CS315 Project 1 Report

Date: 08.03.2020
Language name: zetz
Group number: 22

Group members:	**Full name**	**Student ID**	**Section**
	Halil İbrahim Karakoç	21702419	1
	Suleyman Rahimov	21701671	1
	Uğur Erdem Seyfi	21801744	1

Grammar of the Language

Note: Since we did not want to write all of the unicode characters one by one, we used another meta character μ to denote any unicode character alongside ε which denotes the empty string.

Our program starts with the following rule:

<program> → <statement_list>

1. Statements

<statement_list>→ <statement> | <statement_list> <statement>

<statement> → <declaration_statement> | <assignment_statement> | <if_statement>

| <while_statement> | <do_while_statement> | <for_statement>

| <print_statement> | <set_property_statement> | <delete_statement>

| <function_call_statement> | <comment> | <empty_statement>

<empty_statement> → ε

1.1 Declaration/Assignment Statements

<declaration_statement> → <declaration_expression>;

<assignment_statement> → <assignment_expression>;

<print_statement> → print(<expression>);

1.2 If Else Statements

<if_statement> → if(<boolean_expression>) { <statement_list> } <else_if> <else>?

<else_if> → ε | <else_if> else if(<boolean_expression>) { <statement_list> }

<else> → else { <statement_list> }

1.3 Loops

<while_statement> → while (<boolean_expression>){ <statement_list> }

<do_while_statement> → do { <statement_list> } while (<boolean_expression>);

<for_statement> → for (<declaration_expression>; <boolean_expression>;
<assignment_expression>) { <statement_list> }

1.4 Methods (Functions)

<parameter_list> → <type> <identifier> | <parameter_list> , <type> <identifier>

<arguments> → ε | <parameter_list>

<function_declaration_statement> → <type> <identifier> (<arguments>){ <statement_list> }

<function_call_statement> → <function_expression>;

1.4.1 Special Method Statements

<set_add_property> → .add(<expression_list>)
<set_remove_property> → .remove(<expression_list>)
<set_size_property> → .size()
<set_property_statement> → <identifier>.<set_add_property>;
| <identifier>.<set_remove_property>;
| <identifier>.<set_size_property>;
<delete_statement> → delete <identifier>;

1.5 Comments

<uni_comment> → ε | <unicode_sentence>

<uni_comment_w_newLine> → <uni_comment>

| <uni_comment_w_newLine> \n <uni_comment>

<comment> → /* <uni_comment_w_newLine> */

| // <uni_comment>

2. Expressions

<expression> → <boolean_expression> | <arithmetic_expression> | <string_expression>

| <char_expression> | <set_expression> | <assignment_expression>

| <function_expression>

2.1 Set Operation Expressions

<set_expression> → <set> | <set_t>

<set_t> → <set_term> | <set_term> \ <set_term>

<set_term> → <set_op>(<set_terms>) | (<set_expression>)

<set_terms> → <set_t>,<set_t> | <set_t>,<set_terms> | <set_expression>

2.2 Boolean logical expressions

<set_logical_expression> → <set_expression> <logical_op> <set_expression>

<real_logical_expression> → <arithmetic_expression><logical_op><arithmetic_expression>

<boolean_expression> → <boolean_term>

| <boolean_expression> <condition_op> <boolean_term>

<boolean_term> → !<boolean_expression> | (<boolean_exprression>)

| <boolean_logic_operation>

<boolean_logic_operation> → <eqq_expression> | <real_logical_expression>

| <set_logical_expression>

<eqq_expression> → <eq_operand> <eqq_op> <eq_operand>

<real_logic_expr> → <real> <logic_op> <real>

<eq_operand> → <boolean_expression> | <identifier> | literal | <function_expression>

2.3 Arithmetic expressions

<arithmetic_expression> → <arithmetic_expression> <add_subs_op> <term> | <term>

<term> → <factor> | <term> <mult_div_op> <factor>

<factor> → (<arithmetic_expression>) | <identifier> | <real>

2.4 String & Char Expressions

<string_expression> → <string> | <identifier>

<char_expression> → <char> | <identifier>

2.5 Assignment Expression

<assignment_expression> → <identifier> = <expression>

<declaration_expression> → <type> <identifier> | <type> <assignment_expression>

2.6 Function expression

<function_expression> → <identifier> (<input_parameters>)

