5.3 Стандарт первого уровня компонента сеть

Данный раздел посвящен описанию стандарта хранения компонента сеть на внешних носителях.

5.3.1 Структура компонента

Рассмотрим более подробно структуры данных сети. Как уже было описано в первой части главы, сеть строится иерархически от простых подсетей к сложным. Простейшими подсетями являются элементы. Подсеть каждого уровня имеет свое имя и тип. Существуют следующие типы подсетей: элемент, каскад, слой, цикл с фиксированным числом тактов функционирования и цикл, функционирующий до тех пор, пока не выполнится некоторое условие. Последние четыре типа подсетей будем называть блоками. Имена подсетей определяются при конструировании. В разделе «Имена структурных единиц компонентов» главы «Общий стандарт» приведены правила построения полного и однозначного имен подсети. В качестве примера рассмотрим сеть, конструирование которой проиллюстрировано в первой части главына рис. 2. В описании сети NW однозначное имя первого нейрона второго слоя имеет вид K[2].SN.N[1]. При описании слоя однозначное имя первого нейрона записывается как N[1]. В квадратных скобках указываются номер экземпляра подсети, входящей в непосредственно содержащую ее структуру в нескольких экземплярах.

5.3.2 Сигналы и параметры

При использовании контрастирования для изменения структуры сети и значений обучаемых параметров другим компонентам бывает необходим прямой доступ к сигналам и параметрам сети в целом или отдельных ее подсетей. Для адресации входных и выходных сигналов используются имена InSignals и OutSignals, соответственно. Таким образом, для получения массива входных сигналов второго слоя сети, приведенной на рис. 2, необходимо запросить массив NW.K[2].InSignals, а для получения выходного сигнала всей сети можно воспользоваться любым из следующего списка имен:

- NW.OutSignals;
- NW.N.OutSignals.

Для получения конкретного сигнала из массива сигналов необходимо в конце в квадратных скобках указать номер сигнала. Например, для получения третьего входного сигнала второго слоя сети нужно указать следующее имя – NW.K[2].InSignals[3].

Для получения доступа к параметрам нужно указать имя подсети, к чьим параметрам нужен доступ и через точку ключевое слово Parameters. При необходимости получить конкретный параметр, его номер в квадратных скобках записывается после ключевого слова Parameters.

5.3.3 Обучаемые и не обучаемые параметры и сигналы

При обучении параметров и сигналов (использование обучения сигналов описано во введении) возникает необходимость обучать только часть из них. Так, например, при описании обучения персептрона во второй части этой главы было отмечено, что обучать необходимо только веса связей второго слоя. Для реализации этой возможности используются два массива логических переменных – маска обучаемых параметров и маска обучаемых входных сигналов.

5.3.4 Дополнительные переменные

При описании структуры сетей необходимо учитывать следующую дополнительные переменные, доступные в методах Forw и Back. Для каждой сети при прямом функционировании определен следующий набор переменных:

- InSignals[K] – массив из K действительных чисел, содержащих входные сигналы прямого функционирования.
- OutSignals[N] – массив из N действительных чисел, в которые заносятся выходные сигналы прямого функционирования.
- Parameters[M] – массив из M действительных чисел, содержащих параметры сети.

При выполнении обратного функционирования сети доступны еще три массива:

- Back.InSignals[K] – массив из K действительных чисел, параллельный массиву InSignals, в который заносятся выходные сигналы обратного функционирования.
- Back.OutSignals[N] – массив из N действительных чисел, параллельный массиву OutSignals, содержащий входные сигналы обратного функционирования.
- Back.Parameters[M] – массив из M действительных чисел, параллельный массиву Parameters, в который заносятся вычисленные при обратном функционировании поправки к параметрам сети.

При обучении (модификации параметров или входных сигналов) доступны все переменные обратного функционирования и еще два массива:

- InSignalMask[K] – массив из K логических переменных, параллельный массиву InSignals, содержащий маску обучаемости входных сигналов.
- ParamMask[M] – массив из M логических переменных, параллельный массиву Parameters, содержащий маску обучаемости параметров.

5.3.5 Стандарт языка описания сетей

Язык описания нейронных сетей предназначен для хранения сетей на диске. Следует отметить, что в отличии от таких компонентов, как предобработчик входных сигналов, оценка или задачник описание даже простой сети имеет большой размер. С другой стороны, многие подсети являются стандартными для большинства сетей. Для компонента сеть нет смысла вводить небольшой набор стандартных элементов и подсетей, поскольку этот набор может легко расширяться. Более эффективным является выделение часто употребляемых подсетей в отдельные библиотеки, подключаемые к описаниям конкретных сетей. В приведенных в этой главе примерах описания нейронных сетей выделен ряд библиотек.

5.3.5.1 Ключевые слова языка

В табл. 2 приведен список ключевых слов специфических для языка описания сетей.

Таблица 2.

Ключевые слова языка описания сетей.

Ключевое слово	Краткое описание
1. Back	Метод, осуществляющий обратное функционирование подсети. Префикс сигналов обратного функционирования.
2. Block	Тип аргумента подсети. Означает, что аргумент является подсетью.
3. Cascad	Тип подсети – каскад.
4. Connections	Начало блока описания связей подсети.
5. Contents	Начало блока описания состава подсети.
6. DefaultType	Тип параметров по умолчанию.
7. Element	Тип подсети – элемент.
8. Forw	Метод, осуществляющий прямое функционирования подсети.
9. InSignalMask	Имя, по которому адресуются маски обучаемости входных сигналов подсети.
10. InSignals	Имя, по которому адресуются входные сигналы подсети; начало блока описания входных сигналов.
11. Layer	Тип подсети – слой.
12. Loop	Тип подсети – цикл, выполняемый указанное число раз.
13. MainNet	Начало описания главной сети
14. NetLib	Начало описания библиотеки подсетей.
15. NetWork	Начало описания сети.
16. NumberOf	Функция (запрос). Возвращает число параметров или сигналов в подсети.
17. OutSignals	Имя, по которому адресуются выходные сигналы подсети; начало блока описания выходных сигналов.
18. ParamDef	Заголовок определения типа параметров.
19. Parameters	Имя, по которому адресуются параметры подсети; начало блока описания параметров.
20. ParamMask	Имя, по которому адресуются маски обучаемости параметров подсети.
21. ParamType	Заголовок описания типа параметров.
22. Until	Тип подсети – цикл, выполняемый до тех пор пока не выполнится условие.
23. Used	Начало списка подключаемых библиотек подсетей

5.3.5.2 БНФ языка описания сетей

Обозначения, принятые в данном расширении БНФ и описание ряда конструкций приведены в главе “Общий стандарт” в разделе “Описание языка описания компонентов”.

<Описание библиотеки подсетей> ::= <Заголовок библиотеки> <Описание подсетей> <Конец описания библиотеки>

<Заголовок библиотеки> ::= **NetLib** <Имя библиотеки> [**Used** <Список имен библиотек>]

<Имя библиотеки> ::= <Идентификатор>
<Список имен библиотек> ::= <Имя используемой библиотеки> [,<Список имен библиотек>]
<Имя используемой библиотеки> ::= <Идентификатор>
<Описание подсетей> ::= <Описание подсети> [<Описание подсетей>]
<Описание подсети> ::= {<Описание элемента> | <Описание блока> | <Описание функций>}
<Описание элемента> ::= <Заголовок описания элемента> <Описание сигналов и параметров> [<Описание типов параметров>] [<Определение типов параметров>] [<Описание статических переменных>] [<Установление значений статических переменных>] <Описание методов> <Конец описания элемента>
<Заголовок описания элемента> ::= **Element** <Имя элемента> [*(*<Список формальных аргументов>*)*]
<Имя элемента> ::= <Идентификатор>
<Описание сигналов и параметров> ::= <Описание входных сигналов> <Описание выходных сигналов> [<Описание параметров>]
<Описание входных сигналов> ::= **InSignals** <Константное выражение типа ***Long***>
<Описание выходных сигналов> ::= **OutSignals** <Константное выражение типа ***Long***>
<Описание параметров> ::= **Parameters** <Константное выражение типа ***Long***>
<Описание типов параметров> ::= <Описание типа параметров> [<Описание типов параметров>]
<Описание типа параметров> ::= **ParamType** <Имя типа параметра> <Список>
<Имя типа параметра> ::= <Идентификатор>
<Список> ::= {**Parameters/**<Начальный номер> [..<Конечный номер> [<Шаг>]]*/* | **InSignals/**<Начальный номер> [..<Конечный номер> [<Шаг>]]*/*} [;<Список>]
<Определение типов параметров> ::= <Определение типа параметра> [<Определение типов параметров>]
<Определение типа параметра> ::= **ParamDef** <Имя типа параметра> <Минимальное значение> <Максимальное значение>
<Минимальное значение> ::= <Константное выражение типа ***Real***>
<Максимальное значение> ::= <Константное выражение типа ***Real***>
<Установление значений статических переменных> ::= <Установление параметров ***Подсети***> [;<Установление значений статических переменных>]
<Описание методов> ::= <Описание функционирования вперед> <Описание функционирования назад>
<Описание функционирования вперед> ::= **Forw** [<Описание переменных>] <Тело метода>
<Тело метода> ::= **Begin** <Составной оператор> **End**
<Описание функционирования назад> ::= **Back** [<Описание переменных>] <Тело метода>
<Конец описания элемента> ::= **End** <Имя элемента>
<Описание блока> ::= <Заголовок описания блока> <Описание состава> <Описание сигналов и параметров> [<Описание статических переменных>] [<Установление значений статических переменных>] <Описание связей> [<Определение типов параметров>] <Конец описания блока>
<Заголовок описания блока> ::= {<Описание каскада> | <Описание слоя> | <Описание цикла с фиксированным числом шагов> | <Описание цикла по условию>}
<Описание каскада> ::= **Cascad** <Имя блока> [*(*<Список формальных аргументов блока>*)*]
<Имя блока> ::= <Идентификатор>
<Список формальных аргументов блока> ::= {<Список формальных аргументов> | <Аргумент – подсеть>} [;<Список формальных аргументов блока>]
<Аргумент – подсеть>::= <Список имен аргументов – подсетей> : **Block**
<Список имен аргументов – подсетей> ::= <Имя аргумента – подсети> [,<Список имен аргументов – подсетей>]
<Имя аргумента – подсети> ::= <Идентификатор>
<Описание слоя> ::= **Layer** <Имя блока> [*(*<Список формальных аргументов блока>*)*]
<Описание цикла с фиксированным числом шагов> ::= **Loop** <Имя блока> [*(*<Список формальных аргументов блока>*)*] <Число повторов цикла>
<Число повторов цикла> ::= <Константное выражение типа ***Long***>
<Описание цикла по условию> ::= **Until** <Имя блока> [*(*<Список формальных аргументов блока>*)*] : <Выражение типа ***Logic***>
<Описание состава> ::= **Contents** <Список имен подсетей>
<Список имен подсетей> ::= <Имя подсети> [,<Список имен подсетей>]
<Имя подсети> ::= <Псевдоним>: {<Имя ранее описанной подсети> [*(*<Список фактических аргументов блока>*)*] [*/*<Число экземпляров>*/*] | <Имя аргумента – подсети> [*/*<Число экземпляров >*/*]}
<Псевдоним> ::= <Идентификатор>

