Информатика

ОСНОВЫ АЛГОРИТМИЗАЦИИ

Содержание раздела

1. Понятие алгоритма, его свойства. Основные алгоритмические конструкции
2. Простые и структурированные типы данных
3. Основы программирования
4. Программирование на языке Pascal

1. Понятие алгоритма, его свойства. Основные алгоритмические конструкции

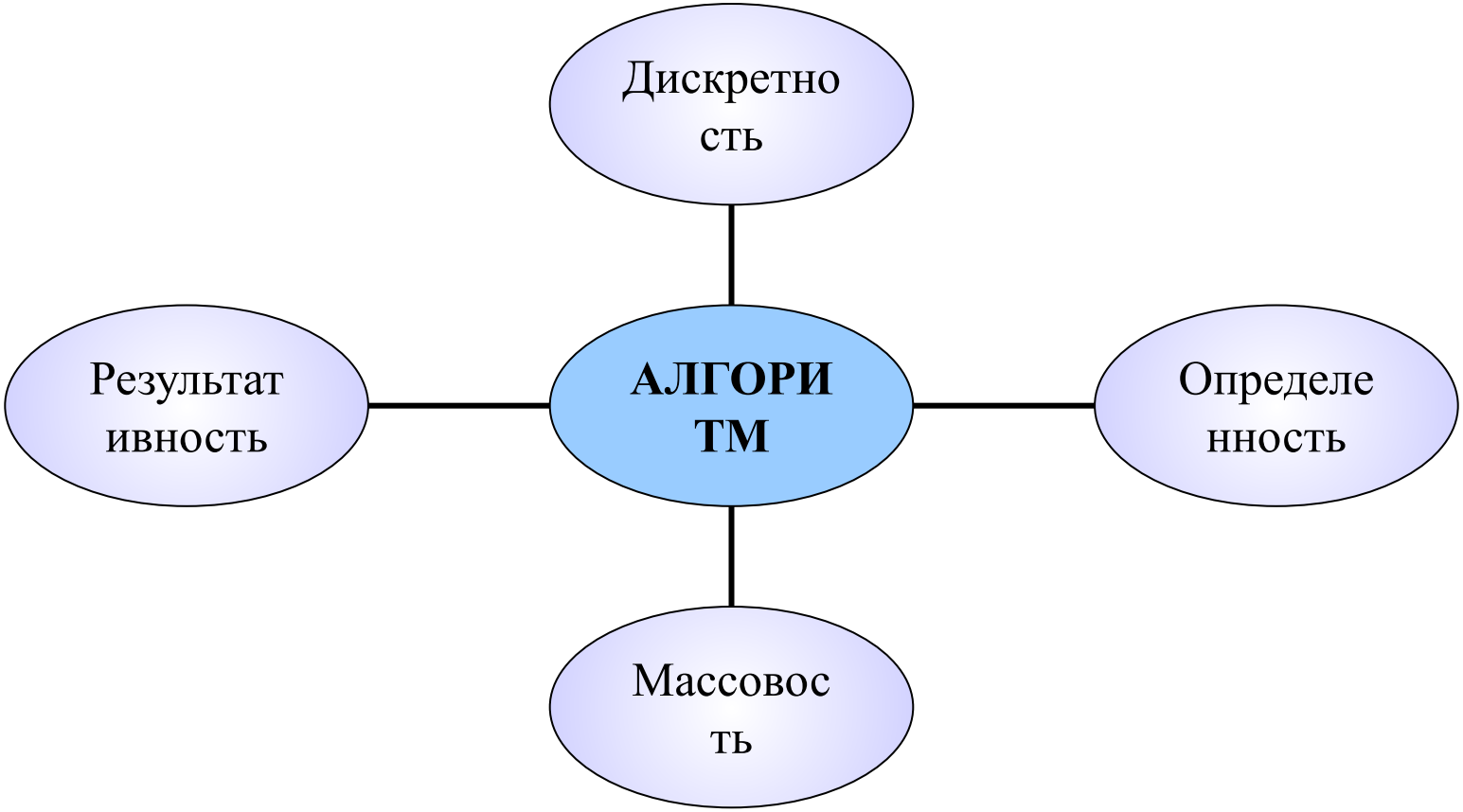
Понятие алгоритма

Алгоритм –упорядоченная совокупность системы правил, определяющая содержание и порядок действий над некоторыми объектами, строгое выполнение которых приводит к решению любой задачи из рассматриваемого класса задач за конечное число шагов.

Понятие алгоритма

Слово «*алгоритм*» появилось в средние века, когда европейцы познакомились со способами выполнения арифметических действий в десятичной системе счисления, описанными узбекским математиком Мухxамедом бен Аль-Хорезми («Аль-Хорезми» – человек из города Хорезми). Слово *алгоритм* – есть результат европейского произношения слов Аль-Хорезми.

Свойства алгоритмов



Свойства алгоритмов

- **Дискретность (разрывность)** – это свойство алгоритма, характеризующее его структуру: каждый алгоритм состоит из отдельных законченных действий.
- **Массовость** – применимость алгоритма ко всем задачам рассматриваемого типа, при любых исходных данных.
- **Определенность** – свойство алгоритма, указывающее на то, что каждый шаг алгоритма должен быть строго определен и не допускать различных толкований; также строго должен быть определен порядок выполнения отдельных шагов.
- **Результативность** - конечность действий алгоритма решения задач, позволяющая получить желаемый результат при допустимых исходных данных за конечное число шагов.

Способы описания алгоритмов

- Словесное описание
- Псевдокод(школьный алгоритмический язык)
- табличный
- Блок-схема
- Программа

Способы описания алгоритмов

Словесное описание представляет структуру алгоритма на естественном языке.

Пример: инструкция по эксплуатации любого прибора бытовой техники (утюг, телевизор, электрочайник), рецепт блюда, правила дорожного движения.

Словесная форма имеет ряд **недостатков**:

— строго не формализуема;

— страдает многословностью записей;

— допускает неоднозначность толкования отдельных предписаний.

Обычно используется на начальных стадиях разработки алгоритма.

Способы описания алгоритмов

Псевдокод – пошагово-словесная запись алгоритма по определенным правилам или соглашениям.

Пример. Алгоритм сложения двух чисел:

1. Ввод a, b.
2. S=a + b.
3. Вывод S.
4. Конец.

Способы описания алгоритмов

Примером псевдокода является школьный алгоритмический язык. Основные служебные слова этого языка представлены в таблице 1.1.

Таблица 1.1 Служебные слова школьного алгоритмического языка

алг (алгоритм)	**сим** (символьный)	**дано**	**для**	**да**
арг (аргумент)	**лит** (литерный)	**надо**	**от**	**нет**
рез (результат)	**лог** (логический)	**если**	**до**	**при**
нач (начало)	**таб**(таблица)	**то**	**знач**	**выбор**
кон (конец)	**нц** (начало цикла)	**иначе**	**и**	**ввод**
цел (целый)	**кц** (конец цикла)	**все**	**или**	**вывод**
вещ (вещественный)	**длин** (длина)	**пока**	**не**	**утв**

Пример записи алгоритма на школьном АЯ

алг Сумма квадратов (**арг цел** n, **рез цел** S)
дано | n > 0
надо | S = 1*1 + 2*2 + 3*3 + ... + n*n
нач цел i
ввод n; S:=0
нц для i **от** 1 **до** n
S:=S + i*I

кц вывод "S = ", S**кон**

Способы описания алгоритмов

Блок-схема – это наглядное графическое представление алгоритма с помощью геометрических фигур, соединенных линиями связями, показывающими порядок выполнения инструкций.

