NMAM Institute of Technology

(An Autonomous Institute Affiliated to VTU, Belagavi) (A unit of

NITTE Education Trust)

NITTE – 574110, UDUPI DIST., KARNATAKA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROJECT REPORT ON

COMPILER DESIGN

SUBMITTED BY:

Manukashyap	**MANJUNATHA PATKAR**
4NM17CS0101	4NM17CS0100
VI SEM, ‘B’ SEC	VI SEM, ‘B’ SEC
Department of CSE	Department of CSE
NMAMIT, Nitte	NMAMIT, Nitte

Under the Guidance of:

Mrs. Minu P. Abraham

Assistant Professor Gd II

Department of Computer Science and Engineering

NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India
(ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC
☎: 08258 - 281039 – 281263, Fax: 08258 – 281265

Department of Computer Science and Engineering

B.E. CSE Program Accredited by NBA, New Delhi from 1-7-2018 to 30-6-2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

"Compiler Design Project"

is a bonafide work carried out by

Manukashyap(4NM17CS101)   Manjunatha Patkar(4NM17CS100)

In partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering
prescribed by Visvesvaraya Technological University,
Belgaum during the year 2019-2020.

It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report.

The Mini project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Signature of Guide   Signature of HOD

ACKNOWLEDGEMENT

We believe that our project will be complete only after we thank the people who have contributed to making this project successful.

First and foremost, we express our deep sense of gratitude and indebtedness to our guide **Mrs. Minu P. Abraham**, Professor, Department of Computer Science and Engineering, for his inspiring guidance, constant encouragement, support and suggestions for improvement during the course of our project.

We sincerely thank **Dr. K.R. Udaya Kumar Reddy**, Head of Department of Computer Science and Engineering, Nitte Mahalinga Adyantaya Memorial Institute of Technology, Nitte.

Our sincere thanks to our beloved principal, **Dr. Niranjan N. Chiplunkar** for giving us an opportunity to carry out our project work at our college and providing us with all the needed facilities.
We also thank all those who have supported us throughout the entire duration of our project.

Finally, we thank the staff members of the Department of Computer Science and Engineering and all our friends for their honest opinions and suggestions throughout the course of our project.

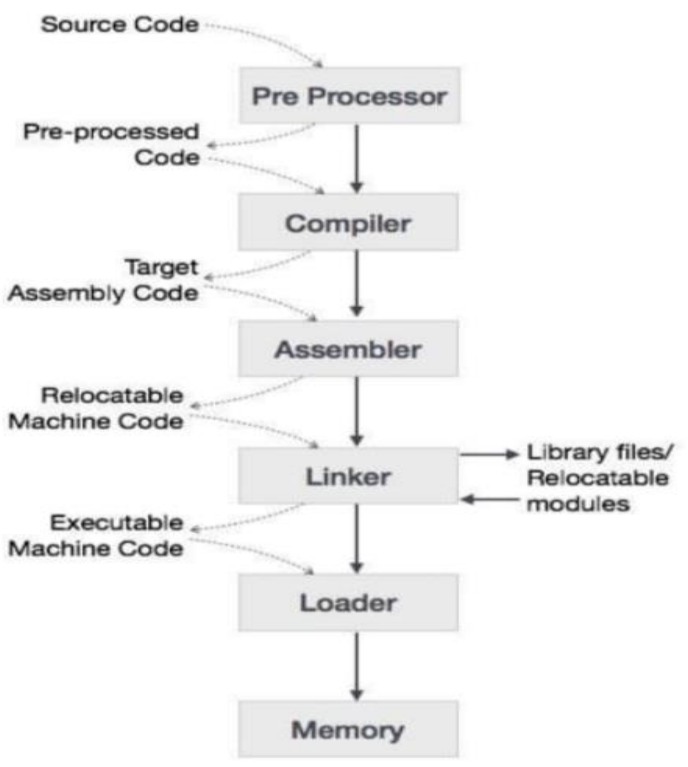
Manukashyap - 4NM17CS0101
Manjunatha Patkar - 4NM17CS100

TABLE OF CONTENTS

ABSTRACT

Computers are balanced mixture of software and hardware. Hardware is just a piece of mechanical device and its functions are being controlled by a compatible software. Hardware understands instructions in the form of electronic charge, which is counterpart of binary language in software programming. Binary language has only two characters ‘0’ and ‘1’ which forms the low level language. Compiler is a software which converts a program written in high level language(Source Language) to low level language(Object/Target/Machine Language).

LANGUAGE PROCESSING SYSTEM:



INTRODUCTION

What is a compiler?

In order to reduce the complexity of designing and building computers, nearly all of these are made to execute relatively simple commands (but do so very quickly). A program for a computer must be built by combining these very simple commands into a program in what is called machine language. Since this is a tedious and error prone process most programming is, instead, done using a high-level programming language. This language can be very different from the machine language that the computer can execute, so some means of bridging the gap is required. This is where the compiler comes in. A compiler translates (or compiles) a program written in a high-level programming language that is suitable for human programmers into the low-level machine language that is required by computers. During this process, the compiler will also attempt to spot and report obvious programmer mistakes. Using a high-level language for programming has a large impact on how fast programs can be developed.

The main reasons for this are:

- Compared to machine language, the notation used by programming languages is closer to the way humans think about problems.
- The compiler can spot some obvious programming mistakes.
- Programs written in a high-level language tend to be shorter than equivalent programs written in machine language.
- Another advantage of using a high-level level language is that the same program can be compiled to many different machine languages and, hence, be brought to run on many different machines.
- On the other hand, programs that are written in a high-level language and automatically translated to machine language may run somewhat slower than programs that are hand-coded in machine language. Hence, some time critical programs are still written partly in machine language.
- A good compiler will, however, be able to get very close to the speed of handwritten machine code when translating well-structured programs.

The Phases Of a Compiler:

The compilation process is a sequence of various phases. Each phase takes input from its previous stage, has its own representation of source program, and feeds its output to the next phase of the compiler. Let us understand the phases of a compiler.

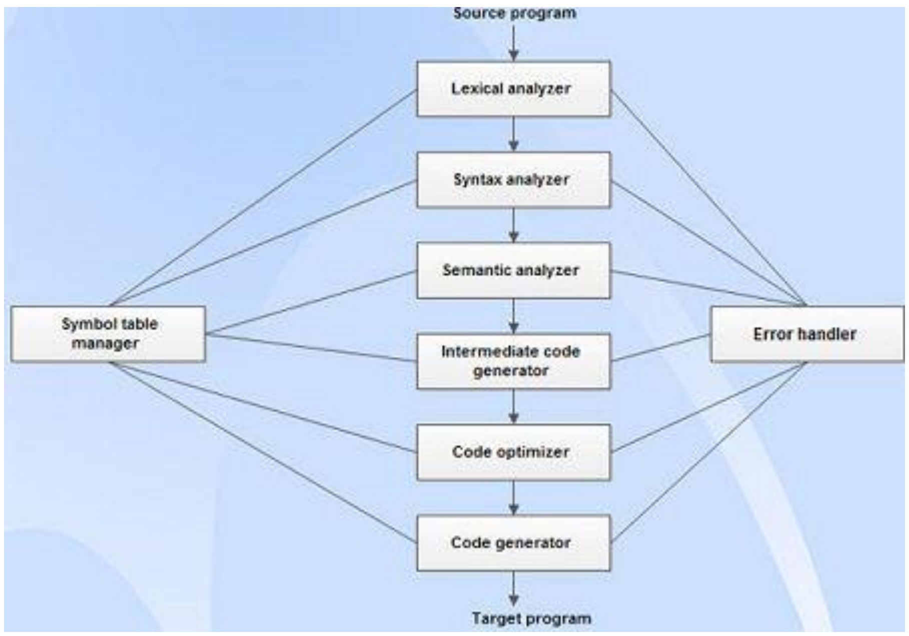


Fig: Phases of Compiler

Lexical Analysis:

The first phase of scanner works as a text scanner. This phase scans the source code as a stream of characters and converts it into meaningful lexemes. Lexical analyzer represents these lexemes in the form of tokens as:

<token-name, attribute-value>

Syntax Analysis:

The next phase is called the syntax analysis or **parsing**. It takes the token produced by lexical analysis as input and generates a parse tree (or syntax tree). In this 3 phase, token arrangements are checked against the source code grammar, i.e., the parser checks if the expression made by the tokens is syntactically correct.

Semantic Analysis:

Semantic analysis checks whether the parse tree constructed follows the rules of language. For example, assignment of values is between compatible data types, and adding string to an integer. Also, the semantic analyzer keeps track of identifiers, their types and expressions; whether identifiers are declared before use or not, etc. The semantic analyzer produces an annotated syntax tree as an output.

Intermediate Code Generation:

After semantic analysis, the compiler generates an intermediate code of the source code for the target machine. It represents a program for some abstract machine. It is in between the high-level language and the machine language. This intermediate code should be generated in such a way that it makes it easier to be translated into the target machine code.

Code Optimization:

The next phase does code optimization of the intermediate code. Optimization can be assumed as something that removes unnecessary code lines, and arranges the sequence of statements in order to speed up the program execution without wasting resources (CPU, memory).

Code Generation:

In this phase, the code generator takes the optimized representation of the intermediate code and maps it to the target machine language. The code generator translates the intermediate code into a sequence of (generally) re-locatable machine code. Sequence of instructions of machine code performs the task as the intermediate code would do.

Symbol Table:

It is a data-structure maintained throughout all the phases of a compiler. All the identifiers' names along with their types are stored here. The symbol table makes it easier for the compiler to quickly search the identifier record and retrieve it. The symbol table is also used for scope management

PROBLEM STATEMENT

Design a compiler for the following pseudocode:

```
int main()
    char operator;
    int firstNumber,secondNumber;

    switch(operator)
    begin
     case '+':
         printf(firstNumber+secondNumber);
         break;

       case '-':
         printf(firstNumber-secondNumber);
         break;
       end
  return 0;
```

LEXICAL ANALYSIS

Lexical analysis is the first phase of a compiler. It takes the modified source code from language pre processors that are written in the form of sentences. The lexical analyzer breaks these syntaxes into a series of tokens, by removing any hitespace or comments in the source code. If the lexical analyzer finds a token invalid, it generates an error. The lexical analyzer works closely with the syntax analyzer.

It reads character streams from the source code, checks for legal tokens, and passes the data to the syntax analyzer when it demands. The main purpose of lexical analysis is to make life easier for the subsequent syntax analysis phase. In theory, the work that is done during lexical analysis can be made an integral part of syntax analysis, and in simple systems this is indeed often done. However, there are reasons for keeping the phases separate:

- **Efficiency:** A lexer may do the simple parts of the work faster than the more general parser can. Furthermore, the size of a system that is split in two may be smaller than a combined system. This may seem paradoxical but, as we shall see, there is a non- linear factor involved which may make a separated system smaller than a combined system.
- **Modularity:** The syntactical description of the language need not be cluttered with small lexical details such as white-space and comments.
- **Tradition:** Languages are often designed with separate lexical and syntactical phases in mind, and the standard documents of such languages typically separate lexical and syntactical elements of the languages.

Token: Token is a sequence of characters that can be treated as a single logical entity.

Typical tokens are:

1. Identifiers   4. Special symbols

2. keywords
3. operators

5. constants

Pattern: A set of strings in the input for which the same token is produced as output. This set of strings is described by a rule called a pattern associated with the token.

Lexeme: A lexeme is a sequence of characters in the source program that is matched by the pattern for a token.

LEXER CODE

```
def tokenize(file):
 f = open(file, 'r')

 operators = { '+': 'Additon Operator', '-' : 'Substraction Operator', '/' : 'Division Operator', '*': 
'Multiplication Operator', '++' : 'increment Operator', '--' : 'Decrement Operator'}
 optr_keys = operators.keys()

 comments = {r'//' : 'Single Line Comment',r'/*' : 'Multiline Comment Start', r'*/' : 'Multiline 
Comment End', '/**/' : 'Empty Multiline comment'}
 comment_keys = comments.keys()

 header = {'.h': 'header file'}
 header_keys = header.keys()

 sp_header_files = {'stdio.h':'Standard Input Output Header', 'stdlib.h' : 'Standard Library 
Header'}

 datatype = {'int': 'Integer','float' : 'Floating Point', 'char': 'Character', 'void' : 'Void'}
 datatype_keys = datatype.keys()

 keyword = {'return' : 'Return', 'case' : 'Case', 'break':'Break','switch' : 'Switch', 'begin' : 'Begin 
keyword', 'end' : 'End keyword'}
 keyword_keys = keyword.keys()

 delimiter = {';':'Semicolon'}
 delimiter_keys = delimiter.keys()

 blocks = {'{' : 'Curly Brace Open', '}' : 'Curly Brace Closed', '(' : 'Open Parenthesis', ')' : 'Close 
Paranthesis'}
 block_keys = blocks.keys()

 builtin_functions = {'printf':'printf', 'main' : 'Main function', 'scanf' : 'scanf'}

 non_identifiers = ['_','`','~','!','@','#','$','%','^','&','|','"','{'
 ,'}','[',']','<','>','?','/', ',','=','\'',':']
```

```
relational_operators = {'==' : 'Relational Equals to', '<=' : 'Less than or equal to',
                        '>=' : 'Greater than or equal to','!=' : 'Not Equal to'}

numerals = ['0','1','2','3','4','5','6','7','8','9','10']

insert_space = re.compile(r"([(<,+\-*=/%;:>)])")

i = f.read()

dataFlag = False
token_list = []

count = 0
flag1=flag2=flag3=0
program = i.split('\n')
print("LEXICAL ANALYSIS:")
print()
for line in program:
   if line == '':
       continue
   count += 1
   print("Line #" +  str(count))
   print("Line :", line)
   line = insert_space.sub(" \\1 ", line)
   line = re.sub(r'\t', '', line)
   tokens = line.split(' ')

   tokens = [x for x in tokens if x]
   print(tokens)
   for token in tokens:
       token = token.strip('')
       if token == '':
           continue

       if token in builtin_functions.keys():
           if(token=='printf'):
               inp.append('t')
           elif(token=='main'):
               inp.append('e')
           print(token, ":", builtin_functions[token])

       elif token in block_keys:
           if(token=='('):
               inp.append('f')
           elif(token==')'):
               inp.append('g')
```

```
    print(token, ":", blocks[token])

elif token in optr_keys:
     print(tokens.index(token))
     print(token, ":", operators[token])
     inp.append('m')

#elif token in comment_keys:
    #print("Comment Type: ", comments[token])

elif '.h' in token:
    print(token, ":", sp_header_files[token])

elif token in non_identifiers:

    if(token=='='and tokens[tokens.index(token) + 1] == '='):
        print((token + "="), ":", relational_operators[token + "="])
        tokens.pop(tokens.index(token) + 1)
        token += "="
        inp.append('o')
    elif(token=='='):
        inp.append('k')
        print(token, ":", "Assignment operator")
    elif(token==','):
        inp.append('i')
        print(token, ":", "comma operator")

    elif(token=='<' and tokens[tokens.index(token) + 1] == '=' ):
         print((token + "="), ":", relational_operators[token + "="])
         tokens.pop(tokens.index(token) + 1)
         token += "="
         inp.append('o')
    elif(token=='<'):
        inp.append('o')
        print(token, ":", "less than")

    elif(token=='>' and tokens[tokens.index(token) + 1] == '='):
         print((token + "="), ":", relational_operators[token + "="])
         tokens.pop(tokens.index(token) + 1)
         token += "="
         inp.append('o')
    elif(token=='>'):
         inp.append('o')
         print(token, ":", "greater than")
    elif(token=='\"'):
         inp.append('u')
```

```
            print(token, ":", "single Quote")
        elif(token==':'):
            inp.append('v')
            print(token, ":", "colon")
        elif(token=='!' and tokens[tokens.index(token) + 1] == '=' ):
            print((token + "="), ":", relational_operators[token + "="])
            tokens.pop(tokens.index(token) + 1)
            token += "="
            inp.append('o')


        elif(token=='!'):
            inp.append('m')
            print(token, ":", "not operator")



        else:
            inp.append('m')
            print(token, ":", "special symbol")

    elif token in delimiter:
        inp.append('h')
        print(token, ":" , delimiter[token])

    elif token in datatype_keys:
        inp.append('a')
        print(token, ":", datatype[token])
        token = "datatype"

    elif token in keyword_keys:
        if(token=='return'):
            inp.append('p')
        elif(token=='switch'):
            inp.append('n')
        elif(token=='begin'):
            inp.append('b')
        elif(token=='end'):
            inp.append('d')
        elif(token=='case'):
            inp.append('q')
        elif(token=='break'):
            inp.append('s')
        elif(token=='printf'):
            inp.append('t')
        print(keyword[token])
```

```
                token = "keyword"

            elif token in numerals:
                inp.append('l')
                print(token, ": Numeric Value")
                token = "num"

            elif (token not in non_identifiers) and ('()' not in token):
                inp.append('j')
                print(token, ": Identifier")
                token = "id"

            token_list.append(token)
        dataFlag = False
        inp.append('c')
        print()
    print(token_list)
    print("Done with LEXICAL ANALYSIS")
    print(inp)
    print()

    f.close()
```

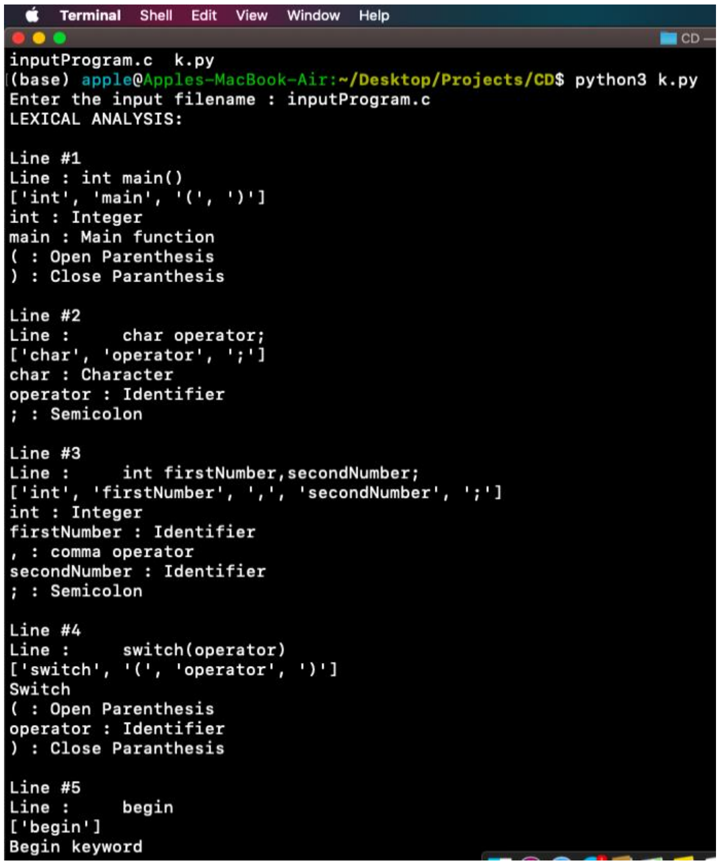
Explanation for the above code:

The above code demonstrates the code for lexical analysis. The code is written in python language. The input(question) is taken from a file called ‘inputProgram.c’.

