

PRADIS

ОПИСАНИЕ ЯЗЫКА PradiSLang

**ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ АВТОМАТИЗАЦИИ
МОДЕЛИРОВАНИЯ НЕСТАЦИОНАРНЫХ ПРОЦЕССОВ В
МЕХАНИЧЕСКИХ СИСТЕМАХ И СИСТЕМАХ ИНОЙ
ФИЗИЧЕСКОЙ ПРИРОДЫ**

ВЕРСИЯ 4.2

Содержание

1. АЛФАВИТ ЯЗЫКА PradiSLang.....	3
2. ОБЩИЕ ПОЛОЖЕНИЯ.....	4
2.1. СИНТАКСИЧЕСКИЕ ЕДИНИЦЫ.....	4
2.2. СТРУКТУРА ПРОГРАММЫ.....	4
3. ОПИСАНИЕ ДАННЫХ - РАЗДЕЛ \$ DATA.....	7
4. ОПИСАНИЕ ОБЪЕКТА - РАЗДЕЛ \$ FRAGMENT.....	8
4.1. ОПИСАНИЕ БАЗОВЫХ УЗЛОВ ФРАГМЕНТА - ПОДЗАГОЛОВОК # BASE..	9
4.2. ОПИСАНИЕ СТРУКТУРЫ ФРАГМЕНТА – ПОДРАЗДЕЛ # STRUCTURE.....	9
4.3. ОПИСАНИЕ ВНЕШНИХ УЗЛОВ ФРАГМЕНТА - ПОДЗАГОЛОВОК # EXTERNAL.....	11
4.4. ОПИСАНИЕ ВЫВОДА ДЛЯ ФРАГМЕНТА – ПОДРАЗДЕЛ # OUTPUT.....	12
4.5. РАСПЕЧАТКА СТРУКТУРЫ ГЛОБАЛЬНОГО ФРАГМЕНТА - ПОДЗАГОЛОВОК # MAP.....	14
5. ИЗОБРАЖЕНИЕ ОБЪЕКТА В ПРОЦЕССЕ РАСЧЕТА - РАЗДЕЛ \$ SHOW.....	15
6. ОПИСАНИЕ ЗАДАНИЯ.....	17
6.1. ЗАМЕНА ПАРАМЕТРОВ В СФОРМИРОВАННОЙ МОДЕЛИ - РАЗДЕЛ \$ REPLACE.....	17
6.2. ВОССТАНОВЛЕНИЕ СОСТОЯНИЯ РАСЧЕТА - ЗАГОЛОВОК \$ RESTORE	19
6.3. ВЫПОЛНЕНИЕ РАСЧЕТА - РАЗДЕЛ \$ RUN.....	19
6.4. ОТОБРАЖЕНИЕ РЕЗУЛЬТАТОВ - РАЗДЕЛ \$ PRINT.....	21
7. ВОЗМОЖНАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ РАЗДЕЛОВ И ПОДРАЗДЕЛОВ В ПРОГРАММЕ НА ЯЗЫКЕ PradiSLang.....	23
8. ИСПОЛЬЗОВАНИЕ ИНСТРУКЦИИ ПРЕПРОЦЕССОРА \$ INCLUDE.....	25
9. РУССКОЯЗЫЧНЫЕ СИНОНИМЫ ЗАГОЛОВКОВ И ПОДЗАГОЛОВКОВ ЯЗЫКА PradiSLang.....	26
10. ПРИМЕРЫ ПРОГРАММ НА ЯЗЫКЕ PradiSLang.....	27
10.1. ПРОГРАММА, СОДЕРЖАЩАЯ ОПИСАНИЕ ОБЪЕКТА И ОПИСАНИЕ ЗАДАНИЯ НА РАСЧЕТ И ОТОБРАЖЕНИЕ РЕЗУЛЬТАТОВ.....	27
10.2. ПРОГРАММА, РЕАЛИЗУЮЩАЯ ПРОДОЛЖЕНИЕ РАСЧЕТА С ПОСЛЕДНЕГО МЕСТА СОХРАНЕНИЯ РЕЗУЛЬТАТОВ.....	29
10.3. ПРОГРАММА, СОДЕРЖАЩАЯ ЗАДАНИЕ НА ОТОБРАЖЕНИЕ РАНЕЕ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ РАСЧЕТА.....	30

1. АЛФАВИТ ЯЗЫКА PradiSLang

В языке *PradiSLang* используются следующие символы:

прописные и строчные буквы латинского алфавита	A-Z, a-z
прописные и строчные буквы русского алфавита	А-Я, а-я
цифры	0-9
знаки унарных арифметических операций	+ -
знак присваивания	=
круглые скобки)(
десятичная точка	.
разделители и признаки производной	" , ; : ' /
	<CR>
признаки заголовка и подзаголовка	\$ #
символы комментариев	} {
символ подчеркивания	—
пробел	

2. ОБЩИЕ ПОЛОЖЕНИЯ

2.1. СИНТАКСИЧЕСКИЕ ЕДИНИЦЫ

Все **инструкции** языка *PradiSLang* записываются в первых 72 позициях строки. Любая инструкция может быть продолжена на следующей строке, при этом предыдущая строка должна заканчиваться соответствующим разделителем. Количество строк продолжения не ограничено.

Действительное число записывается в обычной форме или в форме с десятичным множителем. Пробелы внутри мантиссы и порядка не допустимы. Примеры правильно записанных действительных чисел:

-3 -3.5 -.5 1.5E-3 .7E-10 15.7 e-3

Целые положительные числа записываются последовательностью цифр, не содержащей пробела. Целые отрицательные числа и целый 0 в *PradiSLang* не используются.

Имена в программе используются для определения ссылок на программы, реализующие модели элементов, программы расчета выходных переменных, фрагменты и в других, оговоренных ниже случаях. Имя - это последовательность не более чем из 8 символов, обязательно начинающаяся с буквы. Все имена в *PradiSLang*, за исключением имен фрагментов, записываются прописными буквами латинского алфавита. Допустимыми символами в имени фрагмента являются прописные и строчные буквы латинского и русского алфавита, цифры и символ подчеркивания.

Идентификаторы в программе используются для обозначения списков параметров, конкретных конструктивных элементов, составляющих описываемый объект, выходных переменных и в других, оговоренных ниже случаях. Идентификатор - это предложение из не более, чем 32 символов (прописных и строчных букв, цифр, пробелов, точек и символов подчеркивания), начинающееся с буквы. Только первый пробел из нескольких, стоящих подряд, является значащим и разделяет входящие в предложение слова.

В текст программы могут быть включены **комментарии**. Комментарием считается любая группа символов, заключенная в скобки { }. Вложенность комментариев не допускается.

Пробелы в программе, кроме специально оговоренных случаев, игнорируются.

2.2. СТРУКТУРА ПРОГРАММЫ

PradiSLang относится к типу непроцедурных (описательных) языков и включает в себя средства, необходимые для описания структуры и параметров исследуемого технического объекта.

В общем случае программа на языке *PradiSLang* состоит из следующих основных частей:

- описание данных;
- описание моделируемого объекта;
- описание изображения объекта;
- описание задания на расчет и отображение результатов.

