

# Программирование (Паскаль)

- § 17. Введение
- § 18. Линейные программы
- § 19. Ветвления
- § 20. Программирование  
циклических алгоритмов
- § 21. Массивы
- § 22. Алгоритмы обработки  
массивов

# Программирование (Паскаль)

## § 17. Введение

# Что такое программирование?

**Программирование** — это создание программ для компьютеров. Этим занимаются **программисты**.

Чем занимаются **программисты**:

**анализ задачи** (выделение исходных данных, связей между ними, этапов решения задачи)

системные аналитики

разработка **алгоритмов**

алгоритмисты

написание и отладка **программ**

кодировщики

**тестирование** программ

тестировщики

написание **документации**

технические писатели

# Направления в программировании

---

**системный программист**

операционные системы,  
утилиты, драйверы

**прикладной программист**

прикладные программы, в  
т.ч. для мобильных  
устройств

**веб-программист**

веб-сайты

**программист баз данных**

системы управления  
базами данных

# Простейшая программа

---

название программы

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

комментарии внутри {}  
не обрабатываются



Что делает эта программа?

# Вывод на экран

```
program Hello;  
begin  
  write ( 'Привет!' );  
end.
```

оператор  
вывода

**Оператор** — это команда  
языка программирования.

```
write ( 'Привет' , Вася! );
```



Что плохо?



```
write ( 'Привет, Вася!' );
```

вся строка в  
апострофах

## Переход на новую строку

---

```
write ( ' Привет , Вася ! ' ) ;  
write ( ' Привет , Петя ! ' ) ;
```

ожидание:

```
Привет , Вася !  
Привет , Петя !
```

реальность:

```
Привет , Вася !Привет , Петя !
```

решение:

```
writeln ( ' Привет , Вася ! ' ) ;  
writeln ( ' Привет , Петя ! ' ) ;
```

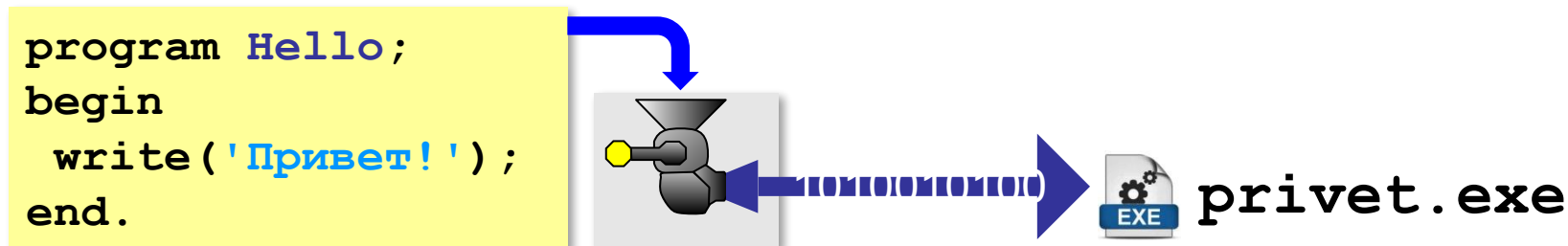
и перейти на  
новую строку

# Системы программирования

**Системы программирования** — это средства для создания новых программ.

**Транслятор** — это программа, которая переводит тексты программ, написанных программистом, в машинные коды (команды процессора).

- **компилятор** — переводит всю программу в машинные коды, строит исполняемый файл (**.exe**)



- **интерпретатор** — сам выполняет программу по частям (по одному оператору).



# Системы программирования

**Отладчик** — это программа для поиска ошибок в других программах.

- **пошаговый режим** — выполнение программы по шагам (по одному оператору)
- **просмотр значений переменных** во время выполнения программы
- **точки останова** – операторы в программе, перед выполнением которых нужно остановиться.

**Среда программирования (IDE):**

- редактор текста программ
- транслятор
- отладчик

# Задачи

---

«В»: Вывести на экран текст «лесенкой»

Вася

пошел

гулять

«С»: Вывести на экран рисунок из букв

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

# Программирование (Паскаль)

## § 18. Линейные программы

## Пример задачи

---

*Задача.* Ввести два числа и вычислить их сумму.

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



Выполнится?

**Псевдокод** – алгоритм на русском языке с элементами языка программирования.



Компьютер не может исполнить псевдокод!

# Зачем нужны переменные?

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

Где запомнить?

**Переменная** — это величина, которая имеет имя, тип и значение. Значение переменной может изменяться во время выполнения программы.

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

объявление переменных

ячейки памяти

# Имена переменных

**Идентификатор** — это имя программы или переменной.

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

заглавные и строчные буквы **НЕ различаются**

**МОЖНО** использовать

- латинские буквы (A-Z, a-z)
- цифры



Имя не может начинаться с цифры!

- знак подчеркивания \_

**НЕЛЬЗЯ** использовать ~~скобки, знаки ", &, |, \*, +, =, !, ? и др.~~

Какие имена правильные?

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

# Работа с переменными

## Присваивание (запись значения)

```
a := 5;
```

оператор  
присваивания

$a \leftarrow 5$

```
a := X;  
a := 18;
```

? Что будет храниться в  $a$ ?

## Вывод на экран

```
write(a);
```

? В чём разница?

```
c := 14;  
write(c);
```

14

```
c := 14;  
write('c');
```

c

# Работа с переменными

## Изменение значения

$$i := i + 1;$$

увеличить на 1

$$i \leftarrow i + 1$$

$$a := 4;$$

$$b := 7;$$

$$a := a + 1;$$

$$b := b + 1;$$

$$a := a + b;$$

$$b := b + a;$$

$$a := a + 2;$$

$$b := b + a;$$

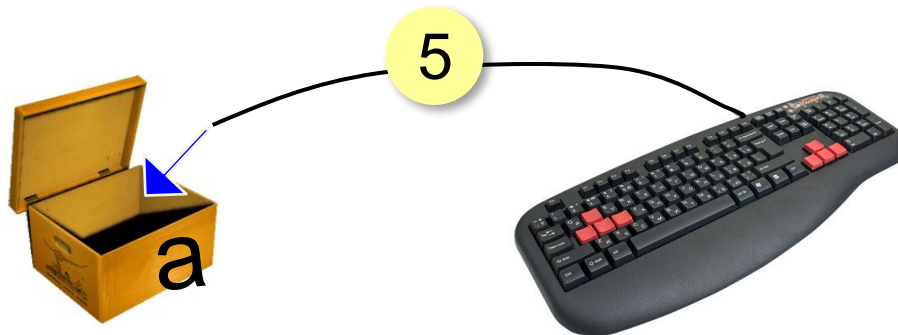
a	b
4	
	7
5	
	8
13	
	21
15	
	36



# Ввод с клавиатуры

Цель – изменить исходные данные, не меняя программу.

```
read(a);
```



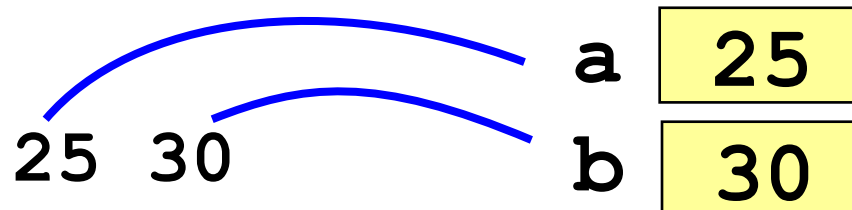
1. Программа ждет, пока пользователь введет значение и нажмет *Enter*.
2. Введенное значение записывается в переменную **a**.

# Ввод с клавиатуры

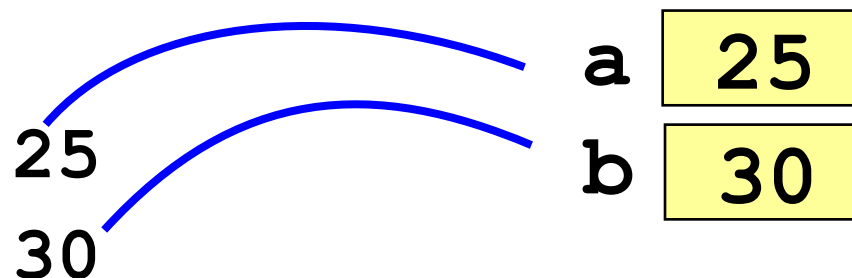
---

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

через пробел:



через *Enter*:



# Программа сложения чисел

```
program Sum;  
var a, b, c: integer;  
begin  
  read(a, b); { ввести два числа }  
  c := a + b;  { вычислить их сумму }  
  write(c) { вывести сумму на экран }  
end.
```



Что плохо?

ожидание:

Введите два числа: 5 7  
5+7=12

реальность:

5 7  
12



Как улучшить диалог?

# write(данных с текстом)

значение  $a$

значение  $b$

значение  $c$

$5+7=12$

ТЕКСТ

```
write(a);  
write('+');  
write(b);  
write('=');  
write(c);
```

→ `write(a, '+', b, '=', c);`

# Программа сложения чисел

---

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



Как переделать для 3-х чисел?

# Задачи

---

**«А»:** Ввести три числа, найти их сумму.

*Пример:*

Введите три числа:

4

5

7

$$4+5+7=16$$

**«В»:** Ввести три числа, найти их сумму и произведение.

*Пример:*

Введите три числа:

4

5

7

$$4+5+7=16$$

$$4*5*7=140$$

# Задачи

---

**«С»:** Ввести три числа, найти их сумму, произведение и среднее арифметическое.

*Пример:*

Введите три числа:

4

5

7

$$4+5+7=16$$

$$4*5*7=140$$

$$(4+5+7) / 3 = 5.333333$$

# Арифметические выражения

$$a \leftarrow \frac{c + b - 1}{2} \cdot d$$

Линейная запись (в одну строку):

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

Операции: + -

\* – умножение

/ – деление

\*\* – возведение в степень ( $x^2 \rightarrow \mathbf{x**2}$ )

```
var x, a, b: integer;  
read(a, b);  
x := a / b;
```



Что плохо?

```
var x: real;
```



# Порядок выполнения операций

---

- 1) действия в скобках
- 2) возведение в степень
- 3) умножение и деление, слева направо
- 4) сложение и вычитание, слева направо

6      5      2      1      3      4

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

## Частное и остаток

---

**div** – деление нацело (остаток отбрасывается)

**mod** – остаток от деления

175 сек = 2 мин 55 сек



Как получить 2 и 55?

```
var t, m, s: integer;
```

```
t := 175;
```

```
m := t div 60;
```

```
s := t mod 60;
```

# Частное и остаток

---



Что получится?

```
n := 123
```

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

```
k := n mod 10;
```

При делении на 10 нацело отбрасывается последняя цифра числа.

