АВ = 4.272

# Задачи

---

**«С»:** Получить случайное трехзначное число и вывести через запятую его отдельные цифры.

**Пример:**

Получено число 123.

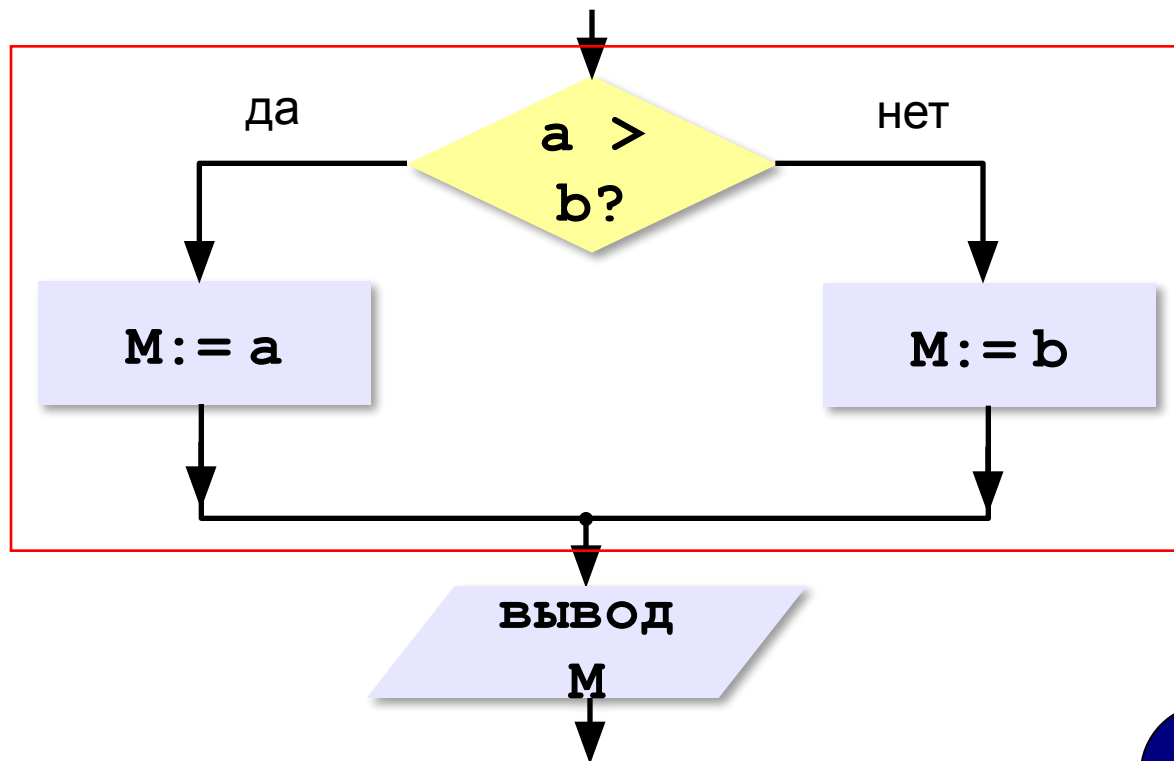
Его цифры 1, 2, 3.

# Программирование на языке Паскаль

## § 57. Ветвления

# Условный оператор

Задача: **изменить порядок действий** в зависимости от выполнения некоторого условия.



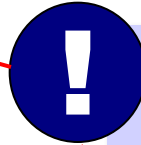
полная  
форма  
ветвления

? Если  $a = b$ ?



# Условный оператор: полная форма

```
if a > b then  
    M := a  
else  
    M := b;
```

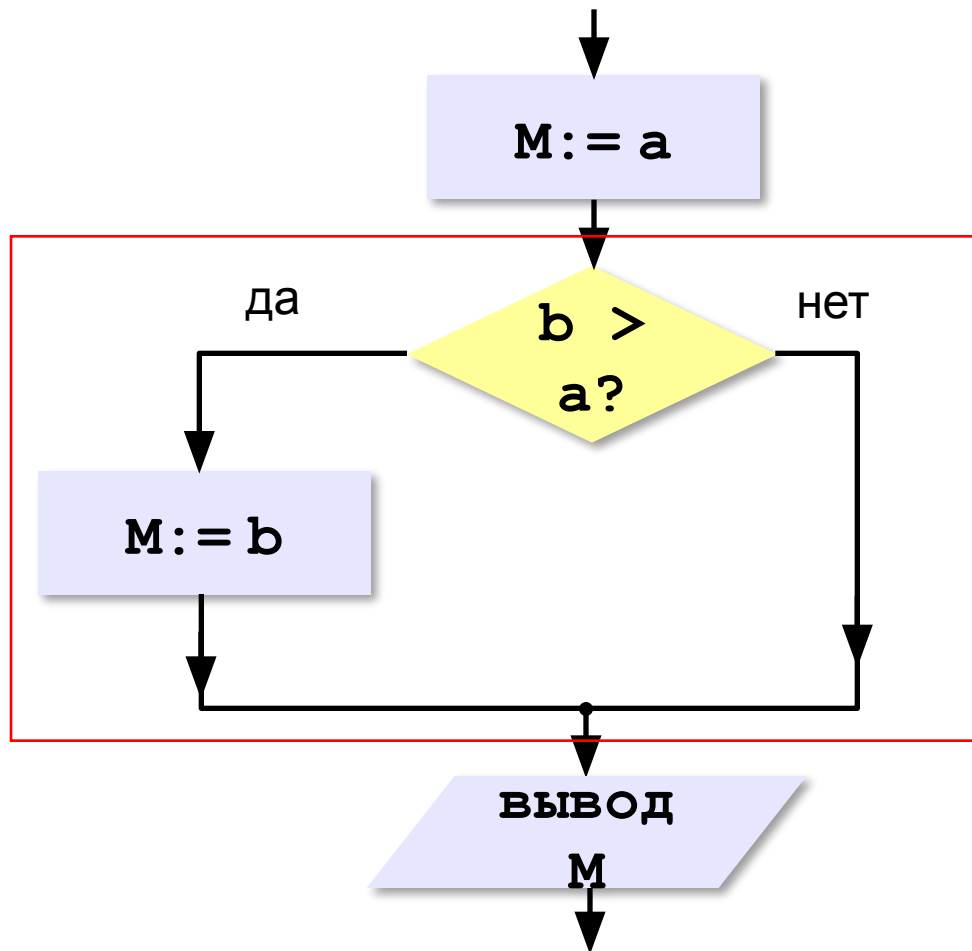


Перед `else` знак « ; »  
**НЕ** ставится!

```
if a > b then begin  
    M := a;  
end  
else begin  
    M := b;  
end;
```

операторные  
скобки

# Условный оператор: неполная форма



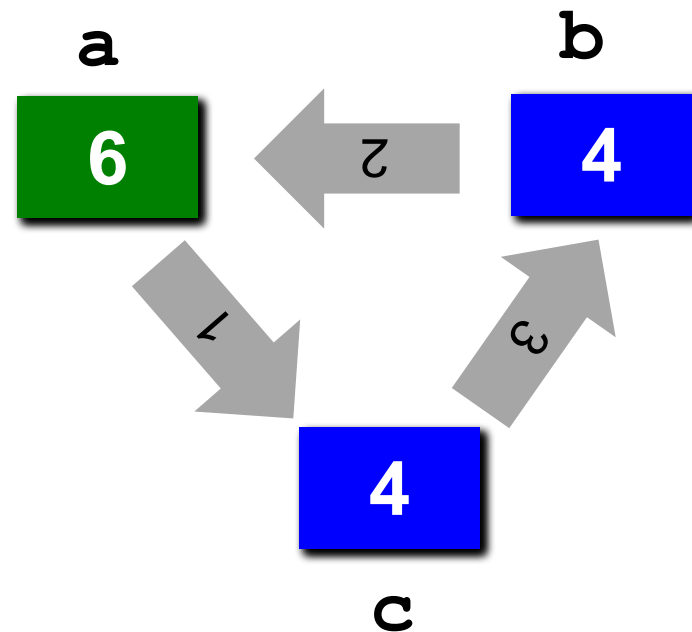
```
M := a;  
if b > a then  
    M := b;
```

неполная  
форма  
ветвления

# Условный оператор

```
if a < b then begin  
  c := a;  
  a := b;  
  b := c  
end;
```

? Что делает?