<input_parameters> → <expression> | <input_parameter>, <expression>

3.1 Non-logical Operators

<set_op> → union | intersect

<add_subs_op> → + | -

<mult_div_op> → * | /

3.2 Logical Operators

<logical_op> → < | > | <= | >=

<condition_op> → & | **|**

<eqq_op> → == | !=

4. Literals

<literal> → <float> | <int> | <unsigned_int> | <set> | <string> | <char> | <boolean>

<type> → char | string | int | float | boolean | set | void

<char> = ‘ <unicode_char> ‘

<string> = “ <unicode_sentence> “

<unsigned_int> → <digit> | <digit> <unsigned_int>

<sign> → - | +

<int> → <sign> <unsigned_int>

<float> → <int> . <unsigned_int>

<real> → <int> | <float>

<boolean> → true | false

<set> → { ε | <expression_list> }

<expression_list> → <expression> | <expression> , <expression_list>

5. Characters

<uppercase_letter> → A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

<lowercase_letter> → a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z

<letter> → <uppercase_letter> | <lowercase_letter>

<digit> → 0|1|2|3|4|5|6|7|8|9

<identifier_symbols> → <letter> | <digit> | _ | $

<identifier> → <letter> | <identifier> <identifier_symbols>

<unicode_sentence> → μ | μ <unicode_sentence>

<reserved_word> boolean | char | const | do | else | float | for | if | int | return | set | void | while

Example Program

```
/*
  A test program to test the various components of the language.
 */

/*
        This part tests the set operations in the language.
*/

set set1 = { 1, 2, 7 };                 // set of integers
set set2 = {1,7,2};                     // set of integers
set set3 = { "Mesut", "Husnu", "Riza" };       // set of strings
set set4 = { set1, set2 };              // set of sets
set set5 = { true, true, true, false };     // set of booleans
set set6 = { true, false };
set stringSet2 = input();               // storing the input in the stringSet2 variable


if ( set1 == set2 ) {
        print( "Two sets are equal" );
}

else {
        print( "Two sets are not equal" );
}


/*
Note: the sets are considered equal when they have the same elements. The number of
elements does not matter.
*/

if ( set5 == set6 ) {
        print( "Given boolean sets are equal" );
}

else {
        print( "Given boolean sets are not equal" );
}

set1.add( 87 );            // adding an element to a set
```

```
set2.remove( 7 );        // removing an element from the set

set set7 = union( set1, set2);
set set8 = intersection( set1, set2 );

if ( set1 > set2 ) {
        print( "set1 is a superset of set2" );
}

if ( set1 < set8 ) {
        print( "set1 is a subset of set8" );
}

delete set1;             // to remove all elements from the set and setting it equal to {}(empty
set)
set2.add(input());       // adding the elements, that user entered, to the set

delete set6;             // set 6 is now equal to empty set

set2 = set2 \ set1;   // set subtraction

/*
        This part tests the set decision statements and loops in the language.
*/

int num1 = 3;
int num2 = 8;
int num3 = 27;
int num4 = -72;
float realNum = 0.99;
char character = 'y';
string str = "Mesut";   // string declaration

if ( ( num1 != num2 ) && ( num1 == num3 ) ) {
        print ("num1 is not equal to num2 and is equal to num3" );
}

else if ( ( num2 == num4 ) || ( num3 == num4 ) ) {
        print ("num2 is equal to num4 or num3 is  equal to num4");
}

else {
```

```
   if ( ( num3 > 0 ) && (! (num3 >= 8)) ) {
         print("num3 is greater than num0 and num3 is not greater than or equal to 8");
   }
}

// incrementing num2 by i multiple times using a for loop
for ( int i = 0; i < num3; i = i + 3 ) {
         num2 = num2 + i;
}

// incrementing num2 until it reaches 1000 using while loop
while ( num2 < 1000 ) {
         num2 = num2 + 10;
}

// incrementing num4 until it is zero using do-while loop
do {
   num4 = num4 + 1;
} while ( num4 < 0 );



/*
         This part tests the function definition and implementation in the language.
*/

int sum( int a, int b ) {
         return a + b;
}

int number1 = 7;
int number2 = 12;

int number3 = sum( number1, number2 );   // number3 is 19 after this method call
```

6. Information about language components

6.1 Variable declarations

The user can declare variables by specifying its type and name. The language requires users to specify the type in order to simplify type checking and increase the reliability of the language. Variable names (identifiers) should begin with a letter and can include digits, letters and underscore character. Also, no reserved word can be a variable.