Остаток от деления на 10 – это последняя цифра числа.

# ФОРМАТНЫЙ ВЫВОД

```
var a, b, c: integer;
a:=1; b:=2; c:=3;
write(a, b, c);
```

123

```
write(a, ' ', b, ' ', c);
```

1 2 3

```
write(a, b:3, c:5);
```

1 2 3  
 3 5

КОЛИЧЕСТВО ЗНАКОВ  
НА ВЫВОД ЧИСЛА



Сколько знаков для вывода *a*?

# Задачи

---

**«А»:** Ввести число, обозначающее количество секунд.  
Вывести то же самое время в минутах и секундах.

**Пример:**

Введите число секунд: **175**  
2 мин. 55 с.

**«В»:** Ввести число, обозначающее количество секунд.  
Вывести то же самое время в часах, минутах и секундах.

**Пример:**

Введите число секунд: **8325**  
2 ч. 18 мин. 45 с

# Задачи

---

**«С»:** Занятия в школе начинаются в 8-30. Урок длится 45 минут, перерывы между уроками – 10 минут. Ввести номер урока и вывести время его окончания.

**Пример:**

**Введите номер урока : 6**  
**13-50**

# ФОРМАТНЫЙ ВЫВОД

```
var x: real;
x:=12.34567891234;
write(x);
```

вариант:

12.345679

6

по умолчанию

```
write(x:10:3);
```

```
____12.346
```

всего на  
число

в дробной  
части

3

10

```
write(x:8:2);
```

```
____12.34
```

```
write(x:2:2);
```

12.34

```
write(x:0:1);
```

12.3

МИНИМАЛЬНО  
ВОЗМОЖНОЕ

# Научный формат чисел

```
var x: real;  
x:=123456789;  
write(x);
```



1.234568e+008

1,234568 · 10<sup>8</sup>

```
var x: real;  
x:=0.0000123456789;  
write(x);
```



1.234568e-005

1,234568 · 10<sup>-5</sup>

КОЛИЧЕСТВО ЗНАКОВ  
МОЖЕТ ОТЛИЧАТЬСЯ



## Операции с вещественными числами

---

**trunc** – целая часть числа (дробная часть отбрасывается)

**round** – округление к ближайшему целому

**frac** – дробная часть

```
x := 1.6;
```

```
write (trunc (x) ) ;
```

1

```
write (round (x) ) ;
```

2

```
write (frac (x) ) ;
```

0.6

# Операции с вещественными числами

---

`sqrt` – квадратный корень

```
x := 2.25;  
write(sqrt(x));
```



```
1.5
```

# Операции с вещественными числами

$$1/3 = 0,33333\dots$$

бесконечно много знаков



Большинство вещественных чисел хранятся в памяти компьютера с ошибкой!

```
var x, y, z: real;  
x := 1/2;  
y := 1/3;  
z := 5/6; { 5/6=1/2+1/3 }  
write (x+y-z);
```

-1.110223e-016

# Задачи

---

**«А»:** Ввести число, обозначающее размер одной фотографии в Мбайтах. Определить, сколько фотографий поместится на флэш-карту объёмом 2 Гбайта.

## Пример:

Размер фотографии в Мбайтах: **6.3**

Поместится фотографий: 325.

# Задачи

---

«В»: Оцифровка звука выполняется в режиме стерео с частотой дискретизации 44,1 кГц и глубиной кодирования 24 бита. Ввести время записи в минутах и определить, сколько Мбайт нужно выделить для хранения полученного файла (округлить результат в большую сторону).

## Пример:

Введите время записи в минутах: **10**

Размер файла **152 Мбайт**

# Задачи

---

«С»: Разведчики-математики для того, чтобы опознать своих, используют числовые пароли. Услышав число-пароль, разведчик должен возвести его в квадрат и сказать в ответ первую цифру дробной части полученного числа. Напишите программу, которая по полученному паролю (вещественному числу) вычисляет число-ответ.

**Пример:**

**Введите пароль: 1.92**

**Ответ: 6**

потому что  $1,92^2 = 3,6864\dots$ , первая цифра дробной части – 6

# Случайные и псевдослучайные числа

## Случайные явления

- встретил слона – не встретил слона
- жеребьёвка на соревнованиях
- лотерея
- случайная скорость (направление выстрела ) в игре
- ...



**Случайные числа** — это последовательность чисел, в которой невозможно предсказать следующее число, даже зная все предыдущие.

# Случайные и псевдослучайные числа



Компьютер неслучаен!

**Псевдослучайные числа** — похожи на случайные, но строятся по формуле.

следующее

предыдущее

$$X_{n+1} := \text{mod}(a * X_n + b, c) \quad | \quad \text{от } 0 \text{ до } c-1$$

$$X_{n+1} := \text{mod}(X_n + 3, 10) \quad | \quad \text{от } 0 \text{ до } 9$$

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

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

зерно

зацикливание



# Датчик случайных чисел

## Целые числа на отрезке:

цел  $K, L$

$K := \text{irand}(1, 6)$  | отрезок  $[0, 6]$

$L := \text{irand}(1, 6)$  | это уже другое число!

англ. *integer* – целый  
*random* – случайный

## Вещественные числа в полуинтервале:

цел  $x, y$

$x := \text{rand}(0, 10)$  | полуинтервал  $[0, 10)$

$y := \text{rand}(0, 10)$  | это уже другое число!

# Датчик случайных чисел

## Целые числа на отрезке:

*random* – случайный

```
var K, L, M: integer;  
K := random(6); { отрезок [0,5] }  
L := random(6)+1; { отрезок [1,6] }  
M := random(b-a+1)+a; { отрезок [a,b] }
```

## Вещественные числа в полуинтервале:

```
var x, y, z, w: real;  
x := random; { полуинтервал [0,1) }  
y := 7*random; { [0,7) }  
z := 7*random + 5; { [5,12) }  
w := (b-a)*random + a; { [a,b) }
```

# Задачи

---

- «А»: В игре «Русское лото» из мешка случайным образом выбираются бочонки, на каждом из которых написано число от 1 до 90. Напишите программу, которая выводит наугад первые 5 выигрышных номеров.
- «В»: + Доработайте программу «Русское лото» так, чтобы все 5 значений гарантированно были бы разными (используйте разные диапазоны).

# Задачи

---

**«С»:** + Игральный кубик бросается три раза (выпадает три случайных значения). Из этих чисел составляется целое число, программа должна найти его квадрат.

## Пример:

Выпало очков :

1 2 3

Число 123

Его квадрат 15129

# Задачи

---

**«D»:** + Получить случайное трёхзначное число и вывести в столбик его отдельные цифры.

**Пример:**

Получено число 123

сотни: 1

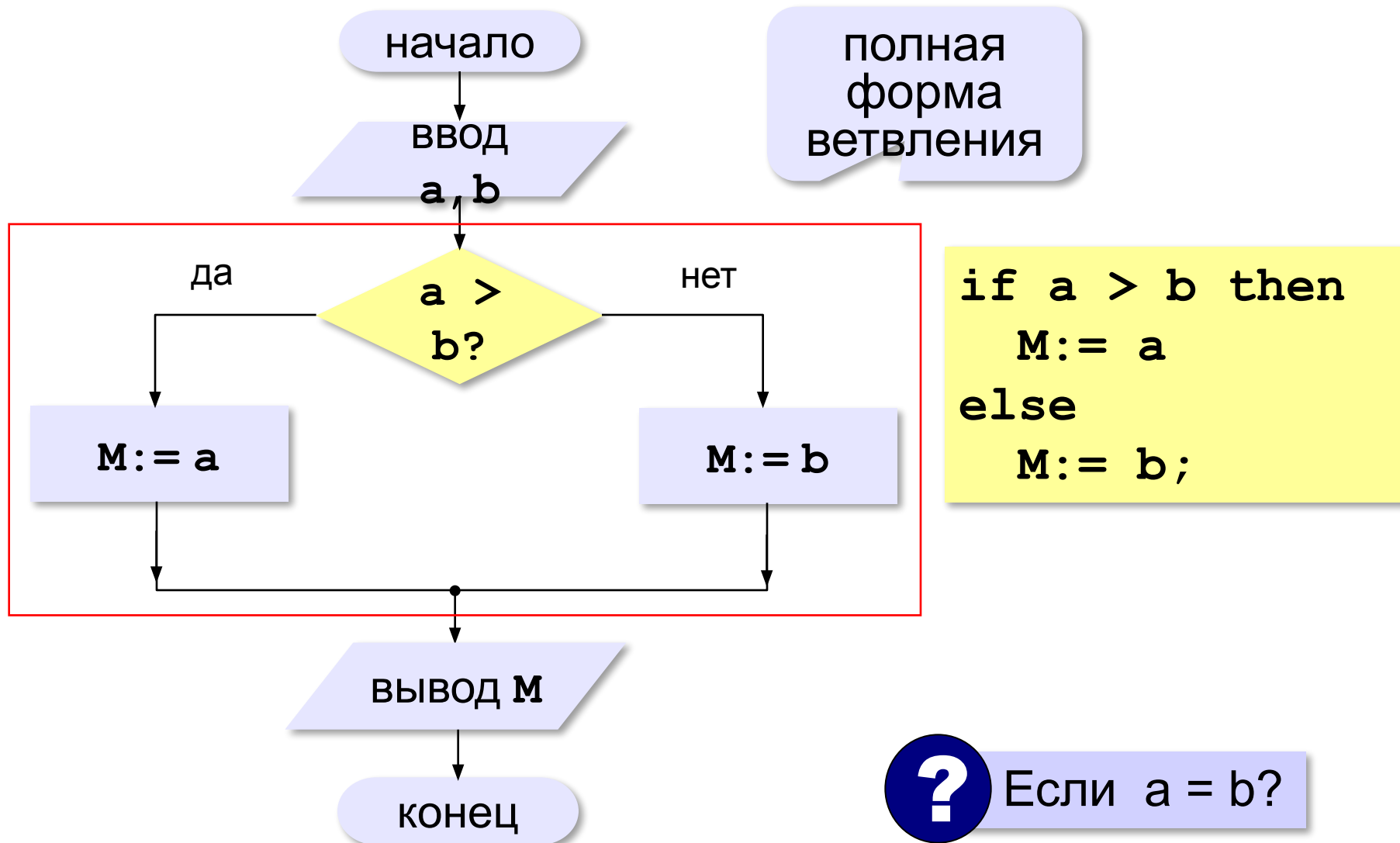
десятки: 2

единицы: 3

# Программирование (Паскаль)

## § 19. Ветвления

# Выбор наибольшего из двух чисел



# Вариант 1. Программа

```
program Maximum;  
var a, b, M: integer;  
begin  
  writeln('Введите два целых числа');  
  read(a, b);  
  if a > b then  
    M := a  
  else  
    M := b;  
  writeln('Наибольшее число ', M);  
end.
```