Графические объекты блок-схем

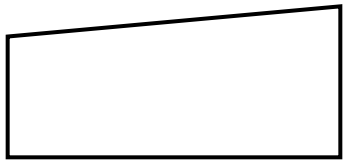
- Начало алгоритма

- Конец алгоритма

- Процесс

- Ввод/вывод данных с неопределенного носителя

Графические объекты блок-схем



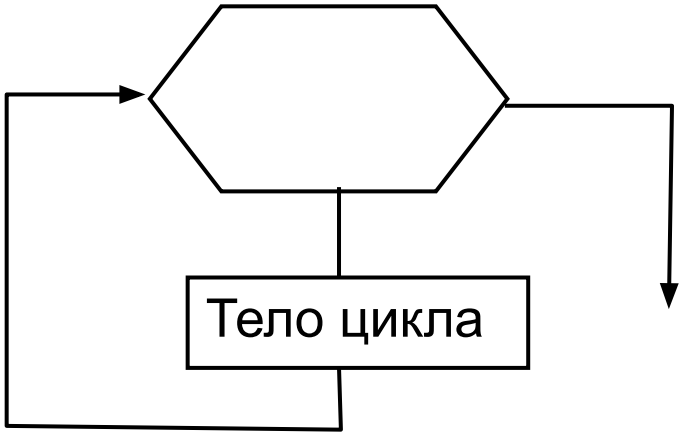
- Ввод с клавиатуры

- Вывод на монитор

- Вывод на печатающее устройство

- Условный блок

Графические объекты блок-схем



- Цикл с параметром

Тело цикла

- Границы цикла

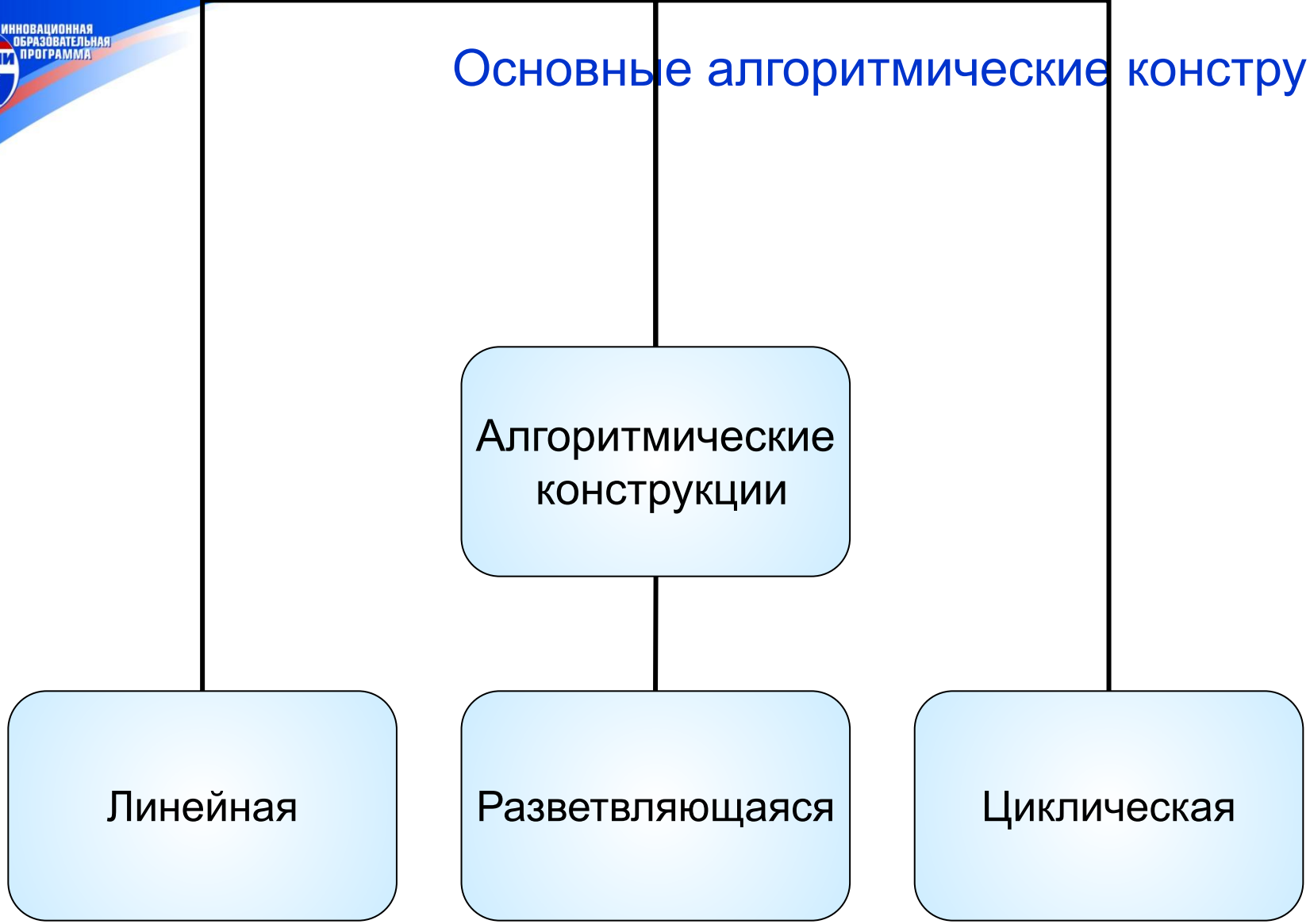
Способы описания алгоритмов

Программа – описание структуры алгоритма на языке программирования.

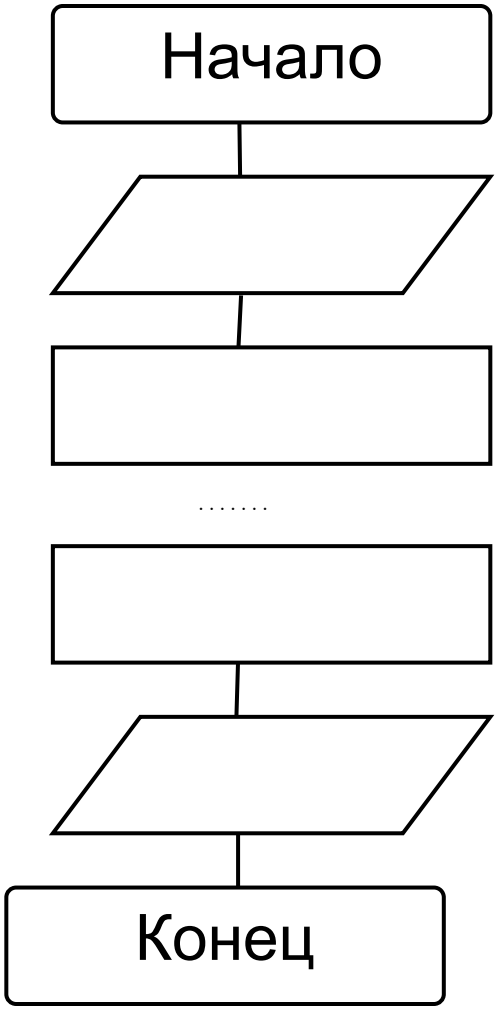
Пример:

```
program sistema ;
var
x, xn, xk, dx, y: real ;
uses crt ;
begin
clrscr;
writeln ('Введите начальное, конечное
значение x и шаг') ;
readln (xn, xk,dx) ;
x:=xn;
repeat
if x > 0 then y:=ln(x)
else y:=sqr(x)+5*x;
writeln ('x=', x:6:2, 'y=',y) ;
x:=x+dx;
until x > xk;
readln;
end.
```

Основные алгоритмические конструкции



Линейная алгоритмическая конструкция



Линейный алгоритм – это описание последовательности действий, которые выполняются однократно в заданном порядке

Примеры линейных алгоритмов

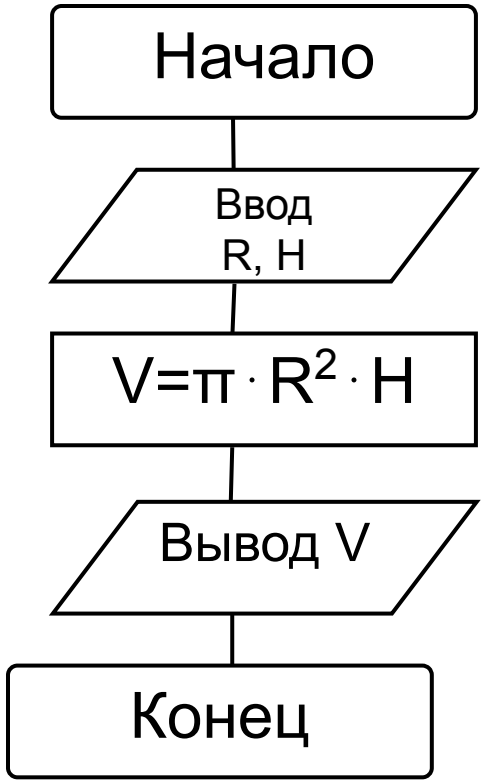
ПРИМЕР 1

Зная радиус основания и высоту цилиндра, вычислить его объем.

Псевдокод:

1. Ввод R и H.
2. $V=\pi \cdot R^2 \cdot H$.
3. Вывод V.
4. Конец.

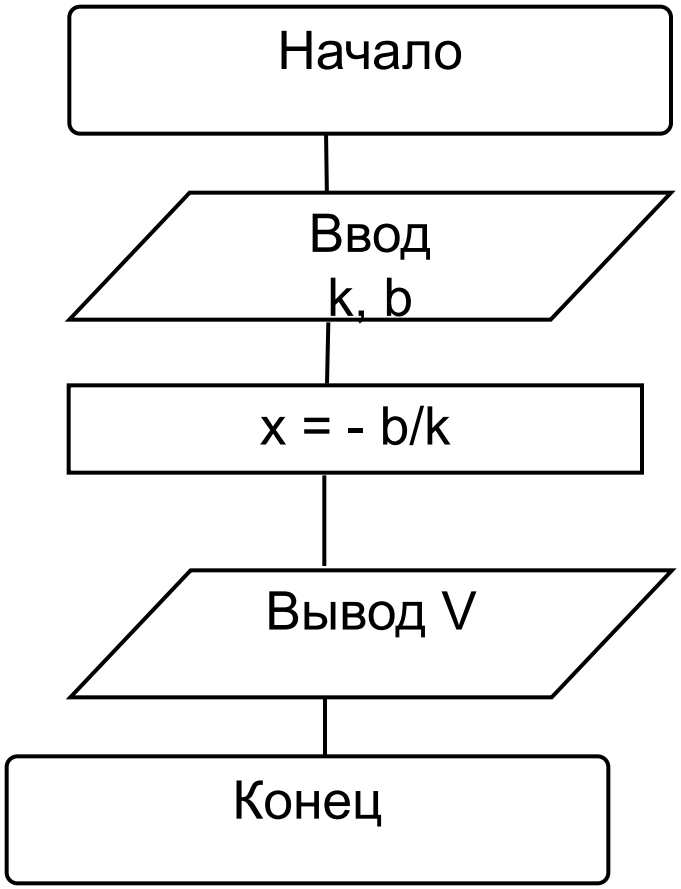
Блок-схема:



ПРИМЕР 2

Составить алгоритм для решения линейного уравнения $k \cdot x+b=0$.

