Link for github repo:

https://github.com/AndreiFoidas/LFTC

https://github.com/AndreiFoidas/LFTC/tree/main/src/main/java

Requirement

Implement a scanner (lexical analyzer): Implement the scanning algorithm and use a custom implemented Symbol Table.

Symbol Table

The symbol table is a custom Hash Table and it runs its operations in O(1). The Hash Table is an ArrayList with a LinkedList on each position. Having the LinkedList ensures that the conflicts are solved, because if two elements hash to the same position, they will be added to the same LinkedList (collision solved by Chaining).

Hash Function

Represents the sum (% the size of the symbol table) of the ASCII code of each character of the given string.

Operations

- add(key): adds the given key to the symbol table (if it doesn't already exist)
- contains(key): returns a Boolean value which indicates whether the given value exists in the symbol table
- search(key): returns a pair of integers that represent the position in the array list and the position in the respective linked list
- hash(key): returns the hash value of the given key

Codification

The Codification class is implemented using a hash map: the key of the hash table is a token (a String) and the value for each token is a unique code that is generated by the following rule: 0 for identifier, 1 for constant and from 2 onwards, unique codes generated incrementally for separators, operators and reserved words.

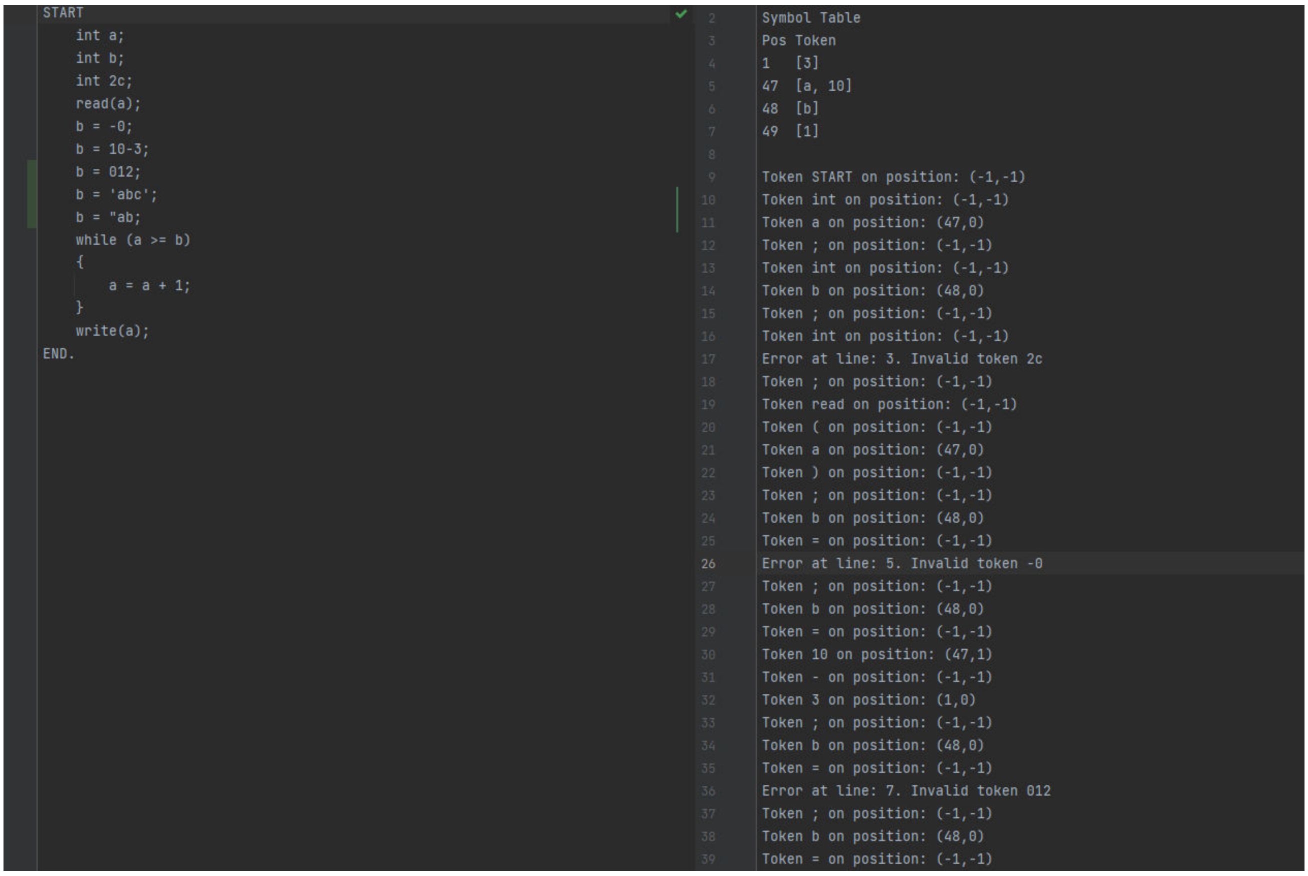
ProgramInternalForm

The ProgramInternalForm is implemented using a hash map: the key represents the code of each token (defined in the Codification class), and the value represents the pair from the SymbolTable for identifiers and constants and a pair (-1, -1) for separators, operators and reserved words.

Scanner

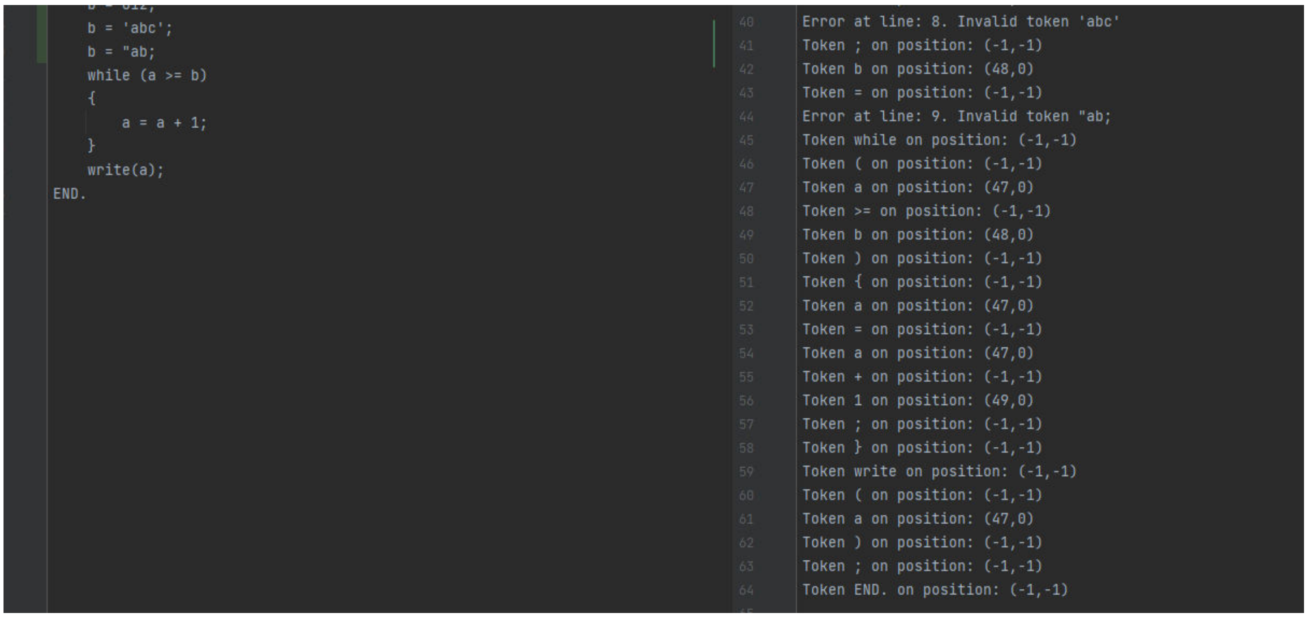
It starts by splitting the program line by line and in each line it will look for the tokens that compose it. If it is a separator, operator or reserved word the position will be the pair (-1, -1). For constants and identifiers it will look up their positions in the SymbolTable. Every token will be added to the PIF with the corresponding code from the Codification class and with its position from the SymbolTable or the (-1, -1) pair accordingly,

