

Введение в Delphi

Структура программы

program <имя программы>;

uses ... ; {модули}

label... ; {метки}

type ... ; {польз. типы данных}

const ...; {константы}

var ...; {переменные}

комментарии в фигурных
скобках не
обрабатываются

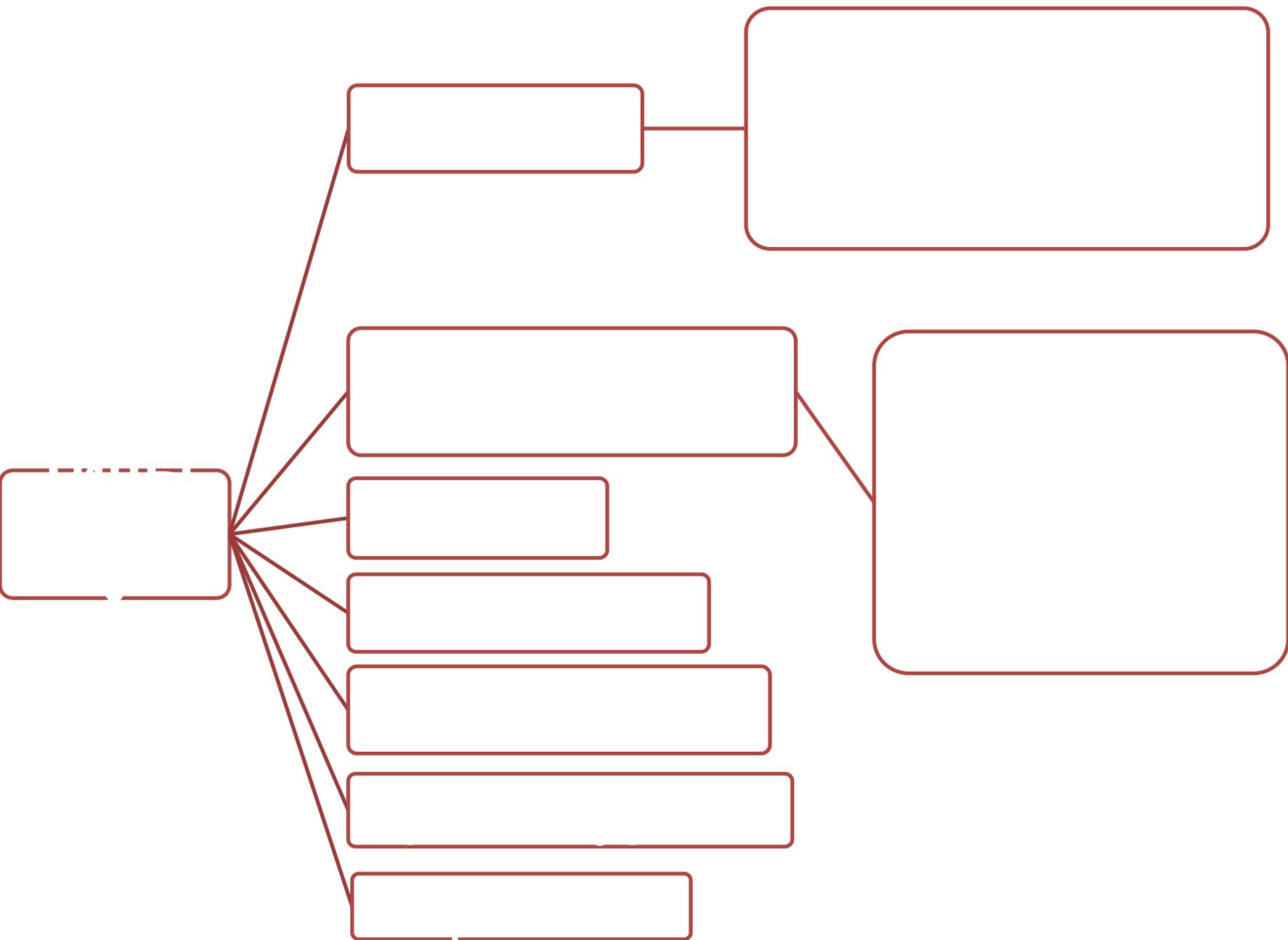
procedure { процедуры }

function { функции }

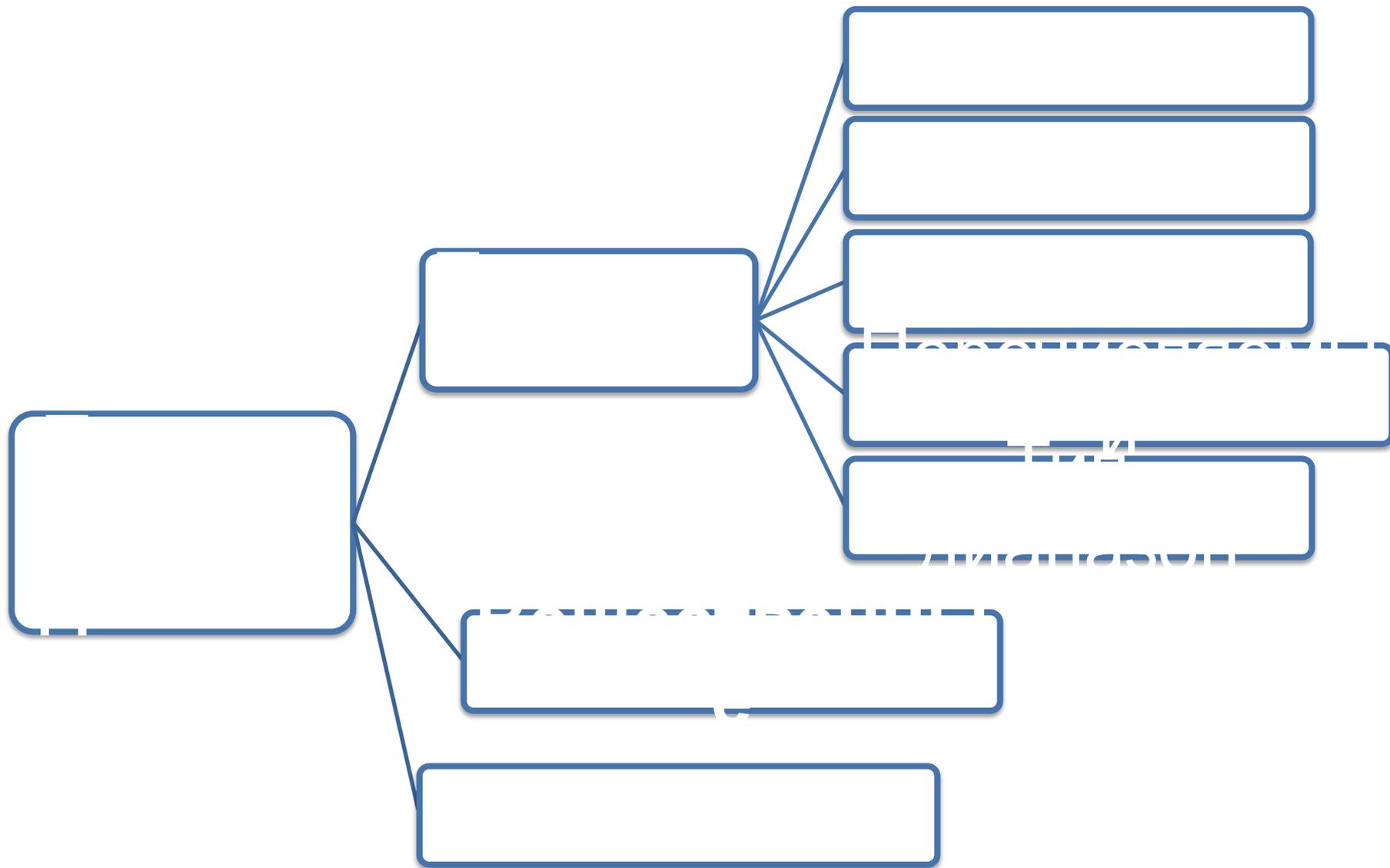
begin

... {основная программа}

end.



Простые типы данных



Порядковые типы данных

- **Порядковые типы отличаются тем, что каждый из них имеет конечное число возможных значений.**
- **Эти значения можно определенным образом упорядочить**
- **С каждым элементом порядкового типа можно сопоставить некоторое число – порядковый номер значения**

Целые типы

Название	Длина, байт	Диапазон значений
Byte	1	0...255
ShortInt	1	-128...+127
SmallInt	2	-32 768 ...+32 767
Word	2	0 ... 65 535
Integer	4	-2 147 483 648 ... +2 147 483 647
LongInt	4	-2 147 483 648 ... +2 147 483 647
LongWord	4	0 ... 4 294 967 295
Int64	8	$-9*10^{18}$... $+9*10^{18}$
Cardinal	4	0 ... +2 147 483 647

Логические типы

- В стандартном Паскале определен тип `Boolean`
- В Delphi для совместимости с Windows добавлены типы: `ByteBool`, `Bool`, `WordBool` и `LongBool`
- Значения логического типа: `True` и `False`

Символьный тип - CHAR

- Для кодировки в Windows используется код ANSI (American National Standard Institute)