Блок-схема:



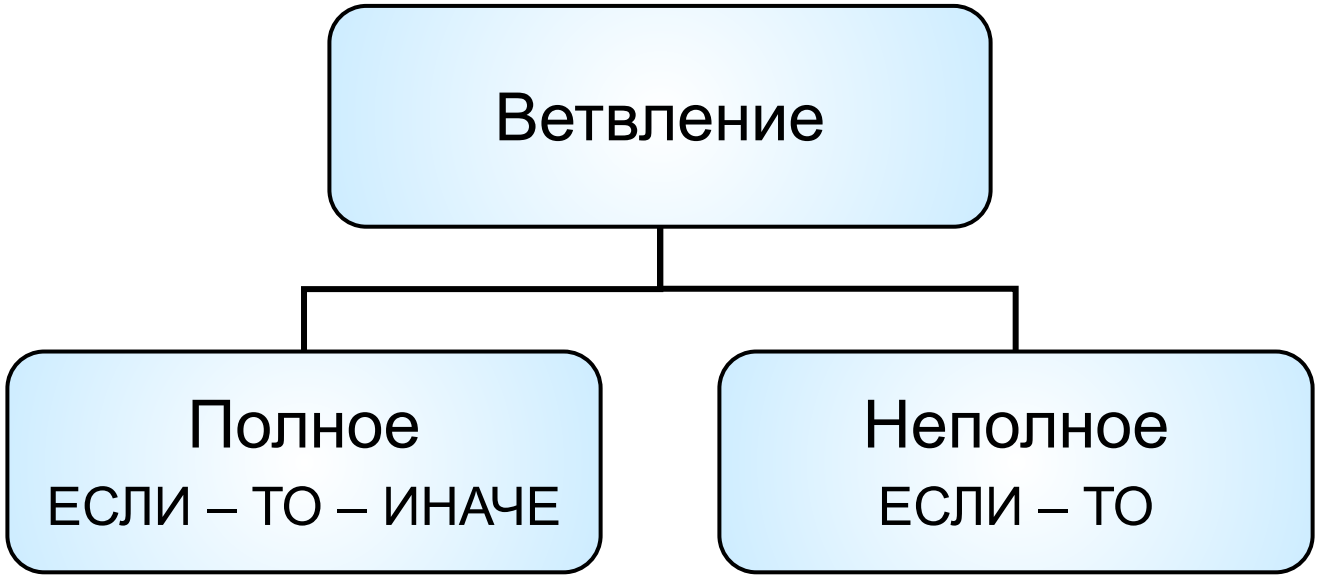
Псевдокод:

1. Ввод k, b.
2. $x = -\frac{b}{k}$
3. Вывод x.
4. Конец.

Разветвляющаяся алгоритмическая конструкция

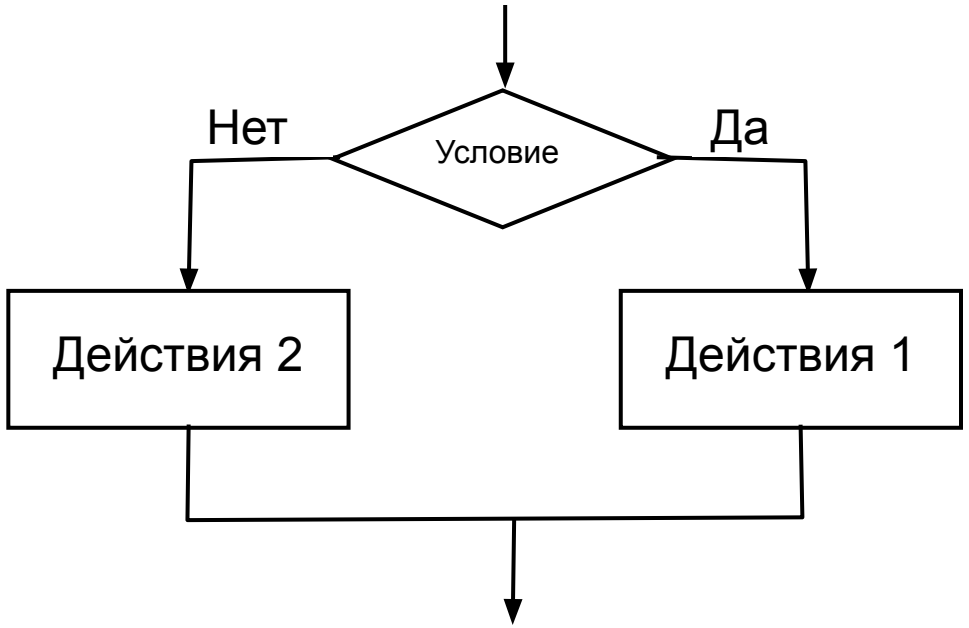
Разветвляющийся алгоритм – такой алгоритм, в котором выполняется либо одна, либо другая последовательность действий, в зависимости от условия.

Разветвляющаяся алгоритмическая конструкция



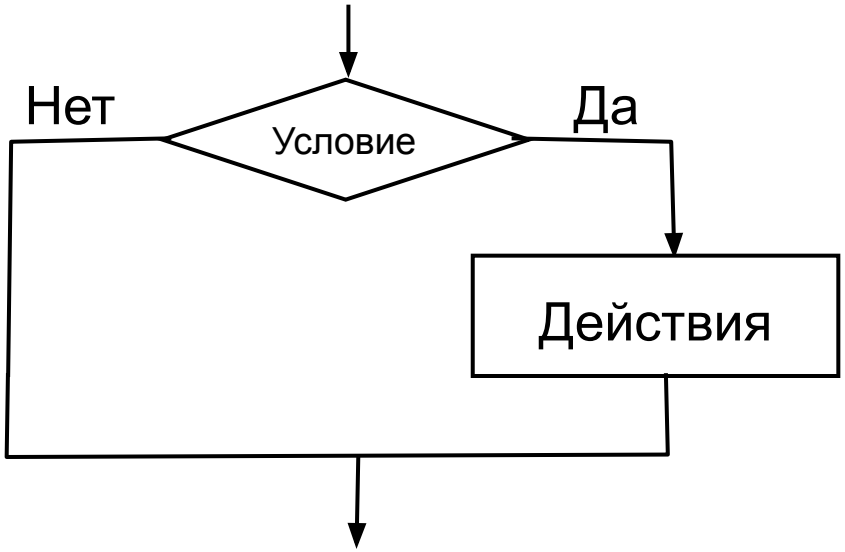
Разветвляющаяся алгоритмическая конструкция

Полное ветвление



Полное ветвление позволяет организовать в алгоритме две ветви (ТО или ИНАЧЕ).

Неполное ветвление



Неполное ветвление предполагает наличие действий только на одной ветви (ТО), вторая ветвь отсутствует.

