ЗАО «Е4-СибКОТЭС»

УТВЕРЖДАЮ
Технический директор
_____ С.М. Пасеко
«____» _____ 2011г.

Разработка алгоритмов и программ расчета на ЭВМ нормативных и фактических технико-экономических показателей работы электростанции при отсутствии автоматизированного сбора и обработки первичной информации для оптимизации краткосрочных режимов работы Новосибирской ТЭЦ-3

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПО ВСТРОЕННОМУ РЕДАКТОРУ ФОРМУЛ В ИНФОРМАЦИОННОЙ СИСТЕМЕ «ИСТОК – СБК»

Шифр работы: 17ЭА-10-3

Начальник отдела автоматизированных систем управления и метрологии — Д.А. Крутилин

Ответственный исполнитель-
Инженер — Д.Ю. Колчин

Новосибирск, 2011 г.

СОДЕРЖАНИЕ

1. Описание формул

1.1. Элементы формул

1.1.1. Используемые символы

Множество символов используемых в формулах можно разделить на четыре группы.

1. Символы, используемые для образования ключевых слов и идентификаторов. В эту группу входят прописные и строчные буквы английского и русского алфавитов, а также символ подчеркивания. Следует отметить, что одинаковые прописные и строчные буквы считаются различными символами.

2. Арабские цифры.

3. Знаки нумерации и специальные символы. Эти символы используются с одной стороны для организации процесса вычислений, а с другой - для передачи интерпретатору определенного набора инструкций.

Символ	Наименование	Символ	Наименование
,	запятая	)	круглая скобка правая
.	точка	(	круглая скобка левая
;	точка с запятой	}	фигурная скобка правая
!	восклицательный знак	{	фигурная скобка левая
|	вертикальная черта	<	меньше
/	дробная черта	>	больше
\	обратная черта	%	процент
*	звездочка	&	амперсанд
+	плюс	=	равно
-	минус	"	кавычки
$	знак доллара	^	знак отрицания

4. Управляющие и разделительные символы. К той группе символов относятся: пробел, символы табуляции, перевода строки, возврата каретки, новая страница и новая строка.

1.1.2. Константы

Целая константа – это десятичное, восьмеричное или шестнадцатеричное число, которое представляет целую величину.

Десятичная константа состоит из одной или нескольких десятичных цифр, причем первая цифра не должна быть нулем (в противном случае число будет воспринято как восьмеричное).

Восьмеричная константа состоит из обязательного нуля и одной или нескольких восьмеричных цифр (среди цифр должны отсутствовать восьмерка и девятка, так как эти цифры не входят в восьмеричную систему счисления).

Шестнадцатеричная константа начинается с обязательной последовательности 0x или 0X и содержит одну или несколько шестнадцатеричных цифр (цифры представляющие собой набор цифр шестнадцатеричной системы счисления: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)

Примеры целых констант:

Десятичная константа	Восьмеричная константа	Шестнадцатеричная константа
16	020	0x10
127	0117	0x2B
240	0360	0XF0

Если требуется сформировать отрицательную целую константу, то используют знак "-" перед записью константы (который будет называться унарным минусом). Например: -0x2A, -088, -16 .

Константа с плавающей точкой – десятичное число, представленное в виде действительной величины с десятичной точкой или экспонентой. Формат имеет вид:

[цифры].[цифры] [E|e [+|-] цифры].

Число с плавающей точкой состоит из целой и дробные части и (или) мантиссы. Для определения отрицательной величины необходимо сформировать константное выражение, состоящее из знака минуса и положительной константы.

Примеры: 115.75, 1.5E-2, -0.025, .075, -0.85E2

Строковая константа – последовательность символов (включая строковые и прописные буквы русского и латинского, а также цифры) заключенные в кавычки (") . Например: "Школа N 35", "город Тамбов", "YZPT КОД".

"строка неопределенной длины".

1.1.3. Идентификатор

Идентификатор состоит из одного или нескольких символов и начинается с буквы или символа подчеркивания. Идентификатор может содержать только буквы, цифры, символы подчеркивания. Два идентификатора, для образования которых используются совпадающие строчные и прописные буквы, считаются различными. Например: abc, ABC, A128B, a128b .

Идентификатор не должен совпадать с ключевыми словами, с зарезервированными словами.