? Можно ли обойтись без переменной **c**?

# Знаки отношений

---

$>$   $<$  больше, меньше

$\geq$  больше или равно

$\leq$  меньше или равно

$=$  равно

$\neq$  не равно

# Вложенный условный оператор

Задача: в переменных **a** и **b** записаны возрасты Андрея и Бориса. Кто из них старше?

?

Сколько вариантов?

```
if a > b then
    writeln('Андрей старше')
else
    if a = b then
        writeln('Одного возраста')
    else
        writeln('Борис старше');
```

?

Зачем нужен?

вложенный  
условный оператор

# Выделение структуры отступами

---

```
if a > b then write('A') else if a = b then  
write('=') else write('Б');
```

```
if a > b then  
    write('A')  
else  
    if a = b then  
        write('=')  
    else write('Б');
```

# Задачи

---

**«А»:** Ввести три целых числа, найти максимальное из них.

**Пример:**

Введите три целых числа:

1 5 4

Максимальное число 5

**«В»:** Ввести пять целых чисел, найти максимальное из них.

**Пример:**

Введите пять целых чисел:

1 5 4 3 2

Максимальное число 5

# Задачи

---

**«С»:** Ввести последовательно возраст Антона, Бориса и Виктора. Определить, кто из них старше.

**Пример:**

Возраст Антона: 15

Возраст Бориса: 17

Возраст Виктора: 16

Ответ: Борис старше всех.

**Пример:**

Возраст Антона: 17

Возраст Бориса: 17

Возраст Виктора: 16

Ответ: Антон и Борис старше Виктора.



# Сложные условия

Задача: набор сотрудников в возрасте **25-40 лет**  
(включительно).

сложное условие

```
if (v >= 25) and (v <= 40) then
    writeln(' подходит ')
else
    writeln(' не подходит ');
```

and

or

xor

not

исключающее  
«ИЛИ»

Приоритет :

- 1) **not**
- 2) **and**
- 3) **or, xor**
- 4) отношения (<, >, <=, >=, =, <>)



Почему скобки обязательны?

# Задачи

---

**«А»:** Напишите программу, которая получает три числа и выводит количество одинаковых чисел в этой цепочке.

**Пример:**

Введите три числа:

5 5 5

Все числа одинаковые.

**Пример:**

Введите три числа:

5 7 5

Два числа одинаковые.

**Пример:**

Введите три числа:

5 7 8

Нет одинаковых чисел.

# Задачи

---

**«В»:** Напишите программу, которая получает номер месяца и выводит соответствующее ему время года или сообщение об ошибке.

**Пример:**

**Введите номер месяца :**

**5**

**Весна .**

**Пример:**

**Введите номер месяца :**

**15**

**Неверный номер месяца .**

# Задачи

---

**«С»:** Напишите программу, которая получает возраст человека (целое число, не превышающее 120) и выводит этот возраст со словом «год», «года» или «лет». Например, «21 год», «22 года», «25 лет».

**Пример:**

Введите возраст: **18**

Вам 18 лет.

**Пример:**

Введите возраст: **21**

Вам 21 год.

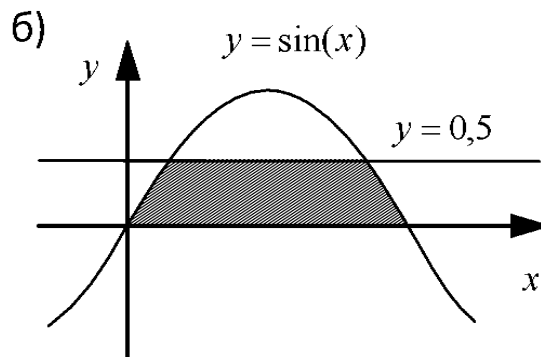
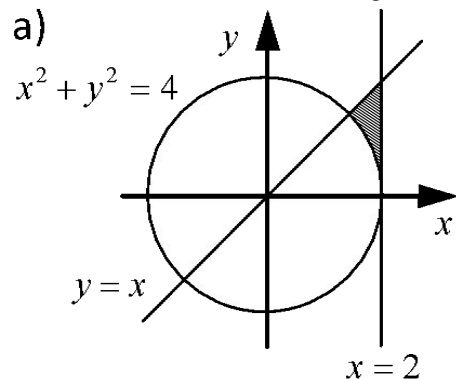
**Пример:**

Введите возраст: **22**

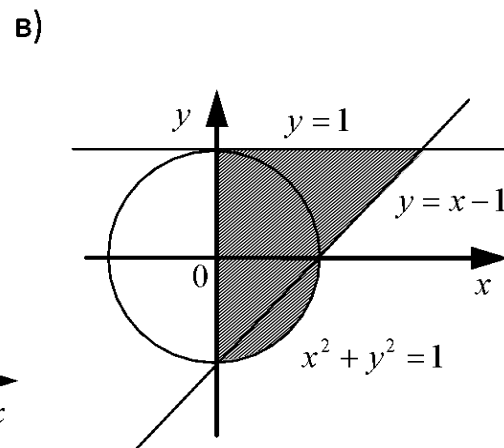
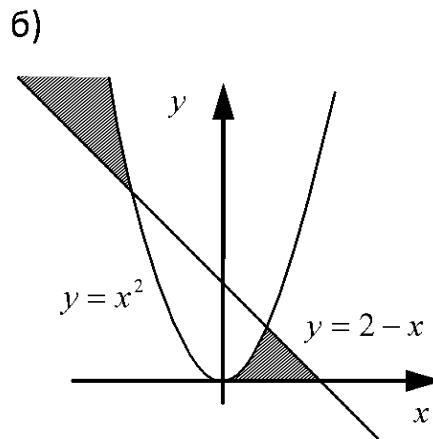
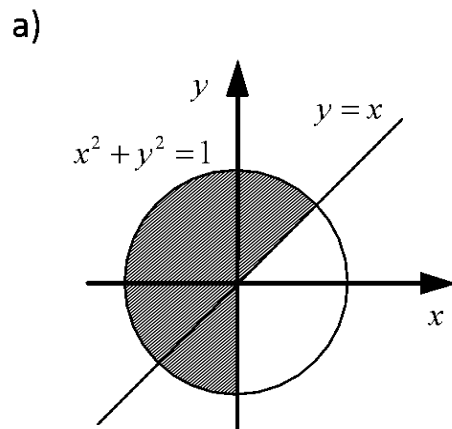
Вам 22 года.