ПРИМЕР 1

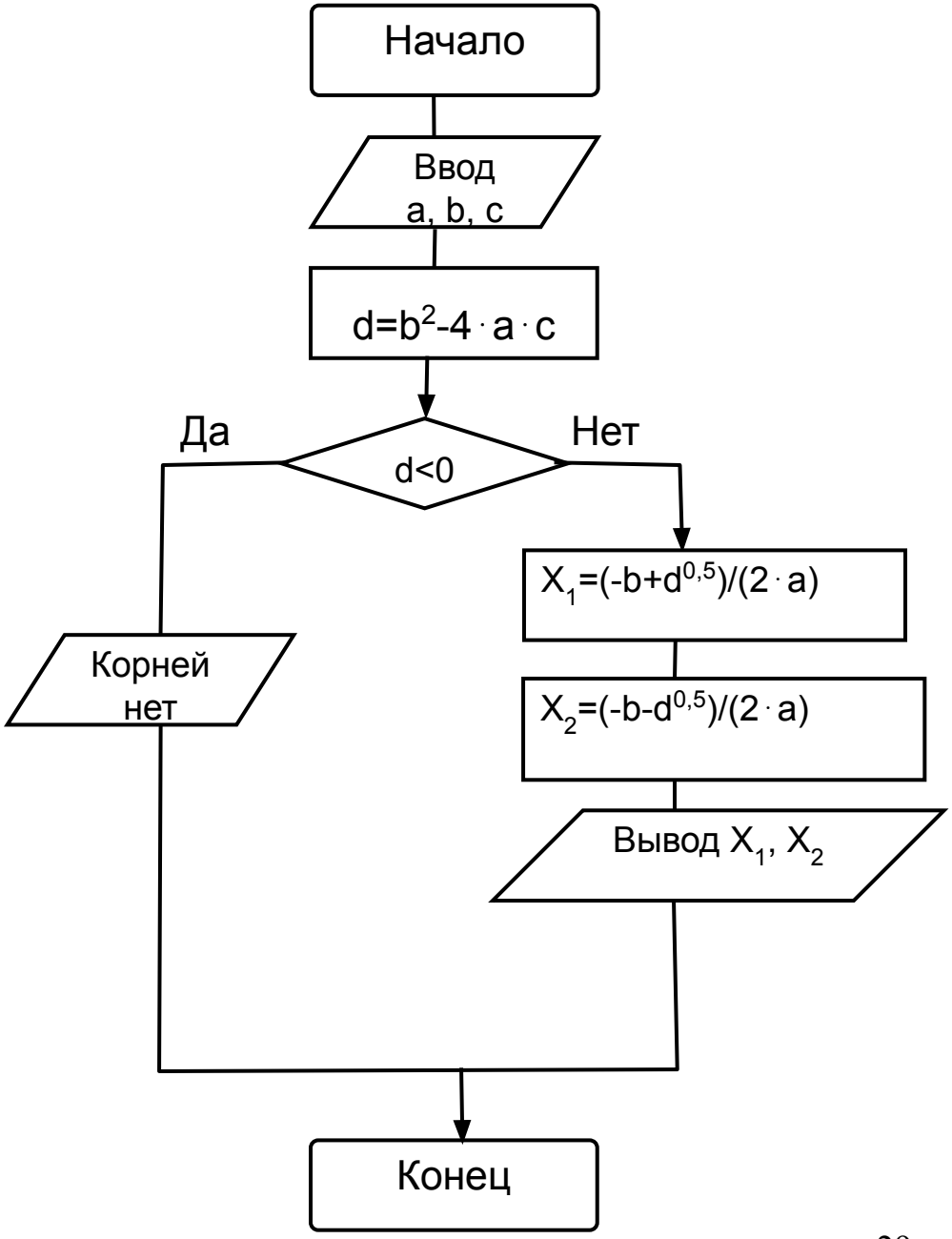
Известны коэффициенты a, b, c квадратного уравнения. Вычислить корни квадратного уравнения.

ПРИМЕР 1

Псевдокод:

1. Ввод a, b, c.
2. $d=b^2-4 \cdot a \cdot c$.
3. ЕСЛИ d<0, ТО «Корней нет», перейти к п.5.
   ИНАЧЕ $X_1=(-b+d^{0,5})/(2 \cdot a)$, $X_2=(-b-d^{0,5})/(2 \cdot a)$.
4. Вывод X_1 и X_2.
5. Конец.

Блок-схема:


Начало
Ввод
a, b, c
d=b2-4 · a · c
Да
d<0
Нет
Корней
нет
X1=(-b+d0,5)/(2 · a)
X2=(-b-d0,5)/(2 · a)
Вывод X1, X2
Конец

ПРИМЕР 2

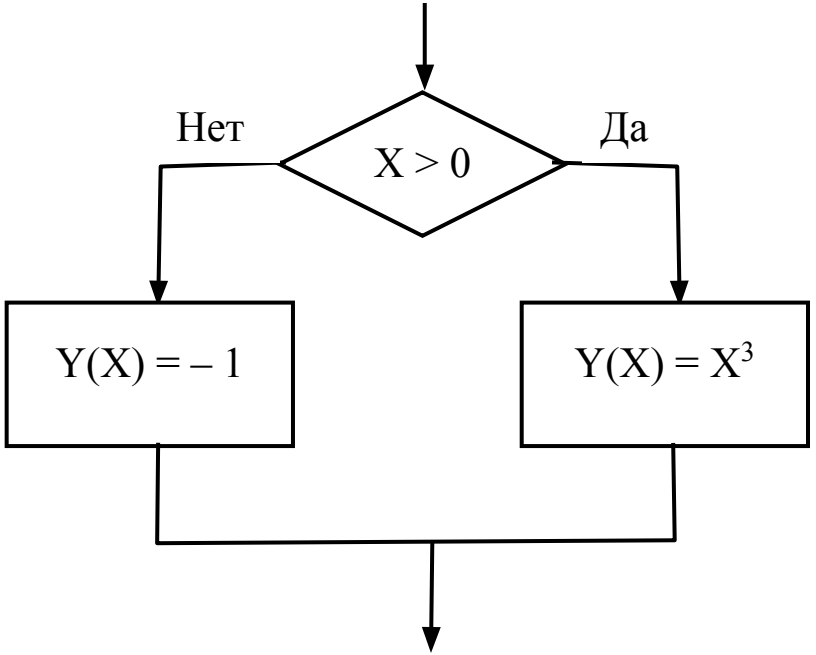
Составить псевдокод и блок-схему к алгоритму вычисления значения функции

$$Y(X)=\begin{cases}-1, x\le 0\\ x^3, x>0\end{cases}$$

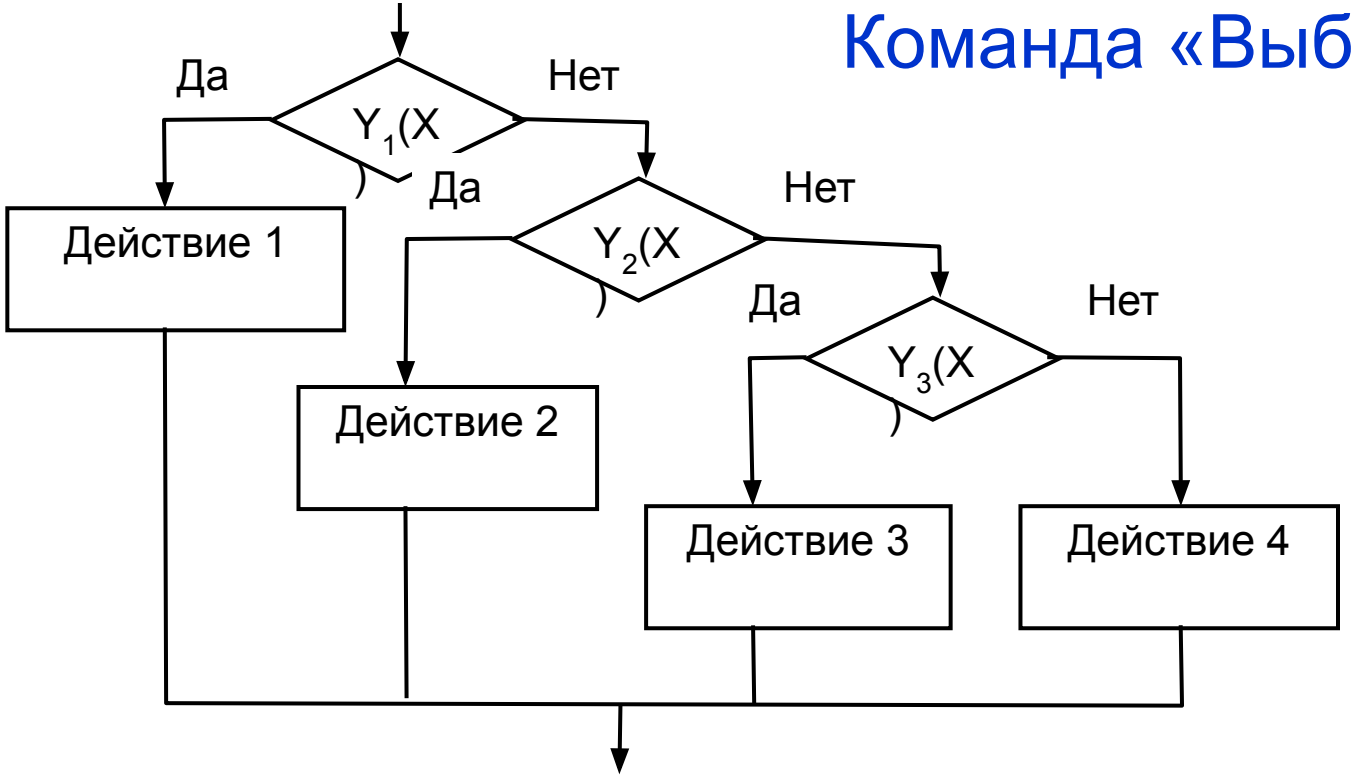
ПРИМЕР 2

Псевдокод:

1. Ввод X.
2. ЕСЛИ X > 0, ТО Y(X) = X^3,
   ИНАЧЕ Y(X) = – 1.
3. Вывод Y(X).
4. Конец.