В соответствии с этим в программе могут присутствовать **разделы** описания данных, описания объекта, описания изображения объекта, описания задания на расчет, описания задания на отображение. Каждый раздел программы начинается **заголовком**. В

разделе описания объекта могут присутствовать **подразделы**, каждый из которых начинается **подзаголовком**.

Заголовки и подзаголовки записываются после символов \$ и # прописными буквами латинского или русского алфавита. Примеры заголовков и подзаголовков:

```
$FRAGMENT
#OUTPUT
$ЗАМЕНИТЬ
$КОНЕЦ
```

Программа на *PradiSLang* должна в обязательном порядке содержать заголовок \$ PRINT и заканчиваться заголовком \$ END. Информация, находящаяся в программе до первого признака заголовка (\$) и после заголовка \$ END, не анализируется, однако она не должна содержать символов, не входящих в алфавит языка.

Описание объекта состоит из одного или нескольких разделов описания объекта. В дальнейшем в этом документе термину **раздел описания объекта** соответствует синоним **фрагмент объекта** (или просто **фрагмент**).

Для описания структуры объекта используются подразделы **описания структуры объекта**. В ходе описания структуры модель объекта представляется в виде совокупности моделей элементов, включенных в библиотеки комплекса. В качестве элементов при описании структуры могут использоваться также фрагменты, описание которых в тексте программы предшествует описанию текущего фрагмента. Элементы соединяются между собой в определенных точках физического пространства. Состояние этих точек характеризуется некоторым количеством **обобщенных координат** (или **степеней свободы**) и их производных. Количество степеней свободы в каждой точке физического пространства зависит в общем случае от размерности пространства, используемых моделей элементов и способа соединения этих элементов. В дальнейшем в этом документе понятие **узел** используется как синоним понятия **степень свободы объекта**. Узлы объекта нумеруются положительными целыми числами.

При описании каждого из элементов задается определенное количество узлов, с которыми этот элемент соединяется. Когда речь идет о степенях свободы отдельного элемента, удобно оперировать понятием **ветвь элемента**. Итак, в дальнейшем в этом документе понятие ветвь элемента используется как синоним понятия **степень свободы элемента**. Номера ветвей элемента определяются порядком их задания при описании элемента. Например, если элемент соединен с объектом в узлах с номерами 30, 1 и 27, то его первой ветвью считается ветвь, соединенная с узлом 30, второй - соединенная с узлом 1, третьей - соединенная с узлом 27. Таким образом, номер ветви элемента не может превышать количество степеней свободы этого элемента.

В ходе расчета *PradiSLang* оперирует переменными, называемыми далее внутренними. К **внутренним переменным** относятся, например в механике, силы, перемещения, скорости, ускорения и т.д., т.е. величины, получаемые непосредственно в ходе решения. Количество таких переменных даже при незначительных размерах модели велико, а во многих случаях может достигать сотен и тысяч. Поэтому нецелесообразно сохранять информацию о каждой внутренней переменной. Непосредственно доступными для отображения являются величины, вычисляемые по различным алгоритмам с использованием внутренних переменных и описанные как **выходные переменные**. Информация о выходных переменных сохраняется после проведения расчетов. Выходные переменные описываются в соответствующих подразделах **описания вывода для фрагмента**.

В тексте программы на языке *PradiSLang* есть возможность определить изображение объекта в ходе расчета. В разделе **описания изображения** объекта определяется соответствие между моделями элементов, входящими в описание структуры объекта, и их графическими образами. Группы графических образов объединяются в **слои**

изображения. Имеется возможность отдельного управления изображением в каждом из слоев. При желании, изображение объекта может быть сгенерировано автоматически из стандартных графических образов по описанию структуры объекта.

Модели элементов, программы расчета выходных переменных, программы реализации графических образов и программы реализации изображения объекта требуют задания определенного количества параметров, которые описываются в виде **списков параметров**. Списки параметров, используемые при описании объекта и задания, как правило, определяются в отдельных разделах **описания данных**. Разделы описания данных не являются обязательными в программе на языке *PradiSLang*. Данные, относящиеся к описанию объекта, его изображения или описанию задания, можно задавать непосредственно в тексте соответствующих разделов. Однако, использование разделов описания данных для создания поименованных списков параметров в значительной степени облегчает работу по подготовке данных, улучшает читаемость программы и облегчает ее дальнейшую эксплуатацию (например, становится доступной такая возможность, как замена параметров в сформированной модели).

В разделах **описания задания** помимо расчета переходного процесса реализованы такие возможности, как замена параметров в уже сформированной модели без осуществления повторной генерации, восстановление состояния расчета с прерванного места и сохранение состояния расчета по желанию пользователя. В каждом задании на языке *PradiSLang* предполагается наличие описания отображения результатов. При этом **отображаемой переменной** может являться какая-либо из выходных переменных. Другими словами, множество отображаемых переменных в каждом конкретном случае является непустым подмножеством множества выходных переменных. Имеется возможность формировать задания только для отображения результатов проведенных ранее расчетов, в которых можно изменять состав отображаемых переменных, масштабы для графиков отображаемых переменных, изменять границы временного интервала, для которого происходит отображение результатов, использовать иные, чем было предусмотрено ранее, программы отображения, произвольным образом комбинировать отображаемые переменные и т.д.

Пример простейшей программы на языке *PradiSLang*:

```
{ описание данных }
$ DATA :
    Масса = 7.5;
    Коэффициент жесткости = 2;  Масштаб = 1
{ описание объекта }
$ FRAGMENT :
    # BASE : 1
    # STRUCTURE :
        Пружина 'K (1 2; Коэффициент жесткости);
        Масса   'M (2; Масса);
        Сила     'F (2; 1.)
    # OUTPUT :
        Перемещение груза ' S (2; Масштаб)
        Скорость груза   ' V (2; Масштаб)
        Усилие на пружине ' X (1:Пружина; 1.)
{ описание задания }
{ описание задания на расчет }
$ RUN :
    Расчет колебаний груза ' SHTERM (END=1.0;
    Скорость груза = (,0.5),
    Перемещение груза = (-0.1,0.1),
    Усилие на пружине )
{ описание задания на отображение результатов }
$ PRINT :
    Колебания груза ' DISP (;
    Скорость груза = (,0.5),
```

```

Перемещение груза ,
Перемещение груза = (-0.005,0.) ,
Усилие на пружине )
$END

```

3. ОПИСАНИЕ ДАННЫХ - РАЗДЕЛ \$ DATA

Использование **поименованных списков параметров** позволяет значительно улучшить читаемость программы на языке *PradiSLang* и обеспечивает возможность изменения значений параметров при работе со сформированной моделью объекта. Для описания поименованных списков параметров служит специальный раздел описания данных. Раздел начинается с заголовка:

\$ DATA: [*имя*]

где *имя* - имя раздела описания данных.

Это *имя* определяет, что раздел данных, описанный под этим заголовком, относится к одноименному разделу описания объекта (одноименному фрагменту). При этом описание блока данных должно предшествовать описанию фрагмента, которому соответствуют эти данные. Каждому фрагменту соответствует один и только один поименованный блок данных.

В начале каждой программы на языке *PradiSLang* может присутствовать один непоименованный заголовок \$ DATA. Данные, описанные в непоименованном заголовке \$ DATA, являются глобальными и доступными для всей программы.