# Задачи

«А»: Напишите условие, которое определяет заштрихованную область.

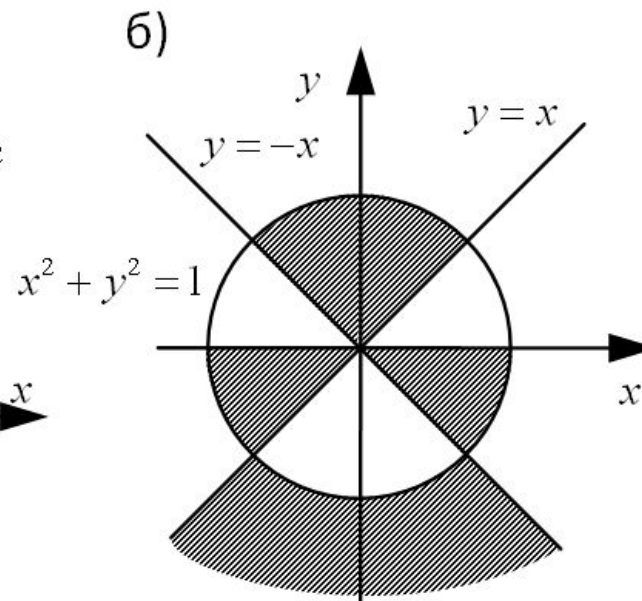
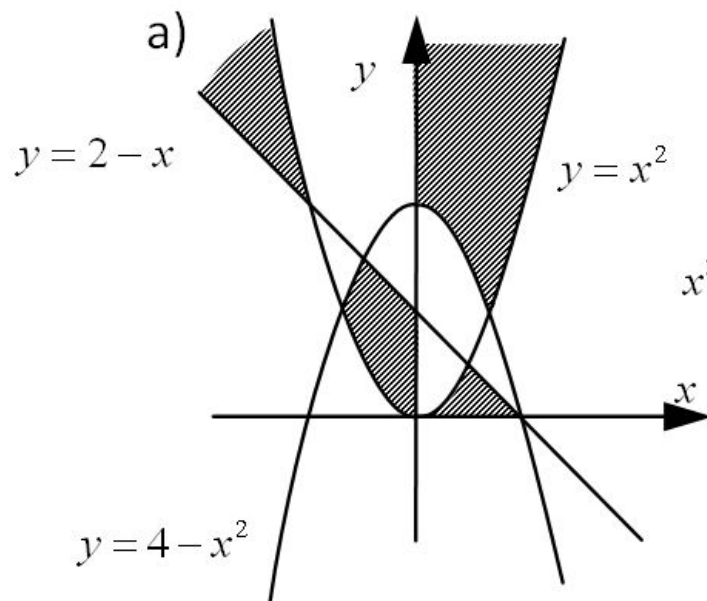


«В»: Напишите условие, которое определяет заштрихованную область.



# Задачи

«С»: Напишите условие, которое определяет заштрихованную область.



# Множественный выбор

---

```
if m = 1 then
    write('январь');
if m = 2 then
    write('февраль');
...
if m = 12 then
    write('декабрь');
```

```
case m of
    1: write('январь');
    2: write('февраль');
    ...
    12: write('декабрь')
else write('ошибка')
end;
```

# Использование списков и диапазонов

---

Число дней в месяце:

```
case m of
  2: d:= 28;   { невисокосный год }
  1,3,5,7,8,10,12: d:= 31
  else d:= 30
end;
```

Социальный статус:

```
case v of
  0..6:  write(' дошкольник ');
  7..17: write(' школьник ');
  else   write(' взрослый ')
end;
```



# Множественный выбор

---

```
var c: char;  
...  
case c of  
  'а': begin  
        writeln('антилопа');  
        writeln('Анапа');  
      end;  
...  
  'я': begin  
        writeln('ягуар');  
        writeln('Якутск');  
      end  
else writeln('ошибка')  
end;
```

несколько  
операторов в  
блоке

# Программирование на языке Паскаль

## **§ 58. Циклические алгоритмы**

# Что такое цикл?

**Цикл** – это многократное выполнение одинаковых действий.

## Два вида циклов:

- цикл с **известным** числом шагов (сделать 10 раз)
- цикл с **неизвестным** числом шагов (делать, пока не надоест)

*Задача.* Вывести на экран 10 раз слово «Привет».



Можно ли решить известными методами?

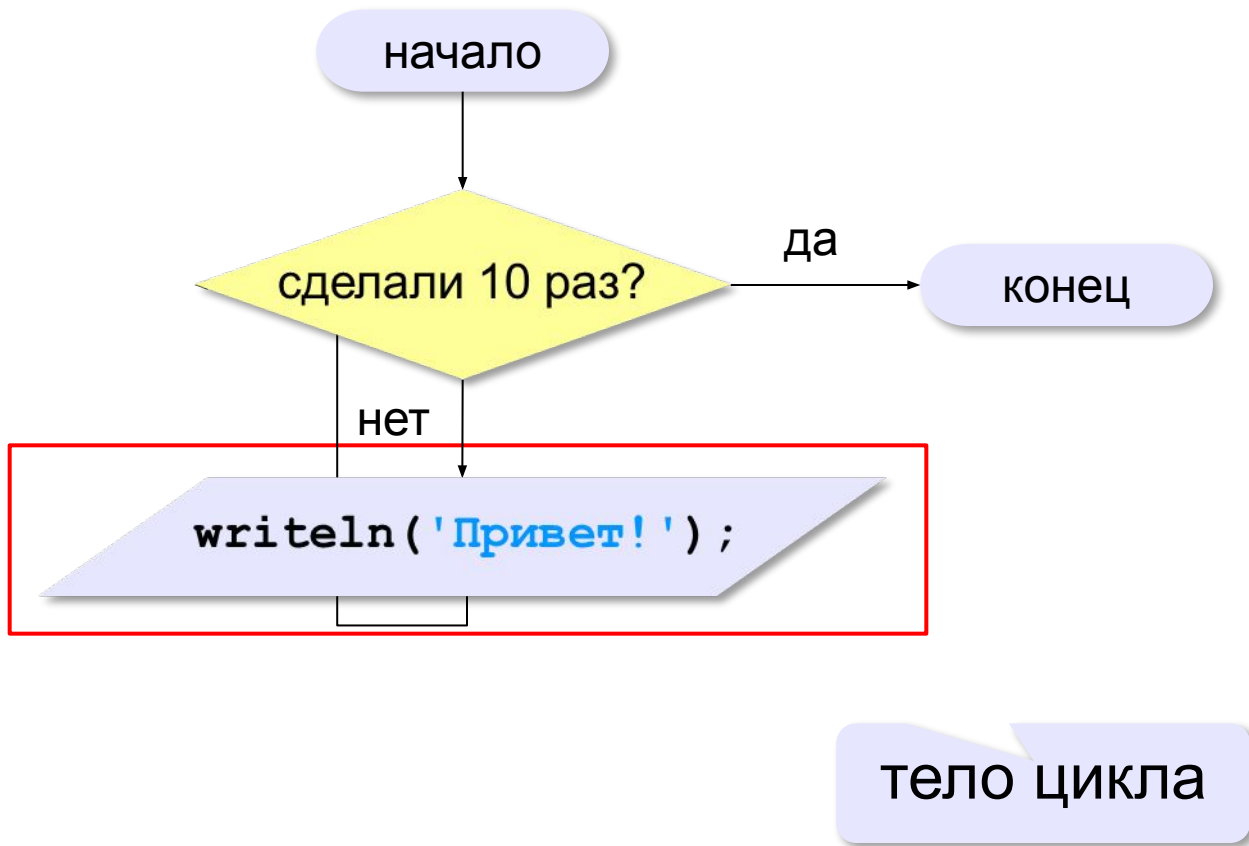
# Повторения в программе

---

```
writeln ( ' Привет ' ) ;  
writeln ( ' Привет ' ) ;  
writeln ( ' Привет ' ) ;  
...  
writeln ( ' Привет ' ) ;
```



