Основы алгоритмизации. Типы алгоритмов. Основные элементы языка программирования.

Лекция №2 по курсу
«Информатика»

Понятие и свойства алгоритма

Алгоритм – это набор точных предписаний, последовательное выполнение которых однозначно приводит задачу к решению за конечное число шагов.

Алгоритм обладает следующими свойствами:

- Детерминированность(определенность,точность) – четкость и ясность всех предписаний: исполнителю алгоритма должно быть точно известно, какая команда алгоритма выполняется следующей («Уходя, гасите свет»)

- Результативность – способность алгоритма приводить к решению задачи за конечное число шагов

- дискретность – предписание представляет собой последовательность четко выраженных отдельных команд, причем, выполнив одну команду, исполнитель выполняет другую команду, промежуточных состояний нет
- массовость (универсальность) – применимость алгоритма к решению задач определенного класса, чем шире этот класс, тем ценнее алгоритм

Существуют следующие способы записи алгоритмов:

- словесно-формульная запись
- графическая запись (схема алгоритма, иначе, графическая схема алгоритма, блок-схема)
- запись на конкретном языке программирования

- ***Словесный способ*** **записи алгоритмов представляет собой описание последовательных этапов обработки данных. Алгоритм задается в произвольном изложении на естественном языке.**

Пример.

Записать алгоритм нахождения **наибольшего общего делителя (НОД)** двух натуральных чисел (алгоритм Евклида).

Алгоритм может быть следующим:

1. задать два числа
2. если числа равны, то взять любое из них в качестве ответа и остановиться, в противном случае продолжить выполнение алгоритма;
3. определить большее из чисел;
4. заменить большее из чисел разностью большего и меньшего из чисел;
5. повторить алгоритм с шага 2.

Графическая схема алгоритма состоит из отдельных блоков, связанных линиями потоков

Каждый блок описывает конкретный шаг алгоритма

Схемы алгоритмов должны соответствовать действующим стандартам на оформление схем алгоритмов, программ, данных и систем

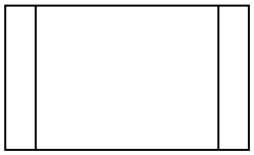
[ГОСТ 19.701-90].

Ниже приводятся некоторые символы, определенные в стандарте и рекомендуемые к использованию в графических схемах алгоритмов.

1. **Процесс**

Символ отображает функцию обработки данных любого вида.

2. **Предопределенный процесс**

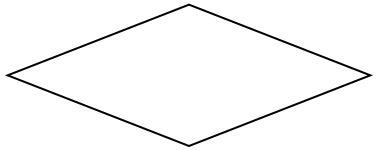


Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле).

3. **Данные**

Символ отображает данные, носитель данных не определен.

4. **Решение**

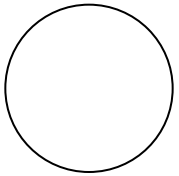


Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и из которых может быть активизирован после вычисления условий, определенных внутри этого символа.

5. **Линия**

Символ отображает поток данных или управления

6. Соединитель

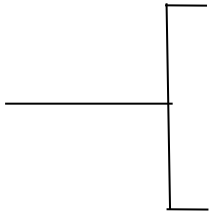


Символ отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линии и продолжения ее в другом месте. Соответствующие символы соединители должны содержать одно и то же уникальное обозначение.

7. Терминатор

Символ отображает начало или конец схемы программы, внешнее использование и источник или пункт назначения данных.

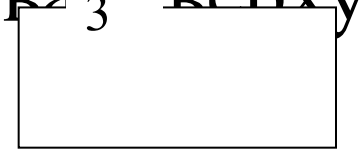
8. Комментарий



Текст, описывающий функцию символа, следует располагать внутри данного символа.

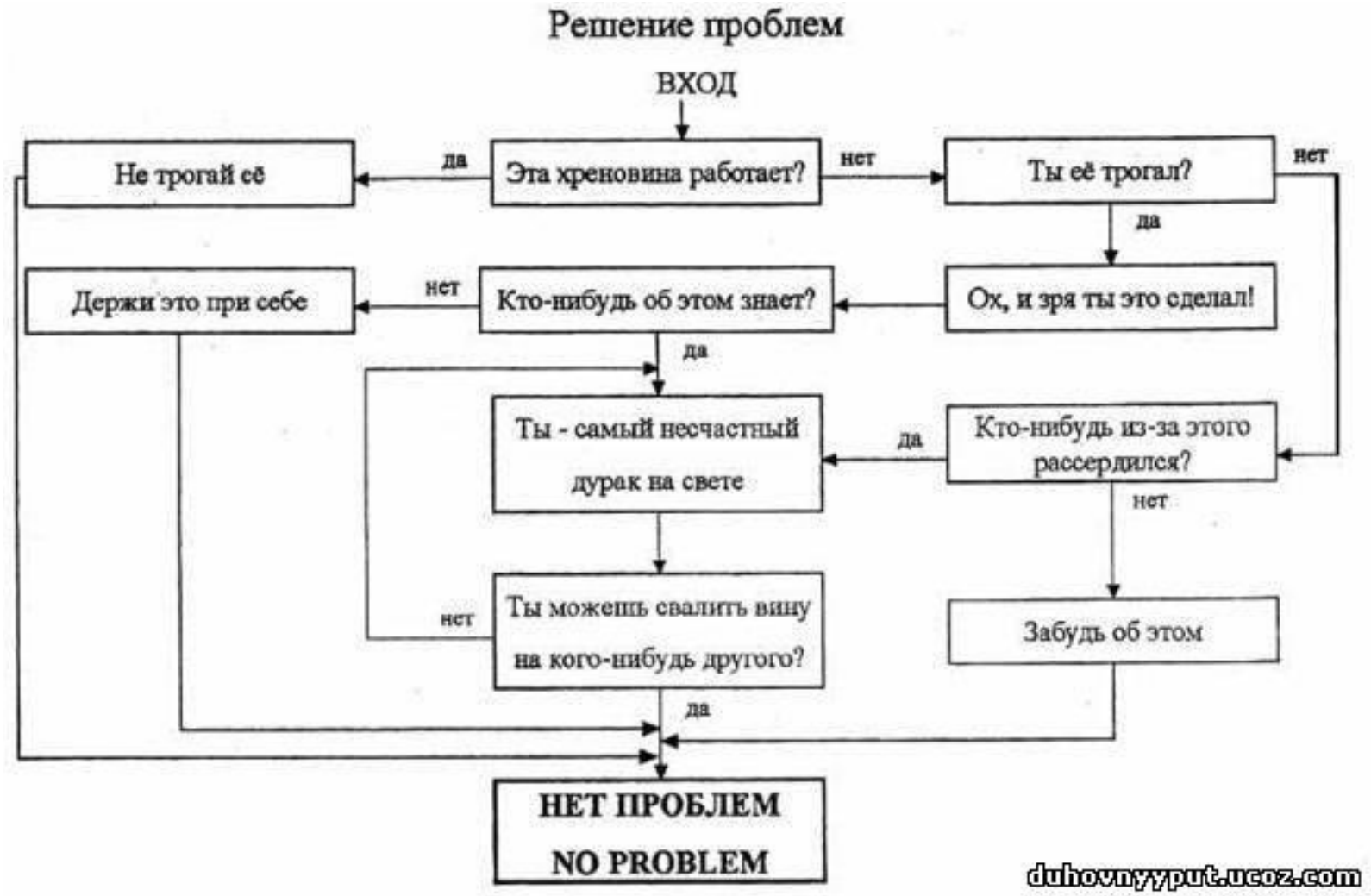
Если текст не помещается внутри символа, следует использовать символ комментария.