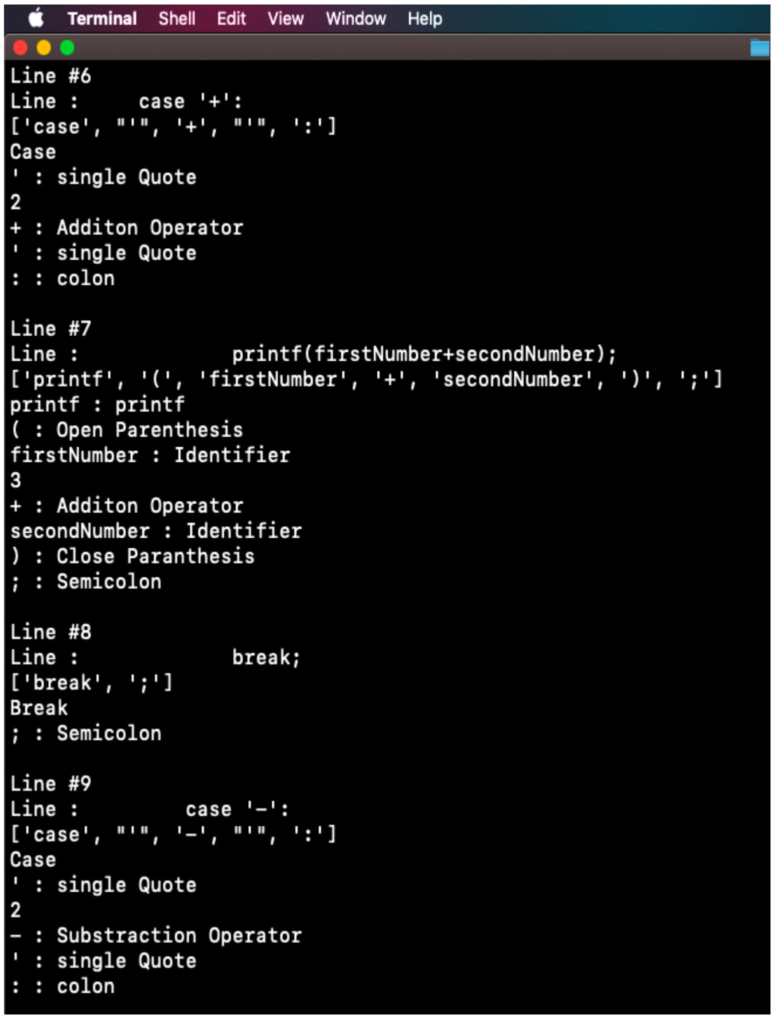
Each character is considered at a time from the input file and is characterized into tokens like keywords, identifiers, semicolon etc.

To get a better understanding of the code the output of lexical analysis for our question is displayed below.

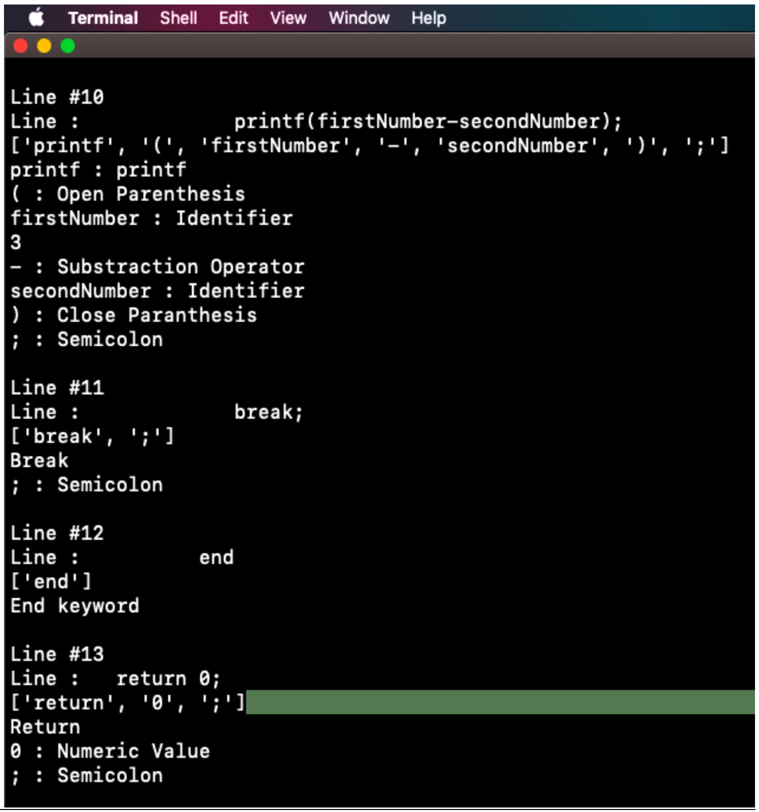
OUTPUT OF LEXICAL ANALYSIS



```
inputProgram.c  k.py
(base) apple@Apples-MacBook-Air:~/Desktop/Projects/CD$ python3 k.py
Enter the input filename : inputProgram.c
LEXICAL ANALYSIS:

Line #1
Line : int main()
['int', 'main', '(', ')']
int : Integer
main : Main function
( : Open Parenthesis
) : Close Paranthesis

Line #2
Line :     char operator;
['char', 'operator', ';']
char : Character
operator : Identifier
; : Semicolon

Line #3
Line :     int firstNumber,secondNumber;
['int', 'firstNumber', ',', 'secondNumber', ';']
int : Integer
firstNumber : Identifier
, : comma operator
secondNumber : Identifier
; : Semicolon

Line #4
Line :     switch(operator)
['switch', '(', 'operator', ')']
Switch
( : Open Parenthesis
operator : Identifier
) : Close Paranthesis

Line #5
Line :     begin
['begin']
Begin keyword
```



```
Line #6
Line :     case '+':
['case', "'", '+', "'", ':']
Case
' : single Quote
2
+ : Additon Operator
' : single Quote
: : colon

Line #7
Line :              printf(firstNumber+secondNumber);
['printf', '(', 'firstNumber', '+', 'secondNumber', ')', ';']
printf : printf
( : Open Parenthesis
firstNumber : Identifier
3
+ : Additon Operator
secondNumber : Identifier
) : Close Paranthesis
; : Semicolon

Line #8
Line :              break;
['break', ';']
Break
; : Semicolon

Line #9
Line :         case '-':
['case', "'", '-', "'", ':']
Case
' : single Quote
2
- : Substraction Operator
' : single Quote
: : colon
```



```
Line #10
Line :             printf(firstNumber-secondNumber);
['printf', '(', 'firstNumber', '-', 'secondNumber', ')', ';']
printf : printf
( : Open Parenthesis
firstNumber : Identifier
3
- : Substraction Operator
secondNumber : Identifier
) : Close Paranthesis
; : Semicolon

Line #11
Line :             break;
['break', ';']
Break
; : Semicolon

Line #12
Line :          end
['end']
End keyword

Line #13
Line :   return 0;
['return', '0', ';']
Return
0 : Numeric Value
; : Semicolon
```

SYNTAX ANALYSIS

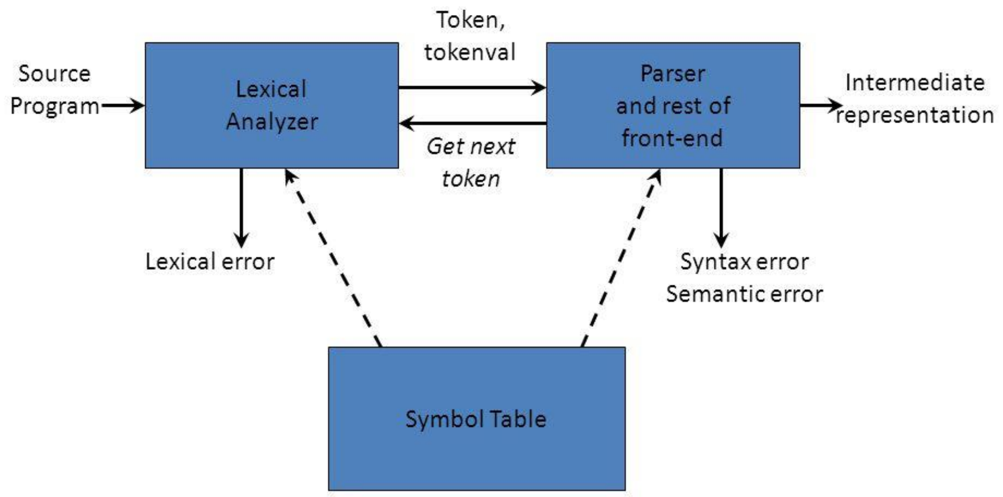
Syntax analysis or parsing is the second phase of a compiler. In this chapter, we shall learn the basic concepts used in the construction of a parser.

Role of the Parser:

In the compiler model, the parser obtains a string of tokens from the lexical analyser, and verifies that the string can be generated by the grammar for the source language.

The parser returns any syntax error for the source language.

Position of a Parser in the Compiler Model



3

- There are three general types' parsers for grammars.
- Universal parsing methods such as the Cocke-Younger-Kasami algorithm and Earley's algorithm can parse any grammar. These methods are too inefficient to use in production compilers.
- The methods commonly used in compilers are classified as either top-down parsing or bottom-up parsing.
- Top-down parsers build parse trees from the top (root) to the bottom (leaves).
- Bottom-up parsers build parse trees from the leaves and work up to the root.
- In both case input to the parser is scanned from left to right, one symbol at a time.
- The output of the parser is some representation of the parse tree for the stream of tokens.
- There are number of tasks that might be conducted during parsing. Such as;

- Collecting information about various tokens into the symbol table.
- Performing type checking and other kinds of semantic analysis.
- Generating intermediate code.
- Syntax Error Handling:
- Planning the error handling right from the start can both simplify the structure of a Compiler and improve its response to errors.
- The program can contain errors at many different levels. e.g.,
  - § Lexical – such as misspelling an identifier, keyword, or operator.
  - § Syntax – such as an arithmetic expression with unbalanced parenthesis.
  - § Semantic – such as an operator applied to an incompatible operand.
  - § Logical – such as an infinitely recursive call.

- Much of the error detection and recovery in a compiler is centered on the syntax analysis phase.
  - One reason for this is that many errors are syntactic in nature or are exposed when the stream of tokens coming from the lexical analyser disobeys the grammatical rules defining the programming language.
  - Another is the precision of modern parsing methods; they can detect the presence of syntactic errors in programs very efficiently.
- The error handler in a parser has simple goals:
  - It should the presence of errors clearly and accurately.
  - It should recover from each error quickly enough to be able to detect subsequent errors.
  - It should not significantly slow down the processing of correct programs.

Error-Recovery Strategies:

There are many different general strategies that a parser can employ to recover from a syntactic error:

1. Panic mode
2. Phrase level
3. Error production
4. Global correction

CONTEXT-FREE GRAMMAR

In this section, we will first see the definition of context-free grammar and introduce terminologies used in parsing technology.

A context-free grammar has four components:

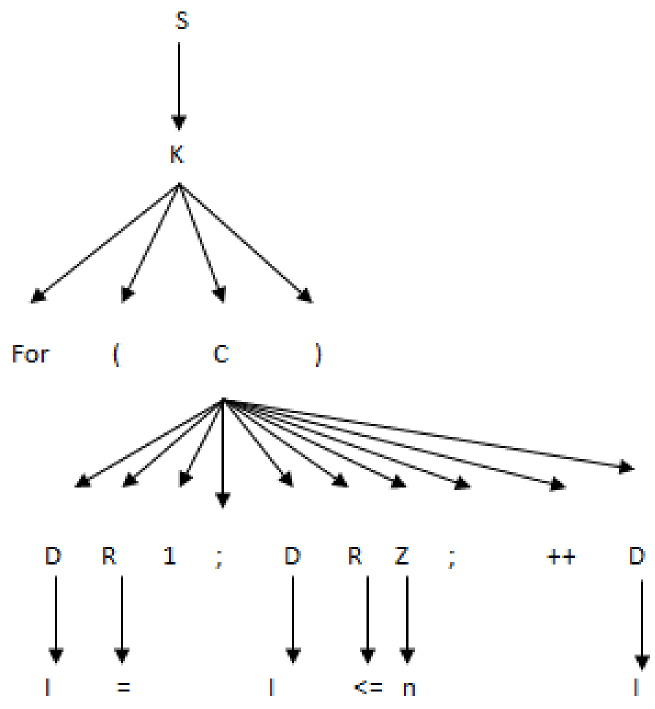
- A set of non-terminals (V). Non-terminals are syntactic variables that denote sets of strings. The non-terminals define sets of strings that help define the language generated by the grammar.
- A set of tokens, known as terminal symbols (Σ). Terminals are the basic symbols from which strings are formed.
- A set of productions (P). The productions of a grammar specify the manner in which the terminals and non-terminals can be combined to form strings. Each production consists of a non-terminal called the left side of the production, an arrow, and a sequence of tokens and/or on- terminals, called the right side of the production.
- One of the non-terminals is designated as the start symbol (S); from where the production begins. The strings are derived from the start symbol by repeatedly replacing a non-terminal (initially the start symbol) by the right side of a production, for that non-terminal.

PARSE TREE

A parse tree is a graphical depiction of a derivation. It is convenient to see how strings are derived from the start symbol. The start symbol of the derivation becomes the root of the parse tree. Let us see this by an example from the last topic.

We take the left-most derivation of a + b * c

The left-most derivation is:



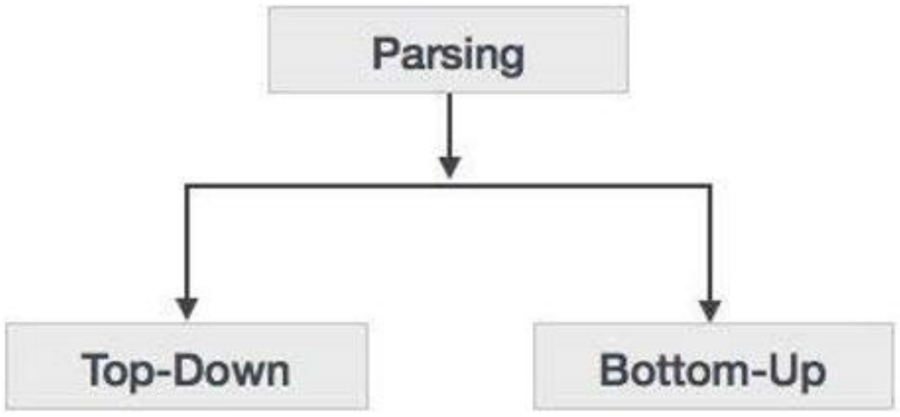
In a parse tree:

- All leaf nodes are terminals.
- All interior nodes are non-terminals.
- In-order traversal gives original input string.

A parse tree depicts associativity and precedence of operators. The deepest sub-tree is traversed first, therefore the operator in that sub-tree gets precedence over the operator which is in the parent nodes.

TYPES OF PARSERS

Syntax analyzers follow production rules defined by means of context-free grammar. The way the production rules are implemented (derivation) divides parsing into two types : top-down parsing and bottom-up parsing.



Top-down Parsing:

When the parser starts constructing the parse tree from the start symbol and then tries to transform the start symbol to the input, it is called top-down parsing.

 Recursive descent parsing: It is a common form of top-down parsing. It is called recursive, as it uses recursive procedures to process the input. Recursive descent parsing suffers from backtracking.

 Backtracking: It means, if one derivation of a production fails, the syntax analyzer restarts the process using different rules of same production. This technique may process the input string more than once to determine the right production.

<u>Bottom-up Parsing:</u>

As the name suggests, bottom-up parsing starts with the input symbols and tries to construct the parse tree up to the start symbol.

<u>NOTE: THE TYPE OF PARSER WE HAVE USED IS BOTTOM-UP PARSER</u>

<u>LL(1) Grammars:</u>

LL(1) GRAMMARS AND LANGUAGES. A context-free grammar G = (VT, VN, S, P) whose parsing table has no multiple entries is said to be LL(1). In the name LL(1),

- the first L stands for scanning the input from left to right,
- the second L stands for producing a leftmost derivation,
- and the 1 stands for using one input symbol of lookahead at each step to make parsing action decision.

A language is said to be LL(1) if it can be generated by a LL(1) grmmar. It can be shown that LL(1) grammars are :

- not ambiguous and
- not left-recursive.

GRAMMAR FOR THE PROBLEM STATEMENT

S’ -> S

S -> datatype MAINFUNC

MAINFUNC -> MAIN STMTS

MAIN -> main ( ) NL

STMTS -> STMT STMTS

STMTS -> STMT

STMT -> DECLARE SC NL

STMT -> SWITCHSTMT

STMT -> case

STMT -> printf (VARNUM operator VARNUM) SC NL

SWITCHSTMT -> switch (“’” condition “’”) SWITCH

SWITCH ->NL begin NL STMT end NL

DECLARE -> datatype DECVARS

DECVARS -> DECVAR

DECVARS -> DECVAR comma DECVARS

DECVAR -> id

DECVAR -> id equals_to VARNUM

VARNUM -> id

VARNUM -> number

RETURNSTMT -> return VARNUM SC NL

We have implemented CLR parsing(LR(1) parsing) as a parsing technique to parse the inputs in syntax analysis phase.

CLR PARSING

In CLR parsing(LR(1)parsing) we will be using LR(1) items. LR(k) item is defined to be an item using lookaheads of length k. So, the LR(1) item is comprised of two parts : the LR(0) item and the lookahead associated with the item.LR(1)parsers are more powerful parser.
For LR(1) items we modify the Closure and GOTO function.

Closure Operation:

```
Closure(I)
repeat
    for (each item [ A -> alpha.Bbeta, a ] in I )
        for (each production B -> gamma in G')
          for (each terminal b in FIRST(betaa))
            add [ B -> .gamma , b ] to set I;
until no more items are added to I;
return I;
```

Goto Operation:

```
Goto(I, X)
begin
 Initialise J to be the empty set;
 for ( each item A -> alpha.Xbeta, a ] in I )
    Add item A -> alphaX.beta, a ] to se J
 return Closure(J)
end
```

LR(1) items:

```
Void items(G')
begin
Initialise C to { closure ({[S' -> .S, $]})};
Repeat
   For (each set of items I in C)
      For (each grammar symbol X)
         if( Goto(I, X) is not empty and not in C)
            Add Goto(I, X) to C
until no new set of items are added to C
end
```

Construction of CLR parsing table:

Input – augmented grammar G'

1. Construct C = { I_0, I_1, ……. I_n} , the collection of sets of LR(0) items for G'.
2. State i is constructed from Ii. The parsing actions for state i are determined as follow :

   i) If [A -> alpha.abeta, b] is in I_i and Goto(I_i , a) = I_j, then set action[i, a] to "shift j". Here 'a' must be terminal.

   ii) If [A -> alpha. , a] is in I_i , A ≠ S', then set action[i, a] to "reduce A -> alpha".

   iii) Is [S' -> S. , $] is in I_i, then set action[i, $] to "accept".

   If any conflicting actions are generated by the above rules we say that the grammar is not CLR.
3. The goto transitions for state i are constructed for all non-terminals A using the rule: if GOTO(Ii, A) = Ij then GOTO [i, A] = j.
4. All entries not defined by rules 2 and 3 are made error.
5. The initial state of the parser is the one constructed from the set containing item S' -> .S,$

Note: If a state has two reductions and both have same lookahead then it will in multiple entries in parsing table thus a conflict. If a state has one reduction and their is a shift from that state on a terminal same as the lookahead of the reduction then it will lead to multiple entries in parsing table thus a conflict.