# Блок-схема цикла



# Как организовать цикл?

```
счётчик := 0
пока счётчик < 10
  writeln('привет');
  увеличить счётчик на 1
```

результат операции  
автоматически  
сравнивается с нулём!

```
счётчик := 10
пока счётчик > 0
  writeln('привет');
  уменьшить счётчик на 1
```



Какой способ удобнее для процессора?

# Цикл с условием

*Задача.* Определить **количество цифр** в десятичной записи целого положительного числа, записанного в переменную  $n$ .

```
счётчик := 0
пока n > 0
    отсечь последнюю цифру n
    увеличить счётчик на 1
```

$n$	счётчик
1234	0

?

Как отсечь последнюю цифру?

```
n := n div 10
```

?

Как увеличить счётчик на 1?

```
счётчик := счётчик + 1
```

# Цикл с условием

начальное значение  
счётчика

условие  
продолжения

заголовок  
цикла

```
count := 0;  
while n > 0 do begin  
    n := n div 10;  
    count := count + 1  
end;
```

тело цикла



Зачем `begin-end`?



Цикл с предусловием – проверка на входе в цикл!



# Цикл с условием

---

При известном количестве шагов:

```
k := 0;  
while k < 10 do begin  
  writeln('привет');  
  k := k + 1  
end;
```

Защивание:

```
k := 0;  
while k < 10 do  
  writeln('привет');
```

# Сколько раз выполняется цикл?

```
a := 4; b := 6;  
while a < b do a := a + 1;
```

2 раза

a = 6

```
a := 4; b := 6;  
while a < b do a := a + b;
```

1 раз

a = 10

```
a := 4; b := 6;  
while a > b do a := a + 1;
```

0 раз

a = 4

```
a := 4; b := 6;  
while a < b do b := a - b;
```

1 раз

b = -2

```
a := 4; b := 6;  
while a < b do a := a - 1;
```

**зацикливание**

# Цикл с постусловием

заголовок  
цикла

```
repeat
```

```
  write ( 'Введите n > 0: ' );  
  read (n)
```

```
until n > 0 ;
```

тело цикла

условие  
окончания

- при входе в цикл условие **не проверяется**
- цикл всегда выполняется **хотя бы один раз**
- в последней строке указывают **условие окончания** цикла, а не условие его продолжения

# Задачи

---

«А»: Напишите программу, которая получает два целых числа  $A$  и  $B$  ( $0 < A < B$ ) и выводит квадраты всех натуральных чисел в интервале от  $A$  до  $B$ .

**Пример:**

Введите два целых числа :

10 12

$10 * 10 = 100$

$11 * 11 = 121$

$12 * 12 = 144$

«В»: Напишите программу, которая получает два целых числа и находит их произведение, не используя операцию умножения. Учтите, что числа могут быть отрицательными.

**Пример:**

Введите два числа :

10 -15

$10 * (-15) = -150$

# Задачи

---

**«С»:** Ввести натуральное число  $N$  и вычислить сумму всех чисел Фибоначчи, меньших  $N$ . Предусмотрите защиту от ввода отрицательного числа  $N$ .

**Пример:**

Введите число  $N$ :

**10000**

Сумма 17710

## Задачи-2

---

**«А»:** Ввести натуральное число и найти сумму его цифр.

**Пример:**

Введите натуральное число:

**12345**

Сумма цифр 15.

**«В»:** Ввести натуральное число и определить, верно ли, что в его записи есть две одинаковые цифры, стоящие рядом.

**Пример:**

Введите натуральное число:

**12342**

Нет.

**Пример:**

Введите натуральное число:

**12245**

Да.

## Задачи-2

---

**«С»:** Ввести натуральное число и определить, верно ли, что в его записи есть две одинаковые цифры (не обязательно стоящие рядом).

**Пример:**

Введите натуральное число:

**12342**

Да .

**Пример:**

Введите натуральное число:

**12345**

Нет .

# Цикл с переменной

Задача. Вывести все степени двойки от  $2^1$  до  $2^{10}$ .



Можно ли сделать с циклом «while»?

```
k := 1;  
n := 2;  
while k <= 10 do  
begin  
  writeln(n);  
  n := n * 2;  
  k := k + 1  
end;
```

```
n := 2;  
for k := 1 to 10 do  
begin  
  writeln(n);  
  n := n * 2  
end;
```



Переменная  $k$  – целая!



# Цикл с переменной: другой шаг

```
var k: integer;
```

целое

целое

```
for k:= 10 downto 1 do  
  writeln(k*k);
```

шаг «-1»



Шаг может быть равен только 1 или «-1» !



Как сделать шаг 2?

```
k:= 1;
```

```
for i:= 1 to 10 do begin  
  writeln(k*k);
```

```
  k:= k + 2
```

```
end;
```

# Сколько раз выполняется цикл?

---

```
a := 1;  
for i := 1 to 3 do a := a + 1;
```

a = 4

```
a := 1;  
for i := 3 to 1 do a := a + 1;
```

a = 1

```
a := 1;  
for i := 1 downto 3 do a := a + 1;
```

a = 1

```
a := 1;  
for i := 3 downto 1 do a := a + 1;
```

a = 4

# Задачи

---

«А»: Найдите все пятизначные числа, которые при делении на 133 дают в остатке 125, а при делении на 134 дают в остатке 111.

«В»: Натуральное число называется **числом Армстронга**, если сумма цифр числа, возведенных в N-ную степень (где N – количество цифр в числе) равна самому числу. Например,  $153 = 1^3 + 5^3 + 3^3$ .  
Найдите все трёхзначные Армстронга.

# Задачи

---

«С»: Натуральное число называется **автоморфным**, если оно равно последним цифрам своего квадрата. Например,  $25^2 = 625$ . Напишите программу, которая получает натуральное число N и выводит на экран все автоморфные числа, не превосходящие N.

**Пример:**

Введите N:

1000

$$1 * 1 = 1$$

$$5 * 5 = 25$$

$$6 * 6 = 36$$

$$25 * 25 = 625$$

$$76 * 76 = 5776$$

# Вложенные циклы

---

Задача. Вывести все простые числа в диапазоне от 2 до 1000.

```
для n от 2 до 1000
  если число n простое то
    writeln(n) ;
```

нет делителей [2.. n-1]:  
проверка в цикле!



Что значит «простое число»?