За заголовком \$ DATA на последующих строках дается описание списков параметров в форме:

```

идентификатор = список
список = число
или список = список, число
или список = ~идентификатор [( номер = число, ... , номер = число)]
[ , список ]

```

<i>идентификатор</i>	Идентификатор списка параметров.
<i>список</i>	Формируемый для использования в описании объекта и его изображения список параметров.
<i>число</i>	Действительное число, задающее соответствующее значение параметра.
<i>номер</i>	Номер параметра, замещаемый в списке параметров с идентификатором <i>~идентификатор</i> на параметр, заданный действительным числом, стоящим после знака равенства.
<i>~идентификатор</i>	Идентификатор списка параметров, определенного ранее.

Порядок следования действительных чисел и их количество должны строго соответствовать порядку следования и количеству параметров того элемента, при описании которого предполагается использовать данный список параметров.

Описание каждого списка параметров должно отделяться от других описаний списков параметров точкой с запятой или концом строки. Каждый список параметров может располагаться в одной или нескольких строках текста программы. В последнем случае каждая переносимая строка должна завершаться соответствующим разделителем (";" или "=").

Пример. Требуется создать глобальный блок данных для всей программы и локальный блок данных для фрагмента "Маятник".

```
$ DATA :
    Масштаб = 1.;
    Модуль упругости = 2.E11
$ DATA : Маятник
    Точка А = 0., 0.;
    Точка В = 1., 0.;
    Материал = 0.001, Модуль упругости;
    Масса = 1.;
    Сила тяжести = 9.81;
    Параметры маятника 1 = Точка А,
                                Точка В,
                                Материал;
    Параметры маятника 2 = Точка А,
                                Точка В (2=1.5),
                                Материал;
```

В результате будут сформированы глобальный блок данных, списки параметров которого будут доступны во всей программе, и локальный блок данных, доступный из фрагмента "Маятник". В этом локальном блоке данных будут определены следующие списки параметров:

```
Точка А                = 0., 0.;
Точка В                = 1., 0.;
Материал               = 0.001, 2.E11;
Масса                 = 1.;
Сила тяжести           = 9.81;
Параметры маятника 1 = 0., 0., 1., 0., 0.001, 2.E11;
Параметры маятника 2 = 0., 0., 1., 1.5, 0.001, 2.E11;
```

4. ОПИСАНИЕ ОБЪЕКТА - РАЗДЕЛ \$ FRAGMENT

Раздел описания объекта (фрагмент) начинается с заголовка:

\$ FR[AGMENT]: [*имя*]

имя - имя фрагмента.

В общем случае структура объекта может быть представлена одним или несколькими включенными друг в друга фрагментами. При этом всегда имеется только один фрагмент самого верхнего уровня, включающий в себя поименованные (обязательно) фрагменты нижнего уровня. Фрагмент верхнего уровня может иметь имя, но может быть и непоименованным.

Степень вложенности фрагментов друг в друга не ограничена. Каждый фрагмент состоит из подразделов, имеющих следующие подзаголовки:

Задание базовых узлов - подзаголовков # BASE (не обязателен; если есть, то следует сразу после заголовка \$ FRAGMENT).

Описание структуры фрагмента - подзаголовок # STRUCTURE (обязателен и следует сразу за подзаголовком # BASE или после заголовка \$ FRAGMENT).

Описание внешних узлов фрагмента (узлов, служащих для соединения с узлами фрагментов, в структуру которых будет включаться описываемый фрагмент) - подзаголовок # EXTERNAL (по необходимости).

Описание выходных переменных - подзаголовок # OUTPUT (по необходимости; однако хотя бы в одном фрагменте задания должен быть определен подзаголовок # OUTPUT).

Требование вывести на печать отладочную информацию о структуре объекта с указанием идентификаторов и имен, включенных во фрагмент моделей элементов (в том числе и входящих во включаемые фрагменты), а также глобальных номеров узлов, с которыми эти элементы соединяются - подзаголовок # MAP (по необходимости).

Подразделы #EXTERNAL, #MAP и #OUTPUT должны следовать за подразделом #STRUCTURE. Порядок следования этих подразделов не существует.

4.1. ОПИСАНИЕ БАЗОВЫХ УЗЛОВ ФРАГМЕНТА - ПОДЗАГОЛОВОК # BASE

Этот подзаголовок служит для описания узлов, кинематические либо другие потенциальные характеристики которых принимаются за 0 (потенциал, температура, давление и т.д.). Например, в механике - это закрепленные узлы, в пневматике - узлы, соединенные с атмосферой и т.д. В дальнейшем такие узлы будем называть **базовыми**.

Структура описания базовых узлов фрагмента:

```
# BASE : узел1 [ , узел2 [ , ... узелn ] ] [ ; ]
```

узел_j – номер узла фрагмента, описываемого как базовый.

Необходимо иметь в виду, что для любого фрагмента в обязательном порядке должен быть определен базовый узел. Если фрагмент нижнего уровня в дальнейшем войдет в описание структуры фрагмента верхнего уровня и необходимо сохранить закрепление, то один из закрепленных узлов фрагмента нижнего уровня (обязательно присутствующий в описании структуры) должен быть описан как внешний и в описании фрагмента верхнего уровня также описан как базовый.

Если подраздел отсутствует, в качестве базового принимается первый узел, встретившийся в описании структуры.

Пример.

Если в системе в качестве базовых принимаются узлы с номерами 5 и 6, то после заголовка \$ FRAGMENT в программе должен присутствовать подзаголовок # BASE :

```
# BASE : 5, 6
```

4.2. ОПИСАНИЕ СТРУКТУРЫ ФРАГМЕНТА – ПОДРАЗДЕЛ # STRUCTURE

Подраздел начинается одноименным подзаголовком:

```
# STRUCT[URE] :
```

После подзаголовка на последующих строках дается описание структуры объекта - список описаний элементов или фрагментов, включенных в данный фрагмент. Элементы этого списка - описания включаемых во фрагмент элементов или фрагментов - отделяются друг от друга разделителем ; или концом строки. Каждое описание включаемых во фрагмент моделей элементов или фрагментов может располагаться на нескольких строках текста программы. При этом переносимая строка должна завершаться соответствующим разделителем ("," - запятая, "'" - апостроф, "." - точка с запятой).

Описание моделей элементов, включаемых во фрагмент, имеет следующий вид:

идентификатор ' *имя* (*узел₁* , *узел₂* ... , *узел_n* ; *список*)

<i>идентификатор</i>	Идентификатор модели элемента.
<i>имя</i>	Имя модели элемента из числа моделей, включенных в библиотеки комплекса.
<i>узел_j</i>	Номер узла фрагмента, к которому подключена j-я ветвь элемента. Номер узла задается как произвольное положительное целое число без знака. Для эффективной работы принятых в комплексе алгоритмов лучше, если номера узлов задаются без существенных пропусков. Количество узлов подключения модели элемента должно строго соответствовать ее описанию.
<i>список</i>	Список параметров модели элемента. Задается по правилам, определенным ранее для заголовка \$ DATA. Количество и последовательность параметров должны полностью соответствовать количеству и порядку следования параметров, определенных в документации по модели элемента.

Описание включения описанных ранее фрагментов в текущий фрагмент производится в соответствии со следующим синтаксисом:

идентификатор ' *имя* (*узел₁* , *узел₂* ... , *узел_n* [; ~*идентификатор1*
= ~*идентификатор2*, ...])

