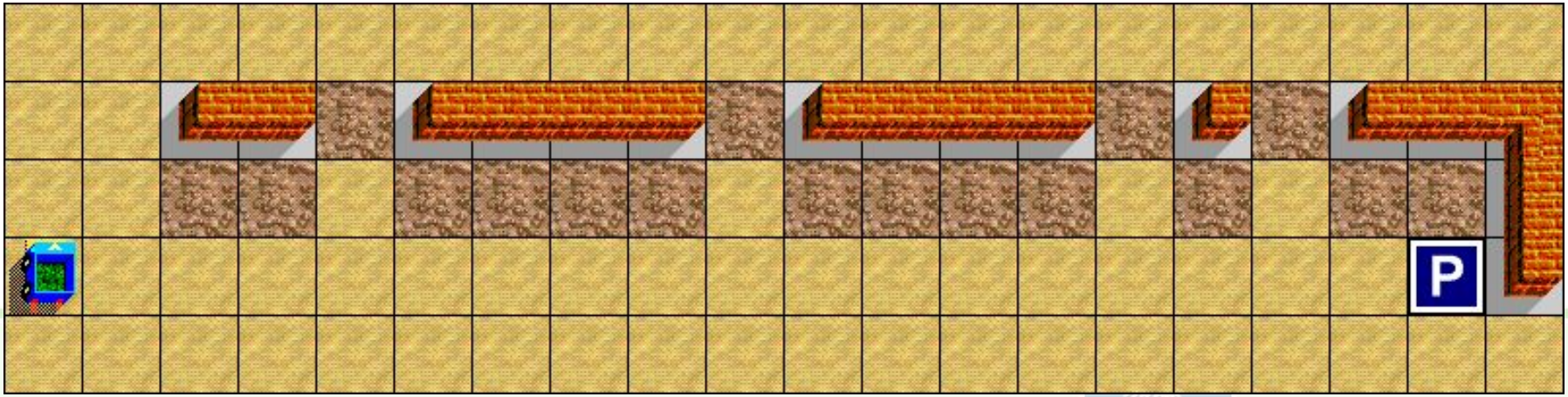
Условный оператор среда Исполнители

Учитель
информатики
МБОУ СОШ №1
с. Александров-Гай
Саратовской области
Гуреева Е.А.

