Instituto Tecnologico y de Estudios Superiores de Monterrey

Compilers Design

TheOnlyLonely

Wednesday 02/06/2021



Jesus Carlos Martinez Gonzalez

A01037089

Contents

[1 Project description 3](#_Toc73496446)

[1.1 Purpose and scope 3](#_Toc73496447)

[1.2 Requirements and test cases definition 3](#_Toc73496448)

[1.3 Software development process 4](#_Toc73496449)

[2 Language description 5](#_Toc73496450)

[2.1 Main features 5](#_Toc73496451)

[2.2 Built-in exceptions 5](#_Toc73496452)

[3 Compiler description 5](#_Toc73496453)

[3.1 Tools used throughout development 5](#_Toc73496454)

[3.2 Lexical analysis 5](#_Toc73496455)

[3.2.1 Construction patterns 5](#_Toc73496456)

[3.2.2 Tokens 5](#_Toc73496457)

[3.3 Syntactical analysis 7](#_Toc73496458)

[3.3.1 Grammatical rules 7](#_Toc73496459)

[3.4 Semantical analysis 8](#_Toc73496460)

[3.4.1 Operations code 8](#_Toc73496461)

[3.4.2 Syntax diagrams 10](#_Toc73496462)

[3.4.3 Semantic considerations 15](#_Toc73496463)

[3.5 Memory management during compilation 16](#_Toc73496464)

[3.5.1 Data structures used during compilation 17](#_Toc73496465)

[4 Virtual machine description 18](#_Toc73496466)

[4.1 Tools used throughout development 18](#_Toc73496467)

[4.2 Memory management during execution 18](#_Toc73496468)

[4.2.1 Data structures used to achieve the execution memory 19](#_Toc73496469)

[5 Tests performed 19](#_Toc73496470)

[6 References 19](#_Toc73496471)

# 1 Project description

## Purpose and scope

The purpose of this project is to generate a rather simple programing language, it should include fundamental concepts such as arithmetic expression handling, variables, decision statements, looping statements and functions, as well as simple graphical outputting capabilities.

The project is not entirely autonomous, and it will rely on third-party tools, mainly to figure out the structure of the source code being read during compilation, and to provide to graphical environment upon which the graphical statements will run.

## Requirements and test cases definition

As stated above, the project must include features found in most popular programming languages, such as arithmetic expression handling, sequential statements such as print and return, non-sequential statements such as decision and looping statements, functions, including recursive calls, and graphical outputting statements, as to generate simple geometric figures.

To test the functionality of these features, the following test cases were defined:

|  |  |  |  |
| --- | --- | --- | --- |
| Test case # | Description | Test | Expected output |
| 0 | Simple sum | print(1 + 2); | 3 |
| 1 | Expression with multiple operator hierarchies | print(1 + 2 \* 3); | 7 |
| 2 | Complex arithmetic expression | print(2 \* (1 + 3) – 2); | 6 |
| 3 | Simple if statement | a = 2;  b = 10;  c = 2;  if (a < b) {  print(c);  } | 2 |
| 4 | Simple while statement | a = 2;  b = 10;  c = 2;  while (a < b) {  a = a + c;  print(a);  } | 4  6  8  10 |
| 5 | Simple from-to statement | from a = 2 to 10 {  print(a);  } | 2  3  4  5  6  7  8  9  10 |
| 6 | A function with return value |  |  |

## Software development process

Throughout the development of the project, an agile approach was followed, with specific weekly goals to be met, where every week the corresponding goals would be planned, designed, implemented, and tested before moving on to the next features on the calendar.

To keep track of the progress being done and that one pending, a progress log was kept in which weekly goals were stablished, the pending tasks were noted, and comments were provided whenever significant setbacks took place. The log follows a simple color scheme to denote the urgency of the backlog, with green representing the project being in a healthy state and an intense red representing a critical one.



On a personal note, regarding the software development process followed during the project, I believe my time management was the single biggest weakness in the entire project, it is ultimately what prevented me from completing the tasks to be implemented and as is now evident through the log, my organization or lack of thereof seriously jeopardized not only the project itself but the also the outcome of the subject for me. The subject itself was rather enlightening and I would certainly like to complete the project on my free time during the summer, however there is no doubt that the most valuable learning I am taking from this subject is the importance of being organized.

Even though I have taken subjects dedicated to the craft of task management, it was only during this course that I have truly gotten angry at myself for not addressing a backlog sooner.



Jesus Carlos Martinez Gonzalez

# 2 Language description

## 2.1 Main features

TheOnlyLonely is a simple programming language which can perform arithmetic operations, store values in variables of types int, float and char. Perform decision statements, including loops, make use of user-defined modules with or without return values and perform some graphical tasks such as drawing lines, circles, arcs and dots.

## 2.2 Built-in exceptions

# 3 Compiler description

## 3.1 Tools used throughout development

TheOnlyLonely was developed on the Python 3.7 programming language and makes use of the PLY lexical and syntactical analysis tool, which seeks to provide extensive input validation, error reporting, and diagnostics. PLY was used both for the lexical analysis as well as for the syntactical.