Tests



```
START
    int a;
    int b;
    int 2c;
    read(a);
    b = -0;
    b = 10-3;
    b = 012;
    b = 'abc';
    b = "ab;
    while (a >= b)
    {
        a = a + 1;
    }
    write(a);
END.
```

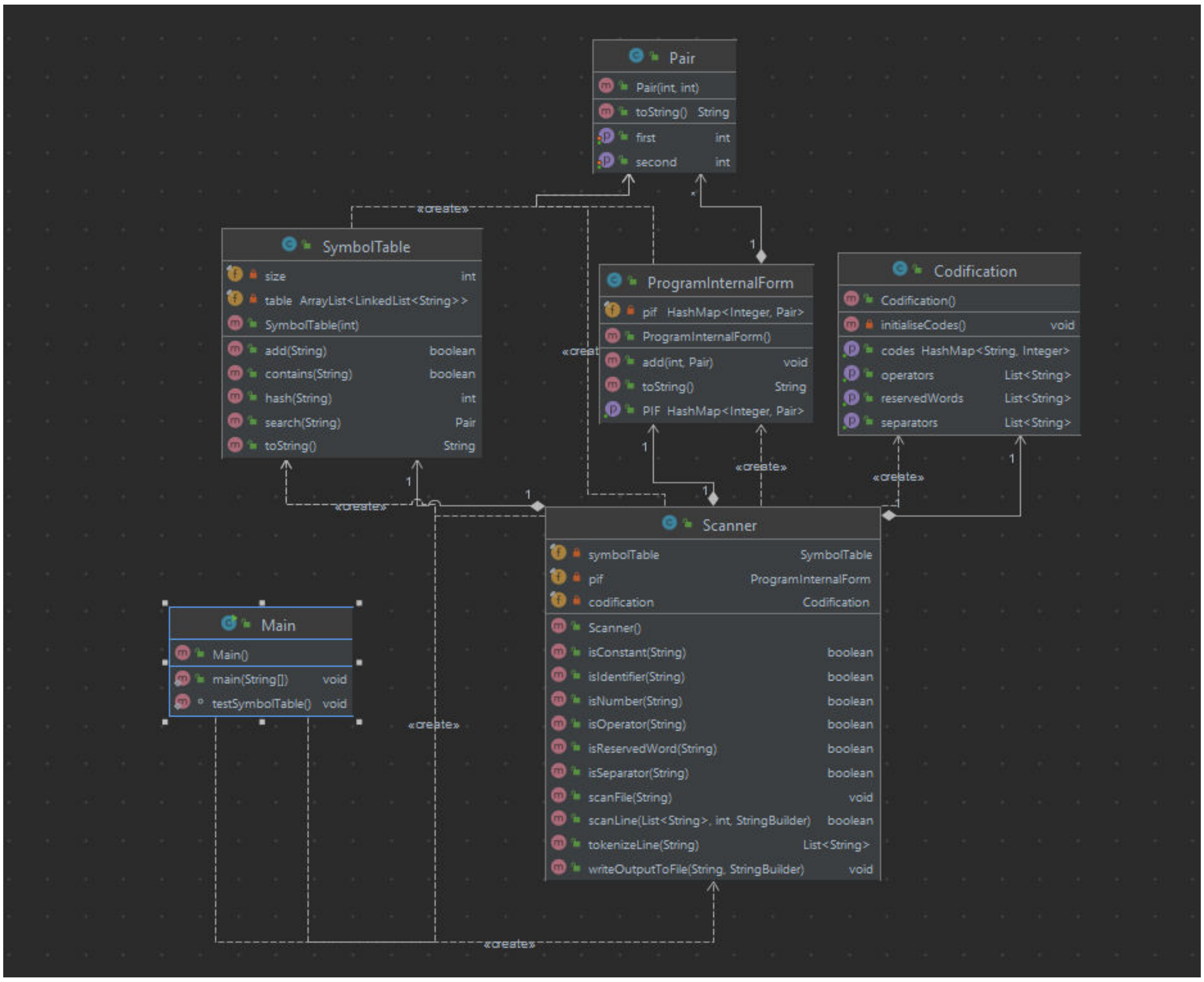
```
Symbol Table
Pos Token
1   [3]
47  [a, 10]
48  [b]
49  [1]

Token START on position: (-1,-1)
Token int on position: (-1,-1)
Token a on position: (47,0)
Token ; on position: (-1,-1)
Token int on position: (-1,-1)
Token b on position: (48,0)
Token ; on position: (-1,-1)
Token int on position: (-1,-1)
Error at line: 3. Invalid token 2c
Token ; on position: (-1,-1)
Token read on position: (-1,-1)
Token ( on position: (-1,-1)
Token a on position: (47,0)
Token ) on position: (-1,-1)
Token ; on position: (-1,-1)
Token b on position: (48,0)
Token = on position: (-1,-1)
Error at line: 5. Invalid token -0
Token ; on position: (-1,-1)
Token b on position: (48,0)
Token = on position: (-1,-1)
Token 10 on position: (47,1)
Token - on position: (-1,-1)
Token 3 on position: (1,0)
Token ; on position: (-1,-1)
Token b on position: (48,0)
Token = on position: (-1,-1)
Error at line: 7. Invalid token 012
Token ; on position: (-1,-1)
Token b on position: (48,0)
Token = on position: (-1,-1)
```



```
    b = 'abc';
    b = "ab;
    while (a >= b)
    {
        a = a + 1;
    }
    write(a);
END.
```

```
Error at line: 8. Invalid token 'abc'
Token ; on position: (-1,-1)
Token b on position: (48,0)
Token = on position: (-1,-1)
Error at line: 9. Invalid token "ab;
Token while on position: (-1,-1)
Token ( on position: (-1,-1)
Token a on position: (47,0)
