Тема 4.4
Программирование на языке MATLAB

Вопросы для изучения

4.14 Программирование разветвляющих алгоритмов
4.15 Операторы условного перехода и выбора
4.16 Логические операции и выражения

4.14 Программирование разветвляющих алгоритмов

Разветвляющийся алгоритм – алгоритм, содержащий хотя бы одно условие, в результате проверки которого ЭВМ обеспечивает переход на один из двух возможных шагов (рисунок 2.3)

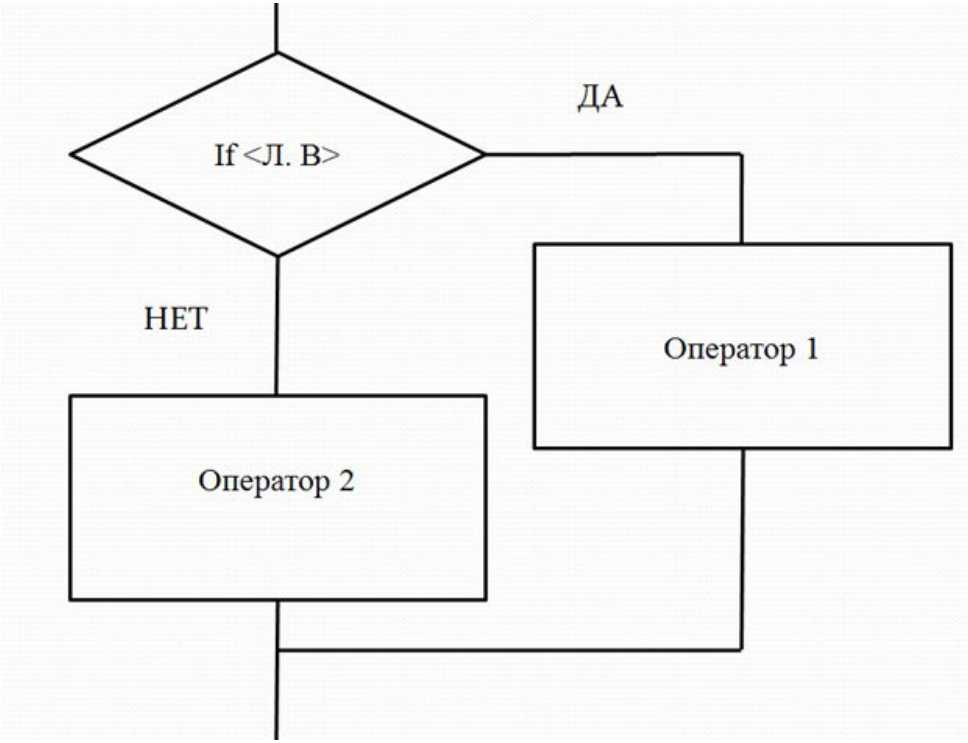


Рисунок 4.1- Разветвляющийся алгоритм

С помощью разветвляющих алгоритмов можно реализовывать логику выполнения операций и создавать повторяющиеся (итерационные, рекуррентные) вычисления.

4.15 Операторы условного перехода и выбора

Для того чтобы иметь возможность реализовать логику в программе используются условные операторы, достигая которых программа делает выбор по какому из возможных направлений двигаться дальше.

В м-языке используют:

- условный оператор if;
- оператор переключения switch.

которые позволяют создать гибкий разветвляющийся алгоритм, при выполнении определенных условий выполняется соответствующий блок операторов или команд MATLAB.

Условный оператор if.

Вариант 1. Полная форма 1

if Условие 1
Инструкции_1

elseif Условие 2
Инструкции_2

else
Инструкции_3

end

где Условие – логическое выражение принимающее значение «истина» или «ложь»

Инструкции – операторы и функции.