<Число экземпляров > ::= <Константное выражение типа ***Long***>
<Имя ранее описанной подсети> ::= <Идентификатор>
<Список фактических аргументов блока> ::= <Фактический аргумент блока> [,<Список фактических аргументов блока>]
<Фактический аргумент блока> ::= {<Фактический аргумент> | <Имя аргумента – подсети>}
<Описание связей> ::= {<Описание распределения ***Входных сигналов, Блока, Подсети, InSignals*** > | <Описание распределения ***Выходных сигналов, Блока, Подсети, OutSignals*** > | <Описание распределения ***Параметров, Блока, Подсети, Parameters*** >}
<Конец описания блока>::= **End** <Имя блока>
<Конец описания библиотеки> ::= **End NetLib**
<Описание сети> ::= <Заголовок описания сети> <Описание подсетей> <Описание главной сети> <Массивы параметров и масок сети> <Конец описания сети>
<Заголовок описания сети> ::= **NetWork** <Имя сети> [**Used** <Список имен библиотек>]
<Имя сети> ::= <Идентификатор>
<Описание главной сети> ::= **MainNet** <Имя ранее описанной подсети> [*(*<Список фактических аргументов блока>*)*]
<Массивы параметров и масок сети> ::= <Массив параметров> <Массив маски обучаемости параметров>
<Массив параметров> ::= **Parameters** <Значения параметров>;
<Значения параметров> ::= <Действительное число> [, <Значения параметров>]
<Массив маски обучаемости параметров> ::= **ParamMask** <Значения маски>;
<Значения маски> ::= <Константа типа ***Logic***> [,<Значения маски>]
<Конец описания сети> ::= **End NetWork**

5.3.5.3 Описание языка описания сетей

В этом разделе приводится детальное описание языка описания сетей, дополняющее БНФ, приведенную в предыдущем разделе и описание общих конструкций, приведенное в главе «Общий стандарт».

5.3.5.3.1 Описание и область действия переменных

Вспомогательные переменные могут потребоваться при описании прямого и обратного функционирования элементов. Переменная действует только в пределах той процедуры, в которой она описана. Кроме явно описанных переменных, в методе Forw доступны также сигналы прямого функционирования и параметры элемента, а в методе Back – входные и выходные сигналы прямого функционирования, выходные сигналы обратного функционирования, параметры элемента и градиент по параметрам элемента. Во всех методах доступны аргументы элемента.

Статические переменные, описываемые после ключевого слова Static, уникальны для каждого экземпляра элемента или блока, и доступны только в пределах блока. Эти переменные могут потребоваться для вычисления условий в цикле типа Until. Возможно использование таких переменных в элементах, например, для хранения предыдущего состояния элемента. Кроме того, в статической переменной можно хранить значения не обучаемых параметров.

5.3.5.3.2 Методы Forw и Back для блоков

Методы Forw и Back для блоков не описываются в языке описания сетей. Это связано с тем, что при выполнении метода Forw блоком происходит вызов метода Forw составляющих блок подсетей (для элементов – метода Forw) в порядке их описания в разделе описания состава блока. При выполнении метода Back происходит вызов методов Back составляющих блок подсетей в порядке обратном порядку их описания в разделе описания состава блока.

5.3.5.3.3 Описание элементов

Описание элемента состоит из следующих основных разделов: заголовка элемента, описания сигналов и параметров, описания статических переменных и описания методов. Заголовок элемента имеет следующий синтаксис:

Element Имя_Элемента (Аргументы элемента)

Аргументы элемента являются необязательной частью заголовка. В следующем разделе приведены описания нескольких элементов. Отметим, что сигмоидный элемент описан двумя способами: с принципиально не обучаемой (S_NotTrain) и с обучаемой (S_Train) характеристикой.

Раздел описания сигналов и параметров следует сразу после заголовка элемента и состоит из указания числа входных и выходных сигналов и числа параметров элемента. Если у элемента отсутствуют параметры, то указание числа параметров можно опустить. В следующем разделе приведены элементы как имеющие параметры (S_Train, Adaptiv_Sum, Square_Sum), так и элементы без параметров (Sum, S_NotTrain, Branch). Концом раздела описания сигналов и параметров служит одно из ключевых слов ParamType, ParamDef, Forw или Back.

Описание типов параметров является необязательной частью описания элемента и начинается с ключевого слова ParamType. Если раздел описания типов параметров отсутствует, то все параметры этого элемента считаются параметрами типа DefaultType. Если в сети должны присутствовать параметры разных типов (например с разными ограничениями на минимальное и максимальное значение) необходимо описать типы параметров. Концом этого раздела служит одно из ключевых слов ParamDef, Forw или Back.

Раздел определения типов параметров является необязательным разделом в описании элемента и начинается с ключевого слова ParamDef. В каждой строке этого раздела можно задать минимальную и масимальную границы изменения одного типа параметров. Если в описании сети встречаются параметры неопределенного типа то этот тип считается совпадающим с типом DefaultType. Описание типа *не обязано* предшествовать описанию параметров этого типа. Так например, определение типа параметров может находиться в описании главной сети. Концом этого раздела служит одно из ключевых слов Forw или Back.

Раздел описания методов состоит из описания двух методов: Forw и Back. Описание метода состоит из заголовка, раздела описания переменных и тела метода. Заголовок имеет вид ключевого слова Forw или Back для соответствующего метода. Раздел описания переменных состоит из ключевого слова Var, за которым следуют описания однотипных переменных, каждое из которых заканчивается символом «;». Необходимо понимать, что описание заголовков методов это не описание заголовка (прототипа) функции, выполняющей тело метода. Ниже приведен синтаксис заголовков методов Forw и Back на момент вызова:

Pascal:

```
Procedure Forw( InSignals, OutSignals, Parameters : PRealArray);
Procedure Back(InSignals, OutSignals, Parameters, Back.InSignals,
    Back.OutSignals, Back.Parameters : PRealArray);
```

C

```
void Forw(PRealArray InSignals, PRealArray OutSignals, PRealArray Parameters)
void Back(PRealArray InSignals, PRealArray OutSignals, PRealArray Parameters,
PRealArray Back.InSignals, PRealArray Back.OutSignals, PRealArray Back.Parameters)
```

В методе Forw в левой части оператора присваивания могут фигурировать имена любых переменных и элементов предопределенного массива выходных сигналов (OutSignals). В выражении, стоящем в правой части оператора присваивания могут участвовать любые переменные, аргументы элемента и элементы предопределенных массивов входных сигналов (InSignals) и параметров (Parameters).

В методе Back в левой части оператора присваивания могут фигурировать имена любых переменных, элементов предопределенных массивов входных сигналов обратного функционирования (Back.InSignals) и параметров (Back.Parameters). В выражении, стоящем в правой части оператора присваивания, могут участвовать любые переменные, аргументы элемента и элементы предопределенных массивов входных (InSignals) и выходных (OutSignals) сигналов и параметров (Parameters).Отметим важную особенность вычисления поправок к параметрам. Поскольку один и тот же параметр может использоваться несколькими элементами, при вычислении поправки к параметру вычисленное значение нужно не присваивать соответствующему элементу массива Back.Parameters, а добавлять. При этом в теле метода элементы массива Back.Parameters не могут фигурировать в правой части оператора присваивания. Эта особенность вычисления поправок к параметрам обрабатывается компонентом сеть.

Описание элемента завершается ключевым словом End за которым следует имя элемента.