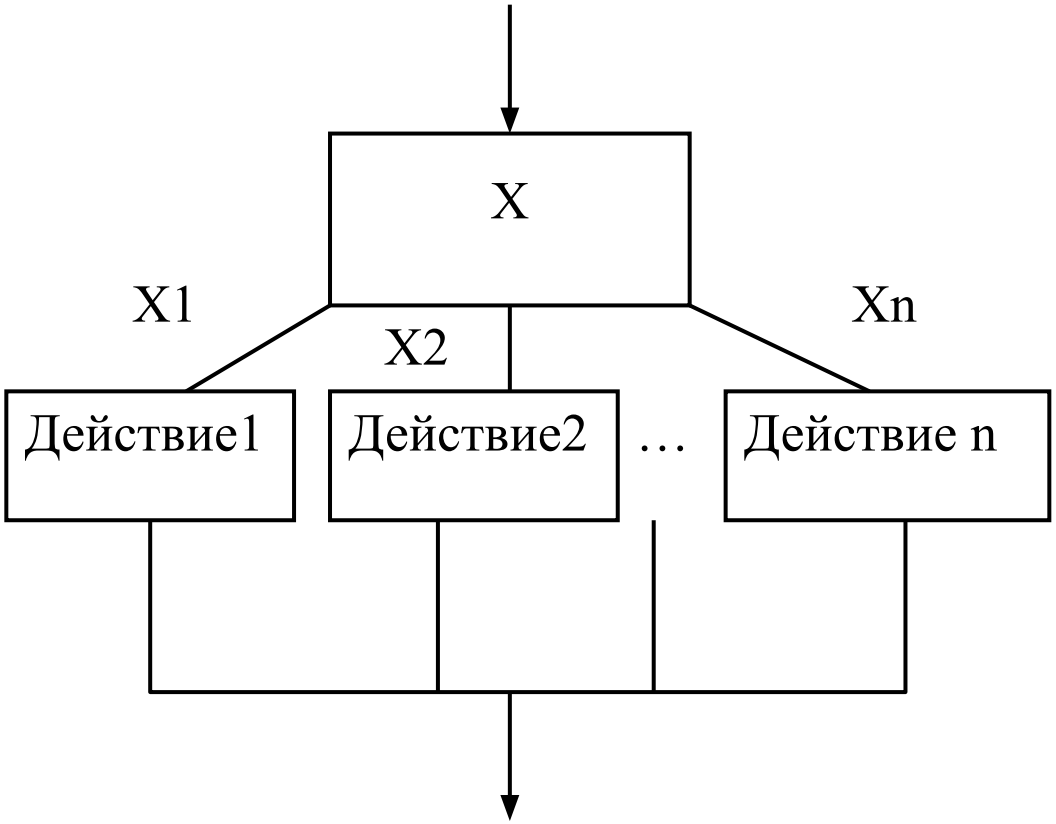


Команда «Выбор»



Перед выполнением команды «выбор» вычисляется значение некоторого выражения X, а затем начинается проверка условий Y1(X), Y2(X)...Yn(X). Проверка продолжается до тех пор, пока не встретится условие, принимающее значение ИСТИНА при данном X.

Команда «Выбор»



ПРИМЕР

Составить блок-схему к программе, которая запрашивает у пользователя номер дня недели и выводит одно из сообщений *«Рабочий день»*, *«Суббота»* или *«Воскресенье».*

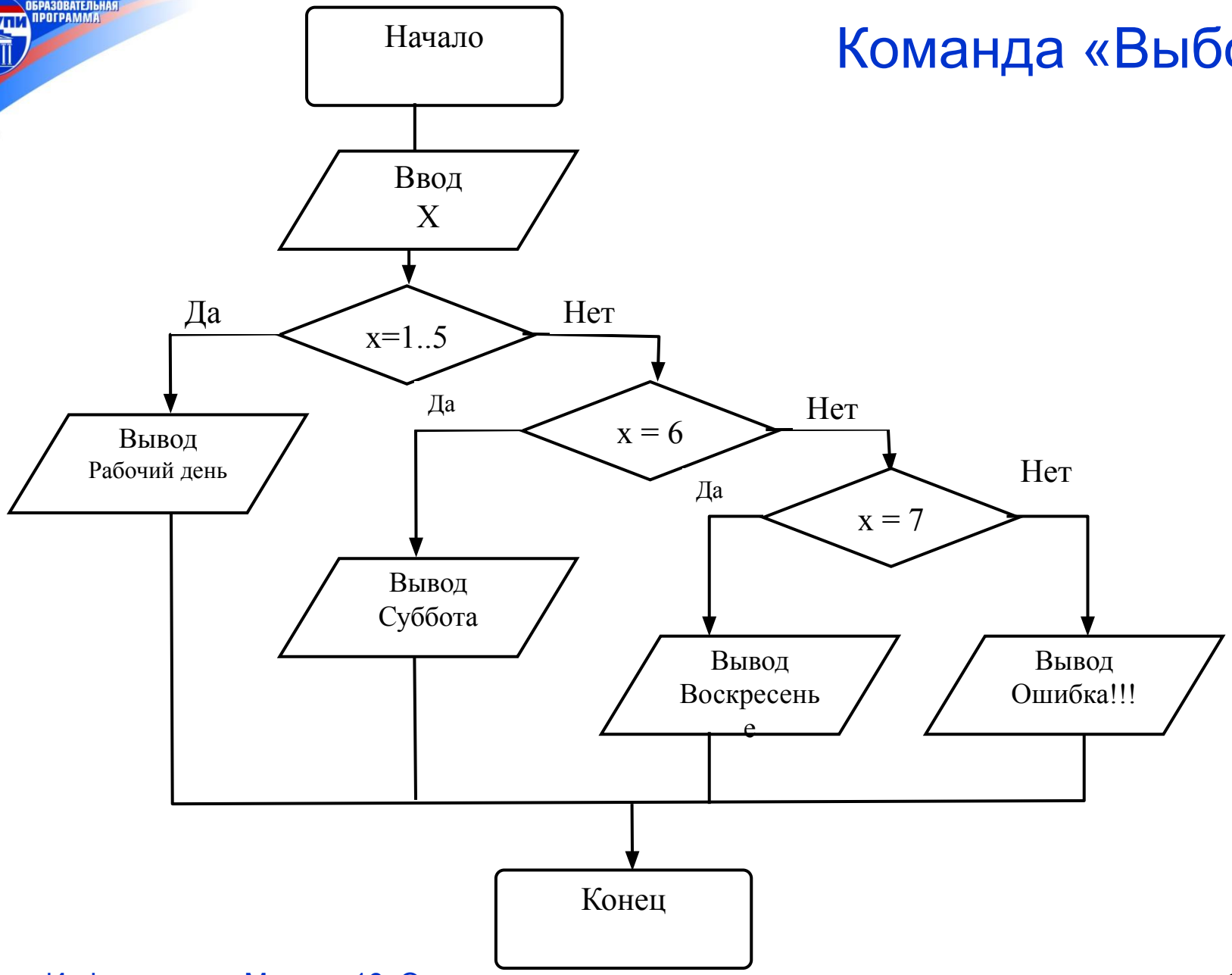
Псевдокод:

1. Ввод X.
2. Выбор **X** :
   1.. 5: вывод «*рабочий день*»;
   6: вывод «*суббота*»;
   7: вывод «*воскресенье*»;

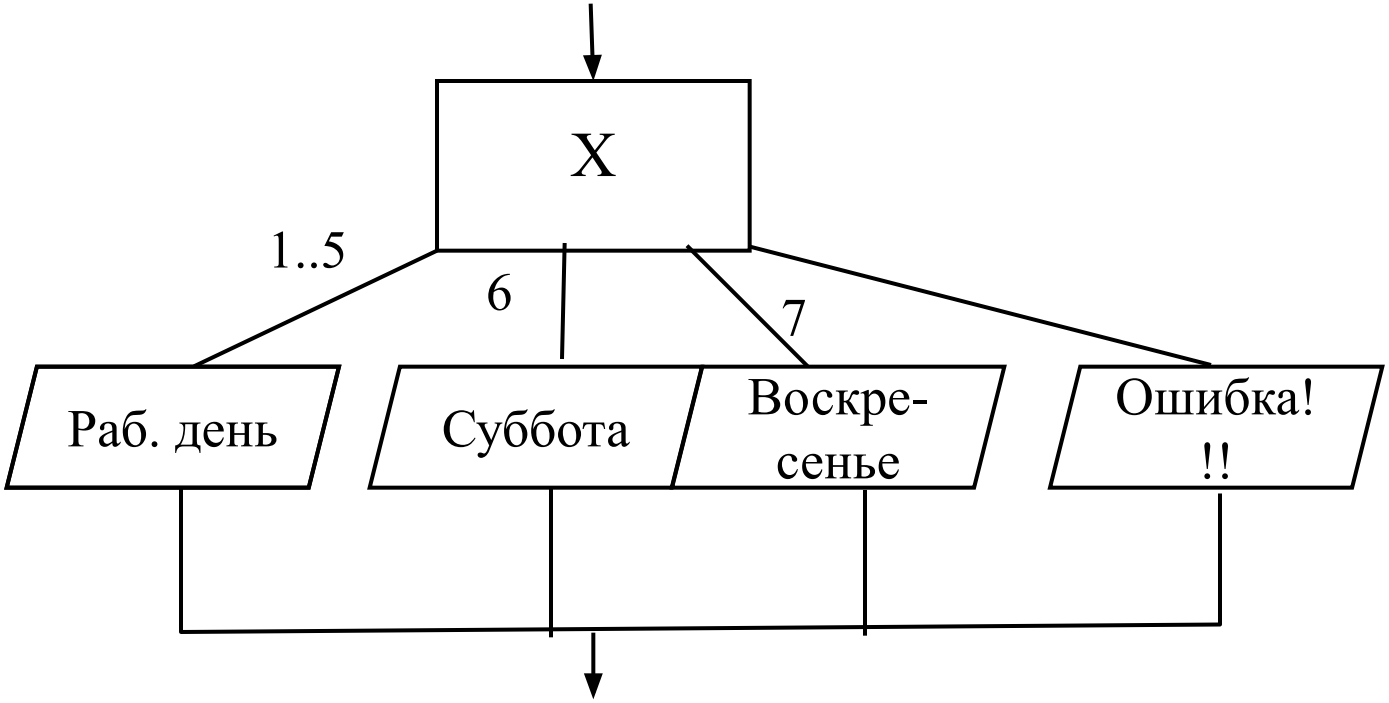
ИНАЧЕ вывод «*Ошибка! Введите число от 1 до 7*».