## 3.2 Lexical analysis

### 3.2.1 Construction patterns

|  |  |
| --- | --- |
| Element | REGEX |
| CT\_FLOAT | \-?[0-9]+\.[0-9]+ |
| CT\_INT | \-?[0-9]+ |
| CT\_CHAR | \'.\' |
| CT\_STRING | \".\*\" |
| ID | [A-Za-z][A-Za-z\_0-9]\* |

### 3.2.2 Tokens

|  |  |
| --- | --- |
| TOKEN | ELEMENT |
| PROGRAM | Program |
| MAIN | main |
| VARS | vars |
| INT | int |
| FLOAT | float |
| CHAR | char |
| VOID | void |
| FUNC | func |
| RETURN | return |
| READ | read |
| PRINT | print |
| IF | if |
| THEN | then |
| ELSE | else |
| DO | do |
| WHILE | while |
| FROM | from |
| TO | to |
| LINE | line |
| DOT | dot |
| CIRCLE | circle |
| ARC | arc |
| PENUP | penup |
| PENDOWN | pendown |
| COLOR | color |
| SIZE | size |
| RESET | reset |
| SETX | setx |
| SETY | sety |
| FORWARD | forward |
| BACKWARD | backward |
| LEFT | left |
| RIGHT | right |
| ROTATEX | rotatex |
| ROTATEY | rotatey |
| SEMICOLON | ; |
| COLON | : |
| COMMA | , |
| L\_PAREN | ( |
| R\_PAREN | ) |
| L\_BRACKET | { |
| R\_BRACKET | } |
| L\_SBRACKET | [ |
| R\_SBRACKET | ] |
| ASSIGN | = |
| AND | & |
| OR | | |
| EQ | == |
| NE | != |
| LTE | <= |
| GTE | >= |
| LT | < |
| GT | > |
| ADD | + |
| SUB | - |
| TIMES | \* |
| DIVIDE | / |

## 3.3 Syntactical analysis

### 3.3.1 Grammatical rules

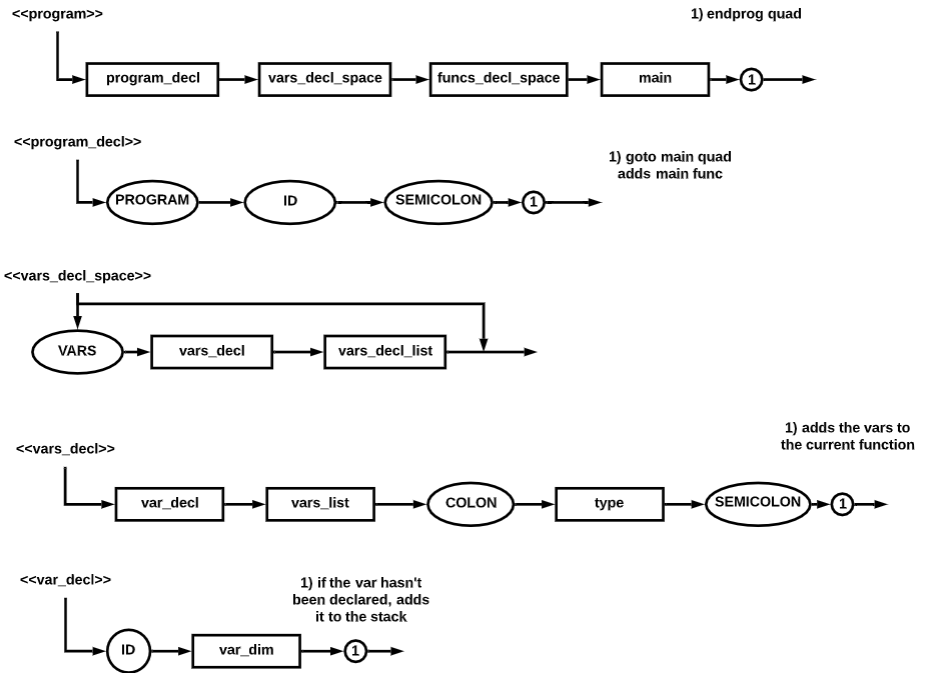
* program -> program\_decl vars\_decl\_space funcs\_decl\_space main
* program\_decl -> PROGRAM ID SEMICOLON
* vars\_decl\_space -> VARS vars\_decl vars\_decl\_list
* varss\_decl var\_decl vars\_list COLON type SEMICOLON
* var\_decl -> ID var\_dim
* var\_dim -> L\_SBRACKET CT\_INT R\_SBRACKET
* vars\_list -> COMMA var\_decl vars\_list empty
* type -> INT | FLOAT | CHAR
* vars\_decl\_list -> vars\_decl vars\_decl\_list | empty
* funcs\_decl\_space -> func\_decl funcs\_decl\_space | empty
* func\_decl -> func\_header vars\_decl\_space func\_body
* func\_header -> func\_init L\_PAREN params\_decl R\_PAREN SEMICOLON
* func\_init -> ret\_type FUNC ID
* ret\_type -> type | VOID
* params\_decl -> param\_decl | empty
* param\_decl -> param params\_list
* param -> ID COLON type
* params\_list COMMA param\_decl | empty
* func\_body -> L\_BRACKET stmnt R\_BRACKET
* stmnt -> return SEMICOLON | assignment SEMICOLON stmnt | read SEMIMCOLON stmnt | print SEMICOLON stmnt | decision SEMICOLON stmnt | loop SEMICOLON stmnt | call SEMICOLON stmnt | graphis SEMICOLON stmnt | empty