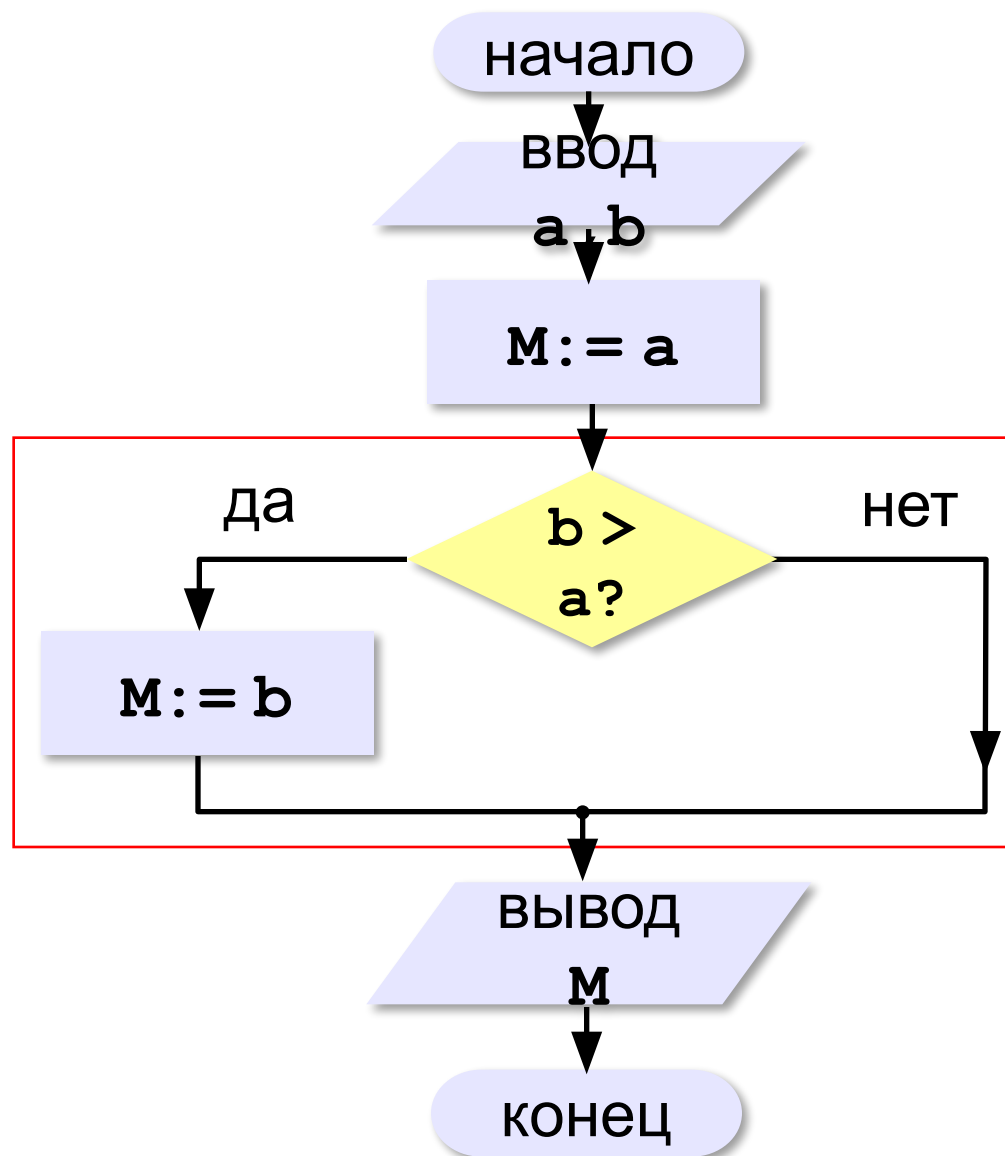
полная форма  
условного  
оператора



Перед **else** не ставится  
точка с запятой!



# Выбор наибольшего из двух чисел-2



неполная  
форма  
ветвления

## Вариант 2. Программа

```
program Maximum2;  
var a, b, M: integer;  
begin  
  writeln('Введите два целых числа');  
  read(a, b);  
  M := a;  
  if b > a then  
    M := b;  
  writeln('Наибольшее число ', M);  
end.
```

неполная  
форма  
условного  
оператора

# Примеры

## Поиск минимального:

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



Что плохо?



Когда работает неверно?

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



Что делает эта программа?

составной  
оператор



Перед end можно не ставить  
точку с запятой!

# В других языках программирования

---

## Python:

```
if a < b:  
    c = a  
    a = b  
    b = c
```

## C:

```
if (a < b) {  
    c = a;  
    a = b;  
    b = c;  
}
```

# Вложенные условные операторы

*Задача.* В переменной **a** записан возраст Антона, а в переменной **b** – возраст Бориса. Определить, кто из них старше.



Сколько вариантов ответа?

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

вложенный  
условный  
оператор

**else** относится к  
ближайшему **if**

# Задачи

---

**«А»:** Ввести два целых числа, найти наибольшее и наименьшее из них.

**Пример:**

Введите два целых числа :

1 5

Наибольшее число 5

Наименьшее число 1

**«В»:** Ввести четыре целых числа, найти наибольшее из них.

**Пример:**

Введите четыре целых числа :

1 5 4 3

Наибольшее 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.** Фирма набирает сотрудников от 25 до 40 лет включительно. Ввести возраст человека и определить, подходит ли он фирме (вывести ответ 'подходит' или 'не подходит').

**Особенность:** надо проверить, выполняются ли два условия одновременно:

**возраст  $\geq 25$**

**возраст  $\leq 40$**



Можно ли решить известными методами?



## Плохое решение

```
program Work;  
var v: integer;  
begin  
  writeln('Введите ваш возраст');  
  read(v);  
  if v >= 25 then  
    if v <= 40 then  
      write('Подходит!')  
    else  
      write('Не подходит.')  else  
    write('Не подходит.');end.
```

вложенный  
условный  
оператор

## Хорошее решение (операция «И»)

```
program Work;  
var v: integer;  
begin  
  writeln('Введите ваш возраст');  
  read(v);  
  if (v >= 25) and (v <= 40) then  
    write('Подходит!')  
  else  
    write('Не подходит.');
```

end.

сложное  
условие



Каждое условие – в скобки!

## Примеры

---

Задача. Вывести 'Да', если число в переменной  $a$  – двузначное.

```
if (10 <= a) and (a <= 99) then  
    write('Да');
```

Задача. Вывести 'Да', если число в переменной  $a$  – двузначное и делится на 7.

```
if (10 <= a) and (a <= 99)  
    and (a mod 7 = 0) then  
    write('Да');
```

## Сложные условия

**Задача.** Самолёт летает по понедельникам и четвергам.  
Ввести номер дня недели и определить, летает ли в этот день самолёт.

**Особенность:** надо проверить, выполняется ли **одно из двух** условий:

день = 1

день = 4

```
if (d = 1) or (d = 4) then  
    write ('Летает')  
else  
    write ('Не летает');
```

СЛОЖНОЕ  
УСЛОВИЕ

## Ещё пример

---

**Задача.** Фирма набирает сотрудников от 25 до 40 лет включительно. Ввести возраст человека и определить, подходит ли он фирме (вывести ответ 'подходит' или 'не подходит'). Использовать «ИЛИ».

```
if (x < 25) or (x > 40) then
  write('Не подходит!')
else
  write('Подходит.');
```

# Простые и сложные условия

Простые условия (отношения) равно

<    <=    >    >=    =    <>    не равно

**Сложное условие** – это условие, состоящее из нескольких простых условий (отношений), связанных с помощью логических операций:

- **И** – одновременное выполнение условий

$x \geq 25 \text{ and } x \leq 40$

- **ИЛИ** – выполнение хотя бы одного из условий

$x \leq 25 \text{ or } x \geq 40$

- **НЕ** – отрицание, обратное условие

$\text{not } (x > 25) \iff x \leq 25$

# Порядок выполнения операций

---

- выражения в скобках
- НЕ (**not**)
- И (**and**)
- ИЛИ (**or**) , исключающее ИЛИ (**xor**)

```
      4      1      6      2      5      3  
if not(a > 2) or (c <> 5) and (b < a) then  
    ...
```

# Сложные условия

Истинно или ложно при  $a := 2$ ;  $b := 3$ ;  $c := 4$ ;

**not** (a > b)

Да

(a < b) **and** (b < c)

Да

(a > c) **or** (b > c)

Нет

(a < b) **and** (b > c)

Нет

(a > c) **and** (b > d)

Нет

**not** (a >= b) **or** (c = d)

Да

(a >= b) **or not** (c < b)

Да

(a > c) **or** (b > c) **or** (b > a)

Да



# Задачи

---

**«А»:** Напишите программу, которая получает три числа - рост трёх спортсменов, и выводит сообщение «По росту.», если они стоят по возрастанию роста, или сообщение «Не по росту!», если они стоят не по росту.

**Пример:**

Введите рост трёх спортсменов:

**165 170 172**

По росту.

**Пример:**

Введите рост трёх спортсменов:

**175 170 172**

Не по росту!

# 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.

# Логические переменные

```
var b: boolean;  
...  
b := True;   { 1 }  
b := False; { 0 }
```

ТОЛЬКО ДВА  
ВОЗМОЖНЫХ  
ЗНАЧЕНИЯ

Пример:

```
var vyh: boolean;  
...  
vyh := (d=6) or (d=7);  
...  
if not vyh then  
    write('Рабочий день.')else  
    write('Выходной!');
```

# Задачи

---

**«А»:** Напишите программу, которая получает с клавиатуры целое число и записывает в логическую переменную значение «да» (True), если это число трёхзначное. После этого на экран выводится ответ на вопрос: «Верно ли, что было получено трёхзначное число?».