5.3.5.3.4 Пример описания элементов

```
NetBibl Elements;                    {Библиотека элементов}

Element Synaps                       {Обычный синапс}
   InSignals 1                       {Один входной сигнал}
   OutSignals 1                      {Один выходной сигнал}
   Parameters 1                      {Один параметр – вес связи}
```

```
    Forw                                    {Начало описания прямого функционирования}
    Begin
            {Вычисление выходного сигнала как произведения входного сигнала на параметр}
        OutSignals[1] = InSignals[1] * Parameters[1]
    End                                     {Конец описания прямого функционирования}

    Back                                    {Начало описания обратного функционирования }
    Begin
            {Вычисление поправки к входному сигналу как произведения поправки к выходному
            сигналу на параметр}
        Back.InSignals[1] = Back.OutSignals[1] * Parameters[1];
            {Вычисление поправки к параметру как суммы ранее вычисленной поправки к
            параметру на произведение поправки к обратному  сигналу на входной сигнал}
        Back.Parameters[1] = Back.Parameters[1] + Back.OutSignals[1] * InSignals[1]
    End                                     {Конец описания обратного функционирования}
End Synaps                                  {Конец описания синапса}

Element Branch(N : Long)                    {Точка ветвления на N выходных сигналов}
    InSignals 1                             {Один входной сигнал}
    OutSignals N                            {N выходных сигналов}

    Forw                                    {Начало описания прямого функционирования}
        Var Long I;                     {I – локальная переменная типа длинное целое – индекс}
    Begin
        For I=1 To N Do                     {На каждый из N выходных сигналов}
            OutSignals[I] = InSignals[1]    {передаем входной сигнал}
    End                                     {Конец описания прямого функционирования}

    Back                                    {Начало описания обратного функционирования }
        Var                                 {Описание локальных переменных}
            Long I;                         {I – длинное целое – индекс}
            Real R;                         {R – действительное – для накопления суммы}
    Begin
        R = 0;
        For I=1 To N Do                     {Поправка ко входному сигналу равна}
            R = R + Back.OutSignals[I];     {сумме поправок выходных сигналов}
        Back. InSignals[1] = R
    End                                     {Конец описания обратного функционирования}
End Branch                                  {Конец описания точки ветвления}

Element Sum(N Long)                         {Простой сумматор на N входов}
    InSignals N                             {N входных сигналов}
    OutSignals 1                            {Один выходной сигнал}

    Forw                                    {Начало описания прямого функционирования}
        Var                                 {Описание локальных переменных}
            Long I;                         {I – длинное целое – индекс}
            Real R;                         {R – действительное – для накопления суммы}
    Begin
        R = 0;
        For I=1 To N Do                     {Выходной сигнал равен сумме входных}
            R = R + InSignals[I];
        OutSignals[1] = R
    End                                     {Конец описания прямого функционирования}

    Back                                    {Начало описания обратного функционирования}
        Var Long I;                     {I – локальная переменная типа длинное целое – индекс}
    Begin
        For I=1 To N Do                     {Поправка к каждому входному сигналу}
            Back.InSignals[I] = Back.OutSignals[1]    {равна поправке выходного сигнала}
    End                                     {Конец описания обратного функционирования}
End Sum                                     {Конец описания простого сумматора}
```

```
Element Mul                                  {Умножитель}
   InSignals 2                               {Два входных сигнала}
   OutSignals 1                              {Один выходной сигнал}

   Forw                                      {Начало описания прямого функционирования }
   Begin
            {Выходной сигнал равен произведению входных сигналов}
      OutSignals[1] = InSignals[1] * InSignals[2]
   End                                       {Конец описания прямого функционирования}

   Back                                      {Начало описания обратного функционирования }
   Begin
            {Поправка к каждому входному сигналу равна произведению поправки выходного
            сигнала на другой входной сигнал}
      Back.InSignals[1] = Back.OutSignals[1] * InSignals[2];
      Back.InSignals[2] = Back.OutSignals[1] * InSignals[1]
   End                                       {Конец описания обратного функционирования}
End Mul                                      {Конец описания умножителя}

Element S_Train          {Обучаемый гиперболический сигмоидный элемент}
   InSignals 1                               {Один входной сигнал}
   OutSignals 1                              {Один выходной сигнал}
   Parameters 1                                 {Один параметр – характеристика}

   Forw                                      {Начало описания прямого функционирования}
   Begin
            {Выходной сигнал равен отношению входного сигнала к сумме параметра и
            абсолютной величины входного сигнала}
      OutSignals[1] = InSignals[1] / (Parameters[1] + Abs(InSignals[1])
   End                                       {Конец описания прямого функционирования}

   Back                                      {Начало описания обратного функционирования}
      Var Real R;                            {R – действительное}
   Begin
            {R – вспомогательная величина для вычисления поправок, равная отношению
            поправки выходного сигнала к квадрату суммы параметра и абсолютной величины
            входного сигнала}
      R = Back.OutSignals[1] / Sqr(Parameters[1] + Abs(InSignals[1]);
            {Поправка к входному сигналу равна произведению вспомогательной величины на
            параметр}
      Back.InSignals[1] = R * Parameters[1];
            {Поправка к параметру равна сумме ранее вычисленной величины поправки и
            произведения вспомогательной величины на входной сигнал}
      Back.Parameters[1] = Back.Parameters[1] + R * InSignals[1]
   End                                       {Конец описания обратного функционирования}
End S_Train              {Конец описания обучаемого гиперболического сигмоидного элемента}

Element S_NotTrain( Char : Real)     {Не обучаемый гиперболический сигмоидный элемент}
                                             {Char – характеристика}
   InSignals 1                               {Один входной сигнал}
   OutSignals 1                              {Один выходной сигнал}

   Forw                                      {Начало описания прямого функционирования}
   Begin
            {Выходной сигнал равен отношению входного сигнала к сумме характеристики и
            абсолютной величины входного сигнала}
      OutSignals[1] = InSignals[1] / (Char + Abs(InSignals[1])
   End                                       {Конец описания прямого функционирования}

   Back                                      {Начало описания обратного функционирования}
   Begin
            {Поправка к входному сигналу равна отношению произведения поправки выходного
            сигнала на характеристику к квадрату суммы характеристики и абсолютной
```

```
            величины входного сигнала}
        Back.InSignals[1]  = Back.OutSignals[1] * Char /  Sqr(Char + Abs(InSignals[1]);
    End                                         {Конец описания обратного функционирования}
End S_NotTrain          {Конец описания не обучаемого гиперболического сигмоидного элемента}

Element Pade(Char : Real)                       {Паде преобразователь Char  – характеристика}
    InSignals 2                                 {Два входных сигнала}
    OutSignals 1                                {Один выходной сигнал}

    Forw                                        {Начало описания прямого функционирования}
    Begin
            {Выходной сигнал равен отношению первого входного сигнала к сумме
            характеристики и второго входного сигнала}
        OutSignals[1] = InSignals[1] / (Char+ InSignals[2])
    End                                         {Конец описания прямого функционирования}

    Back                                        {Начало описания обратного функционирования}
        Var Real R;                             {R – действительное}
    Begin
            {Вспомогательная величина равна поправке к первому входному сигналу –
            отношению поправки выходного сигнала к сумме характеристики и второго
            входного сигнала}
        R = Back.OutSignals[1] / (Char + InSignals[2]);
        Back.InSignals[1] = R;
            {Поправка ко второму входному сигналу равна минус отношению произведения
            первого входного сигнала на поправку выходного сигнала к квадрату суммы
            характеристики и второго входного сигнала}
        Back.InSignals[2] =  -R * OutSignals[1];
    End                                         {Конец описания обратного функционирования}
End Pade                                        {Конец описания Паде преобразователя}

Element Sign_Mirror                             {Зеркальный пороговый элемент}
    InSignals 1                                 {Один входной сигнал}
    OutSignals 1                                {Один выходной сигнал}

    Forw                                        {Начало описания прямого функционирования }
    Begin
        If InSignals[1] > 0 Then OutSignals[1] = 1  {Выходной сигнал равен 1, если входной сигнал}
                            Else OutSignals[1] = 0  {больше нуля, и нулю в противном случае}
    End                                         {Конец описания прямого функционирования}

    Back                                        {Начало описания обратного функционирования}
    Begin
        Back.InSignals[1] = OutSignals[1];  {Поправка к входному сигналу равна выходному сигналу}
    End                                         {Конец описания обратного функционирования}
End Sign_Mirror                                 {Конец описания зеркального порогового элемента}

Element Sign_ Easy                              {Прозрачный  пороговый элемент}
    InSignals 1                                 {Один входной сигнал}
    OutSignals 1                                {Один выходной сигнал}

    Forw                                        {Начало описания прямого функционирования }
    Begin
        If InSignals[1] > 0 Then OutSignals[1] = 1  {Выходной сигнал равен 1, если входной сигнал}
                            Else OutSignals[1] = 0  {больше нуля, и нулю в противном случае}
    End                                         {Конец описания прямого функционирования}

    Back                                        {Начало описания обратного функционирования}
    Begin
            {Поправка к входному сигналу равна поправке к выходному сигналу}
        Back.InSignals[1] = Back.OutSignals[1];
    End                                         {Конец описания обратного функционирования}
End Sign_Easy                                   {Конец описания прозрачного порогового элемента}
```

```
Element Adaptiv_Sum( N : Long)                {Адаптивный сумматор на N входов}
    InSignals N                                   {N входных сигналов}
    OutSignals 1                                  {Один выходной сигнал}
    Parameters N                                  {N параметров – весов связей}

    Forw                                          {Начало описания прямого функционирования}
        Var                                       {Описание локальных переменных}
            Long I;                               {I – длинное целое – индекс}
            Real R;                               {R – действительное – для накопления суммы}
    Begin
        R = 0;                                    {Выходной сигнал равен скалярному }
        For I=1 To N Do                           {произведению массива входных сигналов}
            R = R + InSignals[I] * Parameters[I];  {на массив параметров}
        OutSignals[1] = R
    End                                           {Конец описания обратного функционирования}

    Back                                          {Начало описания обратного функционирования}
        Var Long I;                               {I – локальная переменная типа}
    Begin                                         {длинное целое – индекс}
        For I=1 To N Do Begin
                {Поправка к I-у входному сигналу равна сумме ранее вычисленной поправки и
                произведения поправки выходного сигнала на I-й параметр}
            Back.InSignals[I] = Back.OutSignals[1] * Parameters[I];
                {Поправка к I-у параметру равна произведению поправки выходного сигнала на I-й
                входной сигнал}
            Back. Parameters[I] = Back. Parameters[I] + Back.OutSignals[1] * InSignals[I]
        End
    End                                           {Конец описания обратного функционирования}
End Adaptiv_Sum                                   {Конец описания адаптивного сумматора}

Element Adaptiv_Sum_Plus ( N : Long)          {Адаптивный неоднородный сумматор на N входов}
    InSignals N                                   {N входных сигналов}
    OutSignals 1                                  {Один выходной сигнал}
    Parameters N+1                              {N+1 параметр – веса связей}

    Forw                                          {Начало описания прямого функционирования }
        Var                                       {Описание локальных переменных}
            Long I;                               {I – длинное целое – индекс}
            Real R;                               {R – действительное – для накопления суммы}
    Begin
        R = Parameters[N+1];                      {Выходной сигнал равен сумме N+1 параметра}
        For I=1 To N Do                           {и скалярного произведения массива входных}
            R = R + InSignals[I] * Parameters[I]; {сигналов на массив параметров}
        OutSignals[1] = R
    End                                           {Конец описания прямого функционирования}

    Back                                          {Начало описания обратного функционирования }
        Var Long I;                               {I – локальная переменная типа}
    Begin                                         {длинное целое – индекс}
        For I=1 To N Do Begin
                {Поправка к I-у входному сигналу равна произведению поправки выходного сигнала
                на I-й параметр}
            Back.InSignals[I] = Back.OutSignals[1] * Parameters[I];
                {Поправка к I-у параметру равна сумме ранее вычисленной поправки и произведения
                поправки выходного сигнала на I-й входной
                сигнал}
            Back. Parameters[I] = Back. Parameters[I] + Back.OutSignals[1] * InSignals[I]
        End;
                {Поправка к (N+1)-у параметру равна сумме ранее вычисленной поправки и попраки
                к выходному сигналу}
        Back.Parameters[N+1] = Back.Parameters[N+1] + Back.OutSignals[1]
```

```
    End                                     {Конец описания обратного функционирования}
End Adaptiv_Sum_Plus                        {Конец описания неоднородного адаптивного сумматора}

Element Square_Sum( N : Long)               {Квадратичный сумматор на N входов}
    InSignals N                             {N входных сигналов}
    OutSignals 1                            {Один выходной сигнал}
    Parameters (Sqr(N) + N) Div 2           {N(N+1)/2 параметров – весов связей}

    Forw                                    {Начало описания прямого функционирования}
        Var                                 {Описание локальных переменных}
            Long I,J,K;                     {I,J,K – переменные типа длинное целое }
            Real R;                         {R – действительное – для накопления суммы}
    Begin
        K = 1;                              {K – номер обрабатываемого параметра}
        R = 0;
        For I = 1 To N Do                   {I,J – номера входных сигналов}
            For J = I To N Do Begin
                R = R + InSignals[I] * InSignals[J] * Parameters[K];
                K = K + 1
            End;
                {Выходной сигнал равен сумме всех попарных произведений входных сигналов,
                 умноженных на соответствующие параметры}
        OutSignals[1] = R
    End                                     {Конец описания прямого функционирования}

    Back                                    {Начало описания обратного функционирования }
        Var                                 {Описание локальных переменных}
            Long I, J, K;                   {I,J,K – переменные типа длинное целое }
            Real R;                         {R – действительное}
            Vector W;                       {Массив для накопления промежуточных величин}
    Begin
        For I = 1 To N Do
            W[I] = 0;
        K = 1;                              {K – номер обрабатываемого параметра}
        For I = 1 To N Do
            For J = I To N Do Begin
                {Поправка к параметру равна сумме ранее вычисленной поправки и произведения
                поправки к входному сигналу на произведение сигналов, прошедших через этот
                параметр при прямом функционировании}
                Back.Parameters[K] = Back.Parameters[K] +
                    Back.OutSignals[1] * InSignals[I] * InSignals[J];
                R = Back.OutSignals[1] * Parameters[K];
                W[I] = W[I] + R * InSignals[J];
                W[J] = W[J] + R * InSignals[I];
                K = K + 1
            End;
        For I = 1 To N Do
                {Поправка к входному сигналу равна произведению поправки к выходному сигналу
                на сумму всех параметров, через которые этот сигнал проходил при прямом
                функционировании, умноженных на другие входные сигналы, так же прошедшие
                через эти параметры при прямом функционировании}
            Back.InSignals[1] = W[I]
    End                                     {Конец описания прямого функционирования}
End Square_Sum                              {Конец описания квадратичного сумматора}

Element Square_Sum_Plus( N : Long)          {Адаптивный квадратичный сумматор на N входов}
    InSignals N                             {N входных сигналов}
    OutSignals 1                            {Один выходной сигнал}
    Parameters (Sqr(N) + 3 * N) Div 2 + 1   {N(N+3)/2+1 параметров – весов связей}

    Forw                                    {Начало описания прямого функционирования}
```

```
    Var                                  {Описание локальных переменных}
      Long I, J, K;                      {I,J,K – переменные типа длинное целое }
      Real R;                            {R – действительное – для накопления суммы}
  Begin
    K = 2 * N+1;                         {K – номер обрабатываемого параметра}
    R = Parameters[Sqr(N) + 3 * N) Div 2 + 1];
    For I = 1 To N Do Begin
      R = R + InSignals[I] * Parameters[I] + Sqr(InSignals[I]) * Parameters[N + I];
      For J = I + 1 To N Do Begin
        R = R + InSignals[I] * InSignals[J] * Parameters[K];
        K = K + 1
      End
    End
        {Выходной сигнал равен сумме всех попарных произведений входных сигналов,
      умноженных на соответствующие параметры, плюс сумме всех входных сигналов
    умноженных на соответствующие параметры, плюс последний параметр}
    OutSignals[1] = R
  End                                    {Конец описания прямого функционирования}

  Back                                   {Начало описания обратного функционирования }
    Var                                  {Описание локальных переменных}
      Long I, J, K;                      {I,J,K – переменные типа длинное целое }
      Real R;                            {R – действительное – для накопления суммы}
      Vector W;                          {Массив для накопления промежуточных величин}
  Begin
    For I = 1 To N Do
      W[I] = 0;
    K = 2 * N + 1;                       {K – номер обрабатываемого параметра}
    For I = 1 To N Do Begin
      Back.Parameters[I] = Back.Parameters[I] + Back.OutSignals[1] * InSignals[I];
      Back.Parameters[N + I] = Back.Parameters[N + I] + Back.OutSignals[1] * Sqr(InSignals[I]);
      W[I] = W[I] + Back.OutSignals[1] * (Parameters[I] + 2 * Parameters[N + I] * InSignals[I])
      For J = I + 1 To N Do Begin
        Back.Parameters[K] = Back.Parameters[K] +
          Back.OutSignals[1] * InSignals[I] * InSignals[J];
        R = Back.OutSignals[1] * Parameters[K];
        W[I] = W[I] + R * InSignals[J];
        W[J] = W[J] + R * InSignals[I];
        K = K + 1
      End
    End;
    For I = 1 To N Do
      Back.InSignals[1] = W[I]
  End                                    {Конец описания обратного функционирования}
End Square_Sum_Plus                      {Конец описания адаптивного квадратичного сумматора}
End NetBibl
```

