Исполнитель ‹‹Чертежник››

1. Использование исполнителя

Исполнитель ‹‹Чертежник›› входит в Вашу поставку Кумир, но его функции не являются частью языка программирования. Для его использования необходимо в программе явно указать использование данного исполнителя:

```
использовать Чертежник
| теперь функции чертежника доступны
| для использования в программе
```

2. Команды действий

2.1. опустить перо

Синтаксис:

```
алг опустить перо
```

Опускает перо. При дальнейшем перемещении будет оставаться след.

2.2. поднять перо

Синтаксис:

```
алг поднять перо
```

Поднимает перо. При дальнейшем перемещении будет оставаться след.

2.3. выбрать чернила

Синтаксис:

```
алг выбрать чернила(цвет a)
```

Устанавливает цвет чернил.

- *a* – цвет чернил.

2.4. сместиться в точку

Синтаксис:

```
алг сместиться в точку(вещ x, вещ y)
```

Смещает перо в заданную точку.

- *x* – абсцисса точки,
- *y* – её ордината.

2.5. сместиться на вектор

Синтаксис:

```
алг сместиться на вектор(вещ dx, вещ dy)
```

Смещает перо на вектор (*dx*, *dy*). Иными словами, если до выполнения команды перо находилось в точке `(x, y)`, то после её выполнения, оно будет находиться в точке `(x + dx, y + dy)`.

2.6. написать

Синтаксис:

```
алг написать(вещ ширина, лит текст)
```

Пишет строку, растянутую или сжатую до заданной ширины.

- *ширина* – ширина строки,
- *текст* – текст строки.

3. Алгоритмы контроля обстановки

3.1. @есть отрезок в области

Синтаксис:

```
алг лог @есть отрезок в области(вещ x, вещ y, вещ радиус)
```

Возвращает **да**, если в заданной окружности есть хотя бы один отрезок.

- *x* – абсцисса центра проверочной окружности,
- *y* – ордината центра проверочной окружности,

радиус – радиус проверочной окружности.

Исполнитель ‹‹Комплексные числа››

---

1. Использование исполнителя

Исполнитель ‹‹Комплексные числа›› входит в Вашу поставку Кумир, но его функции не являются частью языка программирования. Для его использования необходимо в программе явно указать использование данного исполнителя:

```
использовать Комплексные числа
| теперь функции комплексных чисел доступны
| для использования в программе
```

2. Тип величины ‹‹компл››

В этом исполнителе реализован новый тип величин – **компл**. Переменная этого типа представляет собой комплексное число. Комплексные числа записываются в виде `a*i + b`. Знак умножения можно опустить.

Скопировать пример

```
использовать Комплексные числа
алг
нач
  компл а = 1i + 1
  компл б = 2*i + 4
  вывод а * (а + б)
кон
```

Пример 1. Представление комплексных чисел

Для того, чтобы ввести комплексное число с клавиатуры, следует вводить его по такому же правилу.

Для типа **компл** реализованы все арифметические операции, кроме возведения в степень. Сравнение же для комплексных чисел неопределено. Поскольку любое действительное число является также комплексным, то переменной типа **компл** можно присвоить переменную не только такого же типа, но и типа **цел** или **вещ**.

3. Команды действий

3.1. Re

Синтаксис:

```
алг вещ Re(компл x)
```

Возвращает действительную (real) часть числа x.

3.2. Im

Синтаксис:

```
алг вещ Im(компл x)
```

Возвращает мнимую (imaginary) часть числа x.

Исполнитель ‹‹Кузнечик››

---

1. Использование исполнителя

Исполнитель ‹‹Кузнечик›› входит в Вашу поставку Кумир, но его функции не являются частью языка программирования. Для его использования необходимо в программе явно указать использование данного исполнителя:

```
использовать Кузнечик
| теперь функции кузнечика доступны
| для использования в программе
```

2. Команды действий

2.1. вперед

Синтаксис:

```
алг вперед x
```

Сделать прыжок длины x вперёд. Длина прыжка (в количестве клеток) указывается при создании обстановки или как параметр исполнителя. В зависимости от конкретной обстановки меняется и

имя данного алгоритма. Например, для обстановки по умолчанию (3, 2), данный алгоритм будет называться `вперед 3`.

- *x* – длина прыжка.

2.2. назад

Синтаксис:

```
алг назад x
```

Сделать прыжок длины x назад. Длина прыжка (в количестве клеток) указывается при создании обстановки или как параметр исполнителя. В зависимости от конкретной обстановки меняется и имя данного алгоритма. Например, для обстановки по умолчанию (3, 2), данный алгоритм будет называться `назад 2`.

- *x* – длина прыжка.

2.3. перекрасить

Синтаксис:

```
алг перекрасить
```

Меняет цвет клетки, в которой в данный момент находится Кузнечик. Если клетка была чистая, закрашивает её. Если закрашена – снимает закрасу.

Исполнитель ‹‹Вертун››

---

1. Общие сведения

Исполнитель Вертун является основным исполнителем системы ПиктоМир, его наличие в КуМире необходимо для перехода от одной системы к другой.

Поле исполнителя - это прямоугольный участок на плоскости, ограниченный со всех сторон стенами и возможно, имеющий стены внутри.

Поле разделено на квадратные клетки, которые могут быть отмеченными точкой или закрашенными.

Исполнителю Вертун необходимо пройти по полю и закрасить те клетки, которые отмечены точкой, но не закрашены.

Вертун может перемещаться по полю, не натыкаясь на стены. Также он может закрашивать клетки или проверять их закрашенность (в некоторых задачах это требуется для определения местоположения Вертуна).

2. Команды действий

2.1. вперед

Синтаксис:

```
алг вперед
```

Перемещает Вертуна на одну клетку вперед. Если перед Вертуном находится стена, то возникает ошибка выполнения ‹‹Робот ударился об стену››.

2.2. повернуть налево

Синтаксис:

```
алг повернуть налево
```

Поворачивает Вертуна на 90 градусов против часовой стрелки.

2.3. повернуть направо

Синтаксис:

```
алг повернуть направо
```

Поворачивает Вертуна на 90 градусов по часовой стрелке.

2.4. закрасить

Синтаксис:

```
алг закрасить
```

Закрашивает клетку, на которой находится Вертун.

3. Команды опроса состояния

3.1. впереди стена

Синтаксис:

```
алг лог впереди стена
```

Возвращает **да**, если перед Вертуном находится стена, или **нет**, если Вертун может спокойно пройти сквозь стену.

3.2. впереди свободно

Синтаксис:

```
алг лог впереди свободно
```

Возвращает **да**, если Вертун может спокойно пройти сквозь стену, или **нет**, если перед Вертуном находится стена.

3.3. клетка закрашена

Синтаксис:

```
алг лог клетка закрашена
```

Возвращает **да**, если клетка, на которой находится Вертун, закрашена, или **нет**, если клетка чистая.

3.4. клетка чистая

Синтаксис:

```
алг лог клетка чистая
```

Возвращает **да**, если клетка, на которой находится Вертун, чистая, или **нет**, если клетка закрашена.

4. Алгоритмы контроля обстановки исполнителя ‹‹Вертун››

Алгоритмы, описанные ниже могут быть вызваны только из ‹‹скрытой›› части программы. Они служат для получении различных данных о текущей обстановке исполнителя **Вертун**.

4.1. Система координат Вертуна

Учительские алгоритмы предполагают получение информации о поле Вертуна с использованием декартовой системы координат. Клетки нумеруются с единицы по двум координатам.

Рисунок 1. Система координат Вертуна. Поле на этом рисунке имеет размеры: 6 клеток в ширину, и 5 клеток в высоту

4.2. @@есть точка

Синтаксис:

```
алг лог @@есть точка(цел x, цел y)
```

Проверяет, есть ли точка в клетке с координатами (*x*, *y*). Если точка в клетке есть, возвращает **да**, иначе – **нет**.