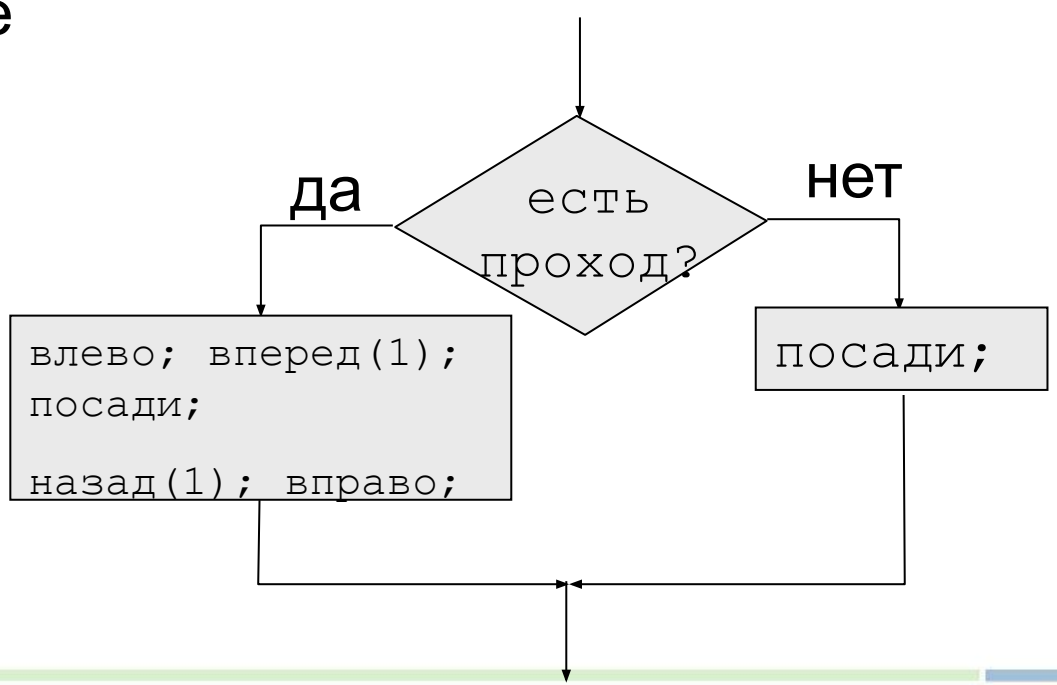
Что такое условный оператор

Рассмотрим новую задачу для Робота. Надо посадить цветы во всех клетках вдоль стены, где нет прохода, а если в этом месте есть проход, войти в него и обработать грядку между стенок. Предполагаем, что длина стены и число

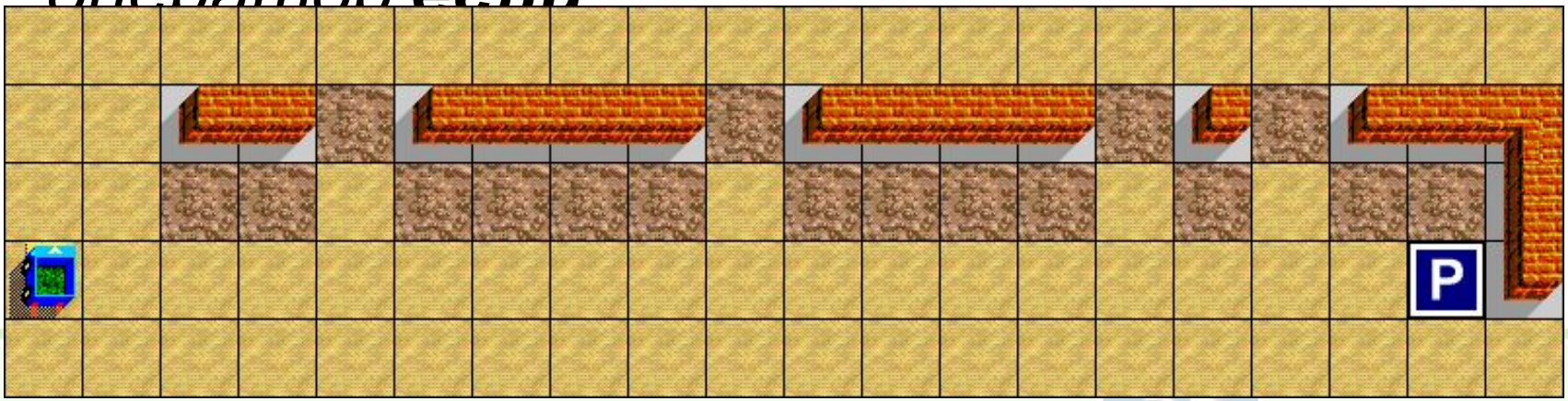


задача z5-3.maz

Для решения этой задачи нам надо научить Робота выполнять разные действия в зависимости от окружающей обстановки. Это можно изобразить на схеме



- Словами это можно сформулировать так: если есть проход (условие ***есть проход*** выполняется), то выполни одну группу команд, если нет – выполни другие команды. В программе для этой цели используется специальный *условный оператор* ***если***



Решение задачи:

```
Выбор
{
вперед(1); направо; вперед(1); /* подойти к началу стены */
пока ( впереди свободно )
   {
   вперед(1);
   если ( слева свободно )
      { /* войти в проход */
      налево; вперед(1);
      посади;
      назад(1); направо;
      }
   иначе
      { посади; }
   вперед(1);
   }
}
```

- Таким образом, мы определили два варианта действий Робота - первый работает тогда, когда обнаружен проход, а второй – когда справа стена.

Правила использования условного оператора

- Условный оператор состоит из двух частей; первая часть начинается ключевым словом ***если*** или ***if*** (от английского “*если*”), после которого в скобках записывается условие.

- Если это условие верно (или *истинно*), то выполняется группа команд, стоящая ниже в фигурных скобках (*блок-если*).

Правила использования условного оператора

- Вторая часть (*блок-иначе*) начинается со слова ***иначе*** или ***else*** (от английского “*иначе*”) и выполняется в том случае, когда условие в скобках *ложно.*

- Нельзя отделять *блок-если* и *блок-иначе,* поскольку они составляют единый оператор.