- Первая половина символов с кодами 0..127 используется для обозначения служебных кодов и символов латинского алфавита
- Вторая половина (коды 128 – 255) определяется используемым шрифтом

Кодировка символов в соответствии с стандартом ANSI

Код	Символ	Код	Символ	Код	Символ	Код	Символ	Код	Символ
0-31	Служеб. коды	48	0	68	D	88	X	108	l
		49	1	69	E	89	Y	109	m
		50	2	70	F	90	Z	110	n
		51	3	71	G	91	[111	o
32	BL	52	4	72	H	92	\	112	p
33	!	53	5	73	I	93]	113	q
34	“	54	6	74	J	94	^	114	r
35	#	55	7	75	K	95	_	115	s
36	\$	56	8	76	L	96	`	116	t
37	%	57	9	77	M	97	a	117	u
38	&	58	:	78	N	98	b	118	v
39	‘	59	;	79	O	99	c	119	w
40	(60	<	80	P	100	d	120	x
41)	61	=	81	Q	101	e	121	y
42	*	62	>	82	R	102	f	122	z
43	+	63	?	83	S	103	g	123	{
44	,	64	@	84	T	104	h	124	
45	-	65	A	85	U	105	i	125	}
46	.	66	B	86	V	106	j	126	~
47	/	67	C	87	W	107	k	127	

Символьный тип - CHAR

- К типу `Char` применимы операции отношения, а также встроенные функции:
- `Chr (B : Byte)` – функция типа *Char*; преобразует выражение *B* типа *Byte* в символ и возвращает его своим значением.

`Chr (65)` => `'A'`

`Chr (77)` => `'M'`

`Chr (63)` => `'?'`

СИМВОЛЬНЫЙ ТИП - CHAR

- `UpCase (CH)` – функция типа *Char*; возвращает прописную букву, если *CH* – строчная латинская буква, в противном случае возвращает сам символ *CH* (ИСХОДНЫЙ).

