Российский государственный университет нефти и газа им. И.М. Губкина

Кафедра Информатики

Дисциплина: ***Программные комплексы общего назначения***

Преподаватель:

к.т.н., доцент
Коротаев
Александр Фёдорович

Этапы решения задач на компьютере

1. Постановка задачи и её математическое описание.
2. Выбор метода решения.
3. Разработка алгоритма решения задачи.
4. Разработка программы (сценария).
5. Отладка и тестирование программы.
6. Проведение расчётов.
7. Анализ полученных результатов и возможная модификация программы.

При решении конкретных задач некоторые из этапов могут отсутствовать или объединяться с другими этапами.

Этап 1. Постановка задачи и её математическое описание

Определяется:

- что мы хотим получить в результате решения задачи;
- какие для этого потребуются исходные данные;
- какие существуют зависимости и соотношения между выходными (зависимыми) и входными (независимыми) переменными и другими параметрами задачи;
- какие существуют ограничения на переменные и параметры задачи.

Анализируются возможные варианты математического описания и выбирается наиболее приемлемый из них.

Данный этап очень часто называют формализацией задачи или построением её **математической модели**.

Этап 2. Выбор метода решения

Хотя математическая модель и задает основные соотношения между величинами, входящими в математическое описание задачи, этих соотношений может быть много, они могут быть достаточно сложными.

Кроме того, есть задачи, в которых математическая модель не даёт явной зависимости выходных переменных от входных.

Простой пример: линейное уравнение

Математическая модель $ax+b=0$,

где a, b - параметры задачи (коэффициенты уравнения);

x - выходная переменная (корень уравнения).

Метод решения: $x = -b/a$, $a \neq 0$.

В общем случае необходимо найти подходящий известный метод или разработать новый метод, который может быть реализован на компьютере.

Этап 3. Разработка алгоритма решения задачи

Происходит переход от математических описаний к чёткой регламентации действий компьютера при выполнении вычислительного процесса. Последовательность этих действий и задает алгоритм решения задачи, т.е. правила, по которым происходит переработка исходных данных в результат решения задачи.

Формально алгоритм можно определить как точное, полное и однозначное описание последовательности действий, направленных на решение поставленной задачи.

Процесс разработки и описания алгоритма решения задачи называют алгоритмизацией.

Этап 4. Разработка программы (сценария)

По существу, это просто кодирование разработанного алгоритма (изложение его на некотором языке, который может быть «понят» компьютером).

Этап 5. Отладка и тестирование программы

Обнаруживаются и исправляются ошибки, допущенные на этапах алгоритмизации и разработки программы, а возможно, и на более ранних этапах. Проверяется правильность работы всех ветвей разветвлённой программы.

Этап 6. Проведение расчётов

Запуск, программы, ввод исходных данных, получение результата.

Этап 7. Анализ полученных результатов

Результаты проверяются на достоверность, оцениваются полученные значения, скорость и точность вычислений.

Возможное внесение изменений в программу, алгоритм, метод решения и даже в постановку задачи.

Алгоритмы и способы их описания

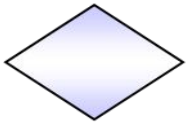
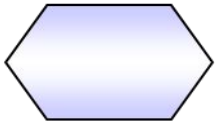
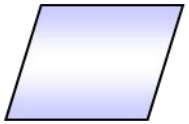
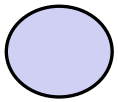
Алгоритм как строго заданная последовательность действий может быть описан различными способами:

- ✔ словесное описание (на естественном языке, например, на русском);
- ✔ графическое описание (в виде схем алгоритмов);
- ✔ на алгоритмическом языке (языке программирования).

Алгоритм не обязательно должен задавать некоторый вычислительный процесс. С его помощью можно задать любую другую строго определенную логическую последовательность.

Например, сборка автомобиля на конвейере.

