|  |
| --- |
| УТВЕРЖДЁН |
| НаноКалькулятор (NanoCalculator) |
| Описание языка |
|  |
| **5 стр.** |
| Носитель данных: электронный документ |
| **04.05.2013** |

В данном документе будут описано назначение и указание общих характеристик языка, его основных областей применения. Так же будет указано описание синтаксиса и семантики базовых и составных элементов языка. Будет указана структура программы и основная точка входа и указаны способы обмена информацией. В конце для опытных программистов можно будет найти указание на встроенное средство отладки программы.

**Содержание**

1. Общие сведения
2. Элементы языка
3. Способы структурирования программы
4. Средства обмена данными
5. Средства отладки программы

**Общие сведения**

Язык предполагает контекстно-свободные грамматики со свойствами LALR(1). LALR(1)- грамматики, являясь подмножеством LR(1)-грамматик, допускают при построении таблиц разбора сокращение общего числа состояний за счет объединения идентичных состояний, различающихся только набором символов-следователей (символов, которые могут следовать после применения одного из правил вывода, если разбор по этому правилу проходил через данное состояние).

Язык предназначен для описания математических выражений и успешно используется для этих целей.

Так как программа написана с использованием утилит lex (flex) и yacc (bison), то фактически она написана на языке соответствующих утилит, со вставками на языке C/С++.

Так же в документе присутствуют описание центральных понятий языка – синтаксиса и лексики.

**Элементы языка**

Язык описывается грамматикой записанной согласно основным принципам РБНФ. В самой грамматике используются лексемы, которые распознаются отдельным образом.

Грамматика языка:

constants

 : PI

 | E

 | CONSTANT

 ;

cast\_expression

 : '(' additive\_expression ')'

 | constants

 ;

max\_func\_expression

 : additive\_expression ',' max\_func\_expression

 | additive\_expression

 ;

min\_func\_expression

 : additive\_expression ',' min\_func\_expression

 | additive\_expression

 ;

multiparam\_functions

 : MAX '(' max\_func\_expression ')'

 | MIN '(' min\_func\_expression ')'

 ;

functions

 : cast\_expression

 | multiparam\_functions

 | SIN '(' additive\_expression ')'

 | COS '(' additive\_expression ')'

 | TG '(' additive\_expression ')'

 | ARCSIN '(' additive\_expression ')'

 | ARCCOS '(' additive\_expression ')'

 | ARCTG '(' additive\_expression ')'

 | SH '(' additive\_expression ')'

 | CH '(' additive\_expression ')'

 | CTH '(' additive\_expression ')'

 | EXP '(' additive\_expression ')'

 | LN '(' additive\_expression ')'

 | LOG '(' additive\_expression ',' additive\_expression ')'

 | ROUND '(' additive\_expression ')'

 | RAND '(' additive\_expression ')'

 | ABS '(' additive\_expression ')'

 ;

unary\_operator\_and\_function

 : '+' functions

 | '-' functions

 ;

multiplicative\_expression

 : functions

 | unary\_operator\_and\_function

 | multiplicative\_expression '\*' functions

 | multiplicative\_expression '/' functions

 | multiplicative\_expression '%' functions

 | multiplicative\_expression '^' functions

 ;

additive\_expression

 : multiplicative\_expression

 | additive\_expression '+' multiplicative\_expression

 | additive\_expression '-' multiplicative\_expression

 ;

translation\_unit

 : additive\_expression

 ;

Лексемы используемые в языке:

CONSTANTS:

({HP}{H}+)

({NZ}{D}\*)

({D}+{E})

({D}\*"."{D}+{E}?)

({D}+"."{E}?)

({HP}{H}+{P})

({HP}{H}\*"."{H}+{P})

({HP}{H}+"."{P})

Где буквами обозначены следующие множества:

D [0-9]

NZ [1-9]

H [a-fA-F0-9]

HP (0[xX])

M [+-]

E ([Ee]{M}?{D}+)

P ([Pp]{M}?{D}+)

"sin"

"cos"

"tg"

"arcsin"

"arccos"

"arctg"

"sh" - гиперболический синус

"ch" - гиперболический косинус

"cth" - гиперболический тангенс

"exp"

"ln" - натуральный логарифм

"log" - логарифм от числа по основанию

"max"

"min"

"round"- округление вещественного до целого

"rand" - псевдо-случайное число

"abs" - взятие модуля от числа

("PI"|"pi"|"Pi"|"pI")

"e"

"("

")"

"-"

"+"

"\*"

"/"

"%"

"^"

","

**Способы структурирования программы**

Как уже было замечено, программа написана на языках под lex/yacc, а потому имеет соответствующую им структуру. Программа состоит из 2-х модулей, по которым генерируются 3 модуля на C/C++, готовых к отдельной компиляции. Компиляция – отдельная, т.к. lex генерирует модуль на языке С, а yacc генерирует модули на языке С++, и поэтому они требуют разных компиляторов.

Модуль написанный под lex состоит из 3-х частей. Первая – это описание множеств символов и вставка на С, имеющая в себе подключение библиотек языка С и объявление внешних переменных. Во второй части описаны правила распознавания лексем. В третьей части, находятся функции необходимые для генерации лексического анализатора утилитой lex.

Модуль написанный под yacc так же состоит из 3-х частей. В первой части описаны лексемы, начальный нетерминал, для лексем и нетерминалов описаны их значения, подключены необходимые библиотеки языка C/C++, а так же объявлены внешние переменные. Во второй части модуля описана грамматика языка и действия, которые необходимо применить в рамках данного проекта. В третьей части описаны необходимые функции на языке С/C++.

Основной функцией, запускающей разбор выражения является функция yyparse().

**Средства обмена данными**

Программа считывает со стандартного потока ввода выражение, которое необходимо вычислить. И записывает результат, или информацию об ошибке на стандартный поток вывода. Из-за написания программы с использованием утилит lex/yacc, связи между различными частями программы в общем случае можно осуществить лишь за счёт глобальных переменных.

**Средства отладки программы**

В программу встроен отладчик, выдающий отладочную информацию о разборе текста на стандартный поток вывода, для его автоматического включения достаточно присвоить глобальной переменной yydebug число 1.