`UpCase ('a') => 'A'`

`UpCase ('M') => 'M'`

Перечисляемый тип

- Перечисляемый тип задается перечислением тех значений, которые он может получать.
- Каждое значение именуется некоторым идентификатором и располагается в списке, обрамленном круглыми скобками,

например:

```
type
```

```
    colors = (red, white, blue);
```

Перечисляемый тип

- **Максимальная мощность перечисляемого типа составляет 65 535 значений**
- **Значения в списке нумеруются, начиная с 0.**
- **Переменные любого перечисляемого типа можно сравнивать или присваивать. По значениям указанных переменных можно организовывать цикл.**

Перечисляемый тип

Ограничения

- **одна и та же константа не может быть употреблена в объявлении разных типов;**
- **недопустимо сравнение констант разных перечисляемых типов;**
- **перечисляемый тип нельзя прямо выводить на печать или вводить с клавиатуры.**

Тип-диапазон

- Тип-диапазон есть подмножество своего базового типа, в качестве которого может выступать любой порядковый тип, кроме типа-диапазона.
- Тип-диапазон задается границами своих значений внутри базового типа:

<мин . знач . > . . <макс . знач . >

Тип-диапазон

type

digit = '0' .. '9' ;

dig2 = 48 .. 57 ;

days =

(mo , tu , we , th , fr , sa , su) ;

WeekEnd = sa .. su ;

Тип-диапазон

- В стандартную библиотеку *Delphi* включены две функции, поддерживающие работу с типами-диапазонами:
- **High (X)** – возвращает максимальное значение типа-диапазона, к которому принадлежит переменная (или тип) X;
- **Low (X)** – возвращает

Тип-диапазон

type

dig2 = 48 .. 57;

...

...

High(dig2) => 57

Low(dig2) => 48

Порядковые типы данных

- К порядковым типам применимы функции:

Функция $\text{Ord}(X)$ возвращает порядковый номер значения X в заданном порядковом типе.

- для целых типов $\text{Ord}(X) = X$
- для логических X : $\text{Ord}(X) = 0$ или 1
- для символьного x : $\text{Ord}(x) \in [0..255]$
- для перечисляемого x : $\text{Ord}(x) \in [0..65535]$

Порядковые типы данных

- К порядковым типам применимы функции:

~~Пред~~(X) возвращает предыдущее значение порядкового типа. Не определено для нижней границы отрезка.

Succ(X)

Возвращает следующее значение порядкового типа. Не определено для верхней границы отрезка.

Порядковые типы данных

Ord. Примеры

- **Целые**

`Ord(10) = 10`

`Ord(-7) = -7`

- **Логические**

`Ord(True) = 1`

`Ord(False) = 0`

- **Символьные**

`Ord('A') = 65`

`Ord('B') = 66`

- **Перечисляемые**

`Ord(sa) = 5`

`Ord(mo) = 0`

Порядковые типы данных

Pred. Примеры

- Целые

`Pred(10) = 9`

`Pred(-7) = -8`

- Символьные

`Pred('B') = 'A'`

`Pred('Z') = 'Y'`

- Логические

`Pred(True) = False`

`Pred(False) = не определено!`

- Перечисляемые

`Pred(sa) = su`

`Pred(mo) = не определено`

Порядковые типы данных

Succ. Примеры

- Целые

`Succ (10) = 11`

`Succ (-7) = -6`

- Логические

`Succ (True) = не определено!`

`Succ (False) = True`

- Перечисляемые

`Succ (mo) = tu`

`Succ (su) = не определено`

- Символьные

`Succ ('B') = 'C'`

`Succ ('Z') = 'a'`

Вещественные типы данных

Название	Количество значащих цифр	Диапазон значений	Дли- на
Real	15...16	$5.0 \cdot 10^{-324} \dots 1.7 \cdot 10^{308}$	8
Real48	11...12	$2.9 \cdot 10^{-39} \dots 1.7 \cdot 10^{38}$	6
Single	7...8	$1.5 \cdot 10^{-45} \dots 3.4 \cdot 10^{38}$	4
Double	15...16	$5.0 \cdot 10^{-324} \dots 1.7 \cdot 10^{308}$	8
Extended	19...20	$3.4 \cdot 10^{-4951} \dots 1.1 \cdot 10^{4932}$	10
Comp	19...20	$-2^{63} \dots +2^{63} - 1$	8
Currency	19...20	$\pm 922\,337\,203\,685\,477.5807$	8

Константы

- **КОНСТАНТЫ:**

- **целые,**

- **вещественные,**

- **логические,**

- **символы,**

- **строки символов,**

- **конструкторы множеств,**

- **nil**

Константы

- **Целые:** [-2147843648 ... + 2147843647]
 - описываются по обычным правилам

- **Вещественные:**

- Описываются с использованием десятичной точки и/или экспоненциальной части

Например: 2.589, 3.14E5 = $3.14 * 10^5$

- **Шестнадцатеричные:** [\$00000000 ... \$FFFFFFFF]

- описываются с использованием знака \$

Константы

- **Символьные:** любой символ ПК
 - Записываются в апострофах или путем указания внутреннего кода (с помощью символа #)
 - `'z'`, `'Ф'`,
 - #97 – символ «а»
 - #90 – символ «Z»
 - `''''` – символ «'»
- **Строковые:** любая последовательность символов (кроме символа CR – возврат каретки), заключенная в апострофы

Константы

- **Конструктор множества – список элементов множества, обрамленных квадратными скобками**

```
[1, 2, 4..7, 12]
```

```
[blue, red]
```

```
[]
```

```
[true]
```