The types in the language are: int, string, char, set, boolean and float. Set is a special type that can include all the other types inside it. A set can include elements of different types. Declaration of set of sets is also possible in the language. Some examples are given below:

```
int sum = 7;
float realNumber;
string str = “Husnu”;
set mixedSet = { 5.3, 1, ‘a’, “Mesut”, {1, 3, 7}, 0.7 };
char character = ‘c’;
```

6.2 Comments

There are two types of comments in the language: single-line comments and multi-line comments. Single-line comments are initiated by //. Multi-line comments are initiated with /* and ended with */. Comments are ignored by the language. Comment examples are below:

```
// Show must go on

/*
        I am just a comment
        Nobody loves me.
        He is just a comment
        Ignored by the language.
        ( Bohemian Rhapsody reference :) )
*/
```

6.3 Function declarations and calls

Functions can be declared by specifying their return type, name and arguments. User can later call this method by providing the parameters. The method will return the value and the user can use it independently. An example of function declaration and call is given below:

```
int subtract( int number1, int number2 ) {
        return ( number1 - number2 );
```

```
}
int num1 = 5;
int num2 = 3;
int num3 = subtract( num1, num2 );  // value 2 will be placed in num3
```

6.4 Loops

Loops are an integral part of the language. We have declared 3 different loops with different functionalities: while loop, for loop and do-while loop. While loop can be declared by specifying its condition in brackets after writing the keyword “while”. Curly brackets are used to indicate the scope of the loop. Do-while loop is similar and it can be declared by writing the keyword “do”, then adding the statements in between curly braces and writing keyword “while” with its condition inside brackets and semicolon at the end of line to indicate that the statement ended. For loops are very different. Their general structure is a little similar to while loops, but the user can specify the loop control variable declaration, condition of the loop and loop variable changes inside brackets. Examples are below:

```
int num1 = 7;
int num2 = 9;
int num3 = 11;
int sum1 = 0;

for ( int i = 0; i < num1; i = i + 2 ) {
        sum1 = sum1 + i;
}

while ( num2 >= 0 ) {
        num2 = num2 - 1;
}

do {
        num3 = num3 + 4;
} while( num3 < 100 );
```

6.5 Decision statements

The language has the following decision statements: if and if-then-else. After writing the keyword “if”, user can specify the condition using a boolean expression. Very complex boolean operators can be written in the language using boolean operators and parentheses. Examples are below:

```
int num = 5;
int num2 = 7;
int num3 = 12;
```

```
if ( num < 6 ) {
        print( “number is less than 6” );
}

else if(  num < 4) {
        print( “number is less than 4” );
}

else {
        print( “number is greater than 6” );
}

if ( num1 == num2 == num3 + 3 ) {
        print( “equal”);
}
```

6.6 Set operations

Set is a type in this language. It can store the elements of all types including itself. A single set can involve elements of multiple types to provide flexibility. Sets can be declared using the “set” keyword and identifier. In the initialization statement, elements, that should be in the set, can be specified in between curly brackets. User can add and remove elements from the set using add() and remove() functions. User can determine whether a set is a superset or a subset of another set using > and < respectively. Union and intersection set of the sets can also be taken using union() and intersect() functions respectively. “delete” keyword can be used to remove all elements from a set and setting it to empty set. Equality operator can be used to check the equality of two or more sets. A\B operation can be used to find the set obtained by removing the elements of set B from set A.

Input() function is probably the most interesting function in the language. It lets the user input something from the console and it recognizes the type of input. User can put the input into a variable of respective type. The same is true for sets as well. Examples are given below:

```
set intSet1 = {1, 3, 5, 6, 8};    // using initializer to initialize a set
set intSet2 = {1, 1,  3, 5, 5, 6, 6, 6, 8, 8, 8};
set floatSet1 = { 0.7, 2.7, 1.9};
set stringSet1 = { “Husnu”, “Mesut”, “Sherlock”};
set stringSet2 = input();    // storing the input in the stringSet2 variable

if ( intSet1 > intSet ) {
        print(“intSet1 is a superset of intSet2”);
}
```

```
if ( intSet1 < intSet ) {
        print("intSet1 is a subset of intSet2");
}

stringSet1.add( "Selin" );    // adding an element to the set
floatSet1.remove( 0.7 );      // removing an element from the set

set unionIntSet = union(intSet1, intSet2);  //getting the union of the sets
set intersectionIntSet = intersect(intSet1, intSet2 ); //intersection of sets

floatSet1.add( input() );   // adding the input element to the set
delete floatSet1;    // floatSet1 is now an empty set.

intSet1 = intSet2 \ intSet1;
```