Условные обозначения в блок-схемах

Название блока	Обозначение	Назначение блока
Процесс		Обработка данных (вычисления)
Решение		Ветвление
Предопределенный процесс		Вызов функции или подпрограммы
Подготовка		Описание цикла с параметром
Данные		Операции ввода/вывода
Терминатор		Начало или завершение
Соединитель		Маркировка разрывов линий

Решить линейное уравнение

$$ax+b=0,$$

где коэффициенты **a, b** – заданы и могут быть любыми числами

Метод решения:

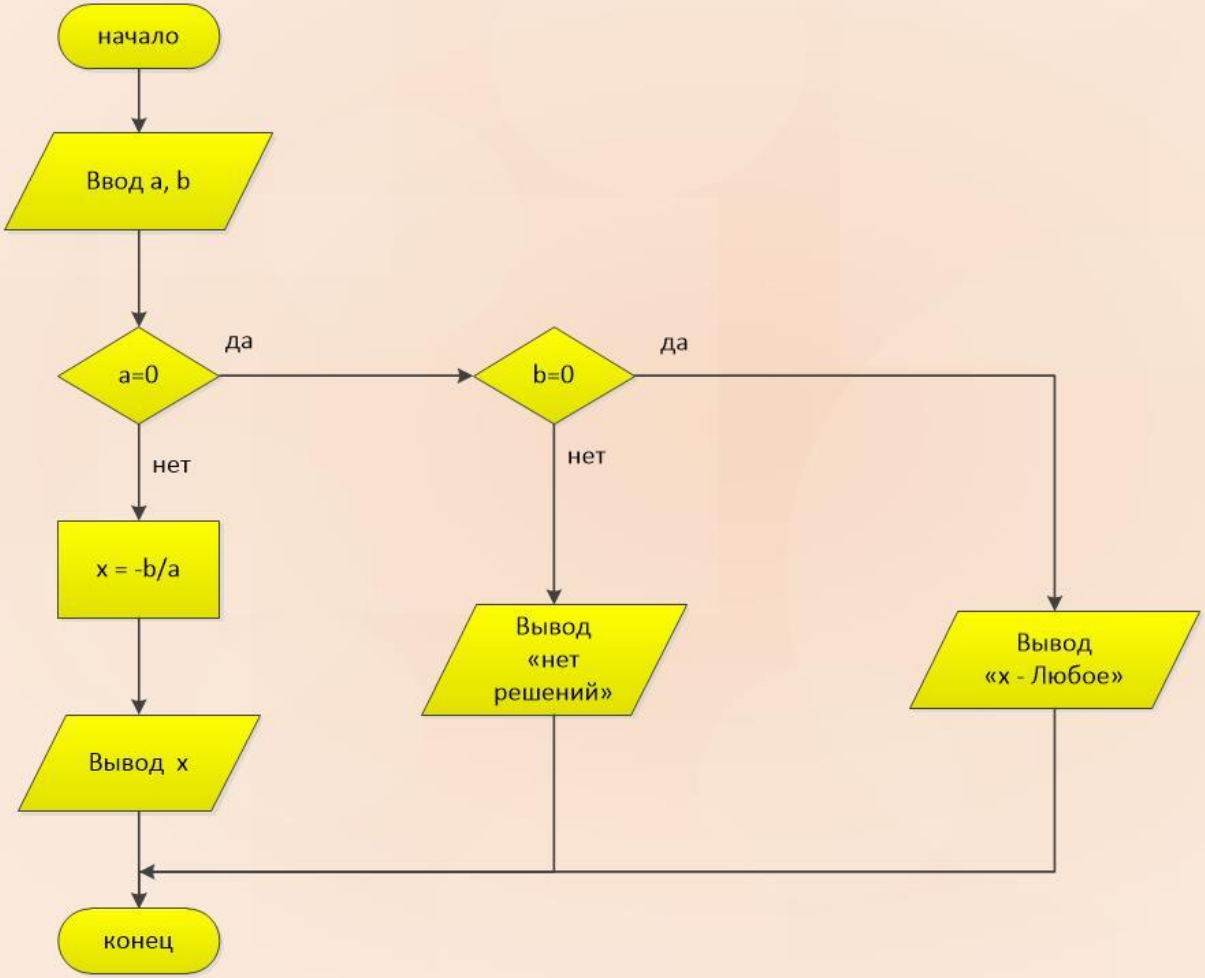
$$x = -b/a, \quad a \neq 0.$$

При **a = 0** :

если **b = 0** , то **x – любое**

если **b ≠ 0** , то **решения нет**

Блок-схема алгоритма решения линейного уравнения



Управляющие структуры языка MatLab

Теоретически доказано, что любые программы можно написать, используя всего 3 управляющие структуры