Константы

- **простые константы:**

```
<ИМЯ КОНСТАНТЫ> = <значение>;
```

Const

```
A = 5;
```

```
X = 5.5;
```

```
ch = 'c';
```

```
lg = true;
```

```
c1 = #97; // СИМВОЛ 'a'
```

Константы

- **Константные выражения:**

**<ИМЯ константы> =
<вычисляемое выражение>;**

Const

A = 5 div 2;

X = Pred('B');

Y = 3.14*Sqrt(5);

Z = Cos(y)+A;

Константы

- Константы-переменные:

```
<ИМЯ КОНСТАНТЫ>:<ТИП  
КОНСТАНТЫ> = <значение>;
```

Const

```
A:char = 'd' ;
```

```
K:real = 5.678 ;
```

```
m:byte = 31 ;
```

Операции

Тип операции	Операции	Приоритет
	()	
Унарные	not, @	
Мультипликативные	*, /, div, mod, and, shl, shr	
Аддитивные	+, -, or, xor	
Отношения	=, <>, <, >, <=, >=, in	

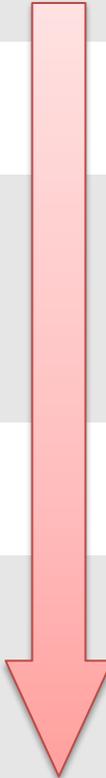


Таблица истинности

X	Y	not X	X and Y	X or Y	X xor Y
False	False	True	False	False	False
False	True	True	False	True	True
True	False	False	False	True	True
True	True	False	True	True	False

Математические процедуры и функции

- Большинство арифметических и математических стандартных процедур и функций описано в модуле `Math`.
- Поэтому перед их использованием необходимо подключить данный модуль в разделе `USES`

```
Uses Math;
```

Abs	Определяет абсолютное значение аргумента
ArcCos	Определяет значение арккосинуса X
ArcSin	Определяет значение арксинуса X
ArcTan	Определяет значение арктангенса X
Ceil	Округляет значение в положительную сторону
Cos	Определяет косинус X
Cotan	Определяет котангенс X
Dec	Уменьшает значение переменной на заданное число единиц (по умолчанию на 1)

DegToRad	Преобразовывает угол из градусов в радианы
Exp	Вычисляет экспоненциальное значение передаваемого аргумента
Floor	Округляет переменную в сторону ближайшего меньшего целого числа
Frac	Возвращает дробную часть аргумента
High	Возвращает наибольшее значение в области значений аргумента
Inc	Увеличивает значение переменной на заданное число единиц (по умолчанию на 1)
Int	Возвращает целую часть аргумента (в вещественном формате)

Ln	Определяет натуральный логарифм передаваемого аргумента
Log10	Определяет десятичный логарифм
Log2	Определяет двоичный логарифм
LogN	Определяет логарифм с основанием N
Low	Возвращает наименьшее значение в области значений аргумента
Odd	Проверяет, является ли аргумент нечетным числом
Ord	Возвращает порядковый номер значения
Pi	Возвращает значение π
Power	Возводит число в степень

Pred	Возвращает предыдущее значение порядковой переменной
RadToDeg	Преобразовывает угол, указанный в радианах, в угол в градусах
Round	Округляет значение вещественной переменной до целочисленного
Sin	Возвращает синус передаваемого аргумента
Sqr	Возводит число в квадрат
Sqrt	Вычисляет корень квадратный
Succ	Возвращает последующее значение порядковой переменной
Tan	Определяет тангенс угла
Trunc	Отсекает дробную часть вещественного числа

Самостоятельно:

1. Определить константы всех видов.
2. Определить приоритеты выполнения операций в выражении
 $x \bmod 5 = 0$ and $y \operatorname{div} 3 = y / 3$ or $z > 0$;
3. Записать выражения:

$$S = \frac{3 \cos 5x^2 - \ln 5}{e^4 - 6x} + 6 \sin^2 x^6$$

$$S = \sqrt{8 \operatorname{tg}(5x) - 5} \sqrt{34} \left[+ \frac{6x \log_2 5x}{\pi^4 - \sin 6x} \right]$$

Переменные

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

Объявление переменных (выделение памяти):

```
var a, b: integer;  
    Q: real;  
    s1, s2: boolean;  
    x: integer;
```

ОПЕРАТОРЫ ЯЗЫКА

Оператор присваивания

Арифметическое выражение может включать

- КОНСТАНТЫ
- имена переменных
- знаки арифметических операций:

+ - * /

умножение

деление

div

mod

деление
нацело

остаток от
деления

- ВЫЗОВЫ ФУНКЦИЙ
- круглые скобки ()

Как изменить значение переменной?

Оператор – это команда языка программирования высокого уровня.

Оператор присваивания служит для изменения значения переменной.

Пример:

```
program Prog_1;
```

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

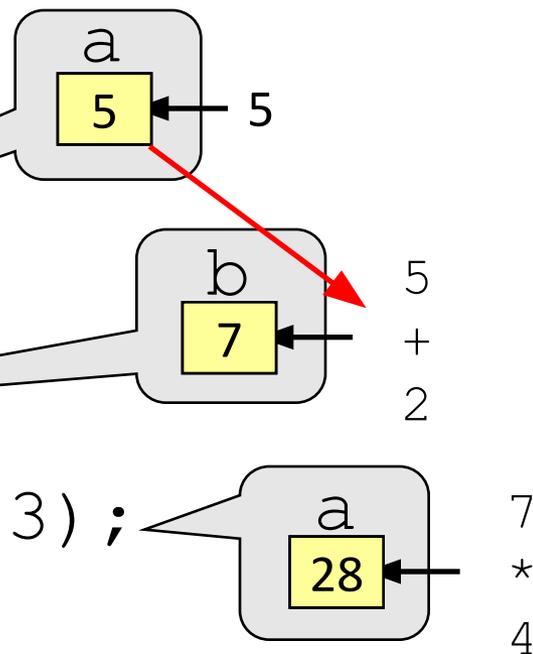
```
begin
```

```
  a := 5;
```

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

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

```
end.
```



Какие операторы неправильные?

```
program Prog_2;  
var a, b: integer;  
    x, y: real;  
begin  
    a := 5;  
    10 := x;  
    y := 7,8;  
    b := 2.5;  
    x := 2*(a + y);  
    a := b + x;  
end.
```