При необходимости блоки в схеме можно нумеровать (например, чтобы иметь возможность ссылаться на тот или иной символ) слева сверху в разъеме символа. Например,



Правила выполнения соединений:

- Стандартное направление линий потока – слева направо и сверху вниз
- Если направление потока отличается от стандартного, это направление указывается стрелками
- В схемах следует избегать пересечения линий
- Линии в схемах должны подходить к символу либо слева, либо сверху, а выходить либо справа, либо снизу.
- Вход в блок и выход из блока следует размещать по центру символа

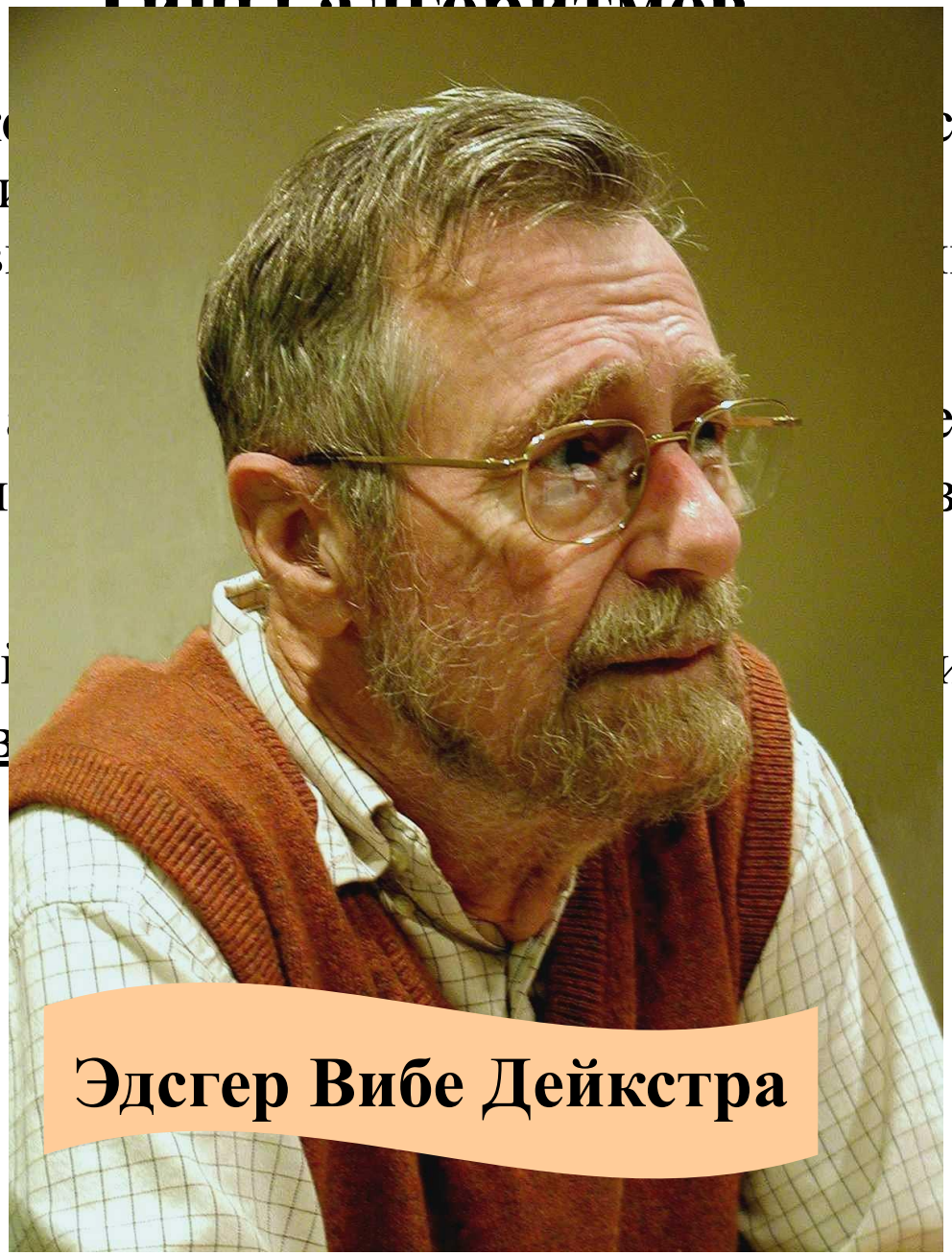

Решение проблем
ВХОД
Эта хреновина работает?
да
Не трогай её
нет
Ты её трогал?
нет
да
Ох, и зря ты это сделал!
Кто-нибудь об этом знает?
нет
Держи это при себе
да
Кто-нибудь из-за этого рассердился?
да
Ты - самый несчастный дурак на свете
нет
Забудь об этом
Ты можешь свалить вину на кого-нибудь другого?
нет
да
НЕТ ПРОБЛЕМ
NO PROBLEM
duhovnyyput.ucoz.com

