2. Общий стандарт

Эта глава посвящена описанию элементов стандарта, общих для всех компонентов нейрокомпьютера.

2.1 Стандарт типов данных

При описании запросов, структур данных, стандартов компонентов нейрокомпьютера необходимо использовать набор первичных типов данных. Поскольку в разных языках программирования типы данных называются по-разному, введем единый набор обозначений для них.

Таблица 1.

Типы данных для всех компонентов нейрокомпьютера

Тип	Длина	Значения	Описание
Real	4 байта	от 1.5 e- 45 до 3.4 e 38	Действительное число. Величина из указанного диапазона. Знак произвольный. В дальнейшем называется «действительное».
Integer	2 байта	от -32768 до 32767	Целое число из указанного диапазона. В дальнейшем называется «целое».
Long	4 байта	от -2147483648 до 2147483647	Целое число из указанного диапазона. В дальнейшем называется «длинное целое».
RealArray	4*N байт	Массив	действительных чисел.
PRealArray	4 байта	Используется	для передачи массивов между компонентами. Имеет значение адреса массива действительных чисел.
IntegerArray	2*N байт	Массив	целых чисел.
PIntegerArray	4 байта	Используется	для передачи массивов между компонентами. Имеет значение адреса массива целых чисел.
LongArray	4*N байт	Массив	длинных целых чисел.
PLongArray	4 байта	Используется	для передачи массивов между компонентами. Имеет значение адреса массива длинных целых чисел.
Logic	1 байт	True, False	Логическая величина. Далее называется «логическая».
LogicArray	N байт	Массив	логических переменных.
PLogicArray	4 байта	Используется	для передачи массивов между компонентами. Имеет значение адреса массива логических переменных.
Color	2 байта	Используется	для задания цветов. Является совокупностью из 16 элементарных (битовых) флагов. См. раздел «Цвет и операции с цветами».
FuncType	4 байта	Адрес	функции. Используется при необходимости передать функцию в качестве аргумента.
String	256 байт	Строка	символов.
PString	4 байта	Адрес	строки символов. Служит для передачи строк в запросах
Visual	4 байта	Отображаемый	элемент. Служит для адресации отображаемых элементов в интерфейсных функциях. Тип значений зависит от реализации библиотеки интерфейсных функций и не может изменяться пользователем иначе, чем через вызов интерфейсной функции.
Pointer	4 байта	Не	типизованный указатель (адрес). Этот тип совместим с любым типизованным указателям.

Числовые типы данных Integer, Long и Real предназначены для хранения различных чисел. Переменные числовых типов допускаются в языках описания всех компонентов нейрокомпьютера. При необходимости записать в один массив числовые переменные различного типа следует использовать функции приведения типов, описанные в разделе «Приведение типов»

Строка. Символьный тип данных предназначен для хранения комментариев, названий полей, имен сетей, оценок и другой текстовой информации. Все строковые переменные занимают 256 байт и могут включать в себя до 255 символов. Первый байт строки содержит длину строки. В переменных типа

строка возможен доступ к любому символу как к элементу массива. При этом длина имеет индекс ноль, первый символ – 1 и т.д.

Указатель на строку. При передаче данных между компонентами сети и процедурами в пределах одного компонента удобно вместо строки передавать указатель на строку, поскольку указатель занимает всего четыре байта. Для этой цели служит тип указатель на строку.

Логический тип используется для хранения логических значений. Значение истина задается предопределенной константой True, значение ложь – False.

Массивы. В данном стандарте предусмотрены массивы четырех типов – логических, целочисленных, длинных целых и действительных переменных. Длины массивов определяются при описании, но все массивы переменных одного типа относятся к одному типу, в отличие от языков типа Паскаль. Использование функций приведения и преобразования типов позволяют получать из этих массивов структуры произвольной сложности. Элементы массивов всегда нумеруются с единицы.

Вне зависимости от типа массива нулевой элемент массива имеет тип Long и содержит длину массива в элементах. На рис. 1 приведена схема распределения памяти всех типов массивов, каждый из которых содержит шесть элементов.

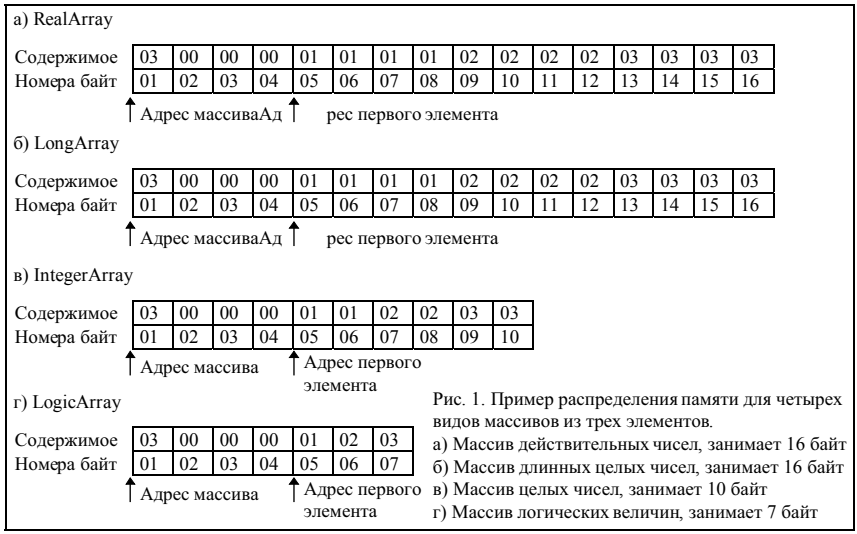

Рис. 1. Пример распределения памяти для четырех видов массивов из трех элементов.
а) Массив действительных чисел, занимает 16 байт
б) Массив длинных целых чисел, занимает 16 байт
в) Массив целых чисел, занимает 10 байт
г) Массив логических величин, занимает 7 байт

Все массивы, как правило, используется только в пределах одного компонента. При передаче массивов между компонентами или между процедурами в пределах одного компонента используются указатели на массивы.

Адрес функции. Этот тип используется для передачи функции в качестве аргумента. Переменная типа FuncType занимает четыре байта и является адресом функции. В зависимости от реализации по этому адресу может лежать либо начало машинного кода функции, либо начало текста функции. В случае передачи текста функции первые восемь байт по переданному адресу содержат слово «Function».

Отображаемый элемент. Переменные типа Visual (отображаемый элемент) Служат для адресации отображаемых элементов в интерфейсных функциях. Тип значений зависит от реализации библиотеки интерфейсных функций и не может изменяться пользователем иначе, чем через вызов интерфейсной функции. Особо следует отметить, что библиотека интерфейсных функций не является частью ни одного из компонентов.

2.2 Переменные типа цвет и операции с цветами

Использование цветов позволяет гибко разбивать множества на подмножества. В нейрокомпьютере возникает необходимость в разбиении на подмножества (раскрашивании) задачника. В этом разделе описывается стандарт работы с переменными типа цвет.

2.2.1 Значение переменной типа цвет (Color)

Переменная типа цвет представляет собой двухбайтовое беззнаковое целое. Однако основное использование предполагает работу не как с целым числом, а как с совокупностью однобитных флагов. При записи на диск используется символьное представление двоичной записи числа с ведущими нулями и разбиением на четверки символом «.» (точка), предваряемая заглавной буквой «B» латинского алфавита, или символьное представление шестнадцатеричной записи числа с ведущими нулями, предваряемая заглавной буквой «H» латинского алфавита. В таблице 2 приведена нумерация флагов (бит) переменной типа Color, их шестнадцатеричное, десятичное и двоичное значение. При использовании в учителе или других компонентах может возникнуть необходимость в присвоении некоторым из флагов или их комбинаций имен. На такое именование не накладывается никаких ограничений, хотя возможно будет выработан стандарт и на названия часто используемых цветов (масок, совокупностей флагов).

Таблица 2

Нумерация флагов (бит) переменной типа Color

Номер	Шестнадцатиричная запись	Десятичная запись	Двоичная запись
0	H0001	1	B.0000.0000.0000.0001
1	H0002	2	B.0000.0000.0000.0010
2	H0004	4	B.0000.0000.0000.0100
3	H0008	8	B.0000.0000.0000.1000
4	H0010	16	B.0000.0000.0001.0000
5	H0020	32	B.0000.0000.0010.0000
6	H0040	64	B.0000.0000.0100.0000
7	H0080	128	B.0000.0000.1000.0000
8	H0100	256	B.0000.0001.0000.0000
9	H0200	512	B.0000.0010.0000.0000
10	H0400	1024	B.0000.0100.0000.0000
11	H0800	2048	B.0000.1000.0000.0000
12	H1000	4096	B.0001.0000.0000.0000
13	H2000	8192	B.0010.0000.0000.0000
14	H4000	16384	B.0100.0000.0000.0000
15	H8000	32768	B.1000.0000.0000.0000

2.2.2 Операции с переменными типа цвет (Color)

В табл. 3 приведены операции с переменными типа Color. Первые пять операций могут использоваться только для сравнения переменных типа Color, а остальные четыре операции – для вычисления выражений типа Color.

Таблица 3

Предопределенные константы операций с переменными типа Цвет (Color)

Код	Обозначение	Вычисляемое выражение	Тип результата	Пояснение
1	CEqual	A = B	Logic	Полное совпадение.
2	CIn	A And B = A	Logic	A содержится в B.
3	CInclude	A And B = B	Logic	A содержит B.
4	CExclude	A And B = 0	Logic	A и B взаимоисключающие.
5	CIntersect	A And B <> 0	Logic	A и B пересекаются.
6	COr	A Or B	Color	Побитное включающее или.
7	CAnd	A And B	Color	Побитное и.
8	CXor	A Xor B	Color	Побитное исключающее или
9	CNot	Not A	Color	Побитное отрицание

В ряде запросов необходимо указать тип операции над цветом. Для передачи таких параметров используется переменная типа Integer. В качестве значений передается содержимое соответствующей ячейки столбца код табл. 3.

2.3 Приведение и преобразование типов

Есть два пути использовать переменную одного типа как переменную другого типа. Первый путь состоит в преобразовании значения к заданному типу. Так, для преобразования целочисленной переменной к действительному типу, достаточно просто присвоить переменной действительного типа целочисленное значение. С обратным преобразованием сложнее, поскольку не ясно что делать с дробной частью. В табл. 4 приведены все типы, которые можно преобразовать присваиванием переменной другого типа. В табл. 5 приведены все функции преобразования типов.

Таблица 4

Преобразование типов прямым присваиванием

Тип переменной, которой производится присваивание	Тип выражения, которое может быть присвоено	Пояснение
Real	Real, Integer, Long	Значение преобразуется к плавающему виду. При преобразовании значения выражения типа Long возможна потеря точности.
Long	Integer, Long	При преобразовании типа Integer, действуют следующие правила. Значение переменной помещается в два младших байта. Если значение выражения больше либо равно нолю, то старшие байты равны H0000, в противном случае старшие байты равны HFFFF.
Integer	Integer, Long	При преобразовании выражения типа Long значение двух старших байт отбрасывается.

При вычислении числовых выражений действуют следующие правила преобразования типов:

1. Выражения вычисляются слева на право.
2. Если два операнда имеют один тип, то результат имеет тот же тип.
3. Если аргументы имеют разные типы, то выражение имеет старший из двух типов. Список числовых типов по убыванию старшинства: Real, Long, Integer.
4. Результат операции деления действительных чисел (операция «/») всегда имеет тип Real, вне зависимости от типов аргументов.

В отличие от преобразования типов приведение типов позволяет по-разному интерпретировать одну область памяти. Функция приведения типа применима только к переменным или элементам массива (преобразование типов применимо и к выражениям). Рекомендуется использовать приведение типов только для типов, имеющих одинаковую длину. Например, Integer и Color или Real и Long. Список функций приведения типов приведен в табл. 6.

Таблица 5

Функции преобразования типов

Имя функции	Тип аргумента	Тип результата	Описание
Real	Real, Integer, Long	Real	Аналогично прямому присваиванию
Integer	Integer, Long	Integer	Аналогично прямому присваиванию
Long	Integer, Long	Long	Аналогично прямому присваиванию
Str	Real, Integer, Long	String	Представляет числовой аргумент в виде символьной строки в десятичном виде
Round	Real	Long	Округляет действительное значение до ближайшего длинного целого. Если значение действительного выражения выходит за диапазон длинного целого, то результат равен нулю.
Truncate	Real	Long	Преобразует действительное значение в длинное целое путем отбрасывания дробной части. Если значение действительного выражения выходит за диапазон длинного целого, то результат равен нулю.
LVal	String	Long	Преобразует длинное целое из символьного представления во внутреннее.
RVal	String	Real	Преобразует действительное число из символьного представления во внутреннее.
StrColor	Color	String	Преобразует внутреннее представление переменной типа Color в соответствии с разд. «Значение переменной типа цвет»
ValColor	String	Color	Преобразует символьное представление переменной типа Color во внутреннее.
Color	Integer	Color	Интерпретирует целое число как значение типа Color.

