MiniC Language Manual

Sanjana Sunil (20171027)

Introduction

MiniC is a programming language that supports the following features:

1. **Datatypes:** Signed and unsigned integers, long, char, bool, 1D and 2D arrays. Arithmetic operations: add, mul, sub, div, mod.
2. **Boolean operators:** and, or, not
3. **Control statements:** if-then, if-then-elif, if-then-elif-then-else, for loop, while loop, ternary operator
4. **Functions:** Call by value parameter passing mechanism, call by reference for arrays, recursion support, I/O routines like printing
5. **Miscellaneous:** Comments, break and continue statements

The keywords of this language are int, uint, long, bool, char, true, false, main, void, func, while, for, if, break, continue, return, input, print.

MiniC is case sensitive and keywords begin with lowercase letters. In MiniC, lexical tokens are separated by whitespace which could include spaces, newlines, tabs etc.

Grammar

The following is the grammar for MiniC language. Both microsyntax and macrosyntax are defined. A couple of notations are used.

1. Non-terminals begin with an upper case letter.
2. Regular expressions have been bolded and are used to define microsyntax.
3. The non-terminal 'Program' (specified below) is the start symbol of the CFG.
4. The symbol '|' denotes 'or' in the CFG, unless it is used in regex (in which case it will be bolded).

Regex notations include the following:

1. **(x)*** denotes zero or more occurrences of x.
2. **[a-zA-Z]** denotes any one English letter.
3. **(x)?** denotes zero or one occurrence of x.
4. **.*** denotes zero or more occurence of any character.
5. **(x | y)** denotes either x or y.
6. **(x)⁺** denotes one or more occurrences of x.

Program → **(Comment)* (VarDecl)* (FunctionDecl)*** func int main() Block

IntLiteral → **[1-9][0-9]*** | 0

LongLiteral → **[1-9][0-9]*** | 0

UintLiteral → **[1-9][0-9]*** | 0

BoolLiteral → true | false

CharLiteral → ‘**[.\n\t\r]**’

Literal → IntLiteral | UintLiteral | BoolLiteral | CharLiteral | LongLiteral

Type → uint | int | bool | char | long

Arithop → + | - | * | / | %

Relop → < | > | >= | <= | == | !=

Condop → && | ||

Binop → Arithop | Relop | Condop

Id → **[a-zA-Z_][0-9a-zA-Z_]***

Variable → Id
| Id [Expr]
| Id [Expr] [Expr]

VarDecl → Type Variable **(, Variable)*** ;
| Type Id = Expr **(, Id = Expr)*** ;

Block → { **(Statement)*** }

FunctionArgument → Type Id
| Type Id [ ]
| Type Id [ ][ ]

FunctionDecl → func **(Type | void)** Id (**(FunctionArgument)? (, FunctionArgument)***)
Block

FunctionCall → Id (**(Expr)? (, Expr)***)

Expr → Variable
| Literal
| FunctionCall
| Expr Binop Expr
| - Expr
| ! Expr
| (Expr)
| (Expr ? Expr : Expr)

Statement → VarDecl
| Variable = Expr ;
| FunctionCall ;
| if (Expr) Block **[elif** (**Expr**) **Block]* [else Block]?**
| for (**((Type)? Variable = Expr)?** ; **(Expr)?** ; **(Variable = Expr)?**) Block
| while (Expr) Block
| break ;
| continue ;
| return **(Expr)?** ;
| Block
| IoStatement
| Comment

Comment → // **.*** | /* **.*** */

IoStatement → input(Variable **(, Variable)***);
| print(**(" .* " | Expr) (, (" .* " | Expr))***);

Semantics

Semantic checks

1. Check if the function returns the same type that was defined in the declaration.
2. Check if indexing of an array is an unsigned integer between 0 and n-1.
3. Check for scope, i.e. a variable has scope only in the block it is defined in.
4. Check for division by zero errors.
5. Check types during expression evaluation e.g. a char cannot be added to an int.

Program

The program consists of optional variable declarations or function declarations that are global to the program. Additionally, it consists of a compulsory method called 'main' from where execution begins. Similar to C or C++, ‘main’ is a function that returns type int which denotes exit status. This method consists of further statements which are executed.

Datatypes

There are 5 main types of datatypes that MiniC supports - int, uint, long, char, bool.

- int includes 32 bit signed integers from -2147483648 to +2147483647.
- uint includes 32 bit unsigned integers from 0 to 4,294,967,295.
- long includes 64 bit signed integers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
- char is denoted in single quotes and includes any ASCII character except single quote and backslash (these must be escaped using an additional backslash). Other characters include ‘\n’ to denote newline, ‘\t’ to denote tab and ‘\r’.
- bool is denoted by 2 values, true or false.

In addition, there are 1D and 2D arrays supported for each of the above datatypes. These can be declared as <name of type> <id> [n] or <name of type> <id> [n] [m], e.g. int arr[10]. Arrays are fixed size and are indexed from 0 to n-1 where n is the size of the array. Variable names could include lowercase letters, uppercase letters or underscore.

Operators

- Arithmetic operators - add, sub, mul, div, mod, unary minus for int
- Relational operators - less than, greater than, less than or equal to, greater than or equal to for int and char, equality, inequality for bool too
- Conditional operators - logical and, or, not for bool, int

Functions

Functions could be any of the types mentioned previously or it could be void type (returns nothing). Whenever a function is to be declared, it is preceded by the keyword ‘func’. Then the type is specified, and the ID of the function. In parenthesis, the type of the arguments along with their name is specified. It returns an expression. When a method is called, it takes as arguments expressions. The language supports pass by value function calls, except for arrays which are passed by reference.

Expressions

Expressions consist of anything that can be evaluated, such as variables, function calls, addition of sub-expressions etc. Operator precedence is unary minus, logical not, mul, div, mod, add, sub, relational, equality, conditional and, conditional or, ternary operator.

Statements

Statements consist of a full line in the program and are generally terminated with a semicolon (except in the case of a block). It consists of control statements that are similar to C. return statements, variable declaration and assignment, comments, print and input statements etc.