HACETTEPE UNIVERSITY

BBM 301 – PROGRAMMING LANGUAGES

PROJECT 1-REPORT

LEX

Student's Name: Zeynep Işıl İSKENDER - - - Ali Utku ÜNLÜ

Student's ID : 21228415 - - - 21228817

Instructors

Nazlı İKİZLER CİNBİŞ | Pınar DUYGULU ŞAHİN | Nebi YILMAZ |Gültekin IŞIK

17 November 2016

Our new programming language is designed to read special datas(like web page's link or plate number etc.) easier. It's not much different than popular programming languages(like C, Java and C++) but with new functions and data types it is a lot of easier to read these datas.

1.Explanations of Data Types

1.1. Basic Types

- Boolean: (FALSE|TRUE)
- String: Combination of characters and numbers.
- Integer: Combination of Numbers.
- Float: Saves floating points.
- Char: Combination of characters.
- Identifier: Start with character and continue with character or numbers.

1.2. Complex Types

- Array: Keeps more than one data given type.

2.Operators

2.1.Arithmetic Operators

- Equal: x=y
- Addition: x+y
- Substraction: x-y
- Multiply: x*y
- Divide: x/y

2.2.Logical Operators

- And: x&&y
- Or: x||y

2.3.Compund Assignment Operators

- Addition Assignment: x+=y
- Substraction Assignment: **x-=y**
- Multiplication Assignment: x*=y
- Division Assignment: x/=y

2.4.Comparison Operators/ Reletional Operators

- Equal to : x==y
- Not equal to: x!=y
- Greater than: x>y
- Less than x<y
- Greater than or equal to: x>=y
- Less than or equal to: x<=y

3.Functions

- Void: Void return type function.
- Printline: Print function with new line.
- Print: Print function.
- Open: Open function.
- Read: Read function.
- File: File function.
- Find: Find function.
- Count: Count function.
- Edit: Edit function.
- To Upper : To upper function.
- To Lower: To lower function.
- Strcmp: Strcmp function.
- Atoi: Atoi function.

4.Conditions and Loops

- While: While loop.
- For: For loop.
- If: If condition.
- Else: Else condition.
- Elseif: Else-If condition.
- Switch: Switch condition.
- Case: Case condition.
- Default: Default condition.
- Return: Return type.

BNF OF THE PROGRAMMING LANGUAGE

<program> -> <function>

|<program> <function>

<function> -> <return_type> <identifier> (<parameter_list>) <block>

<block> -> { <statement_list> }

<return_type> -> <data_type>

<parameter_list> -> <empty>

|<data_type> <identifier>

|<parameter_list> <data_type> <identifier>

identifier> -> <letter>

|<identifier> <letter>

|<identifier> <digit>

<statement_list> -> <statement> ;

|<statement_list > <statement>

<statement> -> <declaration_statement>
|<assign_statement>
|<conditional_statement>
|<loop_statement>
|<function_calling>
|<return_statement>
|<special_definitions>

<declaration_statement> -> <basic_data_type> <identifier>
|<declaration_statement> <assign_operator> <rvalue>
|<declaration_statement> , <identifier>
|<array_type> <identifier> [<integer_literal>]
|<array_type> <identifier> [<integer_literal>] <assign_operator>
{ <identifier_list> }

<assign_statement> -> <lvalue> <assign_operator> <rvalue>
|<lvalue> ++
|<lvalue> --
|<lvalue> **
|<identifier> [<integer_literal>] <assign_operator> <rvalue>

<lvalue> -> <identifier>

<assign_operator> -> =|+=|-=|*=|/=|%=

<rvalue> -> <arithmetic_expression>
|<function_calling>

<arithmetic_expression> -> <term>
|<arithmetic_expression> + <term>
|<arithmetic_expression> - <term>

<term> -> <primary>
|<term> * <primary>
|<term> / <primary>
|<term> % <primary>

<primary> -> <constant> | <identifier>

<constant> -> <string-literal>|<number-literal>|<float-literal>|<charliteral>

<operator> -> +|-|*|/ %

<conditional_statement> -> <if_statement>|<switch_statement>

<if_statement> -> if <boolean_expression> <block>
|<if_statement> else if <block>
|<if_statement> else <block>