Следующие примеры иллюстрируют использование преобразования и приведения типов:

При вычислении следующих четырех выражений, получаются различные результаты

4096*4096=0

Таблица 6

Функции приведения типов

НазваниеТип	результата	Описание
TReal	Real	Четыре байта, адресуемые приводимой переменной, интерпретируются как действительное число.
TInteger	Integer	Два байта, адресуемые приводимой переменной, интерпретируются как целое число.
TLong	Long	Четыре байта, адресуемые приводимой переменной, интерпретируются как длинное целое.
TRealArray	RealArray	Область памяти, адресуемая приводимой переменной, интерпретируются как массив действительных чисел.
TPRealArray	PRealArray	Четыре байта, адресуемые приводимой переменной, интерпретируются как указатель на массив действительных чисел.
TIntegerArray	IntegerArray	Область памяти, адресуемая приводимой переменной, интерпретируются как массив целых чисел.
TPIntegerArray	PIntegerArray	Четыре байта, адресуемые приводимой переменной, интерпретируются как указатель на массив целых чисел.
TLongArray	LongArray	Область памяти, адресуемая приводимой переменной, интерпретируются как массив длинных целых.
TPLongArray	PLongArray	Четыре байта, адресуемые приводимой переменной, интерпретируются как указатель на массив длинных целых.
TLogic	Logic	Адресуемый приводимой переменной байт интерпретируются как логическая переменная.
TLogicArray	LogicArray	Область памяти, адресуемая приводимой переменной, интерпретируются как массив логических переменных.
TPLogicArray	LogicArray	Четыре байта, адресуемые приводимой переменной, интерпретируются как указатель на массив логических переменных.
TColor	Color	Два байта, адресуемые приводимой переменной, интерпретируются как переменная типа цвет.
TFuncType	FuncType	Четыре байта, адресуемые приводимой переменной, интерпретируются как адрес функции.
TString	String	256 бай области памяти, адресуемой приводимой переменной, интерпретируются как строка символов.
TPString	PString	Четыре байта, адресуемые приводимой переменной, интерпретируются как указатель на строку символов.
TVisual	Visual	Четыре байта, адресуемые приводимой переменной, интерпретируются как отображаемый элемент.
TPointer	Pointer	Четыре байта, адресуемые приводимой переменной, интерпретируются как адрес.

Поскольку константа 4096 имеет тип Integer, а 4096*4096=16777216=256*65536 , то есть младшие два байта результата равны нулю.

Long(4096*4096)=0

Поскольку оба сомножителя имеет тип Integer, то и выражение имеет тип Integer. Следовательно, результат умножения равен нулю, который затем преобразуется к типу Long.

Long(4096)*4096=16777216

Поскольку первый сомножитель имеет тип длинное целое, то и выражение имеет тип длинное целое.

4096.0*4096=1.677722E+7

Поскольку первый сомножитель имеет тип Real, то и выражение имеет тип Real. Из-за недостатка точности произошла потеря точности в седьмом знаке.

В следующем примере, используя приведение типов, в массив действительных чисел A размером в 66 элементов складываются: действительное число в первый элемент массива; длинное целое во второй элемент массива и символьную строку в элементы с 3 по 66.

A[1]= 1.677722E+7

TLong(A[2])= 16777216

TString(A[3])=‘Пример приведения типов’

Необходимо отметить, что элементы массива A, начиная со второго, после выполнения приведенного выше фрагмента программы не рекомендуется использовать как действительные числа, поскольку элемент A[2] содержит значение 2.350988E-38, а элемент A[5] – значение -4.577438E-18. Значение элементов, начиная с A[8] (символьная строка 'Пример приведения типов' содержит 23 символа и занимает 24 байта, то есть шесть элементов массива) вообще не зависят от приведенного фрагмента программы и содержат «мусор», который там находился ранее.

В списке типов определены только одномерные массивы. Однако, при необходимости, возможно использование двумерных массивов. Для этого в одномерный массив A необходимо поместить указатели на одномерные массивы. При этом I,J-й элемент двумерного массива записывается в виде:

TPRealArray(A[I])^[J]

В этом примере использована функция приведения типов TPRealArray, указывающая, что I-й элемент массива A нужно интерпретировать как указатель на одномерный массив действительных чисел, и операция «^» указывающая, что вместо указателя на массив TPRealArray(A[I]) используется массив, на который он указывает.

Таким образом, использование функций приведения типов позволяет из одномерных массивов строить структуры произвольной сложности. В языках программирования, таких как C и Паскаль, существует возможность строить пользовательские типы данных. При разработке стандарта эти возможности были исключены, поскольку использование пользовательских типов, облегчая написание программ, сильно затрудняет разработку компилятора или интерпретатора, а при использовании этого языка для описания компонентов нейрокомпьютера необходимость в пользовательских типах данных возникает чрезвычайно редко. Например, при описании примеров всех компонентов, приведенных в данной книге, такая необходимость ни разу не возникла.

2.4 Операции

В данном разделе приведены все операции, которые могут быть использованы при построении выражений различного типа. В табл. 7 приведены операции, которые допустимы в целочисленных выражениях (выражениях типа Integer или Long). В табл. 8 – список, дополняющий список операций из табл. 7 до полного списка операций, допустимых в выражениях действительного типа. В табл. 9 – операции, допустимые при построении логических выражений. В табл. 10 –для выражений типа символьная строка. В табл. 3 – для выражений типа Color.

Таблица 7

Операции, допустимые в целочисленных выражениях

Уровень приоритета	Обозначение	Тип 1-го операнда	Тип 2-го операнда	Тип результата	Название операции
1	*	Integer	Integer	Integer	Умножение
1	*	Long	Integer	Long	Умножение
1	*	Integer	Long	Long	Умножение
1	*	Long	Long	Long	Умножение
1	Div	Integer	Integer	Integer	Целочисленное деление
1	Div	Integer	Long	Long	Целочисленное деление
1	Div	Long	Integer	Long	Целочисленное деление
1	Div	Long	Long	Long	Целочисленное деление
1	Mod	Integer	Integer	Integer	Остаток от деления
1	Mod	Long	Integer	Long	Остаток от деления
1	Mod	Integer	Long	Long	Остаток от деления
1	Mod	Long	Long	Long	Остаток от деления
2	+	Integer	Integer	Integer	Сложение
2	+	Integer	Long	Long	Сложение
2	+	Long	Integer	Long	Сложение
2	+	Long	Long	Long	Сложение
2	–	Integer	Integer	Integer	Вычитание
2	–	Integer	Long	Long	Вычитание
2	–	Long	Integer	Long	Вычитание
2	–	Long	Long	Long	Вычитание
3	And	Integer	Integer	Integer	Побитное И
3	And	Long	Long	Long	Побитное И
3	Or	Integer	Integer	Integer	Побитное включающее ИЛИ

Таблица 7

Операции, допустимые в целочисленных выражениях (продолжение)

Уровень приоритета	Обозначение	Тип 1-го операнда	Тип 2-го операнда	Тип результата	Название операции
3	Or	Long	Long	Long	Побитное включающее ИЛИ
3	Xor	Integer	Integer	Integer	Побитное исключающее ИЛИ
3	Xor	Long	Long	Long	Побитное исключающее ИЛИ
3	Not	Integer	Integer	Integer	Побитное отрицание
3	Not	Long	Long	Long	Побитное отрицание

Таблица 8

Операции, дополняющие список операций из табл. 7 до полного списка операций, допустимых в выражениях действительного типа.

Уровень приоритета	Обозначение	Тип 1-го операнда	Тип 2-го операнда	Тип результата	Название операции
1	*	Real	Integer, Real, Long	Real	Умножение
1	/	Integer, Real, Long	Integer, Real, Long	Real	Деление
1	RMod	Integer, Real, Long	Integer, Real, Long	Real	Остаток от деления
2	+	Real	Integer, Real, Long	Real	Сложение
2	–	Real	Integer, Real, Long	Real	Вычитание

Таблица 9

Операции, допустимые при построении логических выражений

Уровень приорит.	Обозначение	Тип 1-го операнда	Тип 2-го операнда	Тип результата	Название операции
1	>	Integer, Real, Long	Integer, Real, Long	Logic	Больше
1	<	Integer, Real, Long	Integer, Real, Long	Logic	Меньше
1	>=	Integer, Real, Long	Integer, Real, Long	Logic	Больше или равно
1	<=	Integer, Real, Long	Integer, Real, Long	Logic	Меньше или равно
1	=	Integer, Real, Long	Integer, Real, Long	Logic	Равно
1	<>	Integer, Real, Long	Integer, Real, Long	Logic	Не равно
2	And	Logic	Logic	Logic	Логическое И
2	Or	Logic	Logic	Logic	Логическое включающее ИЛИ
2	Xor	Logic	Logic	Logic	Логическое исключающее ИЛИ
2	Not	Logic	Logic	Logic	Логическое отрицание

Таблица 10

Операции для выражений типа символьная строка

Уровень приоритета	Обозначение	Тип 1-го операнда	Тип 2-го операнда	Тип результата	Название операции
1	+	String	String	String	Конкатенация (сцепка) строк.

Во всех таблицах операции размещаются по убыванию приоритета. Для каждой операции указаны допустимые типы операндов, и тип результата, в зависимости от типов операндов.

В табл. 8 приводится необычная операция RMod – остаток от деления действительных чисел. Результат этой функции равен разности между первым операндом и вторым операндом, умноженным на целую часть отношения первого операнда ко второму.

Кроме операций, приведенных в табл. 3 и табл. 7–10, определены две взаимно обратные операции для работы с адресами и указателями:

^ – ставится после переменной типа указатель. Означает, что вместо указателя в выражении используется переменная или массив, на который указывает этот указатель. Не допускается после переменных типа Pointer.

@ – ставится перед именем переменной любого типа. Означает, что в выражении участвует не переменная, а адрес переменной. Используется при присвоении адресов переменных или массивов переменным типа указатель.

2.5 Предопределенные константы

При описании различных компонентов возникает необходимость в использовании некоторого набора стандартизированных констант. Стандартность набора констант особенно необходима при обмене между компонентами. Все константы, приведенные в табл. 11, описываются в тех разделах, где они используются. В табл. 11 для каждой константы указывается ее тип, значение и названия разделов, в которых она описывается.