Типы алгоритмов

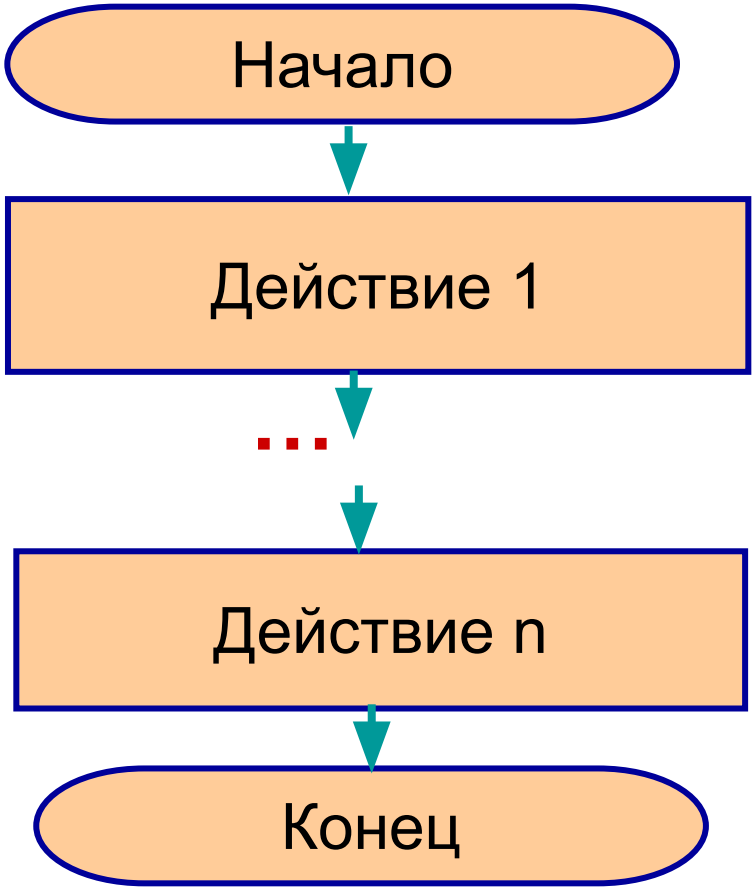
Теорема Дейксти можно реализовать, и следования (линейные), в клические).

Линейный - е действия выполняются оди записаны.

В схеме лине иде типовой структуры **следов**

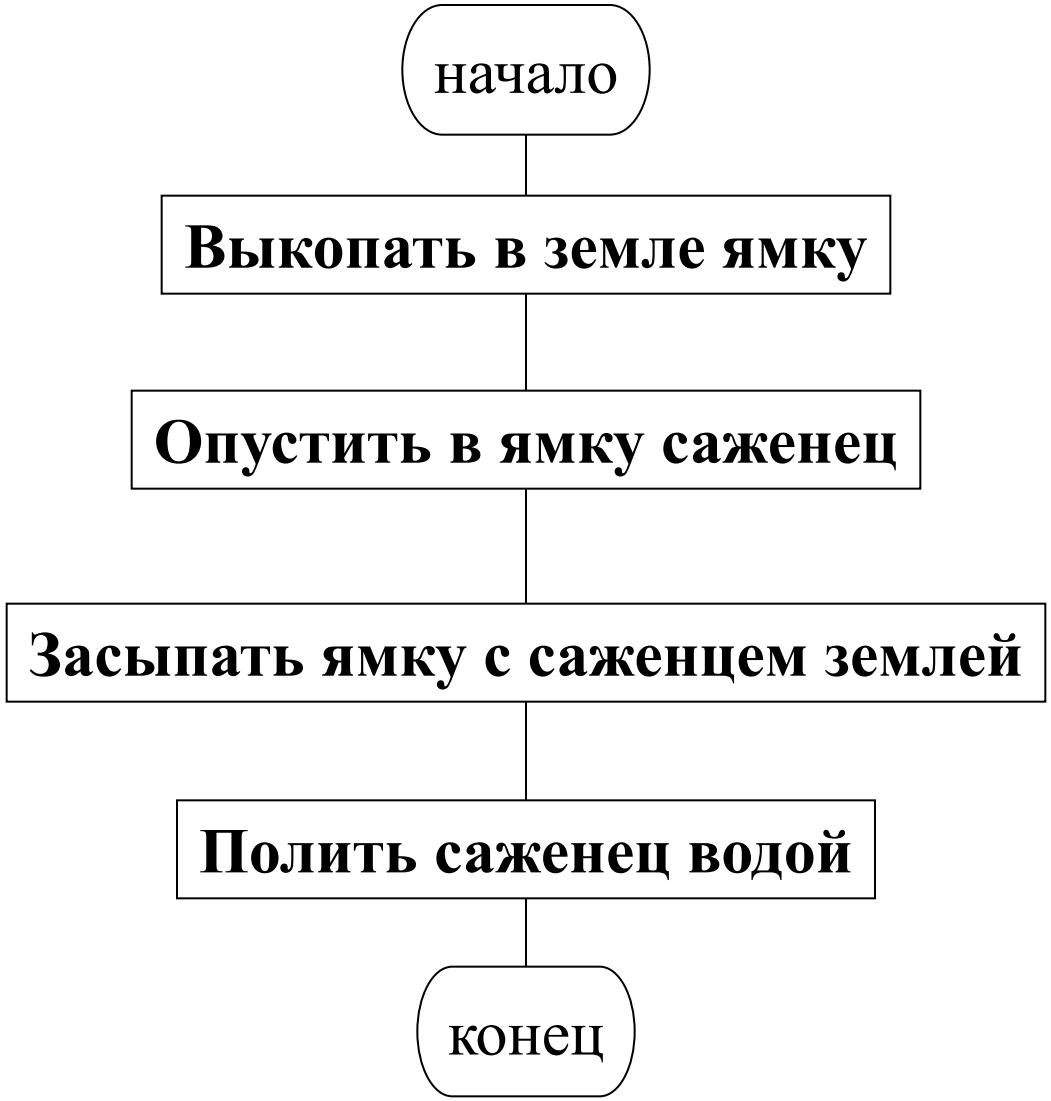


Эдсгер Вибе Дейкстра


Начало
Действие 1
...
Действие n
Конец

Например, алгоритм посадки дерева:

1) **Выкопать в земле ямку;**
2) **Опустить в ямку саженец;**
3) **Засыпать ямку с саженцем землей;**
4) **Полить саженец водой.**


начало
Выкопать в земле ямку
Опустить в ямку саженец
Засыпать ямку с саженцем землей
Полить саженец водой
конец

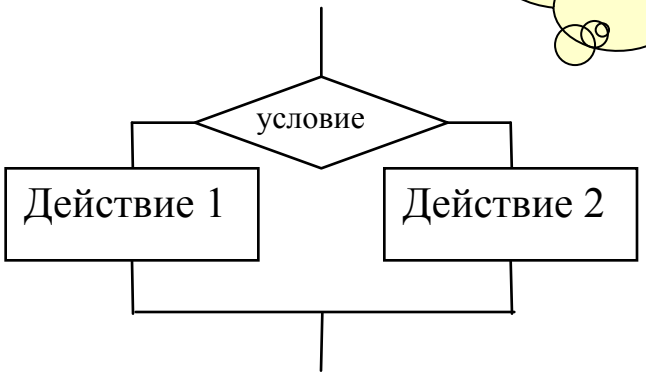
- **Разветвляющийся -** алгоритм, в котором некоторые действия выполняются один раз или не выполняются в зависимости от заданного условия.