4.3. @@клетка закрашена

Синтаксис:

```
алг лог @@клетка закрашена(цел x, цел y)
```

Проверяет, закрашена ли клетка с координатами (*x*, *y*). Если клетка закрашена, возвращает **да**, иначе – **нет**.

4.4. @@положение робота x

Синтаксис:

```
алг цел @@положение робота x
```

Возвращает x -координату текущего положения исполнителя ‹‹Вертун››.

4.5. @@положение робота y

Синтаксис:

```
алг цел @@положение робота y
```

Возвращает y -координату текущего положения исполнителя ‹‹Вертун››.

4.6. @@размер поля x

Синтаксис:

```
алг цел @@размер поля x
```

Возвращает ширину обстановки исполнителя ‹‹Вертун››.

4.7. @@размер поля y

Синтаксис:

```
алг цел @@размер поля y
```

Возвращает высоту обстановки исполнителя ‹‹Вертун››.

4.8. @@финишная

Синтаксис:

```
алг лог @@финишная(цел x, цел y)
```

Проверяет, есть ли финишный флаг в клетке с координатами (*x*, *y*). Если флаг в клетке есть, возвращает **да**, иначе – **нет**.

Исполнитель ‹‹Клавиатура››

---

1. Использование исполнителя

Исполнитель ‹‹Клавиатура›› входит в Вашу поставку Кумир, но его функции не являются частью языка программирования. Для его использования необходимо в программе явно указать использование данного исполнителя:

```
использовать Клавиатура
| теперь функции Клавиатуры доступны
| для использования в программе
```

Внимание: этот исполнитель доступен только при запуске в оконном режиме. Им нельзя пользоваться при решении олимпиадных задач или сдаче ЕГЭ!

2. Тип величины ‹‹сканкод››

В этом исполнителе реализован новый перечислимый тип величин – **сканкод**. Для этого типа реализованы операции проверки на равенство (=, <>), в том числе и с типом **цел**.

Предусмотрены следующие имена констант типа **сканкод**:

Таблица 1. Коды клавиш

Константа	Клавиша
КЛ_НАЗАД или КЛ_BACKSPACE	Клавиша BACKSPACE ↤
КЛ_TAB	Клавиша TAB ⇆
КЛ_ВВОД или КЛ_ENTER или КЛ_RETURN	Клавиша ENTER ↲
КЛ_ПРОБЕЛ или КЛ_SPACE	Клавиша пробела
КЛ_PAGEUP или КЛ_PGUP	Клавиша PAGE UP
КЛ_PAGEDOWN или КЛ_PGDOWN	Клавиша PAGE DOWN
КЛ_HOME	Клавиша HOME
КЛ_END	Клавиша END
КЛ_INSERT	Клавиша INSERT
КЛ_DELETE	Клавиша DELETE
КЛ_ВЛЕВО	Клавиша ←
КЛ_ВПРАВО	Клавиша →
КЛ_ВВЕРХ	Клавиша ↑
КЛ_ВНИЗ	Клавиша ↓
КЛ_F1	Клавиша F1
КЛ_F2	Клавиша F2
КЛ_F3	Клавиша F3
КЛ_F4	Клавиша F4
КЛ_F5	Клавиша F5

КЛ_F6	Клавиша F6
КЛ_F7	Клавиша F7
КЛ_F8	Клавиша F8
КЛ_F9	Клавиша F9
КЛ_F10	Клавиша F10
КЛ_F11	Клавиша F11
КЛ_F12	Клавиша F12
КЛ_1	Клавиша 1 (любая)
КЛ_2	Клавиша 2 (любая)
КЛ_3	Клавиша 3 (любая)
КЛ_4	Клавиша 4 (любая)
КЛ_5	Клавиша 5 (любая)
КЛ_6	Клавиша 6 (любая)
КЛ_7	Клавиша 7 (любая)
КЛ_8	Клавиша 8 (любая)
КЛ_9	Клавиша 9 (любая)
КЛ_0	Клавиша 0 (любая)
КЛ_Q или КЛ_Й	Клавиша Q (в любой раскладке)
КЛ_W или КЛ_Ц	Клавиша W (в любой раскладке)
КЛ_E или КЛ_У	Клавиша E (в любой раскладке)
КЛ_R или КЛ_К	Клавиша R (в любой раскладке)
КЛ_T или КЛ_Е	Клавиша T (в любой раскладке)
КЛ_Y или КЛ_Н	Клавиша Y (в любой раскладке)
КЛ_U или КЛ_Г	Клавиша U (в любой раскладке)
КЛ_I или КЛ_Ш	Клавиша I (в любой раскладке)
КЛ_O или КЛ_Щ	Клавиша O (в любой раскладке)

КЛ_P или КЛ_З	Клавиша P (в любой раскладке)
КЛ_X	Клавиша [(в любой раскладке)
КЛ_Ъ	Клавиша] (в любой раскладке)
КЛ_A или КЛ_Ф	Клавиша A (в любой раскладке)
КЛ_S или КЛ_Ы	Клавиша S (в любой раскладке)
КЛ_D или КЛ_В	Клавиша D (в любой раскладке)
КЛ_F или КЛ_А	Клавиша F (в любой раскладке)
КЛ_G или КЛ_П	Клавиша G (в любой раскладке)
КЛ_H или КЛ_Р	Клавиша H (в любой раскладке)
КЛ_J или КЛ_О	Клавиша J (в любой раскладке)
КЛ_K или КЛ_Л	Клавиша K (в любой раскладке)
КЛ_L или КЛ_Д	Клавиша L (в любой раскладке)
КЛ_Ж	Клавиша ; (в любой раскладке)
КЛ_Э	Клавиша ' (в любой раскладке)
КЛ_Z или КЛ_Я	Клавиша Z (в любой раскладке)
КЛ_X или КЛ_Ч	Клавиша X (в любой раскладке)
КЛ_C или КЛ_С	Клавиша C (в любой раскладке)
КЛ_V или КЛ_М	Клавиша V (в любой раскладке)
КЛ_B или КЛ_И	Клавиша B (в любой раскладке)
КЛ_N или КЛ_Т	Клавиша Q (в любой раскладке)
КЛ_M или КЛ_Ь	Клавиша Q (в любой раскладке)
КЛ_Б	Клавиша , (в любой раскладке)
КЛ_Ю	Клавиша . (в любой раскладке)

Скопировать пример

```
использовать Клавиатура
алг
нач
  сканкод Остановка = КЛ_ВВОД
  цел Клавиша = 0
  нц
    если сигнал клав то
      Клавиша:=код клав
      вывод Клавиша, нс
    все
    ждать(100)
  кц при Клавиша=Остановка
кон
```

Пример 1. Опрос нажатий клавиатуры

3. Алгоритмы исполнителя

3.1. сигнал клав

Синтаксис:

```
алг лог сигнал клав
```

Возвращает **да**, если с момента начала выполнения, или предыдущего вызова этого алгоритма была нажата клавиша.

3.2. код клав

Синтаксис:

```
алг цел код клав
```

Возвращает код символа, который можно сравнивать с одной из величин типа **сканкод**.

3.3. сброс клав

Синтаксис:

```
алг сброс клав
```

Удаляет информацию о событиях нажатия клавиш.

Исполнитель ‹‹Рисователь››

Рисователь – исполнитель системы Кумир. Назначение Рисователя – создавать рисунки на листе. Как и другие исполнители, он имеет систему команд, свою ‹‹обстановку›› – лист, на котором можно рисовать, и окно, в котором виден лист.

Исполнитель разработан по инициативе д. т. н. проф. К. Ю. Полякова, спецификация исполнителя создана совместно группой разработки Кумир и К. Ю. Поляковым.