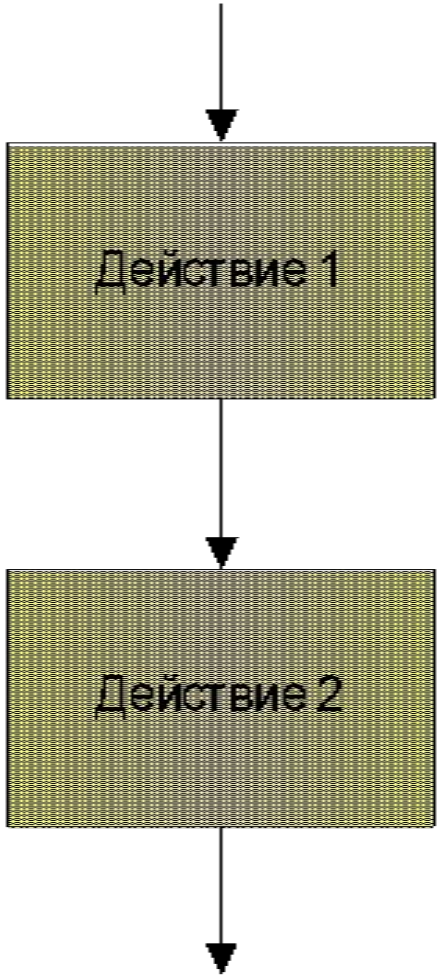
Следование - последовательность операторов (групп операторов), выполняемых последовательно друг за другом;

Выбор (Ветвление) - управляющая структура, которая разветвляет процесс на 2 или несколько направлений в зависимости от выполнения заданного условия;

Повторение (цикл) – оператор или группа операторов может выполняться многократно, до тех пор пока соблюдается заданное условие.