Таблица 11

Предопределенные константы

Идентификатор	Тип	Значение		Раздел
		Шестнад.	Десятич.	
BackInSignals	Integer	H0005	5	Запросы к компоненту сеть
BackOutSignals	Integer	H0006	6	Запросы к компоненту сеть
BackParameters	Integer	H0007	7	Запросы к компоненту сеть
Binary	Integer	H0001	1	Запросы компонента интерпретатор ответа
BinaryPrep	Integer	H0000	0	Запросы компонента предобработчик
BynaryCoded	Integer	H0003	3	Запросы компонента интерпретатор ответа
CAnd	Integer	H0007	7	Операции с переменными типа цвет (Color)
Cascad	Integer	H0002	2	Запросы к компоненту сеть
CEqual	Integer	H0001	1	Операции с переменными типа цвет (Color)
CExclude	Integer	H0004	4	Операции с переменными типа цвет (Color)
CicleFor	Integer	H0003	3	Запросы к компоненту сеть
CicleUntil	Integer	H0004	4	Запросы к компоненту сеть
CIn	Integer	H0002	2	Операции с переменными типа цвет (Color)
CInclude	Integer	H0003	3	Операции с переменными типа цвет (Color)
CIntersect	Integer	H0005	5	Операции с переменными типа цвет (Color)
CNot	Integer	H0009	9	Операции с переменными типа цвет (Color)
COr	Integer	H0006	6	Операции с переменными типа цвет (Color)
CXor	Integer	H0008	8	Операции с переменными типа цвет (Color)
Element	Integer	H0000	0	Запросы к компоненту сеть
Empty	Integer	H0000	0	Запросы компонента интерпретатор ответа
EmptyPrep	Integer	H0003	3	Запросы компонента предобработчик
False	Logic	H00		
FuncPrep	Integer	H0005	5	Запросы компонента предобработчик
InSignalMask	Integer	H0003	3	Запросы к компоненту сеть
InSignals	Integer	H0000	0	Запросы к компоненту сеть
Layer	Integer	H0001	1	Запросы к компоненту сеть
MainVisual	Visible			Интерфейсные функции
Major	Integer	H0002	2	Запросы компонента интерпретатор ответа
mIntegerArray	Integer	H0002	2	Функции управления памятью
mLogicArray	Integer	H0001	1	Функции управления памятью
mLongArray	Integer	H0004	4	Функции управления памятью
ModPrep	Integer	H0004	4	Запросы компонента предобработчик
mRealArray	Integer	H0004	4	Функции управления памятью
Null	Pointer	H00000000	нет	
Ordered	Integer	H0002	2	Запросы компонента предобработчик
OutSignals	Integer	H0001	1	Запросы к компоненту сеть
Parameters	Integer	H0002	2	Запросы к компоненту сеть
ParamMask	Integer	H0004	4	Запросы к компоненту сеть
PositPrep	Integer	H0006	6	Запросы компонента предобработчик
tbAnswers	Integer	H0004	4	Язык описания задачника
tbCalcAnswers	Integer	H0006	6	Язык описания задачника
tbCalcReliability	Integer	H0007	7	Язык описания задачника
tbColor	Integer	H0001	1	Язык описания задачника
tbComment	Integer	H000A	10	Язык описания задачника
tbEstimation	Integer	H0009	9	Язык описания задачника

Таблица 11

Предопределенные константы (продолжение)

Идентификатор	Тип	Значение	Раздел	
		Шестнад.	Десятич.	
tbInput	Integer	H0002	2	Язык описания задачника
tbPrepared	Integer	H0003	3	Язык описания задачника
tbReliability	Integer	H0005	5	Язык описания задачника
tbWeight	Integer	H0008	8	Язык описания задачника
True	Logic	HFF	255 (-1)	
UnknownLong	Integer	H0000	0	Неопределенные значения
UnknownReal	Real	нет	$1E^{-40}$	Неопределенные значения
UnOrdered	Integer	H0001	1	Запросы компонента предобработчик
UserType	Integer	HFFFF	-1	Структурная единица, определенная пользователем.

Три предопределенные константы, приведенные в табл.11, не описываются ни в одном разделе данной книги. Это константы общего пользования. Их значение:

True – значение истина для присваивания переменным логического типа.

False – значение ложь для присваивания переменным логического типа.

Null – пустой указатель. Используется для сравнения или присваивания переменных всех типов указателей.

2.6 Интерфейсные функции

Часто при обучении и тестировании нейронных сетей возникает необходимость отображать на экран некоторую информацию. Например, число предъявлений примеров сети, максимальную оценку и т.п. Вряд ли разумно использовать язык описания компонентов нейрокомпьютера для описания интерфейса. Это противоречит требованию переносимости текста описания компонентов между разными платформами и операционными системами. Для облегчения создания интерфейса, с одной стороны, и обеспечения универсальности описания компонентов нейрокомпьютера, с другой, предложен набор интерфейсных функций. Поддержку этих функций несложно организовать в любой операционной среде и на любой платформе.

В основу набора интерфейсных функций положен принцип объектно-ориентированной машины потока событий. Примером таких систем может служить инструментальная библиотека объектов (классов) Turbo Vision фирмы Борланд, или оконный интерфейс Windows фирмы Майкрософт. Предлагаемый в данном разделе набор интерфейсных функций беднее любой из выше названных интерфейсных библиотек, но позволяет организовать достаточно красивый и удобный интерфейс.

2.6.1 Структура данных интерфейсных функций

Элементом данных в структуре интерфейса является отображаемый элемент. Каждый отображаемый элемент имеет свои координаты относительно начала владельца – отображаемого элемента, содержащего данный. Владельцем отображаемого элемента может являться одно из окон или диалогов или главный элемент. Главный отображаемый элемент является предопределенным и обозначается переменной MainVisual. В программах компонентов нейрокомпьютера не допускается изменение значения переменной MainVisual. Эту переменную стоит рассматривать как константу, однако, в отличие от всех остальных констант ее значение определяется не данным стандартом, а разработчиком библиотеки интерфейсных функций.

В данной работе не рассматривается способ реализации интерфейсных функций и не обсуждаются значения переменных типа Visible. По этому ставится единственное условие – переменные типа Visible могут изменять свои значения только при вызове интерфейсных функций или при присваивании им значений другой переменной того же типа.

Все отображаемые элементы создаются интерфейсными функциями, называющимися, так же как и сам отображаемый элемент. Интерфейсные функции создающие отображаемые элементы возвращают значения типа Visible. Если при вызове создающей отображаемый элемент функции элемент не был создан, то функция возвращает значение Null.

2.6.2 Соглашение о передаче значений отображаемым элементам

Отображаемые элементы могут быть связаны с обычными переменными. В следующих разделах описаны отображаемые элементы, для которых должна быть установлена такая связь. При изменении значения связанной переменной программным путем для отображения этого изменения должна быть вызвана функция Refresh, с переменной типа Visible данного элемента в качестве параметра. При связы-

вании переменной и отображаемого элемента необходимо совпадение типа переменной с связываемым переменной этого отображаемого элемента.

2.6.3 Перечень отображаемых элементов

Название элемента: Window (окно).

Параметры при создании:

BeginX, BeginY – Координаты верхнего левого угла окна относительно владельца.

SizeX, SizeY – Горизонтальный и вертикальный размеры окна.

ScrollX, ScrollY – Целочисленные параметры, задающие наличие у окна горизонтальной и вертикальной полосы прокрутки. Если значение параметра равно нулю, то соответствующая полоса прокрутки отсутствует, при любом другом значении параметра в окно включается соответствующая полоса прокрутки.

Text – Название окна.

Описание элемента. Элемент Window может являться владельцем любых других отображаемых элементов, кроме элементов типа Dialog. При вызове функции Refresh, с элементом типа Window в качестве параметра, обновляется изображение не только самого окна, но и всех отображаемых элементов, для которых это окно является владельцем. Для отображения созданного окна на экране необходимо вставить его (вызвать функцию Insert, с данным окном в качестве второго параметра.) в отображенное на экран окно, диалог или в отображаемый элемент MainVisible. Для того чтобы убрать окно с экрана, необходимо вызвать функцию Delete, с данным окном в качестве параметра. Для уничтожения окна необходимо вызвать функцию Erase, с данным окном в качестве параметра. При этом уничтожаются так же и все отображаемые элементы, для которых данное окно являлось владельцем.

Название элемента: Dialog (Диалог).

Параметры при создании:

BeginX, BeginY – Координаты верхнего левого угла окна диалога относительно владельца.

SizeX, SizeY – Горизонтальный и вертикальный размеры окна диалога.

ScrollX, ScrollY – Целочисленные параметры, задающие наличие у окна диалога горизонтальной и вертикальной полосы прокрутки. Если значение параметра равно нулю, то соответствующая полоса прокрутки отсутствует, при любом другом значении параметра в окно диалога включается соответствующая полоса прокрутки.

Text – Название окна диалога.

Описание элемента. Элемент Dialog может являться владельцем любых других отображаемых элементов. Этот элемент является модальным, то есть во время работы диалога невозможен вызов меню, переход в другие окна и диалоги и т.д. Если одновременно активно несколько диалогов, то модальным является последний по порядку отображения на экране. При закрытии текущего диалога модальность переходит к предыдущему и т.д. При вызове функции Refresh, с элементом типа Dialog в качестве параметра, обновляется изображение не только самого окна диалога, но и всех отображаемых элементов, для которых этот диалог является владельцем. Для отображения созданного диалога на экране необходимо вставить его (вызвать функцию Insert, с данным диалогом в качестве второго параметра.) в отображаемый элемент MainVisible. С этого момента диалог становится модальным. Он остается модальным либо до вставки в MainVisible другого диалога, либо до закрытия диалога. Если модальность потеряна диалогом из-за запуска следующего диалога, то при закрытии последнего диалога статус модальности восстанавливается. Для того чтобы закрыть диалог, необходимо вызвать функцию Delete, с данным диалогом в качестве параметра. Для уничтожения диалога необходимо вызвать функцию Erase, с данным диалогом в качестве параметра. При этом уничтожаются так же и все отображаемые элементы, для которых данный диалог являлся владельцем.

Название элемента: Label (метка).

Параметры при создании:

BeginX, BeginY – Координаты верхнего левого угла метки относительно владельца.

SizeX, SizeY – Горизонтальный и вертикальный размеры метки.

Text – текст метки.

Описание элемента. Элемент Label не может являться владельцем других отображаемых элементов. Этот элемент не связан с переменными. Как правило, он используется для организации отображения в окне или диалоге статической информации или в качестве подписи поля ввода. Если метка связана с полем ввода, то передача управления этой метке автоматически влечет за собой передачу управления связанному с ней полю. Для организации связи метки с полем необходимо вызвать функцию Link, передав ей в качестве первого параметра метку, а вторым параметром тот элемент, с которым необходи-

мо установить связь. Связь может быть установлена только с одним элементом. При повторном вызове функции Link устанавливается связь с новым элементом, а связь с прежним элементом разрывается. Для включения метки в окно или диалог, необходимо вызвать функцию Insert, с окном или диалогом в качестве первого параметра и меткой в качестве второго параметра. Для уничтожения метки необходимо вызвать функцию Erase, с данной меткой в качестве параметра.

Название элемента: StringVisible (строковый элемент).

Параметры при создании:

BeginX, BeginY – Координаты верхнего левого угла элемента относительно владельца.

SizeX, SizeY – Горизонтальный и вертикальный размеры элемента.

Size – размер поля.

Описание элемента. Элемент StringVisible не может являться владельцем других отображаемых элементов. Этот элемент должен быть связан с переменной типа String. Для установления связи используется функция Data со строковым элементом в качестве первого параметра и адресом переменной в качестве второго параметра. Как правило, строковый элемент связывают с меткой. Для организации связи метки со строковым элементом необходимо вызвать функцию Link, передав ей в качестве первого параметра метку, а вторым параметром строковый элемент. Для включения строкового элемента в окно или диалог, необходимо вызвать функцию Insert, с окном или диалогом в качестве первого параметра и строковым элементом в качестве второго параметра. Для уничтожения строки необходимо вызвать функцию Erase, с данным строковым элементом в качестве параметра. Параметр Size задает максимальный размер вводимой строки в символах.

Название элемента: RealVisible (LongVisible) (числовой элемент).

Параметры при создании:

BeginX, BeginY – Координаты верхнего левого угла элемента относительно владельца.

SizeX, SizeY – Горизонтальный и вертикальный размеры элемента.

Min, Max – минимальное и максимальное допустимые значения.

Size – размер поля.

Описание элемента. Элементы RealVisible и LongVisible служат для ввода действительных и длинных целых чисел, соответственно. Любой такой элемент должен быть связан с переменной соответствующего типа (Real или Long) и не может являться владельцем других отображаемых элементов. Для установления связи используется функция Data с числовым элементом в качестве первого параметра и адресом переменной в качестве второго параметра. Как правило, числовой элемент связывают с меткой. Для организации связи метки с числовым элементом необходимо вызвать функцию Link, передав ей в качестве первого параметра метку, а вторым параметром числовой элемент. Для включения числового элемента в окно или диалог, необходимо вызвать функцию Insert, с окном или диалогом в качестве первого параметра и числовым элементом в качестве второго параметра. Для уничтожения числового элемента необходимо вызвать функцию Erase, с данным числовым элементом в качестве параметра.

Название элемента: RadioButtons (переключатели).

Параметры при создании:

BeginX, BeginY – Координаты верхнего левого угла элемента относительно владельца.

SizeX, SizeY – Горизонтальный и вертикальный размеры элемента.

