**Общие сведения о языке программирования**

Язык программирования предназначен для создания программ, описывающих функционирование типового блока «Язык программирования», а также для задания скриптов и переменных в рамках проекта, субмодели или графического изображения блока или видеокадра.

При помощи языка программирования в окне «Скрипт страницы» можно написать программу, которая может производить по мере расчета манипуляции с объектами схемы. Собственный скрипт страницы присутствует на каждом уровне вложенности субмоделей и у каждой из них может быть записана своя локальная программа (скрипт) для обработки блоков на текущем уровне, для формирования значения необходимых переменных и др. Кроме страницы субмодели, свой скрипт могут иметь графические изображения блоков и видеокадров. Каждый блок на схеме дополнительно имеет возможность задавать до 3 скриптов, один из которых выполняется при инициализации блока (OnInitScript), другой на каждом шаге расчета (OnRunScript) и третий – при удалении блока (OnDestroyScript). Секции для задания соответствующих скриптов находятся на вкладке «Общие», окна редактирования свойств блока.

Идентификаторы (имена констант, переменных, меток, функций и процедур) в рамках языка программирования могут содержать буквы латинского и русского алфавита, знак подчеркивания «\_»и цифры. Идентификатор должен начинаться с буквы или знака подчеркивания и может иметь произвольную длину. Не допускается использовать в качестве идентификаторов ключевые слова (эти слова автоматически выделяются при вводе). Строчные и прописные буквы по умолчанию в идентификаторах не различаются (t и T – одинаковые идентификаторы). Данные параметр можно изменить в настройках системы.

Программа состоит из деклараций и операторов. Декларации начинаются с ключевых слов и инициализируют соответствующие переменные и константы:

**сonst** – константы;

**var** – алгебраические переменные;

**init** – динамические (дифференциальные) переменные;

**input** – входные переменные блока «Язык программирования»;

**output** – выходные переменные блока «Язык программирования».

После ключевого слова перечисляются через запятую имена переменных либо констант с возможным заданием начальных значений, например:

**const** a=5.3,b=1.2e-4,c=10,y0=a\*b/c;

**var** x1,v=b;

**output** y=y0;

Если начальное значение не задано, то оно принимается равным 0. Разделителем деклараций и операторов является точка с запятой (;).

В декларациях обязательно должны быть описаны входные, выходные и динамические (дифференциальные) переменные. Остальные переменные могут быть заданы автоматически в соответствии с выражением, которое присваивается переменной. Рекомендуется (но не обязательно) помещать декларации в начало программы. Переменную или константу можно использовать только после того, как она описана в декларации или задана в операторе присваивания.

Доступны следующие системные переменные и константы:

|  |  |  |
| --- | --- | --- |
| **Переменная** | **Тип данных** | **Назначение** |
| time | Вещественное (double) | Модельное время |
| stepsize | Вещественное (double) | Шаг интегрирования |
| goodstep | Двоичное (Boolean) | Флаг «хорошего» шага |
| getderi | Двоичное (Boolean) | Флаг вычисления значений производных |
| setstepflag | Двоичное (Boolean) | Флаг принудительного присвоения шага |
| newstepvalue | Вещественное (double) | Принудительное значение шага при setstepflag = 1 |
| pi | Вещественное (double) | Числовое значение числа π |
| clrf | Строковое (string) | Символ разделителя строк |
| quote | Строковое (string) | Символ кавычка |
| tab | Строковое (string) | Символ табуляции |
| expbase | Строковое (string) | Строковое значение числа e, равное «2.71828182845905» |

Операторы языка программирования включают в себя большое количество встроенных функций для осуществления различных стандартных операций. Все встроенные функции языка программирования делятся на две группы по месту, где возможно их использование:

* функции, доступные в любом месте проекта – как в любом графическом контейнере (субмодель, изображение блока, видеокадр), так и в блоке «Язык программирования», а также в скриптах инициализации, исполнения и уничтожения блока;
* функции, доступные только в скрипте графического контейнера (субмодель, изображение блока, видеокадр) - такие функции недоступны внутри блока «Язык программирования» и скриптах инициализации, исполнения и уничтожения объекта;

Перечень всех функций приведен в разделах 6.7 и 6.8.

Программа, написанная в скрипте может содержать комментарии, заключенные в фигурные скобки:

*{текст комментария}*

либо в виде:

*//текст комментария до конца строки.*

Приведем несколько простых примеров программ для блока «Язык программирования».

**Пример 1.**

*Выходная переменная блока – максимальное значение из двух входных переменных.*

|  |  |
| --- | --- |
|  | **input** u1,u2;  **if** u1>u2 **then** y=u1 **else** y=u2;  **output** y; |

Можно упростить эту программу, используя функцию **max** языка, тогда оператор вычисления выходной переменной запишется в виде y=**max**(u1,u2);

**Пример 2.**

*Выходная переменная запоминает максимальное значение входной переменной.*

|  |  |
| --- | --- |
|  | **input** u;  **if** time=0 **then** y=u **els**e  **if** goodstep **then** y=**max**(y,u);  **output** y; |

Значение time=0 соответствует начальному моменту модельного времени. Переменная y практически задает состояние блока. Расчет таких переменных рекомендуется выполнять только на хорошем шаге, когда булевская переменная goodstep равна 1. В противном случае (при goodstep=0) переменная u может быть вычислена, например, на промежуточной итерации решения алгебраических уравнений, в результате чего значение переменной y будет неверным.

**Пример 3.**

*Блок вычисляет время переходного процесса, т.е. время, в течение которого абсолютное значение входной переменной z уменьшается до z0 и в дальнейшем не превышает этой величины.*

|  |  |
| --- | --- |
|  | **const** z0=0.05;  **input** z;  **output** T=0; *//0 – начальное значение*  **if** goodstep **and** (**abs**(z)>z0) **then** T=time; |

**Пример 4.**

*Блок описывает уравнение Ван-дер-Поля.*

|  |  |
| --- | --- |
|  | **const** mu=1e6;  **init** x1=2,x2=0;  x1’=x2; *//дифференциальные уравнения*  x2’=mu\*((1-x1^2)\*x2-x1);  **output** x1,x2; |

Здесь операторы присваивания задают дифференциальные уравнения и определяют новые переменные x1’,x2’ (производные соответствующих переменных), которые, как и другие переменные, могут быть использованы в математических выражениях.

**Пример 5.**

*Счетчик числа поступающих на вход импульсов.*

|  |  |
| --- | --- |
|  | **input** u;  **output** Nimp=0;  **var** u0=1;  **if** goodstep **and** (u0<=0) **and** (u>0) **then**  **begin** Nimp=Nimp+1; u0=u **end**; |

Импульсы считаются по переднему фронту, т.е. моменту модельного времени, когда входная переменная возрастает и становится положительной.

В следующих подразделах справки приведены описания ключевых слов и функций языка программирования. Символами < и > помечены обязательные идентификаторы. Символами { и } помечены дополнительные идентификаторы.