**Пример:**

Введите число: **165**

Ответ: да.

**Пример:**

Введите число: **1651**

Ответ: нет.

# Задачи

---

**«В»:** Напишите программу, которая получает с клавиатуры трёхзначное число и записывает в логическую переменную значение «да» (True), если это число – палиндром, то есть читается одинаково слева направо и справа налево. После этого на экран выводится ответ на вопрос: «Верно ли, что введённое число – палиндром?».

**Пример:**

Введите число: **165**

Ответ: нет.

**Пример:**

Введите число: **656**

Ответ: да.

# Задачи

---

**«С»:** Напишите программу, которая получает с клавиатуры трёхзначное число и записывает в логическую переменную значение «да» (True), если это все его цифры одинаковы. После этого на экран выводится ответ на вопрос: «Верно ли, что все цифры введённого числа одинаковы?»

**Пример:**

Введите число: **161**

Ответ: нет.

**Пример:**

Введите число: **555**

Ответ: да.

# Экспертная система

**Экспертная система** — это компьютерная программа, задача которой — заменить человека-эксперта при принятии решений в сложной ситуации.

**База знаний** = факты + правила writea:

- если у животного есть перья, то это **птица**;
- если животное кормит детенышей молоком, то это — **млекопитающее**;
- если животное — млекопитающее и ест мясо, то это — **хищник**.

**Диалог:**

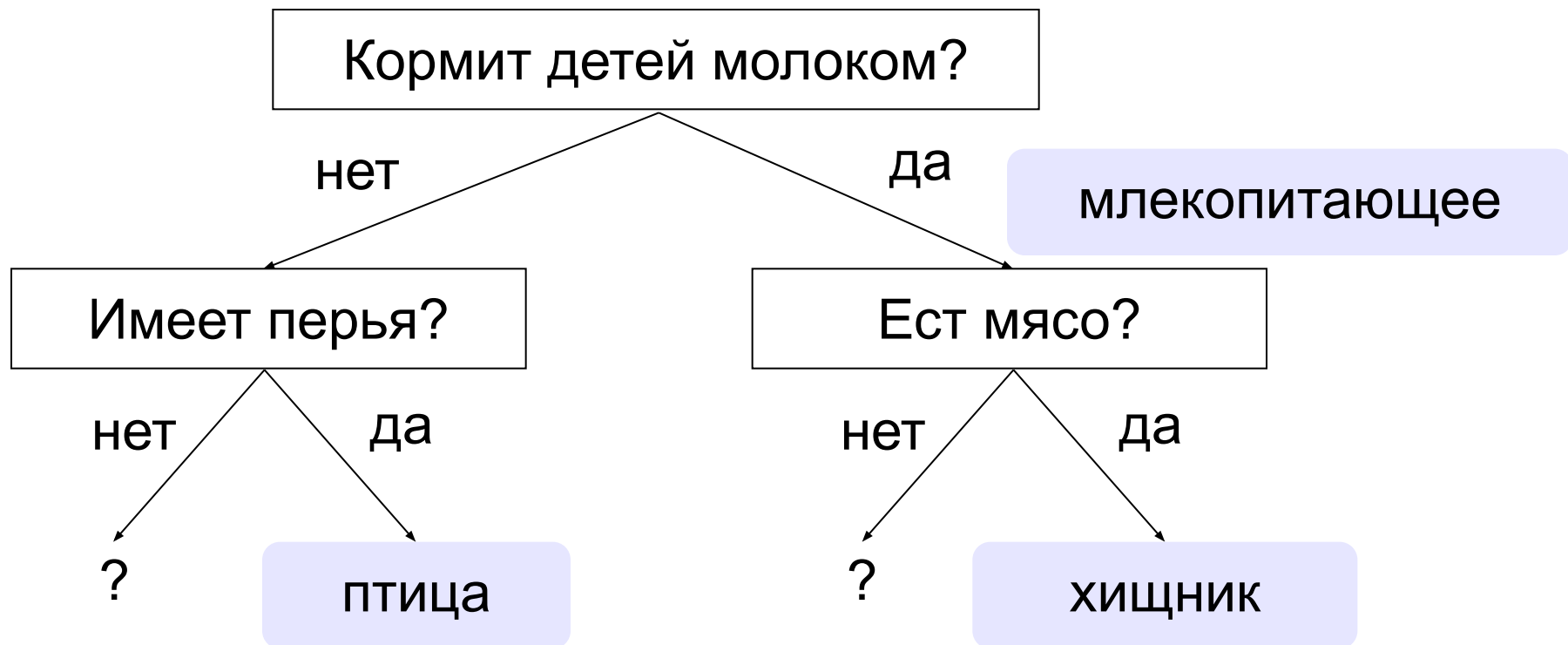
Это животное кормит детей молоком? **Нет**

Это животное имеет перья? **Да**

Это **птица**.



# Дерево решений



# Программирование экспертной системы

Ответы пользователя: **да** и **нет** – символьные строки.

```
var otvet: string;
```

```
...
```

```
write('Кормит детей молоком? ');
```

```
read(otvet);
```

```
if otvet = 'да' then
```

```
    ... { вариант 1 }
```

```
else
```

```
    ... { { вариант 1 }
```

```
        writeln('Млекопитающее.');
```

```
        write('Ест мясо? ');
```

```
        read(otvet);
```

```
        if otvet = 'да' then
```

```
            writeln('Хищник.')
```

```
        else
```

```
            writeln('Не знаю.');
```

# Заглавные и строчные буквы

```
var ответ: string;  
...  
if ответ = 'да' then  
...
```

не работает  
на 'Да'



Как исправить?

```
if (ответ = 'да') or (ответ = 'Да') then  
...
```

Ещё лучше:

```
if LowerCase(ответ) = 'да' then  
...
```

преобразовать все  
заглавные в строчные

```
if UpperCase(ответ) = 'ДА' then  
...
```

# Программирование (Паскаль)

## § 23. Отладка программ

# Виды ошибок

---

**Синтаксические** ошибки – нарушение правил записи операторов языка программирования.

Обнаруживаются транслятором.

**Логические** ошибки – неверно составленный алгоритм.



**Отказ** (ошибка времени выполнения) – аварийная ситуация во время выполнения программы.

**Отладка** – поиск и исправление ошибок в программе.

## Пример отладки программы

---

Программа решения квадратного уравнения

$$ax^2 + bx + c = 0$$

```
program SqEq;  
var a, b, c, D, x1, x2: real;  
begin  
  write('Введите a, b, c: ');  
  read(a, b, c);  
  D:=b*b-4*a*a;  
  x1:=(-b+sqrt(D))/2*a;  
  x2:=(-b-sqrt(D))/2*a;  
  writeln('x1=', x1, ' x2=', x2);  
end.
```

# Тестирование

Тест 1.  $a = 1, b = 2, c = 1.$

Ожидание:

$x1=-1.0 \quad x2=-1.0$

Реальность:

$x1=-1.0 \quad x2=-1.0$



Тест 2.  $a = 1, b = -5, c = 6.$

$x1=3.0 \quad x2=2.0$

$x1=4.791 \quad x2=0.209$



Найден вариант, когда программа работает неверно.  
Ошибка **воспроизводится!**

## Возможные причины:

- неверный ввод данных
- неверное вычисление дискриминанта
- неверное вычисление корней
- неверный write(результатов)

$$D = b^2 - 4ac$$

$$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$$

# Отладочная печать

Идея: выводить все промежуточные результаты.

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

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

```
D:=b*b-4*a*a;
```

```
writeln('D=', D);
```

```
...
```

Результат:

```
Введите a, b, c: 1 -5 6
```

```
1.0 -5.0 6.0
```

```
D=21.0
```

$$D = b^2 - 4ac = 25 - 4 \cdot 1 \cdot 6 = 1$$

```
D:=b*b-4*a*c;
```



Одна ошибка найдена!



# Отладка программы

Тест 1.  $a = 1, b = 2, c = 1.$

Ожидание:

`x1=-1.0 x2=-1.0`

Реальность:

`x1=-1.0 x2=-1.0`



Тест 2.  $a = 1, b = -5, c = 6.$

`x1=3.0 x2=2.0`

`x1=3.0 x2=2.0`



Программа работает верно?

Тест 3.  $a = 8, b = -6, c = 1.$

`x1=0.5 x2=0.25`

`x1=32.0 x2=16.0`



`x1 := (-b+sqrt(D)) / (2*a);`

`x2 := (-b-sqrt(D)) / (2*a);`



Что неверно?

# Задачи

---

«А»: Загрузите программу, которая должна вычислять сумму цифр трёхзначного числа:

```
var N, d1, d2, s: integer;
```

```
begin
```

```
  read('N = ');
```

```
  write(N);
```

```
  d0 := N mod 10;
```

```
  d1 := N mod 100;
```

```
  d2 := N div 100;
```

```
  d0 + d2 := s
```

```
  writeln(s);
```

```
end.
```

Выполните отладку программы:

- исправьте синтаксические ошибки
- определите ситуации, когда она работает неверно
- исправьте логические ошибки.

# Задачи

---

«В»: Доработайте программу из п. А так, чтобы она правильно работала с отрицательными трёхзначными числами: при вводе числа «−123» программа должна выдавать ответ 6.

# Задачи

---

«С»: Загрузите программу, которая должна вычислять наибольшее из трёх чисел:

```
var a, b: integer;  
begin  
  read('a = '); read(a);  
  write('b = '); write(b);  
  read('c = '); read(c);  
  if a > b then M:=a  
  else M:= b; end;  
  if c > b then M:= b  
  else M:= c; end;  
  writeln(M);  
end.
```

Выполните отладку программы:

- исправьте синтаксические ошибки
- определите ситуации, когда она работает неверно
- исправьте логические ошибки.

# Программирование (Паскаль)

## **§ 20. Программирование циклических алгоритмов**

# Зачем нужен цикл?

Задача. Вывести 5 раз «Привет!».

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



А если 5000?

Цикл «N раз»:

```
{ сделай 5 раз }  
  writeln ( ' Привет ' ) ;
```



В Паскале нет такого цикла!