* assignment -> assignee ASSIGN hyper\_Exp
* assignee -> ID atom\_id var\_dim
* hyper\_exp -> super\_exp logic
* super\_exp -> exp relation
* exp -> term add\_sub
* term -> factor times\_divide
* factor -> L\_PAREN hyper\_exp R\_PAREN
* atom -> ID | CT\_INT | CT\_FLOAT | CT\_CHAR | call
* times\_divide -> times\_divide\_op term | empty
* times\_divide\_op -> TIMES | DIVIDE
* add\_sub -> add\_sub\_op exp | empty
* add\_sub\_op -> ADD | SUB
* relation -> rel\_op exp | empty
* rel\_op -> GTE | LTE | GT | LT | NE | EQ
* logic -> log\_op super\_exp | empty
* log\_op -> AND | OR
* call -> ID L\_PAREN args R\_PAREN
* args -> arg | empty
* arg -> hyper\_exp arg\_list
* arg\_list -> COMMA arg | empty
* return -> RETURN L\_PAREN hyper\_exp R\_PAREN
* read -> READ L\_PAREN to\_read R\_PAREN
* to\_read -> id id\_list
* id\_list -> COMMA to\_read
* print -> PRINT L\_PAREN to\_print R\_PAREN
* to\_print -> hyper\_exp printing\_list | CT\_STRING printing\_list
* printing\_list -> COMMA to\_print | empty
* decision -> IF L\_PAREN hyper\_exp R\_PAREN THEN L\_BRACKET stmnt R\_BRACKET else\_block
* else\_block -> ELSE L\_BRACKET stmnt R\_BRACKET | empty
* loop -> conditional | non\_conditional
* conditional -> WHILE L\_PAREN hyper\_exp R\_PAREN DO L\_BRACKET stmnt R\_BRACKET
* non\_conditional -> FROM assignment TO hyper\_exp DO L\_BRACKET stmnt R\_BRACKET
* graphics -> line | dot | circle | arc | penup | pendown | color | size | reset | setx | sety | forward | backward | left | right | rotatex | rotatey
* line -> LINE L\_PAREN exp R\_PAREN
* dot -> DOT L\_PAREN exp R\_PAREN
* circle -> CIRCLE L\_PAREN exp R\_PAREN
* arc -> ARC L\_PAREN exp R\_PAREN
* penup -> PENUP L\_PAREN R\_PAREN
* pendown -> PENDOWN L\_PAREN R\_PAREN
* color -> COLOR L\_PAREN CT\_STRING R\_PAREN
* size -> SIZE L\_PAREN exp R\_PAREN
* reset -> RESET L\_PAREN R\_PAREN
* setx -> SETX L\_PAREN exp R\_PAREN
* sety -> SETY L\_PAREN exp R\_PAREN
* forward -> FORWARD L\_PAREN exp R\_PAREN
* backward -> BACKWARD L\_PAREN exp R\_PAREN
* left -> LEFT L\_PAREN exp R\_PAREN
* right -> RIGHT L\_PAREN exp R\_PAREN
* rotatex -> ROTATEX L\_PAREN exp R\_PAREN
* rotatey -> ROTATEY L\_PAREN exp R\_PAREN
* main -> main\_init func\_body
* maint\_init -> MAIN L\_PAREN R\_PAREN
* empty ->