6.7 Arithmetic operations and expressions

The language supports arithmetic operations like addition, subtraction, multiplication, division. Users can also use brackets to change the order of operations. Assignment statement can be used to store the result of the expressions in the variables. Assignment operator can also be used to give an initial value to the variable in declaration. The language supports complex arithmetic expressions. Examples are given below:

```
int num1 = 3;
int num2 = 5 +10;
int num3 = num2 = num1;
float realNum3 = 2.7;
num1 = num2 + 2;
num2 = num1 / 2;
realNum3 = realNum3 * 3;
num2 = num1 - num2;
int a = ( sub(5, 3) * 2) / 5 + 3 + 7;
```

7. Language evaluation

Our language excels at readability and reliability. However, it is hard to get the balance right, since readability and writability are somewhat inversely related with each other. That is why, writability takes a slight hit.

8. LEX CODE

```
%option main

// Characters
digit [0-9]
uppercase [A-Z]
lowercase [a-z]
letter ({uppercase}|{lowercase})
dot \.
semicolon \;
l_curly \{
r_curly \}
L_bracket \(
r_bracket \)

// Literals
char \'.\'
string \"(.)*\"
unsigned_int {digit}+
int [+-]?{unsigned_int}
float {int}?\.{unsigned_int}
boolean true|false
```

```
// Type keywords
int_type int
float_type float
boolean_type boolean
set_type set
char_type char
string_type string

// Other keywords
delete_keyword delete
return_keyword return
void_keyword void
if if
else else
do do
for_loop for
while_loop while

// Identifiers
identifier_symbols ({letter}|{digit}|_|$)
identifier {letter}{identifier_symbols}*

// Assignment operator
```

eq =

// Set operators

union union

intersect intersect

set_substraction \\

// Arithmetic operators

plus \+

minus -

divide \/

multiply *

remainder %

// Logical operators

greaterorequal \>=

lessorequal \<=

greaterthan \>

lessthan \<

isEqual ==

notEqual !=

// For comments

comment_start \/*

```
comment_end \*\/

single_line_comment \/\/


%%


{dot} printf(" DOT ");

{if} printf(" IF ");

{else} printf(" ELSE ");

{do} printf(" DO ");

{for_loop} printf(" FOR ");

{while_loop} printf(" WHILE ");

{delete_keyword} printf(" DELETE ");

{return_keyword} printf(" RETURN ");

{void_keyword} printf(" VOID ");

{int_type} printf(" INT_TYPE ");

{float_type} printf(" FLOAT_TYPE ");

{boolean_type} printf(" BOOLEAN_TYPE ");

{set_type} printf(" SET_TYPE ");

{char_type} printf(" CHAR_TYPE ");

{string_type} printf(" STRING_TYPE ");


{single_line_comment} printf(" SINGLE_LINE_COMMENT ");

{comment_start} printf(" COMMENT_START ");

{comment_end} printf(" COMMENT_END ");
```

```
{boolean} printf(" BOOLEAN ");

{union} printf(" UNION_OP ");

{intersect} printf(" INTERSECT_OP ");

{set_substraction} printf(" SET_SUBSTRACTION_OP ");

{char} printf(" CHAR ");

{string} printf(" STRING ");

{float} printf(" FLOAT ");

{unsigned_int} printf(" UNSIGNED ");

{int} printf(" INT ");


{l_curly} printf(" LC ");

{r_curly} printf(" RC ");

{L_bracket} printf(" LB ");

{r_bracket} printf(" RB ");


{isEqual} printf(" ISEQUAL_OP ");

{notEqual} printf(" NOTEQUAL_OP ");

{greaterorequal} printf(" GREATEREQUAL_OP ");

{lessorequal} printf(" LESSEQUAL_OP ");

{greaterthan} printf(" GREATERTHAN_OP ");

{lessthan} printf(" LESSTHAN_OP ");

{eq} printf(" ASSIGNMENT_OP ");

{plus} printf(" ADDITION_OP ");
```

```
{minus} printf(" SUBSTRACTION_OP ");

{divide} printf(" DIVISION_OP ");

{multiply} printf(" MULTIPLICATION_OP ");

{remainder} printf(" MOD_OP ");


{semicolon} printf(" SEMICOLON ");


{identifier} printf(" IDENTIFIER ");
```