В схеме разветвляющийся алгоритм представляется в виде типовых структур

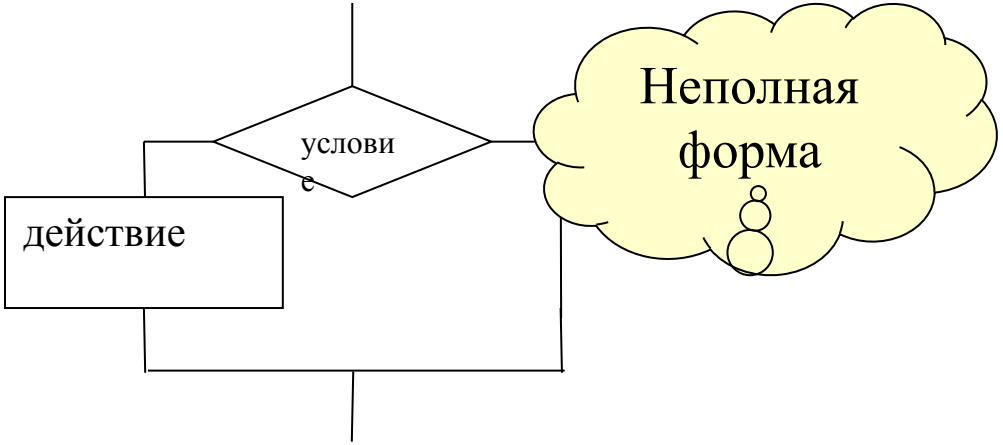
Ветвление и **выбор**

Ветвление

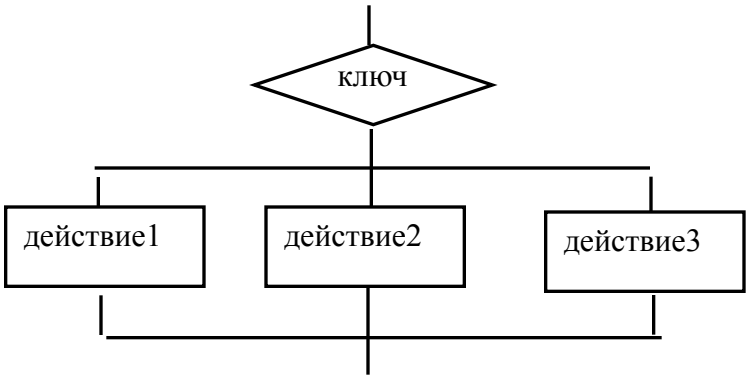
Полная форма



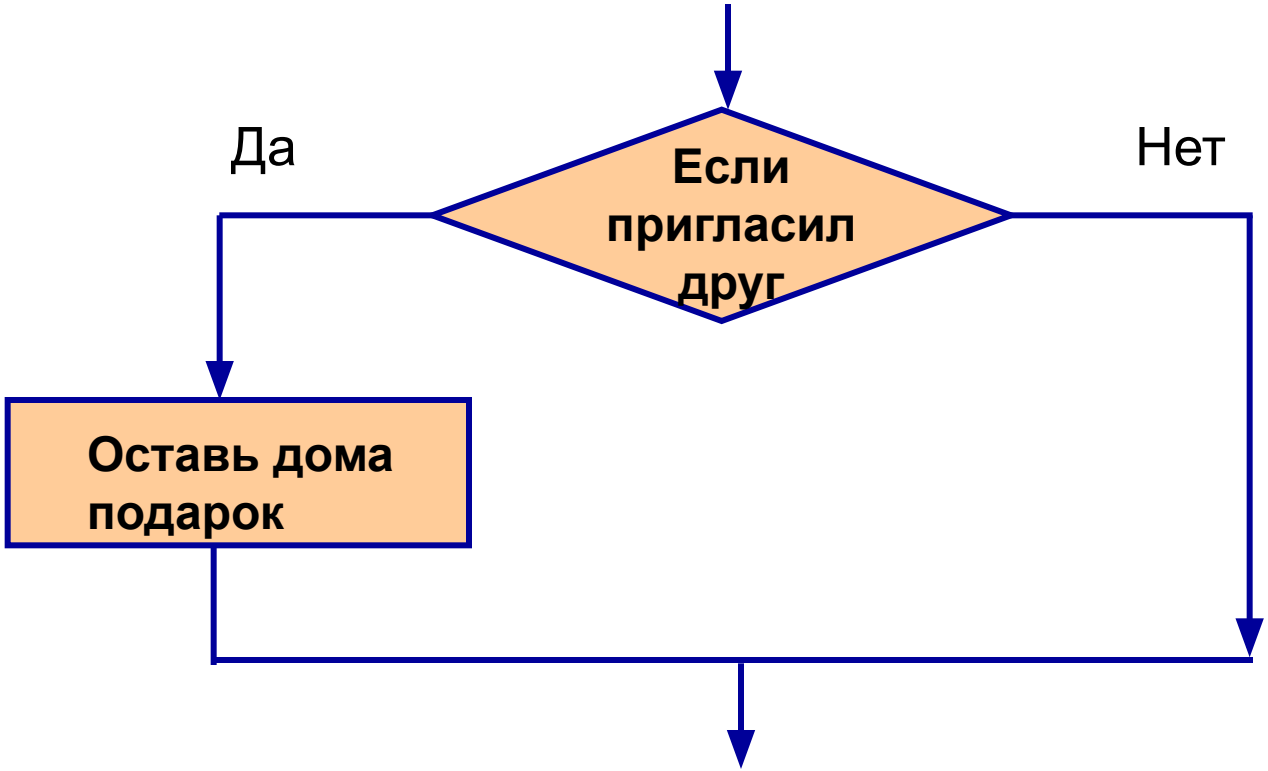
Неполная форма

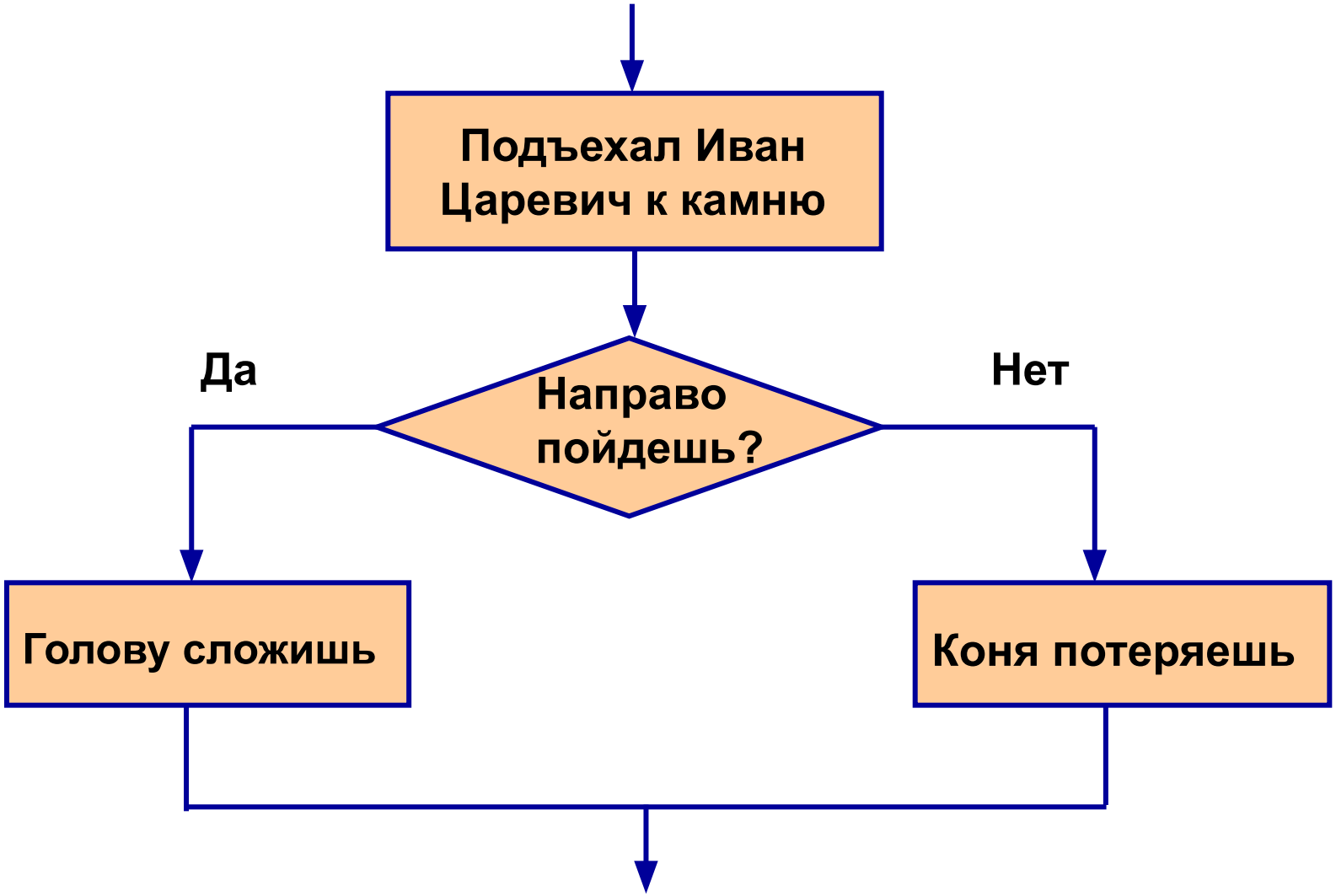


выбор



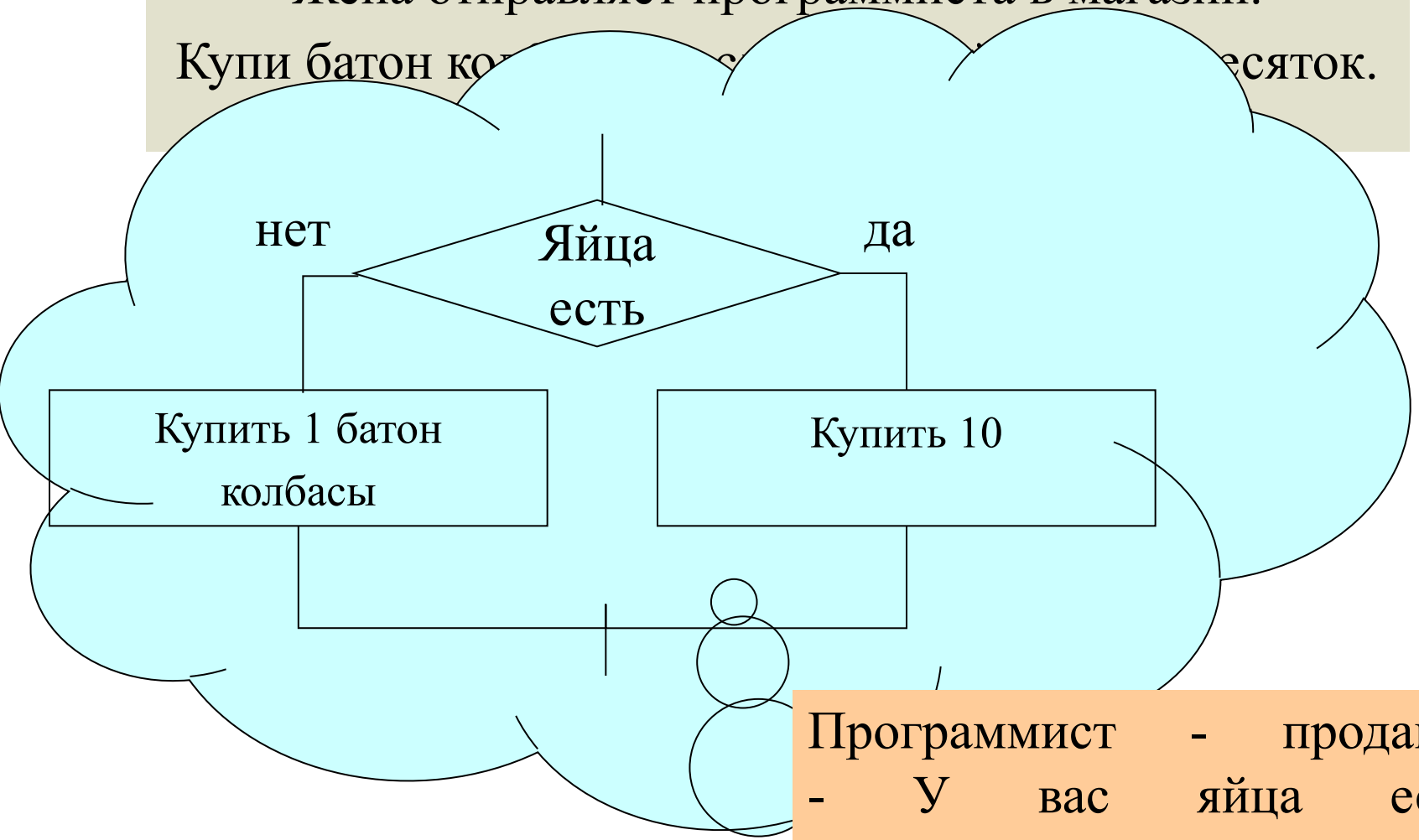
Если друг на день рожденья
Пригласил тебя к себе,
То оставь подарок дома –