---

1. Использование исполнителя

Исполнитель ‹‹Рисователь›› входит в Вашу поставку Кумир, но его функции не являются частью языка программирования. Для его использования необходимо в программе явно указать использование данного исполнителя:

```
использовать Рисователь
| теперь функции рисователя доступны
| для использования в программе
```

2. Тип величины ‹‹цвет››

В исполнителе реализован новый тип величин – **цвет**. Значение величины типа **цвет** может быть получено:

- использованием одного из алгоритмов составления цвета из отдельных компонент;
- присваиванием величине одного из значений: прозрачный, белый, чёрный, серый, фиолетовый, синий, голубой, зелёный, жёлтый, оранжевый, красный.

Значение типа **цвет** записывается без кавычек, вместо буквы ё можно использовать букву е.

Скопировать пример

```
использовать Рисователь
алг радуга
нач
  новый лист(400, 200, белый)
  кисть(красный)
  окружность(200, 200, 200)
  кисть(оранжевый)
  окружность(200, 200, 175)
  кисть(желтый)
  окружность(200, 200, 150)
  кисть(зелёный)
  окружность(200, 200, 125)
  кисть(голубой)
```

```
  окружность(200, 200, 100)
  кисть(синий)
  окружность(200, 200, 75)
  кисть(фиолетовый)
  окружность(200, 200, 50)
  кисть(белый)
  окружность(200, 200, 25)
кон
```

Пример 1. Рисователь рисует радугу

3. Алгоритмы для работы с цветом

3.1. CMYK

Синтаксис:

```
алг цвет CMYK(цел c, цел m, цел y, цел k)
```

Возвращает величину типа **цвет**, составленную из компонент цветовой модели CMYK:

- *c* – величина бирюзовой составляющей цвета;
- *m* – величина пурпурной составляющей цвета;
- *y* – величина желтой составляющей цвета;
- *k* – величина черной составляющей цвета.

Все значения состовляющих должны быть от 0 до 255.

Цветовая модель CMYK основана на аддитивном смешиванием трех красок: чем выше значение отдельных компонент, тем темнее получается цвет.

Как правило, при смешивании трех основных цветов – бирюзового (*c*), пурпурного (*m*) и желтого (*y*), не удается получить настоящий темный цвет. Поэтому к трем основным цветам добавлен черный (*k*).

Рисунок 1. Цветовая модель CMYK

3.2. RGB

Синтаксис:

```
алг цвет RGB(цел r, цел g, цел b)
```

Возвращает величину типа **цвет**, составленную из компонент цветовой модели RGB:

- *r* – величина красной составляющей цвета;
- *g* – величина зеленой составляющей цвета;
- *b* – величина синей составляющей цвета.

Все значения состовляющих должны быть от 0 до 255.

Цветовая модель RGB основана на субтрактивном смешивании трех красок: чем выше значение отдельных компонент, тем светлее получается цвет.

Рисунок 2. Цветовая модель RGB

3.3. HSL

Синтаксис:

```
алг цвет HSL(цел h, цел s, цел l)
```

Возвращает величину типа **цвет**, составленную из компонент цветовой модели HSL:

- *h* – тон цвета (значения в градусах от 0 до 359);
- *s* – значение насыщенности цвета (от 0 до 255);
- *s* – значение интенсивности цвета (от 0 до 255).

3.4. HSV

Синтаксис:

```
алг цвет HSV(цел h, цел s, цел v)
```

Возвращает величину типа **цвет**, составленную из компонент цветовой модели HSV:

- *h* – тон цвета (значения в градусах от 0 до 359);
- *s* – значение насыщенности цвета (от 0 до 255);
- *v* – значение яркости цвета (от 0 до 255).

3.5. CMYKA

Синтаксис:

```
алг цвет CMYKA(цел c, цел m, цел y, цел k, цел a)
```

Возвращает величину типа **цвет**, составленную из компонент цветовой модели CMYK и значения альфа-канала:

- *c* – величина бирюзовой составляющей цвета;
- *m* – величина пурпурной составляющей цвета;
- *y* – величина желтой составляющей цвета;
- *k* – величина черной составляющей цвета;
- *a* – значение непрозрачности цвета (α-значение).

Все значения состовляющих должны быть от 0 до 255.

3.6. RGBA

Синтаксис:

```
алг цвет RGBA(цел r, цел g, цел b, цел a)
```

Возвращает величину типа **цвет**, составленную из компонент цветовой модели RGB и значения альфа-канала:

- *r* – величина красной составляющей цвета;
- *g* – величина зеленой составляющей цвета;
- *b* – величина синей составляющей цвета;
- *a* – значение непрозрачности цвета (α-значение).

Все значения состовляющих должны быть от 0 до 255.

3.7. HSLA

Синтаксис:

```
алг цвет HSLA(цел h, цел s, цел l, цел a)
```

Возвращает величину типа **цвет**, составленную из компонент цветовой модели HSL и значения альфа-канала:

- *h* – тон цвета (значения в градусах от 0 до 359);
- *s* – значение насыщенности цвета (от 0 до 255);
- *l* – значение интенсивности цвета (от 0 до 255);

- *a* – значение непрозрачности цвета (α-значение, от 0 до 255).

3.8. HSVA

Синтаксис:

```
алг цвет HSVA(цел h, цел s, цел v, цел a)
```

Возвращает величину типа **цвет**, составленную из компонент цветовой модели HSL и значения альфа-канала:

- *h* – тон цвета (значения в градусах от 0 до 359);
- *s* – значение насыщенности цвета (от 0 до 255);
- *v* – значение яркости цвета (от 0 до 255);
- *a* – значение непрозрачности цвета (α-значение, от 0 до 255).

3.9. разложить в CMYK

Синтаксис:

```
алг CMYK(цвет цв, аргрез цел c, аргрез цел m, аргрез цел y, аргрез
цел k)
```

Позволяет получить значения параметров модели CMYK для анализируемого цвета.

- *цв* – анализируемый цвет;
- *c* – величина бирюзовой составляющей цвета;
- *m* – величина пурпурной составляющей цвета;
- *y* – величина желтой составляющей цвета;
- *k* – величина черной составляющей цвета.

Все значения состовляющих должны быть от 0 до 255.

3.10. разложить в RGB

Синтаксис:

```
алг RGB(цвет цв, аргрез цел r, аргрез цел g, аргрез цел b)
```

Позволяет получить значения параметров модели RGB для анализируемого цвета.

- *цв* – анализируемый цвет;
- *r* – величина красной составляющей цвета;
- *g* – величина зеленой составляющей цвета;
- *b* – величина синей составляющей цвета.

Все значения состовляющих должны быть от 0 до 255.

Скопировать пример

```
использовать Рисователь
алг
нач
  цвет цв = фиолетовый
  цел r, g, b
  разложить в RGB(цв, r, g, b)
  вывод r, нс, g, нс, b
кон
```

Пример 2. Разложение в RGB

3.11. разложить в HSL

Синтаксис:

```
алг HSL(цвет цв, аргрез цел h, аргрез цел s, аргрез цел l)
```

Позволяет получить значения параметров модели HSL для анализируемого цвета.

- *цв* – анализируемый цвет;
- *h* – тон цвета (значения в градусах от 0 до 359);
- *s* – значение насыщенности цвета (от 0 до 255);
- *l* – значение интенсивности цвета (от 0 до 255).

3.12. разложить в HSV

Синтаксис:

```
алг HSV(цвет цв, аргрез цел h, аргрез цел s, аргрез цел v)
```

Позволяет получить значения параметров модели HSL для анализируемого цвета.

- *цв* – анализируемый цвет;
- *h* – тон цвета (значения в градусах от 0 до 359);
- *s* – значение насыщенности цвета (от 0 до 255);

- *v* – значение яркости цвета (от `0` до `255`).