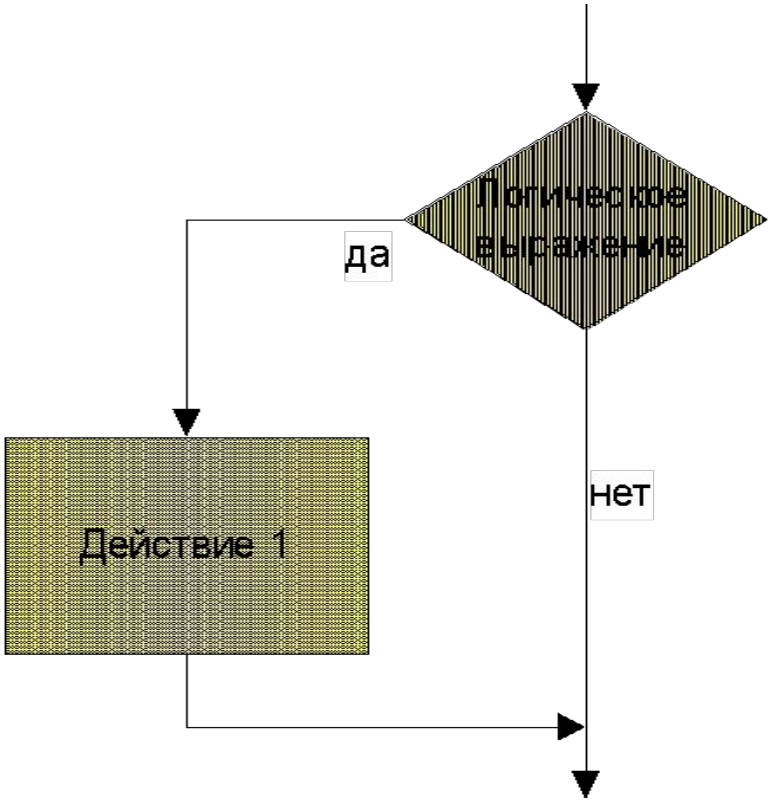
Базовые структуры алгоритмов

Следование – последовательное выполнение действий



Базовые структуры алгоритмов

Ветвление (вариант 1) – если - то



if end

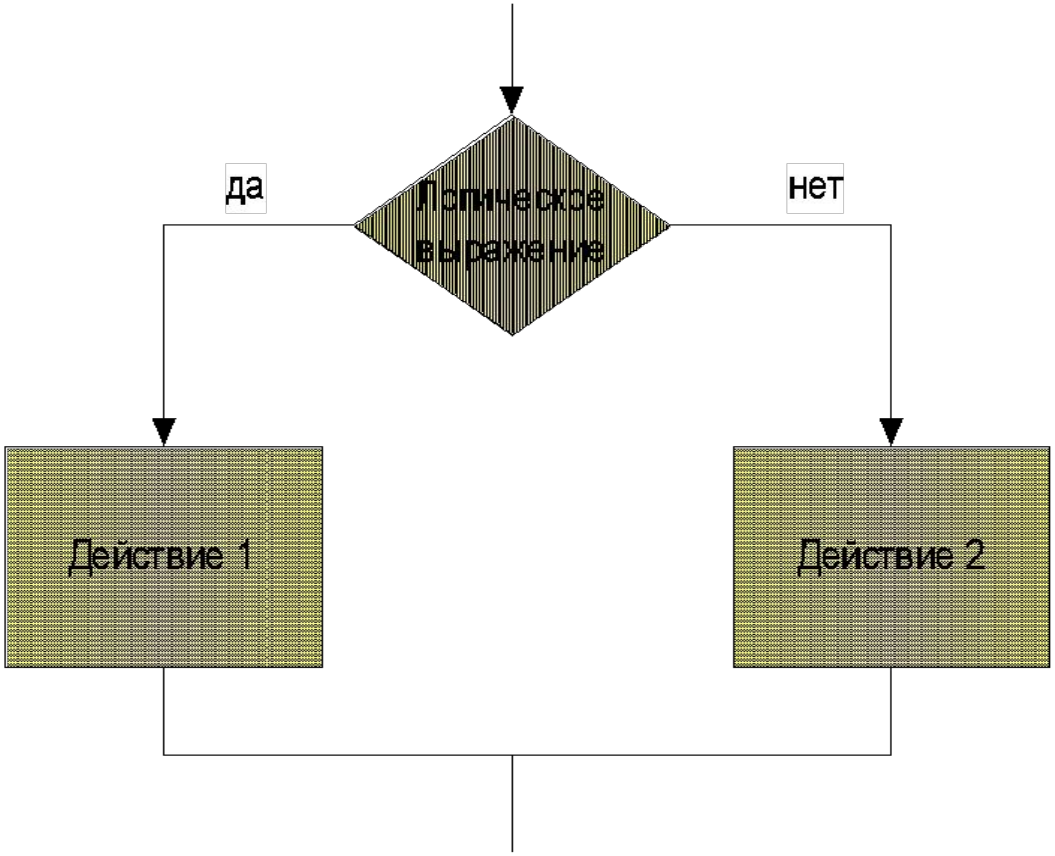
if логическое выражение

Инструкции, выполняемые, когда логическое выражение ***true***

end

Базовые структуры алгоритмов

Ветвление (вариант 2) – если – то - иначе



if else end

