c\_Lobes.h Function List

Private Functions:

/\* Private Function to search the map to see if

the address (token value of the data) exists.

if it does, modifies the address of the bool at Result to TRUE and

ALWAYS returns the iterator.

= end() if not found.

\*/

unordered\_map <int, c\_MemoryCell>::iterator **FindAddressInMap**(int Address, bool &Result, char SideToCheck = 'r')

Public Functions:

**string MakeStringLowerCase**(string strData)

//Updates the RawStringData if exists, adds it if it doesn't exist

//returns True if added, else false if updated

bool **SetMemoryCellRawStringData**(string strData, int Address, char SideToCheck = 'r')

//Gets the RawStringData if exists,

//returns the string value, else bool address of Result modified to false if not found

string **GetMemoryCellRawStringData**( bool &Result, string strData = "", int Address = -1, char SideToCheck = 'r')

//Updates the LowerCaseStringData if exists, adds it if it doesn't exist

//returns True if added, else false if updated

bool **SetMemoryCellLCStringData**(string strData, int Address, char SideToCheck = 'r')

//Returns NULL if doesn't exist, else returns CellWordType

char **GetMemoryCellWordType**(int Address)

//Returns NULL if doesn't exist, else returns CellWordDataLC

string **GetMemoryCellWordLC**(string WordInUC, int Address = -1, char SideToCheck = 'r')

//Returns TRUE if exist and the the WordType was set, else returns false

bool **SetMemoryCellWordType**(int Address,char WordTypeToSet)

//Returns TRUE if exist and the the Purpose was set, else returns false

bool **SetMemoryCellPurpose**(int Address,char CellPurpose)

//Returns char cell purpose

char **GetMemoryCellPurpose**(int Address)

//Returns -1 if cell doesn't exist else returns pointer to next pattern;

int **GetMemoryCellPointerToNextPattern**(int Address,char SideToCheck = 'l')

//Returns -1 if cell doesn't exist else sets the data and returns 0

int **SetMemoryCellPointerToNextPattern**(int Address, int PointerToNextAddress, char SideToCheck = 'l')

//Returns NULL if cell doesn't exist, else returns char value of WordTense

char **GetMemoryCellWordTense**(int Address)

//Returns TRUE if exists and data is set, else returns false

bool **SetMemoryCellWordTense**(int Address, char TenseToSet)

//Returns TRUE if exists and data is set, else returns false

bool **SetMemoryCellGenderClass**(int Address, char GenderToSet)

//Returns char GenderClass from memory cell

char **GetMemoryCellGenderClass**(int Address)

//Returns -1 if doesn't exist, else returns adjective count

int **GetMemoryCellAdjectiveCount**(int Address)

//Checks to see if there is a memorycell for the Address given.

//returns true if so, if not returns false.

bool **GetMemoryCellIsSet**(int Address,char SideToCheck = 'r')

//First checks to see if the cell at the address given exist.

//if not, returns -1.

//otherwise, returns the Adjective count and

//by reference, the tokenized adjective value in the array Adjectives[]

int **GetMemoryCellAdjectives**(int Address,int Adjectives[])

//First checks to see if the cell at the address given exist.

//if not, returns -1.

//otherwise, returns the RelatedNoun count and

//by reference, the tokenized RelatedNoun value in the array RelatedNouns[]

int **GetMemoryCellRelatedNouns**(int Address,int RelatedNouns[])

//First checks to see if the cell at the address given exist.

//if not, returns -1.

//otherwise, returns the Verb count and

//by reference, the tokenized Verb value in the array Verbs[]

int **GetMemoryCellVerbs**(int Address,int Verbs[])

//First checks to see if the cell at the address given exist.

//if not, returns -1.

//otherwise, returns the Adverb count and

//by reference, the tokenized Adverb value in the array Adverbs[]

int **GetMemoryCellAdverbs**(int Address,int Adverbs[])

//First makes sure the address given exists,

//then associates adjective to the address given.

//NOTE:will not duplicate Adjectives.

bool **AssociateMemoryCellAdjective**(int Address, string AdjectiveToAssociate)

//First makes sure the address given exists,

//then associates Noun to the address given.

//NOTE:will not duplicate Nouns.

bool **AssociateMemoryCellNoun**(int Address, string NounToAssociate)

//First makes sure the address given exists,

//then associates Adverb to verb in the address given.

//NOTE:will not duplicate Adverbs.

bool **AssociateMemoryCellAdverbToVerb**(int Address, string AdverbToAssociate, string VerbToAssociate)

//First makes sure the address given exists,

//then associates Verb to adjective in the address given.

//NOTE:will not duplicate Adverbs.

bool **AssociateMemoryCellVerbToAdjective**(int Address, string AdverbToAssociate, string VerbToAssociate)

//First Checks to see if cell address given exists,

//if so, checks to see if NounToCheck is related to this cell address.

//Returns true if so, false if not or cell doesn't exist

bool **GetIsNounRelatedToThisMemoryCell**(int Address, string NounToCheck)

//returns the Memory cell count in RightLobe memory map

int **GetRightLobeCellCount**()

//returns the Memory cell count in LeftLobe memory map

int **GetLeftLobeCellCount**()

//Checks to see if the NewWord exists first,

//if so AND Update = true, Updates the data, returns Installed = false.

//If NewWord doesn't exist,

//Stores all the data and returns Installed = true.

bool **InstallNewWord**(string NewWord, char WordType, char Purpose, bool Update=false)

bool **CheckLinkOfTwoNounsWithAdjectives**(string FirstNoun,string SecondNoun, string& VerbUsage, string MatchedAdjective[], int& MatchedCount)

//stores Pre and Post Construction Patterns.

//if doesn't exist, adds it

//if does exist, updates it

void **SavePreAndPostPatternConstruction**(string PreConstructionPattern,string PostConstructionPattern)

//Returns miniDefinition from memorycell

//Returns "" if memorycell doesn't exist

//can send string data or tokenized string data

//currently rightside cells only, don't see a need for leftside yet

string **GetMemoryCellMiniDefinition**(string strData = "", int Address = 0)

//Sets the memorycell minidefinition if the memorycell exists

//can send string data or the tokenized version of the string data for the address

void **SetMemoryCellMiniDefinition**(string Definition, string strData = "", int Address = 0)

//Sets pCellIsSingular in memorycell

//if memorycell exist

//can send string data or tokenized string data

//currently right side cells only, don't see a need for left side yet

void **SetMemoryCellpCellIsSingular**(char newData, string strData = "", int Address = 0)

//Returns pCellIsSingular from memorycell

//Returns '\0' null, if memorycell doesn't exist.

//can send string data or tokenized string data

//currently right side cells only, don't see a need for left side yet

char **GetMemoryCellpCellIsSingular**(string strData = "", int Address = 0){

//Sets pSingularLocation from memorycell

//if memorycell exist

//can send string data or tokenized string data for address

//send int for new location

//currently right side cells only, don't see a need for left side yet

void **SetpSingularLocation**(int newLocation, string strData = "", int Address = 0)

//Returns pSingularLocation from memorycell

//Returns 0 , if memorycell doesn't exist

//can send string data or tokenized string data for address

//currently right side cells only, don't see a need for left side yet

int **GetpSingularLocation**(string strData = "", int Address = 0)

int **Tokenize** (string str\_Data,bool ForceLowerCase = true)