5.3.5.3.5 Описание блоков

Описание блока состоит из пяти основных разделов: заголовка описания блока, описания сигналов и параметров, описания состава, описания связей и конца описания блока. Существует два типа блоков – каскад и слой (Layer). Различие между этими двумя типами блоков состоит в том, что подсети, входящие в состав слоя, функционируют параллельно и независимо друг от друга, тогда как составляющие каскад подсети функционируют последовательно, причем каждая следующая подсеть использует результаты работы предыдущих подсетей. В свою очередь существует три вида каскадов – простой каскад (Cascad), цикл с фиксированным числом шагов (Loop) цикл по условию (Until). Различие между тремя видами каскадов очевидно – простой каскад функционирует один раз, цикл Loop функционирует указанное в описании число раз, а цикл Until функционирует до тех пор, пока не выполнится указанное в описании условие. В условии, указываемом в заголовке цикла Until, возможно использование сравнений массивов или интервалов массивов сигналов. Например, запись

InSignals=OutSignals

эквивалентна следующей записи

InSignals[1..N]=OutSignals[1..N]

которая эквивалентна вычислению следующей логической функции:

```
Function Equal(InSignals, OutSignals : RealArray) : Logic;
  Var Long I;
    Logic L
  Begin
    L = True
    For I = 1 To N Do
      L = L And (InSignals[I] = OutSignals[I]);
    Equal = L
  End
```

Раздел описания состава следует сразу после заголовка блока за разделом описания сигналов и параметров и начинается с ключевого слова Contents, за которым следуют имена подсетей (блоков или элементов) со списками фактических аргументов, разделенные запятыми. Все имена подсетей должны предваряться псевдонимами. В дальнейшем указание псевдонима полностью эквивалентно указанию имени подсети со списком фактических аргументов или без, в зависимости от контекста. Признаком конца раздела описания состава подсети служит имя подсети за списком фактических аргументов которого не следует запятая.

Раздел описания сигналов и параметров следует за разделом описания состава и состоит из указания числа входных и выходных сигналов и числа параметров блока. В константных выражениях, указывающих число входных и выходных сигналов и параметров можно использовать дополнительно функцию NumberOf с двумя параметрами. Первым параметром является одно из ключевых слов InSignals, OutSignals, Parameters, а вторым – имя подсети со списком фактических аргументов. Функция NumberOf возвращает число входных или выходных сигналов или параметров (в зависимости от первого аргумента) в подсети, указанной во втором аргументе. Использование этой функции необходимо в случае использования блоком аргументов-подсетей. Концом раздела описания сигналов и параметров служит одно из ключевых слов ParamDef, Static или Connections.

Раздел определения типов параметров является необязательным разделом в описании блока и начинается с ключевого слова ParamDef. В каждой строке этого раздела можно задать минимальную и максимальную границы изменения одного типа параметров. Если в описании сети встречаются параметры неопределенного типа, то этот тип считается совпадающим с типом DefaultType. Описание типа *не обязано* предшествовать описанию параметров этого типа. Так, например, определение типа параметров может находиться в описании главной сети. Концом этого раздела служит одно из ключевых слов Connections.

Раздел описания связей следует за разделом описания сигналов и параметров и начинается с ключевого слова Connections. В разделе «Описание распределения сигналов» главы «Общий стандарт» детально описано распределение связей.

Раздел конца описания блока состоит из ключевого слова End, за которым следует имя блока.

