Darrell Percey

Data Structure II

Project 1

Functional Decomposition

**Data Structures**

struct FileNode {

char fileName[NAMELENGTH];

char fileWords[SENTENCESTORAGE];

HashInfoP fileHash[HASHLENGTH];

FileNodeP next;

} FileNode;

Contains all file information. Is a link list of files name sentence and hash table.

struct HashInfo{

HashInfoP next;

char data[HASHINFOLENGTH];

} HashInfo;

Hashnode for hold the information for our hash table. This will be used to compare information.

int fileCount = 0;

Allows me to tell how many files there is.

**Files and Functions**

The main file calls for the math to be done in the matfuncs file and for the simulation to be done in the queue file.

**FileRun.c**

Runs a couple of the Functioins from FileCheck.c to execute the comparsion of files.

**FileCheck.c**

*/\**

*\* Function: hashCode()*

*\**

*\* Description:*

*\* Takes in a data string and sums the ASCII values*

*\* of all the letters and multiples in an i++ fashion.*

*\**

*\*/*

*/\**

*\* Function: getWordCount()*

*\**

*\* Description:*

*\* Ask for the user to input an amount of words they want to compare*

*\* range can be from 2 to 10 and it has error checking.*

*\* It returns the value back to main for phraseSize*

*\**

*\*/*

*/\**

*\* Function: newFileNode(char\*)*

*\**

*\* Description:*

*\* This allocates memeory needed for a new FileNode*

*\* Also returns the temp node back to a pointer for FileNode*

*\**

*\*/*

*/\**

*\* Function: addNewFileNode(char\*, FileNodeP)*

*\**

*\* Description:*

*\* This adds a FileNode to the end of he linked listed FileNodes*

*\* Finds the end then allocates memeory using newFileNode()*

*\**

*\*/*

*/\**

*\* Function: newHashNode(char\*)*

*\**

*\* Description:*

*\* Allocates the memeory for a hashNode and stores the data*

*\* into it for comparsion later. Returns the pointer to a*

*\* HashInfoP in the FileNode array*

*\**

*\*/*

*/\**

*\* Function: addNewFileNode(char\* HashInfoP)*

*\**

*\* Description:*

*\* If there is a collision this function is called*

*\* it will add the new node to the end of the HashInfo*

*\* linked list.*

*\**

*\*/*

*/\**

*\* Function: readFilesIn(FileNodeP)*

*\**

*\* Description:*

*\* Reads the entire string in the file into an string*

*\* array for easier access and for easy of change.*

*\**

*\*/*

*/\**

*\* Function: getFiles()*

*\**

*\* Description:*

*\* Reads all the names of the files in the /datafiles/*

*\* directory and inputs those names into a file named*

*\* inputfile. Also stores the names of the files in the*

*\* FileNodes to start an link list.*

*\**

*\*/*

*/\**

*\* Function: freeNodes(FileNodeP)*

*\**

*\* Description:*

*\* Frees all the memeory allocated in the FileNode*

*\* and all the HashNodes that are connected to that*

*\* FileNode.*

*\**

*\*/*

*/\**

*\* Function: hashFileContent(int, FileNodeP)*

*\**

*\* Description:*

*\* This breaks the file string into phrases of user picked*

*\* sizes and stores them in the HashTable array inside of*

*\* the FileNode structure by passing it to placeHash().*

*\**

*\*/*

*/\**

*\* Function: placeHash(char\* FileNodeP)*

*\**

*\* Description:*

*\* Places the data and in the corresponding FileNode's*

*\* hashtable array in the position of the hashKey that*

*\* is found from hashCode()*

*\**

*\*/*

*/\**

*\* Function: compareFiles(int, FileNodeP)*

*\**

*\* Description:*

*\* This function takes in the top file and compares all other files*

*\* to that orignial one. Then it will call to free the memory and move*

*\* to the next file. In doing so it will then compare that file to the rest.*

*\* Essentially it is a queue type moving down the list.*

*\*/*

*/\**

*\* Function: compareWords(int, FileNodeP, FileNodeP)*

*\**

*\* Description:*

*\* Compares the words of to the two files passed in*

*\* and then returns a value of the amount of phrases*

*\* that were the same.*

*\**

*\*/*

**Conclusion**

Had a file read problem at first but overall it came down to the text files being whacky. I included 4 working datasets in my folder.