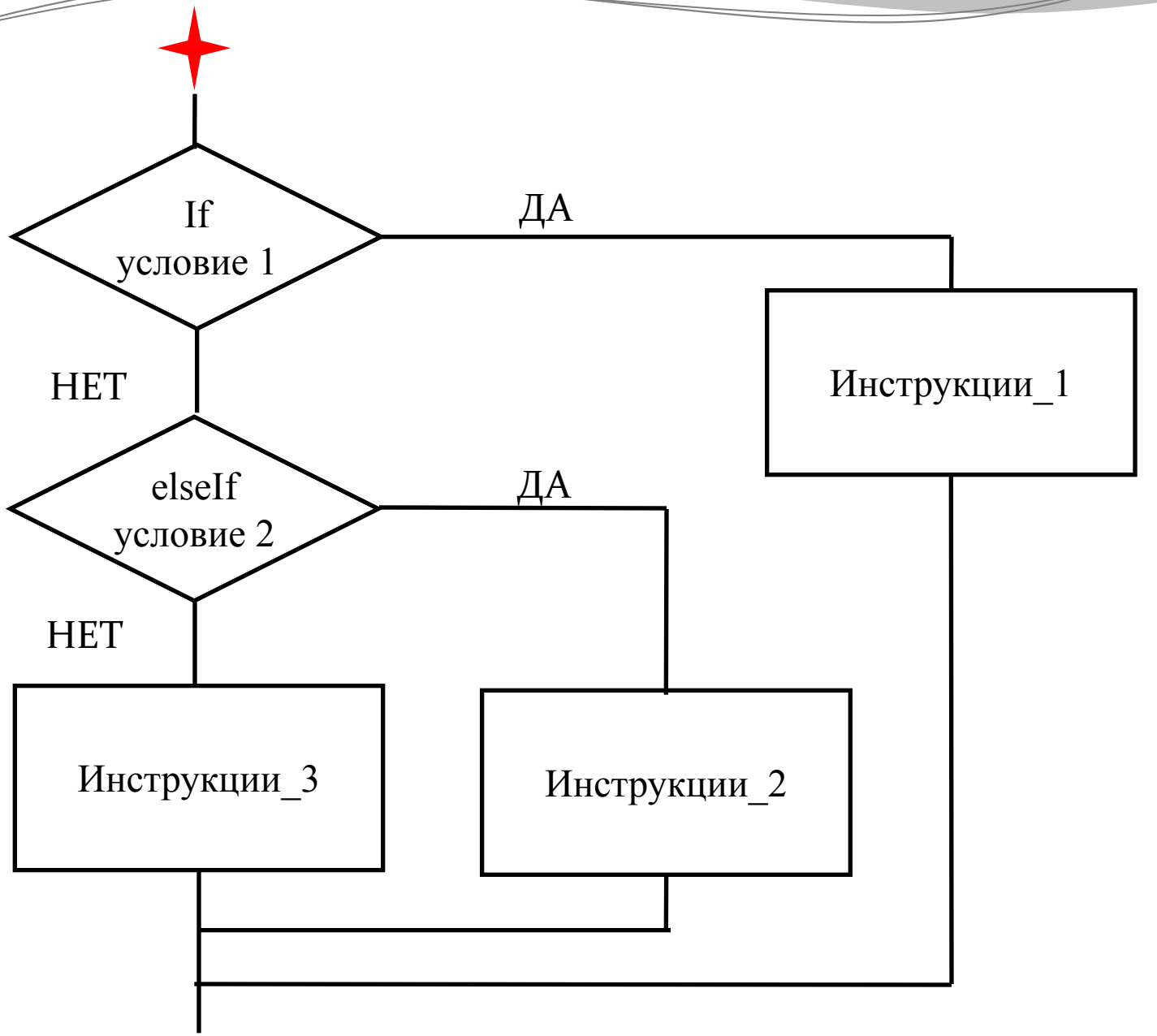
Работа:

- если Условие 1 возвращает логическое значение «истина», выполняются Инструкции_1, и управление передается оператору следующему за оператором if,

- если Условие 1 возвращает логическое значение «ложь», то проверяется Условие 2:

- если Условие 2 возвращает логическое значение «истина», выполняются Инструкции_2, и управление передается оператору следующему за оператором if,

- если Условие 2 возвращает логическое значение «ложь», выполняются Инструкции_3, и управление передается оператору следующему за оператором if,


If
условие 1
ДА
Инструкции_1
НЕТ
elseIf
условие 2
ДА
НЕТ
Инструкции_3
Инструкции_2

