|  |  |
| --- | --- |
| **Ex No: 2** | **SYMBOL TABLE GENERATION** |
| **Date of Exercise** | **21/12/2021** |

|  |
| --- |
| **AIM** |
| Write a program to scan the input program and generate a symbol table. |
| **DESCRIPTION** |
| Symbol table is a data structure containing a record for each identifier with fields for the attributes of the identifier. The attributes may give the information about the Storage allocated for the identifier, Data type and Scope of the identifier. By using these records each of the identifiers can be retrieved easily and quickly. During the Lexical Analysis the identifiers will be found and stored into the Symbol table but the attributes will not be stored in lexical phase. The Compiler and the Assembler will be having their own Symbol tables. The expression will be given as the input and the output obtained will be containing the table of information about the identifiers. |
| **ALGORITHM** |
| 1. Get an expression as input.  2. Separate each and every token.  3. If the token is id then create an entry in symbol table and enter the identifier name and its corresponding data type.  4. Check the data type of each identifier and assign the corresponding memory location.  5. Check whether the identifier is having a value, if so enter that in the symbol table for that identifier  6. If the token is already entered in the table, avoid repeated entry. |
|  |
|  |
| **QUESTION SET** |
| 1. Write a program to generate symbol table for a given input source program. |

**Program:**

import re

keywords = ['if','else','while','int','float','double','for'];

dataTypes = ['int','float','double','long int','long','long long int','long long']

symbolTable = []

def isKeyword(token):

if(token in keywords):

return True

return False

def isDatatype(token):

if(token in dataTypes):

return True

return False

def isIdentifier(token):

if(re.match(r"[a-z][\_0-9a-z]\*$",token)):

return True

return False

def isOperator(token):

if(re.match(r"[<>=+-/\*]+",i)):

return True

return False

def isPunctuation(token):

if(re.match(r"[;,'\".]+$",i)):

return True

return False

def extractFunction(temp,tokens,i):

parameters = []

while(tokens[i]!=')'):

if(isDatatype(tokens[i])):

parameters.append(tokens[i])

i+=1

elif(tokens[i]==','):

i+=1

else:

print(tokens[i])

return False

temp.append('-')

temp.append(len(parameters))

temp.append(', '.join(i for i in parameters))

return temp

def addNewTableEntry(tokens):

i=0

temp = []

if(isKeyword(tokens[i]) and (isDatatype(tokens[i]))):

i+=1

# Edge case

if i >= len(tokens):

return False

if(isIdentifier(tokens[i])):

i+=1

if(tokens[i]=='('):

# Adding Identifier

temp.append(tokens[i-1])

temp.append('-')

# Adding dataTypes

temp.append(tokens[i-2])

i+=1

temp = extractFunction(temp,tokens,i)

return temp

elif(tokens[i]==';' or tokens[i]==',' or tokens[i]=='='):

init\_id = tokens[i-1]

init\_data = tokens[i-2]

while(True):

temp = []

# Adding Identifier

temp.append(init\_id)

# Adding dataTypes

temp.append(init\_data)

if(tokens[i]==';'):

temp.append('-')

temp.append(0)

temp.append('-')

temp.append('-')

return temp

elif(tokens[i]=='='):

temp.append('-')

i+=1

if i >= len(tokens):

return False

temp.append(tokens[i])

temp.append('-')

temp.append('-')

symbolTable.append(temp)

i+=1

if(tokens[i]==';'):

return True

else:

i+=1

if(isIdentifier(tokens[i])):

init\_id = tokens[i]

i+=1

else:

return False

elif(tokens[i]==','):

temp.append('-')

temp.append(0)

temp.append('-')

temp.append('-')

symbolTable.append(temp)

i+=1

if(isIdentifier(tokens[i])):

init\_id = tokens[i]

i+=1

else:

return False

else:

return False

else:

return False

else:

return False

else:

return False

def printSymbolTable():

print("ID\tData Type\tReturn Type\tInitial Value\tNumber of parameters\tType of parameters")

for i in symbolTable:

for j in i:

print(j,end="\t")

print()

idCnt = 0

tokenCnt = 0

while True:

breakFlag = False

tokens = re.split('([^a-zA-Z0-9\_])',input())

tokens = list(filter(lambda x: x != " " and x !='', tokens))

if(tokens[len(tokens)-1]=="END"):

breakFlag = True

if breakFlag:

printSymbolTable()

break

response = addNewTableEntry(tokens)

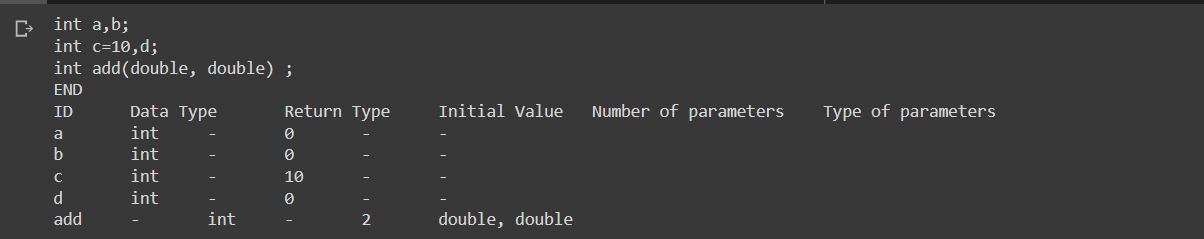
if(type(response)==list):

symbolTable.append(response)

elif(response==False):

print("The given input doesn't come under the two types mentioned in the problem statement")

**Output Screenshot**



**Result**

Thus the program ran successfully and the output was verified.