**16.1 Памятка по алгоритмам в КуМИРе**

**Теория для задач типа "Исполнитель Робот"**

Задачи такого типа часто встречаются в контексте алгоритмического мышления и программирования, и помогают учащимся освоить основные концепты программирования, такие как условия, циклы и команды. Рассмотрим теорию, связанную с ключевыми элементами этих задач.

**1. Представление пространства и клеток**

Робот действует в двумерном пространстве, разбитом на клетки. Каждая клетка может быть свободной или содержать стену, которая препятствует движению Робота. Пространство может быть представлено как матрица, где:

* **Свободные клетки** — это клетки, в которых Робот может перемещаться.
* **Стены** — это клетки, через которые Робот не может пройти.

**2. Команды-приказы**

Робот имеет восемь команд для движения по лабиринту и одну команду для закрашивания клеток:

* **вверх (↑)** — Робот перемещается на одну клетку вверх.
* **вниз (↓)** — Робот перемещается на одну клетку вниз.
* **влево (←)** — Робот перемещается на одну клетку влево.
* **вправо (→)** — Робот перемещается на одну клетку вправо.
* **закрасить** — Робот закрашивает клетку, в которой находится.

Важно, чтобы Робот не двигался через стену. Если попытаться переместиться в клетку с препятствием, Робот разрушится.

**3. Команды проверки условий**

Робот может проверять, свободен ли путь в одном из четырёх направлений:

* **сверху свободно** — проверка, свободна ли клетка сверху.
* **снизу свободно** — проверка, свободна ли клетка снизу.
* **слева свободно** — проверка, свободна ли клетка слева.
* **справа свободно** — проверка, свободна ли клетка справа.

Эти проверки помогают Роботу понять, может ли он двигаться в том или ином направлении.

**4. Условные операторы**

Для выполнения определённого действия в зависимости от состояния условий используется конструкция **"если ... то"**. Эта конструкция позволяет Роботу выполнять определённые действия, если условие истинно.

*Пример:*

если справа свободно то

    вправо

    закрасить

все

Здесь Робот будет двигаться вправо и закрашивать клетку, только если справа нет стены.

**5. Логические связки**

В условиях можно использовать логические операторы:

* **и** — все условия должны быть истинными.
* **или** — хотя бы одно условие должно быть истинным.
* **не** — инвертирует условие.

*Пример:*

если (справа свободно) и (не снизу свободно) то

    вправо

все

Здесь Робот двигается вправо, только если справа нет стены и снизу есть стена.

**6. Цикл "пока"**

Цикл позволяет Роботу повторять действия до тех пор, пока выполняется определённое условие.

*Пример:*

нц пока справа свободно

    вправо

кц

Здесь Робот будет двигаться вправо, пока справа есть свободные клетки.

**7. Применение команд в задачах**

Задачи такого типа могут включать:

* Движение по лабиринту в одном или нескольких направлениях.
* Использование условий для принятия решений, например, когда Робот не может двигаться в одном направлении, но может в другом.
* Закрашивание клеток по пути Робота.
* Повторение действий (например, движение в одном направлении до тех пор, пока это возможно).

**8. Решение задач**

Чтобы решить задачу, нужно правильно комбинировать команды проверки условий, условные операторы и циклы. Для этого необходимо внимательно следить за состоянием лабиринта (стенами и свободными клетками) и корректно строить алгоритм для достижения цели.

**Пример алгоритма**

**Задача:** Робот должен пройти через лабиринт, закрасить все клетки, которые он посещает, и при этом избегать стен.

*Решение:*

нц пока существует путь

    если сверху свободно то

        вверх

        закрасить

    иначе если снизу свободно то

        вниз

        закрасить

    иначе если слева свободно то

        влево

        закрасить

    иначе если справа свободно то

        вправо

        закрасить

    все

кц

Этот *алгоритм* будет двигать **Робота** в одном из четырёх направлений, закрашивая клетки, пока существует свободный путь.