5.3.5.3.6 Пример описания блоков

При описании блоков используются элементы, описанные в библиотеке Elements, приведенной в разд. «Пример описания элементов».

```
NetBibl SubNets Used Elements;          {Библиотека подсетей, использующая библиотеку Elements}

        {Сигмоидный нейрон с произвольным сумматором на N входов}
Cascad NSigm(aSum : Block; N : Long; Char : Real)
        {В состав каскада входит произвольный сумматор на N входов и сигмоидный нейрон
      с необучаемой характеристикой}
  Contents aSum(N), S_NotTrain(Char)
  InSignals NumberOf(InSignals, aSum(N))      {Число входных сигналов определяет сумматор}
  OutSignals 1                                {Один выходной сигнал}
  Parameters NumberOf(Parameters, aSum(N))    {Число параметров определяет сумматор}
  Connections
        {Входные сигналы нейрона – входные сигналы сумматора}
    InSignals[1.. NumberOf(InSignals, aSum(N))] <=>
      aSum.InSignals[1.. NumberOf(InSignals, aSum(N))]
    aSum.OutSignals <=> S_NotTrain.InSignals  {Выход сумматора – вход преобразователя}
```

```
        OutSignals <=> S_NotTrain.OutSignals
                {Параметры нейрона – параметры сумматора }
        Parameters[1.. NumberOf(Parameters, aSum(N))] <=>
            aSum.Parameters[1.. NumberOf(Parameters, aSum(N))]
End                             {Конец описания сигмоидного нейрона с произвольным сумматором}

                {Слой сигмоидных нейронов с произвольными сумматорами на N входов}
Layer Lay1(aSum : Block; N,M : Long; Char : Real)
    Contents Sigm: NSigm(aSum,N,Char)[M]          {В состав слоя входит M нейронов}
    InSignals M * NumberOf(InSignals, Sigm)
                {Число входных сигналов определяется как взятое M раз число
                входных сигналов нейронов. Вместо имени нейрона используем псевдоним}
    OutSignals M                                  {Один выходной сигнал на нейрон}
    Parameters M * NumberOf(Parameters, Sigm)
                {Число параметров определяется как взятое M раз число параметров нейронов}
    Connections
                {Первые NumberOf(InSignals, NSigm(aSum,N,Char)) сигналов первому нейрону, и т.д.}
        InSignals[1..M * NumberOf(InSignals, Sigm)] <=>
            Sigm[1..M].InSignals[1.. NumberOf(InSignals, Sigm)]
                {Выходные сигналы нейронов - выходные сигналы сети}
        OutSignals[1..M] <=> Sigm[1..M].OutSignals
                {Параметры слоя – параметры нейронов}
        Parameters[1..M * NumberOf(Parameters, Sigm)] <=>
            Sigm[1..M].Parameters[1.. NumberOf(Parameters, Sigm)]
End                     {Конец описания слоя сигмоидных нейронов с произвольным сумматором}

                {Слой точек ветвления}
Layer BLay( N,M : Long)
    Contents Branch(N)[M]               {В состав слоя входит M точек ветвления}
    InSignals M                         {По одному входному сигналу на точку ветвления}
    OutSignals M * N                    {N выходных сигналов у каждой точки ветвления}
    Connections
        InSignals[1..M] <=> Branch[1..M].InSignals     {По одному входу на точку ветвления}
                {Выходные сигналы в порядке первый с каждой точки ветвления, затем второй и т.д. }
        OutSignals[1..N * M] <=> Branch[+:1..M].OutSignals[1..N]
End                                     {Конец описания слоя Точек ветвления}

                {Полный слой сигмоидных нейронов с произвольными сумматорами на N входов}
Cascad FullLay(aSum : Block; N,M : Long; Char : Real)
    Contents Br: BLay1(M,N), Ne: Lay1(aSum,N,M,Char)  {Слой точек ветвления и слой нейронов}
    InSignals N                         {Число входных сигналов – число точек ветвления}
    OutSignals M                        {Один выходной сигнал на нейрон}
    Parameters NumberOf(Parameters, Ne)
                {Число параметров определяется как взятое M раз число параметров нейронов}
    Connections
        InSignals[1..N]<=> Br.InSignals[1..N]      {Входные сигналы – слою точек ветвления}
                {Выходные сигналы нейронов - выходные сигналы сети}
        OutSignals[1..M] <=> Ne.OutSignals[1..M]
                {Параметры слоя – параметры нейронов}
        Parameters[1..NumberOf(Parameters, Ne)] <=>
            Ne.Parameters[1.. NumberOf(Parameters, Ne)]
                {Выход слоя точек ветвления – вход слоя нейронов}
        Br.OutSignals[1..N * M] <=> Ne.InSignals[1..N * M]
End                     {Конец описания слоя сигмоидных нейронов с произвольным сумматором}

                {Сеть с сигмоидными нейронами и произвольными сумматорами, содержащая
                    Input – число нейронов на входном слое;
                    Output – число нейронов на выходном слое (число выходных сигналов);
                    Hidden – число нейронов на H>0 скрытых слоях;
                    N – число входных сигналов
                все входные сигналы подаются на все нейроны входного слоя}
```

```
Cascad  Net1(aSum : Block; Char : Real; Input, Output, Hidden, H, N : Long)
                {Под тремя разными псевдонимами используется одна и та же подсеть с разными
                параметрами}
    Contents
        In: FullLay(aSum,N,Input,Char),
        Hid1: FullLay(aSum,Input,Hidden,Char)
        Hid2: FullLay(aSum,Hidden,Hidden,Char)[H-1]              {Пусто при H=1}
        Out: FullLay(aSum,Hidden,Output,Char)
    InSignals N                                  {Число входных сигналов – N}
    OutSignals Output                              {Один выходной сигнал на нейрон}
                {Число параметров определяется как сумма чисел параметров всех подсетей}
    Parameters NumberOf(Parameters, In)+ NumberOf(Parameters, Hid1)+
        (H-1) * NumberOf(Parameters, Hid2)+ NumberOf(Parameters, Out)
    Connections
        InSignals[1..N]<=> In.InSignals[1..N]          {Входные сигналы – входному слою}
                {Выходные сигналы нейронов - с выходного слоя сети}
        OutSignals[1..Output] <=> Out.OutSignals[1.. Output]
                {Параметры сети последовательно всем подсетям}
        Parameters[1..NumberOf(Parameters,In)] <=> In.Parameters[1.. NumberOf(Parameters, In)]
        Parameters[NumberOf(Parameters,In)+1..NumberOf(Parameters,In)+
            NumberOf(Parameters, Hid1)] <=> Hid1.Parameters[1.. NumberOf(Parameters, Hid1)]
        Parameters[NumberOf(Parameters,In)+ NumberOf(Parameters, Hid1)]+1
            ..NumberOf(Parameters,In)+NumberOf(Parameters, Hid1)+
            (H-1) * NumberOf(Parameters, Hid2)] <=>
            Hid2[1..H-1].Parameters[1.. NumberOf(Parameters, Hid2)]
        Parameters[NumberOf(Parameters,In)+ NumberOf(Parameters, Hid1)]+
            (H-1) * NumberOf(Parameters, Hid2)+1..NumberOf(Parameters,In)+
            NumberOf(Parameters,Hid1)+(H-1)*NumberOf(Parameters,Hid2)+
            NumberOf(Parameters, Out)] <=> Out.Parameters[1.. NumberOf(Parameters, Out)]
                {Передача сигналов от слоя к слою}
        In.OutSignals[1..Input] <=> Hid1.InSignals[1..Input]     {От входного к первому скрытому слою}
        Hid1.OutSignals[1..Hidden] <=> Hid2[1].InSignals[1..Hidden]     {От первого скрытого слоя}
    {между скрытыми слоями. При H=1 эта запись пуста}
        Hid2[1..H-2].OutSignals[1.. Hidden] <=> Hid2[2..H-1].InSignals[1.. Hidden]
        Hid2[H-1].OutSignals[1.. Hidden] <=> Out.InSignals[1.. Hidden]{От скрытых – к выходному}
End

                {Полносвязная сеть с M сигмоидными нейронами на K тактов функционирования с
                невыделенным входным слоем на M сигналов}
Loop Circle(aSum : Block; Char : Real; M, K : Long) K
    Contents
        Net: FullLay(aSum,M,M,Char)
    InSignals M                                  {Число входных сигналов – N}
    OutSignals M                                 {Один выходной сигнал на нейрон}
    Parameters NumberOf(Parameters, Net)    {Число параметров определяется слоем FullLay}
    Connections
        InSignals[1..M]<=> Net.InSignals[1..M]          {Входные сигналы – на вход слоя}
        OutSignals[1..M] <=> Net.OutSignals[1.. M]          {Выходные сигналы – на выходе слоя}
                {Все параметры слою}
        Parameters[1..NumberOf(Parameters,Net)] <=> Net.Parameters[1.. NumberOf(Parameters,Net)]
        Net.OutSignals[1..M] <=> Net.InSignals[1..M]      {Замыкаем выход на вход}
End                     {Конец описания слоя сигмоидных нейронов с произвольным сумматором}

                {Полносвязная сеть с M сигмоидными нейронами на K тактов функционирования с
                выделенным входным слоем на N сигналов. Все входные сигналы подаются на вход
                каждого нейрона входного слоя. Все параметры ограничены по абсолютному
                значению единицей}
Cascad  Net2: (aSum : Block; Char : Real; M, K, N : Long)
    Contents
        In: FullLay(aSum,N,M,Char),                  {Входной слой}
```

```
        Net: Circle(aSum,Char,M,K)                    {Полносвязная сеть}
    InSignals N                                       {Число входных сигналов – N}
    OutSignals M                                      {Один выходной сигнал на нейрон}
            {Число параметров определяется как сумма чисел параметров всех подсетей}
    Parameters NumberOf(Parameters, In)+ NumberOf(Parameters, Net)
    ParamDef DefaultType -1 1
    Connections
        InSignals[1..N]<=> In.InSignals[1..N]         {Входные сигналы – входному слою}
            {Выходные сигналы нейронов - с выходного слоя сети}
        OutSignals[1..M] <=> Net.OutSignals[1.. M]
            {Параметры сети последовательно всем подсетям}
        Parameters[1..NumberOf(Parameters, In)] <=> In.Parameters[1.. NumberOf(Parameters, In)]
        Parameters[NumberOf(Parameters,In)+1..
            NumberOf(Parameters,In)+NumberOf(Parameters, Net)]
            <=> Net.Parameters[1.. NumberOf(Parameters, Net)]
            {Передача сигналов от слоя к слою}
        In.OutSignals[1..M] <=> Net.InSignals[1..M]       {От входного к циклу}
        Net.OutSignals[1..M] <=> Net.InSignals[1..M]      {От первого скрытого слоя}
End

            {Нейрон сети Хопфилда из N нейронов}
Cascad Hopf(N : Long)
    Contents Sum(N),Sign_Easy                         {Сумматор и пороговый элемент}
    InSignals N                                       {Число входных сигналов – N}
    OutSignals 1                                      {Число выходных сигналов – 1}
    Parameters NumberOf(Parameters,Sum(N))            {Число параметров – N}
    Connections
        InSignals[1..N]<=> Sum.InSignals[1..N]        {Входные сигналы – сумматору}
            {Выходной сигнал нейрона – выходной сигнал порогового элемета}
        OutSignals <=> Sign_Easy.OutSignals
            {Параметры нейрона – параметры сумматора}
        Parameters[1..NumberOf(Parameters, Sum(N))] <=>
            Sum.Parameters[1.. NumberOf(Parameters, Sum(N))]
        Sum.OutSignals <=> Sign_Easy.InSignals        {Выход сумматора на вход порога}
End

            {Слой нейронов Хопфилда}
Layer HLay(N : Long)
    Contents Hop: Hopf(N)[N]                  {В состав слоя входит N нейронов}
    InSignals N * N                               {N нейронов по N входных сигналов}
    OutSignals N                                  {Один выходной сигнал на нейрон}
    Parameters N * NumberOf(Parameters, Hop)
    Connections
            {Первые NumberOf(InSignals, Hop) сигналов первому нейрону, и т.д.}
        InSignals[1..Sqr(N)] <=> Hop[1..N].InSignals[1..N]
            {Выходные сигналы нейронов - выходные сигналы сети}
        OutSignals[1..N] <=> Hop[1..N].OutSignals
            {Параметры слоя – параметры нейронов}
        Parameters[1..N * NumberOf(Parameters, Hop)] <=>
            Hop[1..N].Parameters[1.. NumberOf(Parameters, Hop)]
End

            {Сеть Хопфилда из N нейронов}
Until Hopfield(N : Long) InSignals=OutSignals
    Contents BLay(N,N),HLay(N)                {Слой точек ветвления и слой нейронов}
    InSignals N                                   {Число входных сигналов – N}
    OutSignals N                                  {Число выходных сигналов – N}
    Parameters N * NumberOf(Parameters,HLay(N))   {Число параметров – N*N}
    Connections
        InSignals[1..N]<=> BLay.InSignals[1..N]   {Входные сигналы – точкам ветвления}
```

```
        {Выходные сигналы нейронов – выходные сигналы сети}
    OutSignals[1..N] <=> HLay.OutSignals[1..N]
    Parameters[1..N*NumberOf(Parameters, HLay(N))] <=>
        HLay.Parameters[1..N*NumberOf(Parameters, HLay(N))]
        {Выход точек ветвления на вход нейронов}
    BLay.OutSignals[1..Sqr(N)] <=> HLay.InSignals[1..Sqr(N)]
    HLay.OutSignals[1..N] <=> BLay.InSignals[1..N]      {Замыкаем конец на начало}
End

End NetLib

NetWork Hop Used SubNets;          {Сеть Хопфилда на пять нейронов}
MainNet Hopfield(5)
Parameters 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;
ParamMask -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1;
End NetWork
```

5.3.5.4 Сокращение описания сети

Предложенный в предыдущих разделах язык описания многословен. В большинстве случаев за счет хорошей структуризации сети можно опустить все разделы описания блока кроме раздела состава. В данном разделе описывается генерация по умолчанию разделов описания сигналов и параметров, и описания связей. Использование механизмов умолчания позволяет сильно сократить текст описания сети.