- Условие ставится только в заголовке *блока-если.*

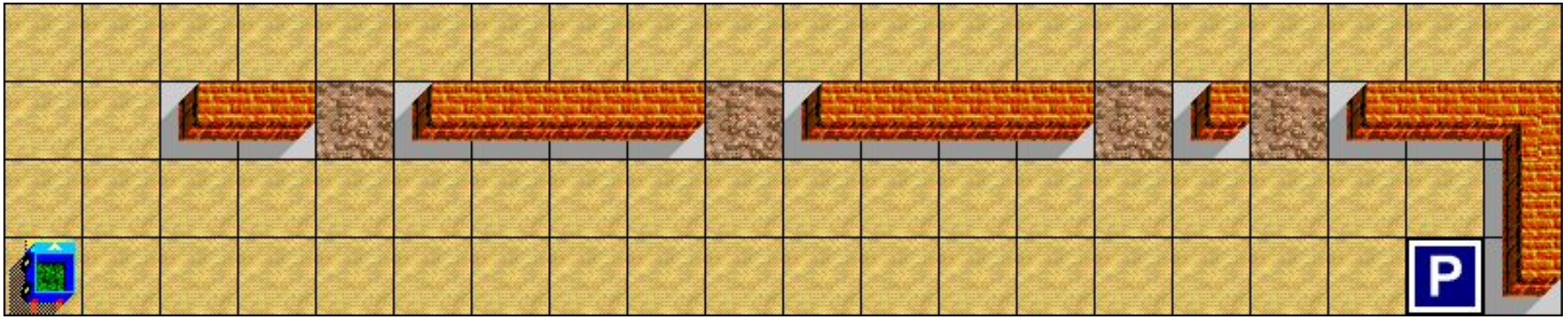
Правила использования условного оператора

- *Блок-иначе* может отсутствовать, если он не нужен; в этом случае мы говорим, что условный оператор записан в сокращенной форме.

- Чтобы было удобнее разбираться в программе, используют отступы так же, как и в циклах: тело *блока-если* и *блока-иначе* сдвигается вправо на 2-3 символа.

Сокращенная форма

- Немного изменим задачу – пусть теперь Роботу надо обрабатывать только по 1 клетке в начале каждого прохода.



- Таким образом, в *блоке-иначе* не осталось ни одной команды – если прохода нет, ничего делать не надо. Поэтому можно использовать сокращенную форму условного оператора – без второй части:

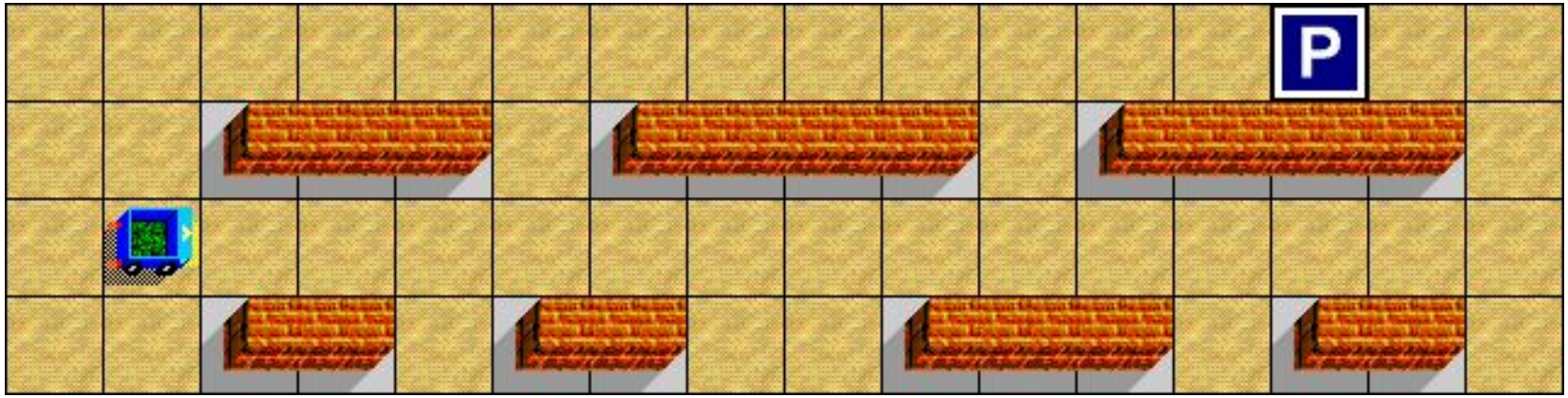
```
Сад2
{
вперед ( 1 );  направо;  вперед ( 1 );
пока ( впереди свободно )
   {
   вперед(1);
   если ( слева свободно )
      {
      налево;  вперед(1);
      посади;
      назад(1);  направо;
      }
}
```

