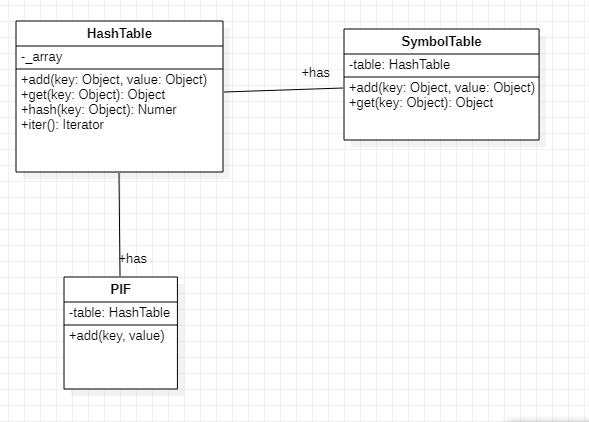
**Lab 3**

Repo: <https://github.com/cinnamonbreakfast/flcd/tree/main/lab3>

**Lab 3 Documentation**

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Pre-condition | Post-condition | Observations |
| Detect(program) | <program>: string – valid name for an input source file | <program>\_PIF.out : contains PIF data  <program>\_ST.out : contains ST data | Using Regex to fetch the program, then send the tokens through filters |
| is\_ident\_const(cod) | cod : token | Boolean | Function checks if the current token is an identifier or constant |
| is\_reserved(cod) | cod : token | Boolean | Checks if the current token is a reserved work or token. |



The algorithm works as following:

It reads the file content and splits by every token (we end up with a list of every token, eg: [entry, {, int, number, ..etc]). Then, we parse the list and we check ***if the token is a reserved word, operator or separator***, and add it to PIF. Otherwise, we check **if the token is an identifier or constant** and add it to SymbolTable, get the index inside SymbolTable, and add the token together (pair) in PIF. **If none** of these two conditions are fulfilled, **there is a lexical error**.

code\_data = re.split('([^a-zA-Z0-9])', line)  
  
code\_data = list(filter(*None*, code\_data))  
code\_data = map(*lambda e*: e.strip(), code\_data)  
code\_data = list(filter(*None*, code\_data))  
  
*for* e *in* code\_data:  
 *if*(is\_reserved(e)):  
 pif.add(e, 0)  
 *elif* is\_ident\_const(e):  
 index = 0  
 *try*:  
 index = st.add(e, 0)  
 pif.add(e, index)  
 *except*:  
 *continue  
 else*:  
 print("Lexical error for " + e)

input:

entry {

int number;

number = 3;

if(number > 5) {

WRITE("SARMALE");

}

}

ST.out is:

Using a HashTable:  
[['number', 0], ['3', 0]]

[]

[['5', 0], ['SARMALE', 0]]

PIF.out is:

('{', 0)

('int', 0)

('number', 0)

(';', 0)

('=', 0)

('3', 1)

(';', 0)

('if', 0)

('(', 0)

('>', 0)

('5', 0)

(')', 0)

('{', 0)

('WRITE', 0)

('(', 0)

('"', 0)

('SARMALE', 1)

('"', 0)

(')', 0)

(';', 0)

('}', 0)

('}', 0)