**Заключение**

Задачи такого типа учат *алгоритмическому мышлению*, помогают развивать навыки работы с условными операторами и циклами, а также учат, как использовать логические связки для создания сложных условий и управления действиями **Робота** в разных ситуациях.

**16.2 Примеры алгоритмов в КуМИРе**

**Задание 1(на языке КуМИР):**

Исполнитель Робот умеет передвигаться по лабиринту, состоящему из клеток на плоскости. Между некоторыми клетками могут находиться стены, через которые Робот пройти не может. У него есть \*\*четыре команды-приказы\*\* для перемещения:

- 'вверх'  
- 'вниз'  
- 'влево'  
- 'вправо'

Каждая из этих команд перемещает Робота на одну клетку в соответствующем направлении. Если Роботу будет дана команда двигаться через стену, он разрушится. Также у Робота есть команда `закрасить`, которая окрашивает текущую клетку, где он находится.

Кроме того, у Робота есть *четыре команды проверки условий*, которые позволяют ему проверять наличие свободного пути в разных направлениях:

- 'сверху свободно'  
- 'снизу свободно'  
- 'слева свободно'  
- 'справа свободно'

Команды проверки условий используются совместно с конструкцией `если` следующего вида:

если условие то

   последовательность команд

все

где *условие* — это одна из команд проверки условий, а \*последовательность команд\* — любые команды-приказы. Например, чтобы переместить Робота на одну клетку вправо, если справа нет стены, и затем закрасить эту клетку, можно использовать такой алгоритм:

если справа свободно то

   вправо

   закрасить

все

Также можно комбинировать несколько проверок условий с помощью логических операторов `и`, `или`, `не`. Например:

если (справа свободно) и (не снизу свободно) то

   вправо

все

Для выполнения циклических действий используется конструкция `цикл пока`:

нц пока условие

   последовательность команд

кц

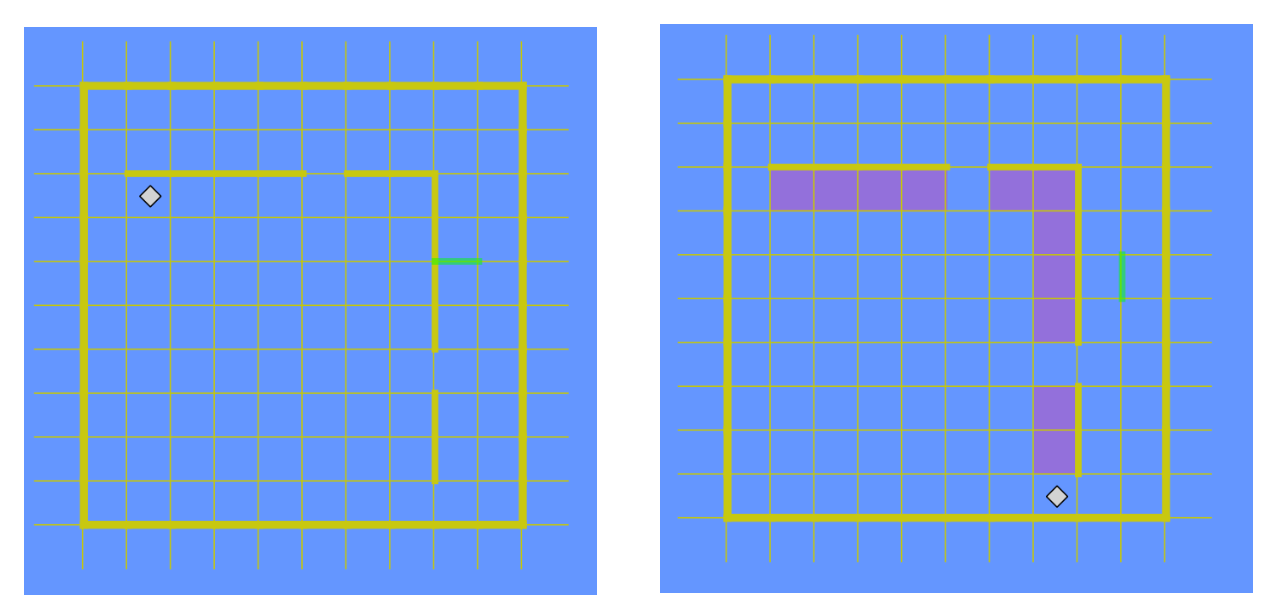
Пример использования цикла для непрерывного движения вправо до тех пор, пока это возможно:

