A New Programming Language
for
Data Structures Visualization

Team Members :

1. R. Vamshi Krishna - IS201411038
2. Ch. Nagaraju - IS201301030

Basic Idea:

- The idea of this project is to create a programming language in which we can visualise the data structures and their operations.
- The data structures we implemented are Array, Stack and Queue.
- The interpreter is developed in python.

Tools/Libraries we used:

- Python-Lex-Yacc Library (PLY) for Lexical Analysis and Parsing.
- Turtle module for graphics.

Lexical Analysis:

- **Tokens :**

('LPARS', 'RPARS', 'NUMBER', 'FLOAT', 'PLUS', 'MINUS', 'MULT', 'DIVI', 'SEMICO', 'EQUAL', 'LE', 'GE', 'LL', 'GG', 'EE', 'STRI', 'COMMA', 'INSER', ‘APPEND’, 'DELET', 'ERASE', 'ID', 'LSBS', 'RSBS', 'OSB', 'CSB')

+

- **Reserved Tokens :**

('Create', 'Array', 'Queue', 'Stack', 'int', 'float, 'string', 'if', 'else', 'for', 'in', 'range', 'prints')

Grammar

```
   statement   : start
       start   : declaration start
               | operations start
               | expression SEMICO start
               | IF condition SEMICO start
               | IF  conditione SEMICO start
               | FOR conditionw SEMICO start
               | PRINTS something SEMICO start
               | empty
 declaration   :  CREATE ds dt ID SEMICO
               | CREATE ds ID SEMICO
               | CREATE ds dt ID OSB NUMBER CSB SEMICO
```

Syntax

Array

❖ Create Array <DT> <ID>[<SIZE>];
Ex : Create Array int A[10];
Create Array float A[8];

❖ <ID>[<index>] = <Value>;
Ex : A[0] = 2;

❖ <ID>.append(<ID>);
Ex : A.append(3);

Stack

❖ Create Stack <DT> <ID>;
Ex : Create Stack int S;
Create Stack string S2;

❖ <ID>.insert(<ID>);
Ex : S.insert(15);

❖ <ID>.delete();
Ex : S.delete();

❖ <ID>.erase();

Queue

❖ Create Queue <DT> <ID>;
Ex : Create Queue int Q;
Create Queue string B;

❖ <ID>.insert(<ID>);
Ex : Q.insert(15);

❖ <ID>.delete();
Ex : Q.delete();

❖ <ID>.erase();

Syntax

If-else loop

```
if (<Expression>) {
    <Expression>;
};
else {
    <Expression>;
};
```

For loop

```
for (<ID> in range <NUM>){
        <Expression>;
};
```

```
Ex:  Create Stack int S;
        for (i in range 5) {
            S.insert(i);
        };
```

- Commenting the line is done by the ‘#’.

Furthermore

- An execution bar is implemented that displays the currently executing operation in the code.
- Type checking and detecting the errors like syntax errors, variable un-declarations or re-declarations are also implemented.

Thank You….

??