Сложные условия

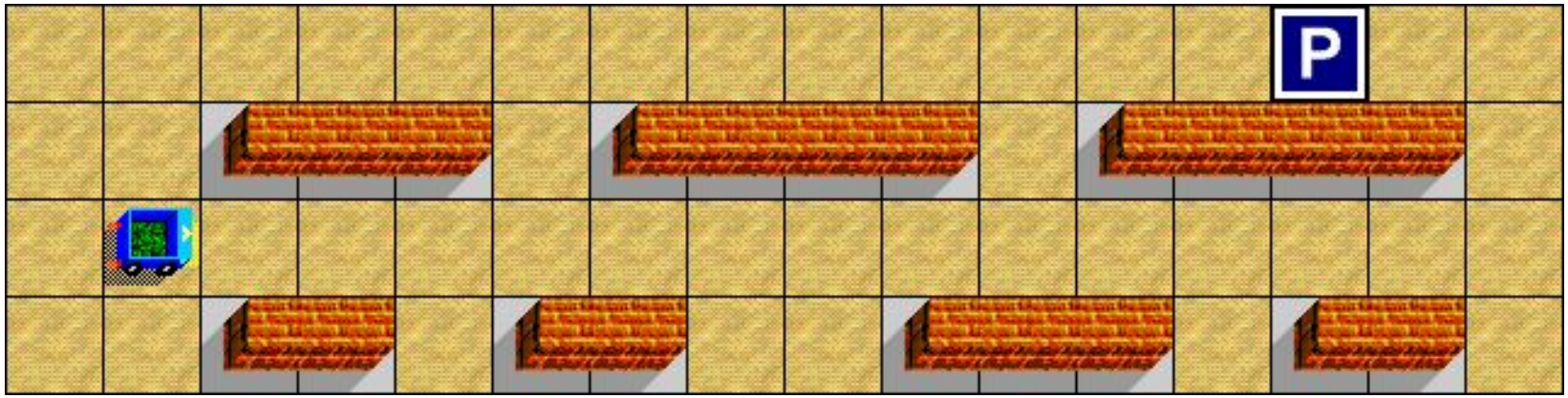
Элективный курс
«Алгоритмы и исполнители»
8 класс

Рассмотрим еще одну задачу для Робота:

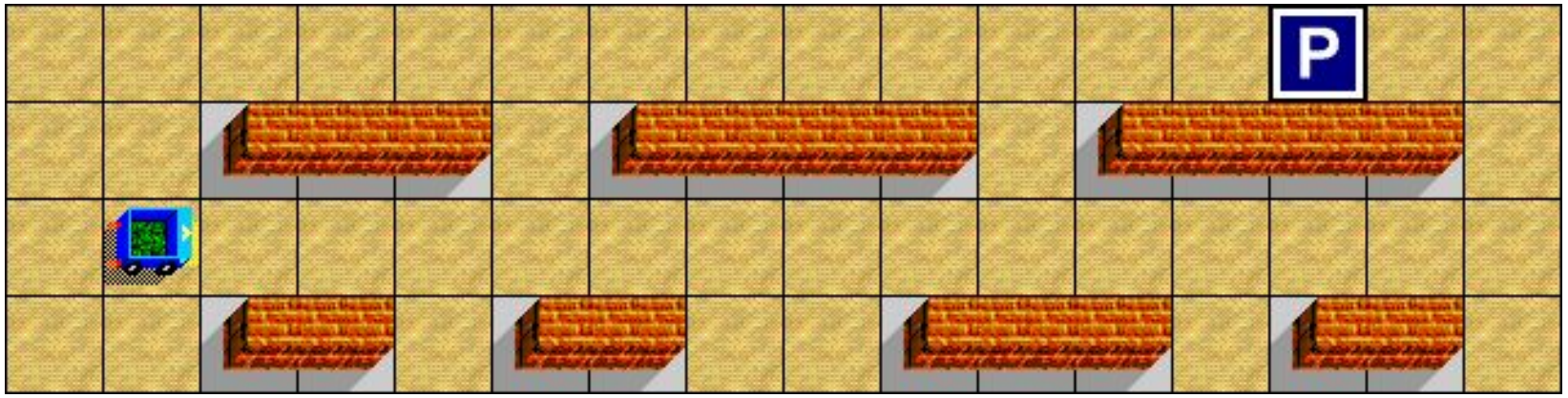
- Ему нужно пройти через коридор с проходами и придти на Базу. Сложность состоит в том, что в обеих стенках есть проходы, сколько их – неизвестно.