# Как организовать цикл?

**!** Нужно запоминать, сколько раз цикл уже выполнен!



```
{счётчик := 0 }  
{пока счётчик < 5 }  
  writeln( 'Привет' );  
  {счётчик := счётчик + 1}
```

ещё не делали

сделали ещё раз

```
var count: integer;  
count := 0;  
while count < 5 do begin  
  writeln( 'Привет' );  
  count := count + 1  
end;
```

составной оператор

# Как организовать цикл?

---

Идея: запоминать, сколько шагов осталось.

```
count := 5;  
while count > 0 do begin  
    writeln('Привет');  
    count := count - 1  
end;
```



## Цикл с предусловием

- условие проверяется при входе в цикл
- как только условие становится ложным, работа цикла заканчивается
- если условие ложно в самом начале, цикл не выполняется **ни разу**

```
while условие do begin  
    . . .  
end;
```

тело цикла



Если условие никогда не станет ложно?

```
while True do begin  
    . . .  
end;
```

бесконечный цикл  
(заикливание)

# Сумма цифр числа

---

*Задача.* Вычислить сумму цифр введённого числа.

$$123 \rightarrow 1 + 2 + 3 = 6$$

Выделить последнюю цифру числа в переменной  $N$ :

$$d := N \bmod 10; \quad 123 \rightarrow 3$$

Отбросить последнюю цифру числа в переменной  $N$ :

$$N := N \operatorname{div} 10; \quad 123 \rightarrow 12$$

Добавить к переменной  $sum$  значение переменной  $d$ :

$$sum := sum + d; \quad sum = 6 \rightarrow 6 + 4 = 10$$
$$d = 4$$

# Сумма цифр числа

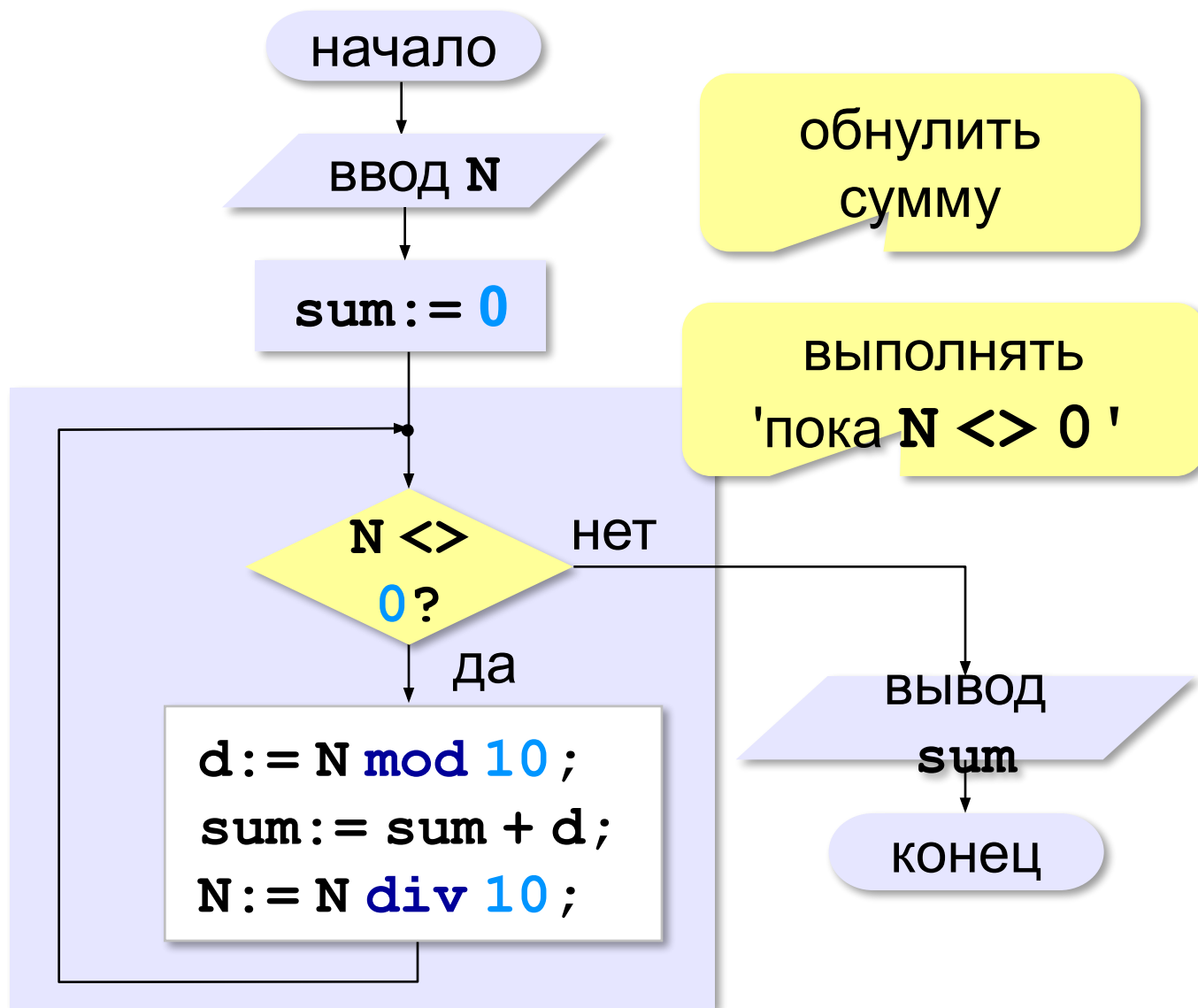
---

- выделяем последнюю цифру числа (**mod**)
- увеличиваем сумму на значение цифры (**sum:=sum+d;**)
- отсекаем последнюю цифру числа (**div**)

N	d	sum
123		0

начальные значения

# Сумма цифр числа



# Сумма цифр числа

```
program SumDigits; , N1
var N, d, sum: integer;
begin
  writeln('Введите целое число');
  read(N);
  sum := 0; N1 := N;
  while N <> 0 do begin
    d := N mod 10;
    sum := sum + d;
    N := N div 10
  end;
  write('Сумма цифр числа ', N1, ' равна ', sum);
end.
```



Что плохо?

# Задачи

---

**«А»:** Напишите программу, которая получает с клавиатуры количество повторений и выводит столько же раз какое-нибудь сообщение.

**Пример:**

Сколько раз повторить? **3**

Привет!

Привет!

Привет!

**«В»:** Напишите программу, которая получает с клавиатуры натуральное число и определяет, сколько раз в его десятичной записи встречается цифра 1.

**Пример:**

Введите число? **311**

Единиц: **2**

# Задачи

---

**«С»:** Напишите программу, которая получает с клавиатуры натуральное число и находит наибольшую цифру в его десятичной записи.

**Пример:**

Введите число: **311**

Наибольшая цифра: **3**

**«D»:** Напишите программу, которая получает с клавиатуры натуральное число и определяет, есть ли в его десятичной записи одинаковые цифры, стоящие рядом.

**Пример:**

Введите число: **553**

Ответ: **да.**

Введите число: **535**

Ответ: **нет.**

# Алгоритм Евклида

**Задача.** Найти наибольший общий делитель (НОД) двух натуральных чисел.

Заменяем большее из двух чисел **разностью** большего и меньшего до тех пор, пока они не станут равны. Это и есть НОД.

$$\begin{aligned}\text{НОД}(a, b) &= \text{НОД}(a-b, b) \\ &= \text{НОД}(a, b-a)\end{aligned}$$



Евклид  
(365-300 до. н. э.)

**Пример:**

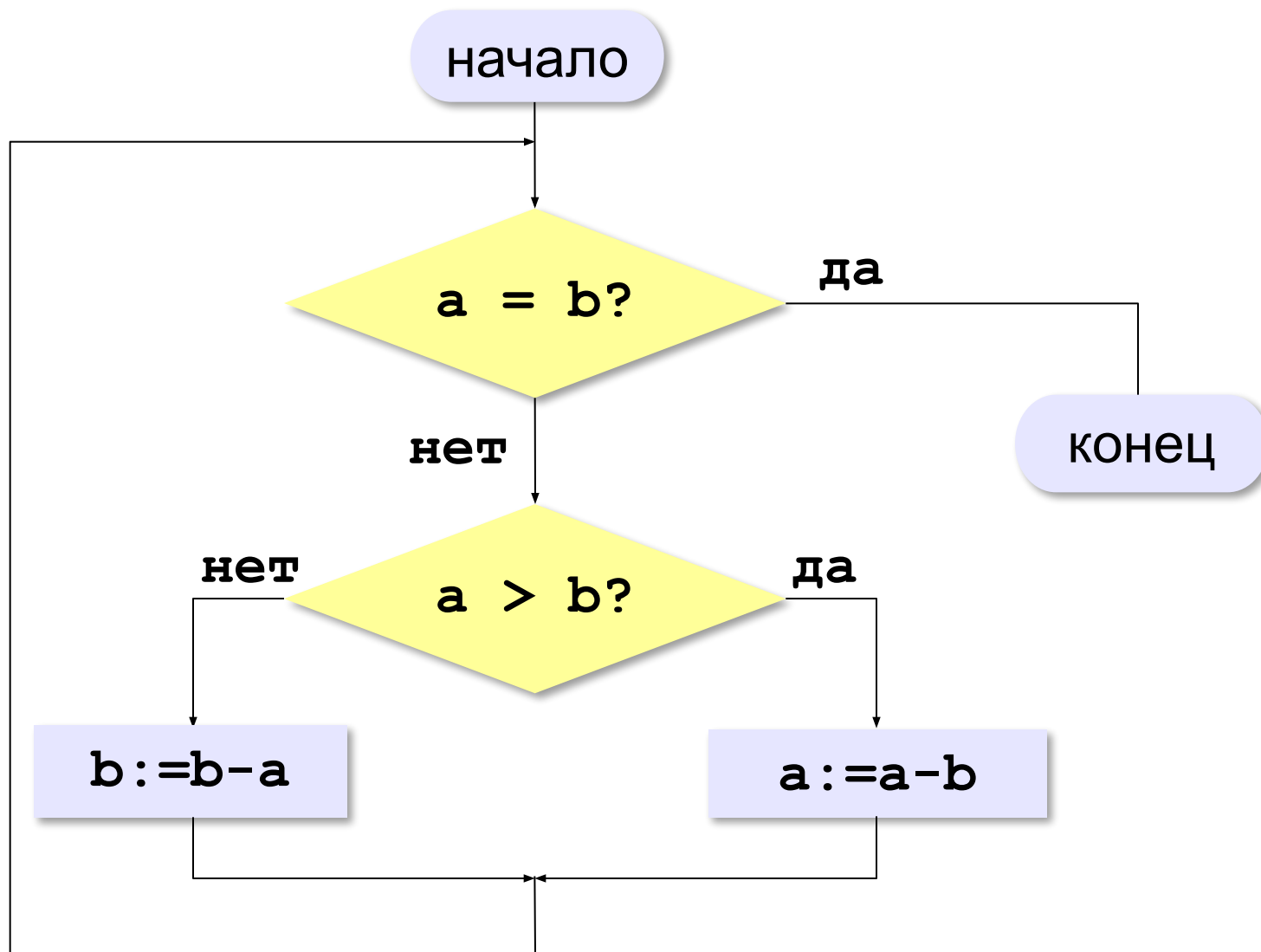
$$\begin{aligned}\text{НОД}(14, 21) &= \text{НОД}(14, 21-14) = \text{НОД}(14, 7) \\ &= \text{НОД}(7, 7) = 7\end{aligned}$$