Для обозначения параметров применяются идентификаторы параметров. Идентификаторы должны начинаться и оканчиваться знаком доллара ($). Между долларами идет запись кода параметра, представленная в TeXовском виде (о которой будет сказано ниже).

1.1.4. Ключевые слова

Ключевые слова – это зарезервированные идентификаторы, которые наделены определенным смыслом. Их можно использовать только в соответствии со значением известным интерпретатору.

Приведем список ключевых слов

break else return if continue while var

1.1.5. Использование комментариев в тексте программы

Комментарий – это набор символов, которые игнорируются интерпретатором.

Комментарии могут быть многострочные и однострочные.

Многострочные комментарии заключены в специальные символы «/*» и «*/». Например:

```
/* комментарии к программе */
/* начало алгоритма */
 или
/* комментарии  можно  записать в следующем виде,  однако надо
   быть осторожным, чтобы внутри последовательности, которая
   игнорируется интерпретатором, не попались операторы программы,
   которые также будут игнорироваться */
```

Но при этом запрещено использование вложенных комментариев.

Неправильное определение комментариев.

```
/* комментарии к алгоритму /* решение краевой задачи */ */
 или
/* комментарии к алгоритму решения */ краевой задачи */
```

Однострочные комментарии начинаются после двойной дробной черты (//) и заканчиваются концом строки.

Пример:

```
// это однострочный комментарий
```

1.2. Типы данных и их объявление

В формулах может использоваться только два допустимых типа данных – это числа с плавающей точкой (все целые числа приводятся к этому типу) и строки.

Переменные, используемые формулы принимают тип значения, которое этой переменной присвоено.

```
b=42;    // переменная b имеет тип число
b="42"; // а теперь она строка
```

Для использования переменных в формулах, их объявления не требуется. Но нужно обратить внимание, что до первого присваивания переменной значение, она имеет значение 0.

Синтаксис явного объявления переменной приведен ниже:

var *имя_переменной* [=*инициатор*];

Инициатор – задает начальное значение, которое присваивается переменной при объявлении.

В силу того, что интерпретатор формул поддерживает объявление переменных по умолчанию, явное объявление переменных может вам никогда не понадобиться. Но оно может быть полезно в конструкциях, описанных в подпункте «Область видимости переменных».

1.3. Обращение к параметрам внутри формул

Каждый параметр, который будет использоваться в формулах, должен иметь свой уникальный код. Для обращения к этому параметру в формуле пишется этот код, заключенный в знаки доллара ($param1$, D^2_4 и т.д.).

Но стоит помнить, что помимо кода с каждым параметром связано время, за которое он рассчитывается, и период, с которым он рассчитывается.

В общем случае период вызываемого и вызывающего параметров может не совпадать, поэтому введены функции агрегации. Всего таких функций 6: *first*, *last*, *maxp*, *minp*, *avg* и *sum*.

Функция *first*() берет первое значение вызываемого параметра за требуемый период. Кроме того, функция *first*() является функцией по умолчанию. Т.е. запись *first*($param6435$) эквивалентна записи $param6435$.

Функция *last*() берет последнее значение параметра за требуемый период.

Функция *maxp*() выбирает максимальное значение за требуемый период.

Функция *minp*() выбирает минимальное значение за требуемый период.

Функция *avg*() рассчитывает среднее арифметическое значение за требуемый период.

Функция *sum*() рассчитывает сумму всех значений параметра за требуемый период.

Иногда значение одного параметра зависит не от текущего значения другого параметра, а от его прошлого или прошлых значений. В этом случае для получения прошлых значений параметра используется задержка.

функция_агрегации (*параметр*, *задержка*)

где *задержка* – это количество периодов вызывающего параметра, на которое смещается запрашиваемый период. Например, если задержка равна 0, то параметр запрашивается за текущий период, если 1, то за прошлый и т.д.

В данной реализации параметру запрещается запрашивать собственное значение (предыдущее или настоящее).

1.4. Выражения

Некоторые выражения могут рассматриваться как операторы. В данном случае оператор имеет вид 'выражение' ';' – выражение с последующей точкой с запятой.

1.4.1. Арифметические операции

К математическим операциям относятся операции сложения, вычитания, умножения и деления.