PARSER CODE

CLR parsing(a bottom up parsing technique) is implemented in the below code

```
st=[]
class Stack:
 def __init__(self):
    self.s = []
    self.st = []
 def pop(self):
    return self.s.pop()
 def push(self, item):
    self.s.append(item)
 def sizeOfStack(self):
    return len(self.s)
 def peak(self):
    return self.s[len(self.s)-1]
 def printStack(self,nonterminals,terminals,t,string22,str1):
    for i in range(len(self.s)):
        if(self.s[i] in terminals):
            self.st.append(terminals[self.s[i]])
        elif(self.s[i] in nonterminals):
            self.st.append(nonterminals[self.s[i]])
        else:
            self.st.append(self.s[i])
    #print(self.st,end=" ")
    str2 = ' '.join(map(str, self.st))
    self.st=[]
    t.add_row([str2,str1,string22])





inpoot=[]
def foo(string ):
 global  index , inpIndex,inp,s,t,nonterminals,terminals,inpoot,temp



 if string[0] == 'r': # do reducing
        temp1 = RulesTable[int(string[1:])]
        temp1 = temp1.split("->")
        g = temp1[1].split(" ")
        for i in range( len(g)*2 ):
            label = s.pop()
        num  = s.peak()
        s.push(temp1[0])
        s.push(int(LR1Table[num][goto[temp1[0]]]))
        index = int(LR1Table[num][goto[temp1[0]]])
        #s.printStack(t,inp,string)
        #s.printStack(nonterminals,terminals,string)
        string22=LR1Table[index][action[inp[inpIndex]]]
        #print(k)
        for l in range(inpIndex,len(inp)):
         inpoot.append(nonterminals[inp[l]])
        #print(string,end=" ")
        str1 =' '.join(map(str, inpoot))
        #print(str1,end=" ")
        #print()
        inpoot=[]
        s.printStack(nonterminals,terminals,t,string22,str1)
 else: # do shifting

        s.push(inp[inpIndex])
        s.push(int(string[1:]))
        inpIndex+=1
        index = s.peak()
        string22=LR1Table[index][action[inp[inpIndex]]]
        for l in range(inpIndex,len(inp)):
          inpoot.append(nonterminals[inp[l]])
        str1 = ""
        str1 = ' '.join(map(str, inpoot))
        inpoot=[]
        s.printStack(nonterminals,terminals,t,string22,str1)

index = 0
inpIndex = 0  # input pointer
nodeNum = 0  # number of node for graphviz
flage =1

s = Stack()
file_name = input("Enter the input filename : ")
tokenize(file_name)
print()

headers=
['States','datatype','begin','NL','end','main','(',')','SC','comma','id','equals_to','number','operator','switch','con
dition','return','case','break','printf','$','S\'','S','MAINFUNC','MAIN','STMTS','STMT','DECLARE','DECV
ARS','DECVAR','EXPRESSION','VARNUM','WHILESTMT','WSTMT','RETURNSTMT']


table=[['0','s2','','','','','','','','','','','','','','','','','','','','','','1','','','','','','','','','','','',''],
['1','','','','','','','','','','','','','','','','','','','','','acc','','','','','','','','','','','','',''],
['2','','s5','','','','','','','','','','','','','','','','','','','','','','3','4','','','','','','','','',''],
['3','','','','','','','','','','','','','','','','','','','','','r1','','','','','','','','','','','','',''],
['4','s14','','','','','','','','','','s15','','','s16','s10','','','','s11','s12','','','','','','6','7','8','','','','9','','13'],
['5','','','s17','','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['6','','','','','','','','','','','','','','','','','','','','','r2','','','','','','','','','','','','',''],
['7','s14','','','','','','','','','','s15','','','s16','s10','','','','s11','s12','r5','','','','','18','7','8','','','','9','','13'],
['8','','','','','','s19','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['9','r7','','','','','','','','','','r7','','','r7','r7','','','','r7','r7','r7','','','','','','','','','','','','',''],
['10','','','','','','','','','','','','','','','','s20','','','','','','','','','','','','','','','','','',''],
['11','','','s21','','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['12','','','','','','s22','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['13','r21','','','','','','','','','','r21','','','r21','r21','','','','r21','r21','r21','','','','','','','','','','','','',''],
['14','','','','','','','','s25','','','','','','','','','','','','','','','','','','','','','23','24','','','',''],
['15','','','s26','','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['16','','','','','','','','s28','','s29','','','','','','','','','','','','','','','','','','','','','27','','',''],
['17','','','','s30','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['18','','','','','','','','','','','','','','','','','','','','','r4','','','','','','','','','','','','',''],
['19','','','','','s31','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['20','','','','','','','','','','','','','','','','','s32','','','','','','','','','','','','','','','','',''],
['21','','','','','','','','s34','','s35','','','','','','','','','','','','','','','','','','','','','33','','',''],
['22','','','','','s36','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['23','','','','','','r8','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['24','','','','','','r9','s37','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['25','','','','','','r11','r11','','s38','','','','','','','','','','','','','','','','','','','','','','','','',''],
['26','','','','','','','','s39','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['27','','','','','','s40','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['28','','','','','','r13','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['29','','','','','','r14','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['30','','','','','s41','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['31','r6','','','','','','','','','','r6','','','r6','r6','','','','r6','r6','r6','','','','','','','','','','','','',''],
['32','','','','','','','','','','','','','','','','s42','','','','','','','','','','','','','','','','','',''],
['33','','','','','','','','','','','','','','','','','s43','','','','','','','','','','','','','','','','',''],
['34','','','','','','','','','','','','','','','','','r13','','','','','','','','','','','','','','','','',''],
['35','','','','','','','','','','','','','','','','','r14','','','','','','','','','','','','','','','','',''],
['36','r20','','','','','','','','','','r20','','','r20','r20','','','','r20','r20','r20','','','','','','','','','','','','',''],
['37','','','','','','','','s25','','','','','','','','','','','','','','','','','','','','','44','24','','','',''],
['38','','','','','','','','s46','','s47','','','','','','','','','','','','','','','','','','','','','45','','',''],
['39','','','','s48','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['40','','','','','s49','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['41','r3','','','','','','','','','','r3','','','r3','r3','','','','r3','r3','','','','','','','','','','','','','',''],
['42','','','','','','','','','','','','','','','','','','s50','','','','','','','','','','','','','','','',''],
['43','','','','','','','','s52','','s53','','','','','','','','','','','','','','','','','','','','','51','','',''],
['44','','','','','','r10','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['45','','','','','','r12','r12','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['46','','','','','','r13','r13','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['47','','','','','','r14','r14','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['48','','','','','s55','','','','','','','','','','','','','','','','','','','','','','','','','','','','54',''],
['49','r17','','','','','','','','','','r17','','','r17','r17','','','','r17','r17','r17','','','','','','','','','','','','',''],
['50','','','','','s56','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['51','','','','s57','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['52','','','','r13','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['53','','','','r14','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['54','r15','','','','','','','','','','r15','','','r15','r15','','','','r15','15','r15','','','','','','','','','','','','',''],
['55','','','','','','','','','','','','s58','','','','','','','','','','','','','','','','','','','','','',''],
['56','r18','','','','','','','','','','r18','','','r18','r18','','','','r18','18','r18','','','','','','','','','','','','',''],
['57','','','','','','s59','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['58','','','','','s60','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['59','','','','','s61','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['60','s14','','','','','','','','','','s70','','','s71','s66','','','','s67','s68','','','','','','62','63','64','','','','65','','69'],
['61','r19','','','','','','','','','','r19','','','r19','r19','','','','r19','r19','r19','','','','','','','','','','','','',''],
['62','','','','','','','','','','','','','s72','','','','','','','','','','','','','','','','','','','','',''],
['63','s14','','','','','','','','','','s70','','r5','s71','s66','','','','s67','s68','','','','','','73','63','64','','','','65','','69'],
['64','','','','','','s74','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['65','r7','','','','','','','','','','r7','','r7','r7','r7','','','','r7','r7','','','','','','','','','','','','','',''],
['66','','','','','','','','','','','','','','','','s75','','','','','','','','','','','','','','','','','',''],
['67','','','s76','','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['68','','','','','','s77','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['69','r21','','','','','','','','','','r21','','r21','r21','r21','','','','r21','r21','','','','','','','','','','','','','',''],
['70','','','s78','','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['71','','','','','','','','s28','','s29','','','','','','','','','','','','','','','','','','','','','79','','',''],
['72','','','','','s80','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['73','','','','','','','','','','','','','r4','','','','','','','','','','','','','','','','','','','','',''],
['74','','','','','s81','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['75','','','','','','','','','','','','','','','','','s82','','','','','','','','','','','','','','','','',''],
['76','','','','','','','','s34','','s35','','','','','','','','','','','','','','','','','','','','','83','','',''],
['77','','','','','s84','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['78','','','','','','','','s85','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['79','','','','','','s86','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['80','r16','','','','','','','','','','r16','','','r16','r16','','','','r16','r16','r16','','','','','','','','','','','','',''],
['81','r6','','','','','','','','','','r6','','r6','r6','r6','','','','r6','r6','','','','','','','','','','','','','',''],
['82','','','','','','','','','','','','','','','','s87','','','','','','','','','','','','','','','','','',''],
['83','','','','','','','','','','','','','','','','','s88','','','','','','','','','','','','','','','','',''],
['84','r20','','','','','','','','','','r20','','r20','r20','r20','','','','r20','r20','','','','','','','','','','','','','',''],
['85','','','','s89','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['86','','','','','s90','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['87','','','','','','','','','','','','','','','','','','s91','','','','','','','','','','','','','','','',''],
['88','','','','','','','','s52','','s53','','','','','','','','','','','','','','','','','','','','','92','','',''],
['89','','','','','s94','','','','','','','','','','','','','','','','','','','','','','','','','','','','93',''],
['90','r17','','','','','','','','','','r17','','r17','r17','r17','','','','r17','r17','','','','','','','','','','','','','',''],
['91','','','','','s95','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['92','','','','s96','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['93','r15','','','','','','','','','','r15','','r15','r15','r15','','','','r15','r15','','','','','','','','','','','','','',''],
['94','','','','','','','','','','','','s97','','','','','','','','','','','','','','','','','','','','','',''],
['95','r18','','','','','','','','','','r18','','r18','r18','r18','','','','r18','r18','','','','','','','','','','','','','',''],
['96','','','','','','s98','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['97','','','','','s99','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['98','','','','','s100','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['99','s14','','','','','','','','','','s70','','','s71','s66','','','','s67','s68','','','','','','101','63','64','','','','65','','69'],
['100','r19','','','','','','','','','','r19','','r19','r19','r19','','','','r19','r19','','','','','','','','','','','','','',''],
['101','','','','','','','','','','','','','s102','','','','','','','','','','','','','','','','','','','','',''],
['102','','','','','s103','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''],
['103','r16','','','','','','','','','','r16','','r16','r16','r16','','','','r16','r16','','','','','','','','','','','','','','']]




print(tabulate(table, headers,tablefmt='grid'))






inp += "$"
#print(inp)
terminals={'S\'':'S\'','S':'S','A':'MAINFUNC','B':'MAIN','C':'STMTS','D':'STMT','E':'DECLARE','F':'EXPRESSION','G':'SWITCHSTMT','H':'RETURSTMT','I':'DECVARS','J':'DECVAR','K':'VARNUM','L':'SWTCH'}
nonterminals={'a':'datatype','b':'begin','c':'NL','d':'end','e':'main','f':'(','g':')','h':'SC','i':'comma','j':'id','k':'equals_to','l':'number','m':'operator','n':'switch','o':'condition','p':'return','q':'case','s':'break','t':'printf','u':'\'','v':':','$':'$'}
action =
{'a':0,'e':1,'f':2,'g':3,'c':4,'h':5,'i':6,'j':7,'k':8,'l':9,'n':10,'b':11,'d':12,'p':13,'q':14,'u':15,'m':16,'v':17,'t':18,'s':19,'$':20}
goto ={'S\'':21 ,'S':22,'A':23,'B':24,'C':25,'D':26,'E':27,'I':28,'J':29,'K':30,'G':31,'L':32,'H':33}


LR1Table = []
LR1Table.append(['s2','','','','','','','','','','','','','','','','','','','','','','1','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','','','','','','acc','','','','','','','','','','','','',''])
LR1Table.append(['','s5','','','','','','','','','','','','','','','','','','','','','','3','4','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','','','','','','r1','','','','','','','','','','','','',''])
LR1Table.append(['s14','','','','','','','','','','s15','','','s16','s10','','','','s11','s12','','','','','','6','7','8','','','','9','','13'])
LR1Table.append(['','','s17','','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','','','','','','r2','','','','','','','','','','','','',''])
LR1Table.append(['s14','','','','','','','','','','s15','','','s16','s10','','','','s11','s12','r5','','','','','18','7','8','','','','9','','13'])
LR1Table.append(['','','','','','s19','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['r7','','','','','','','','','','r7','','','r7','r7','','','','r7','r7','r7','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','s20','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','s21','','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','s22','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['r21','','','','','','','','','','r21','','','r21','r21','','','','r21','r21','r21','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','s25','','','','','','','','','','','','','','','','','','','','','23','24','','','',''])
LR1Table.append(['','','s26','','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','s28','','s29','','','','','','','','','','','','','','','','','','','','','27','','',''])
LR1Table.append(['','','','s30','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','','','','','','r4','','','','','','','','','','','','',''])
LR1Table.append(['','','','','s31','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','','s32','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','s34','','s35','','','','','','','','','','','','','','','','','','','','','33','','',''])
LR1Table.append(['','','','','s36','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
```

```
LR1Table.append(['','','','','','r8','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','r9','s37','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','r11','r11','','s38','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','s39','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','s40','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','r13','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','r14','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','s41','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['r6','','','','','','','','','','r6','','','r6','r6','','','','r6','r6','r6','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','s42','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','','s43','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','','r13','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','','r14','','','','','','','','','','','','','','','','',''])
LR1Table.append(['r20','','','','','','','','','','r20','','','r20','r20','','','','r20','r20','r20','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','s25','','','','','','','','','','','','','','','','','','','','','44','24','','','',''])
LR1Table.append(['','','','','','','','s46','','s47','','','','','','','','','','','','','','','','','','','','','45','','',''])
LR1Table.append(['','','','s48','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','s49','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['r3','','','','','','','','','','r3','','','r3','r3','','','','r3','r3','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','','','s50','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','s52','','s53','','','','','','','','','','','','','','','','','','','','','51','','',''])
LR1Table.append(['','','','','','r10','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','r12','r12','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','r13','r13','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','r14','r14','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','s55','','','','','','','','','','','','','','','','','','','','','','','','','','','','54',''])
LR1Table.append(['r17','','','','','','','','','','r17','','','r17','r17','','','','r17','r17','r17','','','','','','','','','','','','',''])
LR1Table.append(['','','','','s56','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','s57','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','r13','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','r14','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['r15','','','','','','','','','','r15','','','r15','r15','','','','r15','15','r15','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','s58','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['r18','','','','','','','','','','r18','','','r18','r18','','','','r18','18','r18','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','s59','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','s60','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','s61','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['s14','','','','','','','','','','s70','','','s71','s66','','','','s67','s68','','','','','','62','63','64','','','','65','','69'])
LR1Table.append(['r19','','','','','','','','','','r19','','','r19','r19','','','','r19','r19','r19','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','s72','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['s14','','','','','','','','','','s70','','r5','s71','s66','','','','s67','s68','','','','','','73','63','64','','','','65','','69'
])
LR1Table.append(['','','','','','s74','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['r7','','','','','','','','','','r7','','r7','r7','r7','','','','r7','r7','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','s75','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','s76','','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','s77','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['r21','','','','','','','','','','r21','','r21','r21','r21','','','','r21','r21','','','','','','','','','','','','','',''])
LR1Table.append(['','','s78','','','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','s28','','s29','','','','','','','','','','','','','','','','','','','','','79','','',''])
LR1Table.append(['','','','','s80','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
```

```
LR1Table.append(['','','','','','','','','','','','','r4','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','s81','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','','s82','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','s34','','s35','','','','','','','','','','','','','','','','','','','','','83','',''])
LR1Table.append(['','','','','s84','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','s85','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','s86','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['r16','','','','','','','','','r16','','','r16','r16','','','','r16','r16','r16','','','','','','','','','','','','',''])
LR1Table.append(['r6','','','','','','','','','r6','','r6','r6','r6','','','','r6','r6','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','s87','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','','s88','','','','','','','','','','','','','','','',''])
LR1Table.append(['r20','','','','','','','','','r20','','r20','r20','r20','','','','r20','r20','','','','','','','','','','','','','',''])
LR1Table.append(['','','','s89','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','s90','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','','','','','','s91','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','s52','','s53','','','','','','','','','','','','','','','','','','','','','92','',''])
LR1Table.append(['','','','','s94','','','','','','','','','','','','','','','','','','','','','','','','','','','93',''])
LR1Table.append(['r17','','','','','','','','','r17','','r17','r17','r17','','','','r17','r17','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','s95','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','s96','','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['r15','','','','','','','','','r15','','r15','r15','r15','','','','r15','r15','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','s97','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['r18','','','','','','','','','r18','','r18','r18','r18','','','','r18','r18','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','s98','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','s99','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','s100','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['s14','','','','','','','','','s70','','','s71','s66','','','','s67','s68','','','','','','101','63','64','','','','65','','69']
)
LR1Table.append(['r19','','','','','','','','','r19','','r19','r19','r19','','','','r19','r19','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','','','','','','','','','s102','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['','','','','s103','','','','','','','','','','','','','','','','','','','','','','','','','','','',''])
LR1Table.append(['r16','','','','','','','','','r16','','r16','r16','r16','','','','r16','r16','','','','','','','','','','','','','','',''])




RulesTable = []
RulesTable.append("S'->S")
RulesTable.append("S->a A")
RulesTable.append("A->B C")
RulesTable.append("B->e f g c")
RulesTable.append("C->D C")
RulesTable.append("C->D")
RulesTable.append("D->E h c")
RulesTable.append("D->G")
RulesTable.append("E->a I")
RulesTable.append("I->J")
RulesTable.append("I->J i I")
RulesTable.append("J->j")
RulesTable.append("J->j k K")
RulesTable.append("K->j")
RulesTable.append("K->l")
RulesTable.append("G->n f j g L")
RulesTable.append("L->c b c C d c")
RulesTable.append("H->p K h c")
RulesTable.append("D->q u m u v c")
RulesTable.append("D->t f K m K g h c")
RulesTable.append("D->s h c")
RulesTable.append("D->H")



s.push(0)
temp  = LR1Table[index][action[inp[inpIndex]]]
print()
print()
print("SEQUENTIAL PARSING STEPS:")
print()
t = tt.Texttable()
headings = ['Stack','Input','Action']
t.header(headings)

cd=[]
for l in range(0,len(inp)):
        cd.append(nonterminals[inp[l]])
cdstr = " "
cdstr = ' '.join(map(str, cd))
t.add_row(['0',cdstr,temp])



flag=0
#print(temp)
while(temp != "acc"):
 if temp == '':
    #print("ERROR")
    flag=1
    break
 foo(temp)
 temp  = LR1Table[index][action[inp[inpIndex]]]

s = t.draw()
print(s)
#print(temp)
if(flag==1):
    print()
    print("ERROR!! The input doesnt satisfy the grammar")
```

Explanation for the above code:

The above parser code is written in python language.

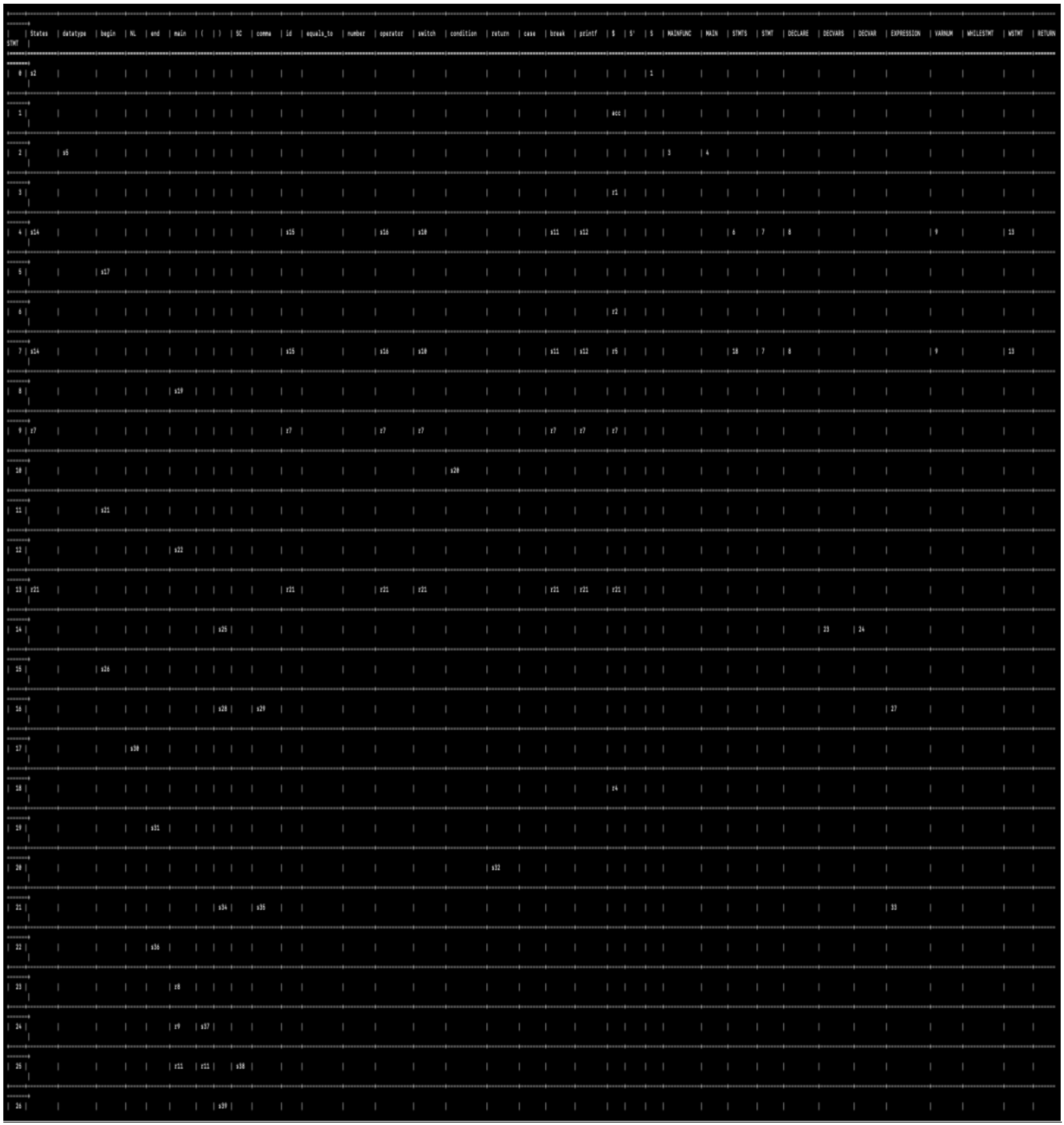
The output of lexical analysis are a set of tokens. In the above code the grammar for the problem statement and the parsing table is defined. CLR parsing(LR(1) parsing) which is a bottom-up technique is used for implementing the parsing logic.