⊖ много шагов при большой разнице чисел:

$$\text{НОД}(1998, 2) = \text{НОД}(1996, 2) = \dots = 2$$




# Алгоритм Евклида





# Алгоритм Евклида

---

```
while a <> b do
  if a > b then
    a := a - b
  else b := b - a;
```

 Где будет НОД? Как его вывести?

 Как вывести НОД в формате  $\text{НОД}(14,21) = 7$ ?

 А без дополнительных переменных?

# Модифицированный алгоритм Евклида

---

Заменяем большее из двух чисел **остатком от деления** большего на меньшее до тех пор, пока меньшее не станет **равно нулю**. Тогда большее — это НОД.

$$\begin{aligned}\text{НОД}(a, b) &= \text{НОД}(\text{mod}(a, b), b) \\ &= \text{НОД}(a, \text{mod}(b, a))\end{aligned}$$

**Пример:**

$$\text{НОД}(14, 21) = \text{НОД}(14, 7) = \text{НОД}(0, 7) = 7$$

# Модифицированный алгоритм

```
while (a <> 0) and (b <> 0) do
  if a > b then
    a := a mod b
  else
    b := b mod a;
```



Где будет НОД? Как его вывести?

```
if a <> 0 then
  write(a)
else
  write(b);
```



```
write ( a+b );
```

# В других языках программирования

---

## Python:

```
while a!=0 and b!=0:  
    if a > b:  
        a = a % b  
    else:  
        b = b % a
```

## C:

```
while (a!=0 && b!=0)  
{  
    if (a > b)  
        a = a % b;  
    else  
        b = b % a;  
}
```

# Задачи

---

«А»: Ввести с клавиатуры два натуральных числа и найти их НОД с помощью алгоритма Евклида.

**Пример:**

Введите два числа:

21 14

НОД (21 , 14) =7

«В»: Ввести с клавиатуры два натуральных числа и найти их НОД с помощью **модифицированного** алгоритма Евклида. Заполните таблицу:

a	64168	358853	6365133	17905514	549868978
b	82678	691042	11494962	23108855	298294835
НОД (a , b)					

# Задачи

---

**«С»:** Ввести с клавиатуры два натуральных числа и сравнить количество шагов цикла для вычисления их НОД с помощью обычного и модифицированного алгоритмов Евклида.

## Пример:

Введите два числа:

**1998 2**

НОД(1998, 2) = 2

Обычный алгоритм: 998

Модифицированный: 1

# Обработка потока данных

*Задача.* На вход программы поступает поток данных — последовательность целых чисел, которая **заканчивается нулём**. Требуется найти сумму элементов этой последовательности.

```
while x<>0 do begin
  { добавить x к сумме }
  { x := следующее число }
end;
```



Откуда возьмётся **x** в первый раз?



# Обработка потока данных

```
var x, sum: integer;  
...  
sum := 0;  
read(x); { ввести первое число }  
while x <> 0 do begin  
    sum := sum + x;  
    read(x); { ввести следующее }  
end;  
write('Сумма ', sum);
```



Как найти сумму положительных?

# Задачи

---

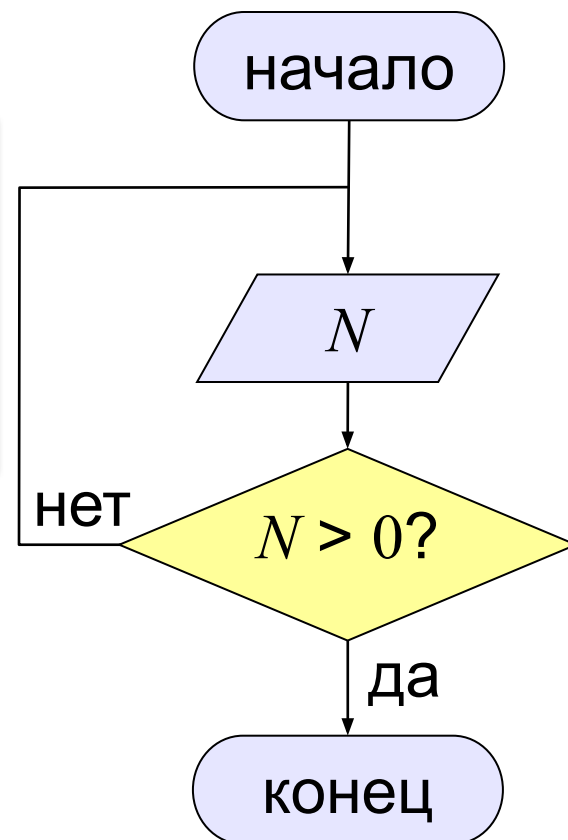
- «А»: На вход программы поступает неизвестное количество чисел целых, ввод заканчивается нулём. Определить, сколько получено чисел, которые делятся на 3.
- «В»: На вход программы поступает неизвестное количество чисел целых, ввод заканчивается нулём. Определить, сколько получено двузначных чисел, которые заканчиваются на 3.
- «С»: На вход программы поступает неизвестное количество чисел целых, ввод заканчивается нулём. Найти максимальное из введённых чётных чисел.

## Цикл с постусловием

- условие проверяется **после** завершения очередного шага цикла
- цикл всегда выполняется хотя бы один раз
- как только условие становится **ИСТИННЫМ**, работа цикла заканчивается

```
repeat  
  write ( 'Введите N>0: ' );  
  read (N) ;  
until N > 0;
```

условие **окончания**  
работы цикла



# Задачи

---

- «А»: Напишите программу, которая предлагает ввести пароль и не переходит к выполнению основной части, пока не введён правильный пароль. Основная часть – вывод на экран «секретных сведений».
- «В»: Напишите программу, которая получает с клавиатуры натуральное число, которое больше 1, и определяет, простое оно или нет. Для этого нужно делить число на все натуральные числа, начиная с 2, пока не получится деление без остатка.
- «С»: Напишите программу, которая получает с клавиатуры два целых числа и вычисляет их произведение, используя только операции сложения.

# Задачи

---

«D»: Напишите программу, которая получает с клавиатуры натуральное число и вычисляет целый квадратный корень из него – наибольшее число, квадрат которого не больше данного числа.

# Цикл по переменной

Задача. Вывести на экран степени числа 2 от  $2^1$  до  $2^{10}$ .

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



Работа с **k** в трёх местах!

Идея: собрать всё вместе.

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

увеличение на 1  
по умолчанию

## Цикл по переменной

---

Задача. Найти сумму чисел от 1 до 1000.

```
var sum, i: integer;  
...  
sum := 0;  
for i:=1 to 1000 do  
    sum := sum + i;
```

Задача. Вывести квадраты чисел от 10 до 1 по убыванию.

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

шаг «-1»

# Цикл по переменной

Задача. Найти сумму чётных чисел от 2 до 1000.

```
sum := 0;  
for i := 1 to 1000 do  
  if i mod 2 = 0 then  
    sum := sum + i;
```

?

Что плохо?

```
sum := 0;  
k := 2;  
for i := 1 to 500 do begin  
  sum := sum + k;  
  k := k + 2  
end;
```

вспомогательная  
переменная

2, 4, 6, ...

всего 500  
чисел



# В других языках программирования

## Python:

диапазон [1;1001)

```
Sum = 0
for i in range(1, 1001):
    Sum += i
```

## C:

```
int sum, i;
sum = 0;
for (i=1; i<=1000; i++)
    sum += i;
```

i=i+1;

sum=sum+i;

## Задачи

---

«А»: Ипполит задумал трёхзначное число, которое при делении на 15 даёт в остатке 11, а при делении на 11 даёт в остатке 9. Напишите программу, которая находит все такие числа.

«В»: С клавиатуры вводится натуральное число  $N$ . Программа должна найти факториал этого числа (обозначается как  $N!$ ) – произведение всех натуральных чисел от 1 до  $N$ . Например,

$$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120.$$

«С»: 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.

# Программирование (Паскаль)

## § 21. Массивы

# Что такое массив?

---



Как ввести 10000 переменных?

**Массив** – это группа переменных одного типа, расположенных в памяти рядом (в соседних ячейках) и имеющих общее имя.

**Надо:**

- выделять память
- записывать данные в нужную ячейку
- читать данные из ячейки

# Выделение памяти (объявление)

 Массив = таблица!