4. Информационные алгоритмы

Алгоритмы этой группы позволяют определить размеры листа и текстовой надписи (в пикселях).

Вызов одного из этих алгоритмов не меняет состояние рисователя.

4.1. высота листа

Синтаксис:

```
алг цел высота листа
```

Возвращает высоту листа.

4.2. ширина листа

Синтаксис:

```
алг цел ширина листа
```

Возвращает ширину листа.

4.3. центр x

Синтаксис:

```
алг цел центр x
```

Возвращает координату середины листа по горизонтали (ширине).

4.4. центр y

Синтаксис:

```
алг цел центр y
```

Возвращает координату середины листа по вертикали (высоте).

4.5. ширина текста

Синтаксис:

```
алг цел ширина текста(лит текст)
```

Возвращает ширину текста в пикселях при заданном шрифте.

- *текст* – измеряемый текст.

4.6. значение в точке

Синтаксис:

```
алг цвет значение в точке(цел x, цел y)
```

Возвращает значение цвета в точке с заданными координатами.

- *x* – координата точки по горизонтали;
- *y* – координата точки по вертикали.

5. Установка параметров рисования

Эти команды нужны для задания параметров рисования линии (толщина, цвет), закрашивания (цвет), рисования текстовых надписей (гарнитура, размер символов, использование жирных или курсивных букв). Эти параметры используются в ко- мандах рисования.

Толщина линий задается в пикселях. Гарнитуры задаются литеральными величинами, список допустимых гарнитур определяется таковым списком в операционной системе.

5.1. перо

Синтаксис:

```
алг перо(цел толщина, цвет окраска контура)
```

Устанавливает параметры контура.

- *толщина* – толщина линии. Изначально равна 1. При толщине 0 линия не рисуется, но заливка внутренней области фигуры производится.
- *окраска контура* – цвет линии вокруг рисуемой фигуры.

5.2. кисть

Синтаксис:

```
алг кисть(цвет заливка)
```

Устанавливает цвет заливки.

- *заливка* – цвет заливки. Изначально прозрачный (рисование только контуров).

5.3. убрать кисть

Синтаксис:

```
алг убрать кисть
```

Отключает кисть. Вся дальнейшая работа с Рисователем будет происходить без заливки. Чтобы снова начать использовать кисть, нужно выполнить кисть.

Скопировать пример

```
использовать Рисователь
алг
нач
  цвет залив = красный
  кисть(залив)
  окружность(0,0,50)
  убрать кисть
  окружность(0,0,100)
кон
```

Пример 3. убрать кисть

5.4. плотность

Синтаксис:

```
алг плотность
```

Устанавливает плотность заливки. Параметр *значение* должен принадлежать промежутку [0; 8].

5.5. шрифт

Синтаксис:

```
алг шрифт(лит гарнитура, цел размер, лог жирный, лог курсив)
```

Устанавливает параметры шрифта, которые используются алгоритмами надпись и ширина текста. Изначальные параметры шрифта соответствуют таковым в настройках операционной системы.

- *гарнитура* – гарнитура шрифта (Arial, Times New Roman и т.д.);
- *размер* – размер шрифта, который определяется как количество пикселей от базовой линии до верхней границы заглавных букв;
- *жирный* – признак использования жирного начертания шрифта;
- *курсив* – признак использования курсивного начертания шрифта.

6. Алгоритмы рисования на листе

Линии и надписи выполняются пером (его толщина и цвет устанавливаются в команде перо). Для пера определено текущее положение, которое устанавливается в команде в точку. Начальное положение – (0, 0). Если точка, которую нужно отметить, находится вне листа – никаких действий не производится.

6.1. в точку

Синтаксис:

```
алг в точку(цел x, цел y)
```

Перемещает перо рисователя в точку, ничего при этом не рисуя.

- *x* – координата целевой точки по горизонтали;
- *y* – координата целевой точки по вертикали.

6.2. линия

Синтаксис:

```
алг линия(цел x1, цел y1, цел x2, цел y2)
```

Рисует отрезок.

- (x1,y1) – координаты начала отрезка;
- (x2,y2) – координаты конца отрезка.

6.3. линия в точку

Синтаксис:

```
алг линия в точку(цел x, цел y)
```

Перемещает перо рисователя в точку, рисуя при этом прямую линию.

- *x* – координата целевой точки по горизонтали;
- *y* – координата целевой точки по вертикали.

6.4. многоугольник

Синтаксис:

```
алг многоугольник(цел n, целтаб xx, целтаб yy)
```

Рисует многоугольник.

- *n* – количество вершин многоугольника;
- *xx* – массив абсцисс вершин;
- *yy* – массив ординат вершин.

Каждая вершина с номером `i` имеет координаты `(xx[i], yy[i])`. Размеры обоих массивов координат должен совпадать с количеством вершин *n*.

6.5. пиксель

Синтаксис:

```
алг пиксель(цел x, цел y, цветзначение)
```

Рисует ровно одну точку размером в один пиксель.

- *x* – координата точки по горизонтали;
- *y* – координата точки по вертикали;
- *значение* – значение цвета точки.

6.6. прямоугольник

Синтаксис:

```
алг прямоугольник(цел x1, цел y1, цел x2, цел y2)
```

Рисует прямоугольник.

- *x1* – координата левой границы прямоугольника;
- *y1* – координата верхней границы прямоугольника;
- *x2* – координата правой границы прямоугольника;
- *y2* – координата нижней границы прямоугольника.

6.7. эллипс

Синтаксис:

```
алг эллипс(цел x1, цел y1, цел x2, цел y2)
```

Рисует эллипс.

- *x1* – координата левой границы прямоугольника, в который вписан эллипс;
- *y1* – координата верхней границы прямоугольника, в который вписан эллипс;
- *x2* – координата правой границы прямоугольника, в который вписан эллипс;
- *y2* – координата нижней границы прямоугольника, в который вписан эллипс.

6.8. окружность

Синтаксис:

```
алг окружность(цел xc, цел yc, цел r)
```

Рисует окружность.

- *xc* – координата центра окружности по горизонтали;
- *yc* – координата центра окружности по вертикали;
- *r* – радиус окружности.

6.9. надпись

Синтаксис:

```
алг надпись(цел x, цел y, лит текст)
```

Рисует надпись.

- *x* – координата левой границы прямоугольника, в который вписана надпись;
- *y* – координата верхней границы прямоугольника, в который вписана надпись;
- *текст* – текст надписи.

6.10. залить

Синтаксис:

```
алг залить(цел x, цел y)
```

Заливает связанную область вокруг точки одним цветом. Область вокруг точки – это связное множество соседних точек и их соседей, которые имеют тот же цвет, что и исходная точка и включают саму точку. Связность понимается только по вертикали и горизонтали.

- *x* – координата исходной точки по горизонтали;
- *y* – координата исходной точки по вертикали.

7. Алгоритмы работы с листами

7.1. новый лист

Синтаксис:

```
алг новый лист(цел ширина, цел высота, цветфон)
```

Создает новый лист для рисования.

- *ширина* – ширина нового листа в пикселях;
- *высота* – высота нового листа в пикселях;
- *фон* – цвет фона нового листа.

7.2. загрузить лист

Синтаксис:

```
алг загрузить лист(лит имя файла)
```

Создает новый лист для рисования, используя в качестве фона внешний графический файл в формате PNG.

- *имя файла* – имя файла с изображением.

7.3. сохранить лист

Синтаксис:

```
алг сохранить лист(лит имя файла)
```

Сохраняет полученный на листе результат во внешний графический файл в формате PNG.

имя файла – имя файла с изображением.