Описание элемента. Элемент RadioButtons служит для задания значения параметрам, которые могут принимать только несколько значений (например, два значения – истина или ложь). Процедура создания элемента RadioButtons сложнее, чем для ранее рассмотренных процедур. Кроме вызова функции RadioButtons, создающей элемент, необходимо несколько раз вызвать функцию AddItem, с элементом RadioButtons в качестве первого аргумента и подписью переключателя в качестве второго аргумента. Элемент RadioButtons должен быть связан с переменной типа Long и не может являться владельцем других отображаемых элементов. Для установления связи используется функция Data с элементом RadioButtons в качестве первого параметра и адресом переменной в качестве второго параметра. Элемент RadioButtons интерпретирует данные, содержащиеся в переменной, следующим образом: первому флагу соответствует младший бит переменной, второму следующий по старшинству и т. д. Элемент не может включать более 32 флагов. Биты с номерами большими числа флагов очищаются (заменяются нулями). Как правило, элемент RadioButtons связывают с меткой. Для организации связи метки с элементом RadioButtons необходимо вызвать функцию Link, передав ей в качестве первого параметра метку, а вторым параметром элемент RadioButtons. Для включения элемента RadioButtons в окно или диалог, необходимо вызвать функцию Insert, с окном или диалогом в качестве первого параметра и элементом

RadioButtons в качестве второго параметра. Для уничтожения элемента RadioButtons необходимо вызвать функцию Erase, с данным элементом RadioButtons в качестве параметра.

Название элемента: CheckBoxes (группа флагов).

Параметры при создании:

BeginX, BeginY – Координаты верхнего левого угла элемента относительно владельца.

SizeX, SizeY – Горизонтальный и вертикальный размеры элемента.

Описание элемента. Элемент CheckBoxes служит для задания значения параметрам, которые являются совокупностью битовых флагов. Процедура создания группы флагов аналогична созданию элемента RadioButtons. Кроме вызова функции CheckBoxes, создающей элемент, необходимо несколько раз вызвать функцию AddItem, с элементом CheckBoxes в качестве первого аргумента и названием флага в качестве второго аргумента. Элемент CheckBoxes должен быть связан с переменной типа Long и не может являться владельцем других отображаемых элементов. Для установления связи используется функция Data с элементом CheckBoxes в качестве первого параметра и адресом переменной в качестве второго параметра. Элемент CheckBoxes интерпретирует данные, содержащиеся в переменной, следующим образом: если значение переменной равно единице, то включен первый переключатель, если двум – то второй, трем – первые два и т. д. Если значение переменной меньше либо равно нулю или больше либо равно два в степени числа переключателей, то оно заменяется на единицу. Как правило, элемент CheckBoxes связывают с меткой. Для организации связи метки с элементом CheckBoxes необходимо вызвать функцию Link, передав ей в качестве первого параметра метку, а вторым параметром элемент CheckBoxes. Для включения группы флагов в окно или диалог, необходимо вызвать функцию Insert, с окном или диалогом в качестве первого параметра и элементом CheckBoxes в качестве второго параметра. Для уничтожения элемента CheckBoxes необходимо вызвать функцию Erase, с данным элементом CheckBoxes в качестве параметра.

Название элемента: Button(кнопка).

Параметры при создании:

BeginX, BeginY – Координаты верхнего левого угла элемента относительно владельца.

SizeX, SizeY – Горизонтальный и вертикальный размеры элемента.

Macro – Адрес функции, вызываемой при нажатии кнопки. В зависимости от реализации по этому адресу может лежать либо начало машинного кода функции, либо начало текста функции. В случае передачи текста функции первые восемь байт по переданному адресу содержат слово «Function».

Описание элемента. Кнопка служит для запуска макроса, который выполняет некоторые действия, являющиеся реакцией на нажатие кнопки. Кнопка не может быть связана с переменными и не может являться владельцем других отображаемых элементов. Для включения кнопки элемента в окно или диалог, необходимо вызвать функцию Insert, с окном или диалогом в качестве первого параметра и кнопкой в качестве второго параметра. Для уничтожения кнопки необходимо вызвать функцию Erase, с данной кнопкой в качестве параметра.

2.6.4 Перечень интерфейсных функций

В данном разделе дано описание всех интерфейсных функций. Приводится синтаксис описания на общем подмножестве языков описания компонентов нейрокомпьютера. Функции приведены в алфавитном порядке. Следует отметить, что, как и в языках описания всех компонентов нейрокомпьютера все аргументы передаются функциям по ссылке (передается не значение аргумента, а его адрес).

AddItem

Function AddItem(Elem : Visible; Text : String) : Logic;

Описание аргументов:

Elem – Отображаемый элемент типа CheckBoxes или RadioButtons.

Text – Название переключателя или флага.

Описание функции:

Эта функция добавляет название переключателя (если первый аргумент типа RadioButtons) или флага (CheckBoxes) к списку элемента, передаваемого функции первым аргументом. Если первый элемент не является элементом типа CheckBoxes или RadioButtons, то функция возвращает значение ложь (False). В случае успешного завершения операции добавления в список функция возвращает значение истина (True). В противном случае возвращается значение ложь (False).

Button

Function Button(BeginX, BeginY, SizeX, SizeY : Long; Macro : PString) : Visible;

Описание аргументов:

BeginX, BeginY – Координаты верхнего левого угла элемента относительно владельца.

SizeX, SizeY – Горизонтальный и вертикальный размеры элемента.

Macro – Адрес функции, вызываемой при нажатии кнопки.

Описание функции:

Эта функция создает отображаемый элемент типа Button. Если создание прошло успешно, то возвращается значение этого элемента (типы значений не оговариваются стандартом, но, как правило, это адрес соответствующей структуры). Если создание элемента завершилось не удачно, то возвращается значение Null.

CheckBoxes

Function CheckBoxes(BeginX, BeginY, SizeX, SizeY : Long) : Visible;

Описание аргументов:

BeginX, BeginY – Координаты верхнего левого угла элемента относительно владельца.

SizeX, SizeY – Горизонтальный и вертикальный размеры элемента.

Описание функции:

Эта функция создает отображаемый элемент типа CheckBoxes с пустым списком переключателей. Для добавления переключателей следует воспользоваться функцией AddItem. Если создание прошло успешно, то возвращается значение этого элемента (типы значений не оговариваются стандартом, но, как правило, это адрес соответствующей структуры). Если создание элемента завершилось не удачно, то возвращается значение Null.

Data

Function Data(Element : Visible; Var Datum) : Logic;

Описание аргументов:

Element – Отображаемый элемент, который связывается с переменной.

Datum – Адрес переменной.

Описание функции:

Эта функция связывает отображаемый элемент (Element) с перемнной Datum. Если элемент Element не допускает установления связи с переменной, то функция возвращает значение ложь (False). В противном случае она устанавливает связь между элементом и переменной и возвращает значение истина (True). Отметим, что функция не проверяет типа переменной. Если вместо адреса переменной типа длинное целое был дан адрес переменной действительного типа, то эта переменная будет интерпретироваться как длинное целое (см. разд. «Функции приведения типов»). Важно отметить, что производится приведение переменной, а не преобразование ее значения.

Delete

Function Delete(Owner, Element : Visible) : Logic;

Описание аргументов:

Owner – Отображаемый элемент типа окно или диалог, из которого происходит удаление.

Element – Удаляемый элемент.

Описание функции:

Эта функция удаляет отображаемый элемент (Element) из его владельца (Owner). Если элемент Owner не является окном или диалогом, или если он не является владельцем элемента Element, то функция возвращает значение ложь (False). В противном случае она удаляет элемент из владельца и возвращает значение истина (True). Отметим, что элемент удаляется, но не уничтожается. Если нет переменной, содержащей удаляемый элемент, то элемент «потеряется», то есть он станет недоступным из программы, но будет занимать память.

Dialog

Function Dialog(BeginX, BeginY, SizeX, SizeY, ScrollX, ScrollY : Long; Text : String) : Visible;

Описание аргументов:

BeginX, BeginY – Координаты верхнего левого угла элемента относительно владельца.

SizeX, SizeY – Горизонтальный и вертикальный размеры элемента.

ScrollX, ScrollY – Целочисленные параметры, задающие наличие у окна горизонтальной и вертикальной полосы прокрутки. Если значение параметра равно нулю, то соответствующая полоса

прокрутки отсутствует, при любом другом значении параметра в окно включается соответствующая полоса прокрутки.

Text – Название окна.

Описание функции:

Эта функция создает отображаемый элемент типа диалог. Если создание прошло успешно, то возвращается значение этого элемента (типы значений не оговариваются стандартом, но, как правило, это адрес соответствующей структуры). Если создание элемента завершилось не удачно, то возвращается значение Null. После создания диалог является пустым.

Erase

Function Erase(Element : Visible) : Logic;

Описание аргументов:

Element – Уничтожаемый элемент.

Описание функции:

Эта функция уничтожает отображаемый элемент (Element). Если аргумент Element является окном или диалогом, то уничтожаются так же все отображаемые элементы, для которых элемент Element является владельцем. Если операция завершена успешно, то функция возвращает значение истина (True). В противном случае – значение ложь (False). Если выполнение функции завершилось неуспешно (функция вернула значение ложь), то элемент может быть поврежден и его дальнейшее использование не гарантирует корректной работы.

Insert

Function Insert(Owner, Element : Visible) : Logic;

Описание аргументов:

Owner – Отображаемый элемент типа окно или диалог, в который производится вставка.

Element – Вставляемый элемент.

Описание функции:

Эта функция вставляет отображаемый элемент (Element) в элемент (Owner). Если элемент Owner не является окном или диалогом, или если Element является диалогом, то функция возвращает значение ложь (False). Такие же действия производятся, в случае, если аргумент Owner совпадает с MainVisible, а Element не является окном или диалогом. В противном случае она вставляет элемент в Owner и возвращает значение истина (True). Вставка окна или диалога в MainVisible вызывает отображение его на экране, а в случае, если вставляется диалог, то ему передается управление.

Label

Function Label(BeginX, BeginY, SizeX, SizeY : Long; Text : String) : Visible;

Описание аргументов:

BeginX, BeginY – Координаты верхнего левого угла элемента относительно владельца.

SizeX, SizeY – Горизонтальный и вертикальный размеры элемента.

Text – текст метки.

Описание функции:

Эта функция создает отображаемый элемент типа Label. Если создание прошло успешно, то возвращается значение этого элемента (типы значений не оговариваются стандартом, но, как правило, это адрес соответствующей структуры). Если создание элемента завершилось не удачно, то возвращается значение Null.

Link

Function Link(Element, Labels : Visible) : Logic;

Описание аргументов:

Owner – Отображаемый элемент, связываемый с меткой.

Element – Отображаемый элемент – метка.

Описание функции:

Эта функция устанавливает связь между меткой Labels и отображаемым элементом Element. Если элемент Labels не является меткой, то функция возвращает значение ложь (False). В противном случае она устанавливает связь и возвращает значение истина (True).

LongVisible

Function LongVisible(BeginX, BeginY, SizeX, SizeY, Min, Max, Size: Long) : Visible;

Описание аргументов:

BeginX, BeginY – Координаты верхнего левого угла элемента относительно владельца.
SizeX, SizeY – Горизонтальный и вертикальный размеры элемента.
Min, Max – минимальное и максимальное допустимые значения.
Size – размер поля в символах.

Описание функции:

Эта функция создает отображаемый элемент типа LongVisible для редактирования и ввода значений типа Long. Если создание прошло успешно, то возвращается значение этого элемента (типы значений не оговариваются стандартом, но, как правило, это адрес соответствующей структуры). Если создание элемента завершилось не удачно, то возвращается значение Null.

RadioButtons

Function RadioButtons(BeginX, BeginY, SizeX, SizeY : Long) : Visible;

Описание аргументов:

BeginX, BeginY – Координаты верхнего левого угла элемента относительно владельца.
SizeX, SizeY – Горизонтальный и вертикальный размеры элемента.

Описание функции:

Эта функция создает отображаемый элемент типа RadioButtons с пустым списком флагов. Для добавления переключателей следует воспользоваться функцией AddItem. Если создание прошло успешно, то возвращается значение этого элемента (типы значений не оговариваются стандартом, но, как правило, это адрес соответствующей структуры). Если создание элемента завершилось не удачно, то возвращается значение Null.

RealVisible

Function RealVisible (BeginX, BeginY, SizeX, SizeY : Long; Min, Max : Real; Size: Long) : Visible;

Описание аргументов:

BeginX, BeginY – Координаты верхнего левого угла элемента относительно владельца.
SizeX, SizeY – Горизонтальный и вертикальный размеры элемента.
Min, Max – минимальное и максимальное допустимые значения.
Size – размер поля в символах.

Описание функции:

Эта функция создает отображаемый элемент типа RealVisible для редактирования и ввода значений типа Real. Если создание прошло успешно, то возвращается значение этого элемента (типы значений не оговариваются стандартом, но, как правило, это адрес соответствующей структуры). Если создание элемента завершилось не удачно, то возвращается значение Null.