<switch_statement> -> switch { <switch_case_statement> }
|switch { <switch_case_statement> <default_statement> }

<switch_case_statement> -> case <boolean_expression> : <statement_list>
|case <boolean_expression> : <statement_list> break
|case <boolean_expression> : <statement_list>

<default_statement> -> default : <statement_list> break
|default : <statement_list>

<loop_statement> -> <while_loop>|<for_loop>

<while_loop> -> while (<boolean_expression>) <block>

<for_loop> -> for (<for_start> <; <boolean_expression> ; <assign_statement>) <block>

<for_start> -> <declaration_statement>
|<assign_statement>
|<empty>

<function_calling> -> <identifier> (<identifier_list>)
|<identifier> . <function_calling>

<identifier_list> -> <empty>
|<call_parameter>
|<identifier_list> , <call_parameter>

<call_parameter> -> <identifier>
|<constant>
|<identifier> [<integer_literal>]
|<identifier> [<identifier>]

BNF OF SPECIAL DEFINITIONS

<special_definitions> -> <web_address>|<student_id>|<tel_no>|<plate_no>|<date>|
<credit_card>

Web Page Part

<web_address> -> <protocols> <subdomain> <domain_name> <port> <path> <query>
<webparameters> <fragment>
|<protocols> <ip>
|<protocols> <ip> <path>

<protocols> -> <protocol> ://
|<empty>

<protocol> -> http|https|ftp|pop|smtp|imap

<subdomain> -> www.
|<identifier>.
|<empty>

<domain_name> -> <identifier> . <identifier>
|<identifier> . <identifier> . <letter> <letter>

<port> -> : <integer_literal>
|<empty>

<path> -> / <identifier>
|<path> / <identifier>
|<empty>

<query> -> ?
|<empty>

<parameters> -> <identifier> = <identifier>
|<parameters> & <identifier> = <identifier>
|<empty>

<fragment> -> # <identifier>
|<empty>

<ip> -> <ip_part> . <ip_part> . <ip_part> . <ip_part>

<ip_part> -> <digit>
|<digit><digit>
|<digit><digit><digit>

Student ID Part:

<student_id> -> <undergraduate>|<graduate>

<undergraduate> -> <registrationyear> <departmentcode> <osymorder>

<registrationyear> -> <year> <integer_literal>

<year> -> 0 | 1 | 2

<departmentcode> -> <integer_literal>

<osymorder> -> <integer_literal>

<graduate> -> <graduatetype> <registrationyear> <departmentcode> <order>

<graduatetype> -> N | H

<order> -> <integer_literal>

Phone Number Part:

<tel_no> -> (0312) <integer_literal> " " <integer_literal> " " <integer_literal>

Plate Number Part:

<plate_no> -> <integer_literal> " " <identifier> " " <integer_literal>

Date Part:

<date> -> <integer_literal> . <integer_literal> . <integer_literal>

Credit Cart Part:

<credit_card> -> <XY> <integer_literal> " " <integer_literal> " " <integer_literal> " " <integer_literal> " "

<XY> -> 44 | 47

<boolean_expression> -> true|false|<identifier>|<logical_expression>

<logical_expression> -> <boolean_expression> <boolean_op> <boolean_expression_sub>
|<boolean_expression> <relation_op> <boolean_expression_sub>

<boolean_op> -> &&|||

<boolean_expression_sub> -> true|false|<identifier>|<constant>

<relation_operation> -> <|<=|>|>=|==|!=

<array_type_id> -> <identifier>

<data_type> -> <basic_data_type>|<complex_data_type>

<basic_data_type> -> <void_type>|<bool_type>|<numerical_type>|<string_type>

<void_type> -> void

<bool_type> -> bool

<numerical_type> -> <int_type>|<float_type>

<int_type> -> int

<float_type> -> float

<string_type> -> string

<complex_data_type> -> <array_type>

<array_type> -> array

<number_literal> -> <integer_literal>|<float_literal>

<integer_literal> -> <digit>
|<integer_literal> <digit>

<float_literal> -> <integer_literal> . <integer_literal>

<string_literal> -> <letter>
|<string_literal> <letter>

<char_literal> -> <letter>

<letter> -> <uppercase_letter>|<lowercase_letter>

<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

<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

<digit> -> 0|1|2|3|4|5|6|7|8|9

<empty> ->