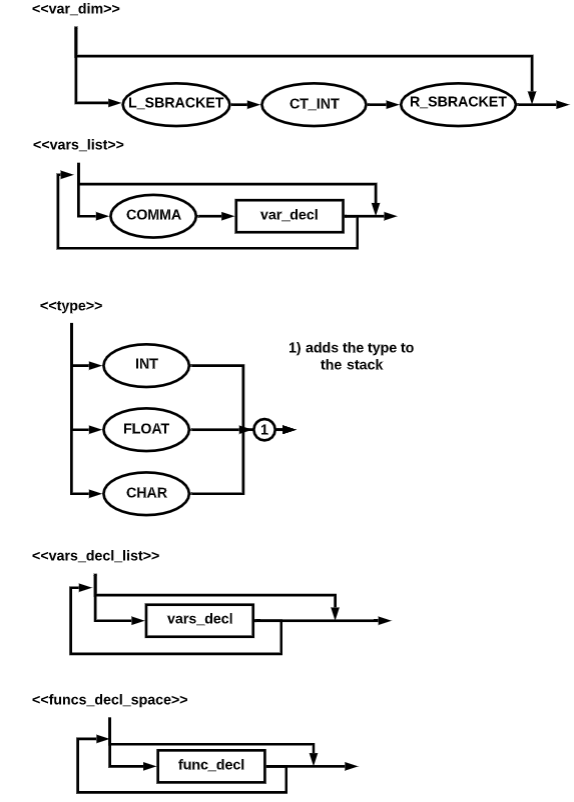
## 3.4 Semantical analysis

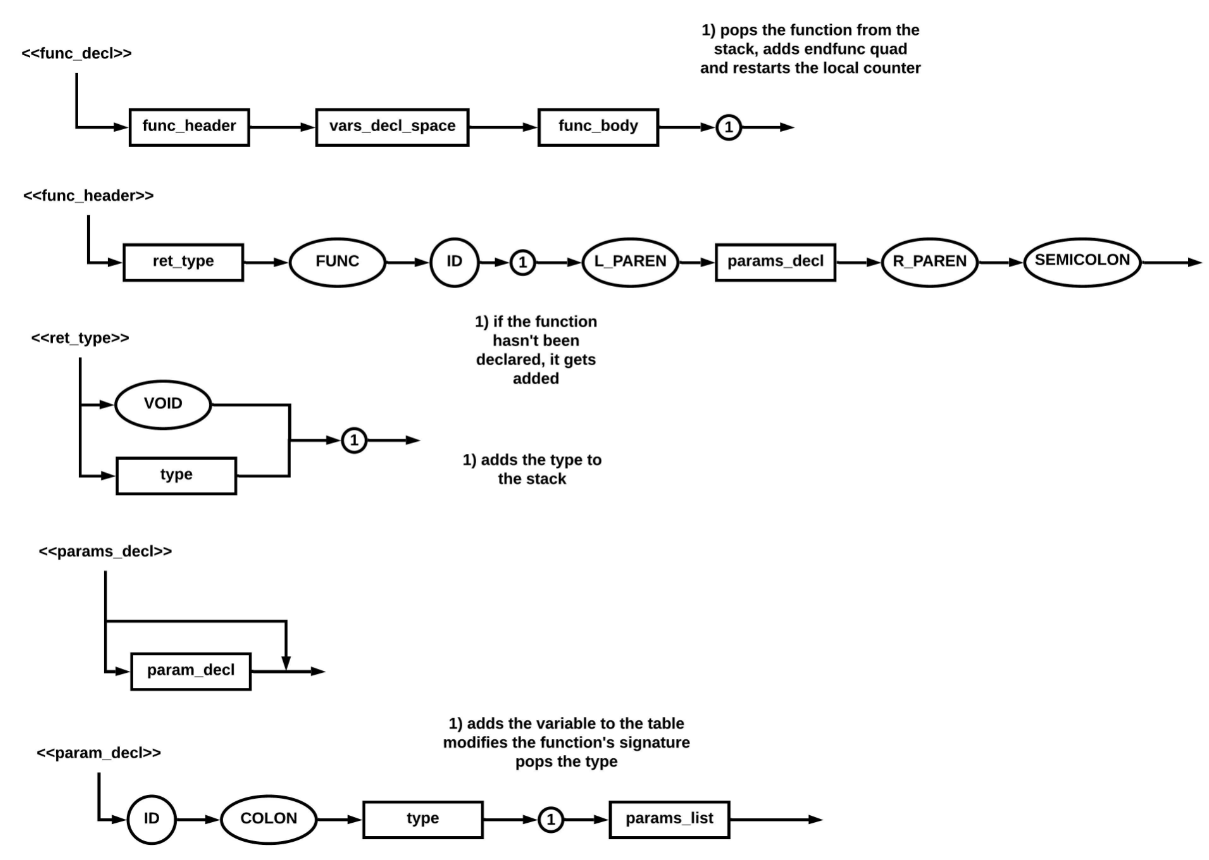
### 3.4.1 Operations code

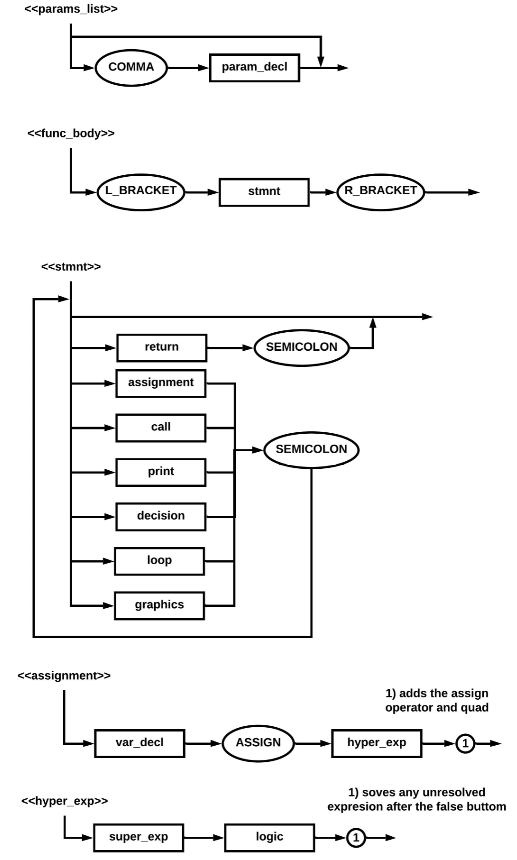
|  |  |
| --- | --- |
| Operator | Semantic code |
| + | 0 |
| - | 1 |
| \* | 2 |
| / | 3 |
| == | 4 |
| != | 5 |
| < | 6 |
| <= | 7 |
| > | 8 |
| >= | 9 |
| & | 10 |
| | | 11 |
| = | 12 |
| print | 13 |
| goto | 14 |
| gotof | 15 |
| endunf | 16 |
| era | 17 |
| param | 18 |
| gosub | 19 |
| return | 20 |
| read | 21 |
| endprog | 22 |
| line | 23 |
| dot | 24 |
| circle | 25 |
| arc | 26 |
| penup | 27 |
| pendown | 28 |
| color | 29 |
| size | 30 |
| reset | 31 |
| setx | 32 |
| sety | 33 |
| forward | 34 |
| backward | 35 |
| left | 36 |
| right | 37 |
| rotatex | 38 |
| rotatey | 39 |