<i>идентификатор</i>	Идентификатор фрагмента, сформированный по общим правилам.
<i>имя</i>	Имя фрагмента из числа определенных ранее.
<i>узел_j</i>	Номер узла описываемого фрагмента, к которому подключен j-й внешний узел включаемого фрагмента. Количество и порядок следования узлов подключения включаемого фрагмента должны строго соответствовать количеству и порядку следования узлов, определенных в подразделе #EXTERNAL включаемого фрагмента.
~ <i>идентификатор1</i>	Идентификатор списка параметров, определенного для включаемого в описание фрагмента. Этот список параметров замещается на список параметров, идентификатор которого указан справа от знака равенства.

~идентификатор2

Идентификатор списка параметров, определенного для текущего фрагмента, который замещает все вхождения ~идентификатор1, встречающиеся в описании включаемого фрагмента.

Внимание! Замещаются те и только те списки параметров, которые явно определены в описании структуры или в описании выходных переменных включаемого фрагмента.

Разделителями для номеров узлов, кроме запятых, могут служить пробелы. Транслятор **различает** модель элемента и фрагмент, имеющие одинаковое имя.

Пример.

Описание моделей элементов, включаемых во фрагмент, может выглядеть следующим образом:

```
Пружина ' К      (1 2; 10);  
          ' М      (2; Масса)  
Маятник ' STRGNO ( 1,2,3,10;  
                  Точка А, 2., 2.5,  
                  Материал (2=0.7e11) )
```

Описание фрагментов, включаемых в текущий фрагмент:

```
Пружина      ' Упор (1 2);  
Столб        ' S    ( 1,2,3,10; Точка А = Точка С )
```

В приведенных примерах предполагается, что ранее в разделах описания данных определены списки параметров:

Точка А, Материал, Масса, Точка С ,

в состав комплекса включены модели элементов:

К, STRGNO, М,

а в соответствующих разделах описания объекта определены фрагменты:

С, Упор

фрагмент S имеет 4 узла, описанных в подразделе #EXTERNAL, а фрагмент Упор - 2 узла.

4.3. ОПИСАНИЕ ВНЕШНИХ УЗЛОВ ФРАГМЕНТА - ПОДЗАГОЛОВКИ # EXTERNAL

Синтаксис описания **внешних узлов** фрагмента (т.е. узлов фрагмента, которые служат для включения описываемого фрагмента во фрагменты более высокого уровня) определяется следующим образом:

EXT[ERNAL]: узел₁ [, узел₂ [, узел_n]] [;]

узел_j - номер узла фрагмента, описываемого как j-й внешний узел.

В строке может быть записано любое целое количество номеров узлов, причем за последним узлом в строке (если он не является последним в списке) обязательно должен следовать разделитель ";".

Пример описания внешних узлов:

```
#EXT: 3, { базовый узел }  
      6,4 { опора эксцентрика }
```

4.4. ОПИСАНИЕ ВЫВОДА ДЛЯ ФРАГМЕНТА – ПОДРАЗДЕЛ # OUTPUT

Подраздел начинается подзаголовком:

#OUTPUT:

После подзаголовка на последующих строках дается описание выходных переменных - список описаний вызовов соответствующих программ расчета выходных переменных. Элементы этого списка - описания вызовов отдельных программ расчета выходных переменных - отделяются друг от друга разделителем “;” (точка с запятой) или концом строки. Каждое описание выходной переменной может располагаться на нескольких строках. В этом случае переносимая строка должна заканчиваться соответствующим разделителем (“,- запятая, ’ ’ - апостроф, “.- точка с запятой).

Описание выходной переменной (= описание вызова соответствующей программы расчета выходной переменной) имеет следующий вид:

идентификатор ' имя (переменная₁ [, переменная₂ [... , переменная_n]]; список)

идентификатор Идентификатор выходной переменной.

имя Имя программы расчета выходных переменных из числа программ, включенных в библиотеки комплекса.

переменная_j Указатель на j-ю внутреннюю переменную, передаваемую в программу расчета выходных переменных. Количество и порядок указателей в описании вызова должны соответствовать их количеству и порядку следования в документации для программы расчета выходных переменных. Синтаксис описания указателя на внутреннюю переменную определен ниже.

список Список параметров программы расчета выходных переменных. Задается по правилам, определенным ранее для заголовка \$ DATA. Количество и последовательность параметров должны полностью соответствовать количеству и порядку следования параметров, определенных в документации для программы расчета выходных переменных.

Допустимыми указателями на внутренние переменные являются:

узел_j Номер узла описываемого фрагмента, потенциальные характеристики которого передаются в программу расчета выходных переменных (основная потенциальная переменная и две ее производных по времени). Под потенциальными характеристиками узла подразумевается:

в механике - перемещение узла $узел_j$ и его производные по времени;

в гидравлике, пневматике - интеграл по времени от давления, давление и его производная по времени для $узла_j$;

в термодинамике - интеграл от температуры, температура и ее производная по времени для $узла_j$;

в общем случае - значение переменной (в терминах которой ищется решение системы дифференциальных уравнений), характеризующей состояние узла, и значения ее производных по времени.

$узел_j'$

Первая производная по времени основной потенциальной переменной $узла_j$ (соответственно скорость, давление, температура, потенциал).

$узел_j''$

Вторая производная по времени основной потенциальной переменной $узла_j$ (ускорение, производная от давления, производная от температуры, производная от потенциала).

I:идентификатор [(ветвь_j)]
F:идентификатор [(ветвь_j)]
Q:идентификатор [(ветвь_j)]

Указатель на **потокową переменную**, передаваемую в программу расчета выходных переменных. Все три формы записи указателя на потокową переменную эквивалентны. Для элемента с идентификатором *идентификатор* характеризует состояние *ветви_j* этого элемента. Необходимо заметить, что нумерация ветвей является локальной для элемента, поэтому значение *ветвь_j* не должно превышать количества узлов этого элемента. Так, для двухузлового элемента ветвь_j может принимать значения 1 или 2. По умолчанию предполагается *ветвь_j* = 1. Под потоковой переменной подразумевается:

в механике - сила (момент), с которыми система действует на элемент. Для определения признака указателя на механическую потокową переменную предпочтительно использовать первую или вторую форму записи указателя (с признаками I или F);

в гидравлике, в пневматике - расход или эквивалентная ему величина, направленная от системы к элементу. Для определения признака указателя на такую потокową переменную предпочтительно использовать первую или третью форму записи указателя (с признаком I или Q);

в термодинамике - тепловой поток, направленный от системы к элементу. Как и в предыдущем случае, предпочтительна первая или третья форма записи (с признаком I или Q);

в общем случае - значение переменной (относительно которой сформулирован закон сохранения - 3-й закон Ньютона, закон Кирхгофа для токов и т.д.), характеризующее поток этой величины от системы к

элементу. Предпочтительна первая форма записи потоковой переменной (с признаком I).

W: идентификатор[(N)]

Указатель на N-ю компоненту **рабочего вектора** (по умолчанию - на 1-ю компоненту) для элемента с идентификатором *идентификатор*, значение которой требуется передать в программу расчета выходных переменных.

Доступные компоненты рабочего вектора каждой модели элемента приводятся в ее описании.

