COMPILERS
LANGUAGE SPECIFICATION

Aayush Goel
20171188

➔Lexical Consideration

- All keywords are lowercase, and identifiers are case-sensitive.
- Reserved Keywords -
  - **Data types:-** int, uint, char, bool
  - **Boolean Operations:-** and, or, not, true, false
  - **Control Statement:-** if, else, cond?opt1:opt2, for, while, return, void, break
  - **Symbols:-** +, -, *, /, %, =, !=, >=, <=, >, <, ==, +=, -=, *=, /=, %=
- int and uint are 32 bits in size
- Character encoding is in ASCII
- Identifiers are separated by whitespaces or tabs.
- String literals are composed of characters - <char>* and denoted by “string”.
- Comments start with // and end with end-of-line.

➔Data Types And Initialisation

- int: 32-bit signed integer - globally initialized as 0
- uint: 32-bit unsigned integer - globally initialized as 0
- char: ASCII character - globally initialized as \0
- bool: true/false - globally initialized as false

➔Location

We have two kinds of locations: local/global scalar variables and global array elements. Each location has a type. Locations of types int and boolean contain integer values and boolean values, respectively. Locations of types int [N] and boolean [N] denote array elements.

➔Scope

- There are namely two scopes:
  - Global
  - Method
- All identifiers declared in the global scope can be accessed anywhere in the program.
- Identifiers declared inside a method are restricted to that method. Every time a method is called the identifiers in the method scope are re-initialized.

➔Expressions

- Expressions follow the normal rules for evaluation. In the absence of other constraints, operators with the same precedence are evaluated from left to right. Parentheses can be used to override normal precedence.
- Integer literals evaluate to their integer value. Character literals evaluate to their integer ASCII.
- The arithmetic and relational operators have usual precedence and meaning.
- The equality operators, == and != are defined for int and boolean types only and can be used to compare any two expressions having the same type.
- The result of a relational operator or equality operator has type boolean.

➔Control Statements

- Ternary Operator: An if-else block written in one statement with structure as <expr>?opt1:opt2, where <expr> is evaluated as true(where opt1 is executed) or false(where opt2 is executed)

- While Loop: It starts with the keyword ‘while’ followed by a set of parenthesis ‘()’ which contains a <expr> which is necessary to result in bool value when evaluated, and cannot be empty. After the parenthesis, there is a block of statements enclosed in ‘{}’, which is called the body of the statement.
- If-else Statement: It starts with the keyword ‘if’ followed by a set of parenthesis ‘()’ which contains a <expr> which is necessary to result in bool value when evaluated, and cannot be empty. The <expr> is evaluated, if it results to true, the true branch is evaluated otherwise the else branch is evaluated. The else branch is not mandatory to follow the if branch, it can be absent.
- For Loop: The first <expr> is the initial value of the loop index variable which can be empty, which is executed once before the execution of the loop and the second <expr> is the test <expr> of the loop which is evaluated as a bool statement and can’t be empty, which if true, the loop will move to the next iteration, else the loop will stop. The block statements are executed in every iteration.

➔Methods

- Parameters are assigned the value of the <expr> written in the place of the parameter in the function call statement and are treated as local variables in the scope of the function
- After a function is called, the block in the function definition is executed, and depending on the type of return value of the function, a value is returned which replaces the function at the call location.

- A function can also be called in the definition of itself(recursion), wherein another copy of the function is created with the new parameter values.
- Every function other than main must be declared at the start of the program(function statement without function body)
- The return value of the function is ignored if it is not part of an expression

➔Grammar

- **Notation**

Representation	Meaning
<x>	x is a non-terminal
x	x is a terminal
[x]	zero or one occurrence of literal x
x*	zero or more occurrence of literal x
x^{+}	comma separated list of one or more x
|	different alternatives

- **Rules**

lexeme_type	**Symbolic Representation**
<program>	**class Program** ‘{’⟨field_declr⟩* ⟨method_declr⟩* ‘}’
<field_declr>	⟨type⟩‘{’ ⟨id⟩ | ⟨id⟩ ‘[’ ⟨int_literal⟩ ‘]’ ‘}’;
<method_declr>	{<type> | void} <id> ([{<type> <id>}$^{+}$,])

	<block>
<block>	‘{’ <var_declr>* <statement>* ‘}’
<var_declr>	<type> <id>$^{+}$;
<type>	**int** | **bool** | **char**
<statement>	⟨location⟩ ⟨assign_op⟩ ⟨expr⟩ | ⟨method_call⟩ | if (⟨expr⟩) ⟨block⟩ [else ⟨block⟩] | for ⟨id⟩ = ⟨expr⟩, ⟨expr⟩ ⟨block⟩ | return [⟨expr⟩] | break | continue | ⟨block⟩
<assign_op>	= | += |− = |∗ = | /= | %=
<method_call>	<method_name> ([<expr>$^{+}$])
<location>	<id> | <id> ‘[’<expr>‘]’
<method_name>	<id>
<expr>	⟨location⟩ | ⟨method_call⟩ | ⟨literal⟩ | ⟨expr⟩ ⟨bin_op⟩ | −⟨expr⟩ | !⟨expr⟩ | (⟨expr⟩)
<bin_op>	⟨arith_op⟩ | ⟨rel_op⟩ | ⟨eq_op⟩ | ⟨cond_op⟩
<arith_op>	+ | − | ∗ | / | %
<rel_op>	< | > | <= | >=
<eq_op>	== | !=
<cond_op>	&& | ||

<id>	<alpha> ⟨alpha_num⟩*
<alpha>	[a-zA-Z]
<alpha_num>	<alpha> | <digit>
<digit>	[0-9]
<hex_digit>	<digit> | [a-fA-F]
<int_literal>	<decimal_interal> | <hex_literal>
<decimal_interal>	<digit> <digit>*
<hex_interal>	0x <hex_digit> <hex_digit>
<bool_literal>	true | false
<char_literal>	‘<char>’
<string_literal>	“<char>*”

➔I/O Routines

- The inbuilt function getInt() | getChar() are used to take input, which returns a value assigned to an identifier.
- The inbuilt function print(⟨id⟩ ∗ ,) is used to display output, where user has to specify the identifier, which needs to be output, If no identifier is provided, new line is displayed.
- The inbuilt functions readLines(⟨id⟩) is used to read data from files, where the identifier represents the filename string.
- The inbuilt functions writeLines(⟨file_id⟩, ⟨data_id⟩, ⟨mode⟩) is used to write data to files, where the identifiers represents the filename string, the char array to be written to the file and the mode of operation {read, write, append} respectively.

➔Semantic Checks

- No identifier is used before it is declared in the scope.
- The same identifier should not be declared in scope.
- Any program should have a function main without any parameters where the execution starts.
- The number of arguments and their types in a function definition should match with the parameters passed in the function call.
- The size of an array should be greater than 0.
- The return value of a function should be the same as in the declaration.
- Arrays should be declared in the global scope.
- <id> [<expr>] should have id as array variable and expr at int_literal.
- Expression in if statement should have bool type.
- Operands of <arith_op> and <rel_op> must have int data type.
- Operands of <eq_op> type should have the same type.
- Break and continue statements must be inside the body of for loop.