Пригодится самому…




Подъехал Иван Царевич к камню
Да
Направо пойдешь?
Нет
Голову сложишь
Коня потеряешь

Жена отправляет программиста в магазин.

Купи батон ко… …есяток.



Программист - продавцу.
- У вас яйца есть?
- Есть!
- ОК. Мне 10 батонов колбасы.

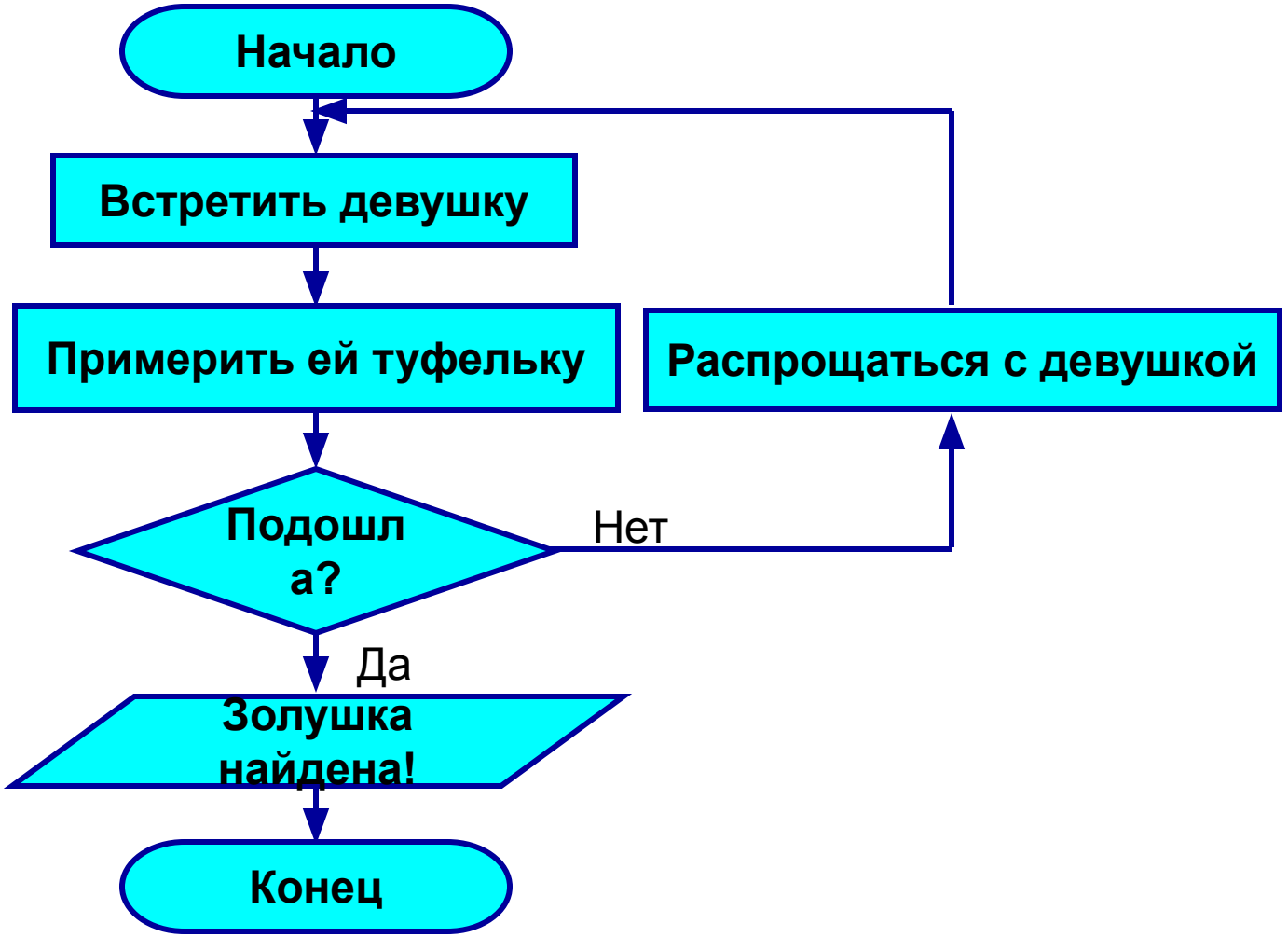
Циклический - алгоритм, в котором некоторая последовательность действий может выполняться несколько раз в зависимости от заданного условия.

