**Lexical Analysis(Scanner)**

# What is a Lexical Analyzer?

A Lexical Analyzer or Scanner is an algorithm that groups the characters of the source code to form the **Tokens**, moreover, it returns the **Internal Representation Number** of these tokens, which is a kind of an **ID** that assigned for each token.

These tokens are divided into 3 kinds:

1. **Names** : Which is any name we have in a program. These in turn are divided into 2 types:
2. **Keywords/Reserved**: which are words such as if/else/while. These names can’t be used as variable names. They have a specific place and function

b. **User Defined Names**, Which are the names declared by the user.

1. **Values** : such as integers(1, 2, 3, 4) or floating point(1.1, 2.34, 5234.123) et
2. **Special Symbols/Tokens** : And these are the logical(==, &&, ||) and arithmetic operations(+, -, \*, /), parenthesis([], {}, ()), or any other tokens that are not from the first or second kind.

Let us apply the scanner to this short segment of code:

while(x>=100)

{

n +=x;

x++

}

This results in this set of tokens :

While , ( , x , >= , 100 , ) , { , n , += , x , ; , x , ++ , } .

Referencing these tokens against a certain keywords table like this one :

|  |  |
| --- | --- |
| **index** | **Symbol** |
| .. | .. |
| 33 | While |
| .. | .. |
| 67 | >= |
| .. | .. |
| .. | .. |

Leads us to these ID's :

|  |  |
| --- | --- |
| **Token** | **Internal Representation Number** |
| While | 33 |
| ( | 84 |
| x | 100 |
| >= | 67 |
| 100 | 200 |
| ) | 85 |
| { | 92 |
| n | 100 |
| += | 77 |
| x | 100 |
| ; | 81 |
| x | 100 |
| ++ | 75 |
| ; | 81 |
| } | 93 |

note that all user defined names have the same number. This is because to the syntax analyzer, it doesn't matter what the variable is, it just matters that there is a variable there.

# Type Checking implementation

During this process of analysis, The Compiler builds what is called the **Symbol Table**. The Symbol Table is the table which contains the user defined name(mostly variables), its type, and its values. The Symbol table for this segment of code would be:

.

.

.

int compute(int,int);

int n;

float x,y;

const int m=10;

.

.

.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Data type** | **Value** |
| compute | function-name | integer | 0 |
| n | varaiable | integer | 0 |
| x | variable | float | 0 |
| y | variable | float | 0 |
| m | constanmat | integer | 10 |

To perform type checking, the compiler takes the name, and checks the **keywords table**, if it is not in the keywords table, it is a **user defined** variable. if it is a user defined variable, it then goes to check the symbol table. If it is not defined in the symbol table, it returns that the variable is not defined (unknown symbol/variable deceleration error), if it is in the symbol table, it retrieves its type. If the operation being performed on the variable is not compatible with the type of the variable, it returns that the operation is not compatible( mismatch - type error ).