Исполнитель ‹‹Робот››

---

1. Введение

1.1. Обстановки Робота

Исполнитель Робот существует в некоторой *обстановке* – прямоугольном поле, разбитом на клетки, между которыми могут стоять стены. Обстановка, в которой находится Робот, называется *текущей* обстановкой Робота. Кроме того, определена еще одна обстановка Робота – *стартовая* обстановка.Выполнение программы начинается со Стартовой обстановки.

Робот может передвигаться по полю, закрашивать клетки, измерять температуру и радиацию. Робот не может проходить сквозь стены, но может проверять, есть ли рядом с ним стена. Робот не может выйти за пределы прямоугольника (по периметру стоит «забор»). Подробно система команд Робота описана ниже.

Удобно представлять себе, что Робот существует всегда. В частности, когда начинается сеанс работы системы Кумир, Робот уже существует и для него определены и текущая, и стартовая обстановка (они совпадают).

Обстановки Робота могут храниться в файлах специального формата (расширение `.fil`).

1.2. Окно наблюдения за Роботом

В Кумире есть специальное устройство – *Окно наблюдения за Роботом* (иногда для краткости будем говорить *Окно Робота*). В этом окне всегда видна текущая обстановка Робота, включая положение самого Робота.

1.3. Управление Роботом из программы

Кумир-программа, управляющая Роботом, должна начинаться со строки `использовать Робот` (подробнее – см. описание языка Кумир). При выполнении этой строки Кумир помещает Робота в некоторую заранее определенную обстановку. Эта обстановка и называется *стартовой* обстановкой Робота.

Таким образом, в каждый момент сеанса работы системы Кумир определены две обстановки Робота – *текущая* и *стартовая*. Текущая обстановка в любой момент показывается в окне наблюдения за Роботом.

Непосредственное управление Роботом из программы осуществляется с помощью *команд Робота*.

1.4. Как установить стартовую обстановку

В системе Кумир есть средства, с помощью которых Школьник может задать нужную ему стартовую обстановку. Это можно сделать двумя способами. Один способ – загрузить стартовую обстановку из указанного Школьником файла. Другой способ – редактировать существующую стартовую обстановку с помощью специального редактора стартовых обстановок.

Редактор стартовых обстановок является частью системы Кумир. Редактирование обстановки происходит в окне наблюдения, но при этом окно наблюдения открывается в специальном режиме – режиме редактирования стартовой обстановки. Редактируемая обстановка может быть сохранена в файл или непосредственно использоваться в качестве стартовой обстановки.

1.5. Ручное управление Роботом

В систему Кумир входит пульт ручного управления Роботом . Этот пульт позволяет вручную управлять Роботом – выдавать команды, входящие в систему команд Робота. Использовать пульт можно в любое время, кроме тех временных промежутков, когда происходит непрерывное выполнение Кумир-программы. В частности, Роботом можно управлять с пульта в те моменты, когда выполнение Кумир программы приостановлено (система Кумир находится в состоянии «Пауза»).

2. Окно наблюдения за Роботом

2.1. Видимое и скрытое состояние окна

Окно наблюдения за Роботом создается в момент начала сеанса работы системы Кумир и доступно до окончания сеанса. Во время сеанса работы окно может находиться в одном из двух состояний – видимо или скрыто (с отображением на панели задач или без).

В момент запуска Кумира окно наблюдения за Роботом скрыто. Чтобы сделать окно видимым, Школьник должен нажать кнопку *Робот* на панели инструментов или же кликнуть по соответственному пункту меню **Окна**. Кроме того, окно Робота автоматически становится видимым при запуске на выполнение программы, содержащей строку `использовать Робот`. Окно Робота становится видимым на том же месте, где оно находилось, когда его последний раз сделали скрытым.

2.2. Свойства окна наблюдения за Роботом

На окне наблюдения за Роботом нет ни меню, ни кнопок. Таким образом, в окне Робота есть только полоса заголовка и рабочее поле.

В левой части заголовка есть надпись «Робот», за которой следует имя файла, в котором хранится стартовая обстановка. Если такого файла нет, то вместо имени файла выводится слово «временная». Правила, определяющие привязку стартовой обстановки к определенному файлу, описаны ниже.

Размер окна наблюдения полностью определяется размерами стартовой обстановки. Пользователь не может менять размер окна с помощью стандартных средств, например, манипулируя мышью.

2.3. Режимы окна наблюдения за Роботом

Окно наблюдения за Роботом может находится в двух режимах:

- **обычном** – применяется при наблюдении за Роботом во время исполнения программ или управлении с пульта,
- **редактирования** – применяется при редактировании стартовой обстановки Робота.

2.4. Что входит в описание обстановки Робота

Обстановка Робота представляет собой прямоугольное поле, окруженное забором и разбитое на клетки. Говоря более точно, обстановка описывается следующими величинами:

1. размеры обстановки — количество строк (`1–128`) и количество столбцов (`1–255`);
2. для каждой клетки:

- наличие стен вокруг клетки,
- признак закрашенности,
- уровень радиации (измеряется в условных единицах, может принимать любое вещественное значение от `0` до `99`),
- температура (измеряется в градусах Цельсия, может принимать любое целое значение от `-273` до `+233`).

Примечание. Нижняя возможная температура – это (приблизительно) абсолютный ноль (0 градусов по шкале Кельвина). Верхняя температура – это температура, при которой горят книги (451 градус по Фаренгейту).

Система команд Робота позволяет ему определить значения всех этих характеристик клетки (см. ниже).

Кроме того, в клетке могут быть пометки, видимые наблюдателю, но не доступные «органам чувств» Робота:

- символы в левом верхнем и левом нижнем углах,
- точка в правом нижнем углу.

Частью описания обстановки является и положение Робота. Как и для чтения пометок, у Робота нет средств, чтобы определить свои координаты.

2.5. Изображение текущей обстановки в окне наблюдения

Изображение текущей обстановка по умолчанию полностью помещается в рабочем поле окна наблюдения за Роботом .

Фон рабочего поля — зеленый. Закрашенные клетки – серые. Между клетками тонкие жёлтые линии. Стены (в том числе – «забор» по периметру прямоугольника обстановки) изображаются толстыми желтыми линиями.

Примечание. Все цвета отображения обстановки являются настраиваемыми. Чтобы изменить тот или иной цвет, нужно использовать пункт **настройки** в меню **программа**.

В клетке рабочего поля окна наблюдения Робот изображается ромбиком. Температура и радиация не показываются, они могут быть только измерены Роботом. Символы в клетках, наоборот, видны человеку, но Робот не умеет их считывать.

Масштаб изображения обстановки можно изменить, используя колёсико мыши, предварительно сделав активным окно наблюдения за Роботом.

2.6. Когда и как меняется текущая обстановка Робота

Текущая обстановка изменяется при выполнении команд Робота, подаваемых из программы или с пульта .

Выполнение команд Робота влияет на текущую обстановку следующим образом. Команды перемещения и закрашивания отображаются в текущей обстановке естественно.

Если команда перемещения выдает отказ, то текущая обстановка не изменяется, а на экране (если он виден) соответствующий угол Робота закрашивается красным. Красный цвет снимается:

- при выполнении команды `использовать Робот`,
- при принудительном помещении Робота в стартовую обстановку (см. ниже).

Команды проверок и измерения радиации и температуры на текущую обстановку не влияют.

Кроме того, в двух случаях происходит принудительное помещение Робота в стартовую обстановку (текущая обстановка становится равной стартовой). Это происходит:

- при запуске Кумир-программы, которая использует Робот (окно наблюдения при этом становится видимым, даже если оно было скрыто);
- при изменении имени файла стартовой обстановки Робота (в этой ситуации невидимое окно остается невидимым).

3. Команды управления Роботом из программы

3.1. Общие сведения

Кумир-программа, управляющая Роботом, должна начинаться со строки `использовать Робот` (подробнее – см. описание языка Кумир). При выполнении этой строки Кумир помещает Робота в *стартовую обстановку*.

