**Document statistics – prolog implementation**

**Input : A text file**

**Output**: Four text files: dictionary of words, Char---number of times, word----number of times, multi-word---number of times.

**Algorithm：**

1.change the contents of the text file to a list of Chars.---- list\_of\_char

2.iterate the List\_of\_char to get Two lists: ---list\_of\_word, list\_of\_three\_word

3.iterate the List\_of\_char, list\_of\_word,list\_of\_three\_word to get three Symbol table:

Char\_ST (key: char, value: number of times).

Word\_ST(key: word,value: number of times).

Three\_word\_ST(key:word,value: number of times).

4.use sort() method to eliminate the duplicate items in list\_of\_char, list\_of\_word,list\_of\_three\_word.

5. sort the list\_of\_word according to a-z order, write the result to a text file--- dictionary.txt

6. according to the “value” of each symbol table, sort the list\_of\_char, list\_of\_word, list\_of\_three\_word.

Write the result to three text file---char\_number.txt, word\_number.txt, multi\_word\_number.txt.

**Key characteristics of prolog:**

Logical program: functions are the main components of the program

Dynamic type binding.

Use capitalized string to represent variable, no variable initialize. If you want to declare a variable “X” whose value is integer or float point, you can use this statement” X is 10”. But if you want to declare other type variable, you’d better use the variable as arguments.

equal(X,Y):-

X = Y.

You Can use equal(a,Y), then Y is a variable that equal to atom a.

Use “,” to represent logical and, “;” to represent logical or. Use “!” to terminate backtracking of the statement.

**Syntax:**

Function(A,B):-

Function1(A,B),

Function2(A,B).

If function1() is true and function2() is true, then function() is true. The argumens will equal the value that could make the function to be true.

**Key Point：**

**How to loop in prolog?**

Example code:

% change the list of strings to a list of atoms.

% base condition

strings\_to\_atoms([],OldList,AtomList):-

AtomList = OldList,

!. % ! means no backtracking happens.

% input list [X|List] , which means the input list consists of X and the remained List. OldList is generally empty list []. AtomList is used to store the return value, it will not change until the last step when the input list is [], the Atomlist store the return value.

strings\_to\_atoms([X|List],OldList,AtomList):-

atom\_string(Y,X), % Change the string X to atom Y

strings\_to\_atoms(List,[Y|OldList],AtomList). % recursively call the function.

**How to use if, else statement in prolog?**

Example:

Function(A,B):-

(function1(C,D), % if C,D can make function1() true, then function2

Function2(E,F);

Function3(G,H)), % else-> function3(G,H).

Function4(I,J). % after the if-else statement, we execute the function4(I,J).