В схеме циклический алгоритм представляется в виде типовой структуры **<u>цикл:</u>**


Цикл "Пока"
(цикл с предусловием)
Условие
Нет
Да
Действия


Цикл с постусловием
Действия
Да
Условие
Нет

Алгоритм поиска Золушки:



Итак, алгоритмы делятся на

- линейные
- разветвляющиеся
- циклические

(можно также выделить в отдельный тип смешанные).

Алгоритмы могут классифицироваться и по другому направлению.

- Комбинаторные алгоритмы:
  - ❖ Общие комбинаторные алгоритмы (например, генерация случайных чисел)
  - ❖ Алгоритмы на графах
  - ❖ Алгоритмы поиска
  - ❖ Алгоритмы сортировки
  - ❖ Алгоритмы слияния
  - ❖ Алгоритмы работы со строками

- Алгоритмы сжатия данных
- Криптографические алгоритмы
- Теоретико-числовые алгоритмы
- Цифровая обработка сигналов

И т.д.

Основные элементы языка программирования Delphi (в версиях 1-6 – Object Pascal)

Object Pascal — результат развития языка Турбо Паскаль, который, в свою очередь, развился из языка Паскаль.

Паскаль был создан …

Внимание, вопрос:

Кто был автором языка программирования Pascal?

A. Блез Паскаль

B. Билл Гейтс

C. Слава КПСС

D. Никлаус Вирт

E. Леди Ада Лавлейс

F. Леди Гага

Я м Виртом в 1968-69 годах.

Назван в честь выдающегося фра
физика, литератора и философа Блеза
первую в мире механическую маши
числа.

Дельфы — древнегреческий город (латинское написание — Delphi).

Оракул - предсказатель будущего, а также человек, все суждения которого признаются непреложной истиной, откровением.

Почему 10 декабря названо Днем программиста ?

Августа Ада Лавлейс – первый программист - родилась 10 декабря 1815 года. Она была единственной дочерью великого английского поэта Джорджа Гордона Байрона (1788 — 1824) и Аннабеллы Байрон, урождённой Милбэнк (1792 — 1860).

Алфавит языка.

Алфавит – совокупность допустимых символов:

- буквы – буквы латинского алфавита, а также знак подчеркивания (_);
- цифры 0..9;
- шестнадцатеричные цифры;
- разделители: исп-ся для отделения др. от друга идентификаторов, чисел, зарезервированных слов.

Можно исп-ть пробел, любой управляющий символ (коды 0.. 31), комментарий;

- специальные символы:
- знаки пунктуации ({}, =,:=,’ и т.д.);
- знаки операций (+, * и т.д.);
- зарезервированные слова (напр., begin, end и др.).

Идентификатор – имя любого объекта программы (переменной, константы, процедуры и др.).

- **Идентификатор может включать буквы латинского алфавита, цифры и символ подчеркивания.**
- **Идентификатор не может начинаться с цифры.**
- **Прописные и строчные буквы в идентификаторах не различаются.**

Т.е. напр., Vasja1, VASJA1 и VaSjA1 – это один и тот же идентификатор

Структура программы в консольном приложении.

Консоль — это монитор и клавиатура, рассматриваемые как единое устройство.

Консольное приложение — программа, предназначенная для работы в операционной системе MS-DOS (или в окне DOS), для которой устройством ввода является клавиатура, а устройством вывода — монитор, работающий в режиме отображения символьной информации (буквы, цифры и специальные знаки).

Консольные приложения удобны как иллюстрации при рассмотрении общих вопросов программирования, когда надо сосредоточиться на сути проблемы,

В программе могут быть следующие разделы:

- заголовок программы
- раздел объявления используемых модуле
- раздел объявления меток
- раздел описаний
- раздел объявления констант
- раздел объявления типов
- раздел объявления переменных
- раздел объявления процедур и функций
- тело программы или раздел операторов
  (обязательный раздел).

Заголовок состоит из зарезервированного слова ***Program*** и имени программы, завершается точкой с запятой. Имя программы может включать буквы, цифры и символ подчеркивания и не может начинаться с цифры.

Порядок размещения разделов произвольный, но

! в любом месте программы можно использовать лишь элементы , которые были определены ранее по тексту программы или являются стандартными элементами языка.

- Тело программы начинается словом ***Begin*** и заканчивается словом ***End*** с точкой, которая является признаком конца программы.
- В любом месте программы могут располагаться комментарии. Комментарии заключаются в скобки {} или в скобки вида (* *) и могут занимать произвольное число строк. Они игнорируются компилятором и служат для пояснения текста программы.

Пример.

Программа, вычисляющая произведение двух чисел.

```
Program Primer;                          {Заголовок программы}

  {$Apptype console}
 uses SysUtils, math; {раздел объявления используемых модулей}
var                  {раздел объявления переменных}
 x,y,p:real;
Begin     {тело программы}
  Write('Введите два числа'); readln(x,y);

   p:=x*y;
  writeln(' Произведение чисел равно ',p)

End.
```

Типы данных.

Под типом данных понимается множество допустимых значений этих данных, а также совокупность операций над ними.

Раздел объявления типов начинается зарезервированным словом *type,* после которого определяются вводимые типы.

Type

<имя типа1>=<определение типа1>;

<имя типа2>=<определение типа2>;

и т.д.

В Object Pascal можно выделить следующие типы данных:

- простые;
- структурированные;
- указатели;
- процедурные типы;
- объекты.

К простым типам относятся :

- целые;
- логический;
- символьный;
- перечисляемый;
- тип-диапазон;
- вещественные типы.