Система команд исполнителя «Робот» включает:

- 5 команд, вызывающих действия Робота (`влево`, `вправо`, `вверх`, `вниз`, `закрасить`);

- 10 команд проверки условий:
  - 8 команд вида [слева/ справа/ сверху/ снизу] [стена/ свободно],
  - 2 команды вида клетка [закрашена/ чистая];
- 2 команды измерения (температура, радиация).

Командам влево, вправо, вверх, вниз, закрасить соответствуют алгоритмы-процедуры языка КуМир. Остальным командам соответствуют алгоритмы-функции, тип этих функций указан ниже.

3.2. Команды-действия

3.2.1. влево

Синтаксис:

```
алг влево
```

Перемещает Робота на одну клетку влево. Если слева стена, выдает отказ.

3.2.2. вправо

Синтаксис:

```
алг вправо
```

Перемещает Робота на одну клетку вправо. Если справа стена, выдает отказ.

3.2.3. вверх

Синтаксис:

```
алг вверх
```

Перемещает Робота на одну клетку вверх. Если сверху стена, выдает отказ.

3.2.4. вниз

Синтаксис:

```
алг вниз
```

Перемещает Робота на одну клетку вниз. Если снизу стена, выдает отказ.

3.2.5. закрасить

Синтаксис:

```
алг закрасить
```

Делает клетку, в которой находится робот, закрашенной.

Ниже приведён пример использования команд-действий Робота.

Скопировать пример

```
использовать Робот

алг
нач
  вправо
  вниз
  влево
  вверх
  закрасить
кон
```

Пример 1. Использование команд-действий Робота

3.3. Команды-проверки

3.3.1. слева свободно

Синтаксис:

```
алг лог слева свободно
```

Возвращает **да**, если Робот может перейти влево, иначе – **нет**.

3.3.2. справа свободно

Синтаксис:

```
алг лог справа свободно
```

Возвращает **да**, если Робот может перейти вправо, иначе – **нет**.

3.3.3. сверху свободно

Синтаксис:

```
алг лог сверху свободно
```

Возвращает **да**, если Робот может перейти вверх, иначе – **нет**.

3.3.4. снизу свободно

Синтаксис:

```
алг лог снизу свободно
```

Возвращает **да**, если Робот может перейти вниз, иначе – **нет**.

3.3.5. слева стена

Синтаксис:

```
алг лог слева стена
```

Возвращает **да**, если слева от Робота находится стена, иначе – **нет**.

3.3.6. справа стена

Синтаксис:

```
алг лог справа стена
```

Возвращает **да**, если справа от Робота находится стена, иначе – **нет**.

3.3.7. сверху стена

Синтаксис:

```
алг лог сверху стена
```

Возвращает **да**, если сверху от Робота находится стена, иначе – **нет**.

3.3.8. снизу стена

Синтаксис:

```
алг лог снизу стена
```

Возвращает **да**, если снизу от Робота находится стена, иначе – **нет**.

3.3.9. клетка закрашена

Синтаксис:

```
алг лог клетка закрашена
```

Возвращает **да**, если клетка закрашена, иначе – **нет**.

3.3.10. клетка чистая

Синтаксис:

```
алг лог клетка чистая
```

Возвращает **нет**, если клетка закрашена, иначе – **да**.

3.4. Команды-измерения

3.4.1. радиация

Синтаксис:

```
алг вещ радиация
```

Возвращает значение радиации в клетке, где находится Робот.

3.4.2. температура

Синтаксис:

```
алг вещ температура
```

Возвращает значение температуры в клетке, где находится Робот.

4. Стартовая обстановка, ее изменение и связь с текущей обстановкой

4.1. Вводные сведения

Стартовая обстановка – это обстановка, в которую Робот будет помещен при запуске Кумир-программы, т. е. при выполнении команды **использовать** Робот.

Со стартовой обстановкой связана специальная настройка Кумира – текстовая строка, в которой записано имя некоторого файла, содержащего описание обстановки Робота. Как правило, в качестве стартовой настройки используется обстановка, хранящаяся в этом файле. Исключения описаны в следующей главе.

4.2. Как определяется стартовая обстановка

Как правило, стартовая обстановка — это обстановка, хранящаяся в *файле стартовой обстановки* (для краткости – *ФСО*).

Из этого правила есть два исключения. Одно из них связано с тем, что файл с указанным именем может не содержать корректного описания обстановки или вообще не существовать. Этот случай рассмотрен ниже в следующей главе.

Другое исключение связано с возможностью упрощенного внесения временных изменений в стартовую обстановку. А именно, если в данный момент в Кумир-системе происходит редактирование файла обстановки, то текущее промежуточное значение этой редактируемой обстановки считается временной стартовой обстановкой. Иными словами, действия пользователя по подготовке обстановки имеют приоритет над содержимым строки с именем ФСО.

4.3. Изменение имени файла стартовой обстановки

Общие сведения

Школьник может изменить файл стартовой обстановки (ФСО) двумя способами:

- с помощью команды «Загрузить обстановку» меню Робота,
- с помощью редактора стартовых обстановок.

Состояние окна наблюдения (видимо/скрыто) при этих действиях не меняется.

При изменении строки с именем ФСО, новая стартовая обстановка становится текущей, что отражается в окне наблюдения за Роботом.

Команда «Загрузить обстановку»

Изменение строки с именем ФСО происходит с помощью стандартного диалога выбора файла. При этом в качестве каталога по умолчанию предлагается каталог текущего файла стартовой обстановки. Если выбранный файл существует и содержит корректную обстановку, то эта обстановка считается стартовой. Она же объявляется текущей, т. е. Робот помещается в эту обстановку. Это отображается в окне наблюдения Робота. Имя файла стартовой обстановки (без директории) показывается в левой части заголовка окна наблюдения за Роботом.

Если с чтением обстановки из выбранного файла возникают какие-либо проблемы, пользователю выдаётся сообщение ‹‹Ошибка открытия файла››. При этом ни стартовая, ни текущая обстановки Робота, а также имя ФСО не изменяются.

Изменение стартовой обстановки с помощью редактора стартовых обстановок Робота

Редактор стартовых обстановок Робота – составная часть системы Кумир. Он позволяет добавлять и удалять стены, менять размеры обстановки и т. п. Изменённая обстановка объявляется стартовой. Её можно сохранить в новый файл, а можно перезаписать текущий .

4.4. Создание новой стартовой обстановки

Для создания новой стартовой обстановки Робота нужно воспользоваться коммандой «Новая обстановка» меню Робот. При этом откроектся специальное окно, где можно будет ввести размер обстановки. После настройки размера и нажатия кнопки «Ок», новая обстановка сразу открывается в режиме редактирования .

4.5. Начальная установка имени файла стартовой обстановки

Запуск системы Кумир

Система Кумир при окончании сеанса работы запоминает в своих настройках значение имени файла со стартовой обстановкой. При новом запуске система Кумир проверяет, существует ли файл с запомненным именем. Если файл существует и содержит корректную обстановку, то эта обстановка считается стартовой. Она же объявляется *текущей*, т. е. Робот помещается в эту обстановку. Это отображается в окне наблюдения Робота. Имя файла стартовой обстановки (без директории) показывается в левой части заголовка окна наблюдения за Роботом.

Если с чтением обстановки из выбранного файла возникают какие-либо проблемы, то стартовой (и одновременно текущей) объявляется стандартная обстановка, хранящаяся в Кумире. В левой части заголовка окна наблюдения за Роботом появляется предупреждающая надпись – «нет файла». Никакой файл с обстановкой при этом не создается, а предыдущее значение имени файла стартовой обстановки не изменяется.