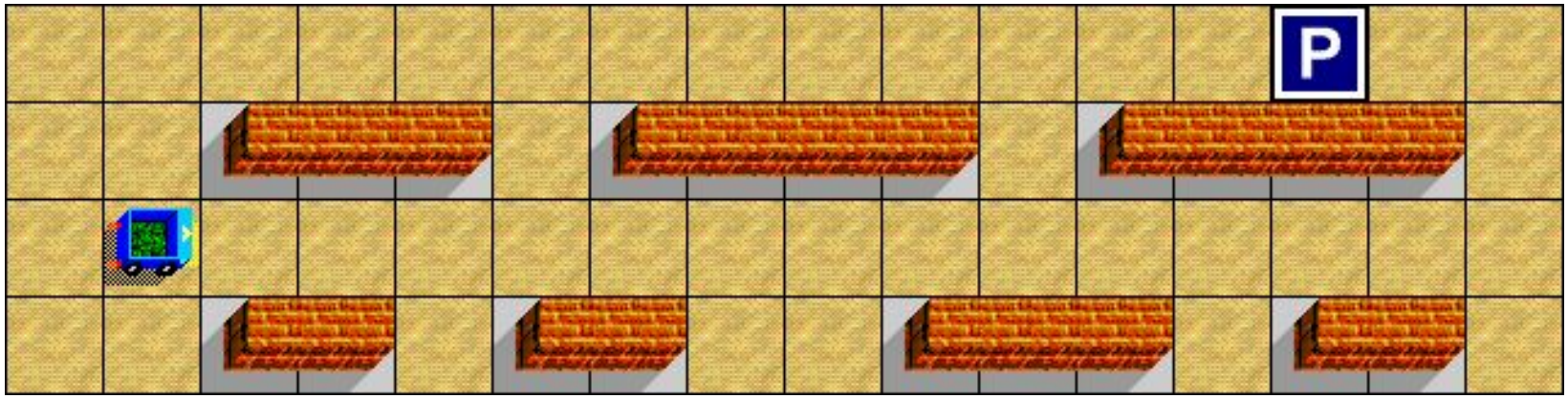
- Мы замечаем, что внутри коридора нет такой клетки, у которой слева и справа – свободные клетки.



- Значит, Роботу надо остановиться, когда слева и справа – свободно, это означает конец коридора.