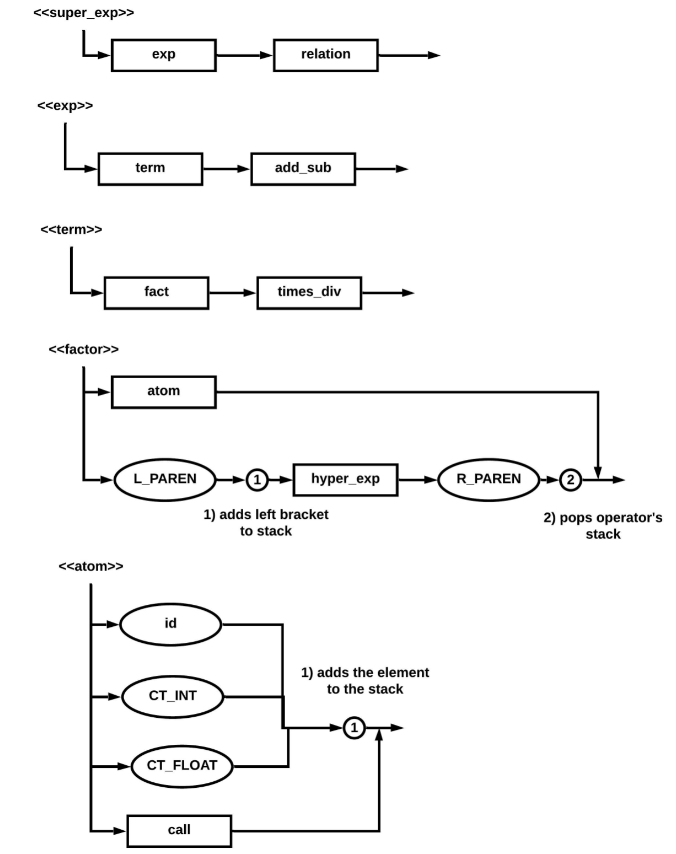
### 3.4.2 Syntax diagrams

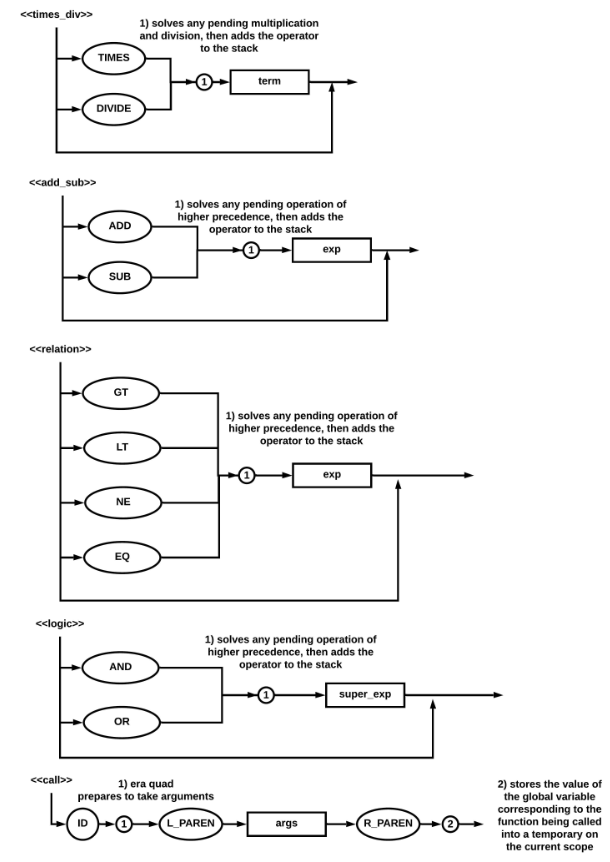


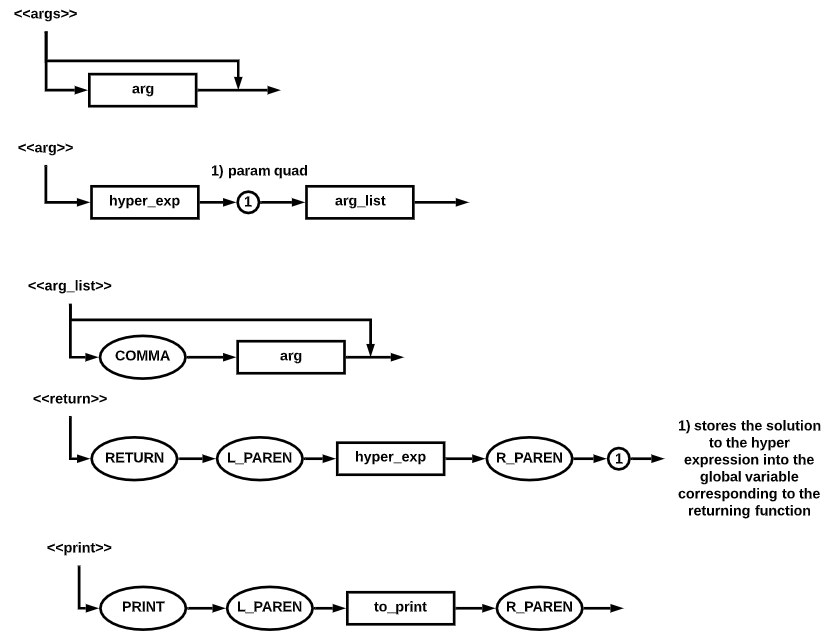


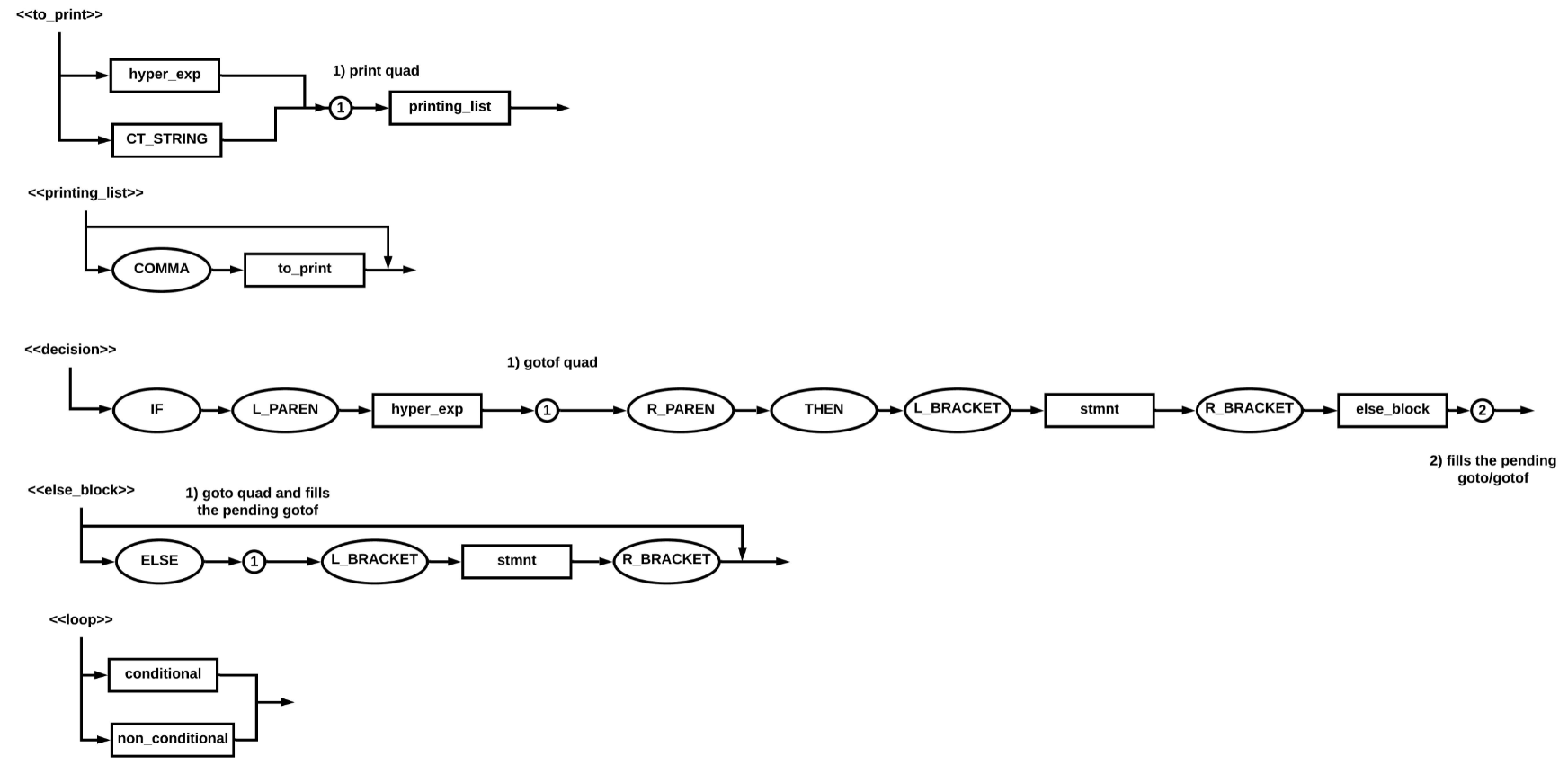


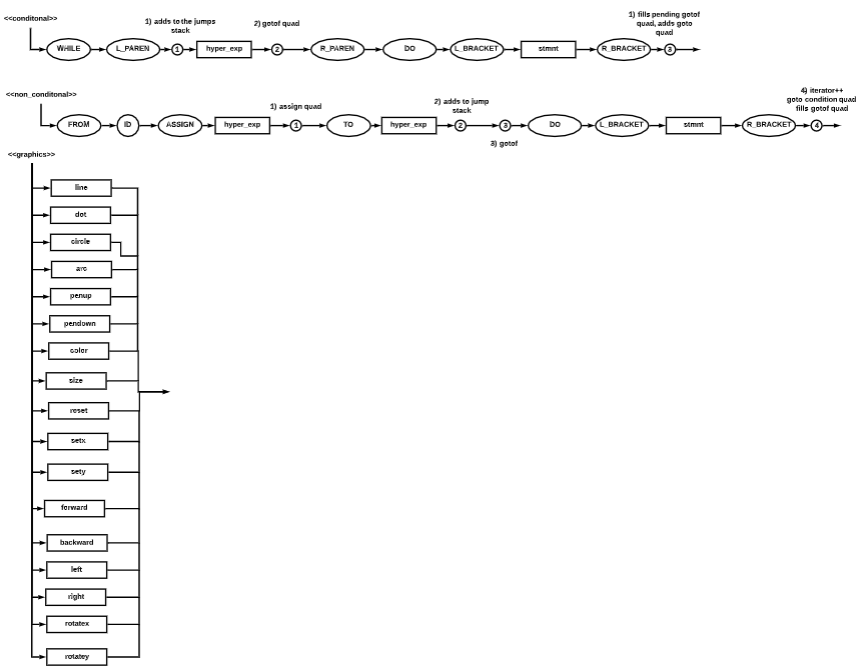


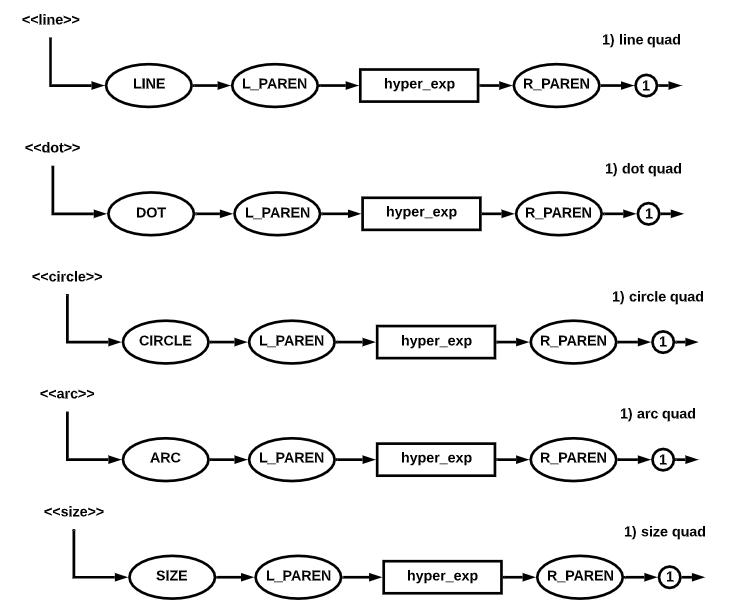


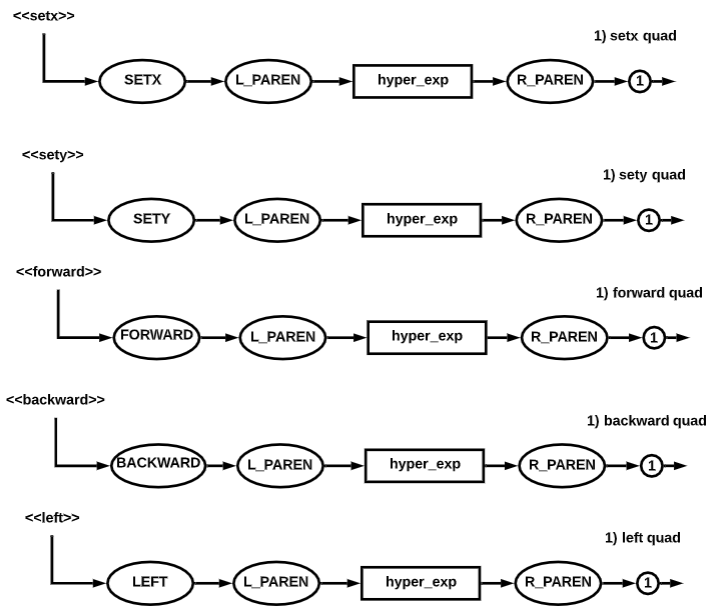


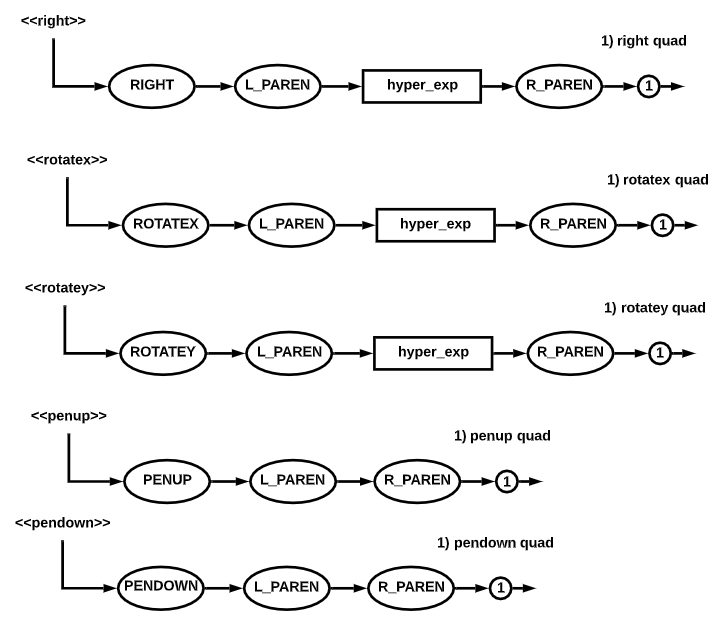


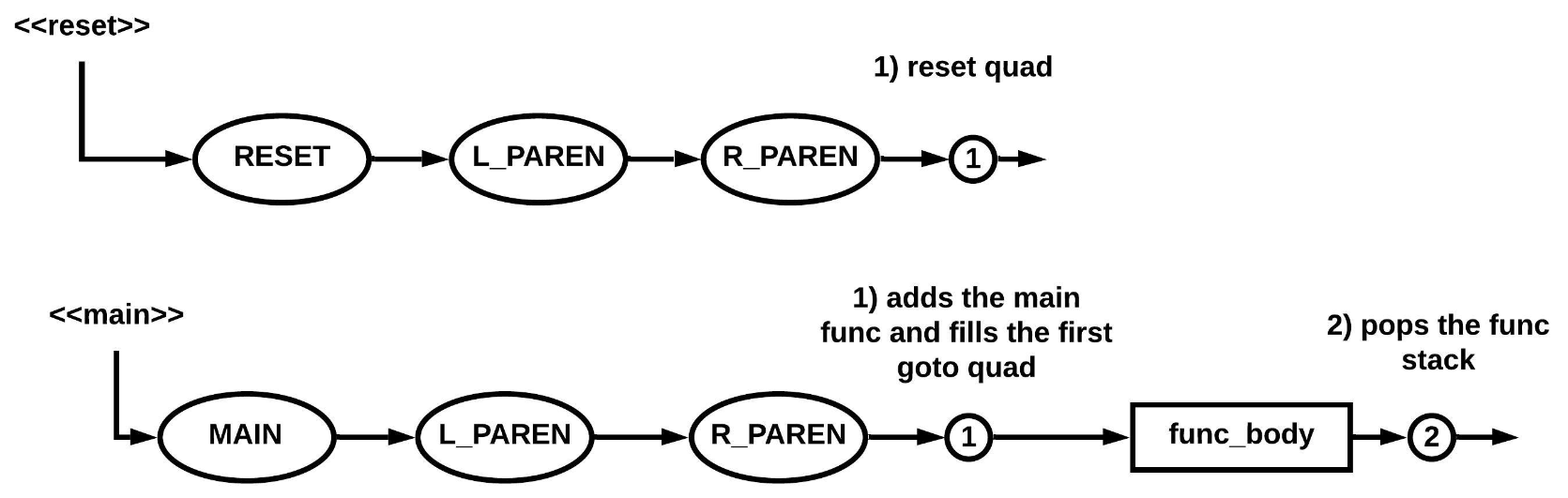












### 3.4.3 Semantic considerations

The following semantic considerations were made for the language, any possible combination not on the table is not supported by the language.

|  |  |  |  |
| --- | --- | --- | --- |
| Type 1 | Operator | Type 2 | Resulting type |
| Int | + | Int | Int |
| Int | - | Int | Int |
| Int | \* | Int | Int |
| Int | / | Int | Int |
| Int | == | Int | Bool |
| Int | != | Int | Bool |
| Int | < | Int | Bool |
| Int | <= | Int | Bool |
| Int | > | Int | Bool |
| Int | >= | Int | Bool |
| Int | & | Int | Bool |
| Int | | | Int | Bool |