5.3.5.4.1 Раздел описания сигналов и параметров

Для всех видов блоков число параметров определяется как сумма чисел параметров всех подсетей, перечисленных в разделе описания состава. Это может приводить к лишним записям, но не повлияет на работу сети. Примером лишней записи может служить генерируемая запись:

Parameters M * **NumberOf(Parameters**,Branch(N))

в описании слоя точек ветвления, поскольку точки ветвления не имеют параметров.

Число входных сигналов блока определяется по следующим правилам:

- для слоя число входных сигналов равно сумме числа входных сигналов всех подсетей, перечисленных в разделе описания состава;
- для каскадов всех видов число входных сигналов блока равно числу входных сигналов подсети, стоящей первой в списке подсетей в разделе описания состава

Число выходных сигналов блока определяется по следующим правилам:

- для слоя число выходных сигналов равно сумме числа выходных сигналов всех подсетей, перечисленных в разделе описания состава;
- для каскадов всех видов число выходных сигналов блока равно числу выходных сигналов подсети, стоящей последней в списке подсетей в разделе описания состава;

Описания всех сетей, приведенные в предыдущем разделе полностью соответствуют правилам генерации. В качестве более общего примера приведем раздел описания сигналов и параметров двух условных блоков.

```
Layer  A
  Contents Net1, Net2[K], Net3
  InSignals NumberOf(InSignals,Net1)+K*NumberOf(InSignals,Net2)
     +NumberOf(InSignals,Net3)
  OutSignals NumberOf(OutSignals,Net1)+K*NumberOf(OutSignals,Net2)
     +NumberOf(OutSignals,Net3)
  Parameters NumberOf(Parameters,Net1)+K*NumberOf(Parameters,Net2)
     +NumberOf(Parameters,Net3)

Cascad  B
  Contents Net1, Net2[K], Net3
  InSignals NumberOf(InSignals,Net1)
  OutSignals NumberOf(OutSignals,Net3)
  Parameters NumberOf(Parameters,Net1)+K*NumberOf(Parameters,Net2)
     +NumberOf(Parameters,Net3)
```

5.3.5.4.2 Раздел описания связей

Раздел описания связей может быть разбит на пять подразделов.

1. Установление связи входных сигналов блока с входными сигналами подсетей.
2. Установление связи выходных сигналов блока с выходными сигналами подсетей.
3. Установление связи параметров блока с параметрами подсетей.
4. Установление связи между выходными сигналами одних подсетей и входными сигналами других подсетей.
5. Замыкание выхода блока на вход блока.

Для слоя раздел описания связей строится по следующим правилам.

1. Все подсети получают входные сигналы в порядке перечисления подсетей в разделе описания состава – первая часть массива входных сигналов слоя отдается первой подсети, следующая – второй и т.д. Если какая-либо подсеть в разделе описания состава указана с некоторым не равным единице числом экземпляров, то считается, что экземпляры этой подсети перечислены в списке в порядке возрастания номера.
2. Выходные сигналы подсетей образуют массив выходных сигналов слоя также в порядке перечисления подсетей в разделе описания состава – первая часть массива выходных сигналов слоя состоит из выходных сигналов первой подсети, следующая – второй и т.д. Если какая-либо подсеть в разделе описания состава указана с некоторым не равным единице числом экземпляров, то считается, что экземпляры этой подсети перечислены в списке в порядке возрастания номера.
3. Подразделы установления связи между выходными сигналами одних подсетей и входными сигналами других подсетей и замыкания выхода блока на вход для слоя отсутствуют.

Для каскадов раздел описания связей строится по следующим правилам:

1. Входные сигналы блока связываются с входными сигналами первой подсети в списке подсетей в разделе описания состава. Если для первой подсети указано не единичное число экземпляров, то все входные сигналы связываются с входными сигналами первого экземпляра подсети.
2. Выходные сигналы блока связываются с выходными сигналами последней подсети в списке подсетей в разделе описания состава. Если для последней подсети указано не единичное число экземпляров, то все выходные сигналы связываются с выходными сигналами последнего (с максимальным номером) экземпляра подсети.
3. Массив параметров блока образуется из массивов параметров подсетей в порядке перечисления подсетей в разделе описания состава – первая часть массива параметров блока состоит из параметров первой подсети, следующая – второй и т.д. Если какая-либо подсеть в разделе описания состава указана с некоторым не равным единице числом экземпляров, то считается, что экземпляры этой подсети перечислены в списке в порядке возрастания номера.
4. Выходные сигналы каждой подсети, кроме последней связываются с входными сигналами следующей подсети в списке подсетей в разделе описания состава. Если какая-либо подсеть в разделе описания состава указана с некоторым не равным единице числом экземпляров, то считается, что экземпляры этой подсети перечислены в списке в порядке возрастания номера.
5. Для блоков типа Cascad замыкание выхода блока на вход блока отсутствует. Для блоков типов Loop и Until замыкание выхода блока на вход блока достигается путем установления связей между выходными сигналами последней подсети в списке подсетей в разделе описания состава с входными сигналами первой подсети в списке подсетей в разделе описания состава. Если какая-либо подсеть в разделе описания состава указана с некоторым не равным единице числом экземпляров, то считается, что экземпляры этой подсети перечислены в списке в порядке возрастания номера.

Описания всех сетей, приведенные в предыдущем разделе полностью соответствуют правилам генерации. В качестве более общего примера приведем раздел описания сигналов и параметров трех условных блоков.

```
Layer A
  Contents Net1, Net2[K], Net3
  InSignals[1..NumberOf(InSignals,Net1)+K*NumberOf(InSignals,Net2)
      +NumberOf(InSignals,Net3)] <=> Net1. InSignals[1..NumberOf(InSignals,Net1)],
      Net2[1..K].InSignals[1..NumberOf(InSignals,Net2)],
      Net3.InSignals[1..NumberOf(InSignals,Net3)]
  OutSignals[1..NumberOf(OutSignals,Net1)+K*NumberOf(OutSignals,Net2)
      +NumberOf(OutSignals,Net3)] <=> Net1. OutSignals[1..NumberOf(OutSignals,Net1)],
      Net2[1..K].OutSignals[1..NumberOf(OutSignals,Net2)],
      Net3.OutSignals[1..NumberOf(OutSignals,Net3)]
```

```
  Parameters[1..NumberOf(Parameters,Net1)+K*NumberOf(Parameters,Net2)
      +NumberOf(Parameters,Net3)] <=> Net1. Parameters[1..NumberOf(Parameters,Net1)],
      Net2[1..K].Parameters[1..NumberOf(Parameters,Net2)],
      Net3.Parameters[1..NumberOf(Parameters,Net3)]

Cascad B
  Contents Net1, Net2[K], Net3
  InSignals[1..NumberOf(InSignals,Net1)] <=> Net1. InSignals[1..NumberOf(InSignals,Net1)]
  OutSignals[1..NumberOf(OutSignals,Net3)] <=>
      Net3.OutSignals[1..NumberOf(OutSignals,Net3)]
  Parameters[1..NumberOf(Parameters,Net1)+K*NumberOf(Parameters,Net2)
      +NumberOf(Parameters,Net3)] <=> Net1. Parameters[1..NumberOf(Parameters,Net1)],
      Net2[1..K].Parameters[1..NumberOf(Parameters,Net2)],
      Net[3].Parameters[1..NumberOf(Parameters,Net3)]
  Net1. OutSignals[1..NumberOf(OutSignals,Net1)],
      Net2[1..K].OutSignals[1..NumberOf(OutSignals,Net2)] <=>
      Net2[1..K].InSignals[1..NumberOf(InSignals,Net2)],
      Net3.InSignals[1..NumberOf(InSignals,Net3)]

Loop  C N
  Contents Net1, Net2[K], Net3
  InSignals[1..NumberOf(InSignals,Net1)] <=> Net1. InSignals[1..NumberOf(InSignals,Net1)]
  OutSignals[1..NumberOf(OutSignals,Net3)] <=>
      Net3.OutSignals[1..NumberOf(OutSignals,Net3)]
  Parameters[1..NumberOf(Parameters,Net1)+K*NumberOf(Parameters,Net2)
      +NumberOf(Parameters,Net3)] <=> Net1. Parameters[1..NumberOf(Parameters,Net1)],
      Net2[1..K].Parameters[1..NumberOf(Parameters,Net2)],
      Net[3].Parameters[1..NumberOf(Parameters,Net3)]
  Net1. OutSignals[1..NumberOf(OutSignals,Net1)],
      Net2[1..K].OutSignals[1..NumberOf(OutSignals,Net2)] <=>
      Net2[1..K].InSignals[1..NumberOf(InSignals,Net2)],
      Net3.InSignals[1..NumberOf(InSignals,Net3)]
  Net3.OutSignals[1..NumberOf(OutSignals,Net3)] <=>
      Net1. InSignals[1..NumberOf(InSignals,Net1)]
```

