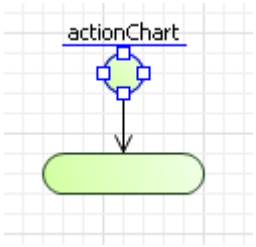
Диаграмма действий



Элемент диаграммы действий.

Создание новой диаграммы действий

Чтобы создать новую диаграмму действий

1. Перетащите элемент **Диаграмма действий** из палитры **Диаграмма действий** на диаграмму типа агентов.
2. Будет нарисована простейшая диаграмма действий, состоящая из начальной точки (задаваемой блоком "диаграмма действий") и блока "вернуть значение".
3. Измените имя диаграммы действий (прямо в текстовом редакторе, который активируется в графическом редакторе при добавлении нового элемента). Это имя будет использоваться для вызова функции, заданной этой диаграммой действий.
4. Начальная точка (элемент "диаграмма действий") задает основные свойства диаграммы - ее тип возвращаемого значения, аргументы, уровень доступа и т.д. Чтобы задать эти свойства, перейдите в панель **Свойства**.
5. Если функция, задаваемая вашей диаграммой действий, только выполняет какие-то действия, но не возвращает никакого результата проведенных вычислений, укажите, что эта функция не возвращает ничего, выбрав в группе кнопок **Действие (не возвращает ничего)**.
6. Если вам нужно, чтобы диаграмма действий возвращала результат проведенных ею вычислений, то вам нужно будет указать, какого типа будет возвращаемое значение. Диаграмма действий может возвращать как значение одного из наиболее часто используемых типов (**int**, **double**, **boolean**, **String**), так и значение любого другого Java класса (в этом случае вам нужно будет выбрать опцию **Другой** и ввести имя типа в расположенном справа поле).
7. Диаграмма действий может быть объявлена *статической*. Статическая диаграмма действий не требует создания экземпляров того типа агентов, в котором она задана. Статическая диаграмма действий мояфункция, объявленная в типе агента MyClass, будет доступна из любого места модели как MyClass.мояфункция(). Чтобы сделать диаграмму действий статической, установите флажок **Статическая** в секции свойств **Специфические**.

8. Вы закончили создание простейшей диаграммы действий. Теперь вам будет нужно добавить в созданную структуру другие блоки диаграммы действий согласно задаваемому вами алгоритму.

Свойства

Основные

Имя – Имя диаграммы действий. Это имя будет использоваться для вызова функции, заданной этой диаграммой действий.

Отображать имя – Если опция выбрана, то имя диаграммы действий будет отображаться в графическом редакторе.

Исключить – Если опция выбрана, то диаграмма действий будет исключена из модели.

Видимость – Если опция выбрана, то диаграмма действий будет отображаться на презентации во время выполнения модели.

Действие (не возвращает значения) – Если опция выбрана, то заданная диаграммой действий функция не будет возвращать значения в результате своего выполнения.

Возвращает значение – Если опция выбрана, то заданная диаграммой действий функция будет возвращать результат проведенных ею вычислений.

Тип возвращаемого значения – Тип возвращаемого диаграммой действий значения. Функция может возвращать как значение одного из наиболее часто используемых типов (**int**, **double**, **boolean**, **String**), так и объект любого Java класса (в этом случае вам нужно будет выбрать опцию **Другой** и ввести имя типа в расположенном справа поле).

Аргументы – Здесь вы можете задать аргументы функции, с помощью которых вы сможете передавать функции данные, необходимые для вычислений. Каждый аргумент задается в отдельной строке таблицы.

Специфические

Цвет заливки – Задает цвет заливки блока. Щелкните мышью внутри элемента управления и выберите нужный цвет из списка наиболее часто используемых цветов или же выберите любой другой цвет с помощью диалога Цвета.

Статическая – Если опция выбрана, то функция, заданная данной диаграммой действий, будет *статической*. Статическая функция не требует создания экземпляров того типа агентов, в котором она задана. Статическая диаграмма действий мояфункция, объявленная в типе агента MyClass, будет доступна из любого места модели как MyClass.мояфункция().

Уровень доступа – Уровень доступа к задаваемой диаграммой действий функции. Есть четыре уровня доступа:
private – функция доступна только из этого типа агентов
protected - функция доступна из этого типа агентов и его подклассов
default - функция доступна из любого места модели
public - функция доступна из всех открытых моделей.