Пример. Описание вызовов программ расчета выходных переменных может выглядеть следующим образом:

```
Сила на пружине      'X      (I:Пружина; 1.);
Перемещение по оси X 'S      (3 ; 1);
Скорость по оси X    'V      (3 ; Масштаб);
Скорость по оси X    'X      (3'; Масштаб);
Ускорение опоры      'ROUT (1", 2";1);
Реакция в опоре маятника 'ROUT ( I:Маятник(1),
                                I:Маятник(2); 1)
Энергия пружины      'X      (W:Пружина; 1.);
```

В приведенных примерах предполагается, что ранее в разделах описания данных определен список параметров:

Масштаб,

в состав комплекса включены программы расчета выходных переменных:

S, X, V, ROUT,

а в соответствующих разделах описания объекта определены элементы с идентификаторами:

Пружина, Маятник

4.5. РАСПЕЧАТКА СТРУКТУРЫ ГЛОБАЛЬНОГО ФРАГМЕНТА - ПОДЗАГЛОВОК # MAP

В результате анализа описания объекта *PradiSLang* получает информацию о структуре якобиана системы дифференциальных уравнений. При этом он всегда использует внутреннюю нумерацию узлов, отличающуюся от нумерации узлов, заданной пользователем. Кроме того, для уменьшения вычислительных затрат после выполнения синтаксического анализа как правило осуществляется оптимальная перенумерация узлов.

Для получения информации об окончательной нумерации узлов в текст глобального фрагмента включается подзаголовок

MAP :

В этом случае после выполнения синтаксического анализа на печать выдается список моделей элементов, их идентификаторов и номеров узлов, полученных после выполнения перенумерации.

Все сообщения программ вычислительного ядра ссылаются на эти номера узлов и номера моделей элементов. Кроме того, эта информация необходима в случае использования средства DEBUG программы интегрирования.

5. ИЗОБРАЖЕНИЕ ОБЪЕКТА В ПРОЦЕССЕ РАСЧЕТА - РАЗДЕЛ \$ SHOW

Раздел описания изображения объекта в процессе расчета следует после глобального раздела описания структуры непосредственно перед разделами описания задания. Раздел описания изображения объекта начинается с заголовка:

\$ SHOW:

После заголовка на последующих строках дается описание изображения объекта - список описаний изображений отдельных слоев, включенных в изображение данного фрагмента. Элементы этого списка отделяются друг от друга разделителем (“,”- точка с запятой) или концом строки. Описание каждого слоя изображения может располагаться в нескольких строках текста программы. В этом случае каждая переносимая строка должна заканчиваться соответствующим разделителем (“,”- запятая, “,”- точка с запятой, “”- апостроф).

В параметрах слоя изображения задаются определенный масштаб изображения и его цвет. Он характеризуется определенной точкой зрения наблюдателя и, возможно, описанием движения этой точки. Количество и последовательность параметров должны полностью соответствовать количеству и порядку следования параметров, определенных в описании программы реализации изображения слоя (LAYER).

Описание отдельного слоя выглядит следующим образом:

```
~идентификатор ' LAYER ([[идентификатор1]][(имя1 ; список1)]  
[,[идентификатор2]][(имя2 ; список2)] ] ...];  
~список [;узел1 узел2 ... узел9])
```

~идентификатор

Идентификатор слоя.

идентификатор_j

Идентификатор модели элемента, графический образ которой включается, в описание слоя. Если при описании какого-либо графического образа идентификатор модели не указывается, считается, что графический образ неподвижен (связан с неподвижной системой координат). Если в описании слоя отсутствует описание каких-либо графических образов, в этот слой включаются образы всех элементов объекта, имеющих идентификаторы. В этом случае изображение каждого элемента строится с использованием графического образа, принятого для этого элемента по умолчанию (“стандартный” графический образ).

имя_j

Имя графического образа, если для изображения объекта используется не стандартный графический образ или графический образ пользователя. Если имя графического образа не задано, то для изображения элемента используется графический образ этого элемента, принятый по умолчанию.

список_j

Список параметров нестандартного графического образа. Задается в соответствии с правилами, определенными ранее для заголовка \$ DATA. Количество и последовательность параметров должны полностью соответствовать количеству и порядку следования параметров, определенных в документации по программе реализации нестандартного графического образа.

~список

Список параметров программы реализации изображения слоя. Задается по правилам, определенным ранее для заголовка \$ DATA. Количество и последовательность параметров должны полностью соответствовать количеству и порядку следования параметров, определенных в документации по программе реализации изображения слоя.

узел_j

Список степеней свободы, с которым связана программа реализации изображения слоя. Если список степеней свободы не задан, то считается, что программа реализации изображения слоя связана с базовым узлом (= с неподвижной системой координат).

Описание изображения объекта может выглядеть как в приведенных ниже примерах.

Пример 1.

\$ SHOW:

```
Изображение объекта 'LAYER( ;Параметры слоя ;  
1 2 3 4 5 6 7 8 9)
```

В приведенном описании в изображение объекта включаются все элементы, имеющие идентификаторы, для которых определены графические образы. Список параметров "Параметры слоя" определен ранее в разделе \$ DATA. Программа реализации изображения слоя связана с узлами 1-9.

Пример 2.

```
Изображение части' LAYER ( Пружина ,  
Тело (SILUET;Контур тела) ;  
Параметры слоя) ;  
Изображение болта' LAYER ( Болт (BOLT;1.1,1,0) ,  
(KONTUR; Крепление болта) ;  
Параметры слоя 2) ;
```

В приведенном описании изображение объекта состоит из двух слоев. Идентификаторы слоев - "Изображение части" и "Изображение болта". Оба описанных слоя связаны с неподвижной системой координат.

В состав слоя "Изображение части" включены элементы с идентификаторами "Пружина" и "Тело". Для изображения элемента "Пружина" используется графический образ по умолчанию, для изображения элемента "Тело" используется нестандартный графический образ SILUET. Параметры этого графического образа задаются списком параметров "Контур тела", а параметры слоя - списком параметров "Параметры слоя". Указанные списки параметров определены в разделе \$ DATA.

В состав слоя "Изображение болта" включен элемент с идентификатором "Болт". Для изображения элемента используется нестандартный графический образ BOLT.

Параметры этого графического образа задаются списком из трех параметров. В этот слой изображения включен также элемент, связанный с неподвижной системой координат. Имя этого графического образа KONTUR, список параметров "Крепление болта" описан в разделе \$ DATA. Параметры программы реализации изображения "Параметры слоя 2" также описаны в разделе \$ DATA.

6. ОПИСАНИЕ ЗАДАНИЯ

Информация о разделах описания задания приводится ниже в том порядке, в котором они могут следовать в тексте программы на языке *PradiSLang*. При этом следует иметь в виду, что в любом случае программа должна заканчиваться заголовком \$ END.