Пример простого выражения:

```
(a+10)/a-a*b;
```

Кроме них в синтаксисе формул присутствуют операции инкремента и декремента: ++a, --b.

Они изменяют значение переменной на 1.

Инкремент и декремент имеют префиксную (++a) и постфиксную (a++) запись. Их отличия заключается в том, что при использовании префиксной формы записи значение переменной сначала увеличивается (уменьшается) на 1, а затем возвращается уже измененное значение. В постфиксной же записи значение переменной также изменяется, но в качестве результата выражения берется её начальное значение.

```
a=1; c=++a*10; // значение с равно 20, а a = 2
a=1; c= a++*10; // значение с равно 10, а a = 2
```

1.4.2. Операции присваивания

Для присваивания переменной значения выражения, используется операции присваивания (=).

```
a=5;
b=a*(b-c/42);
```

Выражения типа a=a+5 можно заменить более кратким выражением a+=5. Это относится ко всем двуместным математическим операциям.

```
a*=15; b/=a;
```

и т.д.

1.4.3. Вызов функций

Синтаксис вызова функции:

имя_функции (*список_аргументов*)

где:

имя_функции – идентификатор вызываемой функции;

список_аргументов – аргументы, передаваемые функции. Аргументы должны перечисляться через запятую. Если функция не имеет аргументов, скобки после имени функции все равно пишутся.

Также функция может иметь аргументы по умолчанию (или не обязательные аргументы). Необязательные аргументы могут быть только последними. Т.е. после необязательного аргумента не может идти обязательный и если пропускается один аргумент, то должны пропускаться и все последующие.

1.4.4. Операции сравнения

Операторы сравнения, как это видно из их названия, позволяют сравнивать между собой два значения.

a == b	Равно	истинно, если a равно b.
a != b	Не равно	истинно, если a не равно b.
a < b	Меньше	истинно, если a строго

		меньше b.
a > b	Больше	истинно, если a строго больше b.
a <= b	Меньше или равно	истинно, если a меньше или равно b.
a >= b	Больше или равно	Истинно, если a больше или равно b

1.4.5. Логические операции

! a	Отрицание	истинно, если a ложно.
a && b	Логическое 'и'	истинно, если и a, и b истинно.
a || b	Логическое 'или'	истинно, если или a, или b истинно.

1.4.6. Приоритеты операций

Приоритет операторов определяет, насколько "тесно" связанны между собой два выражения. Например, выражение 1 + 5 * 3 вычисляется как 16, а не 18, поскольку операция умножения ("*") имеет более высокий приоритет, чем операция сложения ("+"). В случае, если операторы имеют одинаковый приоритет, они будут выполняться слева направо. Круглые скобки могут использоваться для принудительного указания необходимого порядка выполнения операторов. Например, выражение (1 + 5) * 3 вычисляется как 18.

В следующей таблице приведен список операторов, отсортированный по убыванию их приоритетов. Операторы, размещенные в одной строке, имеют одинаковый приоритет, и порядок их выполнения определяется исходя из их ассоциативности

Левая ассоциативность подразумевает, что выражение вычисляется слева направо, правая ассоциативность соответственно подразумевает противоположный порядок.

Ассоциативность	Оператор
неассоциативна	++ --
неассоциативна	! – (унарный)
левая	* / %
левая	+ -
неассоциативна	< <= > >=
неассоциативна	== !=
левая	&&
левая	||
правая	= += -= *= /=
левая	,

2. Управляющие конструкции

2.1. Оператор if

Условный оператор имеет синтаксис:

if(*условие*) *оператор*
или
if(*условие*) *оператор1* else *оператор2*,

где:

условие – любое выражение, если его значение равно 0, то выражение считается *ложным*, иначе выражение *истинно*;

оператор и *оператор1* – оператор, который будет выполняться, если *условие истинно*;

оператор2 – оператор, который будет выполняться, если *условие ложно*.

2.2. Оператор цикла while

Синтаксис оператора while:

while(*условие*) *оператор*,

где:

условие – любое выражение, если его значение равно 0, то выражение считается *ложным*, иначе выражение *истинно*;

оператор – оператор, который будет выполняться пока *условие истинно*.

Условие вычисляется каждый раз перед выполнением *оператора*. Если при первом вычислении *условие* окажется *ложным*, то *оператор* не выполнится никогда.