МИНИМАЛЬНЫЙ  
ИНДЕКС

МАКСИМАЛЬНЫЙ  
ИНДЕКС

```
var A: array[1..5] of integer;  
    V: array[0..5] of real;
```

**Индекс элемента** — это значение, которое указывает на конкретный элемент массива.

```
const N = 10;
```

```
var A: array[1..N] of integer;
```

размер через  
константу

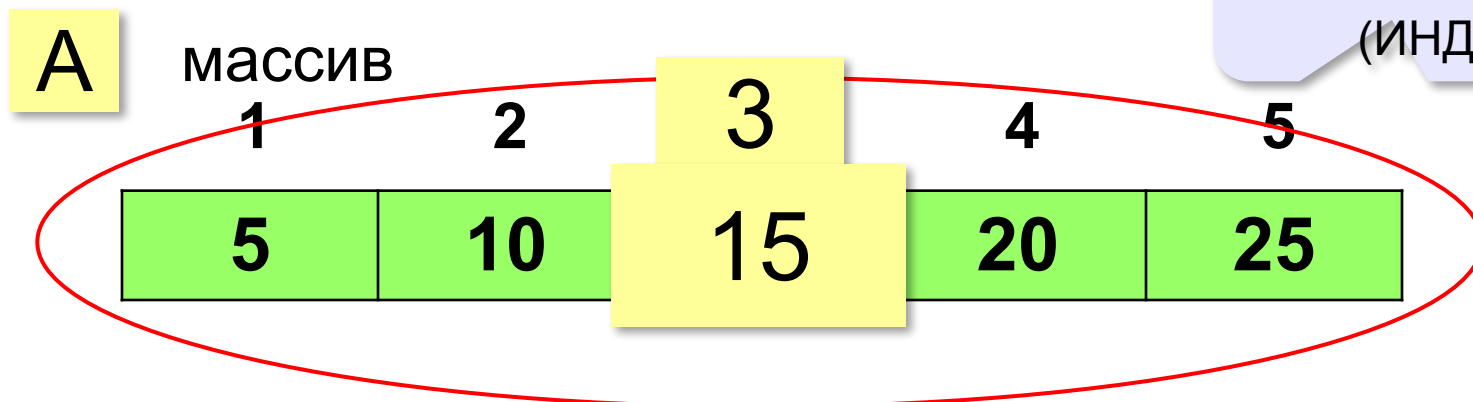
 Зачем?

## Что неправильно?

```
var A: array [1..1] of integer;  
...  
A[5] := 4.5;
```

```
var A: array [1..10] of integer;  
...  
A[15] := 'a';
```

# Обращение к элементу массива



НОМЕР  
элемента массива  
(ИНДЕКС)

А[1]    А[2]    А[3]    А[4]    А[5]

ЗНАЧЕНИЕ  
элемента массива

А[2]

НОМЕР (ИНДЕКС)  
элемента массива: 2

ЗНАЧЕНИЕ  
элемента массива: 10

# Обращение к элементу массива

1	2	3	4	5
23	12	7	43	51

```
var i: integer;  
i := 2;  
A[3] := A[i] + 2*A[i-1] + A[2*i];  
writeln( A[3]+A[5] );
```



Что получится?

```
A[3] := A[2] + 2*A[1] + A[4];  
writeln( A[3]+A[5] );
```

101

152



# Что неверно?

```
var A: array[1..5] of integer;
```

```
x: integer;
```

```
...
```

```
x := 2;
```

```
writeln( A[x-3] );
```

```
A[x+4] := A[x-1] + A[2*x];
```



Что плохо?

```
writeln( A[-1] );  
A[6] := A[1] + A[4];
```

**Выход за границы массива** — это обращение к элементу с индексом, который не существует в массиве.

## Перебор элементов массива

---

```
const N = 10;  
var A: array[1..N] of integer;
```

**Перебор элементов:** просматриваем все элементы массива и, если нужно, выполняем с каждым из них некоторую операцию.

```
for i:=1 to N do begin  
    { здесь работаем с A[i] }  
end;
```

# Заполнение массива

```
for i:=1 to N do  
  A[i] := i;
```



Что произойдёт?

В развёрнутом виде

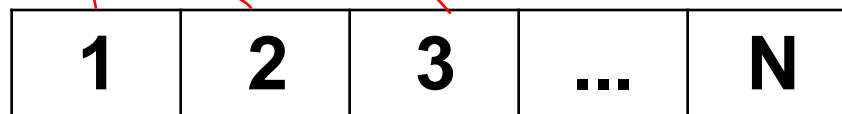
```
A[1] := 1;
```

```
A[2] := 2;
```

```
A[3] := 3;
```

...

```
A[N] := N;
```



# Заполнение массива в обратном порядке

N	...	3	2	1
---	-----	---	---	---

```
A[1] := N;  
A[2] := N-1;  
A[3] := N-2;  
...  
A[N] := 1;
```

```
X := N;  
for i:=1 to N do begin  
  A[i] := X;  
  X := X - 1  
end;
```

**?** Как меняется X?

$X = N, N-1, \dots, 2, 1$

начальное  
значение

уменьшение  
на 1

# Заполнение массива в обратном порядке

N	...	3	2	1
---	-----	---	---	---

$$A[i] := X;$$

**?** Как связаны  $i$  и  $X$ ?

$i$	$X$
1	N
2	N-1
3	N-2
...	...
N	1

+1      -1

```
for i:=1 to N do
  A[i] := N + 1 - i
end;
```

**!** Сумма  $i$  и  $X$  не меняется!

$$i + X = N + 1$$

$$X = N + 1 - i$$

# Вывод массива на экран

```
for i:=1 to N do  
  write( A[i], ' ' );
```



Что плохо?

интервал между значениями

или так:

```
for i:=1 to N do  
  writeln( A[i] );
```

в столбик

или так:

```
write( '[' );  
for i:=1 to N do  
  write( A[i], ', ' );  
writeln( ']' );
```



Как убрать?

[1,2,3,4,5,]

# Ввод с клавиатуры

```
for i:=1 to N do  
  read( A[i] );
```



Что плохо?

## С подсказкой для ввода:

```
for i:=1 to N do begin  
  write( 'A[' , i , ']= ' );  
  read( A[i] )  
end;
```

A[1] = 5

A[2] = 12

A[3] = 34

A[4] = 56

A[5] = 13

# В других языках программирования

## Python:

```
A = [0]*N
for i in range(N):
    A[i] = i + 1
print(A)
```



Нумерация элементов  
всегда с нуля!

## C++:

```
int A[N], i;
for (i = 0; i < N; i++)
    A[i] = i + 1;
for (i = 0; i < N; i++)
    cout << A[i] << " ";
```



# Задачи

---

- «А»: а) Заполните все элементы массива из 10 элементов значением  $X$ , введённым с клавиатуры.
- б) Заполните массив из 10 элементов последовательными натуральными числами, начиная с  $X$  (значение  $X$  введите с клавиатуры).
- «В»: а) Заполните массив из 10 элементов натуральными числами в порядке убывания. Значение  $X$  вводится с клавиатуры. Последний элемент должен быть равен  $X$ , предпоследний равен  $X+1$  и т.д.
- б) Заполните массив из 10 элементов степенями числа 2 (от  $2^1$  до  $2^N$ ), так чтобы первый элемент был равен 2, второй – 4, третий – 8 и т.д.

## Задачи

---

- «С»: а) Заполните массив из 10 элементов степенями числа 2, начиная с конца, так чтобы последний элемент массива был равен 1, а каждый предыдущий был в 2 раза больше следующего.
- б) С клавиатуры вводится целое число  $X$ . Заполните массив из 11 элементов целыми числами, так чтобы средний элемент массива был равен  $X$ , слева от него элементы стояли по возрастанию, а справа – по убыванию. Соседние элементы отличаются на единицу. Например, при  $X = 3$  массив из 5 элементов заполняется так: 1 2 3 2 1.

# Заполнение случайными числами

---

```
for i:=1 to N do begin  
    A[i]:=20+random(81);  
    write(A[i], ' ' )  
end;
```

сразу вывод на  
экран



Какой отрезок?

## Задачи-2

---

**«А»:** Напишите программу, которая заполняет массив из 10 элементов случайными числами в диапазоне  $[0,10]$ , выводит его на экран, а затем выводит на экран квадраты всех элементов массива.

**Пример:**

**Массив:** 5 6 2 3 1 4 8 7

**Квадраты:** 25 36 4 9 1 16 64 49

**«В»:** Напишите программу, которая заполняет массив из 10 элементов случайными числами в диапазоне  $[100,300]$  и выводит его на экран. После этого на экран выводятся средние цифры (число десятков) всех чисел, записанных в массив.

**Пример:**

**Массив:** 142 224 135 257 167 295 126 223 138 270

**Число десятков:** 4 2 3 5 6 9 2 2 3 7

## Задачи-2

---

**«С»:** Напишите программу, которая заполняет массив из 10 элементов случайными числами в диапазоне [100,500] и выводит его на экран. После этого на экран выводятся суммы цифр всех чисел, записанных в массив.

**Пример:**

**Массив:** 162 425 340 128 278 195 326 414 312 177

**Суммы цифр:** 9 11 7 11 17 15 11 9 6 15

# Программирование (Паскаль)

## § 22. Алгоритмы обработки массивов

# Сумма элементов массива

Задача. Найти сумму элементов массива.

```
const N = 10;  
var A: array[1..N] of integer;
```

?

Какие переменные  
нужны?

```
sum := 0;  
for i := 1 to N do  
    sum := sum + A[i];  
writeln( sum );
```


5	2	8	3	1
---	---	---	---	---

i	sum
	0
1	5
2	7
3	15
4	18
5	19

## Сумма не всех элементов массива

---

Задача. Найти сумму чётных элементов массива.

 Что делаем с нечётными?

```
sum := 0;  
for i := 1 to N do  
  if A[i] mod 2 = 0 then  
    sum := sum + A[i];  
writeln( sum );
```



## Задачи

---

- «А»: Напишите программу, которая заполняет массив из 10 элементов случайными числами на отрезке  $[-5; 5]$  и находит сумму положительных элементов.
- «В»: Напишите программу, которая заполняет массив из 10 элементов случайными числами на отрезке  $[-2; 2]$  и находит произведение ненулевых элементов.
- «С»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке  $[100; 1000]$  и находит отдельно сумму элементов в первой и во второй половинах массива.

# Подсчёт элементов по условию

Задача. Найти количество чётных элементов массива.

**?** Какие переменные нужны?

```
var count: integer;  
count := 0;  
for i := 1 to N do  
    if A[i] mod 2 = 0 then  
        count := count + 1;  
writeln( count );
```

переменная-  
счётчик


**?** Что тут делаем?

## Среднее арифметическое

---

Задача. Найти среднее арифметическое элементов массива, которые больше 180 (рост в см).

```
sum := 0;  
for i := 1 to N do  
    if A[i] > 180 then  
        sum := sum + A[i];  
writeln( sum / N );
```



Что плохо?

## Среднее арифметическое

*Задача.* Найти среднее арифметическое элементов массива, которые больше 180 (рост в см).