- Конец.

Команда «Выбор»



2 Вариант

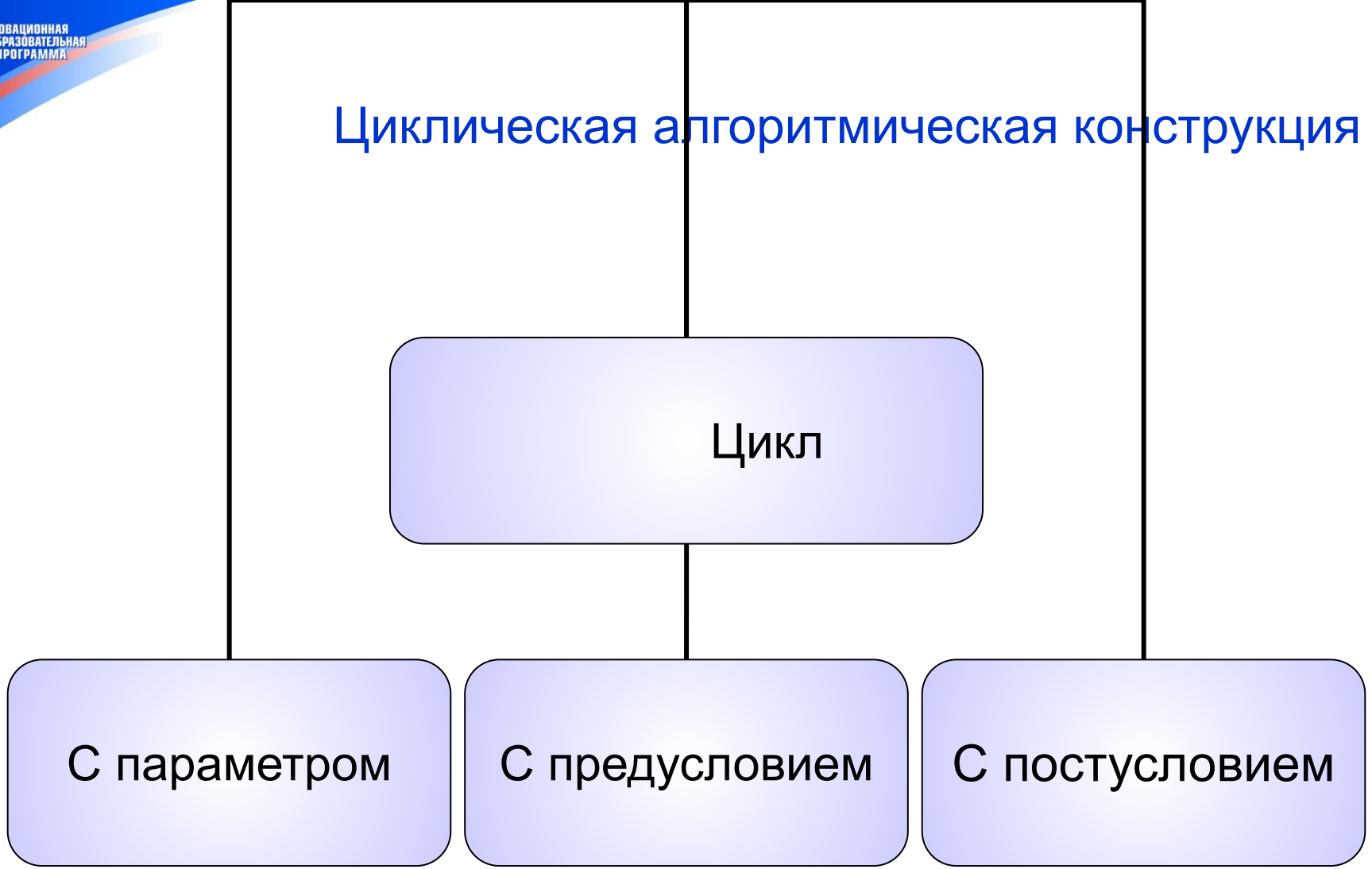


Циклическая алгоритмическая конструкция

Циклической называют алгоритмическую конструкцию, в которой действие выполняется указанное число раз, или, пока не выполнится условие.

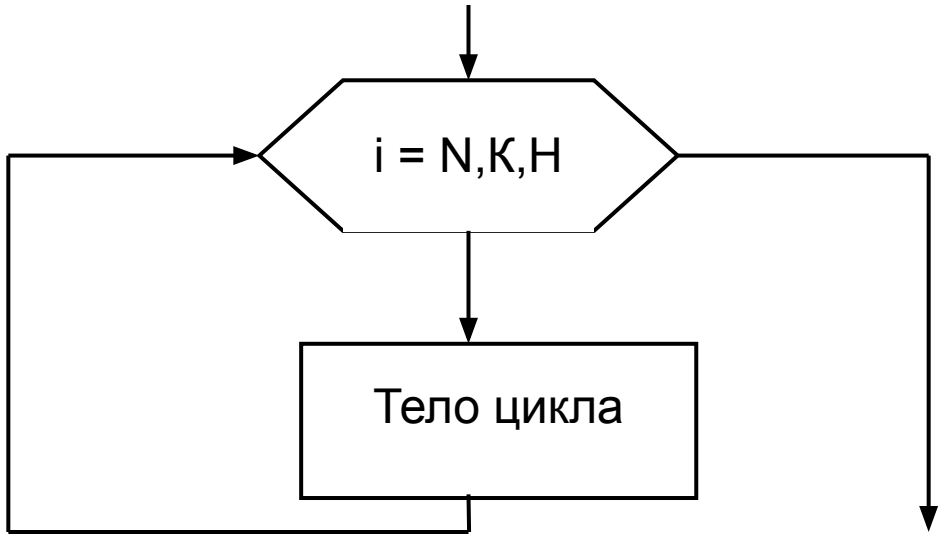
Группа повторяющихся действий цикла называется телом цикла.

Циклическая алгоритмическая конструкция

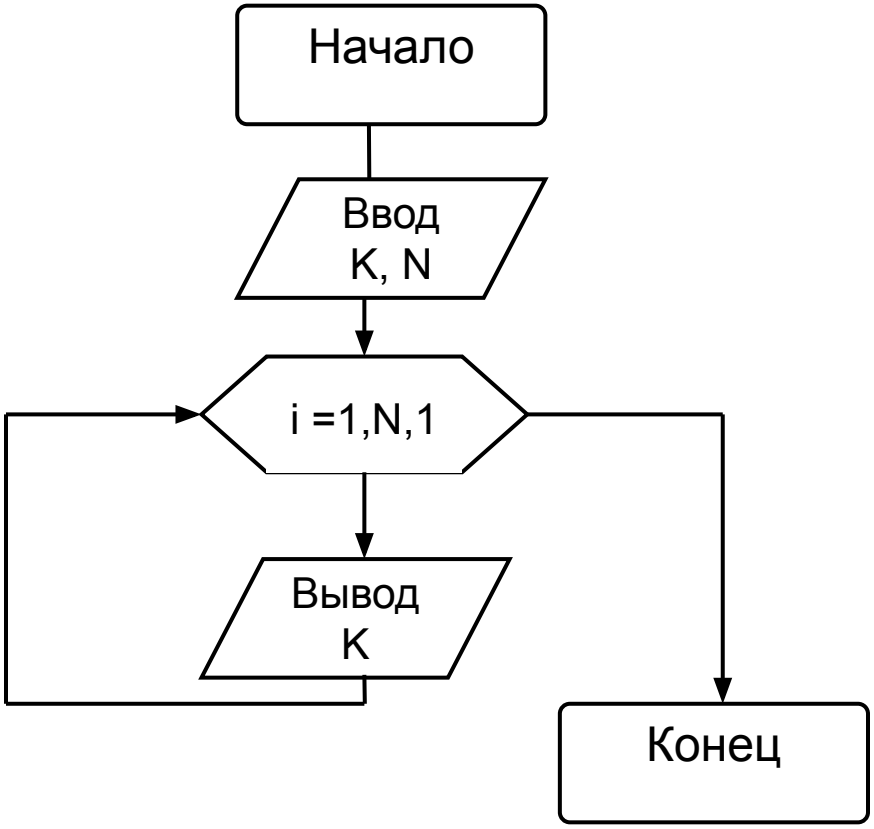


Цикл с параметром

В цикле с параметром число повторений цикла однозначно определено и задается с помощью *начального, конечного значений параметра и шагом его изменения.*