Refresh

Function Refresh(Element : Visible) : Logic;

Описание аргументов:

Element – Отображаемый элемент.

Описание функции:

Эта функция обновляет изображение элемента Element на экране. Если операция прошла успешно, то функция возвращает значение истина (True). В противном случае она возвращает значение ложь (False).

StringVisible

Function StringVisible (BeginX, BeginY, SizeX, SizeY, Size: Long) : Visible;

Описание аргументов:

BeginX, BeginY – Координаты верхнего левого угла элемента относительно владельца.
SizeX, SizeY – Горизонтальный и вертикальный размеры элемента.
Size – размер поля в символах.

Описание функции:

Эта функция создает отображаемый элемент типа StringVisible для редактирования и ввода символьных строк. Если создание прошло успешно, то возвращается значение этого элемента (типы значений не оговариваются стандартом, но, как правило, это адрес соответствующей структуры). Если создание элемента завершилось не удачно, то возвращается значение Null.

Window

Function Window(BeginX, BeginY, SizeX, SizeY, ScrollX, ScrollY : Long; Text : String) : Visible;

Описание аргументов:

BeginX, BeginY – Координаты верхнего левого угла элемента относительно владельца.

SizeX, SizeY – Горизонтальный и вертикальный размеры элемента.

ScrollX, ScrollY – Целочисленные параметры, задающие наличие у окна горизонтальной и вертикальной полосы прокрутки. Если значение параметра равно нулю, то соответствующая полоса прокрутки отсутствует, при любом другом значении параметра в окно включается соответствующая полоса прокрутки.

Text – Название окна.

Описание функции:

Эта функция создает отображаемый элемент типа окно. Если создание прошло успешно, то возвращается значение этого элемента (типы значений не оговариваются стандартом, но, как правило, это адрес соответствующей структуры). Если создание элемента завершилось не удачно, то возвращается значение Null. После создания окно является пустым.

2.7 Строковые функции

В этом разделе описан набор функций для работы со строками, которые могут использоваться в языках описания всех компонентов нейрокомпьютера.

Function SubStr(S : String; Origin, Leng : Integer) : String;

Описание аргументов

S – строка, из которой выделяется фрагмент.

Origin – начальная позиция выделяемого фрагмента в строке S

Leng – длина выделяемого фрагмента.

Выделяет из строки S фрагмент, начинающийся с позиции Origin и длиной Leng символов. Если строка короче чем Origin, то результатом является пустая строка. Если строка длиннее чем Origin символов, но короче чем Origin+Leng символов, то результатом является фрагмент строки S с символа Origin и до конца строки S.

Function Pos(S1, S2 : String) : Integer

Описание аргументов

S1 – строка, в которой ищется вхождение строки S2.

S2 – строка, вхождение которой ищется.

Функция Pos возвращает номер первого символа в строке S1, начиная с которого, в строке S1 полностью содержится строка S2. Если строка S2 ни разу не встретилась в строке S1, то результат равен нулю.

Function Len(S : String) : Integer

Описание аргументов

S – строка, длина которой вычисляется.

Функция Len возвращает длину (число символов) строки S

2.8 Описание языка описания компонентов

В табл. 12 приведен список ключевых слов, общих для всех языков описания компонентов нейрокомпьютера. Кроме того, к ключевым словам относятся типы данных, приведенные в табл. 1; обозначения операций, приведенные в табл. 3, 7, 8, 9, 10; названия функций преобразования (табл. 5) и приведения типов (табл. 6); идентификаторы предопределенных констант, приведенные в табл. 11; имена интерфейсных функций, приведенных в разделе «Перечень интерфейсных функций»; обозначения элементарных функций, приведенных в табл.13; обозначения строковых функций, приведенных в разделе «Строковые функции» и обозначения функций управления памятью из раздела «функции управления памятью».

2.8.1 Передача аргументов функциям

Во всех языках описания компонентов все параметры передаются по ссылке (передается не значение аргумента, а его адрес). Если в качестве фактического аргумента указано выражение, то значение выражения помещается интерпретатором (или компилятором) во временную переменную, имеющую тип, совпадающий с типом формального аргумента, а адрес временной переменной передается в качестве фактического аргумента.

Таблица 12.

Ключевые слова, общие для всех языков описания компонент нейрокомпьютера.

Ключевое слово	Краткое описание
Begin	Начало описания тела процедуры, или операторных скобок.
By	Часть оператора цикла с шагом. Предшествует шагу цикла.
Do	Завершающая часть операторов цикла.
Else	Часть условного оператора. Предшествует оператору, выполняемому, если условие ложно.
End	Конец описания тела процедуры или операторных скобок.
For	Заголовок оператора цикла с шагом.
Function	Заголовок описания функции.
Global	Начло блока описания глобальных переменных.
GoTo	Начало оператора перехода.
If	Начало условного оператора.
Include	Предшествует имени файла, целиком вставляемого в это место описания.
Label	Начало описания меток
Name	Предшествует имени статической переменной.
SetParameters	Признак раздела установления значений параметров.
Static	Начло блока описания статических переменных.
Then	Часть условного оператора, предшествующая оператору, выполняемому, если условие истинно.
To	Часть оператора цикла с шагом. Предшествует верхней границе цикла.
Var	Начло блока описания переменных.
While	Заголовок оператора цикла по условию.

Таблица 13

Элементарные функции, допустимые в языках описания компонент нейрокомпьютера

Обозначение	Значение	Обозначение	Значение
Sin	Синус	Cos	Косинус
Tan	Тангенс	Atan	Арктангенс
Sh	Гиперболический синус	Ch	Гиперболический косинус
Th	Гиперболический тангенс	Lg	Логарифм двоичный
Ln	Логарифм натуральный	Exp	Экспонента
Sqrt	Квадратный корень	Sqr	Квадрат
Abs	Абсолюеное значение	Sign	Знак аргумента (0 – минус)

2.8.2 Имена структурных единиц компонентов

Компоненты предобработчик, сеть, оценка и интерпретатор ответа имеют иерархическую структуру. Часть запросов может быть адресована не всему компоненту, а его структурной единице любого уровня. Для точного указания адресата запроса используется полное имя структурной единицы, которое строится по следующему правилу:

1. Имя компонента является полным именем компонента.
2. Полное имя младшей структурной единицы строится путем добавления справа к имени старшей структурной единицы точки, псевдонима младшей структурной единицы и номера экземпляра младшей структурной единицы, если младших структурных единиц с таким псевдонимом несколько.

Иногда при построении описания компонента требуется однозначное имя структурной единицы. В качестве однозначного имени можно использовать полное имя, но такой подход лишает возможности вставлять подготовленные структурные единицы в структуры более высокого уровня. Для этого вводится понятие однозначного имени структурной единицы: в описании структурной единицы *A* однозначным именем структурной единицы *B*, являющейся частью структурной единицы *A,* является полное имя структурной единицы *B*, из которого исключено полное имя структурной единицы *A*.

2.8.3 Описание синтаксических конструкций

Для описания синтаксиса языков описаний компонентов используется расширенная Бэкусова нормальная форма. В описании БНФ во всей книге приняты следующие обозначения:

- <Имя> – нетерминальный символ – понятие которое было раскрыто или будет раскрыто далее;

- *Имя* – терминальный символ – понятие, не требующее раскрытия;
- **Имя** – ключевое слово – подмножество терминальных символов;
- [Имя] – необязательный элемент – синтаксическая конструкция, заключенная в скобки может отсутствовать;
- {Имя1 | Имя2 | Имя3}– одно из значений – может присутствовать одно и только одно из разделенных символом | значений.

В данном разделе приведено описание общего подмножества языков описания компонентов. В некоторых случаях, когда БНФ описание понятия сложно, а неформальное описание просто и однозначно, в БНФ описание включаются фрагменты неформального описания таких понятий.

Список синтаксических конструкций общего назначения:

<Идентификатор> ::= <Буква> [<Символьная строка>]

<Буква> ::= {*a* | *b* | *c* | *d* | *e* | *f* | *g* | *h* | *i* | *j* | *k* | *l* | *m* | *n* | *o* | *p* | *q* | *r* | *s* | *t* | *u* | *v* | *w* | *x* | *y* | *z* | *A* | *B* | *C* | *D* | *E* | *F* | *G* | *H* | *I* | *J* | *K* | *L* | *M* | *N* | *O* | *P* | *Q* | *R* | *S* | *T* | *U* | *V* | *W* | *X* | *Y* | *Z*}

<Символьная строка> ::= {<Буква> | <Цифра> | _ } [<Символьная стока>]

<Цифра> ::= {*0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9*}

<Число> ::= {<Целое число> | <Действительное число>}

<Целое число> ::= [–] <Положительное целое число>

<Положительное целое число> ::= <Цифра> [<Положительное целое число>]

<Действительное число> ::= <Целое число>[**.**<Положительное целое число>][**e**<Целое число>]

<Целочисленная константа> ::= {<Предопределенная константа типа Integer> | < Предопределенная константа типа Long> | <Целое число>}

<Цветовая константа> ::= *H* <Шестнадцатеричная цифра> <Шестнадцатеричная цифра> <Шестнадцатеричная цифра> <Шестнадцатеричная цифра>

<Шестнадцатеричная цифра> ::= {*0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *A* | *B* | *C* | *D* | *E* | *F* }

<Строковая константа> ::= “<Строка произвольных символов>”

<Логическая константа> ::= {**True** | **False**}

<Строка произвольных символов> – Последовательность произвольных символов из набора ANSI. В этой последовательности допускаются символы национальных алфавитов. При необходиости включить в эту конструкцию символ кавычек, он должен быть удвоен.

<Скалярный тип> ::= {**Long** | **Real** | **Integer** | **Color** | **Logic** | **String** | **PRealArray** | **PIntegerArray** | **PLongArray** | **PLogicArray** | **PString** | **Visual** | **Pointer** | **FuncType**}

<Тип массива> ::= { **RealArray** | **IntegerArray** | **LongArray** | **LogicArray**}

<Константа типа ***Тип***> – константа имеющая тип ***Тип***.

Список синтаксических конструкций для формальных аргументов:

<Список формальных аргументов> ::= <Формальный аргумент> [; <Список формальных аргументов>]

<Формальный аргумент> ::= <Список имен аргументов> : <Скалярный тип>

<Список имен аргументов> ::= <Имя аргумента> [,<Список имен аргументов>]

<Имя аргумента> ::= <Идентификатор>

<Аргумент типа ***Тип***> – одно из следующих понятий:

имя аргумента, который при описании формальных аргументов имел тип ***Тип***

имя элемента аргумента-массива, если элементы массива имеют тип ***Тип***

результат приведения произвольного аргумента или элемента аргумента-массива к типу ***Тип.***

Синтаксические конструкции описания переменных:

<Описание переменных> ::= **Var** <Список описаний однотипных переменных>

<Список описаний однотипных переменных> ::= <Тип переменной> <Список переменных>*;* [<Список описаний однотипных переменных>]

<Список переменных> ::= <Имя переменной> [, <Список переменных>]

<Имя переменной> ::= <Идентификатор>

<Тип переменной> ::= {<Скалярный тип> | <Тип массива>*[*<Целочисленное константное выражение>*]*}

<Переменная типа ***Тип***> – одно из следующих понятий:

имя переменной, которая при описании переменных имела тип ***Тип***

имя элемента массива, если элементы массива имеют тип ***Тип***

результат приведения произвольной переменной или элемента массива к типу ***Тип.***

Синтаксическая конструкция описания глобальных переменных (доступна только в языках описания компонентов учитель и контрастер):

<Описание глобальных переменных> ::= **Global** <Список описаний однотипных переменных>

Синтаксические конструкции описания статических переменных

Статические переменные, как правило, служат для описания параметров компонентов нейрокомпьютера. Использование в именах переменных только символов латинского алфавита и цифр делает идентификаторы универсальными, но неудобными для всех пользователей, кроме англо-говорящих. Для удобства всех остальных пользователей в описании статических переменных предусмотрена возможность использовать дополнительные имена для статических переменных. Однако эти имена служат только для построения интерфейса и не могут быть использованы в описании тела соответствующего компонента. Кроме того, статической переменной можно при описании задать значение по умолчанию.