if логическое выражение

Инструкции, выполняемые, когда логическое выражение ***true***

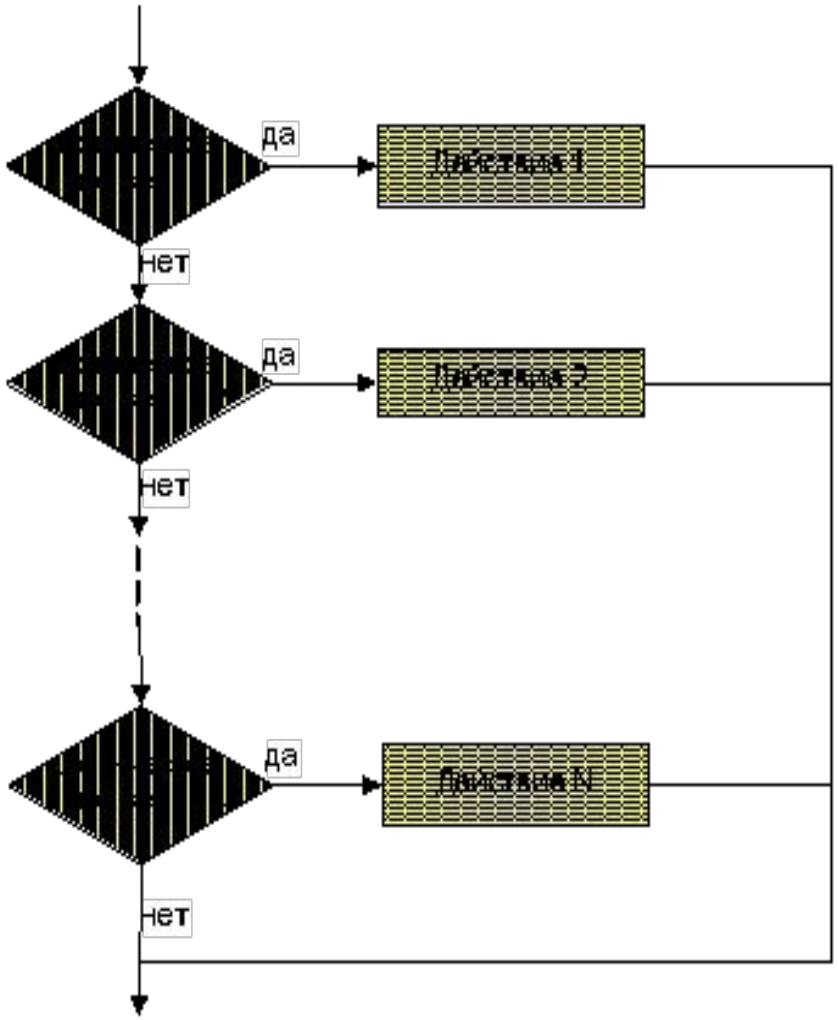
else

Инструкции, выполняемые, когда логическое выражение ***false***

end

Базовые структуры алгоритмов

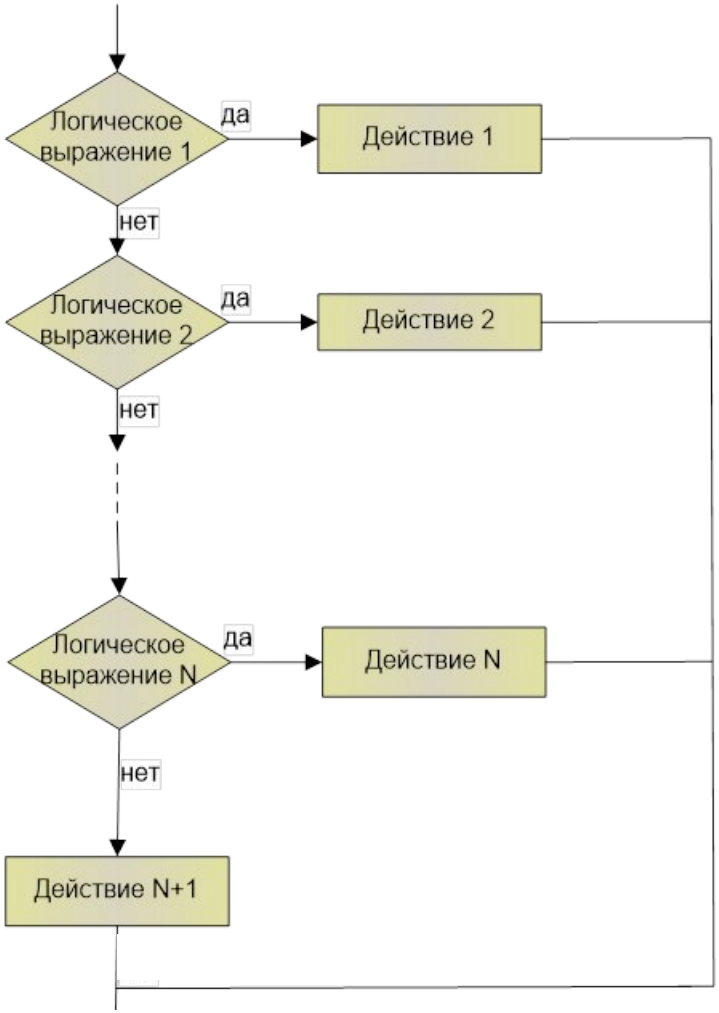
Ветвление (вариант 3) – выбор




```
switch end
switch  выражение
   case значение1
     инструкция1
   case значение2
     инструкция 2
……
end
```