Установка имени файла стартовой обстановки при инсталляции системы Кумир

При инсталляции системы Кумир в качестве имени ФСО записывается имя файла со стандартной обстановкой, который входит в поставку Кумира (`7×7.fil`). Стандартной обстановкой является *пустая* обстановка размера `7×7` с Роботом в левом верхнем углу. Максимально допустимый размер обстановки составляет `128×255`. Слово «пустая» означает:

- в обстановке нет стен (кроме проходящего по периметру забора),
- в клетках нет точек и символов,
- во всех клетках радиация и температура установлены в `0`.

5. Редактирование стартовой обстановки

5.1. Общие сведения

Редактор стартовых обстановок можно использовать:

- для подготовки новых обстановок,
- при отладке Кумир-программ для оперативного внесения небольших изменений в стартовую обстановку.

Запуск редактирования производится с помощью команды ‹‹Редактировать обстановку›› меню Робота. По этой команде окно наблюдения переходит в специальный режим – *режим редактирования стартовой обстановки* (для краткости – *режим редактирования*). Завершить редактирование обстановки можно двумя способами:

- снять галочку с пункта *Редактирование* меню Робота,
- запустить на исполнение любую Кумир-программу, использующую Робот. Тогда окно наблюдения снова перейдёт в обычный режим.

В обоих случаях результаты редактирования сохранятся и стартовая обстановка Робота изменится.

Свойства окна наблюдения в режиме редактирования аналогичны свойствам окна наблюдения за Роботом. Основных отличий – два:

- фон рабочего поля — синий,

- в левом верхнем углу окна есть инструментальные кнопки, позволяющие изменять параметры клетки обстановки.

5.2. Главное меню окна наблюдения в режиме редактирования

Кнопки редактирования обстановки

Опишем кратко назначение кнопок редактирования обстановки по порядку слева направо.

- **Кнопка редактирования меток клетки.** При нажати на эту кнопку в каждой клетке появляются два поля для ввода текста меток (одно вверху клетки, одно внизу). В каждое из этих полей можно ввести только один символ.

- **Кнопка редактирования уровня радиации.** При нажати на эту кнопку справа от кнопок настройки появляется поле для редактирования уровня радиации в клетке. Значение по умолчанию в этом поле равно `55,00`. Это значение можно изменять, кликая на верхний или нижний треугольник в этом поле. Также можно ввести своё собственное значение. Уровень радиации в клетке может колеблется в диапазоне от `0,00` до `99,00`. После получения нужного числа в поле настройки уровня для применения настроек к выбранной клетке нужно просто кликнуть по ней левой кнопкой мыши.

- **Кнопка редактирования уровня температуры.** Механизм редактирования температуры в клетке поностью аналогичен механизму редактирования уровня радиации. Значение по умолчанию в поле редактирования температуры равно `77`. Температура в клетке может колебаться от `-273` до `233`.

5.3. Непосредственное редактирование обстановки

Основное редактирование обстановок Робота ведется непосредственно. Предусмотрены следующие операции непосредственного редактирования:

- **поставить/убрать стену** – щелкнуть по границе между клетками,
- **закрасить/сделать чистой клетку** – щелкнуть по клетке,
- **поставить/убрать точку** – щелкнуть по клетке при нажатой клавише Ctrl,
- **переместить Робота** – тащить мышью,
- **установить температуру, радиацию, метки** – использовать соответственные кнопки в левом верхнем углу окна наблюдения в режиме редактирования обстановки.

5.4. Операции с файлами обстановок

Перечислим все операции с файлами обстановок (расширение `.fil`), предусмотренные в системе Кумир. Соответствующие им команды входят в меню «Робот» главного окна (команды «Сохранить обстановку» и «Загрузить обстановку»).

Система Кумир запоминает файлы, к которым применялись указанные операции. Список последних таких файлов доступен с помощью команды «Недавние обстановки» меню «Робот». Во всех перечисленных ниже операциях с файлами обстановок в качестве директории по умолчанию предлагается директория, использовавшаяся в последней по времени выполнения операции с файлами обстановок.

6. Пульт Робота

6.1. Общие сведения

Пульт Робота располагается в отдельном окне и предназначен для управления Роботом. С помощью пульта можно передавать команды Роботу. Он непосредственно выполняет эти команды.

Пользоваться пультом можно как при видимом окне наблюдения за Роботом, так и когда это окно скрыто (управление Роботом вслепую).

Пульт влияет только на поведение Робота внутри текущей обстановки и на отображение этого в окне наблюдения за Роботом. В частности, манипуляции с пультом Робота не влияют на свернутость/развернутость окна наблюдения за Роботом и обстановку в окне редактора стартовых обстановок.

Окно Пульта создается при запуске системы Кумир. Свойства окна таковы:

- при запуске системы КуМир пульт скрыт,
- пользователь не может менять размеры окна пульта,

Чтобы сделать пульт видимым, нужно воспользоваться командой «Робот-Пульт» меню «Робот» или соответствующей инструментальной клавишей.

Связь Пульта с Роботом возможна всегда, кроме тех моментов, когда идет непрерывное выполнение программы, использующей Робот (т. е. программы, которая содержит строку `использовать Робот`). В частности, не имеет значения, идет редактирование стартовой обстановки или нет. О наличии связи свидетельствуют индикаторы в левом верхнем углу пульта.

6.2. Общий вид Пульта

На пульте Робота есть:

- лампочки-индикаторы:
    - зелёная лампочка-индикатор Связь установлена ,
    - красная лампочка-индикатор Нет связи ;
- табло для семи последних выполненных команд, разделенное на левую часть (собственно команда) и правую часть (реакция Робота);
- вспомогательные кнопки табло:
    - Сброс табло и Скопировать в буфер – кнопки очистки и копирования, находящиеся справа от табло,
    - Прокрутка табло – две кнопки сверху и снизу;
- кнопочная панель из 9 кнопок, служащая для передачи команд Роботу.

6.3. Кнопочная панель. Передача команд Роботу

На панели есть:

- 4 кнопки-стрелки (они соответствуют командам передвижения Робота);
- кнопка закрасить (в центре);
- кнопки температура и радиация (в левом верхнем и левом нижнем углах);

- две кнопки-префикса Стена/Закрашено и Свободно/Чисто (в правом верхнем и левом нижнем углах);

Для передачи команды действия (вверх, вниз, вправо, влево, закрасить, радиация, температура) достаточно нажать соответствующую кнопку.

Чтобы передать команду проверки нужно последовательно нажать две кнопки: сначала – префикс (Стена/Закрашено и Свободно/Чисто), а потом — основную (закрасить или одну из стрелок).

Примеры:

1. Чтобы передать Роботу команду проверки слева свободно? нужно сначала нажать кнопку-префикс Свободно/Чисто , а затем кнопку влево .
2. Чтобы передать Роботу команду проверки клетка закрашена? нужно сначала нажать кнопку-префикс Стена/Закрашено , а затем кнопку закрасить .

После того, как нажата кнопка-префикс, она меняет цвет. Если после кнопки-префикса нажата кнопка, отличная от стрелок и кнопки закрасить , то исходная кнопка-префикс просто будет забыта.

6.4. Использование табло

На табло выводятся все команды, передаваемые Роботу вместе с реакцией Робота на эти команды. В ответ на команды перемещения в правой части появляется надпись «ОК» или «Отказ». Если с Роботом нет связи, на табло выдается «Нет связи».

Вывод на табло при первом включении Пульта в сеансе работы системы Кумир или после нажатия кнопки Сброс табло начинается с верхней строки. Когда табло заполнилось, начинается прокрутка и вывод идет в последнюю строку табло. Выше и ниже табло есть две кнопки для прокрутки его содержимого на одну строку вверх/вниз.

7. Робот и главное меню

7.1. Общие сведения

К Роботу относится меню Робот главного окна системы «Кумир» (полностью), а также две команды из меню Окна . Для действий Показать поле Робота и Пульт Робота есть инструментальные кнопки на панели кнопок.