нц пока справа свободно

   вправо

кц

**Задание:**



На бесконечном поле расположены две стены: горизонтальная и вертикальная. Они пересекаются таким образом, что правый конец горизонтальной стены соединяется с верхним концом вертикальной стены. Длина каждой стены неизвестна, но известно, что каждая стена имеет ровно один проход шириной в одну клетку. Робот изначально расположен под самым левым краем горизонтальной стены.

Необходимо написать алгоритм, который закрасит все клетки, находящиеся прямо под горизонтальной стеной и левее вертикальной стены, исключая сами проходы. После завершения работы Робот может оказаться в любом месте поля. Важно, чтобы Робот не разрушился и выполнил свою задачу корректно независимо от длины стен и местоположения проходов.

**Решение:**

Давайте разберём решение задачи шаг за шагом, основываясь на вашем предложенном алгоритме.

Можно условие задачи расшифровать кратко следующим образом: нам нужно закрасить все клетки, которые находятся непосредственно под горизонтальной стеной и левее вертикальной стены, при этом не закрашивая проходы. Робот начинает движение из клетки, находящейся под самой левой частью горизонтальной стены.

*Предлагаемые этапы решения задачи:*

1. Закраска всех клеток под горизонтальной стеной

нц пока не сверху свободно

   закрасить

   вправо

кц

Этот цикл проходит вдоль всей горизонтальной стены, начиная с самого начала, и закрашивает каждую клетку под ней. Цикл продолжается до тех пор, пока Робот не достигнет прохода в стене (то есть пока сверху остаётся свободное пространство).

2. Перемещение к началу вертикальной стены

нц пока сверху свободно

   вправо

кц

После того как Робот дошёл до прохода в горизонтальной стене, этот цикл позволяет ему перейти через проход и продолжить движение вправо до тех пор, пока он не дойдёт до вертикальной стены.

3. Закраска клеток слева от вертикальной стены

нц пока справа свободно

   закрасить

   вправо

кц

Теперь Робот движется вдоль вертикальной стены, закрашивая все клетки слева от неё. Этот цикл заканчивается, когда Робот достигает прохода в вертикальной стене.

4. Возвращение назад и закраска оставшихся клеток

нц пока не справа свободно

   закрасить

   вниз

кц

Когда Робот доходит до прохода в вертикальной стене, он опускается вниз и продолжает закрашивать оставшиеся клетки, двигаясь обратно к начальной позиции.

5. Завершение закрашивания

нц пока справа свободно

   вниз

кц

Этот цикл завершает перемещение Робота вниз, пока он не достигнет нижней границы области, которую необходимо закрасить.

6. Окончательная проверка и закраска

нц пока не справа свободно

   закрасить

   вниз

кц

Заключительный цикл гарантирует, что все оставшиеся клетки будут закрашены перед окончанием программы.

Объеденим все шаги решения задачи в один алгоритм:

# Шаг 1: Закраска всех клеток под горизонтальной стеной

нц пока не сверху свободно

закрасить

вправо

кц

# Шаг 2: Переход к вертикальной стене

нц пока сверху свободно

вправо

кц

# Шаг 3: Закраска клеток слева от вертикальной стены

нц пока справа свободно

закрасить

вправо

кц

# Шаг 4: Возвращение назад и закраска оставшихся клеток

нц пока не справа свободно

закрасить

вниз

кц

# Шаг 5: Завершающее перемещение вниз

нц пока справа свободно

вниз

кц

# Шаг 6: Окончательное завершение закрашивания

нц пока не справа свободно

закрасить

вниз

кц

Этот *алгоритм*решает поставленную задачу корректно, обеспечивая закрашивание всех необходимых клеток без разрушения **Робота**.