The input to the parsing phase is the output of lexical analysis phase ie the set of tokens.
Using the parsing table and grammar this input is parsed by implementing CLR parsing.

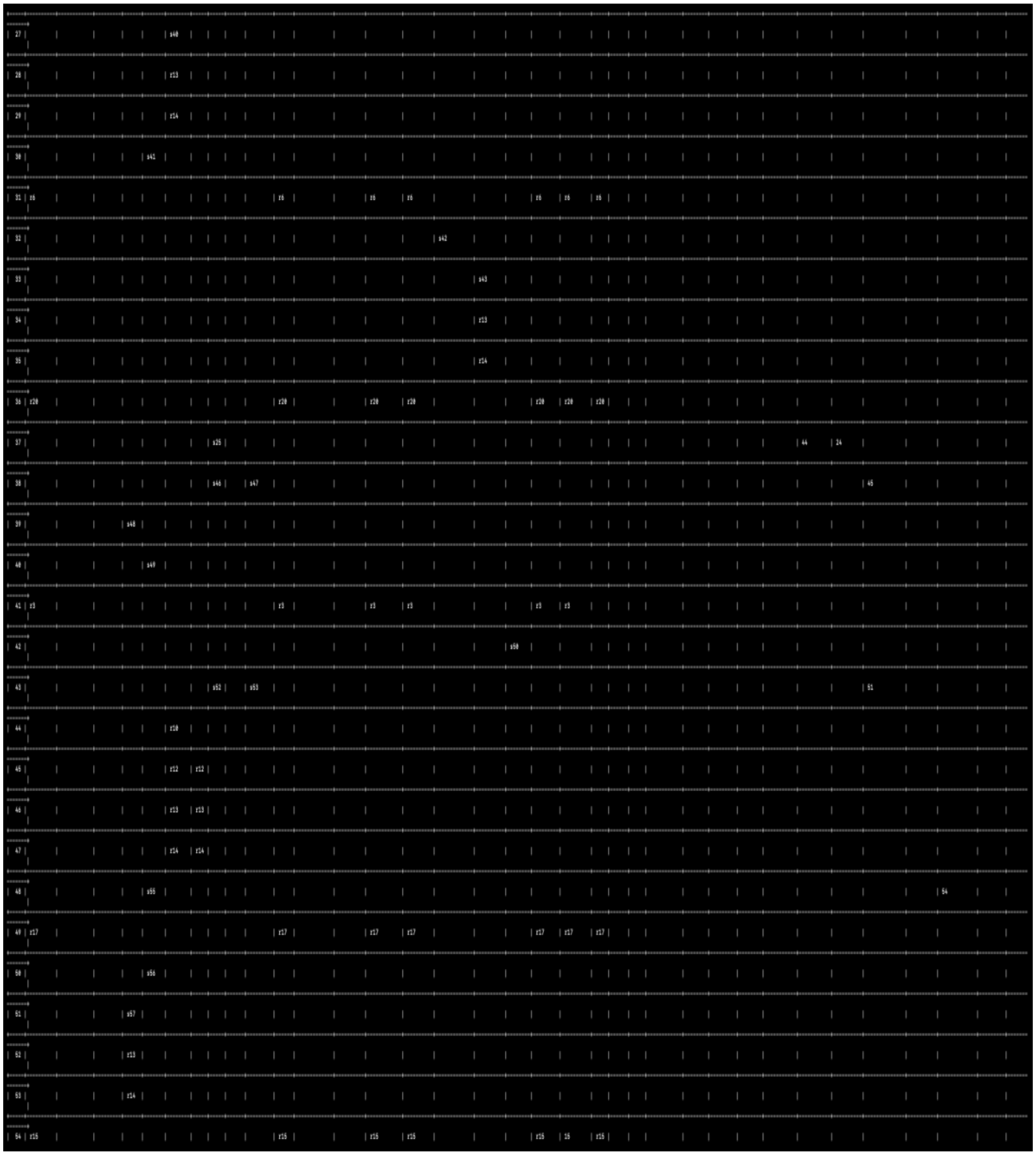
The input will be accepted by the grammar at the end of this phase if the input satisfies the grammar or else it will not be accepted.

OUTPUT OF SYNTAX ANALYSIS

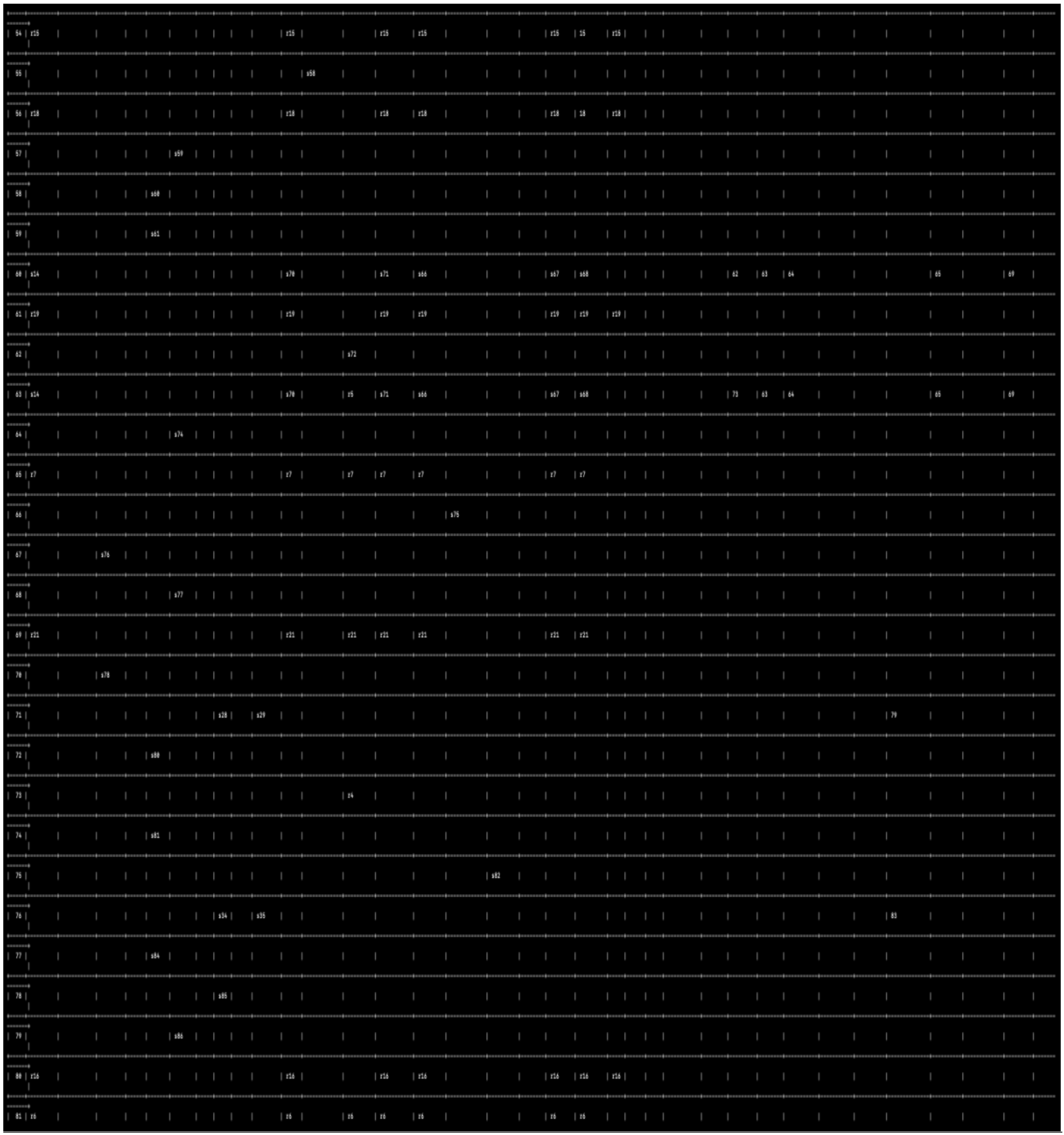
LR(1) PARSIG TABLE



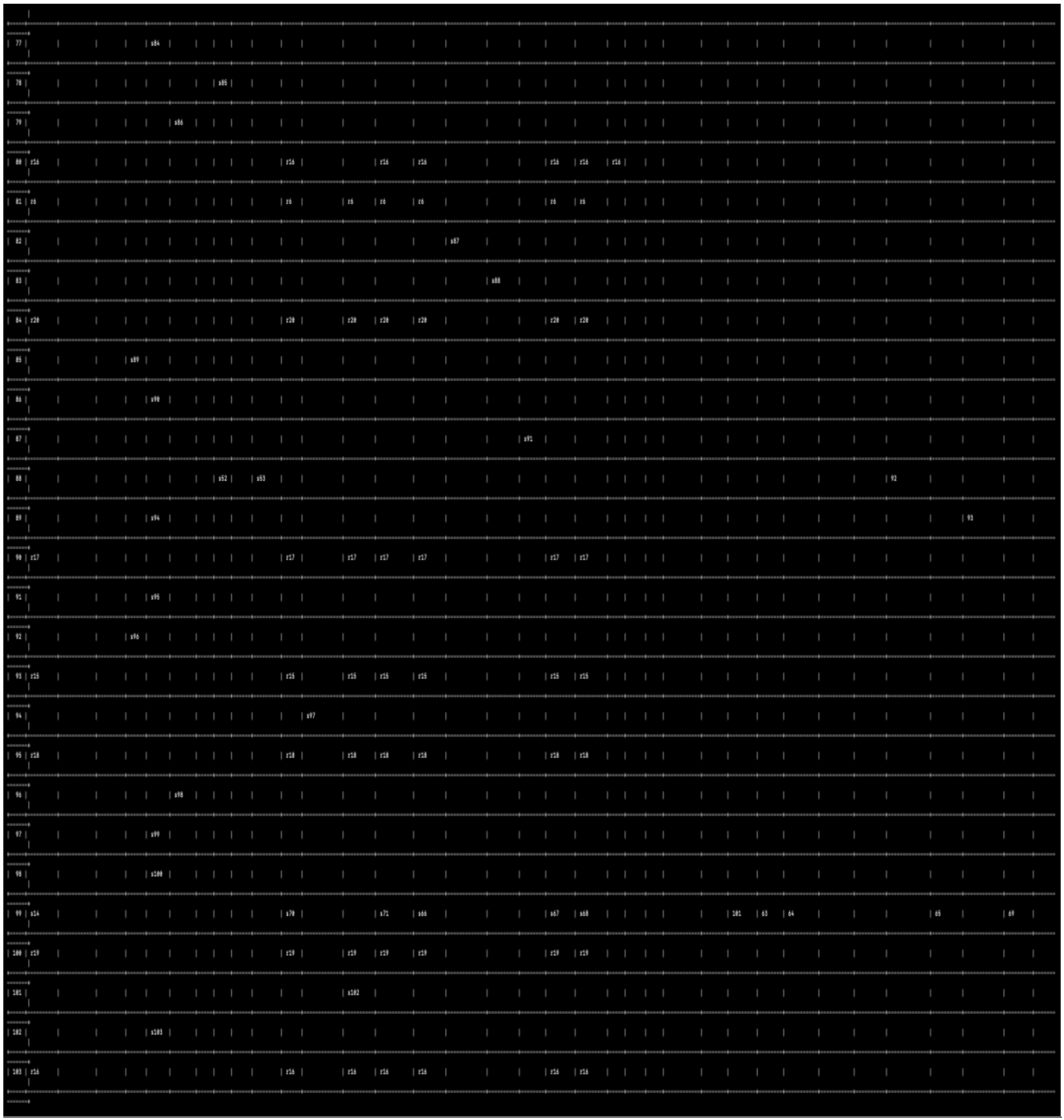
	States	datatype	begin	NL	end	main	(	)	SC	comma	id	equals_to	number	operator	switch	condition	return	case	break	printf	$	S'	S	MAINFUNC	MAIN	STMTS	STMT	DECLARE	DECVARS	DECVAR	EXPRESSION	VARNUM	WHILESTMT	WSTMT	RETURN STMT
0	s2																						1												
1																					acc														
2		s5																						3	4										
3																					r1														
4	s14										s15			s16	s10				s11	s12						6	7	8				9		13	
5			s17																																
6																					r2														
7	s14										s15			s16	s10				s11	s12	r5					18	7	8				9		13	
8						s19																													
9	r7										r7			r7	r7				r7	r7	r7														
10																s20																			
11			s21																																
12						s22																													
13	r21										r21			r21	r21				r21	r21	r21														
14								s25																					23	24					
15			s26																																
16								s28		s29																					27				
17				s30																															
18																					r4														
19					s31																														
20																	s32																		
21								s34		s35																					33				
22					s36																														
23						r8																													
24						r9	s37																												
25						r11	r11		s38																										
26								s39																											



27						s40																													
28						r13																													
29						r14																													
30					s41																														
31	r6										r6			r6	r6				r6	r6	r6														
32																s42																			
33																	s43																		
34																	r13																		
35																	r14																		
36	r20										r20			r20	r20				r20	r20	r20														
37								s25																					44	24					
38								s46		s47																					45				
39				s48																															
40					s49																														
41	r3										r3			r3	r3				r3	r3															
42																		s50																	
43								s52		s53																					51				
44						r10																													
45						r12	r12																												
46						r13	r13																												
47						r14	r14																												
48					s55																												54		
49	r17										r17			r17	r17				r17	r17	r17														
50					s56																														
51				s57																															
52				r13																															
53				r14																															
54	r15										r15			r15	r15				r15	15	r15														

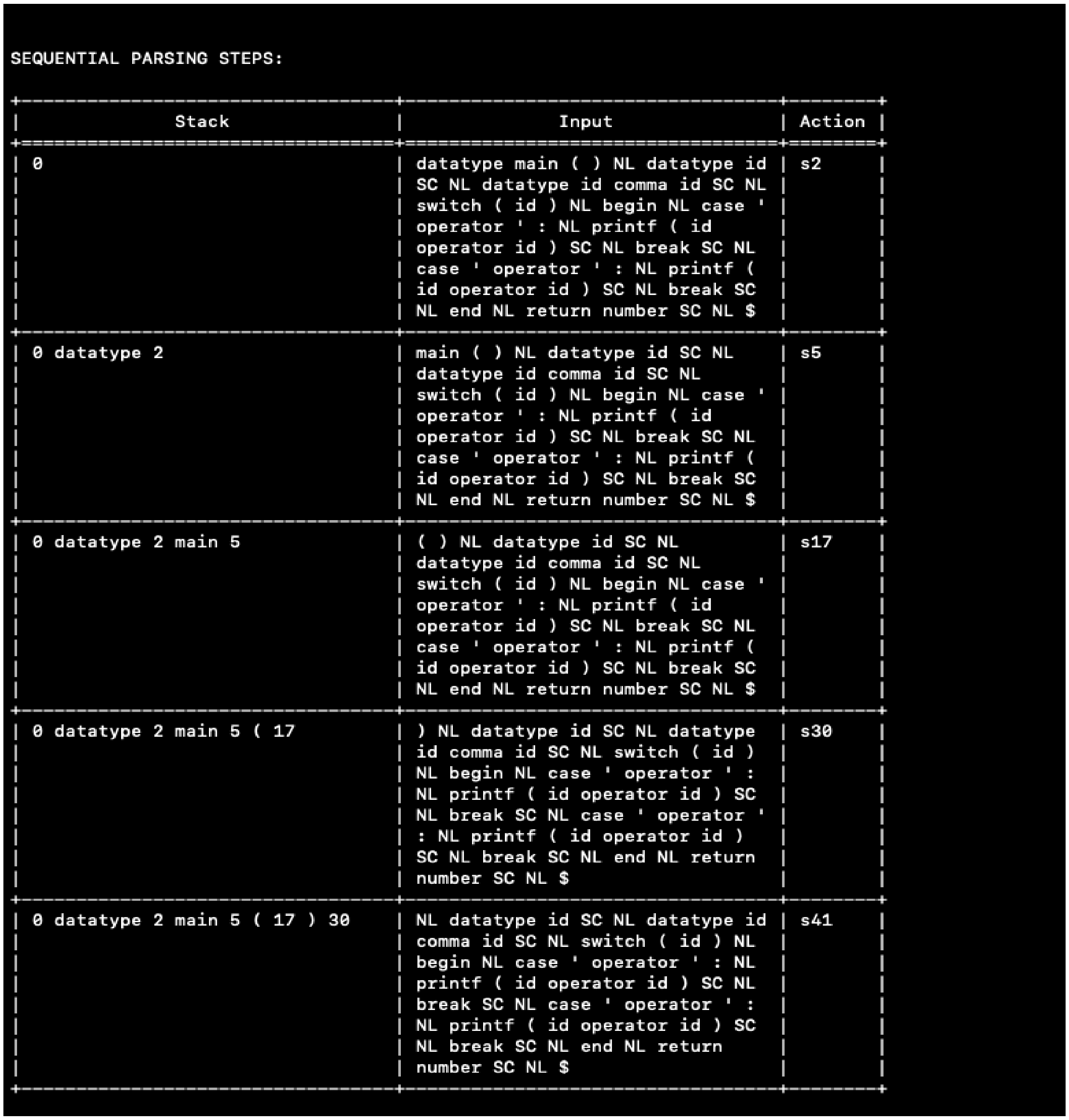


54	r15										r15			r15	r15				r15	15	r15													
55												s58																						
56	r18										r18			r18	r18				r18	18	r18													
57						s59																												
58					s60																													
59					s61																													
60	s14										s70			s71	s66				s67	s68						62	63	64				65		69
61	r19										r19			r19	r19				r19	r19	r19													
62													s72																					
63	s14										s70		r5	s71	s66				s67	s68						73	63	64				65		69
64						s74																												
65	r7										r7		r7	r7	r7				r7	r7														
66																s75																		
67			s76																															
68						s77																												
69	r21										r21		r21	r21	r21				r21	r21														
70			s78																															
71								s28		s29																					79			
72					s80																													
73													r4																					
74					s81																													
75																	s82																	
76								s34		s35																					83			
77					s84																													
78								s85																										
79						s86																												
80	r16										r16			r16	r16				r16	r16	r16													
81	r6										r6		r6	r6	r6				r6	r6														