Единицы измерения (сист. динамика) – Если опция выбрана, то вы можете задать единицы измерения для возвращаемого диаграммой действий значения.

Вызов диаграммы действий

Для вызова заданной с помощью диаграммы действий функции используется тот же синтаксис, что и для самих функций. Вы должны указать имя диаграммы действий, за которым следуют скобки, в которых через запятую перечисляются передаваемые диаграмме значения аргументов (если они есть), например:

```
мояФункция()

move(15, 35)
```

Аргументы диаграммы действий

Если вам будет нужно передавать заданной с помощью диаграммы действий функции какие-то данные, необходимые для проведения вычислений, вы можете использовать для этого *аргументы*.

Чтобы задать аргументы диаграммы действий

1. Выберите элемент **Диаграмма действий** в графическом редакторе или в панели **Проекты**.
2. Задайте аргументы в таблице **Аргументы** в панели **Свойства**. Каждый аргумент задается в отдельной строке таблицы.
3. Введите имя аргумента в ячейке **Имя**.
4. Укажите тип аргумента в ячейке **Тип**. Щелкните мышью в ячейке и выберите нужный вам тип из выпадающего списка, либо же введите его самостоятельно.

Если вы зададите аргументы, то при каждом вызове диаграммы действия вам будет нужно передавать ей значения этих аргументов (в том же порядке, в каком они заданы в таблице **Аргументы**).

AnyLogic позволяет изменять порядок следования аргументов в таблице с помощью кнопок  и  .

Чтобы удалить аргумент, выделите соответствующую строку в таблице и щелкните по кнопке  .

Конвертация диаграммы действий в функцию

Если объем кода, заданный в вашей диаграмме действий с помощью элементов **Код**, достаточно существенен, вам может быть удобнее продолжить работу над этим алгоритмом не как с диаграммой действий, а как с обычной функцией AnyLogic (в текстовом виде). Для таких случаев AnyLogic предлагает возможность преобразования диаграммы действия в функцию (обратная конвертация невозможна).

Чтобы преобразовать диаграмму действий в функцию

1. Щелкните правой кнопкой мыши по элементу **Диаграмма действий** в графическом редакторе.
2. Выберите команду **Преобразовать в функцию** из контекстного меню. Вы увидите значок функции, появившийся рядом с диаграммой действий в графическом редакторе. Имя созданной функции будет отличаться от имени диаграммы действий последней цифрой.
3. Если вы уверены, что хотите продолжить работу над алгоритмом как с функцией, удалите вашу диаграмму действий.
4. Измените имя функции, чтобы оно соответствовало имени удаленной диаграммы действий.
5. Продолжите работу над алгоритмом в кодовом редакторе, расположенном в секции свойств **Тело функции** созданной функции.

6. Диаграммы действий. Визуальное задание алгоритмов

7. Сложное имитационное моделирование не может существовать без возможности задания алгоритмов, обычно выполняющих определенную обработку данных или вычисления. AnyLogic поддерживает *диаграммы действий* - структурированные блок-схемы, позволяющие задавать алгоритмы графически в стиле структурированного программирования. Мы используем широко известное расширение подхода, предложенного в свое время Дейкстра. Суть подхода состоит в том, что алгоритмы разбиваются в подразделы с одной точкой входа. Утверждается, что трех способов объединения программ — упорядочения, повторения и выбора — достаточно для задания алгоритма любой сложности. Такой стиль

сводит понимание целого алгоритма к пониманию составляющих его частностей.

8. Диаграммы действий облегчают задание алгоритмов, делая необязательным знание синтаксиса Java операторов.
9. По сути диаграммы действий используются для визуального задания функций, и вы можете использовать для этого обычные функции, как вы и делали раньше. Но использование диаграмм действий дает еще одно преимущество: с их помощью вы можете визуализировать алгоритмы, делая их более понятными для других пользователей модели.
10. Стоит отметить, что по мере развития языка программирования Java, появляется все большее количество операторов Java, которые не могут быть представлены в диаграмме действий AnyLogic никак иначе, как написанием Java кода в блоках-"заглушках" **Код**. В этом случае логика работы данных сегментов кода никак не визуализируется, и смысл использования именно диаграммы действия для задания алгоритма, теряется. В таких случаях мы советуем сконвертировать вашу диаграмму действий в обычную функцию AnyLogic (обратная конвертация не предусмотрена).
11. Диаграмма действий собирается из блоков, расположенных на странице **Диаграмма действий** панели **Палитра**:

Диаграмма действий

Код	Блок **Код** позволяет добавлять в вашу диаграмму действий фрагменты кода. С помощью таких блоков вы можете задавать какие-то определенные действия, которые вы хотите выполнить в процессе выполнения алгоритма.
Решение (If.. Else)	Блок **Решение (If.. Else)** является простейшим способом ветвления алгоритма. Он обеспечивает выполнение фрагментов кода в соответствии с условием. У блока есть две исходящие ветви - *true* и *false*. С помощью других блоков диаграммы действий вы можете задать последовательность действий для каждой из этих ветвей. Когда управление дойдет до данного блока, будет приниматься решение о том, по какой ветви блока управление пойдет дальше. Если заданное для блока *условие* будет выполнено, то будет выбрана ветвь *true*. В противном случае - ветвь *false*.

 Локальная переменная	Используется для объявления новой локальной переменной в диаграмме действий. Локальная переменная будет видна не во всей диаграмме действий, а только в той ее части, которая следует за точкой объявления переменной.
 Цикл While	**Цикл While** является одним из трех блоков диаграммы действий, предназначенных для реализации циклов итераций. Циклы необходимы для того, чтобы повторить некоторые действия несколько раз. **Цикл While** выполняется до тех пор, пока заданное для этого цикла *условие* будет истинно (принимает значение true). Как только условие принимает значение "ложно", цикл завершается и идёт переход к следующему блоку диаграммы действий. **Цикл While** очень похож на **Цикл Do While**. Единственным отличием является то, что истинность выражения проверяется не в конце каждой итерации, а в начале. Следовательно, **Цикл While** может не выполниться ни разу (истинность выражения проверяется в начале каждой итерации, и если с самого начала значением выражения будет ложно, то выполнение цикла будет сразу же прекращено).
 Цикл Do While	**Цикл Do While** является одним из трех блоков диаграммы действий, предназначенных для реализации циклов итераций. Циклы необходимы для того, чтобы повторить некоторые действия несколько раз. **Цикл Do While** выполняется до тех пор, пока заданное для этого цикла *условие* будет истинно (принимает значение true). Как только условие принимает значение "ложно", цикл завершается и идёт переход к следующему блоку диаграммы действий. **Цикл Do While** очень похож на **Цикл While**. Единственным отличием является то, что истинность выражения проверяется не в начале каждой итерации, а в конце. Следовательно, первая итерация цикла **Do While** выполнится обязательно (истинность выражения првоеряется только в конце итерации).

	Цикл For	Цикл. Есть две формы цикла: **Итератор по коллекции**: итеративно проходит по всем элементам указанной коллекции. На каждой итерации выполняется заданное действие, в котором доступен очередной элемент коллекции. **Цикл со счетчиком**: выполняет заданные для этого цикла действия несколько раз, до тех пор, пока не выполнится заданное условие.
	Вернуть значение (Return)	Блок **Вернуть значение (Return)** играет две роли: во-первых, он определяет, какое значение будет возвращать диаграмма действия (если ее *тип возвращаемого значения* не *void*), и во-вторых, немедленно возвращает это значение, завершая тем самым процесс.
	Выход из цикла (Break)	Блок **Выход из цикла** управляет выполнением цикла. Он останавливает текущую итерацию цикла (и опционально также выходит из этого цикла, не выполняя не только текущую, но и оставшиеся итерации).

12. Чтобы нарисовать диаграмму действий, вначале нужно поместить на диаграмму типа агентов элемент  **Диаграмма действий**. При этом будет создана базовая конструкция, состоящая из начальной точки (собственно блока **Диаграмма действий**) и блока **Вернуть значение (Return)**. После этого вы можете добавлять согласно задаваемому вами алгоритму другие блоки диаграммы действий в созданную структуру.
13. **Вызов диаграммы действий**
14. Для вызова заданной с помощью диаграммы действий функции используется тот же синтаксис, что и для самих функций. Вы должны указать имя диаграммы действий, за которым следуют скобки, в которых через запятую перечисляются передаваемые диаграмме значения аргументов (если они есть), например:
15. мояФункция()
16. move(15, 35)