| Int | = | Int | int |
| int | == | Float | Bool |
| Int | != | Float | Bool |
| Int | < | Float | Bool |
| Int | <= | Float | Bool |
| Int | > | Float | Bool |
| Int | >= | Float | Bool |
| Int | & | Float | Bool |
| Int | | | Float | Bool |
| Float | == | Int | Bool |
| Float | != | Int | Bool |
| Float | < | Int | Bool |
| Float | <= | Int | Bool |
| Float | > | Int | Bool |
| Float | >= | Int | Bool |
| Float | & | Int | Bool |
| Float | | | Int | Bool |
| Float | + | Float | Float |
| Float | - | Float | Float |
| Float | \* | Float | Float |
| Float | / | Float | Float |
| Float | == | Float | Bool |
| Float | != | Float | Bool |
| Float | < | Float | Bool |
| Float | <= | Float | Bool |
| Float | > | Float | Bool |
| Float | >= | Float | Bool |
| Float | & | Float | Bool |
| Float | | | Float | Bool |
| Float | = | Float | Float |
| Char | == | Char | Bool |
| Char | != | Char | Bool |
| Char | < | Char | Bool |
| Char | <= | Char | Bool |
| Char | > | Char | Bool |
| Char | >= | Char | Bool |
| Char | = | Char | char |

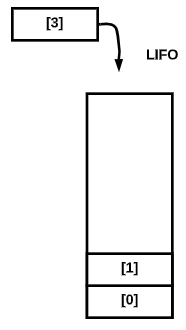
## 3.5 Memory management during compilation

During the compilation process, memory is mostly an illusion, since in compilation no variable values are known, no values are stored and hence the addresses corresponding to the variables are merely scope-specific and type-specific counters.

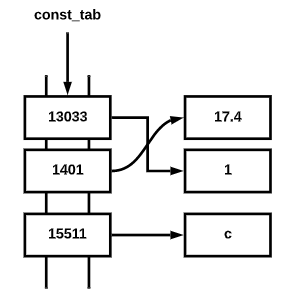
### 3.5.1 Data structures used during compilation

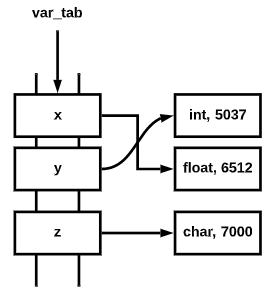
The following data structures are used during the compilation process:

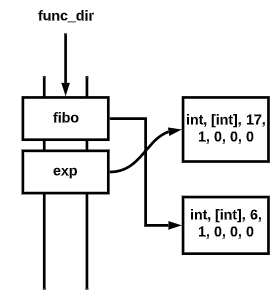
* Stacks: Several stacks are used in the form of python lists, lists are used in this context since the most common operations to be performed on them are pops, in python, popping a list has a 0(1) time complexity.



* Constants table: A very simple python dictionary, with the addresses (counters) being the keys for the values, although lists would have gotten the job done, in Python, dictionaries outperform lists when it comes to indexing.



* Variables table: Also a Python dictionary, picked for the same reasons as before.
  + 
* Functions directory: Yet another Python dictionary



# 4 Virtual machine description

## 4.1 Tools used throughout development

The only tool used during execution which isn’t present in compilation is the Turtle Graphics library, which is a built-in Python library which provides graphical output capabilities, such as a drawing board and instructions.

## 4.2 Memory management during execution

### 4.2.1 Data structures used to achieve the execution memory

# 5 Tests performed

# 6 References

* David M. Beazley. (2018). PLY (Python Lex-Yacc). 02/06/2021, de N.A Sitio web: <https://www.dabeaz.com/ply/ply.html>
* N.A. (2021). Turtle graphics. 02/06/2021, de The Python Software Foundation Sitio web: [https://docs.python.org/3/library/turtle.html](https://docs.python.org/3/library/turtle.html#turtle.colormode)