5.3.5.4.3 Частично сокращенное описание

Если описываемый блок должен иметь связи, устанавливаемые не так, как описано в разд. «Раздел описания связей», то соответствующий раздел описания блока может быть описан явно полностью или частично. Если какой либо раздел описан частично, то действует следующее правило: те сигналы, параметры и их связи, которые описаны явно, берутся из явного описания, а те сигналы, параметры и их связи, которые не фигурируют в явном описании берутся из описания по умолчанию. Так, в приведенном в разд. «Пример описания блоков» описании слоя точек ветвления BLay невозможно использование генерируемого по умолчанию подраздела установления связи выходных сигналов блока с входными сигналами подсетей. Возможно следующее сокращенное описание.

```
                {Слой точек ветвления}
Layer BLay( N,M : Long)
   Contents Branch(N)[M]                {В состав слоя входит M точек ветвления}
   Connections
                {Выходные сигналы в порядке первый с каждой точки ветвления, затем второй и т.д. }
      OutSignals[1..N * M] <=> Branch[+:1..M].OutSignals[1..N]
End                                     {Конец описания слоя Точек ветвления}
```

5.3.5.4.4 Пример сокращенного описания блоков

При описании блоков используются элементы, описанные в библиотеке Elements, приведенной в разд. "Пример описания элементов".

```
NetBibl SubNets Used Elements;          {Библиотека подсетей, использующая библиотеку Elements}

                {Сигмоидный нейрон с произвольным сумматором на N входов}
Cascad NSigm(aSum : Block; N : Long; Char : Real)
```

```
            {В состав каскада входит произвольный сумматор на N входов и сигмоидный нейрон
        с необучаемой характеристикой}
    Contents aSum(N), S_NotTrain(Char)
End

            {Слой сигмоидных нейронов с произвольными сумматорами на N входов}
Layer Lay1(aSum : Block; N,M : Long; Char : Real)
    Contents Sigm: NSigm(aSum,N,Char)[M]                    {В состав слоя входит M нейронов}
End

            {Слой точек ветвления}
Layer BLay( N,M : Long)
    Contents Branch(N)[M]                   {В состав слоя входит M точек ветвления}
    Connections
            {Выходные сигналы в порядке первый с каждой точки ветвления, затем второй и т.д. }
        OutSignals[1..N * M] <=> Branch[+:1..M].OutSignals[1..N]
End

            {Полный слой сигмоидных нейронов с произвольными сумматорами на N входов}
Cascad FullLay(aSum : Block; N,M : Long; Char : Real)
    Contents BLay1(M,N), Lay1(aSum,N,M,Char)   {Слой точек ветвления и слой нейронов}
End                     {Конец описания слоя сигмоидных нейронов с произвольным сумматором}

            {Сеть с сигмоидными нейронами и произвольными сумматорами, содержащая
                Input – число нейронов на входном слое;
                Output – число нейронов на выходном слое (число выходных сигналов);
                Hidden – число нейронов на H>0 скрытых слоях;
                N – число входных сигналов
            все входные сигналы подаются на все нейроны входного слоя}
Cascad  Net1(aSum : Block; Char : Real; Input, Output, Hidden, H, N : Long)
            {Под тремя разными псевдонимами используется одна и таже подсеть с разными
            параметрами. Использование псевдонимов необходимо даже при сокращенном
            описании}
    Contents
        In: FullLay(aSum,N,Input,Char),
        Hid1: FullLay(aSum,Input,Hidden,Char)
        Hid2: FullLay(aSum,Hidden,Hidden,Char)[H-1]             {Пусто при H=1}
        Out: FullLay(aSum,Hidden,Output,Char)
End

            {Полносвязная сеть с M сигмоидными нейронами на K тактов функционирования с
            невыделенным входным слоем на M сигналов. Все параметры ограничены по
            абсолютному значению единицей}
Loop Circle(aSum : Block; Char : Real; M, K : Long) K
    Contents
        FullLay(aSum,M,M,Char)
    ParamDef DefaultType -1 1
End

            {Полносвязная сеть с M сигмоидными нейронами на K тактов функционирования с
            выделенным входным слоем на N сигналов. Все входные сигналы подаются на вход
            каждого нейрона входного слоя}
Cascad Net2: (aSum : Block; Char : Real; M, K, N : Long)
    Contents
        In: FullLay(aSum,N,M,Char),                 {Входной слой}
        Net: Circle(aSum,Char,M,K)                  {Полносвязная сеть}
End

Cascad Hopf(N : Long)                               {Нейрон сети Хопфилда из N нейронов}
    Contents Sum(N),Sign_Easy                       {Сумматор и пороговый элемент}
End

            {Слой нейронов Хопфилда}
```

```
Layer HLay(N : Long)
   Contents Hop: Hopf(N)[N]                    {В состав слоя входит N нейронов}
End

                {Сеть Хопфилда из N нейронов}
Until Hopfield(N : Long)  InSignals=OutSignals
   Contents BLay(N,N),HLay(N)                  {Слой точек ветвления и слой нейронов}
End

End NetLib
```

5.4 Стандарт второго уровня компонента сеть

В данном разделе главы рассмотрены все запросы, исполняемые компонентом сеть. Прежде чем приступать к описанию стандарта запросов компонента сеть следует выделить выполняемые им функции. Что должен делать компонент сеть? Очевидно, что прежде всего он должен уметь выполнять такие функции, как функционирование вперед (работа обученной сети) и назад (вычисление вектора поправок или градиента для обучения), модернизацию параметров (обучение сети) и входных сигналов (обучение примера). Кроме того компонент сеть должен уметь читать сеть с диска и записывать ее на диск. Необходимо так же предусмотреть возможность создавать сеть и редактировать ее структуру. Эти две функциональные возможности не связаны напрямую с работой (функционированием и обучением) сети. Таким образом, необходимо выделить сервисную компоненту – редактор сетей. Компонент редактор сетей позволяет создавать и изменять структуру сети, модернизировать обучаемые параметры в «ручном» режиме.

5.4.1 Запросы к компоненте сеть

Запросы к компоненту сеть можно разбить на пять групп:

1. Функционирование.
2. Изменение параметров.
3. Работа со структурой.
4. Инициация редактора и конструктора сетей.
5. Обработка ошибок.

Поскольку компонент сеть может работать одновременно с несколькими сетями, большинство запросов к сети содержат явное указание имени сети. Отметим, что при генерации запросов в качестве имени сети можно указывать имя любой подсети. Таким образом, иерархическая структура сети, описанная в стандарте языка описания сетей, позволяет работать с каждым блоком или элементом сети как с отдельной сетью. Ниже приведено описание всех запросов к компоненту сеть. Каждый запрос является логической функцией, возвращающей значение истина, если запрос выполнен успешно, и ложь – при

Таблица 3.

Значения предопределенных констант

НазваниеВеличинаЗначение		
InSignals	0	Входные сигналы прямого функционирования
OutSignals	1	Выходные сигналы прямого функционирования
Parameters	2	Параметры
InSignalMask	3	Маска обучаемости входных сигналов
ParamMask	4	Маска обучаемости параметров
BackInSignals	5	Входные сигналы обратного функционирования (поправки)
BackOutSignals	6	Выходные сигналы обратного функционирования (поправки)
BackParameters	7	Поправки к параметрам
Element	0	Тип подсети – элемент
Layer	1	Тип подсети – слой
Cascad	2	Тип подсети – простой каскад
CicleFor	3	Тип подсети – цикл с заданным числом проходов
CicleUntil	4	Тип подсети – цикл по условию

ошибочном завершении исполнения запроса.

При вызове ряда запросов используются предопределенные константы. Их значения приведены в табл. 3.

5.4.2 Запросы на функционирование

Два запроса первой группы позволяют проводить прямое и обратное функционирование сети. По сути эти запросы эквивалентны вызову методов Forw и Back сети или ее элемента.

5.4.2.1 Выполнить прямое Функционирование (Forw)

Описание запроса:
Pascal:

```
Function Forw ( Net : PString; InSignals : PRealArray ) : Logic;
```

C:

```
Logic Forw(PString Net, PRealArray InSignals)
```

Описание аргумента:
Net – указатель на строку символов, содержащую имя сети.
InSignals – массив входных сигналов сети.
Назначение – проводит прямое функционирование сети, указанной в параметре Net.
Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если в качестве аргумента Net дан пустой указатель, или указатель на пустую строку, то исполняющим запрос объектом является первая сеть в списке сетей компонента сеть.
3. Если список сетей компонента сеть пуст или имя сети, переданное в аргументе Net в этом списке не найдено, то возникает ошибка 301 – неверное имя сети, управление передается обработчику ошибок, а обработка запроса прекращается.
4. Вызывается метод Forw сети, имя которой было указано в аргументе Net.
5. Если во время выполнения запроса возникает ошибка, то генерируется внутренняя ошибка 304 - ошибка прямого функционирования. Управление передается обработчику ошибок. Выполнение запроса прекращается. В противном случае выполнение запроса успешно завершается.

5.4.2.2 Выполнить обратное Функционирование (Back)

Описание запроса:
Pascal:

```
Function Back( Net : PString; BackOutSignals : PRealArray) : Logic;
```

C:

```
Logic Back(PString Net, PRealArray BackOutSignals)
```

Описание аргумента:
Net – указатель на строку символов, содержащую имя сети.
BackOutSignals – массив производных функции оценки по выходным сигналам сети.
Назначение – проводит обратное функционирование сети, указанной в параметре Net.
Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если в качестве аргумента Net дан пустой указатель, или указатель на пустую строку, то исполняющим запрос объектом является первая сеть в списке сетей компонента сеть.
3. Если список сетей компонента сеть пуст или имя сети, переданное в аргументе Net в этом списке не найдено, то возникает ошибка 301 – неверное имя сети, управление передается обработчику ошибок, а обработка запроса прекращается.
4. Вызывается метод Back сети, имя которой было указано в аргументе Net.
5. Если во время выполнения запроса возникает ошибка, то генерируется внутренняя ошибка 305 - ошибка обратного функционирования. Управление передается обработчику ошибок. Выполнение запроса прекращается. В противном случае выполнение запроса успешно завершается.

5.4.3 Запросы на изменение параметров.

Ко второй группе запросов относятся четыре запроса: Modify – модификация параметров, обычно называемая обучением, ModifyMask – модификация маски обучаемых синапсов, NullGradient – обнуление градиента и RandomDirection – сгенерировать случайное направление спуска.

5.4.3.1 Провести обучение (Modify)

Описание запроса:
Pascal:

Function Modify(Net : PString; OldStep, NewStep : Real; Tipe : Integer; Grad : PRealArray) : Logic;

C:

Logic Modify(PString Net, Real OldStep, Real NewStep, Integer Tipe, PRealArray Grad)

Описание аргументов:

Net – указатель на строку символов, содержащую имя сети.