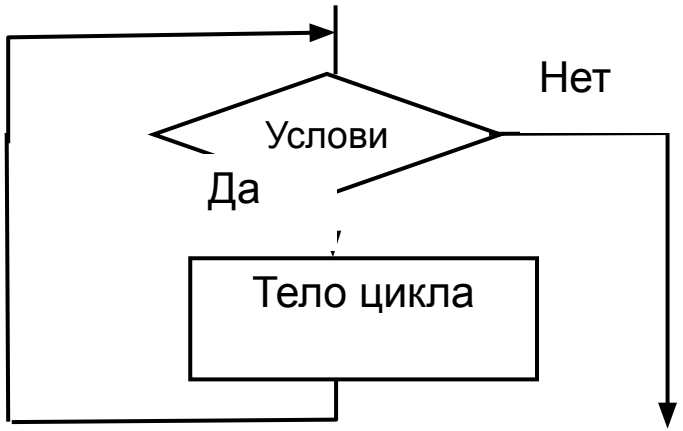


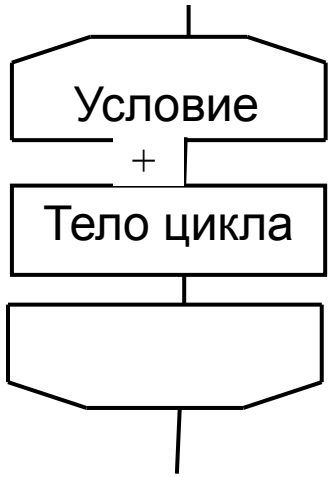
ПРИМЕР. Даны целые числа K и N (N>0). Вывести N раз число K.



Цикл с предусловием

Действия внутри этого цикла повторяются, пока выполняется условие в блоке ветвления, причем сначала проверяется условие, а затем выполняется действие.



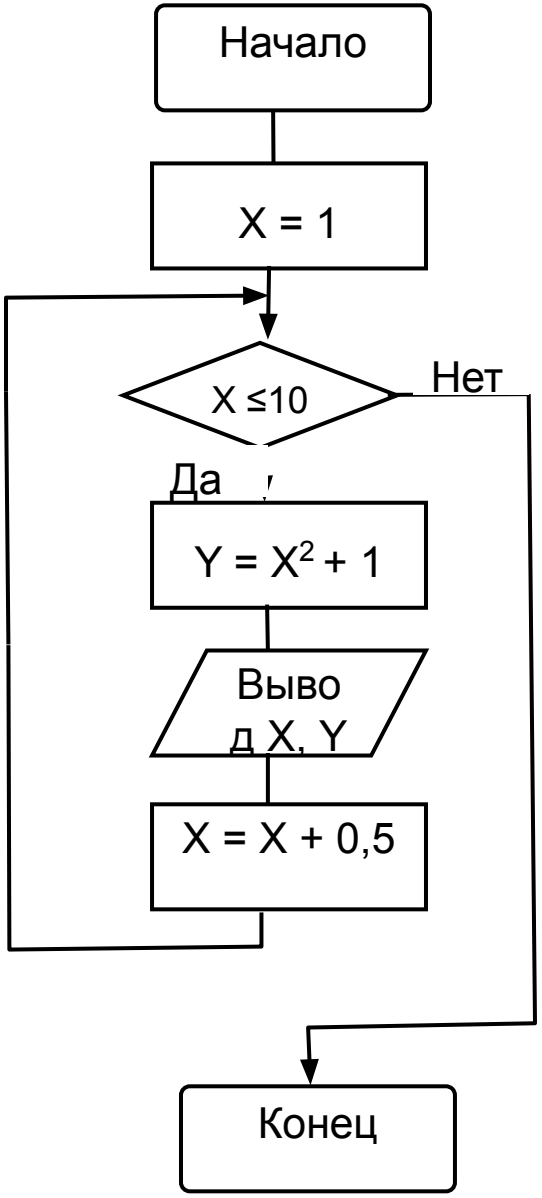


Цикл с предусловием

ПРИМЕР. Найти значение всех $Y = X^2 + 1$ при X, изменяющемся от 1 до 10 с шагом 0,5.

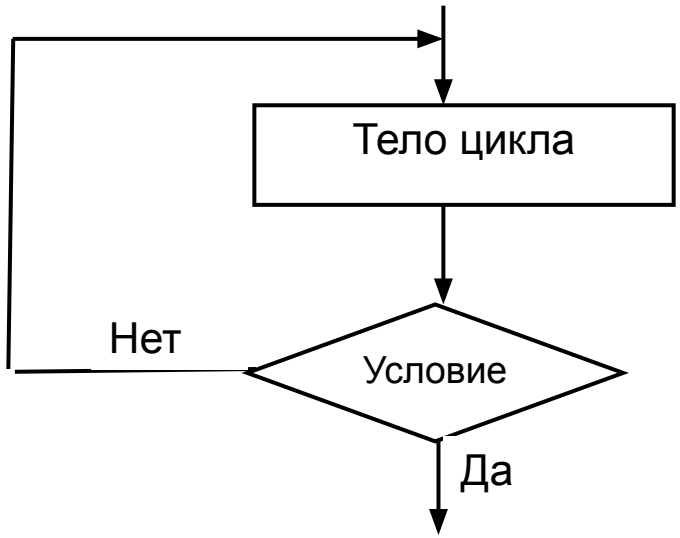
Цикл с предусловием

ПРИМЕР. Найти значение всех $Y = X^2 + 1$ при X, изменяющемся от 1 до 10 с шагом 0,5.



Цикл с постусловием

Тело цикла с постусловием всегда будет выполнено хотя бы один раз. Оно будет выполняться до тех пор, пока значение условного выражения ЛОЖНО. Как только условное выражение принимает значение ИСТИНА, цикл завершается.



Цикл с постусловием

ПРИМЕР. Составить блок-схему к программе, которая запрашивает у пользователя положительные числа, считает их сумму и количество. Как только введено отрицательное число или ноль, программа завершается.

Цикл с постусловием

ПРИМЕР. Составить блок-схему к программе, которая запрашивает у пользователя положительные числа, считает их сумму и количество. Как только введено отрицательное число или ноль, программа завершается.

Правило суммирования:

1. Необходимо задать начальное значение суммы S = 0.
2. В теле циклической конструкции выполнить команду: *S = S + <слагаемое>.*

Цикл с постусловием

ПРИМЕР. Составить блок-схему к программе, которая запрашивает у пользователя положительные числа, считает их сумму и количество. Как только введено отрицательное число или ноль, программа завершается.

Правило суммирования:

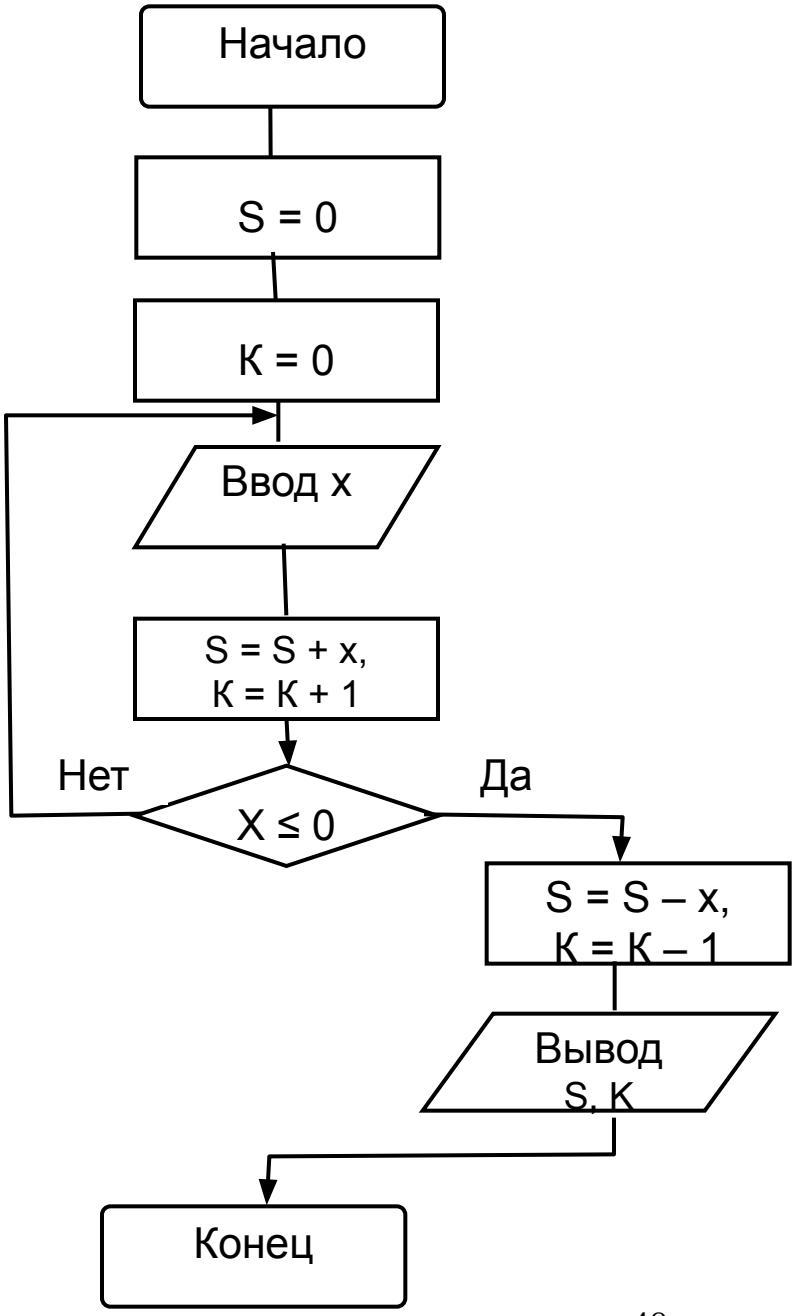
1. Необходимо задать начальное значение суммы S = 0.
2. В теле циклической конструкции выполнить команду: *S = S + <слагаемое>.*

Правило счетчика:

1. Начальное значение счетчика К = 0.
2. В теле цикла выполнить команду *К = К + 1.*

Псевдокод:

1. S = 0, K = 0.
2. Ввод x.
3. S = S + x.
4. K = K + 1.
5. Если x ≤ 0, ТО переходим к шагу 6. ИНАЧЕ переходим к п. 2.
6. S = S – x, K = K – 1.
7. Вывод S, K
8. Конец.