# Вложенные циклы

---

```
for n:= 2 to 1000 do begin
  count:= 0;
  for k:= 2 to n-1 do
    if n mod k = 0 then
      count:= count + 1;
  if count = 0 then
    writeln(n)
end;
```

ВЛОЖЕННЫЙ ЦИКЛ

# Вложенные циклы

---

```
for i:=1 to 4 do
  for k:=1 to i do
    writeln(i, ' ', k);
```

```
1 1
2 1
2 2
3 1
3 2
3 3
4 1
4 2
4 3
4 4
```



Как меняются переменные?



Переменная внутреннего цикла изменяется быстрее!


# Поиск простых чисел: как улучшить?

$$n = k \cdot m, \quad k \leq m \Rightarrow k^2 \leq n \Rightarrow k \leq \sqrt{n}$$

```
while k <= sqrt(n) do begin
  ...
end;
```

 Что плохо?

```
count := 0;
k := 2;
while k*k <= n do begin
  if n mod k = 0 then
    count := count + 1;
  k := k + 1;
end;
```

 Как ещё улучшить?

```
while (k*k <= n) and (count = 0)
do begin
  ...
end;
```



# Задачи

---

**«А»:** Напишите программу, которая получает натуральные числа  $A$  и  $B$  ( $A < B$ ) и выводит все простые числа в интервале от  $A$  до  $B$ .

**Пример:**

**Введите границы диапазона:**

**10 20**

**11 13 17 19**

**«В»:** В магазине продается мастика в ящиках по 15 кг, 17 кг, 21 кг. Как купить ровно 185 кг мастики, не вскрывая ящики? Сколькими способами можно это сделать?

# Задачи

---

«С»: Ввести натуральное число  $N$  и вывести все натуральные числа, не превосходящие  $N$  и делящиеся на каждую из своих цифр.

**Пример:**

Введите  $N$ :

15

1 2 3 4 5 6 7 8 9 11 12 15

# Программирование на языке Паскаль

## **§ 59. Процедуры**

# Зачем нужны процедуры?

```
writeln('Ошибка программы');
```

много раз!

```
program withProc;
```

```
var n: integer;
```

```
procedure Error;
```

```
begin
```

```
    writeln('Ошибка программы')
```

```
end;
```

```
begin
```

```
    read(n);
```

```
    if n < 0 then Error;
```

```
    ...
```

```
end.
```

ВЫЗОВ  
процедуры

# Что такое процедура?

**Процедура** – вспомогательный алгоритм, который выполняет некоторые действия.

- текст (расшифровка) процедуры записывается **до основной программы**
- в программе может быть **много процедур**
- чтобы процедура заработала, нужно **вызвать** её по имени из основной программы или из другой процедуры

# Процедура с параметрами

*Задача.* Вывести на экран запись целого числа (0..255) в 8-битном двоичном коде.

много раз!

**Алгоритм:**

$$178 \Rightarrow 10110010_2$$

**?** Как вывести первую цифру?

$$n := \begin{matrix} 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \text{разряды} \\ \text{1} & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{matrix} _2$$

$n \text{ div } 128$

$n \text{ mod } 128$

**?** Как вывести вторую цифру?

$n1 \text{ div } 64$

# Процедура с параметрами

*Задача.* Вывести на экран запись целого числа (0..255) в 8-битном двоичном коде.

**Алгоритм:**

```
k := 128;
while k > 0 do begin
  write(n div k);
  n := n mod k;
  k := k div 2;
end;
```

178 ⇒ 10110010



Результат зависит от n!

n	k	ВЫВОД
178	128	1

# Процедура с параметрами

```
program binCode;
```

```
  procedure printBin (n: integer);
```

```
  var k: integer;
```

```
  begin
```

```
    k := 128;
```

```
    while k > 0 do begin
```

```
      write (n div k);
```

```
      n := n mod k;
```

```
      k := k div 2
```

```
    end
```

```
  end;
```

```
begin
```

```
  printBin (99)
```

```
end.
```

локальная  
переменная

**Параметры** – данные,  
изменяющие работу  
процедуры.

значение параметра  
(аргумент)



# Несколько параметров

---

```
procedure printSred(a: integer;  
                   b: integer);  
begin  
    write((a+b)/2);  
end.
```

```
procedure printSred(a, b: integer);  
begin  
    write((a+b)/2);  
end.
```

# Задачи

---

«А»: Напишите процедуру, которая принимает параметр – натуральное число  $N$  – и выводит на экран линию из  $N$  символов '–'.

**Пример:**

Введите  $N$ :

10

-----

«В»: Напишите процедуру, которая выводит на экран в столбик все цифры переданного ей числа, начиная с первой.

**Пример:**

Введите натуральное число:

1234

1

2

3

4

# Задачи

---

«С»: Напишите процедуру, которая выводит на экран запись переданного ей числа в римской системе счисления.

**Пример:**

**Введите натуральное число:**

**2013**

**MMXIII**

# Изменяемые параметры

*Задача.* Написать процедуру, которая меняет местами значения двух переменных.

```
program Exchange;
var x, y: integer;
```

передача по  
значению

```
procedure Swap(a, b: integer);
var c: integer;
begin
  c := a; a := b; b := c;
end;
```

```
begin
  x := 2; y := 3;
  Swap(x, y);
  write(x, ' ', y)
end.
```



Процедура работает с копиями переданных значений параметров!

2 3



Почему не работает?

# Изменяемые параметры

переменные могут  
ИЗМЕНЯТЬСЯ

```
procedure Swap ( var a, b: integer );  
var c: integer;  
begin  
    c := a; a := b; b := c;  
end;
```

передача по  
ССЫЛКЕ

## ВЫЗОВ:

```
var a, b: integer;  
...  
Swap (a, b); { правильно }  
Swap (2, 3); { неправильно }  
Swap (a, b+3); { неправильно }
```

# Задачи

---

**«А»:** Напишите процедуру, которая переставляет три переданные ей числа в порядке возрастания.

**Пример:**

Введите три натуральных числа:

10 15 5

5 10 15

**«В»:** Напишите процедуру, которая сокращает дробь вида  $M/N$ . Числитель и знаменатель дроби передаются как изменяемые параметры.

**Пример:**

Введите числитель и знаменатель дроби:

25 15

После сокращения:  $5/3$

# Задачи

---

«С»: Напишите процедуру, которая вычисляет наибольший общий делитель и наименьшее общее кратное двух натуральных чисел и возвращает их через изменяемые параметры.

**Пример:**

Введите два натуральных числа:

**10 15**

НОД (10, 15) = 5

НОК (10, 15) = 30

# Программирование на языке Паскаль

## **§ 60. Функции**



# Что такое функция?

**Функция** – это вспомогательный алгоритм, который возвращает *значение-результат* (число, символ или объект другого типа).

**Задача.** Написать функцию, которая вычисляет сумму цифр числа.

**Алгоритм:**

```
сумма := 0;  
while n <> 0 do begin  
    сумма := сумма + n mod 10;  
    n := n div 10  
end;
```

# Сумма цифр числа

```
program Sum;
```

```
function sumDigits (n: integer) : integer ;
```

```
var sum: integer;
```

```
begin
```

```
    sum := 0;
```

```
    while n <> 0 do begin
```

```
        sum := sum + n mod 10;
```

```
        n := n div 10;
```

```
    end;
```

```
    sumDigits := sum
```

```
end;
```

тип результата

передача  
результата

```
begin
```

```
    writeln (sumDigits (12345) )
```

```
end.
```

# Использование функций

---

```
x := 2 * sumDigits (n+5) ;  
z := sumDigits (k) + sumDigits (m) ;  
if sumDigits (n) mod 2 = 0 then begin  
    writeln ('Сумма цифр чётная') ;  
    writeln ('Она равна ', sumDigits (n))  
end ;
```



Функция, возвращающая целое число, может использоваться везде, где и целая величина!