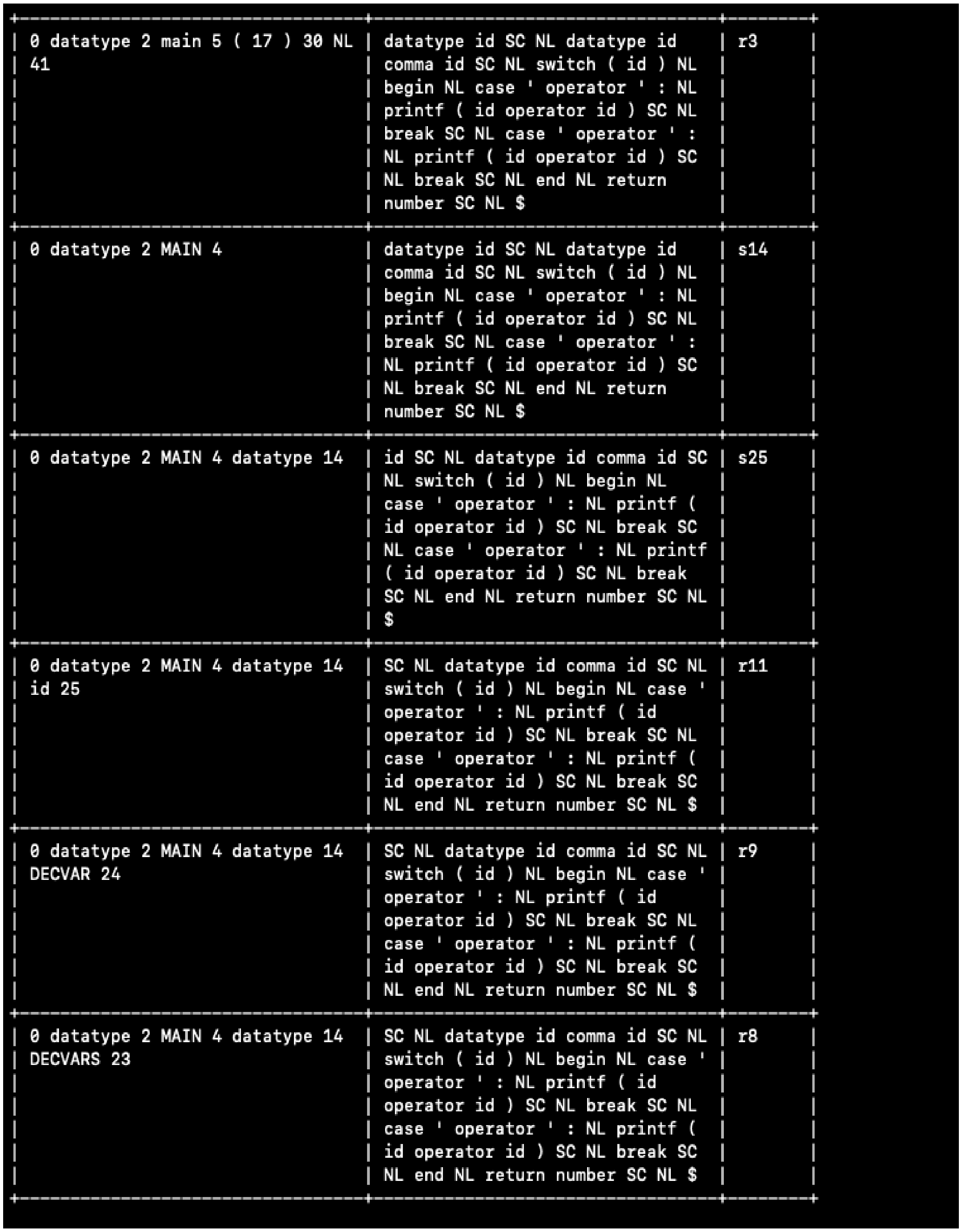


```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 77  |     |     |     |     | s84 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 78  |     |     |     |     |     |     |     | s85 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 79  |     |     |     |     |     | s86 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 80  | r16 |     |     |     |     |     |     |     |     |     | r16 |     |     | r16 | r16 |     |     |     | r16 | r16 | r16 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 81  | r6  |     |     |     |     |     |     |     |     |     | r6  |     | r6  | r6  | r6  |     |     |     | r6  | r6  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 82  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | s87 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 83  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | s88 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 84  | r20 |     |     |     |     |     |     |     |     |     | r20 |     | r20 | r20 | r20 |     |     |     | r20 | r20 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 85  |     |     |     | s89 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 86  |     |     |     |     | s90 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 87  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | s91 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 88  |     |     |     |     |     |     |     | s52 |     | s53 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 92  |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 89  |     |     |     |     | s94 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 93  |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 90  | r17 |     |     |     |     |     |     |     |     |     | r17 |     | r17 | r17 | r17 |     |     |     | r17 | r17 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 91  |     |     |     |     | s95 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 92  |     |     |     | s96 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 93  | r15 |     |     |     |     |     |     |     |     |     | r15 |     | r15 | r15 | r15 |     |     |     | r15 | r15 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 94  |     |     |     |     |     |     |     |     |     |     |     | s97 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 95  | r18 |     |     |     |     |     |     |     |     |     | r18 |     | r18 | r18 | r18 |     |     |     | r18 | r18 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 96  |     |     |     |     |     | s98 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 97  |     |     |     |     | s99 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 98  |     |     |     |     | s100|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 99  | s14 |     |     |     |     |     |     |     |     |     | s70 |     |     | s71 | s66 |     |     |     | s67 | s68 |     |     |     |     |     | 101 | 63  | 64  |     |     |     | 65  |     | 69  |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 100 | r19 |     |     |     |     |     |     |     |     |     | r19 |     | r19 | r19 | r19 |     |     |     | r19 | r19 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 101 |     |     |     |     |     |     |     |     |     |     |     |     | s102|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 102 |     |     |     |     | s103|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 103 | r16 |     |     |     |     |     |     |     |     |     | r16 |     | r16 | r16 | r16 |     |     |     | r16 | r16 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

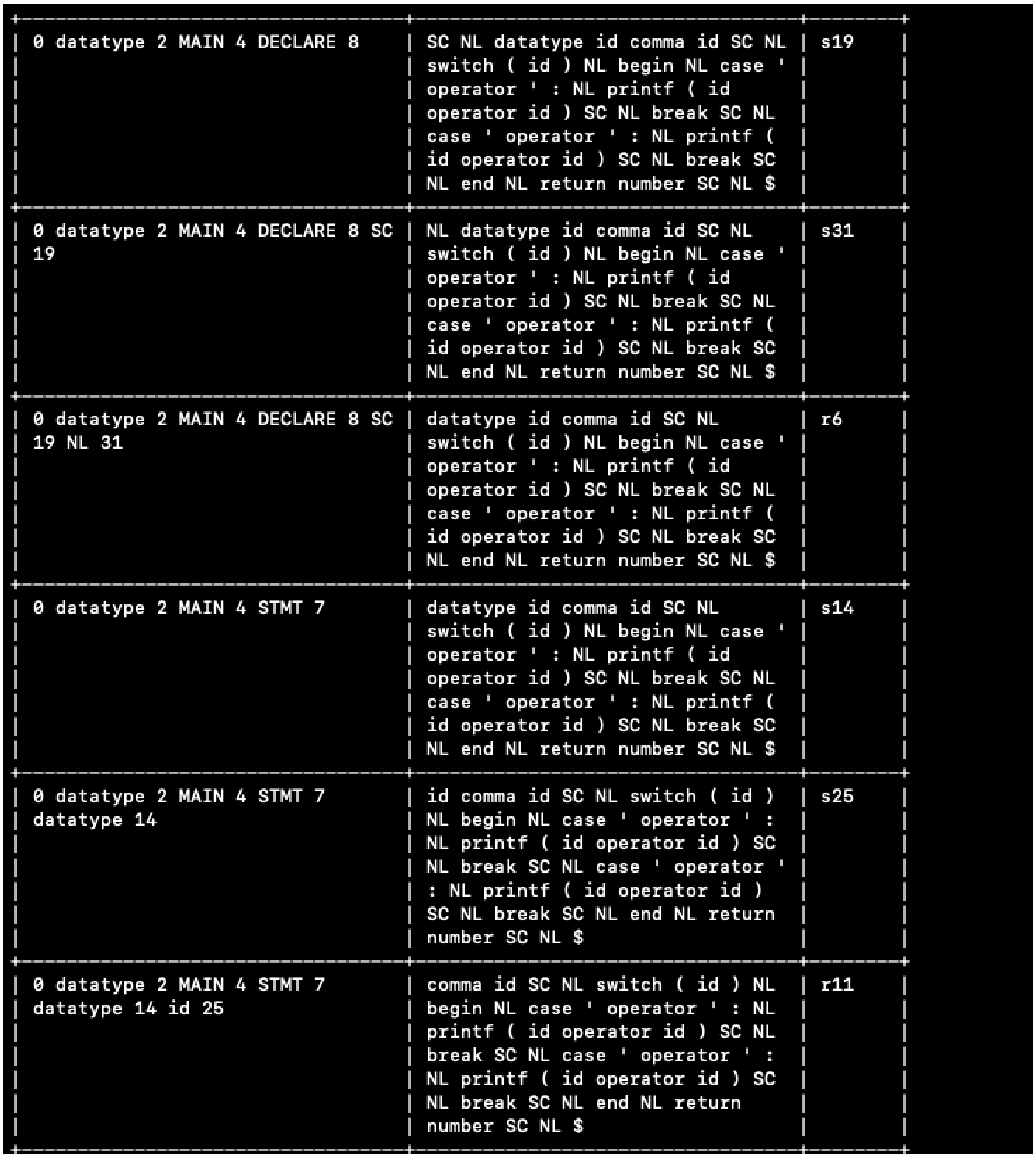
SEQUENTIAL PARSING STEPS:



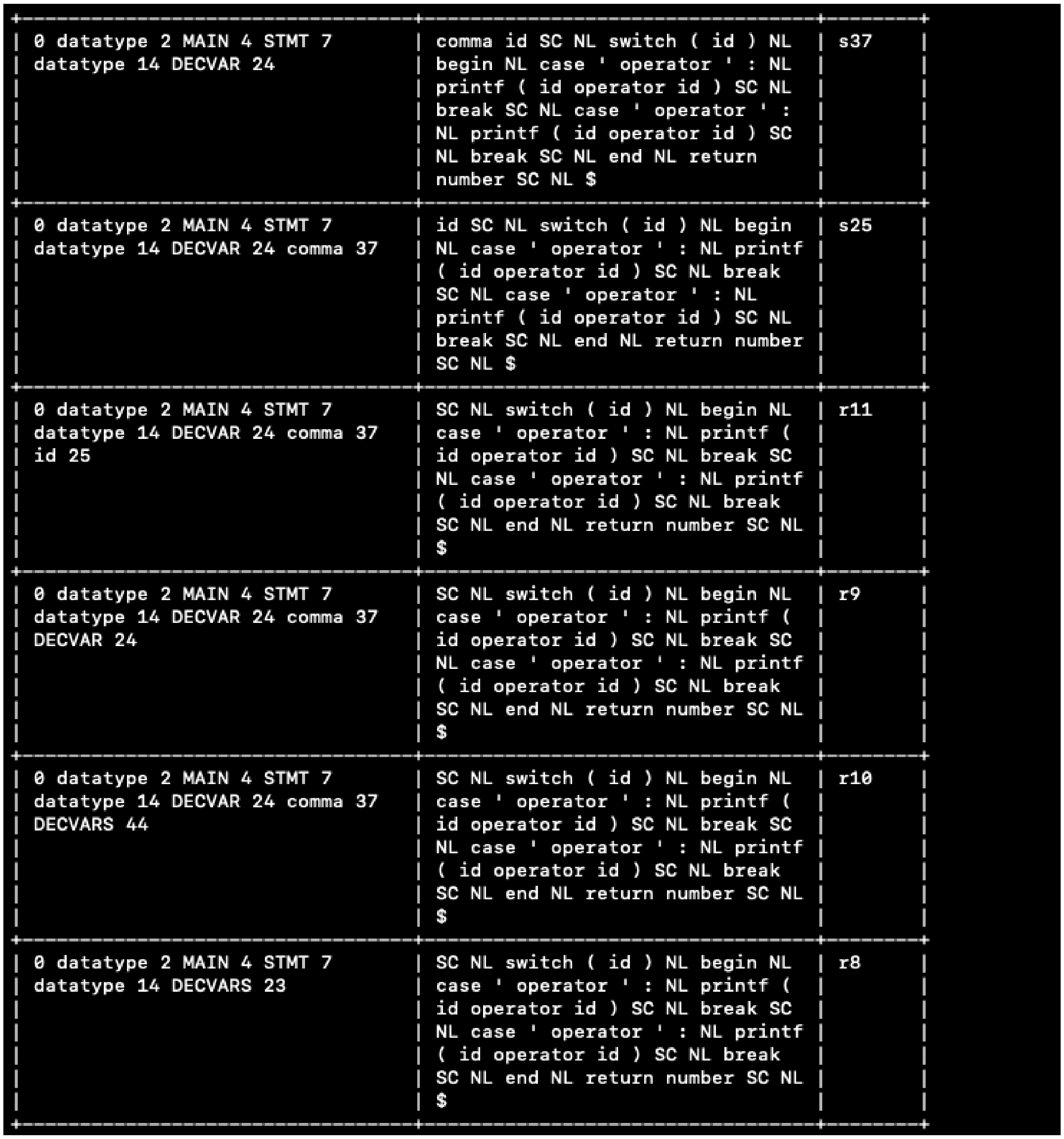
```
SEQUENTIAL PARSING STEPS:

+-----------------------------------+-----------------------------------+--------+
|               Stack               |               Input               | Action |
+===================================+===================================+========+
| 0                                 | datatype main ( ) NL datatype id  | s2     |
|                                   | SC NL datatype id comma id SC NL  |        |
|                                   | switch ( id ) NL begin NL case '  |        |
|                                   | operator ' : NL printf ( id       |        |
|                                   | operator id ) SC NL break SC NL   |        |
|                                   | case ' operator ' : NL printf (   |        |
|                                   | id operator id ) SC NL break SC   |        |
|                                   | NL end NL return number SC NL $   |        |
+-----------------------------------+-----------------------------------+--------+
| 0 datatype 2                      | main ( ) NL datatype id SC NL     | s5     |
|                                   | datatype id comma id SC NL        |        |
|                                   | switch ( id ) NL begin NL case '  |        |
|                                   | operator ' : NL printf ( id       |        |
|                                   | operator id ) SC NL break SC NL   |        |
|                                   | case ' operator ' : NL printf (   |        |
|                                   | id operator id ) SC NL break SC   |        |
|                                   | NL end NL return number SC NL $   |        |
+-----------------------------------+-----------------------------------+--------+
| 0 datatype 2 main 5               | ( ) NL datatype id SC NL          | s17    |
|                                   | datatype id comma id SC NL        |        |
|                                   | switch ( id ) NL begin NL case '  |        |
|                                   | operator ' : NL printf ( id       |        |
|                                   | operator id ) SC NL break SC NL   |        |
|                                   | case ' operator ' : NL printf (   |        |
|                                   | id operator id ) SC NL break SC   |        |
|                                   | NL end NL return number SC NL $   |        |
+-----------------------------------+-----------------------------------+--------+
| 0 datatype 2 main 5 ( 17          | ) NL datatype id SC NL datatype   | s30    |
|                                   | id comma id SC NL switch ( id )   |        |
|                                   | NL begin NL case ' operator ' :   |        |
|                                   | NL printf ( id operator id ) SC   |        |
|                                   | NL break SC NL case ' operator '  |        |
|                                   | : NL printf ( id operator id )    |        |
|                                   | SC NL break SC NL end NL return   |        |
|                                   | number SC NL $                    |        |
+-----------------------------------+-----------------------------------+--------+
| 0 datatype 2 main 5 ( 17 ) 30     | NL datatype id SC NL datatype id  | s41    |
|                                   | comma id SC NL switch ( id ) NL   |        |
|                                   | begin NL case ' operator ' : NL   |        |
|                                   | printf ( id operator id ) SC NL   |        |
|                                   | break SC NL case ' operator ' :   |        |
|                                   | NL printf ( id operator id ) SC   |        |
|                                   | NL break SC NL end NL return      |        |
|                                   | number SC NL $                    |        |
+-----------------------------------+-----------------------------------+--------+
```



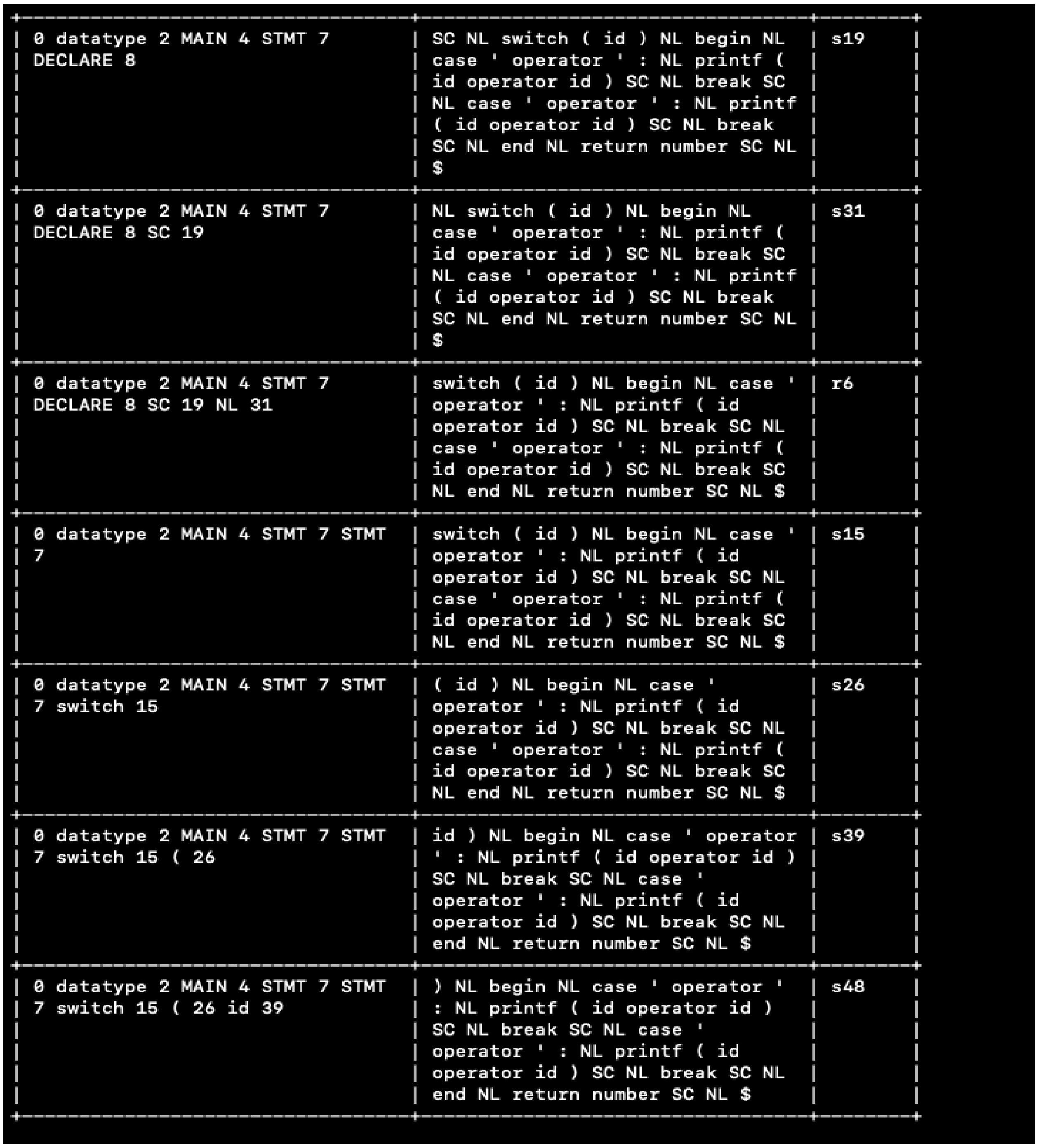
```
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 main 5 ( 17 ) 30 NL | datatype id SC NL datatype id     | r3      |
| 41                                | comma id SC NL switch ( id ) NL   |         |
|                                   | begin NL case ' operator ' : NL   |         |
|                                   | printf ( id operator id ) SC NL   |         |
|                                   | break SC NL case ' operator ' :   |         |
|                                   | NL printf ( id operator id ) SC   |         |
|                                   | NL break SC NL end NL return      |         |
|                                   | number SC NL $                    |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4               | datatype id SC NL datatype id     | s14     |
|                                   | comma id SC NL switch ( id ) NL   |         |
|                                   | begin NL case ' operator ' : NL   |         |
|                                   | printf ( id operator id ) SC NL   |         |
|                                   | break SC NL case ' operator ' :   |         |
|                                   | NL printf ( id operator id ) SC   |         |
|                                   | NL break SC NL end NL return      |         |
|                                   | number SC NL $                    |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 datatype 14   | id SC NL datatype id comma id SC  | s25     |
|                                   | NL switch ( id ) NL begin NL      |         |
|                                   | case ' operator ' : NL printf (   |         |
|                                   | id operator id ) SC NL break SC   |         |
|                                   | NL case ' operator ' : NL printf  |         |
|                                   | ( id operator id ) SC NL break    |         |
|                                   | SC NL end NL return number SC NL  |         |
|                                   | $                                 |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 datatype 14   | SC NL datatype id comma id SC NL  | r11     |
| id 25                             | switch ( id ) NL begin NL case '  |         |
|                                   | operator ' : NL printf ( id       |         |
|                                   | operator id ) SC NL break SC NL   |         |
|                                   | case ' operator ' : NL printf (   |         |
|                                   | id operator id ) SC NL break SC   |         |
|                                   | NL end NL return number SC NL $   |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 datatype 14   | SC NL datatype id comma id SC NL  | r9      |
| DECVAR 24                         | switch ( id ) NL begin NL case '  |         |
|                                   | operator ' : NL printf ( id       |         |
|                                   | operator id ) SC NL break SC NL   |         |
|                                   | case ' operator ' : NL printf (   |         |
|                                   | id operator id ) SC NL break SC   |         |
|                                   | NL end NL return number SC NL $   |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 datatype 14   | SC NL datatype id comma id SC NL  | r8      |
| DECVARS 23                        | switch ( id ) NL begin NL case '  |         |
|                                   | operator ' : NL printf ( id       |         |
|                                   | operator id ) SC NL break SC NL   |         |
|                                   | case ' operator ' : NL printf (   |         |
|                                   | id operator id ) SC NL break SC   |         |
|                                   | NL end NL return number SC NL $   |         |
+-----------------------------------+-----------------------------------+---------+
```



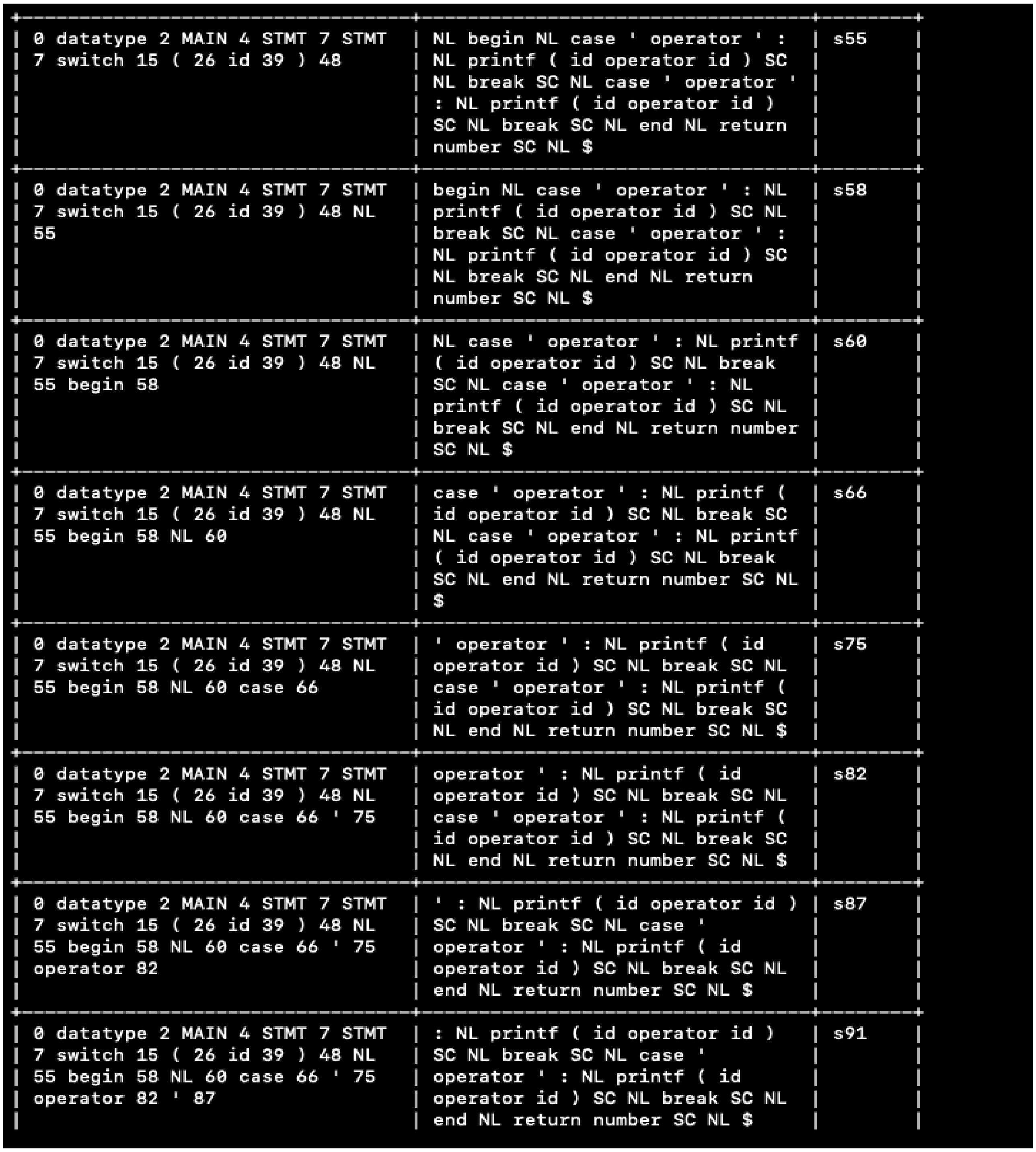
```
+------------------------------------+------------------------------------+---------+
| 0 datatype 2 MAIN 4 DECLARE 8      | SC NL datatype id comma id SC NL   | s19     |
|                                    | switch ( id ) NL begin NL case '   |         |
|                                    | operator ' : NL printf ( id        |         |
|                                    | operator id ) SC NL break SC NL    |         |
|                                    | case ' operator ' : NL printf (    |         |
|                                    | id operator id ) SC NL break SC    |         |
|                                    | NL end NL return number SC NL $    |         |
+------------------------------------+------------------------------------+---------+
| 0 datatype 2 MAIN 4 DECLARE 8 SC   | NL datatype id comma id SC NL      | s31     |
| 19                                 | switch ( id ) NL begin NL case '   |         |
|                                    | operator ' : NL printf ( id        |         |
|                                    | operator id ) SC NL break SC NL    |         |
|                                    | case ' operator ' : NL printf (    |         |
|                                    | id operator id ) SC NL break SC    |         |
|                                    | NL end NL return number SC NL $    |         |
+------------------------------------+------------------------------------+---------+
| 0 datatype 2 MAIN 4 DECLARE 8 SC   | datatype id comma id SC NL         | r6      |
| 19 NL 31                           | switch ( id ) NL begin NL case '   |         |
|                                    | operator ' : NL printf ( id        |         |
|                                    | operator id ) SC NL break SC NL    |         |
|                                    | case ' operator ' : NL printf (    |         |
|                                    | id operator id ) SC NL break SC    |         |
|                                    | NL end NL return number SC NL $    |         |
+------------------------------------+------------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7         | datatype id comma id SC NL         | s14     |
|                                    | switch ( id ) NL begin NL case '   |         |
|                                    | operator ' : NL printf ( id        |         |
|                                    | operator id ) SC NL break SC NL    |         |
|                                    | case ' operator ' : NL printf (    |         |
|                                    | id operator id ) SC NL break SC    |         |
|                                    | NL end NL return number SC NL $    |         |
+------------------------------------+------------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7         | id comma id SC NL switch ( id )    | s25     |
| datatype 14                        | NL begin NL case ' operator ' :    |         |
|                                    | NL printf ( id operator id ) SC    |         |
|                                    | NL break SC NL case ' operator '   |         |
|                                    | : NL printf ( id operator id )     |         |
|                                    | SC NL break SC NL end NL return    |         |
|                                    | number SC NL $                     |         |
+------------------------------------+------------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7         | comma id SC NL switch ( id ) NL    | r11     |
| datatype 14 id 25                  | begin NL case ' operator ' : NL    |         |
|                                    | printf ( id operator id ) SC NL    |         |
|                                    | break SC NL case ' operator ' :    |         |
|                                    | NL printf ( id operator id ) SC    |         |
|                                    | NL break SC NL end NL return       |         |
|                                    | number SC NL $                     |         |
+------------------------------------+------------------------------------+---------+
```



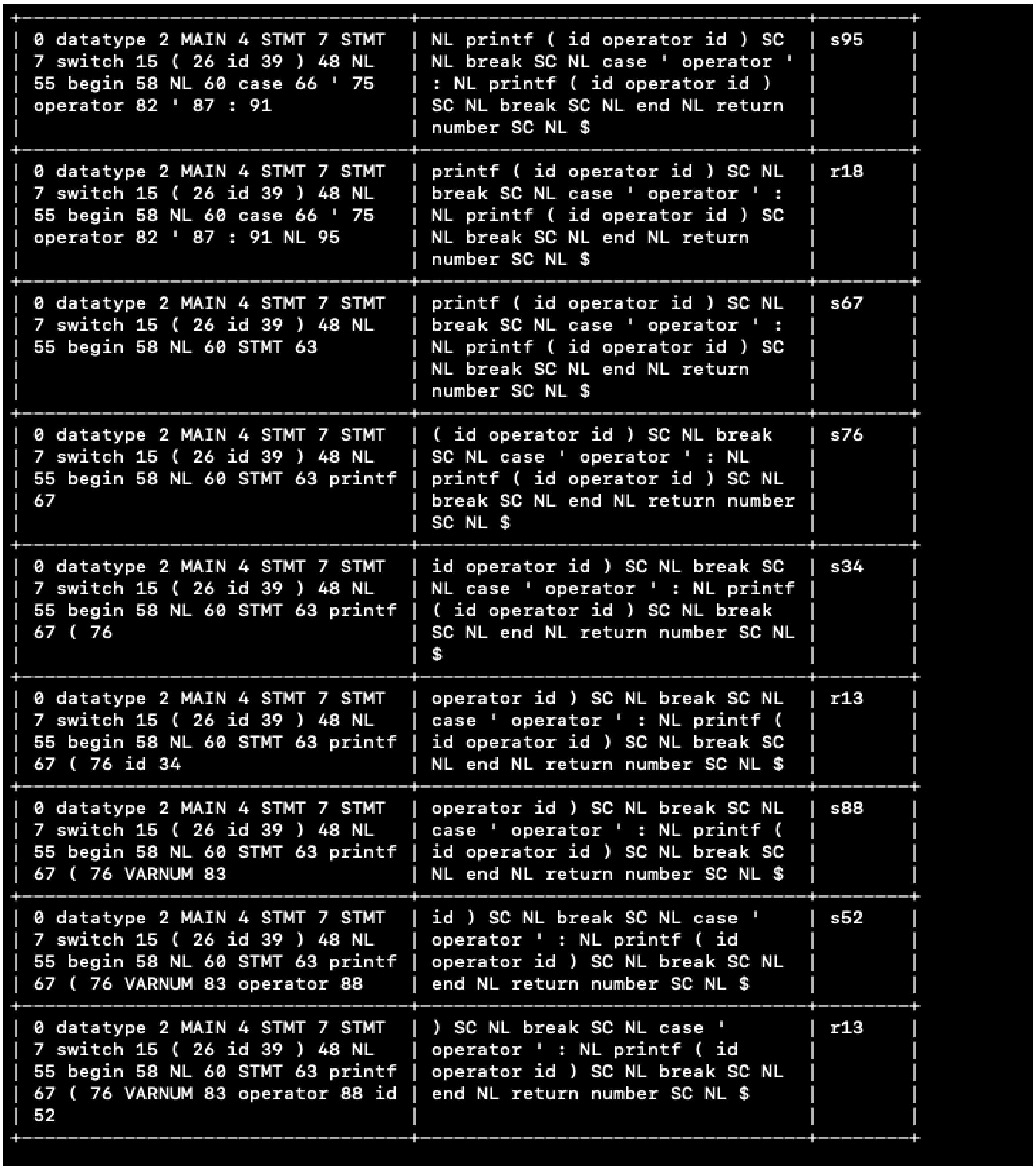
```
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7        | comma id SC NL switch ( id ) NL   | s37     |
| datatype 14 DECVAR 24             | begin NL case ' operator ' : NL   |         |
|                                   | printf ( id operator id ) SC NL   |         |
|                                   | break SC NL case ' operator ' :   |         |
|                                   | NL printf ( id operator id ) SC   |         |
|                                   | NL break SC NL end NL return      |         |
|                                   | number SC NL $                    |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7        | id SC NL switch ( id ) NL begin   | s25     |
| datatype 14 DECVAR 24 comma 37    | NL case ' operator ' : NL printf  |         |
|                                   | ( id operator id ) SC NL break    |         |
|                                   | SC NL case ' operator ' : NL      |         |
|                                   | printf ( id operator id ) SC NL   |         |
|                                   | break SC NL end NL return number  |         |
|                                   | SC NL $                           |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7        | SC NL switch ( id ) NL begin NL   | r11     |
| datatype 14 DECVAR 24 comma 37    | case ' operator ' : NL printf (   |         |
| id 25                             | id operator id ) SC NL break SC   |         |
|                                   | NL case ' operator ' : NL printf  |         |
|                                   | ( id operator id ) SC NL break    |         |
|                                   | SC NL end NL return number SC NL  |         |
|                                   | $                                 |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7        | SC NL switch ( id ) NL begin NL   | r9      |
| datatype 14 DECVAR 24 comma 37    | case ' operator ' : NL printf (   |         |
| DECVAR 24                         | id operator id ) SC NL break SC   |         |
|                                   | NL case ' operator ' : NL printf  |         |
|                                   | ( id operator id ) SC NL break    |         |
|                                   | SC NL end NL return number SC NL  |         |
|                                   | $                                 |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7        | SC NL switch ( id ) NL begin NL   | r10     |
| datatype 14 DECVAR 24 comma 37    | case ' operator ' : NL printf (   |         |
| DECVARS 44                        | id operator id ) SC NL break SC   |         |
|                                   | NL case ' operator ' : NL printf  |         |
|                                   | ( id operator id ) SC NL break    |         |
|                                   | SC NL end NL return number SC NL  |         |
|                                   | $                                 |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7        | SC NL switch ( id ) NL begin NL   | r8      |
| datatype 14 DECVARS 23            | case ' operator ' : NL printf (   |         |
|                                   | id operator id ) SC NL break SC   |         |
|                                   | NL case ' operator ' : NL printf  |         |
|                                   | ( id operator id ) SC NL break    |         |
|                                   | SC NL end NL return number SC NL  |         |
|                                   | $                                 |         |
+-----------------------------------+-----------------------------------+---------+
```



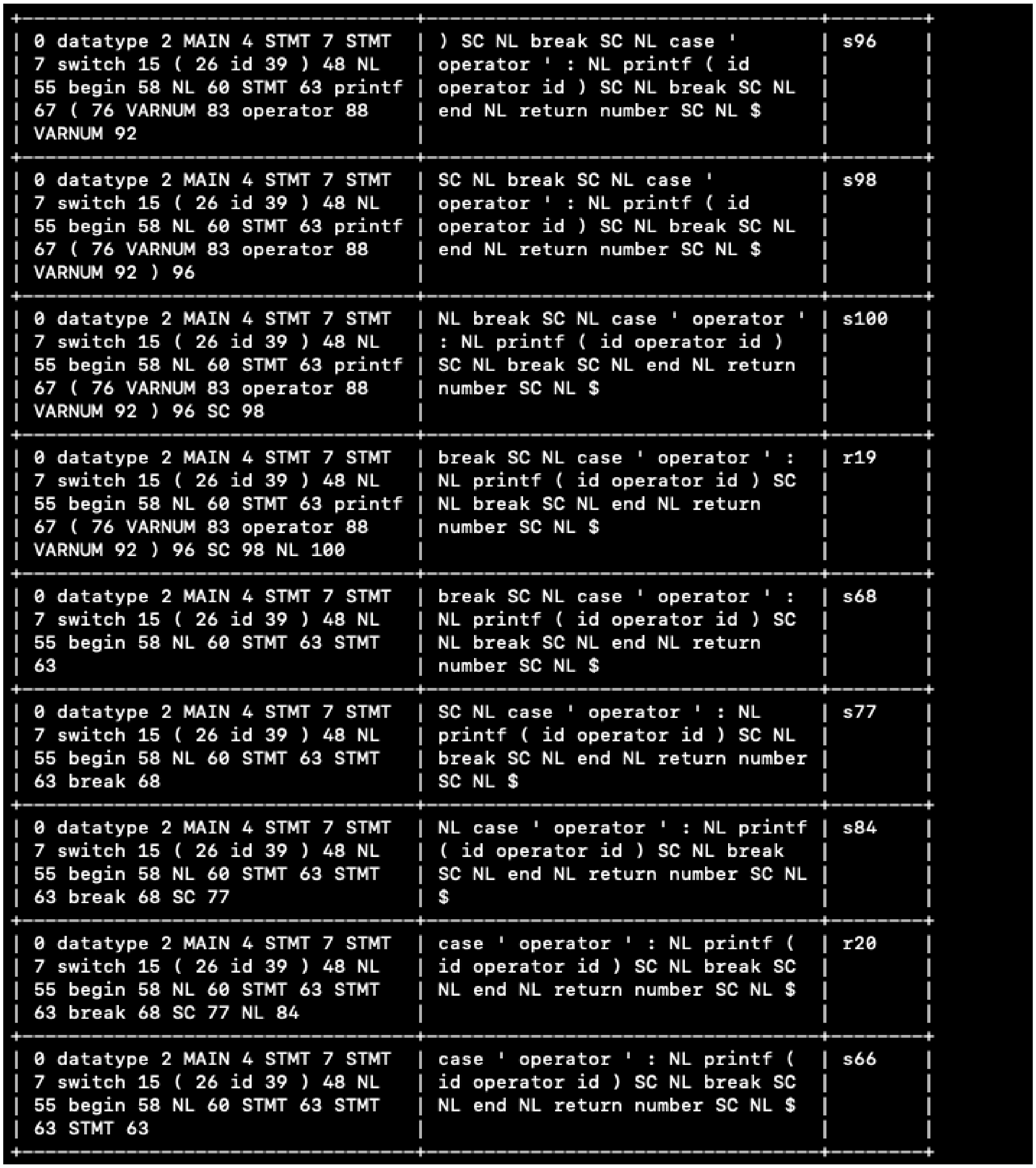
```
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7        | SC NL switch ( id ) NL begin NL   | s19     |
| DECLARE 8                         | case ' operator ' : NL printf (   |         |
|                                   | id operator id ) SC NL break SC   |         |
|                                   | NL case ' operator ' : NL printf  |         |
|                                   | ( id operator id ) SC NL break    |         |
|                                   | SC NL end NL return number SC NL  |         |
|                                   | $                                 |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7        | NL switch ( id ) NL begin NL      | s31     |
| DECLARE 8 SC 19                   | case ' operator ' : NL printf (   |         |
|                                   | id operator id ) SC NL break SC   |         |
|                                   | NL case ' operator ' : NL printf  |         |
|                                   | ( id operator id ) SC NL break    |         |
|                                   | SC NL end NL return number SC NL  |         |
|                                   | $                                 |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7        | switch ( id ) NL begin NL case '  | r6      |
| DECLARE 8 SC 19 NL 31             | operator ' : NL printf ( id       |         |
|                                   | operator id ) SC NL break SC NL   |         |
|                                   | case ' operator ' : NL printf (   |         |
|                                   | id operator id ) SC NL break SC   |         |
|                                   | NL end NL return number SC NL $   |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | switch ( id ) NL begin NL case '  | s15     |
| 7                                 | operator ' : NL printf ( id       |         |
|                                   | operator id ) SC NL break SC NL   |         |
|                                   | case ' operator ' : NL printf (   |         |
|                                   | id operator id ) SC NL break SC   |         |
|                                   | NL end NL return number SC NL $   |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | ( id ) NL begin NL case '         | s26     |
| 7 switch 15                       | operator ' : NL printf ( id       |         |
|                                   | operator id ) SC NL break SC NL   |         |
|                                   | case ' operator ' : NL printf (   |         |
|                                   | id operator id ) SC NL break SC   |         |
|                                   | NL end NL return number SC NL $   |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | id ) NL begin NL case ' operator  | s39     |
| 7 switch 15 ( 26                  | ' : NL printf ( id operator id )  |         |
|                                   | SC NL break SC NL case '          |         |
|                                   | operator ' : NL printf ( id       |         |
|                                   | operator id ) SC NL break SC NL   |         |
|                                   | end NL return number SC NL $      |         |
+-----------------------------------+-----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | ) NL begin NL case ' operator '   | s48     |
| 7 switch 15 ( 26 id 39            | : NL printf ( id operator id )    |         |
|                                   | SC NL break SC NL case '          |         |
|                                   | operator ' : NL printf ( id       |         |
|                                   | operator id ) SC NL break SC NL   |         |
|                                   | end NL return number SC NL $      |         |
+-----------------------------------+-----------------------------------+---------+
```



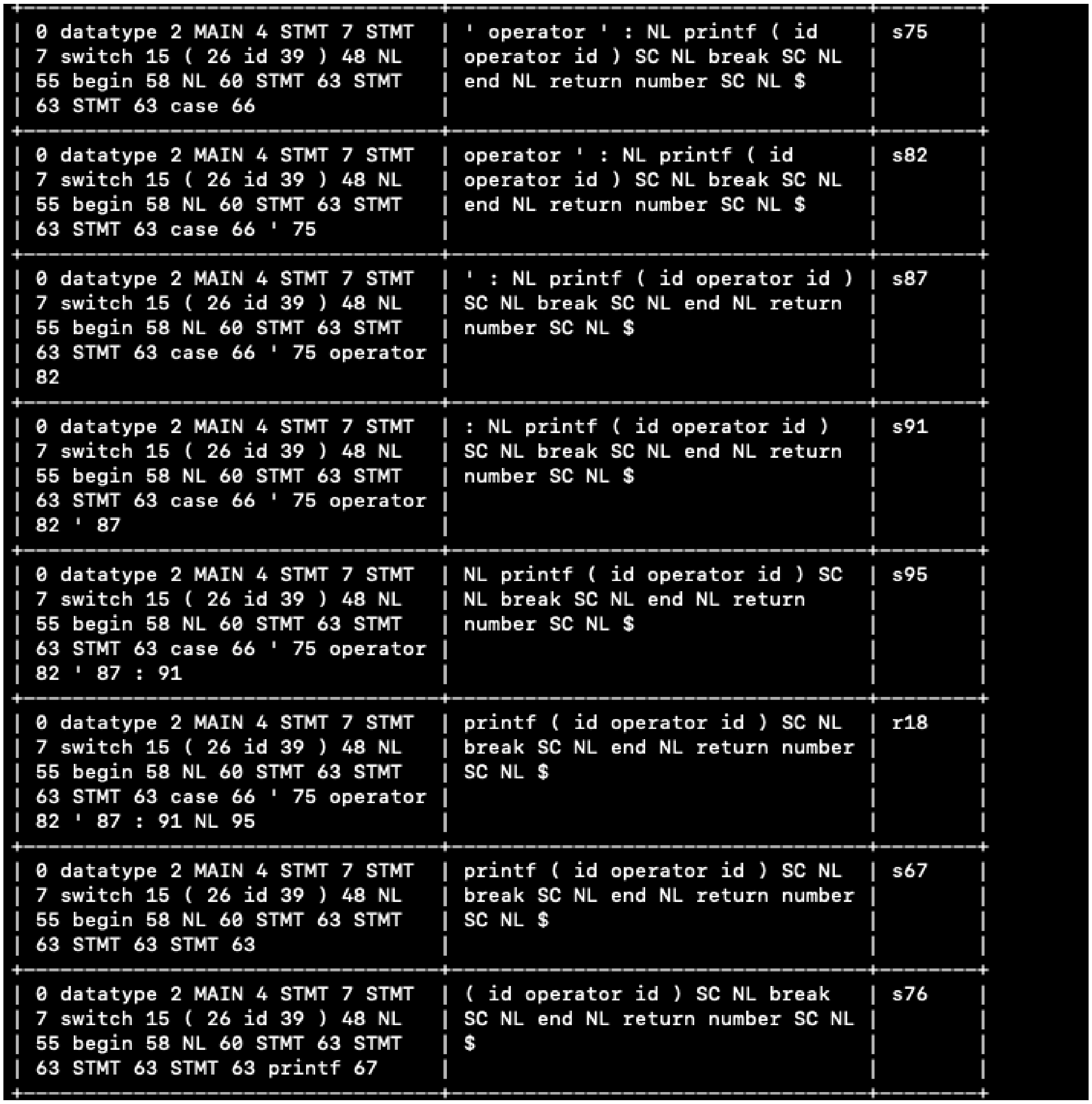
```
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | NL begin NL case ' operator ' :   | s55      |
| 7 switch 15 ( 26 id 39 ) 48       | NL printf ( id operator id ) SC   |          |
|                                   | NL break SC NL case ' operator '  |          |
|                                   | : NL printf ( id operator id )    |          |
|                                   | SC NL break SC NL end NL return   |          |
|                                   | number SC NL $                    |          |
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | begin NL case ' operator ' : NL   | s58      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | printf ( id operator id ) SC NL   |          |
| 55                                | break SC NL case ' operator ' :   |          |
|                                   | NL printf ( id operator id ) SC   |          |
|                                   | NL break SC NL end NL return      |          |
|                                   | number SC NL $                    |          |
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | NL case ' operator ' : NL printf  | s60      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | ( id operator id ) SC NL break    |          |
| 55 begin 58                       | SC NL case ' operator ' : NL      |          |
|                                   | printf ( id operator id ) SC NL   |          |
|                                   | break SC NL end NL return number  |          |
|                                   | SC NL $                           |          |
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | case ' operator ' : NL printf (   | s66      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | id operator id ) SC NL break SC   |          |
| 55 begin 58 NL 60                 | NL case ' operator ' : NL printf  |          |
|                                   | ( id operator id ) SC NL break    |          |
|                                   | SC NL end NL return number SC NL  |          |
|                                   | $                                 |          |
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | ' operator ' : NL printf ( id     | s75      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | operator id ) SC NL break SC NL   |          |
| 55 begin 58 NL 60 case 66         | case ' operator ' : NL printf (   |          |
|                                   | id operator id ) SC NL break SC   |          |
|                                   | NL end NL return number SC NL $   |          |
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | operator ' : NL printf ( id       | s82      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | operator id ) SC NL break SC NL   |          |
| 55 begin 58 NL 60 case 66 ' 75    | case ' operator ' : NL printf (   |          |
|                                   | id operator id ) SC NL break SC   |          |
|                                   | NL end NL return number SC NL $   |          |
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | ' : NL printf ( id operator id )  | s87      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | SC NL break SC NL case '          |          |
| 55 begin 58 NL 60 case 66 ' 75    | operator ' : NL printf ( id       |          |
| operator 82                       | operator id ) SC NL break SC NL   |          |
|                                   | end NL return number SC NL $      |          |
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | : NL printf ( id operator id )    | s91      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | SC NL break SC NL case '          |          |
| 55 begin 58 NL 60 case 66 ' 75    | operator ' : NL printf ( id       |          |
| operator 82 ' 87                  | operator id ) SC NL break SC NL   |          |
|                                   | end NL return number SC NL $      |          |
```



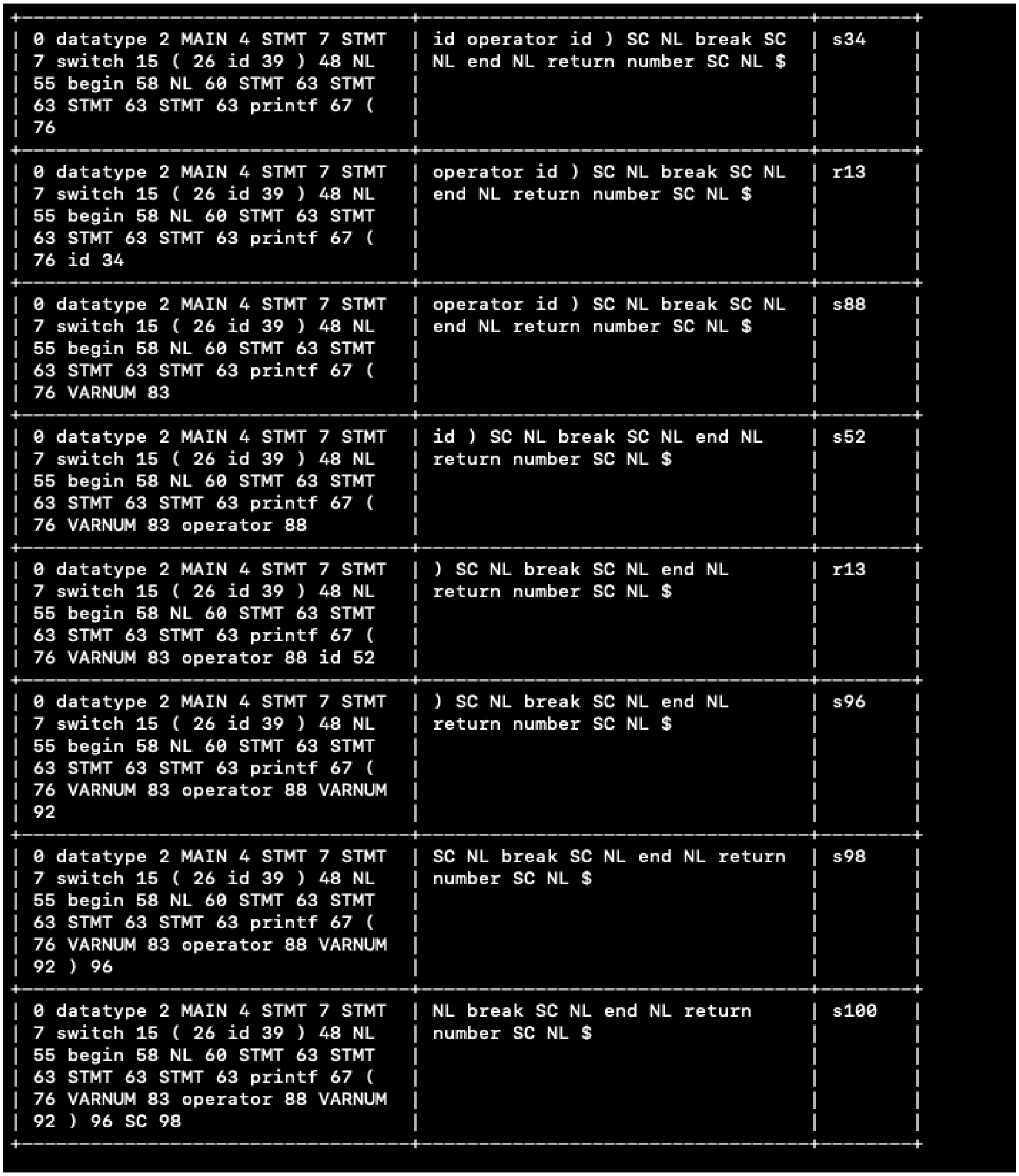
```
+----------------------------------+----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | NL printf ( id operator id ) SC  | s95     |
| 7 switch 15 ( 26 id 39 ) 48 NL   | NL break SC NL case ' operator ' |         |
| 55 begin 58 NL 60 case 66 ' 75   | : NL printf ( id operator id )   |         |
| operator 82 ' 87 : 91            | SC NL break SC NL end NL return  |         |
|                                  | number SC NL $                   |         |
+----------------------------------+----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | printf ( id operator id ) SC NL  | r18     |
| 7 switch 15 ( 26 id 39 ) 48 NL   | break SC NL case ' operator ' :  |         |
| 55 begin 58 NL 60 case 66 ' 75   | NL printf ( id operator id ) SC  |         |
| operator 82 ' 87 : 91 NL 95      | NL break SC NL end NL return     |         |
|                                  | number SC NL $                   |         |
+----------------------------------+----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | printf ( id operator id ) SC NL  | s67     |
| 7 switch 15 ( 26 id 39 ) 48 NL   | break SC NL case ' operator ' :  |         |
| 55 begin 58 NL 60 STMT 63        | NL printf ( id operator id ) SC  |         |
|                                  | NL break SC NL end NL return     |         |
|                                  | number SC NL $                   |         |
+----------------------------------+----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | ( id operator id ) SC NL break   | s76     |
| 7 switch 15 ( 26 id 39 ) 48 NL   | SC NL case ' operator ' : NL     |         |
| 55 begin 58 NL 60 STMT 63 printf | printf ( id operator id ) SC NL  |         |
| 67                               | break SC NL end NL return number |         |
|                                  | SC NL $                          |         |
+----------------------------------+----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | id operator id ) SC NL break SC  | s34     |
| 7 switch 15 ( 26 id 39 ) 48 NL   | NL case ' operator ' : NL printf |         |
| 55 begin 58 NL 60 STMT 63 printf | ( id operator id ) SC NL break   |         |
| 67 ( 76                          | SC NL end NL return number SC NL |         |
|                                  | $                                |         |
+----------------------------------+----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | operator id ) SC NL break SC NL  | r13     |
| 7 switch 15 ( 26 id 39 ) 48 NL   | case ' operator ' : NL printf (  |         |
| 55 begin 58 NL 60 STMT 63 printf | id operator id ) SC NL break SC  |         |
| 67 ( 76 id 34                    | NL end NL return number SC NL $  |         |
+----------------------------------+----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | operator id ) SC NL break SC NL  | s88     |
| 7 switch 15 ( 26 id 39 ) 48 NL   | case ' operator ' : NL printf (  |         |
| 55 begin 58 NL 60 STMT 63 printf | id operator id ) SC NL break SC  |         |
| 67 ( 76 VARNUM 83                | NL end NL return number SC NL $  |         |
+----------------------------------+----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | id ) SC NL break SC NL case '    | s52     |
| 7 switch 15 ( 26 id 39 ) 48 NL   | operator ' : NL printf ( id      |         |
| 55 begin 58 NL 60 STMT 63 printf | operator id ) SC NL break SC NL  |         |
| 67 ( 76 VARNUM 83 operator 88    | end NL return number SC NL $     |         |
+----------------------------------+----------------------------------+---------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | ) SC NL break SC NL case '       | r13     |
| 7 switch 15 ( 26 id 39 ) 48 NL   | operator ' : NL printf ( id      |         |
| 55 begin 58 NL 60 STMT 63 printf | operator id ) SC NL break SC NL  |         |
| 67 ( 76 VARNUM 83 operator 88 id | end NL return number SC NL $     |         |
| 52                               |                                  |         |
+----------------------------------+----------------------------------+---------+
```



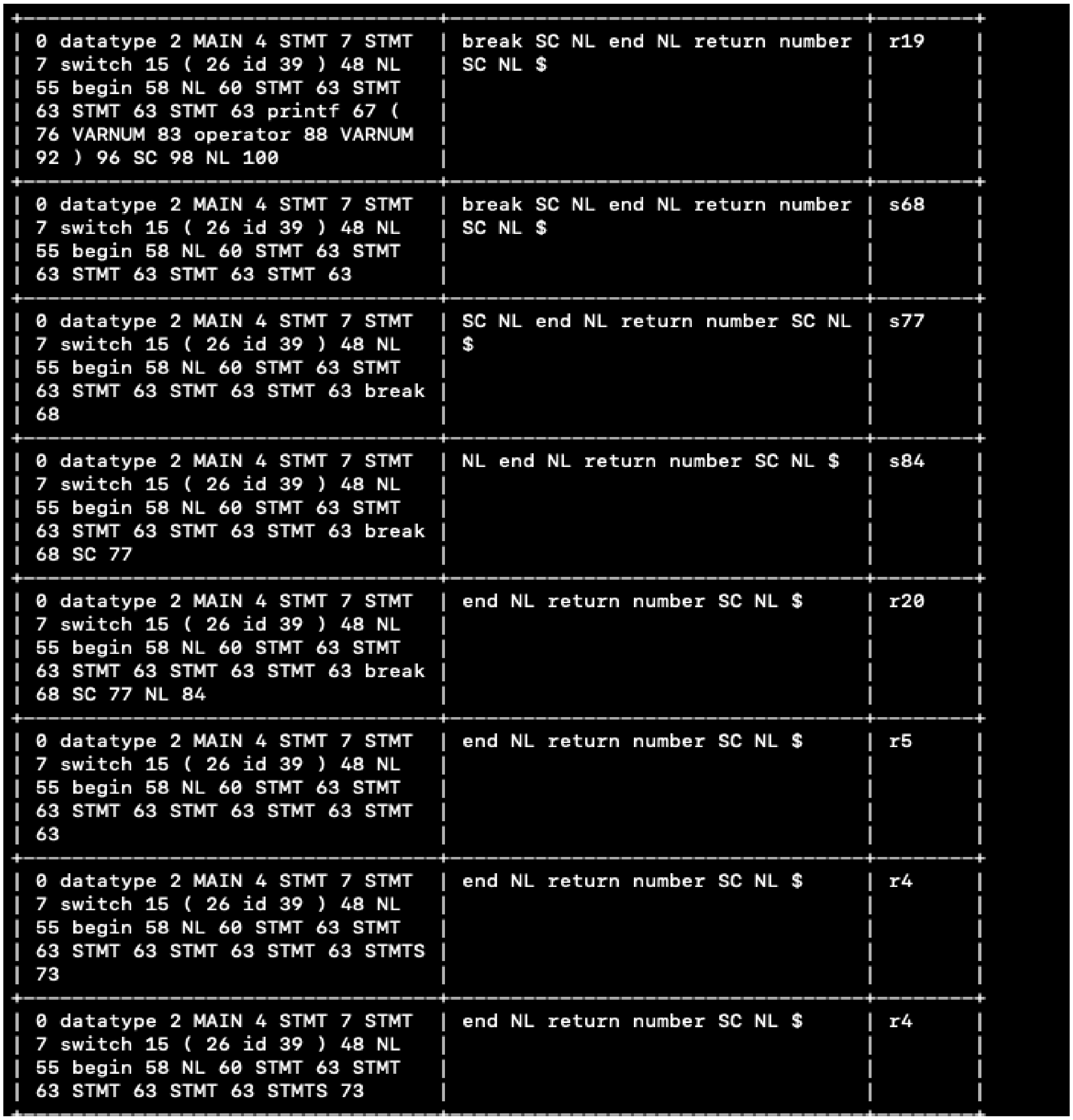
```
+----------------------------------+----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | ) SC NL break SC NL case '       | s96      |
| 7 switch 15 ( 26 id 39 ) 48 NL   | operator ' : NL printf ( id      |          |
| 55 begin 58 NL 60 STMT 63 printf | operator id ) SC NL break SC NL  |          |
| 67 ( 76 VARNUM 83 operator 88    | end NL return number SC NL $     |          |
| VARNUM 92                        |                                  |          |
+----------------------------------+----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | SC NL break SC NL case '         | s98      |
| 7 switch 15 ( 26 id 39 ) 48 NL   | operator ' : NL printf ( id      |          |
| 55 begin 58 NL 60 STMT 63 printf | operator id ) SC NL break SC NL  |          |
| 67 ( 76 VARNUM 83 operator 88    | end NL return number SC NL $     |          |
| VARNUM 92 ) 96                   |                                  |          |
+----------------------------------+----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | NL break SC NL case ' operator ' | s100     |
| 7 switch 15 ( 26 id 39 ) 48 NL   | : NL printf ( id operator id )   |          |
| 55 begin 58 NL 60 STMT 63 printf | SC NL break SC NL end NL return  |          |
| 67 ( 76 VARNUM 83 operator 88    | number SC NL $                   |          |
| VARNUM 92 ) 96 SC 98             |                                  |          |
+----------------------------------+----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | break SC NL case ' operator ' :  | r19      |
| 7 switch 15 ( 26 id 39 ) 48 NL   | NL printf ( id operator id ) SC  |          |
| 55 begin 58 NL 60 STMT 63 printf | NL break SC NL end NL return     |          |
| 67 ( 76 VARNUM 83 operator 88    | number SC NL $                   |          |
| VARNUM 92 ) 96 SC 98 NL 100      |                                  |          |
+----------------------------------+----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | break SC NL case ' operator ' :  | s68      |
| 7 switch 15 ( 26 id 39 ) 48 NL   | NL printf ( id operator id ) SC  |          |
| 55 begin 58 NL 60 STMT 63 STMT   | NL break SC NL end NL return     |          |
| 63                               | number SC NL $                   |          |
+----------------------------------+----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | SC NL case ' operator ' : NL     | s77      |
| 7 switch 15 ( 26 id 39 ) 48 NL   | printf ( id operator id ) SC NL  |          |
| 55 begin 58 NL 60 STMT 63 STMT   | break SC NL end NL return number |          |
| 63 break 68                      | SC NL $                          |          |
+----------------------------------+----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | NL case ' operator ' : NL printf | s84      |
| 7 switch 15 ( 26 id 39 ) 48 NL   | ( id operator id ) SC NL break   |          |
| 55 begin 58 NL 60 STMT 63 STMT   | SC NL end NL return number SC NL |          |
| 63 break 68 SC 77                | $                                |          |
+----------------------------------+----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | case ' operator ' : NL printf (  | r20      |
| 7 switch 15 ( 26 id 39 ) 48 NL   | id operator id ) SC NL break SC  |          |
| 55 begin 58 NL 60 STMT 63 STMT   | NL end NL return number SC NL $  |          |
| 63 break 68 SC 77 NL 84          |                                  |          |
+----------------------------------+----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT  | case ' operator ' : NL printf (  | s66      |
| 7 switch 15 ( 26 id 39 ) 48 NL   | id operator id ) SC NL break SC  |          |
| 55 begin 58 NL 60 STMT 63 STMT   | NL end NL return number SC NL $  |          |
| 63 STMT 63                       |                                  |          |
+----------------------------------+----------------------------------+----------+
```



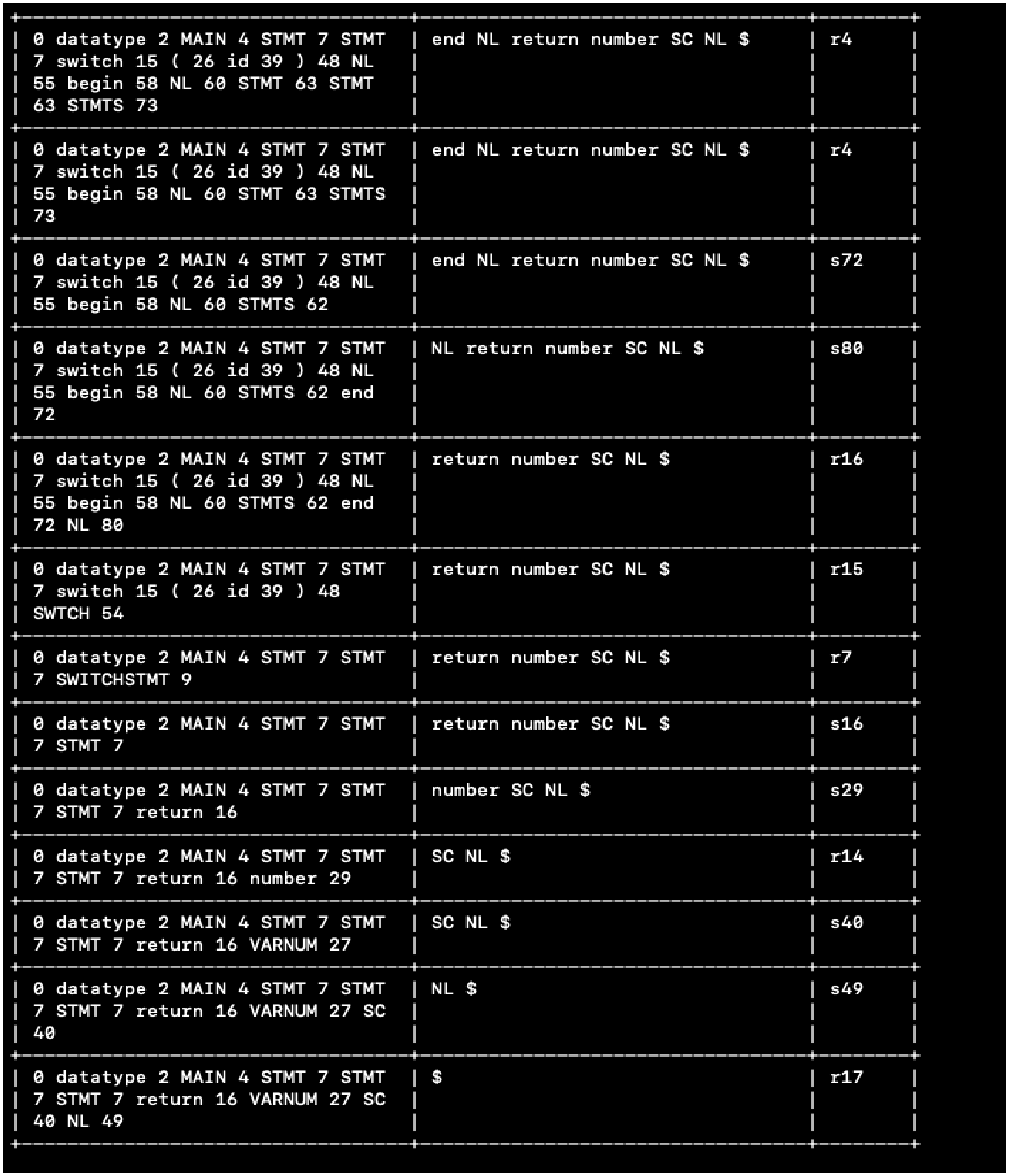
```
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | ' operator ' : NL printf ( id     | s75      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | operator id ) SC NL break SC NL   |          |
| 55 begin 58 NL 60 STMT 63 STMT    | end NL return number SC NL $      |          |
| 63 STMT 63 case 66                |                                   |          |
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | operator ' : NL printf ( id       | s82      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | operator id ) SC NL break SC NL   |          |
| 55 begin 58 NL 60 STMT 63 STMT    | end NL return number SC NL $      |          |
| 63 STMT 63 case 66 ' 75           |                                   |          |
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | ' : NL printf ( id operator id )  | s87      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | SC NL break SC NL end NL return   |          |
| 55 begin 58 NL 60 STMT 63 STMT    | number SC NL $                    |          |
| 63 STMT 63 case 66 ' 75 operator  |                                   |          |
| 82                                |                                   |          |
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | : NL printf ( id operator id )    | s91      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | SC NL break SC NL end NL return   |          |
| 55 begin 58 NL 60 STMT 63 STMT    | number SC NL $                    |          |
| 63 STMT 63 case 66 ' 75 operator  |                                   |          |
| 82 ' 87                           |                                   |          |
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | NL printf ( id operator id ) SC   | s95      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | NL break SC NL end NL return      |          |
| 55 begin 58 NL 60 STMT 63 STMT    | number SC NL $                    |          |
| 63 STMT 63 case 66 ' 75 operator  |                                   |          |
| 82 ' 87 : 91                      |                                   |          |
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | printf ( id operator id ) SC NL   | r18      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | break SC NL end NL return number  |          |
| 55 begin 58 NL 60 STMT 63 STMT    | SC NL $                           |          |
| 63 STMT 63 case 66 ' 75 operator  |                                   |          |
| 82 ' 87 : 91 NL 95                |                                   |          |
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | printf ( id operator id ) SC NL   | s67      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | break SC NL end NL return number  |          |
| 55 begin 58 NL 60 STMT 63 STMT    | SC NL $                           |          |
| 63 STMT 63 STMT 63                |                                   |          |
+-----------------------------------+-----------------------------------+----------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | ( id operator id ) SC NL break    | s76      |
| 7 switch 15 ( 26 id 39 ) 48 NL    | SC NL end NL return number SC NL  |          |
| 55 begin 58 NL 60 STMT 63 STMT    | $                                 |          |
| 63 STMT 63 STMT 63 printf 67      |                                   |          |
+-----------------------------------+-----------------------------------+----------+
```



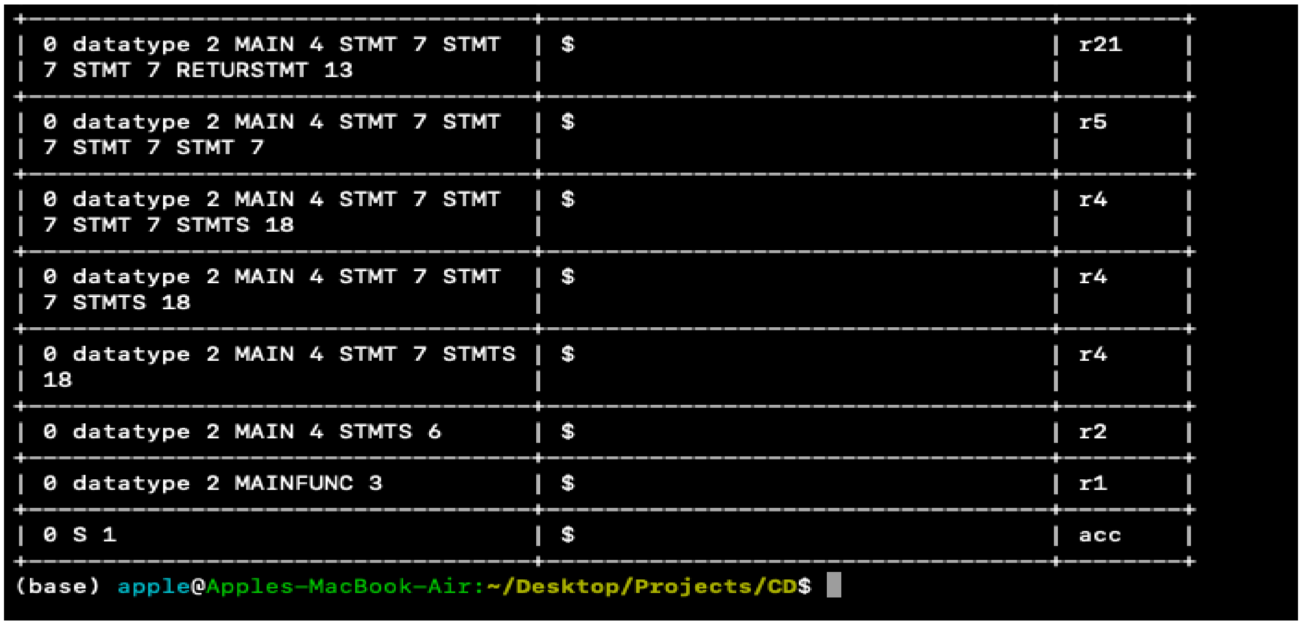
```
+---------------------------------+---------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | id operator id ) SC NL break SC | s34    |
| 7 switch 15 ( 26 id 39 ) 48 NL  | NL end NL return number SC NL $ |        |
| 55 begin 58 NL 60 STMT 63 STMT  |                                 |        |
| 63 STMT 63 STMT 63 printf 67 (  |                                 |        |
| 76                              |                                 |        |
+---------------------------------+---------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | operator id ) SC NL break SC NL | r13    |
| 7 switch 15 ( 26 id 39 ) 48 NL  | end NL return number SC NL $    |        |
| 55 begin 58 NL 60 STMT 63 STMT  |                                 |        |
| 63 STMT 63 STMT 63 printf 67 (  |                                 |        |
| 76 id 34                        |                                 |        |
+---------------------------------+---------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | operator id ) SC NL break SC NL | s88    |
| 7 switch 15 ( 26 id 39 ) 48 NL  | end NL return number SC NL $    |        |
| 55 begin 58 NL 60 STMT 63 STMT  |                                 |        |
| 63 STMT 63 STMT 63 printf 67 (  |                                 |        |
| 76 VARNUM 83                    |                                 |        |
+---------------------------------+---------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | id ) SC NL break SC NL end NL   | s52    |
| 7 switch 15 ( 26 id 39 ) 48 NL  | return number SC NL $           |        |
| 55 begin 58 NL 60 STMT 63 STMT  |                                 |        |
| 63 STMT 63 STMT 63 printf 67 (  |                                 |        |
| 76 VARNUM 83 operator 88        |                                 |        |
+---------------------------------+---------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | ) SC NL break SC NL end NL      | r13    |
| 7 switch 15 ( 26 id 39 ) 48 NL  | return number SC NL $           |        |
| 55 begin 58 NL 60 STMT 63 STMT  |                                 |        |
| 63 STMT 63 STMT 63 printf 67 (  |                                 |        |
| 76 VARNUM 83 operator 88 id 52  |                                 |        |
+---------------------------------+---------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | ) SC NL break SC NL end NL      | s96    |
| 7 switch 15 ( 26 id 39 ) 48 NL  | return number SC NL $           |        |
| 55 begin 58 NL 60 STMT 63 STMT  |                                 |        |
| 63 STMT 63 STMT 63 printf 67 (  |                                 |        |
| 76 VARNUM 83 operator 88 VARNUM |                                 |        |
| 92                              |                                 |        |
+---------------------------------+---------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | SC NL break SC NL end NL return | s98    |
| 7 switch 15 ( 26 id 39 ) 48 NL  | number SC NL $                  |        |
| 55 begin 58 NL 60 STMT 63 STMT  |                                 |        |
| 63 STMT 63 STMT 63 printf 67 (  |                                 |        |
| 76 VARNUM 83 operator 88 VARNUM |                                 |        |
| 92 ) 96                         |                                 |        |
+---------------------------------+---------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | NL break SC NL end NL return    | s100   |
| 7 switch 15 ( 26 id 39 ) 48 NL  | number SC NL $                  |        |
| 55 begin 58 NL 60 STMT 63 STMT  |                                 |        |
| 63 STMT 63 STMT 63 printf 67 (  |                                 |        |
| 76 VARNUM 83 operator 88 VARNUM |                                 |        |
| 92 ) 96 SC 98                   |                                 |        |
+---------------------------------+---------------------------------+--------+
```