Базовые структуры алгоритмов

Ветвление (вариант 4) – выбор - иначе



switch otherwise end

```
switch  выражение
   case значение1
      инструкция1
   case значение2
      инструкция 2
……
otherwise
     инструкция N+1
end
```

Базовые структуры алгоритмов

Повторение – цикл с условием

while end

while выражение

тело цикла

end

Базовые структуры алгоритмов

Повторение – цикл с параметром

for end

for var=a1:a2:a3

тело цикла

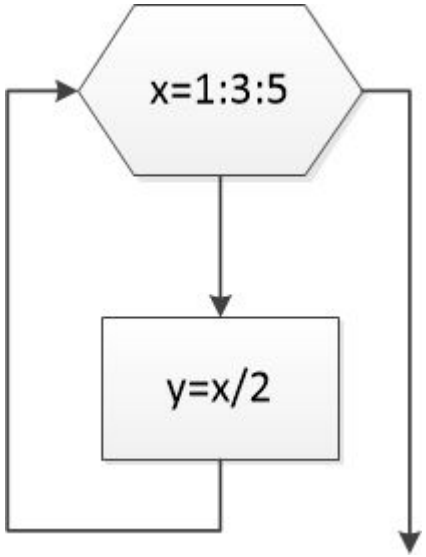
end

var – параметр цикла
a1 – начальное значение параметра
a2 – шаг изменения параметра
a3 – конечное значение параметра

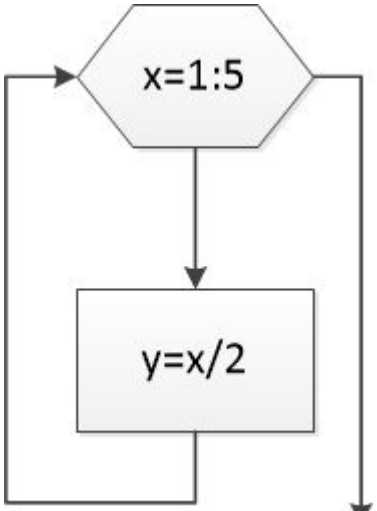
Если **a2** опущено, шаг равен **1**

Оператор цикла for

```
>> for x=1:3:5   y=x/2
end
y =    0.5000
y =    2
```




```
>> for x=1:5   y=x/2
end
y =    0.5000
y =    1
y =    1.5000
y =    2
y =    2.5000
```



Если инструкций несколько, они разделяются , или ;

Организации диалога в MatLab

Функция **input** позволяет вывести в командном окне запрос пользователю и получить на него ответ
x=input('запрос')
В ответ на запрос пользователь может ввести с клавиатуры значение или выражение.

Функция **disp(выражение)**
служит для вывода в командное окно результатов вычислений или текстовых сообщений. При её использовании результат ничему не присваивается. Входным аргументом может быть массив, выражение, текстовая строка, заключённая в апострофы

Пример с расширенной формой структуры выбора

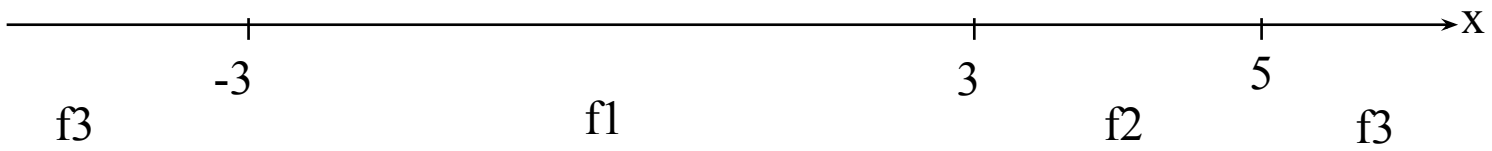
```
t=input('Введите температуру:   ' )
if(t<0)
   disp ('Мороз')
elseif  (t<10)
   disp('Прохладно')
elseif  (t<25)
   disp('Тепло')
else
   disp('Жарко')
end
```

