I have designed 2 classes to complete this assignment

They are

1. Gerp:

The gerp implementation, it has a hash table as its private member, and a vector for record the files’ information of a FSTree.

1. HashTable. This is the main data structure for this assignment, which will index the content of each file, record the word’s information( the file path, the line number…).

And there are some structs for help:

* File

typedef struct {

std::string path;

vector<std::string> lines;

}File;

This is used to store the file information in a file vector:

std::vector<File> files;

while building the index, the program firstly traverse the file tree to generate this file vector, std::string path is the file path name of a file , vector<std::string> lines; has all of the lines of a file, the subscript of this vector equals the line number-1.

Since a file can be represent by its position in the vector<File> files,This could be called the FileID. So it is needn’t to store all of these information in the hash table, just use a FileID, which can reduce some memory cost.

* Entry of HashTable

typedef struct {

int fileID; // the fileID can help us to find the file's path

int lineNumber; //the lineNumber helps us to locate the line in fileID file

}Value;

typedef struct Entry {

std::string key; // key is used for query. It is a word.

size\_t hash; //hash used for speed up the re-hash algorithm

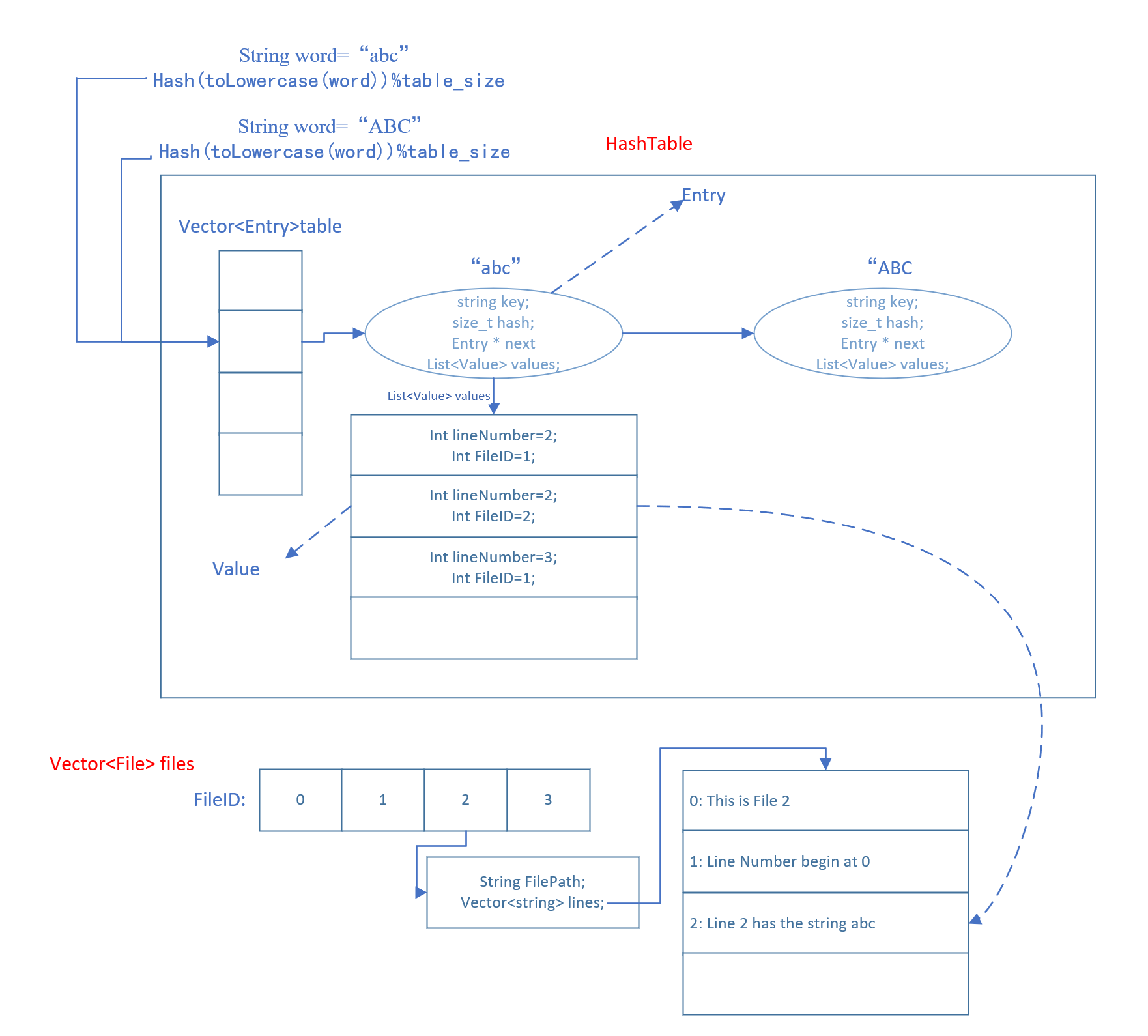
std::list<Value> values; // a list of Value, it contains all the word’s occurrence in each file.

Entry\* next; //to the next entry

}Entry;

So a hash table is something like Entry[] hash\_table;

A possible structure of the hash table could be :



The class HashTable may have these public interface:

size\_t getIndex(std::string original\_string);// compute the hash of a string using its lowercase

void reHash(); // the rehash algorithm for expanding the capacity of hash table

std::list<Value>& getValueList(std::string key); // get a word’s value list, it contains the occurrence of this word and used for output.

bool contains(std::string key); // predicate if a word has been in the hash table

void insert(std::string key, Value value); //insert a word’s occurrence into hash table. This will be called while building index.

* Gerp Class

The possible interfaces and data members that I designed of Gerp class are:

//generate the vector filePaths by the FSTree

void generateFileVec(FSTree &tree);

//build index of each file's content using filePaths vector

void buildIndex()

//query a word true: found it ; false: not found

bool query(std::string word, bool isCaseInsensitive)

//save the query result to the output file

void dump(std::vector<Value> values, bool isCaseInsensitive)

//runs the gerp command

void run(FSTree & tree,std::string outputFileName)

// firstly build the index(hashtable).

// then use a loop for getting command from input, and execute the command

void run(FSTree & tree,std::string outputFileName)

private:

HashTable hashTable; //a hash table for query and store the index

std::string outputFileName; // the file where the query information output to;

std::vector<File> files; // each file's information the path and content of each line,

//the subscript of this vector is the ID of the file, this will be used in hashtable

The flow chart of this program could be :