Целые

Тип	Диапазон значений	Размер в байтах
integer	-2147483648 .. 2147483647	4
byte	0..255	1

Вещественные

Тип	Диапазон значений	Размер в байтах
real	$5.0*10^{-324}$ ..$1.7*10^{308}$	8
double	$5.0*10^{-324}$ ..$1.7*10^{308}$	8
extended	$3.6*10^{-4951}$ ..$1.7*10^{4932}$	10

Символьный тип.

Обозначается словом **Char**.

Для размещения данных типа char требуется 1байт.

Значениями данных символьного типа могут являться любые символы из расширенного набора символов для ПЭВМ.

(Каждому символу приписывается целое число или код в диапазоне 0..255. Первая половина символов соответствует стандарту ASCII / American Standart Code for Information Interchange – американский стандартный код для обмена информацией/. Вторая половина символов с кодами 128..255 может меняться на ПЭВМ разных типов).

Логический тип.

Тип **Boolean** представляет собой тип данных, любой элемент которого может принимать только два значения: **True** (истина) или **False** (ложь).

Для размещения данных типа Boolean требуется 1 байт памяти

Константы.

Константами называются параметры программы, значения которых не меняются в процессе ее выполнения.

Существует 2 способа использования констант:

- непосредственное использование значения константы;
- использование идентификатора (имени) константы.

Задание констант именами осуществляется в разделе объявления констант, который начинается словом **Const.**

Const

<имя константы1>=<значение 1>;

<имя константы2>=<значение 2>; и т.д.

Имя константы формируется согласно основному правилу формирования идентификаторов(см. выше).

Напр., *Const Max=1345;*

x_2=10.5;

35r, f-47 , вася – недопустимые имена констант.

Константы могут быть целого, вещественного, символьного, логического и строкового типа.

Целые. В изображении целых к. только знак и цифры. (-45, 509, +35)

Вещественные. В своем изображении могут содержать знак, цифры, десятичную точку, показатель степени - символ E или e.

Сущ. две формы записи вещ. констант:

а) естественная 10.6 -0.001

б) экспоненциальная 0.107E+02 -0.1e-02

Строковые и символьные константы.

Строка символов(или строковая константа) – это последовательность любого, в том числе и равного нулю, количества символов из стандартного набора символов ПЭВМ, расположенных на одной строке и заключенного в апострофы. Значимое кол-во символов 126.

Строка, состоящая из одного символа, называется символьной константой.

Напр. ' '

' студент группы ТМ-11 '

' f= '

' h '

Переменные.

Переменные — элементы программы, значения которых могут изменяться в процессе ее выполнения.

Имя переменной придумывает программист.

Имя переменной формируется согласно основному правилу формирования идентификаторов (см. выше).

Желательно, чтобы имя переменной несло смысловую нагрузку.

Все используемые в программе переменные должны быть определены с указанием их типов.

Раздел объявления переменных выглядит сл. образом:

Var

<список переменных 1>:<тип 1>;

<список переменных 1>:<тип 1>; и т.д.

Напр.

var

x,summa:real;

priznak:boolean;

Переменные размещаются в оперативной памяти ЭВМ и имеют размер в соответствии с объявленным типом.

Операции.

В Object Pascal сущ. след. операции:

арифметические, логические, операции со строками, операции отношения, операция с битами информации, адресная операция @.

Арифметические операции (АО) применимы только к величинам целых и вещественных типов.

Существуют следующие **Арифметические операции** (расположим их в порядке убывания приоритета):

- / и *
- div (целочисленное деление)
- mod (остаток от деления целых чисел)
- + и -

Стандартные функции.

В языке П. существует ряд заранее разработанных подпрограмм-функций, которые можно использовать как готовые объекты.

Стандартные арифметические функции.

Функция в математике	Функция в Object Pascal	Пример
$\|x\|$	abs(x)	abs(x+5)
x^2	sqr(x)	sqr(x/y)

$\sqrt{x}$	sqrt(x)	sqrt(5*z)
Ln x	ln(x)	ln(y-4.5)
e^x	exp(x)	exp(f/3)
sin x	sin(x)	sin(5*x)
cos x	cos(x)	
arctg x	arctan(x)	
$\pi = 3.14159265$	Pi	

Аргумент функции всегда заключается в круглые скобки !

Аргумент ф-й sin и cos указывается в радианах.

$$\log_b a = \frac{\ln a}{\ln b}$$

$$tg\, x = \frac{\sin x}{\cos x}$$

$$\arcsin x = arctg \frac{x}{\sqrt{1 - x^2}}$$

$$\arccos x = arctg \frac{\sqrt{1 - x^2}}{x}$$

В Паскале нет операции возведения в степень. Поэтому, если степень простая, то можно поступать, исходя из определения степени. Напр., $x^5=(x^2)^2x$ sqr(sqr(x))*x

В более сложных случаях для x>0 можно воспользоваться формулой

$$x^y = e^{y \cdot \ln x}$$

Если к программе подключить модуль **Math**, добавив в нее строку программы

Uses math;

можно использовать следующие функции из этого модуля:

Функция в математике	Функция в программе	Пример
arcsin x	arcsin (x)	arcsin (x+5)
arccos x	Arccos(x)	Arccos(2* x)
x^a	Power(x,a)	Power(x+3,5*v)
tg x	tan(x)	tan(5+z)

Выражения.

Выражение – это синтаксическая единица языка, определяющая способ вычисления некоторого значения. Выражения формируются из констант, переменных, функций, знаков операций и круглых скобок.

Примеры арифметических выражений:

$$3{,}5 + \frac{\sqrt{x^2 - 2}}{\ln^2 x - \beta}$$

3.5+sqrt(x*x-2) / (ln(x) * ln(x)-b)

$$\frac{2\cos^2 x - 2{,}5}{\left|tgx^4 - 1{,}2\right|} + \varphi^{2{,}3}$$

(2 * sqr(cos(x))-2.5) / abs(sin(x*x*x*x) / cos(x*x*x*x)-1.2) +

exp(2.3 * ln(fi))

или при подключенном модуле **Math**

(2*sqr(cos(x))-2.5)/abs(tan(x*x*x*x)-1.2)+power(fi,2.3)