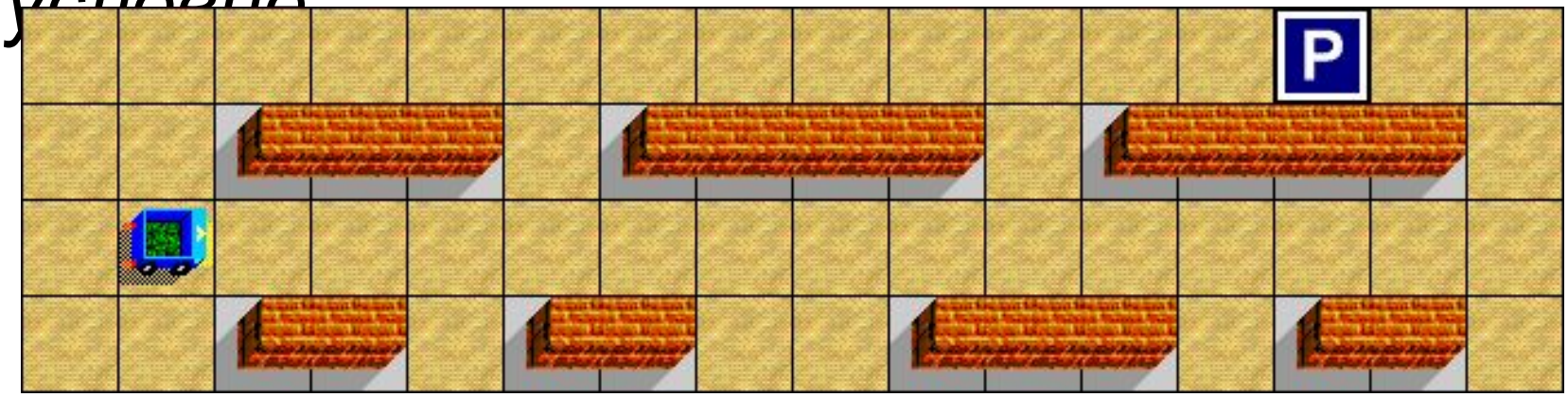


- Теперь можно сформулировать алгоритм прохода через весь коридор на русском языке – иди вперед, пока слева стена **ИЛИ** справа стена



Сложное условие

- В этом словесном алгоритмах мы объединяли логические команды Робота с помощью операции **ИЛИ,** получив из двух простых условий одно *сложное условие*



То же самое можно делать и в программе:

```
Посадка
{
вперед(1);
пока ( слева стена или справа стена )
  вперед (1);
налево;
вперед(2);
налево;
пока ( не база ) вперед ( 1 );
}
```

Сложные условия

- **Сложное условие –** это условие, состоящее из простых условий и **логических операций**:

НЕ отрицание

И логическое умножение

ИЛИ логическое сложение

Правила использования сложных условий

- Простейшими условиями являются логические команды исполнителей (например, ***слева_стена***) и **логические отношения** между значениями

> <	больше, меньше	`т > 5,  2+n < x`
>=	больше или равно	`a >= 2* x+5`
<=	меньше или равно	`c+2*d <= 5*v`
==	равно	`d == 2+c`
<>	не равно	`a != b`

Правила использования сложных условий

- В условии “равно” ставится два знака равенства; чтобы не запутаться, надо запомнить, что если переменная изменяется (*оператор присваивания*), то надо ставить один знак “=“, а если не меняется (*логическое отношение*), то два.

Правила использования сложных условий

- Сложные условия составляются из нескольких простых; простые условия объединяются с помощью **логических операций.**

- Операция **"И"** требует *одновременного* выполнения двух условий, например:

сверху_стена **И** снизу_стена

Правила использования сложных условий

Операция "**ИЛИ**" обозначается требует выполнения *хотя бы одного* из двух условий (или обоих вместе), например:

вверху_стена **ИЛИ** снизу_стена

Правила использования сложных условий

Иногда удобно использовать логическую операцию **“НЕ”**, которая отрицает значение логического выражения, например условия

a < b и ***НЕ (b >= a)*** означают одно и то же.

Правила использования сложных условий

Устанавливается такой **приоритет** (старшинство) логических отношений и операций:

1) сначала выполняются операции в скобках, затем ...
2) операции **“НЕ”**, затем ...
3) логические отношения (>, <, >=, <=, ==, !=), затем ...
4) операции **“И”**
5) и в последнюю очередь операции **“ИЛИ”**.

Правила использования сложных условий

- Для изменения порядка выполнения операций используются скобки.

Используемые источники информации

1. Сайт Константина Полякова. «Исполнители». Изучаем алгоритмы http://kpolyakov.narod.ru/school/robots/robots.htm

2. Методическая служба «Бином». Управление исполнителем Робот http://metodist.lbz.ru/authors/informatika/3/flash/gl3/3.php

3. Шаблон для данной презентации взят с сайта http://pedsovet.su/load/321-1-0-37562