Вариант 2. Полная форма 2

If Условие

Инструкции_1 **else**

Инструкции_2

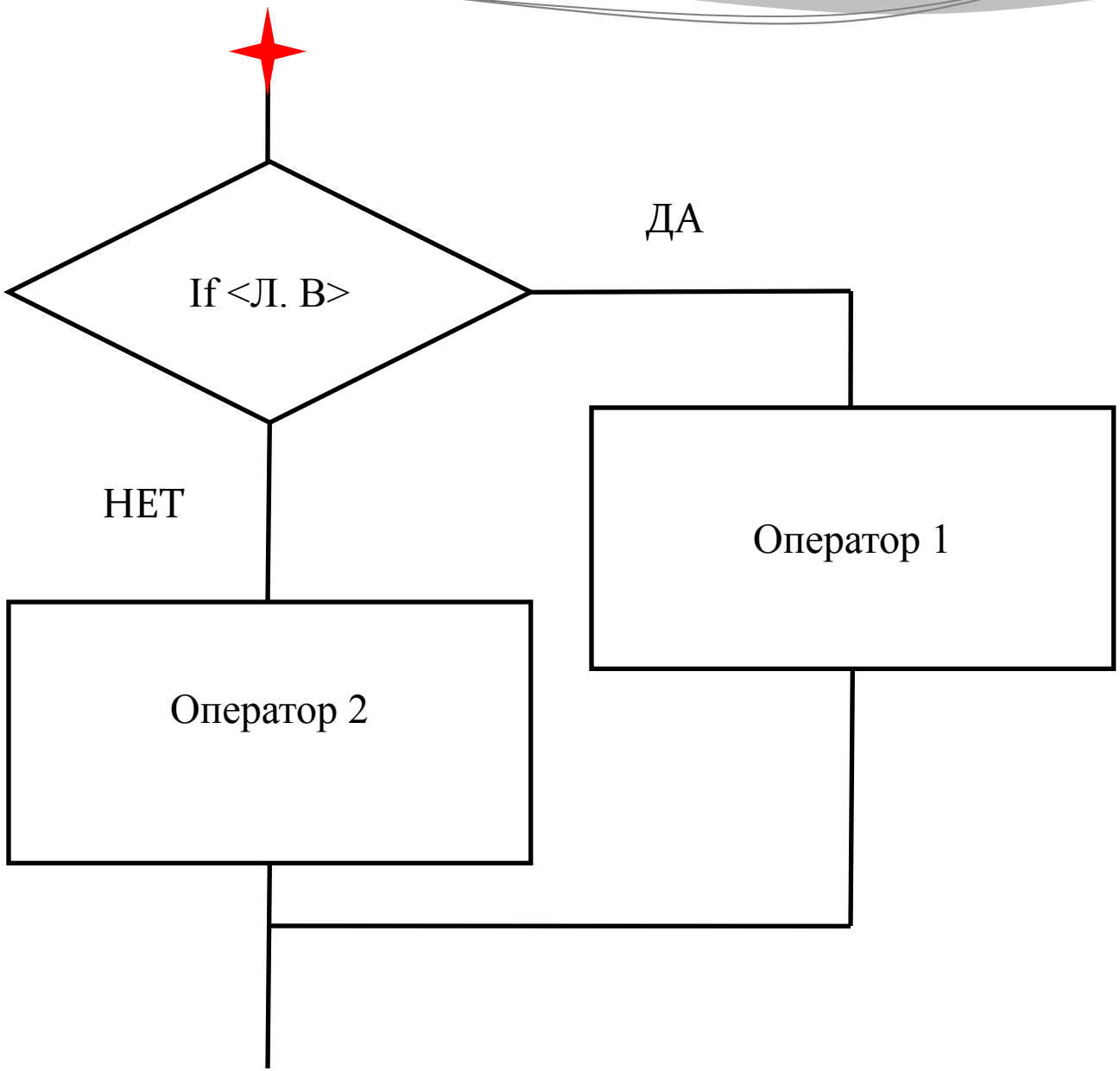
end

Работа :

- если Условие возвращает логическое значение «истина», выполняются Инструкции_1 и управление передается оператору следующему за оператором if.

