SER 502
Spring 2021
Project Team 7

giaa

Arizona State University

April 2021

Team Members

Arizona State University

Abhishek Mohabe

Graduate Student, Computer Software Engineering

Apoorva Giliyal

Graduate Student, Computer Software Engineering

Itiparna Mahala

Graduate Student, Computer Software Engineering

Gautham Krishna

Graduate Student, Computer Software Engineering

Agenda

- Introduction
- Tools
- Language Design
- Language Grammar
- Language Features
- Steps for Execution
- Execution Artifacts

Introduction

GIAA comes from the initials of all the four team members(Gautham, Itiparna, Abhishek, Apoorva). The token is generated using python file.

Parse tree generation is done using prolog.

Language design:

- Program starts with “begin” and “end”.
- Input files have a .GIAA extension.
- Lexer.py is used to generate a list of tokens which are parsed using Prolog to give the final output.

Tools Used

- SWI Prolog (Compilation)
- Python 3.9 (Tokens)
- SWI Prolog (Parser)
- SWI Prolog (Interpreter)

Language Design

Source Code

The sample source code of programming language is written in the **.GIAA** format

Lexical Analyser

The source code is then passed into a **Lexical Analyser** developed in **Python**

Parser

The language parses the list of tokens and generates a syntactically correct **parse tree**

Interpreter

The parse tree generated passed through **syntax driven semantics** to generate the desired **output of the program**

Language Grammar

The language grammar has been created with DCG.

SWISH File Edit Examples Help 274 users online

Program

```
:- table boolean/3, expression/3, term/3.

%to parse the program
program(t_program(A)) -->['begin'], block(A), ['end'].

%to parse the block
block(t_block(A)) --> ['{'], block_section(A), ['}'].
block_section(t_block(A, B)) --> statements(A), block_section(B).
block_section(t_block(A)) --> statements(A).

%to parse the different type of statements
statements(t_statements(X)) --> declaration(X), [;].
statements(t_statements(X)) --> assignment(X), [;].
statements(t_statements(X)) --> expression(X), [;].
statements(t_statements(X)) --> boolean(X), [;].
statements(t_statements(X)) --> printstatements(X), [;].
statements(t_statements(X)) --> ifcondition(X).
statements(t_statements(X)) --> ternarycondition(X), [;].
statements(t_statements(X)) --> forloop(X).
statements(t_statements(X)) --> whileloop(X).
statements(t_statements(X)) --> forrange(X).
statements(t_statements(X)) --> iterator(X), [;].

%to parse variable declaration
```

Language Features

- **DataTypes**
  - Int-1,2,3,4
  - Bool-true/false
  - String-”Prolog”

- **Arithmetic Operations**
  - Addition- +
  - Subtraction-’-’
  - Multiplication-’*’
  - Division-’/’

- **Relational Operators**
  - Equal to ==
  - Not equal to !=
  - Greater than >
  - Lesser than <
  - Greater than or equal to >=
  - Lesser than or equal to <=

- **Increment Operator ++**

- **Decrement Operator --**

Language Features

Statements

Assignment Statement
bool flag=true;

Print Statement
print x;

Declaration Statement
Int a;

If condition

```
if (x>= 10)
{
x=x+2;
}
else
{
flag=false;
}
```

NOTE: else is not compulsory.

Language Features

For loop

```
for(int i=1;i<15;i++)
{
print i;
}
```

For loop with range

```
for i in range(0:20)
{
print i;
}
```

While loop

```
while(x != 10)
        {
        y = y * 2;
        x = x + 2;
        z = x * y;
        }
```

<u>Ternary Operator</u>

```
x > y ? print x ; : print y;;
```

Steps for Execution

- The first step in the execution by GIAA is creating a input file containing the program with the .GIAA extension.
- This input file is used as input for the Lexer.
- After creating the input file, the next step would be to open swipl on the terminal
- Compilation of the giaa.pl using the command:

  ?-[‘path to the giaa.pl file’].

- Running the input file containing program

  ?-giaa(‘path to lexer.py file’,’path to input file with .GIAA extension’).

Sample Program

```
begin
{
        int x=2;
        int y=3;
        int z=0;
        while(x != 10)
        {
        y = y * 2;
        x = x + 2;
        z = x * y;
        }
        print x;
        print y;
        print z;
}
end
```

Execution Artifacts

```
Abhisheks-MacBook-Pro:~ abhishek$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.3.22)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['/Users/abhishek/Desktop/SER502-Spring2021-Team7/src/GIAA.pl'].
Warning: /Users/abhishek/Desktop/SER502-Spring2021-Team7/src/GIAA.pl:1:
Warning:    Singleton variables: [Output]
true.
```

```
?- giaa('/Users/abhishek/Desktop/SER502-Spring2021-Team7/src/Lexer.py','/Users/abhishek/Desktop/SER502-Spring2021-Team7/data/boolVar.GIAA').
GIAA Programming Language v1.0
SER 502 - Spring 2021 - Team 7
@Authors - Abhishek Mohabe, Apoorva Giliyal, Gautham Krishna, Itiparna Mahala

Executing...../Users/abhishek/Desktop/SER502-Spring2021-Team7/data/boolVar.GIAA

List of Tokens:
[begin,{,bool,flag,=,true,;,int,x,=,28,;,if,(,x,>=,10,),{,x,=,x,+,2,;,},else,{,flag,=,false,;,},print,x,;,print,flag,;,},end]

Parse Tree:
t_program(t_block(t_block(t_statements(t_declarebool(bool,t_id(flag),true)),t_block(t_statements(t_declareint(int,t_id(x),t_num(28))),t_block(t_statements(t_i
f_cond(t_condition(t_id(x),>=,t_num(10)),t_block(t_block(t_statements(t_assign(t_id(x),t_add(t_id(x),t_num(2))))))),t_block(t_block(t_statements(t_assign(t_id(
flag),t_id(false)))))))),t_block(t_statements(t_print(t_id(x))),t_block(t_statements(t_print(t_id(flag)))))))))

Output:
30
true
true .

?-
```

