Programming Languages & Translators

(COMS W4115)

Department of Computer Science

Columbia University

Summer 2007

XML Document Manipulation Language (XDML)

Language Reference Manual

Luba Leyzerenok
ll2310@columbia.edu
June 11, 2007

A. Introduction

This manual describes XDML language.

B. Lexical Conventions

B1. Tokens

There are five classes of tokens: identifiers, keywords, string literals, operators and other separators.

B2. Comments

The characters // introduce a comment and comment continues until the end of the line.

B3. Identifiers

An identifier is a sequence of letters, digits and an underscore. The first character must be an either letter or underscore. An identifier may have any length.

B4. Keywords

The following identifiers are reserved for use as keywords, and may not be used otherwise:

`element`	**`for`**	**`if`**
`else`	**`int`**	**`string`**

B5. Constants

There are several kinds of constants:

constant:

integer-constant

string-constant

element-const

B5.1 Integer constant

An integer constant consists of a sequence of digits and taken to be decimal.

B5.2 String constant

A string constant is a sequence of characters surrounded by double quotes as in "…". To represent characters such double quote, newline, backslash and some other characters inside the string constant, the following escape sequence may be used:

Newline	\n
Backslash	\\
Question mark	\?
Less than	\<
Grater than	\>

B5.3 Element constant

An element constant is a sequence of characters surrounded by "<…>".Newlines are not allowed in the element constants.

C. Basic Types

There are several fundamental types. They are integers, strings and elements.
The integer type is a 32bit integer.

D. Expressions

D1. Primary Expressions

Primary expressions are identifiers, constants, strings, or expressions in parentheses.

primary-expression:

identifier

constant

string

(expression)

D2. Postfix Expressions

The operators in postfix expressions group left to right.

postfix-expression:

primary-expression

postfix-expression("string")

postfix-expression.identifier

Function call is a postfix expression, called the function designator, followed by parenthesis containing a string variable or literal.

E. Operators

E1. Unary operator " – "

An operand of the unary " – " operator must have a type **`int`**. The result is negative of the operand.

E2. Additive Operators " + " and " – "

The additive operators + and – group left-to-right. The operands of the + and – must have type **`int`**. The result of the + operator is the sum of the operands. The result of the – operator is the difference of the operands.

E3. Equality operator

The = = operator group left-to-right and evaluates to either 0 or 1. 0 is returned when the operands are equal; 1 otherwise.

E4. Assignment operator

The assignment operator groups right-to-left. The operands must be of the same type.

assignment-expression:

unary-expression assignment-operator assignment-expression

F. Declarators

Declarators are used to specify the data types of an identifier.

F1. Variable declarations

Declarations have the syntax:

Declaration

Type identifier

G. Statements

Statements are executed in sequence and fall into one of the following categories:

statement:

expression-statement

selection-statement

iteration-statement

G1. Expression statement

Expression statement have the form

expression-statement:

expression

G2. Selection statement

Selection statements choose one of several flows of control. In both forms of the if statement, the expression, which must have integer type, is evaluated and if it compares equal to 0, the first substatement is executed.

Selection-statement:

if (expression) statement

if (expression) statement else statement

G3. Iteration statement

Iterations statements specify looping.

iteration-statement:

for(expression; expression; expression;)

In the for statement, the first expression is evaluated once, and thus specifies initialization for the loop. The second expression is evaluated before each iteration and if it becomes equal 0, the loop is terminated. This expression must evaluate to an integer data type. The third expression is evaluated after each iteration and this specifies re-initialization for the loop.

H. Input and Output

XDML program can take a file name, a string, as input. Input of any other type will be discarded.

To read an input file, a file has to be open first. Open call has the following syntax:

open (filename);

Reading a file into a variable of element type, has the following syntax:

read(filename);

To print to standard output, the syntax is as follows:

print (string);

To close a file:

close (filename);