6.1. ЗАМЕНА ПАРАМЕТРОВ В СФОРМИРОВАННОЙ МОДЕЛИ - РАЗДЕЛ \$ REPLACE

Раздел \$ REPLACE применяется для замены списков параметров в уже сформированной рабочей программе. Он используется в том случае, если не нужно менять структуру модели, список выходных переменных и описание изображения модели в ходе расчета. Раздел \$ REPLACE позволяет осуществлять новый расчет модели с заданием каких-либо новых параметров моделей элементов, программ расчета выходных переменных либо программ реализации изображения без выполнения новой сборки рабочей программы. Действие инструкций по замене параметров, приведенных в разделе \$ REPLACE, сохраняется только на протяжении текущего расчета (т.е., списки параметров в базе данных уже сформированной модели не меняются). Если следующий расчет для этой модели будет происходить без REPLACE, то в нем будут использованы параметры из первоначального текста задания. Раздел начинается с заголовка:

\$ REPLACE :

После заголовка на последующих строках дается описание замещаемых списков параметров. Один замещаемый список параметров отделяется от другого разделителем “;” (точка с запятой) или концом строки. Синтаксис описания замещаемых списков параметров:

имя = список

имя

Имя замещаемого списка параметров.
Внимание! Замещаются те и только те списки параметров, которые явно содержатся в описании структуры глобального фрагмента, описании его выходных переменных или в разделе описания изображения объекта в ходе расчета.

список

Список параметров, задаваемый по правилам, определенным для раздела \$ DATA. Количество и порядок следования параметров в этом списке должны соответствовать количеству и порядку следования параметров в замещаемом списке.

Раздел \$ REPLACE может следовать после раздела \$ DATA либо быть первым в тексте программы на языке *PradiSLang*. В случае использования возможности \$ REPLACE описание объекта и описание изображения объекта в ходе расчета в программе должно отсутствовать.

Пример замены списков параметров, явно присутствующих в описании структуры. Текст задания для рабочей программы выглядит следующим образом:

```
$ DATA:
Коэффициент жесткости = 1.0E6
Значение зазора        = 1.5E-2
Параметры зазора       = Значение зазора,
Коэффициент жесткости
$ FRAGMENT:
# BASE: 1
# STRUCT:
Масса 1  'M(2; 5)
Масса 2  'M(3; 5)
Упор     'UPRL(2,3; Параметры зазора)
Пружина  'K(1,2; Коэффициент жесткости)
Сила     'F(3; -1E3)
# OUTPUT:
Скорость 'V(3; 1)
$ RUN:
Расчет   'SHTERM (END=1)
$ PRINT:
Результат 'DISP ()
$ END
```

Процедура замены параметра “Коэффициент жесткости” в блоке данных для описанного выше фрагмента:

```
$ REPLACE:
Коэффициент жесткости = 90000
$ RUN:
Расчет   'SHTERM (END=1)
$ PRINT:
Результат 'DISP ()
$ END
```

В результате для элемента “Пружина” будет переопределен список параметров:

```
Коэффициент жесткости = 90000
```

потому что заменяемый параметр присутствует явно в его списке параметров. Тогда как для элемента “Упор”, в списке параметров которого заменяемый параметр содержится не явно, будет задан список параметров:

```
Параметры зазора = 1.5E-2, 1.0E6
```

6.2. ВОССТАНОВЛЕНИЕ СОСТОЯНИЯ РАСЧЕТА - ЗАГОЛОВОК \$ RESTORE

Заголовок \$ RESTORE используется в том случае, если для данной модели технической системы уже проводилось интегрирование с сохранением состояния расчета и требуется продолжить расчет с момента последнего сохранения. Заголовок может быть только первым в тексте задания на языке *PradiSLang* или следовать сразу после раздела \$ REPLACE. В случае использования возможности \$ RESTORE описание объекта и описание изображения объекта в ходе расчета в программе должно отсутствовать.

6.3. ВЫПОЛНЕНИЕ РАСЧЕТА - РАЗДЕЛ \$ RUN

Раздел начинается заголовком:

\$ RUN :

Этот раздел в программе может быть только один. Если заголовок \$ RUN следует после описания объекта или является первым в тексте программы на языке *PradiSLang*, расчет для заданной модели выполняется с нулевого момента времени. В этом случае результаты всех предыдущих расчетов для данной модели утрачиваются. Если заголовок \$ RUN следует после заголовка \$ RESTORE, делается попытка восстановить состояние расчета с последнего места сохранения.

После заголовка \$ RUN на последующих строках идет описание вызовов программ интегрирования. Описание отдельных вызовов разделяется символом “;” (точка с запятой) или концом строки.

Описание каждого из вызовов программы интегрирования характеризуется соответствующим рассчитываемым интервалом времени, параметрами точности и режимами отображения информации по ходу расчета. Для каждой программы интегрирования может быть задан список оперативно отображаемых в ходе расчета выходных переменных из списка, определенного ранее в подразделах # OUTPUT. Считается, что каждая последующая программа интегрирования продолжает расчет с того момента времени, на котором он был прерван предыдущей программой интегрирования. В комплексе *PradiSLang* имеется возможность интерактивно управлять завершением программы интегрирования. В случае интерактивного прерывания соглашение о продолжении расчета последующими программами интегрирования остается в силе.

Сохранение текущего состояния расчета происходит с временным шагом, задаваемым пользователем и, автоматически, по окончании интегрирования.

В настоящее время в состав комплекса входят программы интегрирования системы дифференциальных уравнений II-го порядка неявным методом Штермера и методом Ньюмарка. Описание вызова программы интегрирования выглядит следующим образом:

```
[ ~идентификатор ]' ~имя (имя1 = число [ , имя2 = число [ , ...имяn = число ] );  
идентификатор1 [(номер)] [ = ([ параметр11 ], [ параметр12 ], ... ) ]  
[ идентификатор2 [(номер)] [ = ([ параметр21 ], [ параметр22 ], ... ) ]  
[ идентификаторn [(номер)] [ = ([ параметрn1 ], [ параметрn2 ], ... ) ] ] ]
```

~идентификатор

Идентификатор программы интегрирования.

~имя

Имя программы интегрирования (“SHTERM”, “NEWMARK”).

<i>имя_j</i>	Имя j-го ключевого параметра из списка ключевых параметров, определенных в паспорте программы интегрирования.
<i>число</i>	Действительное число, задающее значение ключевого параметра.
<i>идентификатор_i</i>	Полный идентификатор i-й оперативно отображаемой переменной из списка переменных, описанных в подзаголовке #OUTPUT. Полный идентификатор оперативно отображаемой переменной должен содержать полный путь от фрагмента верхнего уровня к собственно идентификатору переменной в подразделе #OUTPUT включенного фрагмента. Этот путь состоит из последовательности идентификаторов фрагментов, разделенных символом “/”. Последовательность перечисления фрагментов соответствует порядку включения этих фрагментов друг в друга, при этом в начале последовательности находятся фрагменты более высокого уровня.
<i>номер</i>	Номер компоненты (для многокомпонентных отображаемых переменных). Если номер компоненты отсутствует, предполагается, что задано отображение первой компоненты.
<i>параметр_{ij}</i>	j-й позиционный параметр, характеризующий i-ю оперативно отображаемую переменную. Назначение и количество этих параметров определяется в описании программы интегрирования. В случае, если в списке параметров j-й параметр пропущен, его отсутствие отмечается запятой. Отсутствие позиционного параметра означает, что программа интегрирования должна воспользоваться значением этого параметра по умолчанию.

Если необходимо в состав оперативно отображаемых переменных включить весь список переменных, определяемый совокупностью всех подразделов OUTPUT программы, то вызов программы интегрирования не содержит списка отдельных отображаемых переменных и имеет вид:

[~идентификатор] ' **SHTERM** (имя1 = число [, имя2 = число [, ...имяn =
число]])

Примечание. Значения ключевых параметров, не встречающихся в описании вызова программы интегрирования, принимаются по умолчанию равными значениям этих параметров, указанным в паспорте программы интегрирования.