имя переменной должно
быть слева от знака :=

целая и дробная часть
отделяются точкой

нельзя записывать
вещественное значение в
целую переменную

Ручная прокрутка программы

```
program Prog_2;  
var a, b: integer;  
begin  
  a := 5;  
  b := a + 2;  
  a := (a + 2) * (b - 3);  
  b := a div 5;  
  a := a mod b;  
  a := a + 1;  
  b := (a + 14) mod 7;  
end.
```

a	b
?	?
5	
	7
28	
	5
3	
4	
	4

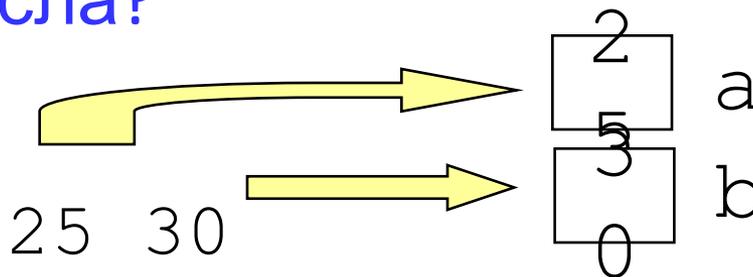
Оператор ввода

`read (a);` { ввод значения
переменной `a` }

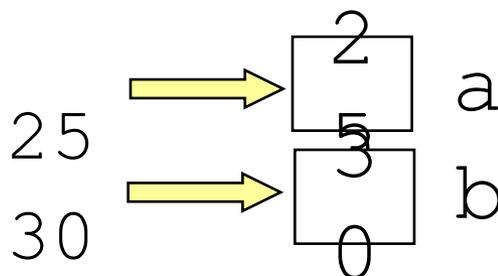
`read (a, b);` { ввод значений
переменных `a` и `b` }

Как вводить два числа?

через пробел:



через *Enter*:



Форматы вывода

```
program qq;  
var i: integer;  
    x: real;  
begin  
    i := 15;  
    writeln ( '>', i, ' <' );  
    writeln ( '>', i:5, '<' );  
    x := 12.345678;  
    writeln ( '>', x, '<' );  
    writeln ( '>', x:10, '<' );  
    writeln ( '>', x:7:2, '<' );  
end.
```

ВСЕГО
СИМВОЛОВ

ВСЕГО
СИМВОЛОВ

В дробной
части

```
>15<  
>  15<  
  
>1.234568E+001<  
> 1.23E+001<  
>  12.35<
```

Составной оператор:

- Это последовательность произвольных операторов программы, заключенных в операторные скобки: `begin ... end`.
- Как правило, составной оператор используется внутри других операторов языка Delphi
- Позволяет интерпретировать группу операторов как один оператор.
- Допускается любая степень вложенности составных операторов.

```
begin  
    .....  
    begin  
        .....  
        .....  
    end;  
end;
```