- если возвращает логическое значение «ложь», выполняются Инструкции_2 и управление передается оператору следующему за оператором if.

```
x = 5;

if      x > 0
        disp(1);      % выполняется, если x > 0
elseif  x < 0
        disp(-1);     % выполняется, если x < 0
else
        disp(0);      % выполняется, если x = 0
end
```


If <Л. В>
ДА
НЕТ
Оператор 1
Оператор 2

Вариант 3. Сокращенная форма

if Условие

Инструкции

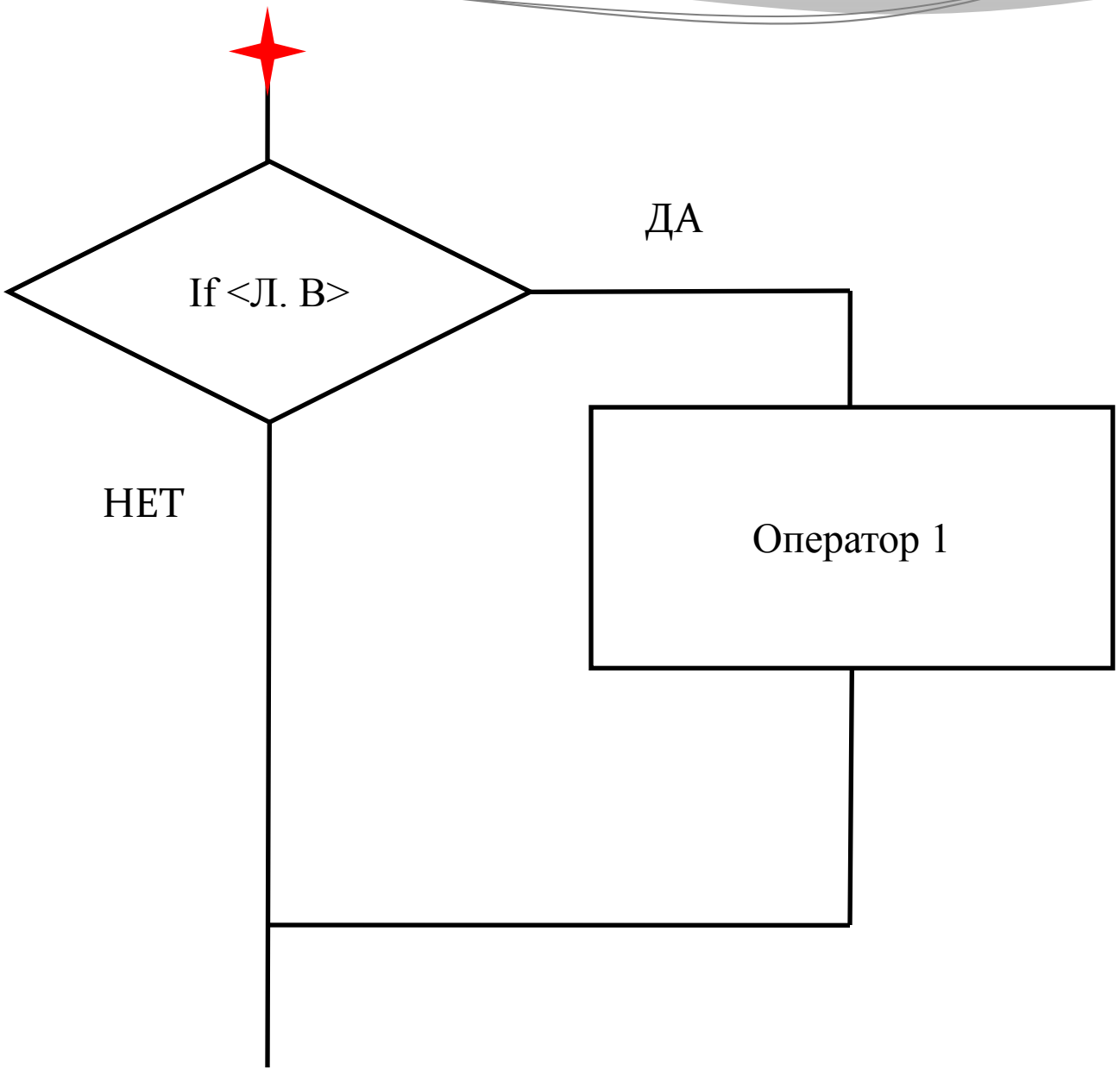
end

Работа:

- если Условие возвращает логическое значение «истина», выполняются Инструкции, составляющие тело структуры if...end, и управление передается оператору следующему за оператором if.