<Описание статических переменных> ::= **Static** <Список описаний статических переменных>

<Список описаний статических переменных> ::= <Описание статической переменной>; [<Список описаний статических переменных>]

<Описание статической переменной> ::= <Тип переменной> <Имя переменной> [**Name** <Имя статической переменной>] [**Default** <Значение по умолчанию>]

<Имя статической переменной> ::= <Строковая константа>

<Значение по умолчанию> ::= <Константное выражение типа <Тип переменной>>

Синтаксические конструкции описания функций

<Описание функций> ::= <Описание функции> [<Описание функций>]

<Описание функции> ::= <Заголовок функции> <Описание переменных> <Описание меток> <Тело функции>

<Заголовок функции> ::= **Function** <Имя функции>[*(*<Список формальных аргументов>*)*] : <Скалярный тип>*;*

<Описание меток> ::= **Label** <Список меток>*;*

<Список меток> ::= <Имя метки> [, <Список меток>]

<Имя метки> ::= <Идентификатор>

<Тело функции> ::= **Begin** <Составной оператор> **End;**

<Составной оператор> ::= [<Имя метки>:] <Оператор> [; <Составной оператор>]

<Оператор> ::= {<Оператор присваивания> | <Оператор ветвления> | <Оператор цикла> | <Оператор перехода> | <Операторные скобки>}

<Оператор присваивания> ::= <Допустимое имя переменной> **=** <Выражение>

<Оператор ветвления> ::= **If** <Логическое выражение> **Then** <Оператор> [**Else** <Оператор>]

<Оператор цикла> ::= { <Цикл For> | <Цикл While> }

<Цикл For> ::= **For** <Имя переменной> = <Целочисленное выражение> **To** <Целочисленное выражение> [**By** <Целочисленное выражение>] **Do** <Оператор>

<Цикл While> ::= **While** <Логическое выражение> **Do** <Оператор>

<Оператор перехода> ::= **GoTo <**Имя метки**>**

<Операторные скобки> ::= **Begin** <Составной оператор> **End**

<Функция типа ***Tun*****>** – функция, возвращающая величину типа ***Tun***.

<Допустимое имя переменной> – допустимой переменной являются все переменные, описанные в данной функции или в данном процедурном блоке, глобальные переменные данного компонента. Для возвращения значения функции, в левой части оператора присваивания должно стоять имя функции.

Синтаксические конструкции описания выражений:

<Выражение> ::= { <Выражение типа ***Long***> | <Выражение типа ***Real***> | <Выражение типа ***Integer***> | <Выражение типа ***Color***> | <Выражение типа ***Logic***> | <Выражение типа ***String***> | <Выражение типа ***Pointer***>}

<Целочисленное выражение> ::= { <Выражение типа ***Long***> | <Выражение типа ***Integer***>}

<Выражение типа ***Tun***> ::= [<Префиксная операция типа ***Tun***>] <Операнд типа ***Tun***> [<Операция типа ***Tun***> <Операнд типа ***Tun***>]

<Операция типа ***Long***> ::= {**+** | **–** | ***** | **Div** | **Mod** | **And** | **Or** | **Xor**}

<Операция типа ***Real***>::= {**+** | **–** | ***** | **/** | **RMod** }

<Операция типа ***Integer***> ::= {**+** | **–** | ***** | **Div** | **Mod** | **And** | **Or** | **Xor**}

<Операция типа ***Color***> ::= {**COr** | **CAnd** | **CXor**}

<Операция типа ***Logic***> ::= {**And** | **Or** | **Xor**}
<Операция типа ***String***> ::= +
<Префиксная операция типа ***Long***> ::= { **–** | **Not** }
<Префиксная операция типа ***Real***>::= **–**
<Префиксная операция типа ***Integer***> ::= { **–** | **Not** }
<Префиксная операция типа ***Color***> ::= **CNot**
<Префиксная операция типа ***Logic***> ::= **Not**
<Операнд типа ***Logic***> ::= ::= {<Результат сравнения> | <Выражение типа ***Logic***> | *(*<Выражение типа ***Logic***>*)* | <Константа типа ***Logic***> | <Переменная типа ***Logic***> | <Аргумент типа ***Logic***> | <Вызов функции типа ***Logic***>}
<Результат сравнения типов ***Long, Integer, Real***> ::= *(*<Выражение типа ***Long, Integer, Real***> {> | < | >= | <= | = | <>} <Выражение типа ***Long, Integer, Real***> *)*
<Результат сравнения типа ***Color***> ::= *(*<Выражение типа ***Color***> {**CEqual** | **CIn** | **CInclude** | **CExclude** | **CIntersect**} <Выражение типа ***Color***> *)*
<Результат сравнения типа ***String***> ::= *(*<Выражение типа ***String***> {= | <>} <Выражение типа ***String***> *)*
<Операнд типа ***Tun***> ::= {<Выражение типа ***Tun***> | *(*<Выражение типа ***Tun***>*)* | <Константа типа ***Tun***> | <Переменная типа ***Tun***> | <Аргумент типа ***Tun***> | <Вызов функции типа ***Tun***>}
<Вызов функции типа ***Tun***> ::= <Имя функции типа ***Tun***> [*(*<Список фактических аргументов>*)*]
<Список фактических аргументов> ::= <Выражение> [,<Список фактических аргументов>]
<Константное выражение типа ***Tun***> – <Выражение типа ***Tun***> в операндах которого не могут фигурировать переменные и функции, описанные пользователем.
<Числовое выражение> ::= { <Выражение типа ***Long***> | <Выражение типа ***Real***> | <Выражение типа ***Integer***>}

Синтаксические конструкции задания значений статическим переменным

Эта конструкция служит для задания значений параметрам (статическим переменным) компонентов. Для компонента сеть она может встречаться не только при описании главной сети, но и при описании любой составной подсети. В специальных выражениях типа ***Tun*** могут участвовать только стандартные функции и аргументы той структурной единицы, в которой находится блок задания значений статическим переменным. При этом специальное выражение, задающее значение параметра должно иметь тип, совместимый с типом статической переменной, которой присваивается это значение.

<Установление параметров ***Структурной единицы***> ::= <Однозначное имя ***Структурной единицы***> [*[*[<Переменная цикла>:] <Начальный номер> [..<Конечный номер> [:<Шаг>] *]]*] **SetParameters <**Список значений параметров**>**
<Переменная цикла> ::= <Идентификатор>
<Начальный номер> ::= <Константное выражение типа ***Long***>
<Конечный номер> ::= <Константное выражение типа ***Long***>
<Шаг> ::= <Константное выражение типа ***Long***>
<Список значений параметров> ::= <Значение параметра> [,<Список значений параметров>]
<Значение параметра> ::= <Специальное выражение типа ***Tun***>
<Специальное выражение типа ***Tun***> ::= [<Префиксная операция типа ***Tun***>] <Специальный операнд типа ***Tun***> [<Операция типа ***Tun***> <Специальный операнд типа ***Tun***>]
<Специальный операнд типа ***Tun*** > ::= {<Специальное выражение типа ***Tun*** > | **<**Константа типа ***Tun***> | <Переменная цикла> | (<Специальное выражение типа ***Tun*** > | <Аргумент типа ***Tun***> | <Вызов функции типа ***Tun***>)>

Синтаксические конструкции описания распределения сигналов или параметров:

Данная конструкция имеет четыре аргумента, имеющих следующий смысл:

Данное – сигнал или параметр.
Объект – предобработчик, интерпретатор, оценка, сеть.
Подобъект – частный предобработчик, частный интерпретатор, частная оценка, подсеть.
<Идентификатор данных> – одно из ключевых слов Signals, Parameters, Data, InSignals, OutSignals.

<Описание распределения ***Данных, Объекта, Подобъекта, <Идентификатор данных>***> ::= **Connections** <Описание групп соответствий ***Данных***>
<Описание групп соответствий ***Данных***> ::= <Описание группы соответствий ***Данных***> [;<Описание групп соответствий ***Данных***>]
<Описание группы соответствий ***Данных***> ::= <Блок сигналов ***Подобъекта***> <=> {<Блок сигналов ***Объекта***> | <Блок сигналов ***Подобъекта***>}

<Блок сигналов ***Подобъекта***> ::= <Описатель сигналов ***Подобъекта***> [;<Блок сигналов ***Подобъекта***>]
<Описатель сигналов ***Подобъекта***> ::= { **For** <Переменная цикла> = <Начальный номер> **To** <Конечный номер> [**Step** <Шаг>] **Do** <Блок сигналов ***Подобъекта***> **End** | <Список ***Данных Подобъекта***>}
<Переменная цикла> ::= <Идентификатор>
<Список ***Данных Подобъекта***> ::= <***Данное Подобъекта***>[; <Список ***Данных Подобъекта***>]
<***Данное Подобъекта***> ::= <Псевдоним>[*[*<Номер экземпляра>*]*]**.<*Идентификатор данных*>** [*[*<Номер ***Данного***>*]*]
<Номер экземпляра> ::= {<Специальное выражение типа ***Long***> | [+:]<Начальный номер> [..<Конечный номер> [:<Шаг>]]}
<Номер ***Данного***> {<Специальное выражение типа ***Long***> | [+:]<Начальный номер> [..<Конечный номер> [:<Шаг>]]}
<Блок ***Данных Объекта***> ::= <Описатель ***Данных Объекта***> [; <Блок ***Данных Объекта***>]
<Описатель ***Данных Объекта***> ::= { **For** <Переменная цикла> = <Начальный номер> **To** <Конечный номер> [**Step** <Шаг>] **Do** <Блок ***Данных Объекта***> **End** | <Список ***Данных Объекта***> }
<Список ***Данных Объекта***> ::= <***Данное Объекта***>[; <Список ***Данных Объекта***>]
<***Данное Объекта***> ::= **<*Идентификатор данных*>** [*[*<Номер ***Данного***>*]*]

2.8.4 Комментарии

Для понятности описаний компонентов в них необходимо включать комментарии. Комментарием является любая строка (или несколько строк) символов, заключенных в фигурные скобки. Комментарий может находиться в любом месте описания компонента. При интерпретации или компиляции описания комментарии игнорируются (исключаются из текста).

2.8.5 Область действия переменных

Все идентификаторы состоят из произвольных комбинаций латинских букв, цифр и подчерков. Первым символом имени обязательно является буква. Использование букв только латинского алфавита связано с тем, что коды, используемые большинством компьютеров, имеют одинаковую кодировку для букв латинского алфавита, тогда как для букв национальных алфавитов других стран кодировка различна не только от компьютера к компьютеру но и от одной операционной системы к другой.

Заглавные и прописные буквы не различаются ни в именах, ни в ключевых словах.

Языки описания некоторых компонентов позволяют описывать глобальные переменные. Эти переменные доступны во всех функциях и процедурных блоках данного компонента. Функциям и процедурным блокам других компонентов эти переменные недоступны. Все остальные переменные (описанные в блоках Var и Static) являются локальными и доступны только в пределах той функции или процедурного блока, в котором они описаны. Статические переменные сохраняют свое значение между вызовами функций или процедурных блоков, тогда как переменные, описанные в блоках Var не сохраняют. В некоторых компонентах определены стандартные переменные и массивы (см. например описание языка описания нейронных сетей). В таких разделах область доступности предопределенных переменных оговаривается отдельно.

Переменная Error является глобальной для всех компонентов. Глобальной является также переменная ErrorManager. Однако не рекомендуется использование этих переменных путем прямого обращения к ним. Для получения значения переменной Error служит запрос GetError, исполняемый макрокомпонентом нейрокомпьютер.

2.8.6 Основные операторы

Оператор присваивания состоит из двух частей, разделенных знаком “=“. В левой части оператора присваивания могут участвовать имена любых переменных. В выражении, стоящем в правой части оператора присваивания могут участвовать любые переменные, аргументы процедурного блока и константы. В случае несоответствия типа выражения в правой части и типа переменной в левой части оператора присваивания производится приведение типа. Все выражения вычисляются слева на право с учетом старшинства операций.

Оператор ветвления. Оператор ветвления состоит из трех частей, каждая из которых начинается соответствующим ключевым словом. Первая часть – условие, начинается с ключевого слова If и содержит логическое выражение. В зависимости от значения вычисленного логического выражения выполняется Then часть (истина) или Else часть (ложь). Третья (Else) часть оператора может быть опущена. Каждая из выполняемых частей состоит из ключевого слова и оператора. При необходимости выполнить несколько операторов, необходимо использовать операторные скобки Begin End.