# Задачи

---

**«А»:** Напишите функцию, которая находит наибольший общий делитель двух натуральных чисел.

**Пример:**

Введите два натуральных числа:

**7006652 112307574**

$\text{НОД}(7006652, 112307574) = 1234.$

**«В»:** Напишите функцию, которая определяет сумму цифр переданного ей числа.

**Пример:**

Введите натуральное число:

**123**

Сумма цифр числа 123 равна 6.

# Задачи

---

**«С»:** Напишите функцию, которая «переворачивает» число, то есть возвращает число, в котором цифры стоят в обратном порядке.

**Пример:**

**Введите натуральное число:**

**1234**

**После переворота: 4321.**

# Логические функции

---

*Задача.* Найти все простые числа в диапазоне от 2 до 100.

```
program PrimeNum;  
var i: integer;  
begin  
  for i:=2 to 100 do  
    if isPrime(i) then  
      writeln(i)  
end.
```

функция,  
возвращающая  
логическое значение  
(True/False)

# Функция: простое число или нет?



Какой алгоритм?

логическое значение  
(True/False)

```
function isPrime (n: integer) : boolean ;
var count, k: integer;
begin
  count := 0;
  k := 2;
  while (k*k <= n) and (count = 0) do begin
    if n mod k = 0 then
      count := count + 1;
    k := k + 1;
  end;
  isPrime := (count = 0)
end;
```

```
if count = 0 then
  isPrime := True
else isPrime := False
```

# Логические функции: использование

---



Функция, возвращающая логическое значение, может использоваться везде, где и логическая величина!

```
read(n) ;  
while isPrime(n) do begin  
    writeln('простое число') ;  
    read(n)  
end ;
```



# Задачи

---

**«А»:** Напишите логическую функцию, которая определяет, является ли переданное ей число совершенным, то есть, равно ли оно сумме своих делителей, меньших его самого.

**Пример:**

Введите натуральное число:

**28**

Число 28 совершенное.

**Пример:**

Введите натуральное число:

**29**

Число 29 не совершенное.

# Задачи

---

**«В»:** Напишите логическую функцию, которая определяет, являются ли два переданные ей числа взаимно простыми, то есть, не имеющими общих делителей, кроме 1.

**Пример:**

Введите два натуральных числа:

**28 15**

Числа 28 и 15 взаимно простые.

**Пример:**

Введите два натуральных числа:

**28 16**

Числа 28 и 16 не взаимно простые.

# Задачи

---

«С»: Простое число называется гиперпростым, если любое число, получающееся из него откидыванием нескольких цифр, тоже является простым. Например, число 733 – гиперпростое, так как и оно само, и числа 73 и 7 – простые. Напишите логическую функцию, которая определяет, верно ли, что переданное ей число – гиперпростое. Используйте уже готовую функцию `isPrime`, которая приведена в учебнике.

## Пример:

Введите натуральное число:

733

Число 733 гиперпростое.

## Пример:

Введите натуральное число:

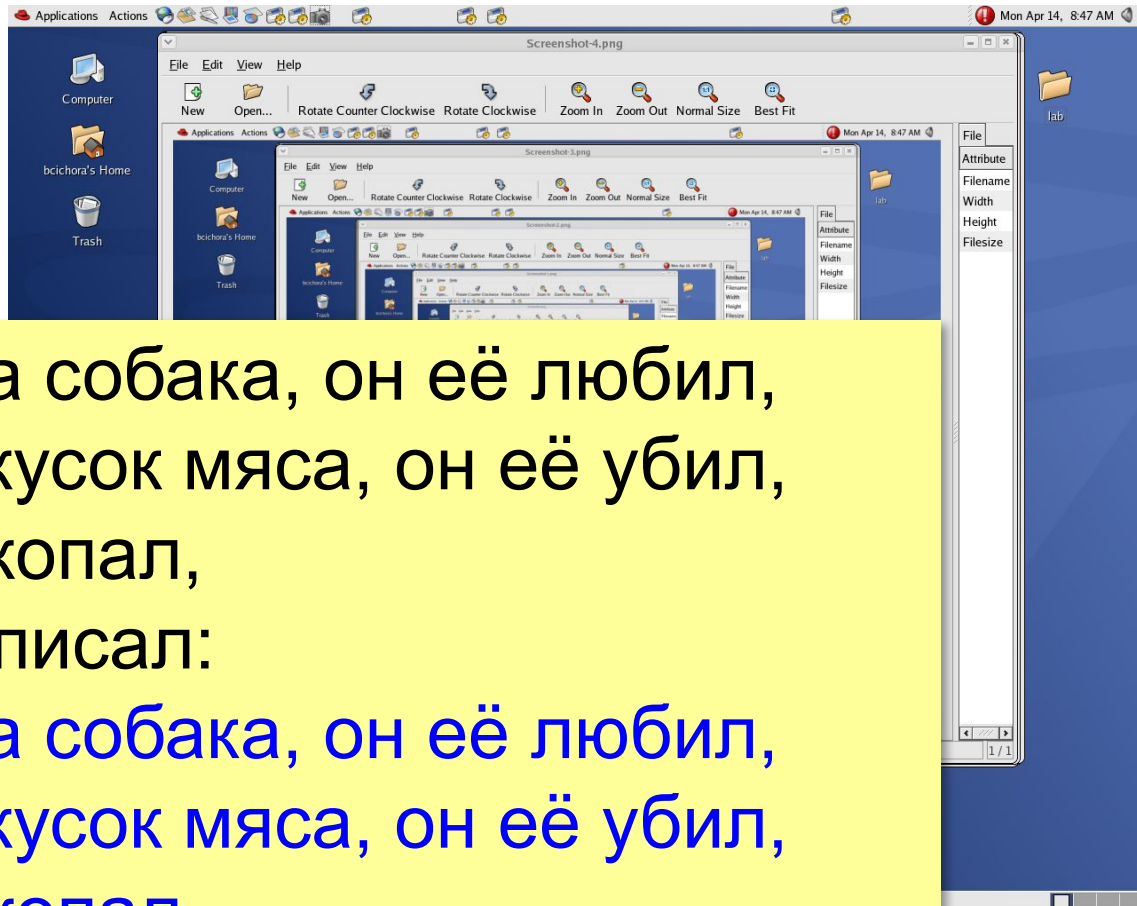
19

Число 19 не гиперпростое.

# Программирование на языке Паскаль

## **§ 61. Рекурсия**

# Что такое рекурсия?



У попа была собака, он её любил,  
 Она съела кусок мяса, он её убил,  
 В землю закопал,  
 Надпись написал:  
 У попа была собака, он её любил,  
 Она съела кусок мяса, он её убил,  
 В землю закопал,  
 Надпись написал:

...

# Что такое рекурсия?

---

## Натуральные числа:

- 1 – натуральное число
- если  $n$  – натуральное число, то  $n + 1$  – натуральное число

индуктивное  
определение

**Рекурсия** — это способ определения множества объектов через само это множество на основе заданных простых базовых случаев.