Execution Artifacts

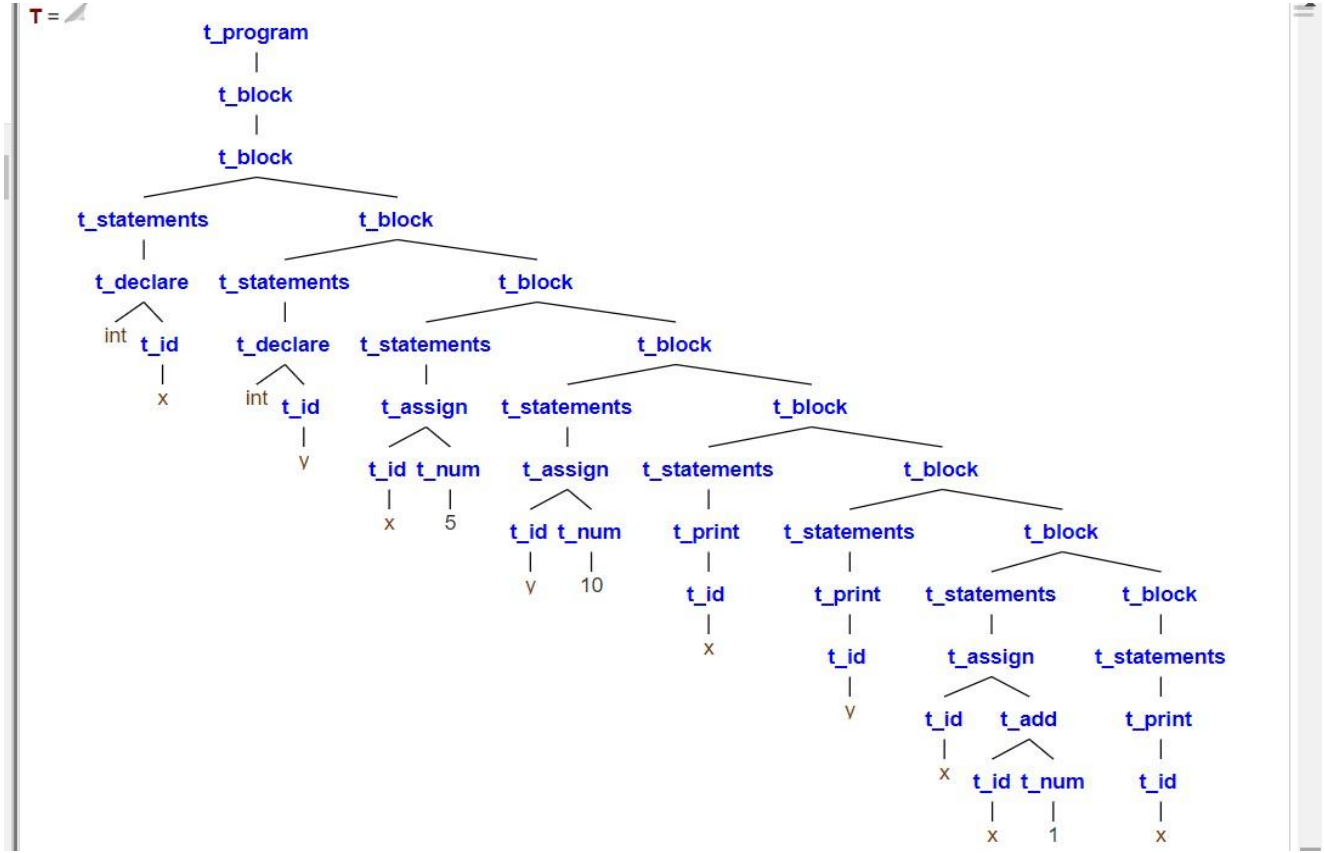
```
program(T,[begin,'{',int,x,;,int,y,;,x,=,5,;,y,=,10,;,print,x,;,print,y,;,x,=,x,+,1,;,print,x,;,'}',end ],[]),eval_program(T,F).
```

```
5
10
6
F = [(int,x,6), (int,y,10)] ,
T = t_program(t_block(t_block(t_statements(t_declare(int, t_id(x))), t_block(t_statements(t_declare(int, t_id(y))),
t_block(t_statements(t_assign(t_id(x), t_num(5))), t_block(t_statements(t_assign(t_id(y), t_num(10))),
t_block(t_statements(t_print(t_id(x))), t_block(t_statements(t_print(t_id(y))), t_block(t_statements(t_assign(t_id(x), t_add(t_id(x),
t_num(1)))), t_block(t_statements(t_print(t_id(x))))))))))))
```

Next | 10 | 100 | 1,000 | Stop

```
?- program(T,[begin,'{',int,x,;,int,y,;,x,=,5,;,y,=,10,;,print,x,;,print,y,;,x,=,x,+,1,;,print,x,;,'}',end ],
[]),eval_program(T,F).
```

Execution Artifacts - Parse Tree



THANK YOU