Цикл For имеет следующий вид:

For Переменная_цикла = Начальное_значение **To** Конечное_значение [**By** Шаг] **Do** <Оператор>

Переменная цикла должна быть одного из целочисленных типов. В ходе выполнения оператора она пробегает значения от Начальное_значение до Конечное_значение с шагом Шаг. Если описание шага опущено, то шаг равен единице. При каждом значении переменной цикла из диапазона выполняется оператор, являющийся телом цикла. Если в теле цикла необходимо выполнить несколько операторов, то необходимо воспользоваться операторными скобками. Допускается любое число вложенных циклов. Выполнение цикла в зависимости от соотношения между значениями Начальное_значение, Конечное_значение и Шаг приведено в табл. 14.

Таблица 14.

Способ выполнения цикла в зависимости от значений параметров цикла.

Конечное значение	Шаг	Способ выполнения
>Начального значения	>0	Цикл выполняется пока переменная цикла ≤ Конечного значения
<Начального значения	>0	Тело цикла не выполняется
=Начальному значению	≠0	Тело цикла выполняется один раз
>Начального значения	<0	Тело цикла не выполняется
<Начального значения	<0	Цикл выполняется пока переменная цикла ≥ Конечного значения
	=0	Тело цикла не выполняется

Цикл While. Тело цикла выполняется до тех пор, пока верно логическое выражение. Проверка истинности логического выражения производится перед выполнением тела цикла. Если тело цикла должно содержать более одного оператора, то необходимо использовать операторные скобки.

2.8.7 Описание распределения сигналов

Раздел описания распределения сигналов начинается с ключевого слова Connections. За ключевым словом Connections следует одна или несколько групп соответствий. Каждая группа соответствий состоит из правой и левой частей, разделенных символами «<=>« и описывает соответствие имен сигналов (параметров) различных структурных единиц. Каждая часть группы соответствий представляет собой список сигналов (параметров) или интервалов сигналов (параметров), разделенных между собой символом «;». Указанные в левой и правой частях сигналы (параметры) отождествляются. Если при указании сигнала (параметра) не указано имя подобъекта, то это сигнал (параметр) описываемого объекта. Использование интервала сигналов (параметров) в правой или левой части группы соответствий равносильно перечислению сигналов (параметров), с номерами, входящими в интервал, начиная с начального номера с шагом, указанным после символа «:». Если шаг не указан, то он полагается равным единице. Число сигналов в правой и левой частях группы соответствий должно совпадать. Если интервал пуст (например [2..1:1]), то описываемая им группа сигналов считается отсутствующей и пропускается. При использовании в описании соответствий явных циклов, во всех выражениях внутри цикла возможно использование переменной цикла. При этом подразумевается следующий порядок перечисления: Сначала изменяется номер в самом правом интервале, далее во втором справа, и т.д. В последнюю очередь изменяются значения переменных цикла явных циклов в порядке их вложенности (переменная самого внутреннего цикла меняется первой и т.д.). Рассмотрим следующий пример описания группы соответствий блока, содержащего две сети Net с 3 входами каждая. Ниже приведено две различных структуры связей по несколько эквивалентных вариантов описания.

Случай 1. Естественный порядок связей.

Вариант 1.

InSignals[1] <=> Net[1].**InSignals**[1]
InSignals[2] <=> Net[1].**InSignals**[2]
InSignals[3] <=> Net[1].**InSignals**[3]
InSignals[4] <=> Net[2].**InSignals**[1]
InSignals[5] <=> Net[2].**InSignals**[2]
InSignals[6] <=> Net[2].**InSignals**[3]

Вариант 2.

InSignals[1..6] <=> Net[1..2].**InSignals**[1..3]

Вариант 3.

InSignals[1]; **InSignals**[2]; **InSignals**[3]; **InSignals**[4]; **InSignals**[5]; **InSignals**[6] <=>
For I=1 **To** 3 **Do For** J=1 **To** 2 **Do** Net[J].**InSignals**[I] **End End**

Случай 2. Другой порядок связей.

Вариант 1.

InSignals[1] <=> Net[2].**InSignals**[3]
InSignals[2] <=> Net[1].**InSignals**[3]

InSignals[3] <=> Net[2].**InSignals**[2]
InSignals[4] <=> Net[1].**InSignals**[2]
InSignals[5] <=> Net[2].**InSignals**[1]
InSignals[6] <=> Net[1].**InSignals**[1]

Вариант 2.

InSignals[1..6] <=> **For** I=3 **To** 1 **Step** -1 **Do** Net[2..1:-1].**InSignals**[I] **End**

Вариант 3.

InSignals[6..1:-2]; **InSignals**[5..1:-2]<=>
For I=1 **To** 3 **Do For** J=1 **To** 2 **Do** Net[J].**InSignals**[I] **End End**

2.8.8 Функции управления памятью

Для создания массивов и освобождения занимаемой ими памяти используются следующие функции:

Создание массива.

Function NewArray(Type : Integer; Size : Long) : PRealArray;

Описание аргументов:

Type – задает размер элемента массива и является одной из предопределенных констант, приведенных в табл. 15.

Size – число элементов в массиве.

Описание исполнения.

1. Если аргумент Type не совпадает ни с одной из предопределенных констант, приведенных в табл. 15, то возвращается значение Null, исполнение функции завершается.
2. Создается массив, занимающий Size*Type+4 байта.
3. Адрес массива возвращается как результат.

Таблица 15.

Предопределенные константы типов элементов массивов

Идентификатор	Значение	Описание
mRealArray	4	Размер элемента – 4 байта
mIntegerArray	2	Размер элемента – 2 байта
mLongArray	4	Размер элемента – 4 байта
mLogicArray	1	Размер элемента – 1 байт

Освобождение массива.

Function FreeArray(Type : Integer; Array : PRealArray) : Logic;

Описание аргументов:

Type – задает размер элемента массива и является одной из предопределенных констант, приведенных в табл. 14.

Array – адрес массива. Память, занимаемая этим массивом, должна быть освобождена.

Описание исполнения.

1. Если аргумент Type не совпадает ни с одной из предопределенных констант, приведенных в табл. 15, то возвращается значение False, исполнение функции завершается.
2. Освобождается память размером TReal(Array[0])*Type+4 байта.
3. Аргументу Array присваивается значение Null

Пересоздание массива.

Function ReCreateArray(Type : Integer; Array : PRealArray; Size : Long) : Logic;

Описание аргументов:

Type – задает размер элемента массива и является одной из предопределенных констант, приведенных в табл. 15.

Array – адрес массива.

Size – число элементов в массиве.

Описание исполнения.

1. Если аргумент Type не совпадает ни с одной из предопределенных констант, приведенных в табл. 15, то возвращается значение False, исполнение функции завершается.
2. Если аргумент Array не равен Null, и TReal(Array[0]) равен Size, то возвращается значение True, выполнение функции завершается.
3. Если аргумент Array не равен Null, и TReal(Array[0]) не равен Size, то освобождается память размером TReal(Array[0])*Type+4 байта. Аргументу Array присваивается значение Null
4. Аргументу Array присваивается значение NewArray(Type,Size), возвращается значение True, исполнение функции завершается.

2.9 Использование памяти

Ряд запросов, исполняемых различными компонентами, возвращают в качестве ответа указатели на массивы. В этих случаях действуют следующие правила:

1. Если компонент получил пустой указатель (Null), то он сам создает массив необходимой длины.
2. Если передан непустой указатель, но существующей длины массива недостаточно, то компонент освобождает память, занятую под переданный массив и создает новый массив необходимой длины.
3. Освобождение памяти после использования массива лежит на вызывающем компоненте.

Если одному из компонентов не хватает памяти для выполнения запроса, то этот компонент может передать макрокомпоненту нейрокомпьютер запрос на дополнительную память. В этом случае макрокомпонент нейрокомпьютер передает всем компонентам запрос **FreeMemory.** При исполнении данного запроса каждый компонент должен освободить всю память, не являющуюся абсолютно необходимой для работы. Например, компонент задачник может для быстроты обработки держать в памяти все обучающее множество. Однако абсолютно необходимой является память, достаточная для хранения в памяти одного примера.

Запрос на освобождение памяти исполняется каждым компонентом и не включен в описания запросов компонентов, приведенные в следующих главах.

2.10 Обработка ошибок

Схема обработки ошибок достаточно проста по своей идее - каждый новый обработчик ошибок может обрабатывать только часть ошибок, а обработку остальных может передать ранее установленному обработчику. Пользователь может организовать обработку ошибок и не прибегая к установке обработчика ошибок - обработчик ошибок по умолчанию почти во всех случаях устанавливает номер последней ошибки в переменную Error, которая может быть считана с помощью запроса GetError и обработана прямо в компоненте, выдавшем запрос.

Если обработчик ошибок устанавливает номер последней ошибки в переменной Error, то все запросы, поступившие после момента установки, завершаются неуспешно. Это состояние сбрасывается при вызове запроса «дать номер ошибки».

2.10.1 Процедура обработки ошибок

Процедура обработки ошибок должна удовлетворять следующим требованиям:

I. Это должна быть процедура с дальним типом адресации. Формат описания процедуры обработки ошибок

Pascal:

```
Procedure ErrorFunc( ErrorNumber : Long ); Far;
```

C:

```
void far ErrorFunc(Long ErrorNumber)
```

II. После обработки ошибок процедура может вызвать ранее установленный обработчик ошибок. Адрес ранее установленного обработчика ошибок процедура обработки ошибок получает в ходе следующей процедуры:

A. Вызов процедуры с нулевым номером ошибки означает, что в следующем вызове будет передан адрес старой процедуры обработки ошибок.

B. Значение аргумента ErrorNumber при вызове, следующем непосредственно за вызовом с нулевым номером ошибки, должно интерпретироваться как адрес старой процедуры обработки ошибок.

Ниже приведено описание запросов, связанных с обработкой ошибок и исполняемых макрокомпонентом нейрокомпьютер.

2.10.2 Установить обработчик ошибок (OnError)

Описание запроса:

Pascal:

```
Function OnError( NewError : ErrorFunc ) : Logic;
```

C:

```
Logic OnError(ErrorFunc NewError)
```

Описание аргументов:

NewError - адрес новой процедуры обработки ошибок.

Назначение -

Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Вызов NewError с аргументом 0 - настройка на установку цепочки обработки ошибок.
3. Вызов NewError с аргументом ErrorManager (вместо длинного целого передается адрес старой процедуры обработки ошибок).
4. ErrorManager := NewError

2.10.3 Дать номер ошибки (GetError)

Описание запроса:
Pascal:

```
Function GetError : Integer;
```

C:

```
Integer GetError()
```

Назначение - возвращает номер последней необработанной ошибки и сбрасывает ее.
Описание исполнения.

1. GetError := Error
2. Error := 0

Списки ошибок, возникающих в различных компонентах, даны в разделах «Ошибки компоненты ...», в соответствующих главах. Все номера ошибок каждого компонента являются трехзначными числами и начинаются с номера компонента, указанного в колонке «Ошибка» табл. 16.

2.11 Запросы, однотипные для всех компонент

Ряд запросов обрабатывается всеми компонентами, кроме компонента исполнитель, носящего вспомогательный характер. Один из таких запросов – FreeMemory – был описан в разделе «Управление памятью», а два запроса, связанных с обработкой ошибок – в разделе «Обработка ошибок». В данном разделе приводятся описания остальных запросов, имеющих одинаковый смысл для всех компонентов. В отличие от ранее описанных запросов эти запросы опираются на структуру исполняющего компонента, поэтому к имени запроса добавляется префикс, задающий компонента. Список префиксов приведен в табл. 16. Единственным исключением из числа компонентов, исполняющих перечисленные в данном разделе запросы, является компонент исполнитель.

Таблица 16

Префиксы компонент

ПрефиксЗапросаОшибки	ПрефиксКомпонента	
ex	0	Исполнитель
tb	1	Задачник
pr	2	Предобработчик
nn	3	Сеть
es	4	Оценка
ai	5	Интерпретатор ответа
in	6	Учитель
cn	7	Контрастер

Все описываемые в данном разделе запросы можно разбить на четыре группы:

1. Установление текущего компонента.
2. Запросы работы со структурой компонента.
3. Запросы на получение или изменение параметров структурной единицы.
4. Запуск редактора компонента.

Все имена запросов начинаются с символов «xx», которые необходимо заменить на префикс из табл. 16 чтобы получить имя запроса для соответствующего компонента. При указании ошибок используется символ «n», который нужно заменить на соответствующий префикс ошибки из табл. 16.