РАЗВЕТВЛЯЮЩИЕ АЛГОРИТМЫ

Разветвляющиеся алгоритмы

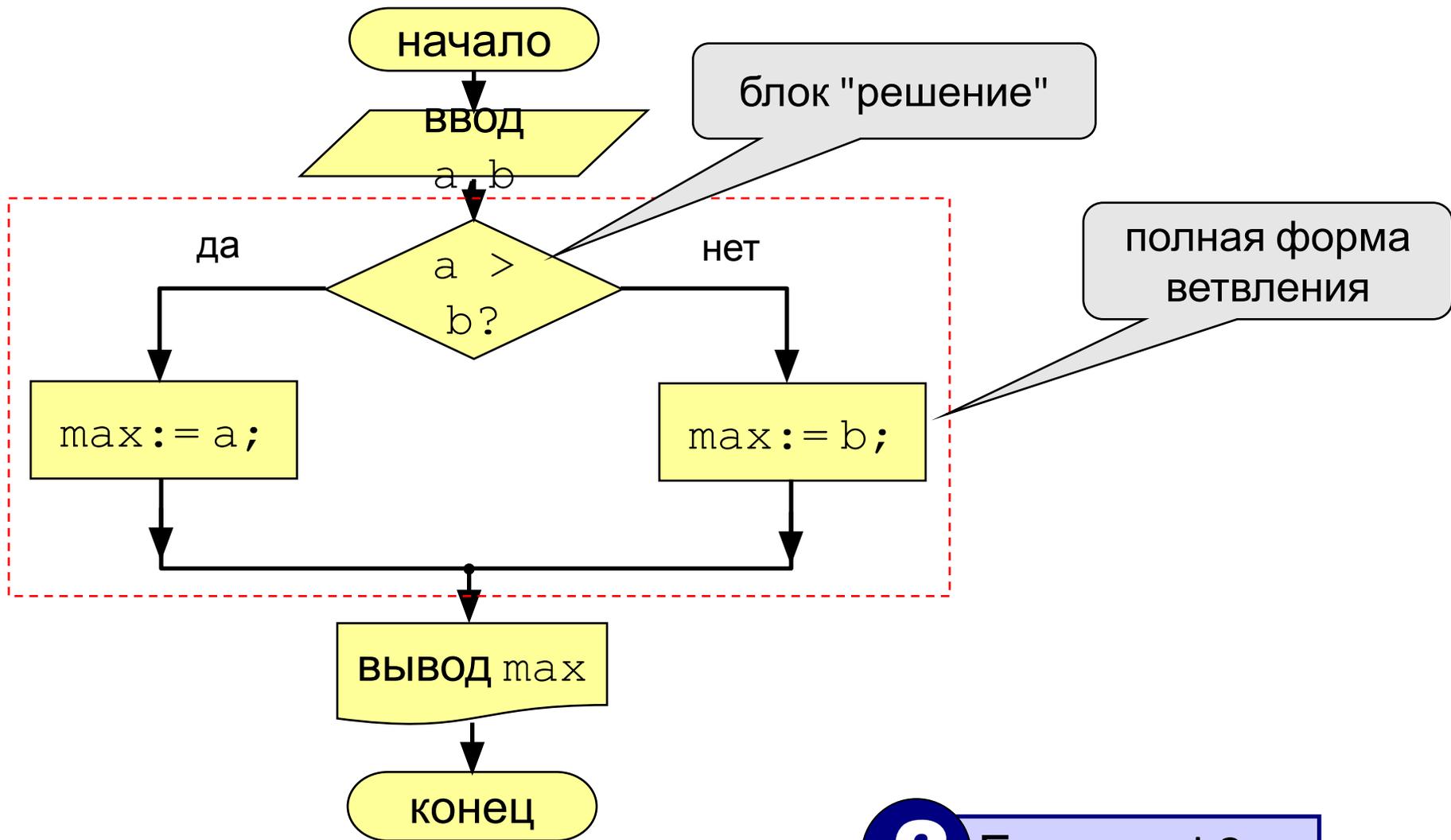
Задача. Ввести два целых числа и вывести на экран наибольшее из них.

Идея решения: надо вывести на экран первое число, если оно больше второго, или второе, если оно больше первого.

Особенность: действия исполнителя зависят от некоторых условий (*если ... иначе ...*).

Алгоритмы, в которых последовательность шагов зависит от выполнения некоторых условий, называются **разветвляющимися**.

Блок-схема



❓ Если $a = b$?

Программа

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



полная форма
условного
оператора

Условный оператор

```
if <условие> then begin
    {что делать, если условие верно}
end
else begin
    {что делать, если условие неверно}
end;
```

Особенности:

- перед *else* **НЕ** ставится точка с запятой
- часть *else ...* может отсутствовать (неполная форма)
- если в блоке один оператор, можно убрать слова *begin* и *end*

Что неправильно?

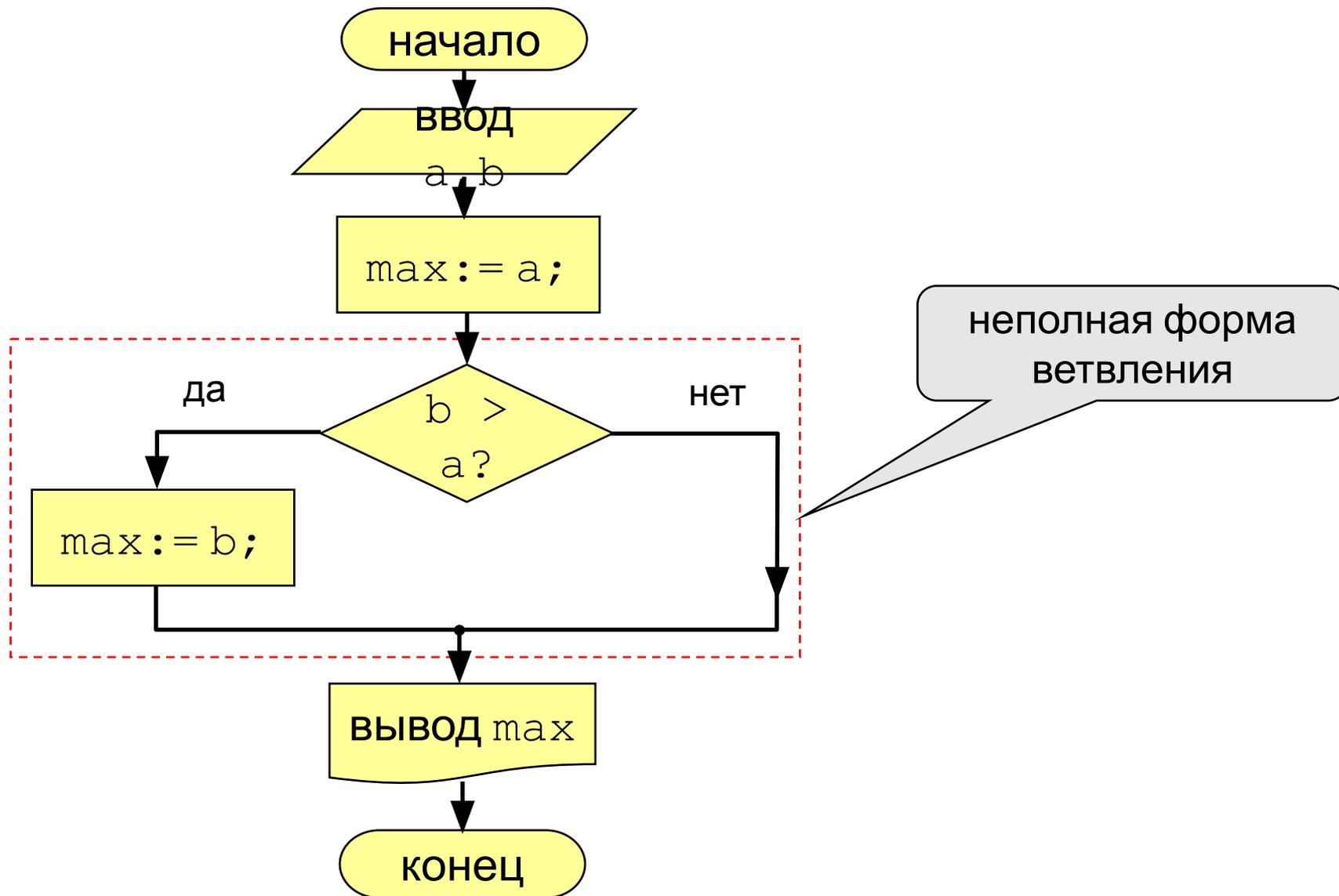
```
if a > b then begin
  a := b
end
else begi
  b :=n a
end;
```