- если Условие не выполняется дает логическое «ложь», то Инструкции также не выполняются, а управление передается оператору следующему за оператором if.

```
x = 1;

if x >= 0 & x <= 2

        disp('x принадлежит диапазону от 0 до 2');

else

        disp('x  не принадлежит диапазону от 0 до 2');

end
```


If <Л. В>
ДА
НЕТ
Оператор 1

В общем случае применение этих структур достаточно очевидное. Приведем только один общий пример:

```
% пример использования структуры if-elseif-else

if (a ==0)
    disp('a- ноль')       elseif a==1

    disp('a- единица') elseif a>=2

     disp('a- двойка или больше')  else

    disp('a меньше двух, но не ноль и не единица')

end
```

```
if (a ==0)
        disp('a- ноль')       elseif a==1
        disp('a- единица') elseif a>=2
        disp('a- двойка или больше')  else
        disp('a меньше двух, но не ноль и не единица')
end
```

Оператор выбора Switch

Для осуществления множественного выбора (или ветвления) используется конструкция с переключателем типа switch.

Вариант 1. Полная форма

switch Выражение селектор

case список констант 1, Список инструкций_1

case список констант 2, Список инструкций_2

. . .

case список констант n, Список инструкций_n

otherwise, Список инструкций_N+1

end

Работа:

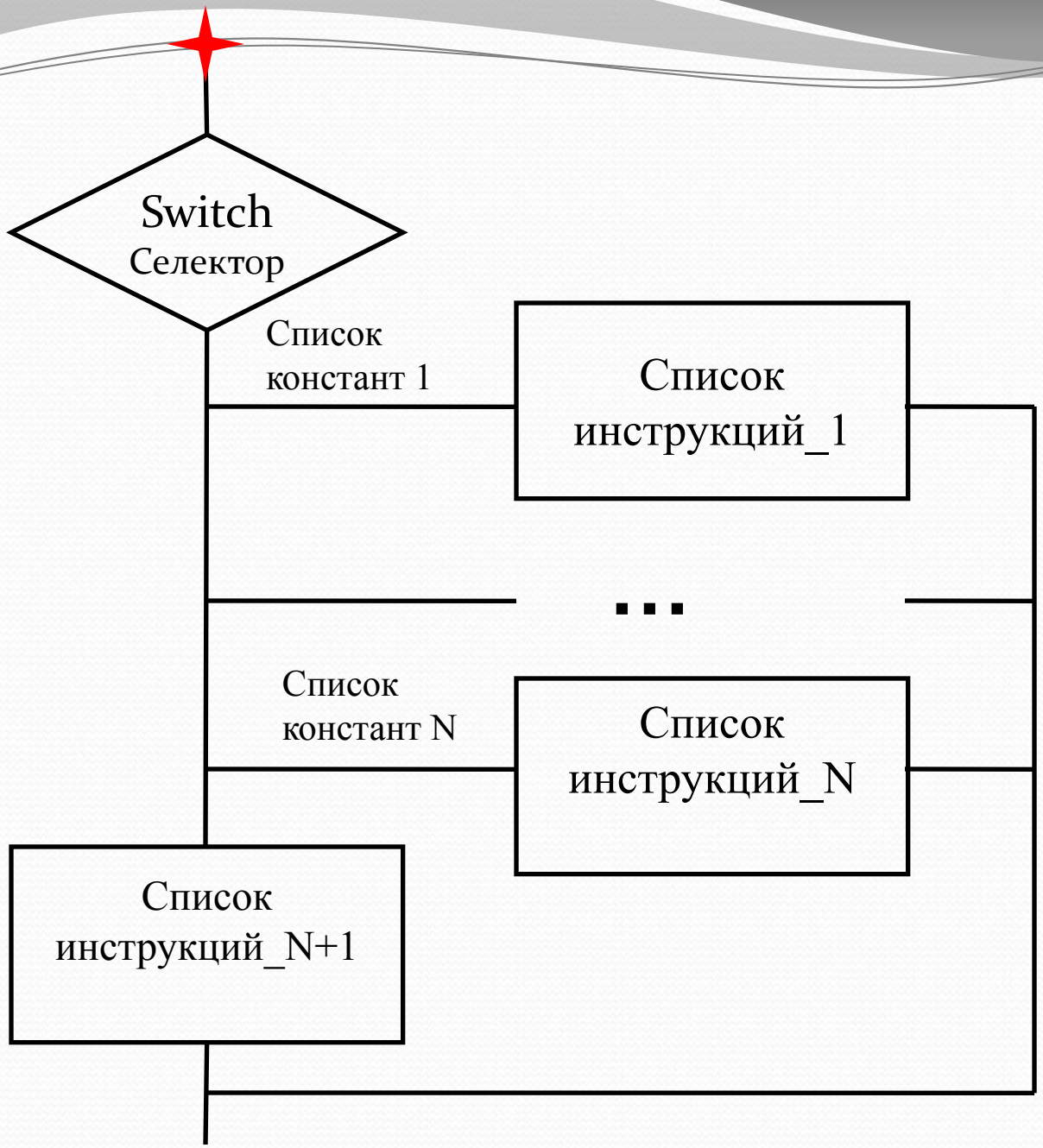
- вычисляется выражение селектор;