OldStep, NewStep – параметры обучения.

Tipe – одна из констант InSignals или Parameters.

Grad – адрес массива поправок или пустой указатель.

Назначение – проводит обучение параметров или входных сигналов сети, указанной в параметре Net.

Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если в качестве аргумента Net дан пустой указатель, или указатель на пустую строку, то исполняющим запрос объектом является первая сеть в списке сетей компонента сеть.
3. Если список сетей компонента сеть пуст или имя сети, переданное в аргументе Net в этом списке не найдено, то возникает ошибка 301 – неверное имя сети, управление передается обработчику ошибок, а обработка запроса прекращается.
4. Если аргумент Grad содержит пустой указатель, то поправки берутся из массива Back.Parameters или Back.InputSignals в зависимости от значения аргумента Tipe.
5. В зависимости от значения аргумента Tipe для каждого параметра или входного сигнала P, при условии, что соответствующий ему элемент маски обучаемости, соответствующей аргументу Tipe равен -1 (значение истина) выполняется следующая процедура:

   P1=P*OldStep+DP*NewStep.

   Если для типа, которым описан параметр P, заданы минимальное и максимальное значения, то:

   P2=Pmin, при P1<Pmin

   P2=Pmax, при P1>Pmax

   P2=P1 в противном случае

5.4.3.2 Изменить маску обучаемости (ModifyMask)

Описание запроса:

Pascal:

Function ModifyMask(Net : PString; Tipe : Integer; NewMask: PLogicArray) : Logic;

C:

Logic Modify(PString Net, Integer Tipe, PLogicArray NewMask)

Описание аргументов:

Net – указатель на строку символов, содержащую имя сети.

Tipe – одна из констант InSignals или Parameters.

NewMask – новая маска обучаемости.

Назначение – Заменяет маску обучаемости параметров или входных сигналов сети, указанной в параметре Net.

Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если в качестве аргумента Net дан пустой указатель, или указатель на пустую строку, то исполняющим запрос объектом является первая сеть в списке сетей компонента сеть.
3. Если список сетей компонента сеть пуст или имя сети, переданное в аргументе Net в этом списке не найдено, то возникает ошибка 301 – неверное имя сети, управление передается обработчику ошибок, а обработка запроса прекращается.
4. В зависимости от значения параметра Tipe заменяет маску обучаемости параметров или входных сигналов на переданную в параметре NewMask.

5.4.3.3 Обнулить градиент (NullGradient)

Описание запроса:

Pascal:

Function NullGradient(Net : PString) : Logic;

C:

Logic NullGradient(PString Net)

Описание аргументов:
Net – указатель на строку символов, содержащую имя сети.

Назначение – производит обнуление градиента сети, указанной в параметре Net.
Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если в качестве аргумента Net дан пустой указатель, или указатель на пустую строку, то исполняющим запрос объектом является первая сеть в списке сетей компонента сеть.
3. Если список сетей компонента сеть пуст или имя сети, переданное в аргументе Net в этом списке не найдено, то возникает ошибка 301 – неверное имя сети, управление передается обработчику ошибок, а обработка запроса прекращается.
4. Обнуляются массивы Back.Parameters и Back.OutSignals.

5.4.3.4 Случайное направление спуска (RandomDirection)

Описание запроса:
Pascal:

```
Function RandomDirection( Net : PString; Range : Real ) : Logic;
```

C:

```
Logic RandomDirection(PString Net, Real Range)
```

Описание аргументов:
Net – указатель на строку символов, содержащую имя сети.
Range – относительная ширина интервала, на котором должны быть распределены значения случайной величины.

Назначение – генерирует вектор случайных поправок к параметрам сети.
Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если в качестве аргумента Net дан пустой указатель, или указатель на пустую строку, то исполняющим запрос объектом является первая сеть в списке сетей компонента сеть.
3. Если список сетей компонента сеть пуст или имя сети, переданное в аргументе Net в этом списке не найдено, то возникает ошибка 301 – неверное имя сети, управление передается обработчику ошибок, а обработка запроса прекращается.
4. Замещают все значения массива Back.Parameters на случайные величины. Интервал распределения случайной величины зависит от типа параметра, указанного при описании сети (ParamType) и аргумента Range. Полуширина интервала определяется как произведение полуширины интервала допустимых значений параметра, указанных в разделе ParamDef описания сети на величину Range. Интервал распределения случайной величины определяется как [-Полуширина; Полуширина].

5.4.4 Запросы, работающие со структурой сети.

К третьей группе относятся запросы, позволяющие изменять структуру сети. Часть запросов этой группы описана в разд. "Остальные запросы".

5.4.4.1 Вернуть параметры сети (nwGetData)

Описание запроса:
Pascal:

```
Function nwGetData(Net : PString; DataType : Integer; Var Data : PRealArray) : Logic;
```

C:

```
Logic nwGetData(PString Net, Integer DataType, PRealArray* Data)
```

Описание аргументов:
Net – указатель на строку символов, содержащую имя сети.
DataType – одна из восьми предопределенных констант, описывающих тип данных сети.
Data – возвращаемый массив параметров сети.

Назначение – возвращает параметры, входные или выходные сигналы сети, указанной в аргументе Net.
Описание исполнения.

1. Если в качестве аргумента Net дан пустой указатель, или указатель на пустую строку, то исполняющим запрос объектом является перавя сеть в списке сетей компонента сеть.

2. Если имя сети, переданное в аргументе Net не найдено в списке сетей компонента сеть или этот список пуст, то возникает ошибка 301 – неверное имя сети, управление передается обработчику ошибок, а обработка запроса прекращается.
3. Если значение, переданное в аргументе DataType больше семи или меньше нуля, то возникает ошибка 306 – ошибочный тип параметра сети, управление передается обработчику ошибок, а обработка запроса прекращается.
4. В массиве Data возвращаются указанные в аргументе DataType параметры сети.

5.4.4.2 Установить параметры сети (nwSetData)

Описание запроса:
Pascal:

```
Function nwSetData(Net : PString; DataType : Integer; Var Data : RealArray) : Logic;
```

C:

```
Logic nwSetData(PString Net, Integer DataType, RealArray* Data)
```

Описание аргументов:
Net – указатель на строку символов, содержащую имя сети.
DataType – одна из восьми предопределенных констант, описывающих тип данных сети.
Data – массив параметров для замещения текущего массива параметров сети.

Назначение – замещает параметры, входные или выходные сигналы сети, указанной в аргументе Net на значения из массива Data.
Описание исполнения.

1. Если в качестве аргумента Net дан пустой указатель, или указатель на пустую строку, то исполняющим запрос объектом является первая сеть в списке сетей компонента сеть.
2. Если имя сети, переданное в аргументе Net не найдено в списке сетей компонента сеть или этот список пуст, то возникает ошибка 301 – неверное имя сети, управление передается обработчику ошибок, а обработка запроса прекращается.
3. Если значение, переданное в аргументе DataType больше семи или меньше нуля, то возникает ошибка 306 – ошибочный тип параметра сети, управление передается обработчику ошибок, а обработка запроса прекращается.
4. Значения параметров (входных или выходных сигналов) сети заменяются на значения из массива Data. Если длинны массива Data недостаточно для замены значений всех параметров (входных или выходных сигналов), то замещаются только столько элементов массива параметров (входных или выходных сигналов) сколько элементов в массиве Data. Если длинна массива Data больше длинны массива параметров (входных или выходных сигналов), то заменяются все элементы вектора параметров (входных или выходных сигналов), а лишние элементы массива Data игнорируются.

5.4.4.3 Нормализовать сеть (NormalizeNet)

Описание запроса:
Pascal:

```
Function NormalizeNet(Net : PString) : Logic;
```

C:

```
Logic NormalizeNet(PString Net)
```

Описание аргумента:
Net – указатель на строку символов, содержащую имя сети.

Назначение – нормализация сети, указанной в аргументе Net.
Описание исполнения.

1. Если в качестве аргумента Net дан пустой указатель, или указатель на пустую строку, то исполняющим запрос объектом является первая сеть в списке сетей компонента сеть.
2. Если имя сети, переданное в аргументе Net не найдено в списке сетей компонента сеть или этот список пуст, то возникает ошибка 301 – неверное имя сети, управление передается обработчику ошибок, а обработка запроса прекращается.
3. Из сети удаляются связи, имеющие нулевой вес и исключенные из обучения. Нумерация сигналов и параметров сохраняется.
4. Из структуры сети удаляются «немые» участки – элементы и блоки, выходные сигналы которых не являются выходными сигналами сети в целом и не используются в качестве входных сигналов другими подсетями. Нумерация сигналов и параметров сохраняется.

5. Производится замена элементов, ставших «прозрачными» – путем замыкания входного сигнала на выходной, удаляются простые однородные сумматоры с одним входом и точки ветвления с одним выходом; адаптивные однородные сумматоры с одним входом заменяются синапсами. Нумерация сигналов и параметров сохраняется.
6. В каждом блоке производится замена имен подсетей на псевдонимы.
7. Производится изменение нумерации сигналов и параметров сети.

5.4.5 Остальные запросы

Ниже приведен список запросов, исполнение которых описано в главе "Общий стандарт":

nwSetCurrent – Сделать сеть текущей
nwAdd – Добавление сети
nwDelete – Удаление сети
nwWrite – Запись сети
nwGetStructNames – Вернуть имена подсетей
nwGetType – Вернуть тип подсети
nwEdit – Редактировать компоненту сеть
OnError – Установить обработчик ошибок
GetError – Дать номер ошибки
FreeMemory – Освободить память

В запросе nwGetType в переменной TypeId возвращается значение одной из предопределенных констант, перечисленных в табл. 3.

Следует заметить, что два запроса nwGetData (Получить параметры) и nwSetData (Установить параметры) имеют название, совпадающее с названием запросов, описанных в главе "Общий стандарт", но они имеют другой набор аргументов.

5.4.5.1 Ошибки компонента сеть

В табл. 4 приведен полный список ошибок, которые могут возникать при выполнении запросов компонентом сеть, и действия стандартного обработчика ошибок.

Таблица 4

Ошибки компонента сеть и действия стандартного обработчика ошибок.

№Название ошибкиСтандартная обработка		
301	Неверное имя сетиЗанесение	номера в Error
302	Ошибка считывания сетиЗанесение	номера в Error
303	Ошибка сохранения сетиЗанесение	номера в Error
304	Ошибка прямого функционированияЗанесение	номера в Error
305	Ошибка обратного функционированияЗанесение	номера в Error
306	Ошибочный тип параметра сетиЗанесение	номера в Error