Пример. Описания вызовов программы интегрирования могут выглядеть следующим образом:

```
Расчет колебаний маятника  'SHTERM (END=1,OUT=1.e-3,
                               SAVE=0.1 )
{ продолжение расчета }
                               'SHTERM ( END=1.5 )
```

6.4. ОТОБРАЖЕНИЕ РЕЗУЛЬТАТОВ - РАЗДЕЛ \$ PRINT

Раздел начинается заголовком:

\$ PRINT :

Этот раздел в программе может быть только один. Если заголовок \$ PRINT является первым в тексте программы, предполагается, что требуется осуществить отображение результатов для проведенного ранее расчета. Если заголовок \$ PRINT следует после заголовка \$ RUN, то проводится отображение результатов для текущего расчета.

Наличие этого раздела в программе обязательно.

После заголовка на последующих строках идет описание вызовов программ отображения. Описание отдельных вызовов разделяется символом “;” (точка с запятой) или концом строки.

Описание вызова программы отображения выглядит следующим образом:

```
[~идентификатор]' ~имя (имя1 = число [, имя2 = число [, ...имяn = число ] );
идентификатор1 [(номер)] [ = ([ параметр11 ], [ параметр12 ], ... ) ]
[ идентификатор2 [(номер)] [ = ([ параметр21 ], [ параметр22 ], ... ) ]
[ идентификаторn [(номер)] [ = ([ параметрn1 ], [ параметрn2 ], ... ) ] ] ]
```

<i>~идентификатор</i>	Идентификатор программы отображения.
<i>~имя</i>	Имя программы отображения из списка программ отображения, включенных в состав комплекса.
<i>имя_j</i>	Имя j-го ключевого параметра из списка ключевых параметров, определенных в документации по программе отображения.
<i>число</i>	Действительное число, задающее новое значение ключевого параметра.
<i>идентификатор_i</i>	Полный идентификатор i-й отображаемой переменной из списка переменных, описанных в подзаголовке OUTPUT. Полный идентификатор отображаемой переменной должен содержать полный путь от фрагмента верхнего уровня к собственно идентификатору переменной в подразделе OUTPUT включенного фрагмента. Этот путь состоит из последовательности идентификаторов фрагментов, разделенных символом “/”. Последовательность перечисления фрагментов соответствует порядку включения этих фрагментов друг в друга, при этом в начале последовательности находятся фрагменты более высокого уровня.

<i>номер</i>	Номер компоненты (для многокомпонентных отображаемых переменных). Если номер компоненты отсутствует, предполагается, что задано отображение первой компоненты.
<i>параметр_{ij}</i>	j-й позиционный параметр, характеризующий i-ю отображаемую переменную. Назначение и количество этих параметров определяется в паспорте программы отображения. В случае, если в списке параметров j-й параметр пропущен, его отсутствие отмечается запятой. Отсутствие позиционного параметра означает, что программа отображения должна определить его значение автоматически.

Если необходимо в состав отображаемых переменных включить весь список переменных, определяемый совокупностью всех подразделов #OUTPUT программы, то вызов программы отображения не содержит списка отдельных отображаемых переменных и имеет вид:

[~идентификатор]' ~имя (имя₁ = число [, имя₂ = число [, ...имя_n = число])

Пример. Описания вызовов программ отображения могут выглядеть следующим образом:

```

Основные характеристики процесса 'DISP ( ;
    Сила на пружине = (0, 1.e6) ,
    Перемещение по оси X = (10, Амплитуда ) ,
    Скорость по оси X,
    Перемещение опоры = ( , 0.001) )
Все параметры процесса 'DISP (END=1.)
Таблица отдельных компонентов 'TABL ( OUT=1.e-1, START=0.5;
    Пружина/Деформация(2) = (0, ) ,
    Показатель травмируемости(3) )

```

В приведенных примерах предполагается, что ранее в разделах описания данных определен список параметров "Амплитуда", в библиотеке программ отображения имеются программы DISP, TABL, в подразделе OUTPUT фрагмента с идентификатором "Пружина", включенного непосредственно во фрагмент верхнего уровня, определен вывод многокомпонентной переменной "Деформация", в подразделе OUTPUT фрагмента верхнего уровня определен вывод переменных "Сила на пружине", "Перемещение по оси X", "Скорость по оси X", "Перемещение в опоре", "Показатель травмируемости".

7. ВОЗМОЖНАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ РАЗДЕЛОВ И ПОДРАЗДЕЛОВ В ПРОГРАММЕ НА ЯЗЫКЕ PradiSLang

Как следует из настоящего документа, большинство разделов и подразделов в программе на языке *PradiSLang* может появляться в строго определенном порядке. Здесь представляется уместным привести возможный порядок следования разделов и подразделов.

Можно выделить три отличающихся друг от друга типа программ на языке *PradiSLang*. Это, во-первых, программа, предусматривающая формирование новой модели объекта с последующим выполнением задания. Во-вторых, это программа, предусматривающая выполнение расчета для уже сформированной модели. И, в-третьих, это программа, предусматривающая отображение полученных ранее для модели результатов.

В списке перечисления разделов и подразделов символами (***) обозначены обязательные разделы и подразделы. Разделы и подразделы, отмеченные символами (**) должны появляться в программе хотя бы один раз, отмеченные символами (*) не являются обязательными, но их отсутствие влечет за собой выдачу предупреждающего сообщения.

Последовательность разделов и подразделов в случае формирования новой модели объекта:

1. \$ **DATA** неименованный (глобальный)
2. \$ **DATA** именованные и \$ **FRAGMENT** именованные.

Именованный блок данных обязательно предшествует соответствующему фрагменту.

Порядок следования подразделов в разделе \$FRAGMENT:

- (*) 2.a # **BASE**
- (***) 2.b # **STRUCTURE**
- (**) 2.c # **OUTPUT**
- 2.d # **MAP** и # **EXTERNAL**

для # MAP и # EXTERNAL строгий порядок не определен, но им обязательно должен предшествовать подраздел # STRUCTURE.

- (***) 3. \$ **FRAGMENT** именованный или неименованный (глобальный).
- 4. \$ **SHOW**
- (***) 5. \$ **RUN**
- (***) 6. \$ **PRINT**
- (***) 7. \$ **END**

Последовательность разделов в случае выполнения расчета для уже сформированной модели:

- 1. \$ **DATA** неименованный (глобальный)
- 3. \$ **REPLACE**
- (*) 4. \$ **RESTORE**
- (***) 5. \$ **RUN**
- (***) 6. \$ **PRINT**
- (***) 7. \$ **END**

Последовательность разделов в случае выполнения задания на отображение результатов:

- 1. \$ **DATA** неименованный (глобальный)
- (***) 3. \$ **PRINT**
- (***) 4. \$ **END**

8. ИСПОЛЬЗОВАНИЕ ИНСТРУКЦИИ ПРЕПРОЦЕССОРА \$ INCLUDE

Для включения в текст текущего задания текста другого файла, используется инструкция препроцессора \$INCLUDE.

Формат инструкции:

\$ INCLUDE : *имя_файла*

В строке, содержащей \$INCLUDE, не должно присутствовать других предложений входного языка.