- последовательно проверяется совпадение значения селектора со значениями списка констант:

- если совпадение с каким либо списком констант есть, то выполняются инструкции соответствующие этому списку констант, при этом следующие далее списки констант не проверяются и управление передается оператору следующему за switch;

- если совпадений ни с одним списком констант нет, то выполняются инструкции следующие за словом otherwise и управление передается оператору следующему за switch.

Список констант задается отдельными значениями констант либо диапазонами констант (см. пример)


Switch
Селектор
Список
констант 1
Список
инструкций_1
...
Список
константN
Список
инструкций_N
Список
инструкций_N+1

Вариант 2. Сокращенная форма

```
switch Выражение селектор
    case список констант 1, Список инструкций_1
    case список констант 2, Список инструкций_2
    . . .
    case список констант n, Список инструкций_n
end
```

Работа:

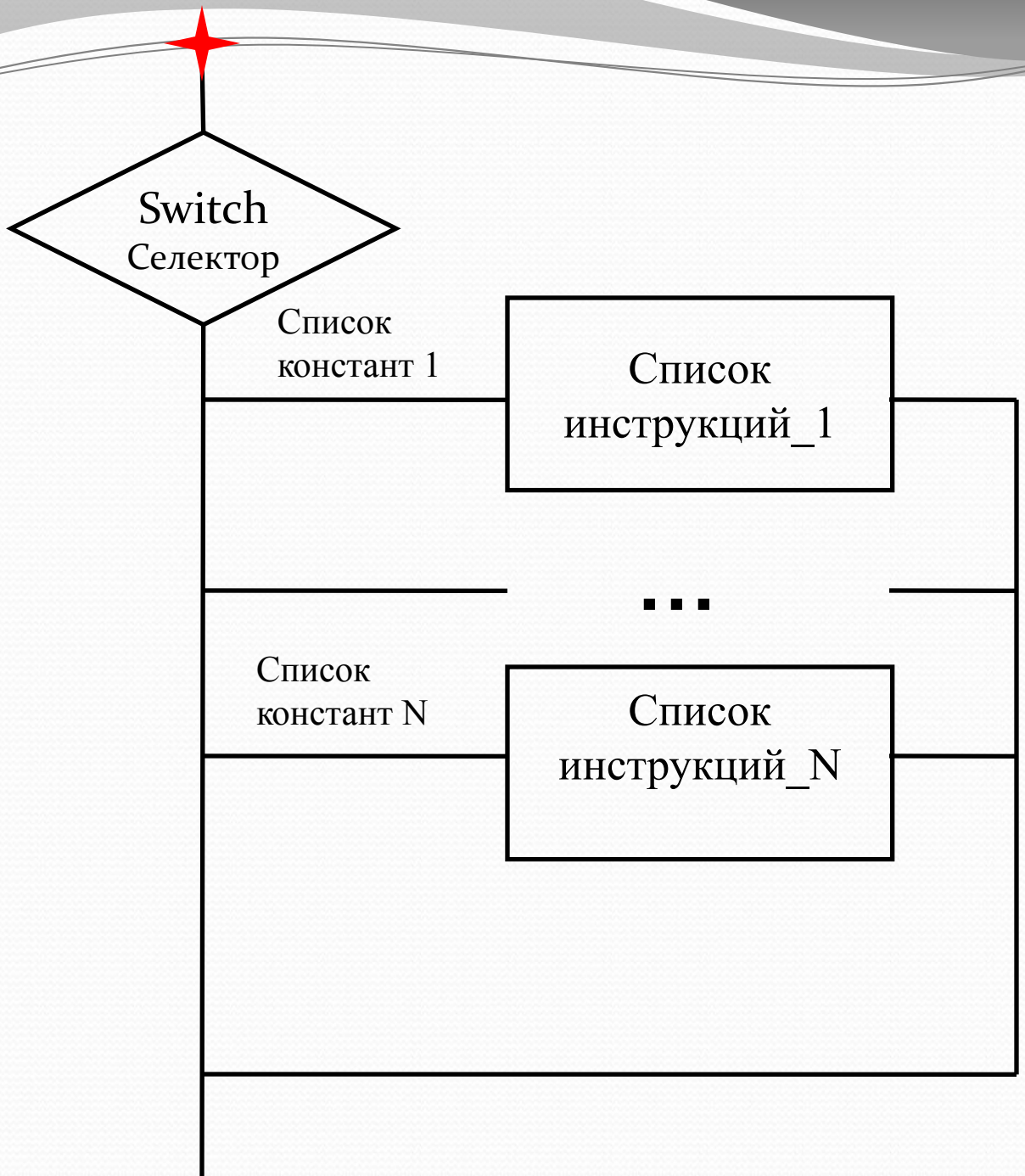
- вычисляется выражение селектор;