Ниже отдельно описана каждая из команд меню Робот .

7.2. Команды меню «Робот»

- **Загрузть обстановку** – открывает диалог выбора ФСО.
- **Недавние обстановки** – выводит список последних использовавшихся ФСО.
- **Вернуть исходную обстановку** – делает текущей стартовую обстановку.
- **Сохранить обстановку** – открывает диалог сохраниения обстановки в файл.
- **Редактировать обстановку** – переводит окно наблюдения в режим редактирования стартовой обстановки.

- **Авто размер окна** – устанавливает масштаб отображения обстановки по умолчанию.
- **Новая обстановка** – открывает окно создания обстановки.

7.3. Команды меню «Окна»

- **Робот** – открывает окно наблюдения за исполнителем Робот.
- **Робот - Пульт** – открывает окно пульта исполнителя Робот.

8. Алгоритмы контроля обстановки исполнителя «Робот»

Алгоритмы, описанные ниже могут быть вызваны только из ‹‹скрытой›› части программы. Они служат для получении различных данных о текущей обстановке исполнителя **Робот**.

8.1. @@размер поля

Синтаксис:

```
алг @@размер поля(рез цел строк, рез цел столбцов)
```

После выполнения данного алгоритма в переменные *строк* и *столбцов* будут записаны соответствующие размеры обстановки.

8.2. @@закрашена

Синтаксис:

```
алг лог @@закрашена(цел строка, цел столбец)
```

Возвращает да если клетка, расположенная в строке *строка* и столбце *столбец* закрашена.

8.3. @@метка

Синтаксис:

```
алг лог @@метка(цел строка, цел столбец)
```

Возвращает да если клетка, расположенная в строке *строка* и столбце *столбец* отмечена точкой.

8.4. @@робот

Синтаксис:

```
алг @@робот(рез цел строка, рез цел столбец)
```

После выполнения данного алгоритма в переменные *строка* и *столбец* будет записано текущее положение Робота.

8.5. @@верхняя буква

Синтаксис:

```
алг лог @@верхняя буква(цел строка, цел столбец)
```

Возвращает верхний символ из клетки *строка*/ *столбец*. Если верхнего символа в клетке нет – возвращает пробел.

8.6. @@нижняя буква

Синтаксис:

```
алг лог @@нижняя буква(цел строка, цел столбец)
```

Возвращает нижний символ из клетки *строка*/ *столбец*. Если нижнего символа в клетке нет – возвращает пробел.

8.7. @@температура

Синтаксис:

```
алг лог @@температура(цел строка, цел столбец)
```

Возвращает значение температуры в клетке *строка*/ *столбец*.

8.8. @@радиация

Синтаксис:

```
алг лог @@радиация(цел строка, цел столбец)
```

Возвращает уровень радиации в клетке *строка*/ *столбец*.

Исполнитель ‹‹Черепаха››

---

1. Использование исполнителя

Исполнитель ‹‹Черепаха›› входит в Вашу поставку Кумир, но его функции не являются частью языка программирования. Для его использования необходимо в программе явно указать использование данного исполнителя:

```
использовать Черепаха
| теперь функции черепахи доступны
| для использования в программе
```

2. Команды действий

2.1. поднять хвост

Синтаксис:

```
алг поднять хвост
```

Черепаха поднимает хвост. Теперь при перемещении Черепаха *не будет* чертить линию.

2.2. опустить хвост

Синтаксис:

```
алг опустить хвост
```

Черепаха опускает хвост. Теперь при перемещении Черепаха *будет* чертить линию.

2.3. вперед

Синтаксис:

```
алг вперед(цел a)
```

Черепаха перемещается вперёд на заданное количество точек (пикселей).

- *а* – количество пикселей, на которое переместится Черепаха.

2.4. назад

Синтаксис:

```
алг назад(цел а)
```

Черепаха перемещается назад на заданное количество точек (пикселей).

- *а* – количество пикселей, на которое переместится Черепаха.

2.5. влево

Синтаксис:

```
алг влево(цел угол)
```

Черепаха поворачивается влево на заданный угол.

- *угол* – значение угла (в градусах), на который повернётся Черепаха.

2.6. вправо

Синтаксис:

```
алг вправо(цел угол)
```

Черепаха поворачивается вправо на заданный угол.

угол – значение угла (в градусах), на который повернётся Черепаха.

Исполнитель ‹‹Водолей››

---

1. Использование исполнителя

Исполнитель ‹‹Водолей›› входит в Вашу поставку Кумир, но его функции не являются частью языка программирования. Для его использования необходимо в программе явно указать использование данного исполнителя:

```
использовать Водолей
| теперь функции водолея доступны
| для использования в программе
```

2. Команды действий

2.1. наполни A

Синтаксис:

```
алг наполни A
```

Наполняет колбу A.

2.2. наполни B

Синтаксис:

```
алг наполни B
```

Наполняет колбу B.

2.3. наполни C

Синтаксис:

```
алг наполни C
```

Наполняет колбу C.

2.4. вылей A

Синтаксис:

```
алг вылей A
```

Выливает всю жидкость из колбы A.

2.5. вылей B

Синтаксис:

```
алг вылей B
```

Выливает всю жидкость из колбы B.

2.6. вылей C

Синтаксис:

```
алг вылей C
```

Выливает всю жидкость из колбы C.

2.7. перелей из A в B

Синтаксис:

```
алг перелей из A в B
```

Переливает жидкость из колбы A в колбу B до тех пор, пока не случится одно из событий (или оба одновременно):

- жидкость в колбе A закончилась,
- колба B заполнилась.

2.8. перелей из A в C

Синтаксис:

```
алг перелей из A в C
```

Переливает жидкость из колбы A в колбу C до тех пор, пока не случится одно из событий (или оба одновременно):

- жидкость в колбе A закончилась,
- колба C заполнилась.

2.9. перелей из B в A

Синтаксис:

```
алг перелей из B в A
```

Переливает жидкость из колбы B в колбу A до тех пор, пока не случится одно из событий (или оба одновременно):

- жидкость в колбе B закончилась,
- колба A заполнилась.

2.10. перелей из B в C

Синтаксис:

```
алг перелей из B в C
```

Переливает жидкость из колбы B в колбу C до тех пор, пока не случится одно из событий (или оба одновременно):

- жидкость в колбе B закончилась,
- колба C заполнилась.

2.11. перелей из C в A

Синтаксис:

```
алг перелей из C в A
```

Переливает жидкость из колбы C в колбу A до тех пор, пока не случится одно из событий (или оба одновременно):

- жидкость в колбе C закончилась,
- колба A заполнилась.

2.12. перелей из C в B

Синтаксис:

```
алг перелей из C в B
```

Переливает жидкость из колбы C в колбу B до тех пор, пока не случится одно из событий (или оба одновременно):

- жидкость в колбе C закончилась,
- колба B заполнилась.

3. Алгоритмы контроля обстановки

3.1. размер A

Синтаксис:

```
алг цел размер A
```

Возвращает объём колбы A.

3.2. размер B

Синтаксис:

```
алг цел размер B
```

Возвращает объём колбы B.

3.3. размер C

Синтаксис:

```
алг цел размер C
```

Возвращает объём колбы C.

3.4. в сосуде A

Синтаксис:

```
алг цел в сосуде A
```

Возвращает количество жидкости в сосуде A.

3.5. в сосуде B

Синтаксис:

```
алг цел в сосуде B
```

Возвращает количество жидкости в сосуде B.

3.6. в сосуде C

Синтаксис:

```
алг цел в сосуде C
```

Возвращает количество жидкости в сосуде C.

3.7. @решено

Синтаксис:

```
алг лог @решено
```

Возвращает **да**, если хотя бы в одной колбе имеется ровно столько жидкости, сколько требуется в данной обстановке.