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

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

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

Вариант 2. Блок-схема



Вариант 2. Программа

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

неполная форма
условного
оператора

Вариант 2Б. Программа

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

Что неправильно?

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

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

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

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

Самостоятельно:

«1»: Ввести три числа и найти наибольшее из них.

Пример:

Введите три числа:

4 15 9

Наибольшее число 15

«2»: Ввести пять чисел и найти наибольшее из них.

Пример:

Введите пять чисел:

4 15 9 56 4

Наибольшее число 56

Сложные условия

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

- `not` – НЕ (отрицание, инверсия)
- `and` – И (логическое умножение, конъюнкция, одновременное выполнение условий)
- `or` – ИЛИ (логическое сложение, дизъюнкция, выполнение хотя бы одного из условий)
- `xor` – исключающее ИЛИ (выполнение только одного из двух условий, но не обоих)

Простые условия (отношения)

<

<=

>

>=

равно

=

не равно

<>

Сложные условия

Порядок выполнения

- выражения в скобках
- not
- and
- or, xor
- <, <=, >, >=, =, <>

Особенность – каждое из простых условий обязательно заключать в скобки.

Пример

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

Сложные условия

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

`not (a > b)`

True

`(a < b) and (b < c)`

True

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

True

`(a < c) or (b < c) and (b < a)`

True

`(a < b) xor not (b > c)`

FALSE

Для каких значений x истинны условия:

`(x < 6) and (x < 10)`

`(x < 6) and (x > 10)`

`(x > 6) and (x < 10)`

`(x > 6) and (x > 10)`

`(x < 6) or (x < 10)`

`(x < 6) or (x > 10)`

`(x > 6) or (x < 10)`

`(x > 6) or (x > 10)`

$(-\infty, 6)$	$x < 6$
\emptyset	
$(6, 10)$	
$(10, \infty)$	$x > 10$
$(-\infty, 10)$	$x < 10$
$(-\infty, 6) \cup (10, \infty)$	
$(-\infty, \infty)$	
$(6, \infty)$	$x > 6$

Задания

«1»: Ввести номер месяца и вывести название времени года.

Пример:

Введите номер месяца:

4

весна

«2»: Ввести возраст человека (от 1 до 150 лет) и вывести его вместе с последующим словом "год", "года" или "лет".

Пример:

Введите возраст:

24

Вам 24 года

Введите возраст:

57

Вам 57 лет

ОПЕРАТОР ВЫБОРА

Оператор выбора

Задача: Ввести номер месяца и вывести количество дней в этом месяце.

Решение: Число дней по месяцам:

28 дней – 2 (февраль)

30 дней – 4 (апрель), 6 (июнь), 9 (сентябрь), 11 (ноябрь)