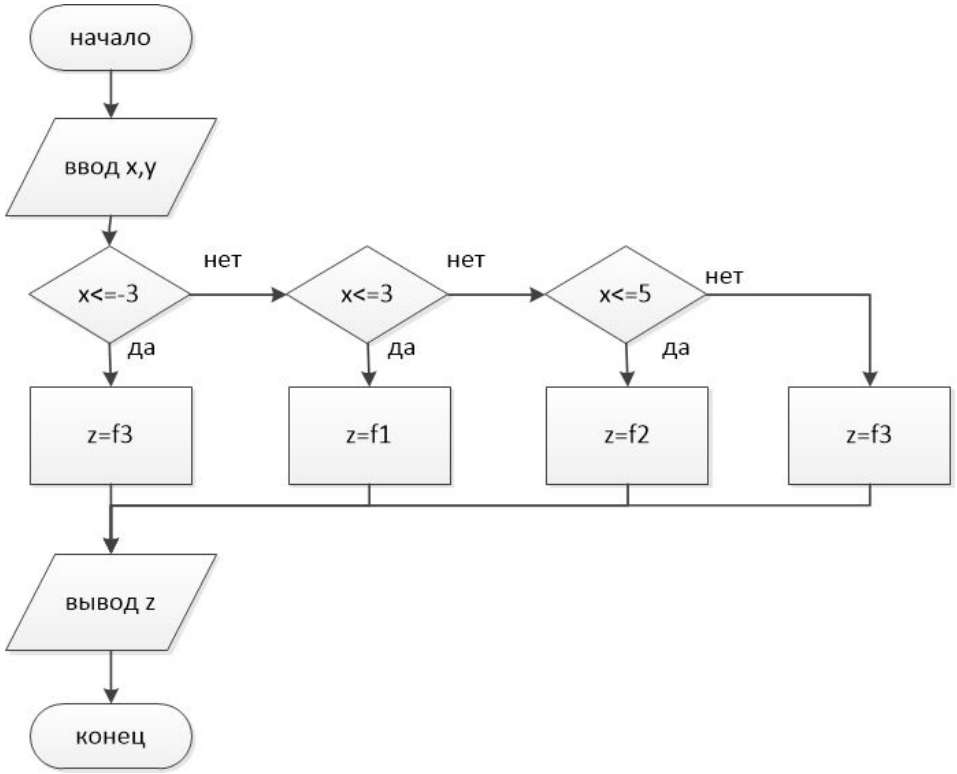
Пример с структурой множественного выбора

```
a=input('месяц?   ');
switch a
case   ('декабрь', 'январь','февраль')
       disp('зима')
case   ('март', 'апрель','май')
       disp('весна')
case   ('июнь', 'июль','август')
       disp('лето')
case   ('сентябрь', 'октябрь','ноябрь')
       disp('осень')
otherwise
      disp('неизвестное время года')
   end
```

Пример к лаб. раб. №1

$$Z=\begin{cases}\dfrac{Log_3(x-4y)}{\sqrt{2x+4}}, & при\ \ -3<x\le 3,\\ \dfrac{e^{x-1}}{\left|(x^2-4y)\right|}+\sin^2(3x-\pi/3), & при\ \ 3<x\le 5,\\ tg(\pi/5-2x)+(x-y)^3, & в\ остальных\ случаях\end{cases}$$






```
x=input('введите x=');
y=input('введите y=');
if x<=-3    z=f3(x,y)
   elseif x<=3  z=f1(x,y)
      elseif x<=5   z=f2(x,y)
else z=f3(x,y)
end
```

..\К лаб 1\вариант\L1_3f.m

Элементы алгебры логики

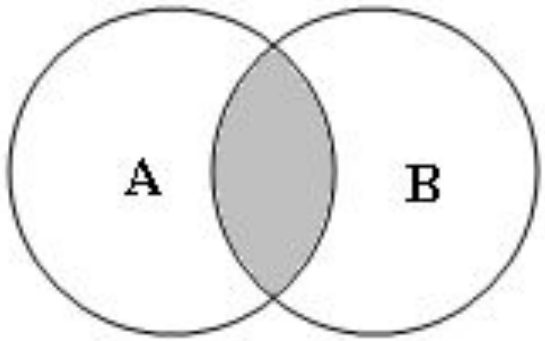
КОНЪЮНКЦИЯ (логическое умножение)