## Числа Фибоначчи:

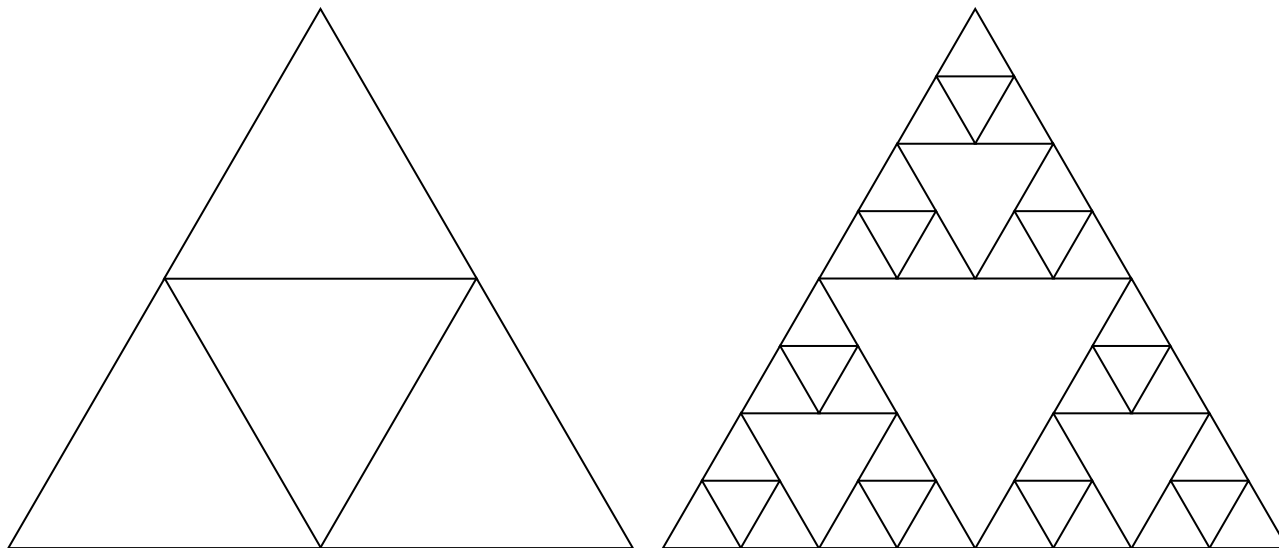
- $F_1 = F_2 = 1$
- $F_n = F_{n-1} + F_{n-2}$  при  $n > 2$

**1, 1, 2, 3, 5, 8, 13, 21, 34, ...**

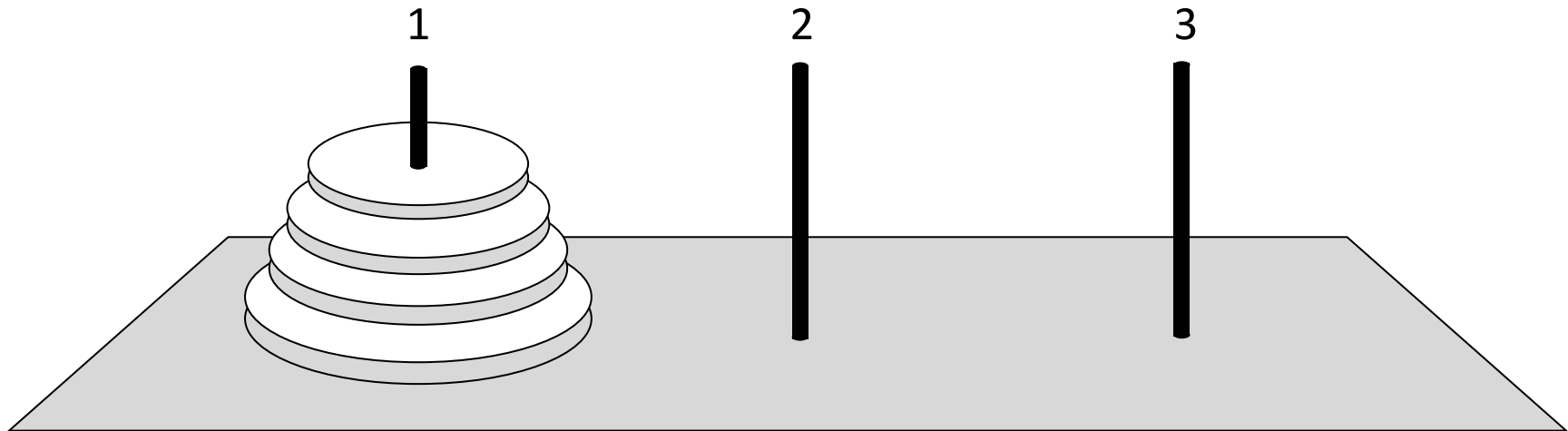
# Фракталы

**Фракталы** – геометрические фигуры, обладающие самоподобием.

**Треугольник Серпинского:**



# Ханойские башни



- за один раз переносится один диск
- класть только меньший диск на больший
- третий стержень вспомогательный

перенести (n, 1, 3)

перенести (n-1, 1, 2)

1 -> 3

перенести (n-1, 2, 3)



# Ханойские башни – процедура

СКОЛЬКО

откуда

куда

```

proceduer Hanoi (n, k, m: integer);
var p: integer;
begin
  p := 6 - k - m;
  Hanoi (n-1, k, p);
  writeln(k, ' -> ', m);
  Hanoi (n-1, p, m)
end;

```

номер вспомогательного  
стержня (1+2+3=6!)

рекурсия

p := 6 - k - m;

Hanoi (n-1, k, p);

рекурсия

writeln(k, ' -> ', m);

Hanoi (n-1, p, m)

end;



Что плохо?



**Рекурсия никогда не остановится!**

# Ханойские башни – процедура

Рекурсивная процедура (функция) — это процедура (функция), которая вызывает сама себя напрямую или через другие процедуры и функции.

```

proceduer Hanoi (n, k, m: integer);
var p: integer;
begin
  if n = 0 then exit;
  p := 6 - k - m;
  Hanoi (n-1, k, p);
  writeln(k, ' -> ', m);
  Hanoi (n-1, p, m)
end;

```

условие выхода из  
рекурсии

```

program HanoiTower;
...
begin
  Hanoi (4, 1, 3)
end.

```

# Вывод двоичного кода числа

```
procedure printBin (n: integer);  
begin  
  if n = 0 then exit;  
  printBin (n div 2);  
  write (n mod 2)  
end;
```

условие выхода из  
рекурсии

напечатать все  
цифры, кроме  
последней

`printBin ( 0 )`



вывести  
последнюю цифру



Как без рекурсии?

# Вычисление суммы цифр числа

```
function sumDig(n: integer): integer;
var sum: integer;
нач
    sum := n mod 10;
    if n >= 10 then
        sum := sum + sumDig(n div 10);
    sumDig := sum
end;
```

последняя цифра

рекурсивный вызов



Где условие окончания рекурсии?

sumDig(1234)

4 + sumDig(123)

4 + 3 + sumDig(12)

4 + 3 + 2 + sumDig(1)

4 + 3 + 2 + 1

# Алгоритм Евклида

**Алгоритм Евклида.** Чтобы найти НОД двух натуральных чисел, нужно вычитать из большего числа меньшее до тех пор, пока меньшее не станет равно нулю. Тогда второе число и есть НОД исходных чисел.

```
function NOD (a, b: integer): integer;  
begin  
  if (a = 0) or (b = 0) then begin  
    NOD := a + b;  
    exit  
  end;  
  if a > b then  
    NOD := NOD (a - b, b)  
  else NOD := NOD (a, b - a)  
end;
```

условие окончания  
рекурсии

рекурсивные вызовы

# Задачи

---

**«А»:** Напишите рекурсивную функцию, которая вычисляет НОД двух натуральных чисел, используя модифицированный алгоритм Евклида.