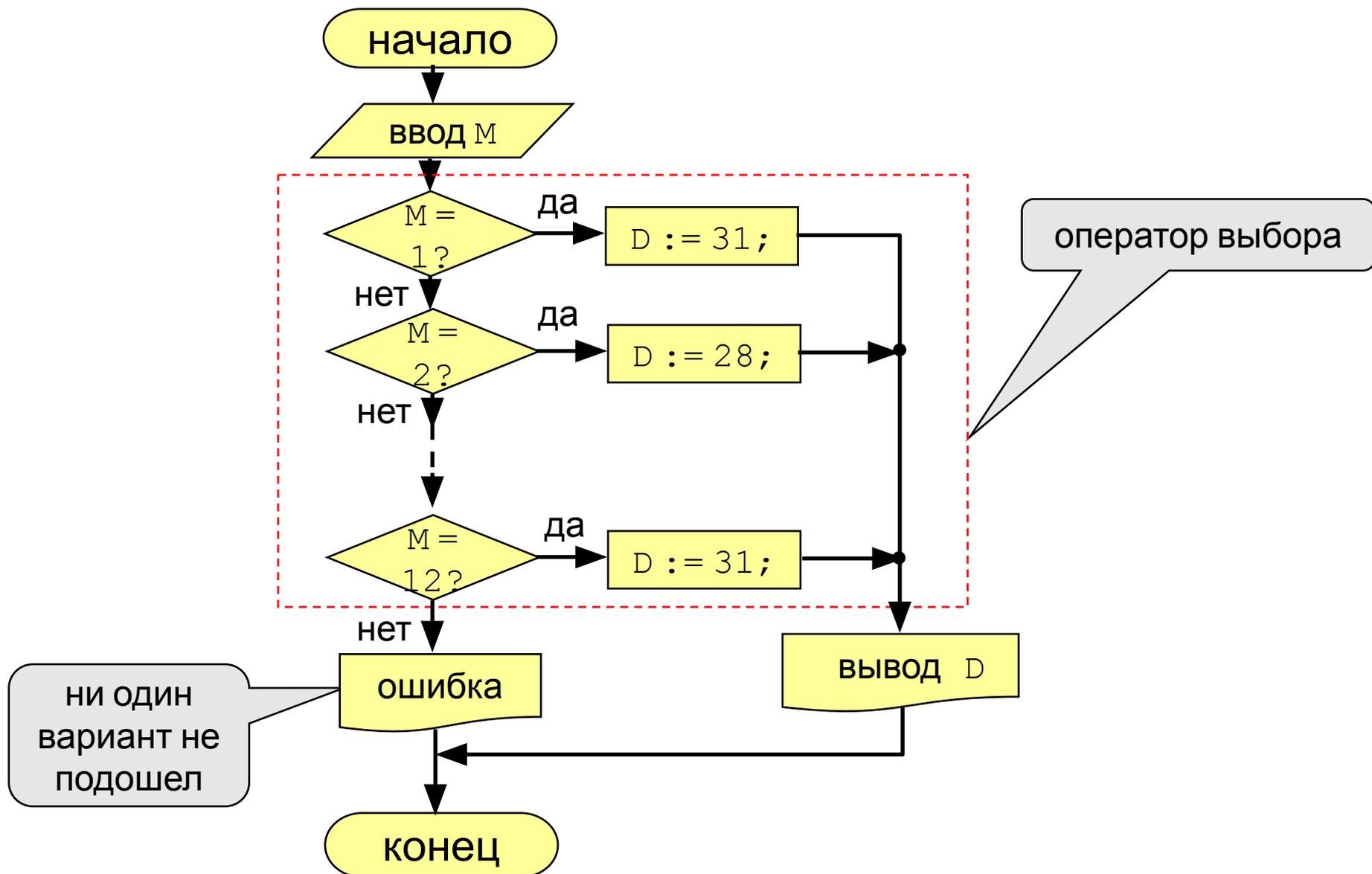
31 день – 1 (январь), 3 (март), 5 (май), 7 (июль),
8 (август), 10 (октябрь), 12 (декабрь)

Особенность: Выбор не из двух, а из нескольких вариантов в зависимости от номера месяца.



Можно ли решить известными методами?

Алгоритм



Оператор выбора - Case

- Оператор выбора позволяет выбрать одно из нескольких возможных продолжений программы.
- Параметром, по которому осуществляется выбор, служит *ключ выбора* – выражение любого порядкового типа

Оператор выбора - Case

```
case <ключ_выбора> of  
    <константа_выбора_1> : <оператор_1>;  
    <константа_выбора_2> : <оператор_2>;  
    <константа_выбора_3> : <оператор_3>;  
    .....  
    <константа_выбора_N> : <оператор_N>;  
    [else <оператор_N+1>]  
end;
```

Программа

```
program qq;  
var M, D: integer;  
begin  
  writeln('Введите номер месяца:');  
  read ( M );  
  case M of  
    2:          begin D := 28; end;  
    4, 6, 9, 11: begin D := 30; end;  
    1, 3, 5, 7, 8, 10, 12: D := 31;  
    else          D := -1;  
  end;  
  if D > 0 then  
    writeln('В этом месяце ', D, ' дней.')  else  
    writeln('Неверный номер месяца');  
end.
```

НИ ОДИН
вариант не
подошел

Оператор выбора

Особенности:

- после `case` может быть имя переменной или арифметическое выражение целого типа (`integer`)

```
case i+3 of
  1: begin a := b; end;
  2: begin a := c; end;
end;
```

ИЛИ СИМВОЛЬНОГО ТИПА (`char`)

```
var c: char;
...
case c of
  'a': writeln('Антилопа');
  'б': writeln('Барсук');
  else writeln('Не знаю');
end;
```

Оператор выбора

Особенности:

- если нужно выполнить только один оператор, слова `begin` и `end` можно не писать

```
case i+3 of
  1: a := b;
  2: a := c;
end;
```

- нельзя ставить два одинаковых значения

```
case i+3 of
  1: a := b;
  1: a := c;
end;
```

Оператор выбора

Особенности:

- значения, при которых выполняются одинаковые действия, можно группировать

перечисление

диапазон

смесь

```
case i of
  1:           a := b;
  2, 4, 6:    a := c;
  10..15:     a := d;
  20, 21, 25..30: a := e;
  else writeln('Ошибка');
end;
```

Что неправильно?

```
case a of
  2: begin a := b;
  4: a := c;
end;
```

```
case a of
  2: a := b ;
  4: a := c
end;
```

```
case a of
  2..5: a := b;
  4: a := c;
end;
```

```
case a of
  0..2: a := b;
  3..6: a := c;
end;
```

```
case a+c/2 of
  2: a := b;
  4: a := c;
end;
```

```
begin
case a of
  2: a := b; d := 0; end;
  4: a := c;
end;
```

Задания (с защитой от неверного ввода)

"4": Ввести номер месяца и вывести количество дней в нем, а также число ошибок при вводе.

Пример:

Введите номер месяца:

-2

Введите номер месяца:

11

В этом месяце 30 дней.

Вы вводили неверно 1 раз.

Введите номер месяца:

2

В этом месяце 28 дней.

Вы вводили неверно 0 раз.

"5": Ввести номер месяца и номер дня, вывести число дней, оставшихся до Нового года.

Пример:

Введите номер месяца:

12

Введите день:

25

До Нового года осталось 6 дней.