- в естественном языке соответствует союзу **и**
- в алгебре логики обозначается **&** или **∧**
- в языках программирования - **and**
- в MatLab - **&**

Таблица истинности

A	*B*	*A&B*
0	0	0
0	1	0
1	0	0
1	1	1

Диаграмма Венна



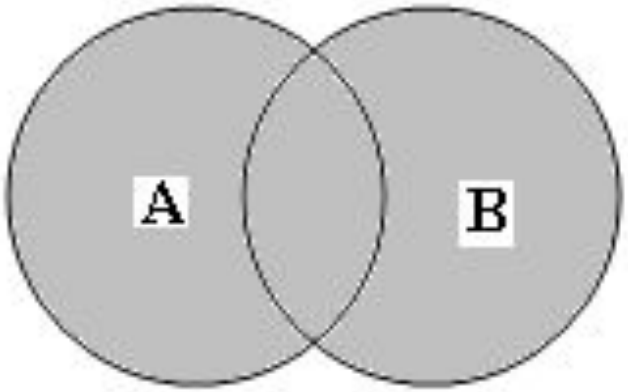
ДИЗЪЮНКЦИЯ (логическое сложение)

- в естественном языке соответствует союзу **или**
- в алгебре логики обозначается $\vee$
- в языках программирования - **or**
- в MatLab - |

Таблица истинности

A	*B*	AVB
0	0	0
0	1	1
1	0	1
1	1	1

Диаграмма Венна



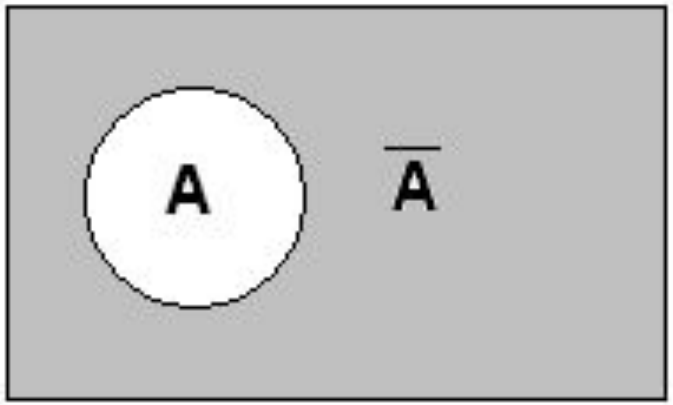
ИНВЕРСИЯ (отрицание)

- в естественном языке соответствует частице **не**
- в алгебре логики обозначается $\overline{A}$
- в языках программирования - **not**
- в MatLab - ~

Таблица истинности

A	$\overline{A}$
0	1
1	0

Диаграмма Венна



ИМПЛИКАЦИЯ (логическое следование)

- в естественном языке соответствует обороту
  если ..., то ...
- в алгебре логики обозначается **=>**

Таблица истинности

A	*B*	*A*=>B
0	0	1
0	1	1
1	0	0
1	1	1

ЭКВИВАЛЕНЦИЯ (равнозначность)

- в естественном языке соответствует обороте речи **тогда и только тогда**
- в алгебре логики обозначается **<=>**

Таблица истинности

A	***B***	***A<=>B***
0	0	1
0	1	0
1	0	0
1	1	1

Приоритеты в логических операторах

1) ~ — НЕ 2) **&** — И 3) | — ИЛИ

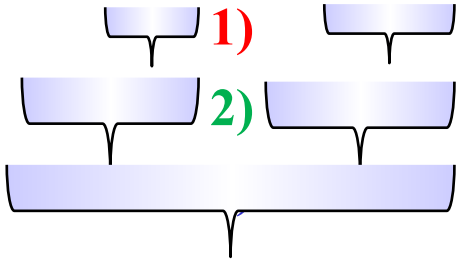
Операторы отношения

< > >= <= == ~=

Приоритеты: **1)** арифметические
2) отношения
3) логические

>> x>3&x<7 соответствует 3 < x < 7

>> x>3-y&x<5+z



Закон общей инверсии (законы де Моргана)

- *Для логического сложения:*

$$\overline{A \vee B} = \overline{A} \,\&\, \overline{B}$$

- *Для логического умножения:*

$$\overline{A \,\&\, B} = \overline{A} \vee \overline{B}$$