Token >= on position: (-1,-1)
Token b on position: (48,0)
Token ) on position: (-1,-1)
Token { on position: (-1,-1)
Token a on position: (47,0)
Token = on position: (-1,-1)
Token a on position: (47,0)
Token + on position: (-1,-1)
Token 1 on position: (49,0)
Token ; on position: (-1,-1)
Token } on position: (-1,-1)
Token write on position: (-1,-1)
Token ( on position: (-1,-1)
Token a on position: (47,0)
Token ) on position: (-1,-1)
Token ; on position: (-1,-1)
Token END. on position: (-1,-1)
```

Class Diagram


Pair
Pair(int, int)
toString() String
first int
second int
«create»
SymbolTable
size int
table ArrayList<LinkedList<String>>
SymbolTable(int)
add(String) boolean
contains(String) boolean
hash(String) int
search(String) Pair
toString() String
ProgramInternalForm
pif HashMap<Integer, Pair>
ProgramInternalForm()
add(int, Pair) void
toString() String
PIF HashMap<Integer, Pair>
Codification
Codification()
initialiseCodes() void
codes HashMap<String, Integer>
operators List<String>
reservedWords List<String>
separators List<String>
Scanner
symbolTable SymbolTable
pif ProgramInternalForm
codification Codification
Scanner()
isConstant(String) boolean
isIdentifier(String) boolean
isNumber(String) boolean
isOperator(String) boolean
isReservedWord(String) boolean
isSeparator(String) boolean
scanFile(String) void
scanLine(List<String>, int, StringBuilder) boolean
tokenizeLine(String) List<String>
writeOutputToFile(String, StringBuilder) void
Main
Main()
main(String[]) void
testSymbolTable() void
«create»
«create»
«create»
«create»
«create»
«creat
1
1
1
1
1
1