Какие переменные нужны?

```
sum := 0;  
count := 0;  
for i := 1 to N do  
  if A[i] > 180 then begin  
    count := count + 1;  
    sum := sum + A[i];  
  end;  
writeln( sum/count )
```



Что тут делаем?

# Задачи

---

- «А»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке  $[0; 200]$  и считает число элементов, которые делятся на 10.
- «В»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке  $[0; 200]$  и считает число двузначных чисел в массиве.
- «С»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке  $[10; 100]$  и считает число пар соседних элементов, сумма которых делится на 3.

# Обработка потока данных

**Задача.** С клавиатуры вводятся числа, ввод завершается числом 0. Определить, сколько было введено положительных чисел.

- 1) нужен счётчик
- 2) счётчик увеличивается
- 3) нужен цикл
- 4) это цикл с условием (число шагов неизвестно)

 ?

Когда увеличивать счётчик?

 ?

Какой цикл?

**счётчик = 0**

**пока не введён 0:**

**если введено число > 0 то**

**счётчик := счётчик + 1**

# Обработка потока данных

```
var x, count: integer;  
count := 0;  
read( x );  
while x <> 0 do begin  
  if x > 0 then  
    count := count + 1;  
  read( x );  
end;  
writeln( count );
```

откуда взять x?



Что плохо?

## Найди ошибку!

---

```
var x, count: integer;  
count := 0;  
read( x );  
while x <> 0 do begin  
    if x > 0 then  
        count := count + 1;  
end read( x );  
writeln( count );
```



# Найди ошибку!

```
var x, count: integer;  
count := 0;  
while x = 0 do begin  
  if x ><> then  
    count := count + 1;  
  read( x );  
end;  
writeln( count );
```

# Обработка потока данных

**Задача.** С клавиатуры вводятся числа, ввод завершается числом 0. Найти сумму введённых чисел, оканчивающихся на цифру "5".

- 1) нужна переменная для суммы
- 2) число добавляется к сумме, если оно заканчивается на "5"
- 3) нужен цикл с условием

```
сумма := 0
```

```
пока не введён 0:
```

```
если число оканчивается на "5" то
```

```
сумма := сумма + число
```



Как это записать?

```
if x mod 10 = 5 then
```

## Обработка потока данных

**Задача.** С клавиатуры вводятся числа, ввод завершается числом 0. Найти сумму введённых чисел, оканчивающихся на цифру "5".

```
var x, sum: integer;  
sum := 0;  
read( x );  
while x <> 0 do begin  
    if x mod 10 = 5 then  
        sum := sum + x;  
    read( x )  
end;  
writeln( sum );
```



Чего не хватает?

# Найди ошибку!

---

```
var x, sum: integer;  
sum := 0;  
read( x ); 0 do begin  
    if x mod 10 = 5 then  
        sum := sum + x;  
    read( x )  
end;  
writeln( sum );
```

# Задачи

---

- «А»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Определить, сколько получено чисел, которые делятся на 3.
- «В»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Определить, сколько получено двузначных чисел, которые заканчиваются на 3.

# Задачи

---

- «С»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Найти среднее арифметическое всех двузначных чисел, которые делятся на 7.
- «D»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Найти максимальное из введённых чётных чисел.

# Перестановка элементов массива

---



Как поменять местами значения двух переменных  $a$  и  $b$ ?

вспомогательная  
переменная

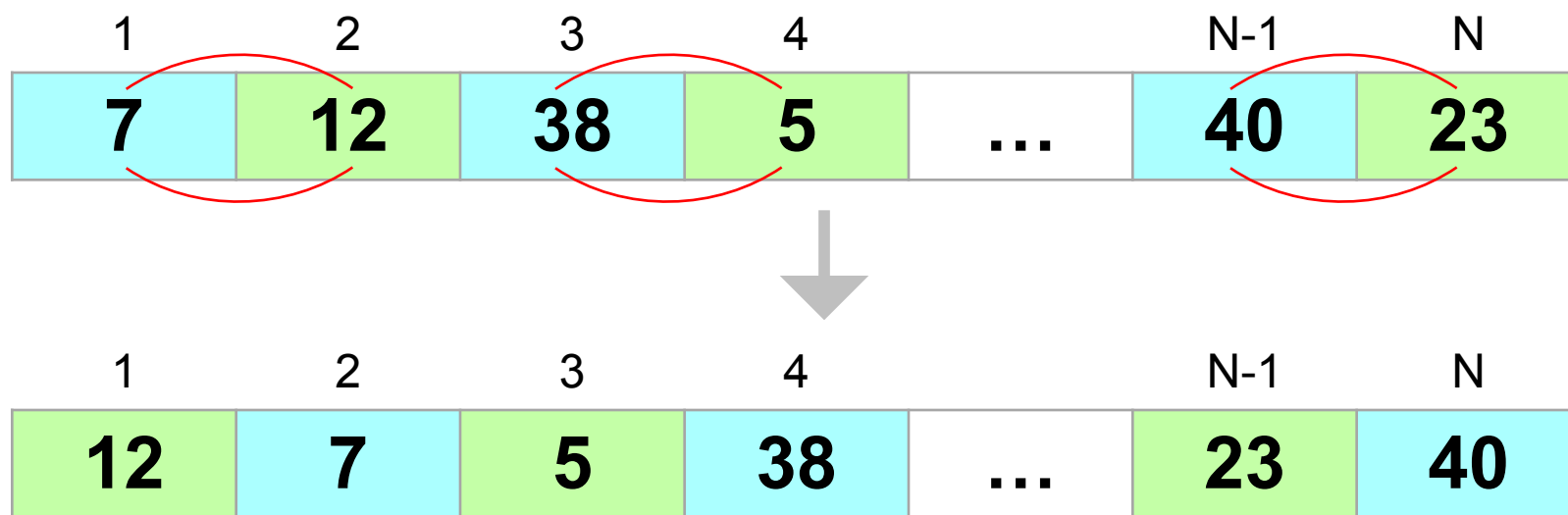
```
c := a;  
a := b;  
b := c;
```

элементы массива:

```
c := A[i];  
A[i] := A[k];  
A[k] := c;
```

## Перестановка пар соседних элементов

**Задача.** Массив  $A$  содержит чётное количество элементов  $N$ . Нужно поменять местами пары соседних элементов: первый со вторым, третий — с четвёртым и т. д.





# Перестановка пар соседних элементов

```
for i:=1 to N do begin
```

```
    поменять местами A[i] и A[i+1]
```

```
end;
```

?

Что плохо?

1	2	3	4	5	6
7	12	38	5	40	23
12	7	38	5	40	23
12	38	7	5	40	
12	38	5	7	40	23
12	38	5	40	7	23
12	38	5	40	23	7

выход за границы массива

?

# Перестановка пар соседних элементов

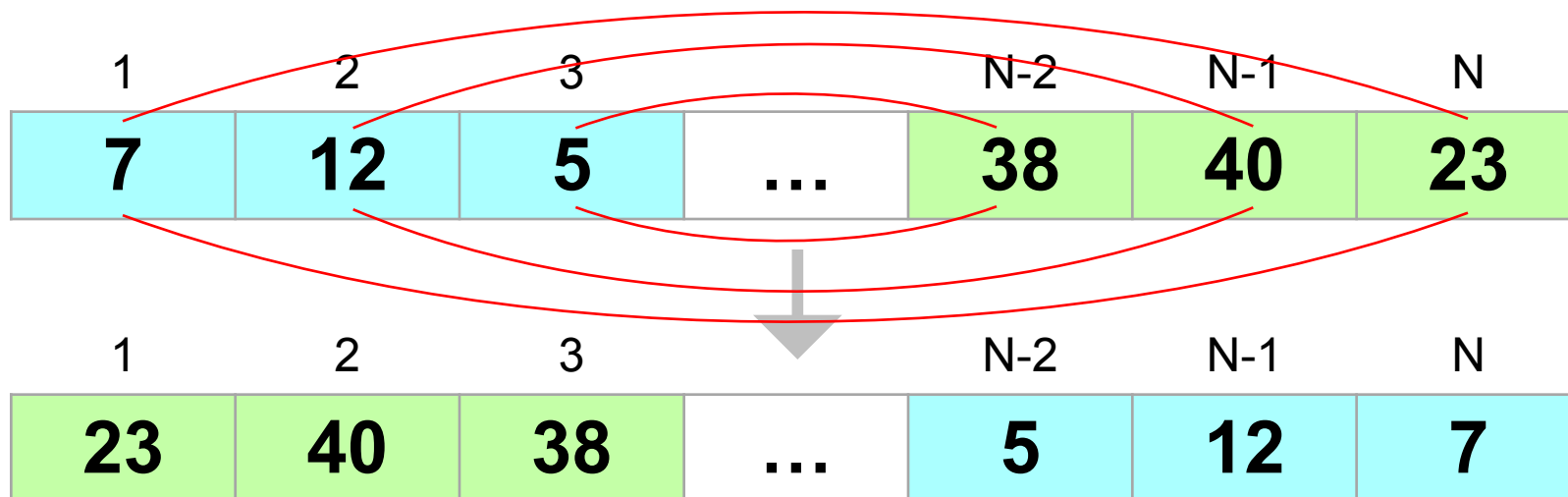
не выходим за  
границу

```
i := 1;
while i < N do begin
  { переставляем A[i] и A[i+1] }
  c := A[i];
  A[i] := A[i+1];
  A[i+1] := c;
  i := i + 2      { к следующей паре }
end;
```

$A[1] \leftrightarrow A[2], A[3] \leftrightarrow A[4], \dots, A[N-1] \leftrightarrow A[N]$

# Реверс массива

Задача. Переставить элементы массива в обратном порядке (выполнить *реверс*).



$$A[1] \leftrightarrow A[N]$$

$$1+N = N+1$$

$$A[2] \leftrightarrow A[N-1]$$

$$2+N-1 = N+1$$

$$A[i] \leftrightarrow A[N+1-i]$$

$$i+??? = N+1$$

$$A[N] \leftrightarrow A[1]$$

$$N+1 = N+1$$

# Реверс массива

```
for i:=1 to N div 2 do begin
  поменять местами A[i] и A[N+1-i]
end;
```

?

Что плохо?

1	2	3	4
7	12	40	23
23	12	40	7
23	40	12	7
23	12	40	7
7	12	40	23

i=1

i=2

i=3

i=4

?

Как исправить?

# 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. иллюстрации художников издательства «Бином»
2. авторские материалы