2.3. Оператор break

Синтаксис оператора:

break;

Оператор **break** прекращает выполнение цикла и передает управление оператору, следующему за циклом. Например:

```
while(1)
{
b+=10;
if(b>1000)break;
}
```

Конструкция типа while(1) – это бесконечный цикл, но оператор break позволяет выйти из этого цикла.

2.4. Оператор continue

Синтаксис оператора:

continue;

Оператор **continue** прерывает выполнение оператора в цикле и начинает новую итерацию. Т.е. снова вычисляется условие для оператора цикла.

В операторах **break** и **continue** следует учитывать вложенность циклов.

2.5. Оператор return

Синтаксис оператора:

return *возвращаемое_значение*;

где:
возвращаемое_значение – значение, которое будет возвращено при расчете параметра.

Оператор **return** прекращает выполнение расчета текущего параметра, а результатом расчета принимает *возвращаемое_значение*.

```
if(flag==1) return a;
if(flag==2) return b;
if(flag==3) return c;
```

Если в тексте параметра не используется оператор return, то результатом вычислений считается результат последнего выражения.

2.6. Структурный оператор

В случае, когда необходимо выполнить несколько операций в том месте, где должен идти только один оператор (например, в условном операторе или в операторе цикла) используют структурный оператор.

Синтаксис структурного оператора:

```
{
последовательность_операторов;
}
```

2.7. Область видимости переменных

Локальные переменные объявленные внутри параметра видны только внутри этого параметра и их значение не сохраняется при разных вызовах параметра. Это значит, что если использовать не назначенную переменную, её значение в начале параметра будет всегда 0.

Но существует две встроенные глобальные константы (PI и E).

Но существуют ограничения по видимости и внутри параметра.

Переменные объявленные внутри блока недоступны снаружи этого блока.

```
while(p<10)
{
++p;
c+=p;          // т.к. это первое использование c, то она создается
//и уничтожается при каждом проходе внутри цикла
}
return c; // здесь c равно 0
```

но если перед циклом добавить одну строчку с использованием c, то такая переменная создастся на более высоком уровне вложенности и будет доступна внутри цикла.

```
c=0;           // так же можно использовать var c;
// или любое другое выражение, использующее c.
while(p<10)
{
++p;
c+=p;
}
return c;      // в этом случае c равно 55
```

Также есть возможность использовать разные переменные, имеющие одинаковое имя.

Рассмотрим пример двух вложенных циклов использующие счетчики цикла.

```
while(++c<=20)
{
        var c=0;
        while(c<20)
        {
                …
                ++c;
        }
}
```

3. Справочник стандартных функций

abs(a)	Возвращает абсолютное значение заданного числа.
acos(a)	Возвращает угол, косинус которого равен указанному числу.
asin(a)	Возвращает угол, синус которого равен указанному числу.
atan(a)	Возвращает угол, тангенс которого равен указанному числу.
atan2(a, b)	Возвращает угол, тангенс которого равен отношению двух указанных чисел.
ceiling(a)	Возвращает наименьшее целое число, которое больше или равно заданному числу.
cos(a)	Возвращает косинус указанного угла.
cosh(a)	Возвращает гиперболический косинус указанного угла.
exp(a)	Возвращает *e*, возведенное в указанную степень.
floor(a)	Возвращает наибольшее целое число, которое меньше или равно указанному числу.
log(a)	Возвращает логарифм указанного числа.
logb(a, b)	Возвращает логарифм указанного числа в системе счисления с основанием b.
log10(a)	Возвращает логарифм с основанием 10 указанного числа.
max(a, b)	Возвращает большее из двух указанных чисел.
min(a, b)	Возвращает меньшее из двух чисел.
pow(a,b)	Возвращает указанное число, возведенное в указанную степень. (a^b)
round(a, b)	Округляет значение до ближайшего целого или указанного количества десятичных знаков.
sign(a)	Возвращает значение, определяющее знак числа.
sin(a)	Возвращает синус указанного угла.
sinh(a)	Возвращает гиперболический синус указанного угла.
sqrt(a)	Возвращает квадратный корень из указанного числа.
tan(a)	Возвращает тангенс указанного угла.
tanh(a)	Возвращает гиперболический тангенс указанного угла.
trunc(a)	
truncate(a)	Вычисляет целую часть числа.