```
+-----------------------------------+-----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | break SC NL end NL return number | r19    |
| 7 switch 15 ( 26 id 39 ) 48 NL    | SC NL $                           |        |
| 55 begin 58 NL 60 STMT 63 STMT    |                                   |        |
| 63 STMT 63 STMT 63 printf 67 (    |                                   |        |
| 76 VARNUM 83 operator 88 VARNUM   |                                   |        |
| 92 ) 96 SC 98 NL 100              |                                   |        |
+-----------------------------------+-----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | break SC NL end NL return number | s68    |
| 7 switch 15 ( 26 id 39 ) 48 NL    | SC NL $                           |        |
| 55 begin 58 NL 60 STMT 63 STMT    |                                   |        |
| 63 STMT 63 STMT 63 STMT 63        |                                   |        |
+-----------------------------------+-----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | SC NL end NL return number SC NL | s77    |
| 7 switch 15 ( 26 id 39 ) 48 NL    | $                                 |        |
| 55 begin 58 NL 60 STMT 63 STMT    |                                   |        |
| 63 STMT 63 STMT 63 STMT 63 break |                                   |        |
| 68                                |                                   |        |
+-----------------------------------+-----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | NL end NL return number SC NL $  | s84    |
| 7 switch 15 ( 26 id 39 ) 48 NL    |                                   |        |
| 55 begin 58 NL 60 STMT 63 STMT    |                                   |        |
| 63 STMT 63 STMT 63 STMT 63 break |                                   |        |
| 68 SC 77                          |                                   |        |
+-----------------------------------+-----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | end NL return number SC NL $     | r20    |
| 7 switch 15 ( 26 id 39 ) 48 NL    |                                   |        |
| 55 begin 58 NL 60 STMT 63 STMT    |                                   |        |
| 63 STMT 63 STMT 63 STMT 63 break |                                   |        |
| 68 SC 77 NL 84                    |                                   |        |
+-----------------------------------+-----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | end NL return number SC NL $     | r5     |
| 7 switch 15 ( 26 id 39 ) 48 NL    |                                   |        |
| 55 begin 58 NL 60 STMT 63 STMT    |                                   |        |
| 63 STMT 63 STMT 63 STMT 63 STMT   |                                   |        |
| 63                                |                                   |        |
+-----------------------------------+-----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | end NL return number SC NL $     | r4     |
| 7 switch 15 ( 26 id 39 ) 48 NL    |                                   |        |
| 55 begin 58 NL 60 STMT 63 STMT    |                                   |        |
| 63 STMT 63 STMT 63 STMT 63 STMTS |                                   |        |
| 73                                |                                   |        |
+-----------------------------------+-----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT   | end NL return number SC NL $     | r4     |
| 7 switch 15 ( 26 id 39 ) 48 NL    |                                   |        |
| 55 begin 58 NL 60 STMT 63 STMT    |                                   |        |
| 63 STMT 63 STMT 63 STMTS 73       |                                   |        |
+-----------------------------------+-----------------------------------+--------+
```



