Lab No. 02 – Advanced Lexical Analyzer

Creating patterns using regular expressions:

Let's dig deeper into building regular expressions. Here are some samples and their corresponding regex.

Sl.	Pattern	REGEX
1.	Any decimal numbers	`[0-9]+`
2.	Any float or double numbers	`[0-9]+\.[0-9]+`
3.	Any decimal number from 72 to 1235	`7[2-9]|[8-9][0-9]|[1-9][0-9][0-9]|1[0-1][0-9][0-9]|12[0-2][0-9]|123[0-5]`
4.	Any double number from 12.02 to 99.93	`12\.(0[2-9]|[1-9][0-9])|1[3-9]\.[0-9][0-9]|[2-8][0-9]\.[0-9][0-9]|9[0-8]\.[0-9][0-9]|99\.([0-8][0-9]|9[0-3])`
5.	Any class A IP address	`([0-9]|[0-9][0-9]|1[0-1][0-9]|12[0-7])(\.(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[0-9][0-9]|[0-9])){3}`
6.	Any alphabet string that starts with a capital letter vowel	`^[AEIOU][a-zA-Z]*`
7.	Any alphabet string that ends with a small letter vowel	`[a-zA-Z]*[aeiou]$`
8.	Any string that has a number in it	`[a-zA-Z0-9]*[0-9]+[a-zA-Z0-9]*`

Explanation of the patterns:

- **Pattern 1 - Decimal numbers:** As all decimal numbers are composed of the character 0-9 so a simple pattern of repeating 0-9 for 1 or infinite times is enough to recognize any decimal numbers.
- **Pattern 2 - Double/Float Numbers:** Double numbers vary from decimal numbers by a (.) between the numbers. So on both sides we have the same pattern except a (.) in the middle.
- **Pattern 3 - Decimal number range:** We need to treat the numbers as characters so we need to break down the numbers into recognizable ranges. See that, we have divided the numbers from 72 – 79, 80 – 99, 100 – 999, 1000 – 1199, 1200 – 1229 and 1230 – 1235, and then just created REGEX for each of the ranges and used an OR operation among them.
- **Pattern 4 - Double numbers range:** Same principle as decimal numbered ranges except we need an extra (.) between the numbers.
- **Pattern 5 - IP addresses:** As we are using the class A IP address i.e., 0.0.0.0 – 127.255.255.255, so the 4th Quadrant has the pattern 0-127, and the rest of the three quadrants have the same pattern of 0-255. So, we repeated the pattern for 3 quadrants using the {value} operator.
- **Pattern 6 - Vowel letter starting:** To match at the start of a string we used a (^) sign. The rest are only for all possible alphabets.
- **Pattern 7 - Vowel letter ending:** Same principle as above except for matching at the end of the string we used a ($) operator.
- **Pattern 8 - Number within a string:** As the number can be either at the front or middle or end of a string we enclosed it with possible pattern of a string and repeated the string patterns for 0 or infinite times.

Complete Code:

Now, let’s combine the given patterns into a complete program.

```
%{
//For today we do not need any variables
%}
%%
[0-9]+ {printf("%s - Found a decimal number.\n",yytext);}
[0-9]+\.[0-9]+ {printf("%s - Found a double number.\n",yytext);}
123[0-5]|12[0-2][0-9]|1[0-1][0-9][0-9]|[1-9][0-9][0-9]|[8-9][0-9]|7[2-9] {printf("%s - Found a decimal number within 72 - 1235.\n",yytext);}
12\.(0[2-9]|[1-9][0-9])|1[3-9]\.[0-9][0-9]|[2-8][0-9]\.[0-9][0-9]|9[0-8]\.[0-9][0-9]|99\.([0-8][0-9]|9[0-3]) {printf("%s - Found a double number within 12.02 - 99.93.\n",yytext);}
([0-9]|[0-9][0-9]|1[0-1][0-9]|12[0-7])(\.(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[0-9][0-9]|[0-9])){3} {printf("%s - Found a class A IP address.\n",yytext);}
^[AEIOU][a-zA-Z]* {printf("%s - Found an alphabet string that starts with a capital letter vowel.\n",yytext);}
[a-zA-Z]*[aeiou]$ {printf("%s - Found an alphabet string that ends with a small letter vowel.\n",yytext);}
[a-zA-Z0-9]*[0-9]+[a-zA-Z0-9]* {printf("%s - Found a string that has a number in it.\n",yytext);}
%%
int main(){
      FILE *file;
      file = fopen("code.c", "r") ;
      if (!file) {
            printf("couldnot open file");
            exit (1);
      }
      else {
            yyin = file;
      }
      yylex();
}
```

Now write some possible tokens in a file called “code.c” and run the program to check which lexemes are accepted.

Tasks for LAB 02:

1. Create regular expressions for accepting the following strings:
   a. Any double numbers of the range 23.343 - 99.999
   b. Any string that starts and ends with a vowel, e.g., apostle, oscillate etc.
   c. Any class B IP address.
   d. Any alphanumeric strings that either starts with or ends with a digit.
   e. Any possible name of a person (think of syntaxes followed in naming of a person)
   f. Valid student IDs from batches 61 – 70 of CSE department.