THE PROGRAMMING LANGUAGE NINC

DR. GODFREY C. MUGANDA
NORTH CENTRAL COLLEGE

This paper defines the syntax and semantics of the NINC programming language, an interpreter for which will be implemented by students of CSC 306. The language is a small imperative language with features very similar to those found in languages such as C, Pascal, and Ada95. To keep the implementation of the language doable within a 10 week course, NINC lacks many features found in an industry strength programming language. It does contain enough features, however, so that a person who successfully implements an interpreter for NINC should be able to see how to implement an interpreter for those other languages.

Briefly, NINC contains three types: integer, boolean, and strings. The use of strings is severely curtailed: they can only be printed. Thus for all practical purposes, NINC supports only two types. NINC supports variables and a full complement of operators on the integer and boolean types, as well as procedures and functions capable of accessing both global variables, local variables, and parameters, although only value parameters are supported. Structures and arrays are not supported.

The syntax of the language is described using the following EBNF form, a variant of context free grammars. The nonterminal symbols are given in uppercase, whereas the terminal symbols (tokens) are given in lower case. The tokens written in bold face are reserved words. The empty string can appear as part of some syntactic constructs: it is denoted by the Greek letter ϵ.

PROGRAM ::= VARDECS PROGBODY
VARDECS ::= { VARDECLIST }
VARDECLIST ::= TYPE id {, id };
TYPE ::= **int** | **bool**
PROGBODY ::= **void main** ( ) BLOCKSTMT
BLOCKSTMT ::= "{" STMTLIST "}"
STMTLIST ::= {STMT;}
STMT ::= BLOCKSTMT | IFSTMT | WHILESTMT | ASSIGNSTMT
STMT ::= SWITCHSTMT | INPUTSTMT | OUTPUTSTMT
STMT ::= **break** | **continue** | **exit** | ϵ
IFSTMT ::= **if** (EXPR) STMT [**else** STMT]
WHILESTMT ::= **while** (EXPR) STMT
SWITCHSTMT ::= **switch** (EXPR) SWITCHBODY
SWITCHBODY ::= "{" {CASESTMT}[**default** : STMTLIST] "}"
CASESTMT ::= **case** CASELABEL : STMTLIST
ASSIGNSTMT ::= id = EXPR
INPUTSTMT ::= **cin** >> id { >> id }
OUTPUTSTMT ::= **cout** << OUTPUTEXPR { << OUTPUTEXPR }

OUTPUTEXPTR ::= EXPR | **endl** | string
CASELABEL ::= number | **true** | **false**
EXPR ::= SIMPLEEXPR [RELOP SIMPLEEXPR]
SIMPLEEXPR ::= TERM { ADDOP TERM }
TERM ::= FACTOR {MULTOP FACTOR }
FACTOR ::= -FACTOR | ! FACTOR
FACTOR ::= number | **false** | **true**
FACTOR ::= id | (EXPR)
RELOP ::= == | < | <= | > | >= | !=
MULTOP ::= $*$ | / | % | &&
ADDOP ::= + | - | ||

A NINC program consists of a possibly empty global variable declaration part, followed by a sequence of procedures or functions. Here is the traditional "Hello World" program written in NINC.

```
void main( )
{
  cout << "Hello World";
}
```

Here is a program that reads in two numbers and prints their sum:

```
int a, b;
void  main( )
{
  cout << "Enter 2 integers";
  cin >> a; cin >> b;
  cout << "The sum is : " << a + b << endl;
}
```

Strings will be delimited by double quotes.