- последовательно проверяется совпадение значения селектора со значениями списка констант:

- если совпадение с каким либо списком констант есть, то выполняются инструкции соответствующие этому списку констант, при этом следующие далее списки констант не проверяются и управление передается оператору следующему за switch;

- если совпадений ни с одним списком констант нет, то управление передается оператору следующему за switch.

Константы выбора могут быть объединены в множества с помощью {}, напримар, {5, 7, 8, 4}.


Switch
Селектор
Список
констант 1
Список
инструкций_1
...
Список
констант N
Список
инструкций_N

```
switch n

        case {10,9},    disp (‘Отлично’),
        case {8,7,6},   disp (‘Хорошо’),
        case {5,4},     disp (‘Удовлетворительно’),
        case {3,2},     disp (‘Плохо’)
        case 1:         disp (‘Все пропало’)

    otherwise           disp (‘Неверная оценка’)

end
```

```
ch='c';

switch ch

    case 'a', ch='A';
    case 'b', ch='B';
    case 'c', ch='C';
    case 'd', ch='D';
    case 'e', ch='E';
    ...
    case 'z', ch='Z';

end

disp(ch);
```

4.16 Логические операции и выражения

В качестве условий в операторе if используют логические выражения которые строятся из арифметических выражений, операндов (значений, констант, переменных, функций) логических операций и отношений.

Операторы отношения служат для сравнения двух величин, векторов или матриц, все операторы отношения имеют две сравниваемые величины и записываются, как показано в таблице знаками или комбинациями знаков

Функция	Оператор (синтаксис)
Равно	= = (x = = y)
Не равно	~ = (x ~ = y)
Меньше	< (x < y)
Больше	> (x > y)
Меньше или равно	< = (x < = y)
Больше или равно	> = (x > = y)

Данные операторы выполняют поэлементное сравнение векторов или матриц одинакового размера и логическое выражение принимает значение 1 (True), если элементы идентичны, и значение 0 (False) в противном случае.

Логические операторы служат для реализации поэлементных логических операций над элементами одинаковых по размеру массивов согласно таблице 4.2.

Таблица 4.2 – Логические операторы

Функция	Оператор (синтаксис)
Логическое И	**&; *and* (*and* (*a*, *b*))**
Логическое ИЛИ	**|; *or* (*or* (*a*, *b*))**
Логическое НЕ	**~ ; *not* (*not* (*a*, *b*))**
Исключающее ИЛИ	***xor* (*xor* (*a*, *b*))**
Верно, если все элементы вектора равны нулю	***any* (*any* (a))**
Верно, если все элементы вектора не равны нулю	***all* (*all* (*a*))**

Простые логические выражения

if a < b	Истинно, если переменная a меньше переменной b и ложно в противном случае.
if a > b	Истинно, если переменная a больше переменной b и ложно в противном случае.
if a == b	Истинно, если переменная a равна переменной b и ложно в противном случае.
if a <= b	Истинно, если переменная a меньше либо равна переменной b и ложно в противном случае.
if a >= b	Истинно, если переменная a больше либо равна переменной b и ложно в противном случае.
if a ~= b	Истинно, если переменная a не равна переменной b и ложно в противном случае.