2. Простые и структурированные типы данных

Простые типы данных

Целые, вещественные числа, логические величины – все это *простые* типы данных (или *базовые*).

Переменная – это именованный объект (ячейка памяти), который может изменять свое значение. Кроме имени и значения, переменная имеет *тип*, определяющий, какая информация находится в памяти.

Простые типы данных

Если переменные присутствуют в программе, на протяжении всего времени ее работы – их называют *статическими*.

Простые типы данных

Если переменные присутствуют в программе, на протяжении всего времени ее работы – их называют *статическими*.

Переменные, создающиеся и уничтожающиеся на разных этапах выполнения программы, называют *динамическими*.

Простые типы данных

Если переменные присутствуют в программе, на протяжении всего времени ее работы – их называют *статическими*.

Переменные, создающиеся и уничтожающиеся на разных этапах выполнения программы, называют *динамическими*.

Данные, значения которых не изменяются на протяжении работы программы, называют постоянными или константами.

Структурированные типы данных. Одномерные и двумерные массивы

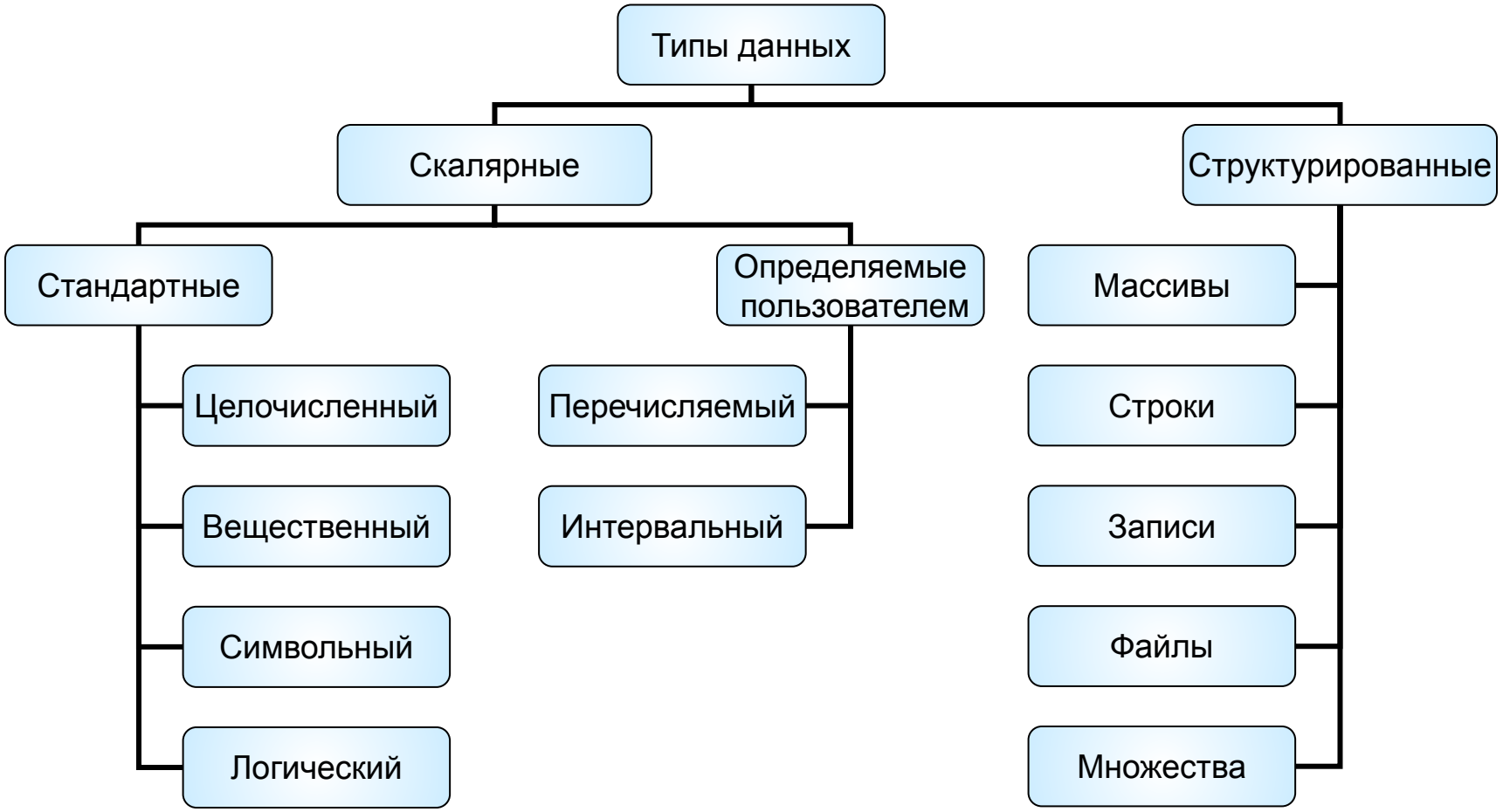
Тип данных, позволяющий хранить вместе под одним именем несколько переменных, называется структурированным.

К структурированным типам данных относятся:

- *массивы,*
- *строки,*
- *множества,*
- *записи,*
- *файлы*.

Значениями этих типов всегда являются простые типы.

Типы данных


Типы данных
Скалярные
Структурированные
Стандартные
Определяемые пользователем
Массивы
Целочисленный
Перечисляемый
Строки
Вещественный
Интервальный
Записи
Символьный
Файлы
Логический
Множества

Целочисленные типы

Определяют константы, переменные и функции, значения которых реализуются множеством целых чисел, допустимых в данной ЭВМ.

Тип	Диапазон	Размер (в байтах)
Byte	0….255	1
Word	0….65535	2
Integer	-32768…32767	2
Shortint	-128…127	1
Longint	2147483648..2147483647	4

Вещественные типы

Определяют те данные, которые реализуются подмножеством действительных чисел, допустимых в данной ЭВМ.

Тип	Диапазон	Размер (в байтах)
Real	2.9E-39…1.7E38	6
Single	1.5E-45…3.4E38	4
Double	5E-324…1.7E308	8
Extended	3.4E-49321..1E4932	10

Стандартные типы данных

Символьный тип представляет собой любой символ, который может быть отображен на экране дисплея. Он занимает 1 байт и может быть описан с помощью служебного слова *char*.

В тексте программы значения переменных и константы символьного типа должны быть заключены в апострофы.

Логический тип (boolean) определяет те данные, которые могут принимать логические значения TRUE и FALSE.

Типы, определяемые пользователем

Перечислимый тип задается непосредственным перечислением значений, которые может принимать переменная данного типа.

Например:

```
var
a, b: (read, blue, green);
```

Интервальный тип позволяет задавать две константы, которые определяют границы изменения переменных данного типа.

Например:

```
var
a: 1..10;
```

Структурированные типы

Массив – это совокупность данных одного и того же типа. Для описания массивов используется служебное слово array.

```
var
a: array[1..10] of integer;
```

Строки – последовательность символов.
При использовании в выражениях строка заключается в апострофы. Ее длина ограничена 255 символами. Для описания переменных строкового типа используется служебное слово string.

```
var
a: string[10];
```

Арифметические операции

Операция	Действие	Тип операндов	Тип результата
+	Сложение	Целый, вещественный	Целый, вещественный
-	Вычитание	Целый, вещественный	Целый, вещественный
*	Умножение	Целый, вещественный	Целый, вещественный
/	Деление	Целый, вещественный	Целый, вещественный
Div	Деление нацело	Целый	Целый
Mod	Остаток от деления	Целый	Целый
And	«И»	Целый	Целый
Shl	Сдвиг влево	Целый	Целый
Shr	Сдвиг вправо	Целый	Целый
Or	«ИЛИ»	Целый	Целый
Xor	Исключающее «ИЛИ»	Целый	Целый
-	Отрицание	Целый	Целый
Not	Логическое отрицание	Целый	Целый

Структура программы

Раздел описаний содержит:

- раздел описания констант;
- раздел описания типов;
- раздел описания переменных;
- раздел описания процедур и функций.