**Пример:**

Введите два натуральных числа:

**7006652 112307574**

**НОД (7006652 , 112307574) = 1234 .**

**«В»:** Напишите рекурсивную функцию, которая раскладывает число на простые сомножители.

**Пример:**

Введите натуральное число:

**378**

**378 = 2\*3\*3\*3\*7**

# Задачи

---

**«С»:** Дано натуральное число  $N$ . Требуется получить и вывести на экран количество всех возможных *различных* способов представления этого числа в виде суммы натуральных чисел (то есть,  $1 + 2$  и  $2 + 1$  – это один и тот же способ разложения числа 3).  
Решите задачу с помощью рекурсивной процедуры.

**Пример:**

**Введите натуральное число:**

**4**

**Количество разложений: 4.**

# Как работает рекурсия?

Факториал:

$$N! = \begin{cases} 1, & N = 1 \\ N \cdot (N-1)!, & N > 1 \end{cases}$$

```
function Fact(N: integer): integer;
begin
  writeln('-> N = ', N);
  if N <= 1 then
    Fact := 1
  else Fact := N * Fact(N-1);
  writeln('<- N = ', N)
end;
```

```
-> N = 3
   -> N = 2
       -> N = 1
           <- N = 1
               <- N = 2
                   <- N = 3
```

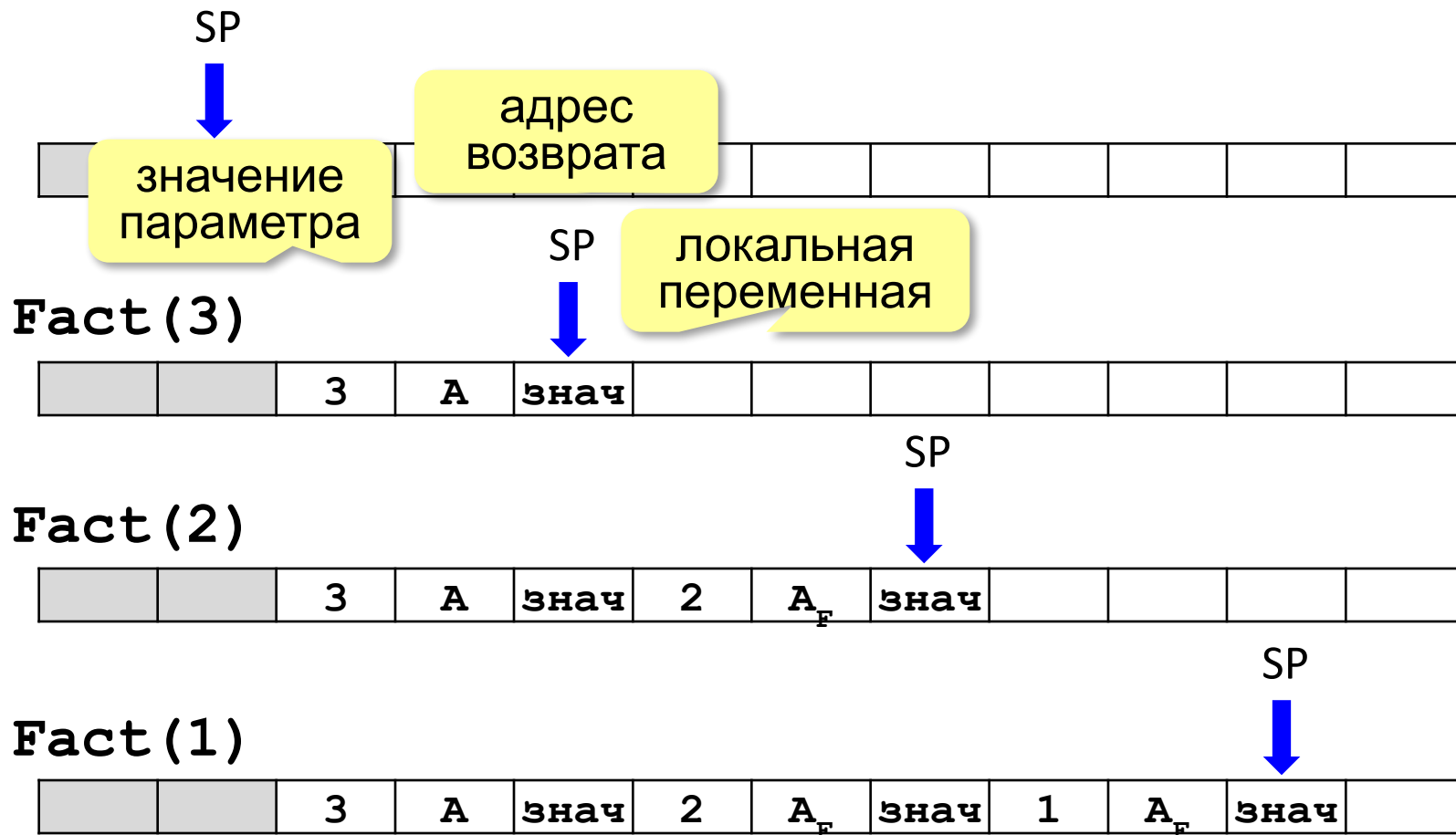


Как сохранить состояние функции перед рекурсивным вызовом?



# Стек

**Стек** – область памяти, в которой хранятся локальные переменные и адреса возврата.



# Рекурсия – «за» и «против»

---

- с каждым новым вызовом расходуется память в стеке (возможно переполнение стека)
- затраты на выполнение служебных операций при рекурсивном вызове



▪ программа становится более короткой и понятной



▪ возможно переполнение стека

▪ замедление работы



Любой рекурсивный алгоритм можно заменить итерационным!

итерационный  
алгоритм

```
function Fact(N: integer) :  
    integer;  
var i, F: integer;  
begin  
    F := 1;  
    for i := 1 to N do  
        F := F * i;  
    Fact := F  
end;
```

# Конец фильма

---

**ПОЛЯКОВ Константин Юрьевич**

д.т.н., учитель информатики

ГБОУ СОШ № 163, г. Санкт-Петербург

[kpolyakov@mail.ru](mailto:kpolyakov@mail.ru)

**ЕРЕМИН Евгений Александрович**

к.ф.-м.н., доцент кафедры мультимедийной

дидактики и ИТО ПГГПУ, г. Пермь

[eremin@pspu.ac.ru](mailto:eremin@pspu.ac.ru)

# Источники иллюстраций

---

1. [old-moneta.ru](http://old-moneta.ru)
2. [www.random.org](http://www.random.org)
3. [www.allruletka.ru](http://www.allruletka.ru)
4. [www.lotterypros.com](http://www.lotterypros.com)
5. [logos.cs.uic.edu](http://logos.cs.uic.edu)
6. [ru.wikipedia.org](http://ru.wikipedia.org)
7. иллюстрации художников издательства «Бином»
8. авторские материалы