Все содержимое указанного файла включается в то место программы, где расположена эта инструкция. Если включаемый файл сам содержит инструкции \$INCLUDE, то перед включением этого файла в текст текущего задания выполняются инструкции \$INCLUDE для текста во включаемом файле. Вложенность инструкций \$INCLUDE не ограничена. Инструкции \$INCLUDE могут быть использованы в любом месте программы. С их помощью в обрабатываемое задание может быть включен любой фрагмент программы на входном языке *PradiSLang*.

Инструкция \$INCLUDE выполняется на стадии препроцессорной обработки задания, после чего следует синтаксический анализ текста. В случае, если указанный для включения в текст текущего задания файл не будет обнаружен препроцессором, выдается соответствующее сообщение и обработка текущего задания прекращается.

9. РУССКОЯЗЫЧНЫЕ СИНОНИМЫ ЗАГОЛОВКОВ И ПОДЗАГОЛОВКОВ ЯЗЫКА PradiSLang

Для разработки приложений могут оказаться полезными русскоязычные синонимы заголовков и подзаголовков входного языка:

\$ DATA:	\$ ДАННЫЕ:
\$ FRAGMENT:	\$ ФРАГМЕНТ:
# BASE:	# БАЗА:
# STRUCTURE:	# СТРУКТУРА:
# OUTPUT:	# ВЫВОД:
# EXTERNAL:	# ВНЕШНИЕ:
\$ SHOW:	\$ ПРОСМОТР:
\$ RESTORE:	\$ ПРОДОЛЖИТЬ:
\$ REPLACE:	\$ ЗАМЕНИТЬ:
\$ RUN:	\$ ВЫПОЛНИТЬ:
\$ PRINT:	\$ ОТОБРАЗИТЬ:
\$ END	\$ КОНЕЦ

Правила использования русскоязычных синонимов полностью эквивалентны правилам использования соответствующих основных конструкций входного языка.

10. ПРИМЕРЫ ПРОГРАММ НА ЯЗЫКЕ PradiSLang

10.1. ПРОГРАММА, СОДЕРЖАЩАЯ ОПИСАНИЕ ОБЪЕКТА И ОПИСАНИЕ ЗАДАНИЯ НА РАСЧЕТ И ОТОБРАЖЕНИЕ РЕЗУЛЬТАТОВ

Математическая модель стержневого маятника с упругой опорой и сосредоточенными на концах маятника точечными инерционными элементами. С осью маятника через редуктор соединяется двигатель с линейной механической характеристикой.

```
$ DATA:
  { Описание глобальных данных }
  Модуль упругости = 2.E11 {Па} ;
  Масштаб          = 1.; {масштаб для выводимых величин}
$ DATA : Привод
  { Описание данных для фрагмента Привод }
  Пусковой момент двигателя = 45      {н*м}
  Скорость холостого хода   = 2        {1/с}
  К.п.д. редуктора          = 0.95;
  Передаточное отношение   = 2
  Номинальный момент       = 100.;
  Жесткость = 1.E6 {н*м}
  Параметры двигателя =
  Пусковой момент двигателя,
  Скорость холостого хода
  Параметры редуктора =
  Передаточное отношение,
  К.п.д. редуктора,
  Номинальный момент, Жесткость
  { Моменты инерции ротора и маховика задаются в описании структуры привода. }
$ FRAGMENT : Привод
  # BASE : 1 { Базовая - первая степень свободы}
  # STRUCT :
    Двигатель ' DVL (2 1; Параметры двигателя);
    Ротор ' М (2 ; 0.1);
    Редуктор ' REDN (2 3; Параметры редуктора);
    Маховик ' М (3 ; 1);
  # EXTERNAL : 1, 3
  { Степени свободы, по которым описываемый фрагмент соединяется с другими фрагментами системы}
  # OUTPUT :
    Угловая скорость двигателя 'V (2; Масштаб);
$ DATA : Маятник
  { Описание данных для фрагмента Маятник}
  Точка А = 0., 0; Точка В = 1, 0.; {координаты точек}
  Материал = 1.E-8, 1.E-5, Модуль упругости; {J, А, Е}
  Масса = 10.; Момент инерции = 0.1;
  Жесткость опоры = 1.E4 {н*м}
  Сила тяжести = 98.1;
$ FRAGMENT : Маятник
  # BASE : 1
  # STRUCT :
    Опора Х ' К (2 1; Жесткость опоры); Опора У ' К (3 1;
                                           Жесткость опоры)
    Стержень ' BALKAD (2 3 4 5 6 7; Точка А,
                                   Точка В,
                                   Материал)
```

```

{Точечные инерционные элементы}
      'M (2 ; Масса); 'M (3 ; Масса);
      'M (5 ; Масса); 'M (6 ; Масса);
      'M (7 ; Момент инерции);
{Включение описанного выше фрагмента "Привод"}
Привод ' Привод (1 4);
{Силы тяжести }
Сила в опоре 'F ( 3; Сила тяжести);
Сила на свободном конце ' F ( 6; Сила тяжести);
# OUTPUT
Момент на двигателе 'X (I:Двигатель; Масштаб)
Перемещение в опоре по оси X 'S (2; Масштаб)
Перемещение в опоре по оси Y 'S (3; Масштаб)
Угол поворота маятника 'S (4; Масштаб)
Перемещение свободного конца X 'S (5; Масштаб)
Перемещение свободного конца Y 'S (6; Масштаб)
Момент на выходном валу редуктора 'X (I:Редуктор(2);1.)
Реакция в опоре по оси X 'X (I:Опора X; Масштаб)
Реакция в опоре по оси Y 'X (I:Опора Y; Масштаб)
$ RUN
{Задание на расчет переходного процесса}
Расчет поведения маятника 'SHTERM ( END = 1.,
                                     {считать 1с процесса}
                                     SAVE = 0.1)
{проводить сохранение текущего состояния расчета каждые 0.1с
                                     процесса}
$ PRINT
Результаты расчета маятника ' DISP ()
{строить графики для всех выводимых величин}
$ END

```

10.2. ПРОГРАММА, РЕАЛИЗУЮЩАЯ ПРОДОЛЖЕНИЕ РАСЧЕТА С ПОСЛЕДНЕГО МЕСТА СОХРАНЕНИЯ РЕЗУЛЬТАТОВ

```
$ RESTORE {восстановить состояние расчета}
$ RUN :
    Продолжение расчета ' SHTERM (END = 1.5)
$ PRINT :
    Результаты расчета маятника ' DISP ( ;
                                Угол поворота маятника
    {построить график только этой величины,
     масштабирование автоматическое} )
$ END
```

10.3. ПРОГРАММА, СОДЕРЖАЩАЯ ЗАДАНИЕ НА ОТОБРАЖЕНИЕ РАНЕЕ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ РАСЧЕТА

```
$ PRINT :
    Параметры двигателя ' DISP ( ;
                                Угол поворота маятника,
                                Момент на двигателе,
    {масштаб определяется автоматически}
                                Привод/Угловая скорость двигателя = (,3)
    {нижняя граница графика определяется автоматически,
    верхняя задана (= 3) . } )
Маятник      ' DISP ( ;
                                Угол поворота маятника,
                                Реакция в опоре по оси X,
                                Реакция в опоре по оси Y)

$ END
```