Далее данном разделе компонентом также называются экземпляры компонента, а не только часть программы. Например, одна из загруженных нейронных сетей, а не только программный компонент сеть.

2.11.1 Запрос на установление текущего компонента

К этой группе запросов относится один запрос – xxSetCurrent – не исполняемый компонентом задачник.

2.11.1.1 Сделать текущей (xxSetCurrent)

Описание запроса:
Pascal:

```
Function xxSetCurrent( CompName : PString) : Logic;
```

C:

```
Logic xxSetCurrent(PString CompName)
```

Описание аргумента:

CompName – указатель на строку символов, содержащую имя компонента, которого надо сделать текущим..

Назначение – ставит указанного в параметре CompName компонента из списка загруженных компонентов на первое место в списке.

Описание исполнения.

1. Если список компонентов пуст или имя компонента, переданное в аргументе CompName, в этом списке не найдено, то возникает ошибка n01 – неверное имя компонента, управление передается обработчику ошибок, а обработка запроса прекращается.
2. Указанный в аргументе CompName компонент переносится в начало списка.

2.11.2 Запросы, работающие со структурой компонента.

К этой группе относятся запросы, позволяющие выяснить структуру компонента, прочитать ее или сохранить на диске.

2.11.2.1 Добавление нового экземпляра (xxAdd)

Описание запроса:

Pascal:

```
Function xxAdd( CompName : PString ) : Logic;
```

C:

```
Logic xxAdd(PString CompName)
```

Описание аргумента:

CompName – указатель на строку символов, содержащую имя файла компонента или адрес описания компонента.

Назначение – добавляет новый экземпляр компонента в список компонентов.

Описание исполнения.

1. Если в качестве аргумента CompName дана строка, первые четыре символа которой составляют слово File, то остальная часть строки содержит имя компонента и после пробела имя файла, содержащего компонента. В противном случае считается, что аргумент CompName содержит указатель на область памяти, содержащую описание компонента в формате для записи на диск. Если описание не вмещается в одну область памяти, то допускается включение в текст описания компонента ключевого слова Continue, за которым следует четыре байта, содержащие адрес следующей области памяти.
2. Экземпляр компонента считывается из файла или из памяти и добавляется *первым* в список компонентов (становится текущим).
3. Если считывание завершается по ошибке, то возникает ошибка n02 – ошибка считывания компонента, управление передается обработчику ошибок, а обработка запроса прекращается.

2.11.2.2 Удаление экземпляра компонента (xxDelete)

Описание запроса:

Pascal:

```
Function xxDelete( CompName : PString) : Logic;
```

C:

```
Logic xxDelete(PString CompName)
```

Описание аргумента:

CompName – указатель на строку символов, содержащую полное имя компонента.

Назначение – удаляет указанного в параметре CompName компонента из списка компонентов.

Описание исполнения.

1. Если список компонентов пуст или имя компонента, переданное в аргументе CompName, в этом списке не найдено, то возникает ошибка n01 – неверное имя компонента, управление передается обработчику ошибок, а обработка запроса прекращается.

Заметим, что попытка удаления младшей структурной единицы приводит к удалению всего компонента содержащего данную структурную единицу.

2.11.2.3 Запись компонента (xxWrite)

Описание запроса:
Pascal:

Function xxWrite(CompName : PString; FileName : PString) : Logic;

C:

Logic xxWrite(PString CompName, PString FileName)

Описание аргументов:
CompName – указатель на строку символов, содержащую имя компонента.
FileName – имя файла или адрес памяти, куда надо записать компонента.

Назначение – сохраняет в файле или в памяти компонента, указанного в аргументе CompName .
Описание исполнения.

1. Если в качестве аргумента CompName дан пустой указатель, или указатель на пустую строку, то исполняющим запрос объектом является текущий компонент.
2. Если список компонентов пуст или имя компонента, переданное в аргументе CompName, в этом списке не найдено, то возникает ошибка n01 – неверное имя компонента, управление передается обработчику ошибок, а обработка запроса прекращается.
3. Если в качестве аргумента FileName дана строка, первые четыре символа которой составляют слово File, то остальная часть строки содержит имя файла, для записи компонента. В противном случае FileName должен содержать пустой указатель. В этом случае запрос вернет в нем указатель на область памяти, куда будет помещено описание компонента в формате для записи на диск. Если описание не вмещается в одну область памяти, то в текст будет включено ключевое слово Continue, за которым следует четыре байта, содержащие адрес следующей области памяти.
4. Если во время сохранения компонента возникнет ошибка, то генерируется ошибка n03 – ошибка сохранения компонента, управление передается обработчику ошибок, а обработка запроса прекращается.

2.11.2.4 Вернуть имена структурных единиц (xxGetStructNames)

Описание запроса:
Pascal:

Function xxGetStructNames(CompName : PString; Var Names : PRealArray) : Logic;

C:

Logic xxGetStructNames(PString CompName, RealArray* Names)

Описание аргументов:
CompName – указатель на строку символов, содержащую имя компонента или полное имя его структурной единицы.
Names – массив указателей на имена структурных единиц.

Назначение – возвращает имена всех компонентов в списке компонентов или имена всех структурных единиц структурной единицы, указанной в аргументе CompName .
Описание исполнения.

1. Если в качестве аргумента CompName дан пустой указатель, или указатель на пустую строку, то исполняющим запрос объектом является соответствующий программный компонент. В качестве ответа в указателе Names возвращается массив, каждый элемент которого является указателем на *не подлежащую изменению* символьную строку, содержащую имя компонента из списка. После адреса имени последнего компонента следует пустой указатель. Выполнение запроса успешно завершается.
2. Если имя компонента, переданное в аргументе CompName, не найдено в списке компонентов, то возникает ошибка n01 – неверное имя компонента, управление передается обработчику ошибок, а обработка запроса прекращается.
3. Возвращается массив, каждый элемент которого является указателем на *не подлежащую изменению* символьную строку, содержащую псевдоним структурной единицы, являющейся частью структурной единицы, указанной в аргументе CompName. Имена структурных единиц перечисляются в порядке следования в разделе описания состава структурной единицы, имя которой указано в аргументе CompName. Если одна из структурных единиц задана в описании состава несколькими экземплярами, то имя каждого экземпляра возвращается отдельно. После указателя на имя последней структурной единицы следует пустой указатель.

2.11.2.5 Вернуть тип структурной единицы (xxGetType)

Описание запроса:

Pascal:

Function xxGetType(CompName , TypeName : PString; Var TypeId : Integer) : Logic;

C:

Logic xxGetType(PString CompName, PString TypeName, Integer TypeId)

Описание аргументов:

CompName – указатель на строку символов, содержащую полное имя структурной единицы.

TypeName – возвращает указатель на строку символов, содержащую имя структурной единицы, данное ей при описании.

TypeId – одна из предопределенных констант, соответствующая типу структурной единицы.

Назначение – возвращает имя и тип структурной единицы.

Описание исполнения.

1. Если список компонентов пуст или имя компонента, переданное в аргументе CompName, в этом списке не найдено, то возникает ошибка n01 – неверное имя компонента, управление передается обработчику ошибок, а обработка запроса прекращается.
2. В переменной TypeId возвращается тип структурной единицы. Значения предопределенных констант, соответствующих различным типам структурных единиц различных компонентов приведены в табл. 11 и в соответствующих разделах глав, содержащих описания компонентов.
3. Если структурная единица является стандартной, то указателю TypeName присваивается значение пустого указателя. Если структурная единица имеет пользовательский тип (значение аргумента TypeId равно -1), то указатель TypeName устанавливается на строку, содержащую имя, данное указанной в аргументе CompName структурной единице при ее описании.

2.11.3 Запросы на изменение параметров.

К группе запросов на изменение параметров относятся три запроса: xxGetData – получить параметры структурной единицы. xxGetName – получить названия параметров и xxSetData – установить значения параметров структурной единицы.

2.11.3.1 Получить параметры (xxGetData)

Описание запроса:

Pascal:

Function xxGetData(CompName : PString; Var Param : PRealArray) : Logic;

C:

Logic xxGetData(PString CompName, PRealArray* Param)

Описание аргументов:

CompName – указатель на строку символов, содержащую полное имя структурной единицы.

Param – адрес массива параметров.

Назначение – возвращает массив параметров структурной единицы, указанной в аргументе CompName .

Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если список компонентов пуст или имя компонента, переданное в аргументе CompName, в этом списке не найдено, то возникает ошибка n01 – неверное имя компонента, управление передается обработчику ошибок, а обработка запроса прекращается.
3. В массив, адрес которого передан в аргументе Param, заносятся значения параметров. Параметры заносятся в массив в порядке описания в разделе описания статических переменных. Статические переменные, описанные вне описания структурных единиц, считаются параметрами компонента.

2.11.3.2 Получить имена параметров (xxGetName)

Описание запроса:

Pascal:

Function xxGetName(CompName : PString; Var Param : PRealArray) : Logic;

C:

Logic xxGetName(PString CompName, PRealArray* Param)

Описание аргументов:
CompName – указатель на строку символов, содержащую полное имя структурной единицы.
Param – адрес массива указателей на названия параметров.

Назначение – возвращает массив указателей на названия параметров структурной единицы, указанной в аргументе CompName .

Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если список компонентов пуст или имя компонента, переданное в аргументе CompName, в этом списке не найдено, то возникает ошибка n01 – неверное имя компонента, управление передается обработчику ошибок, а обработка запроса прекращается.
3. В массив, адрес которого передан в аргументе Param, заносятся адреса символьных строк, содержащих названия параметров.

2.11.3.3 Установить параметры (xxSetData)

Описание запроса:
Pascal:

```
Function xxSetData( CompName : PString; Param : PRealArray ) : Logic;
```

C:

```
Logic xxSetData(PString CompName, PRealArray Param)
```

Описание аргументов:
CompName – указатель на строку символов, содержащую полное имя структурной единицы.
Param – адрес массива параметров.

Назначение – заменяет значения параметров структурной единицы, указанной в аргументе CompName , на значения, переданные, в аргументе Param.

Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если список компонентов пуст или имя компонента, переданное в аргументе CompName, в этом списке не найдено, то возникает ошибка n01 – неверное имя компонента, управление передается обработчику ошибок, а обработка запроса прекращается.
3. Параметры, значения которых хранятся в массиве, адрес которого передан в аргументе Param, передаются указанной в аргументе CompName структурной единице.
4. Если исполняющим запрос компонентом является интерпретатор ответа (aiSetData), то генерируется запрос SetEstIntParameters к компоненту оценка. Аргументы генерируемого запроса совпадают с аргументами исполняемого запроса.

2.11.4 Инициация редактора компоненты.

К этой группе запросов относится запрос, который инициирует работу не рассматриваемых в данной работе компонентов – редакторов компонентов.

2.11.4.1 Редактировать компонента (xxEdit)

Описание запроса:
Pascal:

```
Procedure xxEdit(CompName : PString);
```

C:

```
void xxEdit(PString CompName )
```

Описание аргумента:
CompName – указатель на строку символов – имя файла или адрес памяти, содержащие описание редактируемого компонента.

Если в качестве аргумента CompName дана строка, первые четыре символа которой составляют слово File, то остальная часть строки содержит имя компонента и после пробела имя файла, содержащего описание компонента. В противном случае считается, что аргумент CompName содержит указатель на область памяти, содержащую описание компонента в формате для записи на диск. Если описание не вмещается в одну область памяти, то допускается включение в текст описания компонента ключевого слова Continue, за которым следует четыре байта, содержащие адрес следующей области памяти.

Если в качестве аргумента CompName передан пустой указатель или указатель на пустую строку, то редактор создает новый экземпляр компонента.

2.12 Описание задачи, используемой для примера

В главах, посвященных описанию предобработчика, задачника, интерпретатора ответа и оценки в качестве примера используется метеорологическая задача. Входная база данных содержит значения следующих показателей:

Температура воздуха – действительное число, изменяющееся от 273 до 393 градусов Кельвина.

Облачность – бинарный признак, означающий наличие (2) или отсутствие облачности (1).

Направление ветра – неупорядоченный качественный признак, принимающий одно из восьми значений: 1 – северный, 2 – северо-восточный, 3 – восточный, и т.д.

Осадки – упорядоченный качественный признак, принимающий следующие значения: 1 – без осадков, 2 – слабые осадки, 3 – сильные осадки.

В качестве ответов требуется предсказать значения тех же показателей через 8 часов. Такая постановка задачи не очень логична, но позволяет продемонстрировать возможности языков описания компонентов нейрокомпьютера.