```
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | end NL return number SC NL $     | r4     |
| 7 switch 15 ( 26 id 39 ) 48 NL  |                                  |        |
| 55 begin 58 NL 60 STMT 63 STMT  |                                  |        |
| 63 STMTS 73                     |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | end NL return number SC NL $     | r4     |
| 7 switch 15 ( 26 id 39 ) 48 NL  |                                  |        |
| 55 begin 58 NL 60 STMT 63 STMTS |                                  |        |
| 73                              |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | end NL return number SC NL $     | s72    |
| 7 switch 15 ( 26 id 39 ) 48 NL  |                                  |        |
| 55 begin 58 NL 60 STMTS 62      |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | NL return number SC NL $         | s80    |
| 7 switch 15 ( 26 id 39 ) 48 NL  |                                  |        |
| 55 begin 58 NL 60 STMTS 62 end  |                                  |        |
| 72                              |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | return number SC NL $            | r16    |
| 7 switch 15 ( 26 id 39 ) 48 NL  |                                  |        |
| 55 begin 58 NL 60 STMTS 62 end  |                                  |        |
| 72 NL 80                        |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | return number SC NL $            | r15    |
| 7 switch 15 ( 26 id 39 ) 48     |                                  |        |
| SWTCH 54                        |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | return number SC NL $            | r7     |
| 7 SWITCHSTMT 9                  |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | return number SC NL $            | s16    |
| 7 STMT 7                        |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | number SC NL $                   | s29    |
| 7 STMT 7 return 16              |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | SC NL $                          | r14    |
| 7 STMT 7 return 16 number 29    |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | SC NL $                          | s40    |
| 7 STMT 7 return 16 VARNUM 27    |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | NL $                             | s49    |
| 7 STMT 7 return 16 VARNUM 27 SC |                                  |        |
| 40                              |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | $                                | r17    |
| 7 STMT 7 return 16 VARNUM 27 SC |                                  |        |
| 40 NL 49                        |                                  |        |
+---------------------------------+----------------------------------+--------+
```



```
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | $                                | r21    |
| 7 STMT 7 RETURSTMT 13           |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | $                                | r5     |
| 7 STMT 7 STMT 7                 |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | $                                | r4     |
| 7 STMT 7 STMTS 18               |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMT | $                                | r4     |
| 7 STMTS 18                      |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMT 7 STMTS | $                               | r4     |
| 18                              |                                  |        |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAIN 4 STMTS 6     | $                                | r2     |
+---------------------------------+----------------------------------+--------+
| 0 datatype 2 MAINFUNC 3         | $                                | r1     |
+---------------------------------+----------------------------------+--------+
| 0 S 1                           | $                                | acc    |
+---------------------------------+----------------------------------+--------+
(base) apple@Apples-MacBook-Air:~/Desktop/Projects/CD$
```

The above snapshots show the parsing steps for the input(problem statement) and since the input satisfies the grammar the input has been parsed successfully and is accepted by the grammar.

CONCLUSION

In lexical analysis when we give a program statement as input, the keywords, identifiers, operators numbers etc are displayed. Each of these are the tokens of the statement.

In Syntax analysis, on giving the output of lexical analysis as input (set of tokens), the input string is parsed as per the grammar and the parsing table. If the input string is parsed completely then it is accepted by the grammar or else there is an error which indicates that the grammar does not accepts the string.

BIBLIOGRAPHY

- PRINCIPLES OF COMPLIER DESIGN-ALFRED AHO, JEFFERY D. ULMAN

- https://stackoverflow.com/