pa3.cpp

#include "pa3.hpp"

//Initialize the List Objects

List list**;**

List keywords**;**

List identifiers**;**

List constants**;**

List operators**;**

List delimiters**;**

List errors**;**

List**::**List**()**    **{**

    start **=** **NULL;**

    now **=** **NULL;**

    temp **=** **NULL;**

**}**

int main**(**int argc**,** char**\*\*** argv**)** **{**

    //If-Else Statement to test for valid command-line input

    string arg**;**

    string txt**;**

    int length**;**

**if(**argv**[**1**]** **!=** **NULL)** **{**

      arg **=** argv**[**1**];**

**if(**argv**[**2**]** **!=** **NULL)** **{**

            cout **<<** "Error, too many input types given." **<<** endl**;**

**return** 0**;**

**}**

**else**    **{**

            length **=** arg**.**length**();**

**for(**int i **=** length**-**4**;** i **<** length**;** i**++)** **{**

                txt **+=** argv**[**1**][**i**];**

**}**

**if(**txt**.**compare**(**".txt"**))** **{**

                cout **<<** "Error, no valid .txt ending." **<<** endl**;**

**return** 0**;**

**}**

**}**

**}**

**else**   **{**

        cout **<<** "Error, text file given." **<<** endl**;**

**return** 0**;**

**}**

    //If-Else Statement to test if a text file exists, and then runs the program

    ifstream inFile**(**argv**[**1**]);**

**if(**inFile**.**good**())**   **{**

        string line**;**

**while(!**inFile**.**eof**())**   **{**

             inFile **>>** line**;**

             list**.**RemoveCharacters**(**line**);**

**}**

      inFile**.**close**();**

**}**

**else**    **{**

        cout **<<** "Error, non-existent file." **<<** endl**;**

**return** 0**;**

**}**

    cout **<<** "The maximum depth of the nested loop(s) is " **<<** list**.**CheckDepth**()** **<<** endl **<<** endl**;**

    list**.**PushBack**();**

    string keyword **=** keywords**.**PrintList**();**

    cout **<<** "Keywords: " **<<** keyword **<<** endl**;**

    string identifier **=** identifiers**.**PrintList**();**

    cout **<<** "Indentifiers: " **<<** identifier **<<** endl**;**

    string constant **=** constants**.**PrintList**();**

    cout **<<** "Constants: " **<<** constant **<<** endl**;**

    string ops **=** operators**.**PrintList**();**

    cout **<<** "Operators: " **<<** ops **<<** endl**;**

    string delimiter **=** delimiters**.**PrintList**();**

    cout **<<** "Delimiters: " **<<** delimiter **<<** endl**;**

    string error **=** errors**.**PrintList**();**

    cout **<<** "Syntax Error(s): " **<<** error **<<** endl **<<** endl**;**

**return** 0**;**

**}**

//This function tests how many nested loops exist

int List**::**CheckDepth**()** **{**

    now **=** start**;**

    int count **=** 0**;**

    int max **=** 0**;**

**if(**start **!=** **NULL)**   **{**

**while(**now**->**next **!=** **NULL)** **{**

**if(**CheckUpper**(**now**->**data**))**   **{**

                count**++;**

                now **=** now**->**next**;**

**while(!(**CheckUpper**(**now**->**data**)** **||** now**->**next **==** **NULL))**   **{**

**if(**CheckNumber**(**now**->**data**)** **&&** CheckDelimiter**(**now**->**next**->**data**))**   **{**

                        count **-=** 3**;**

**}**

                    now **=** now**->**next**;**

**}**

**}**

**else**    **{**

**while(!(**CheckUpper**(**now**->**data**)** **||** now**->**next **==** **NULL))**   **{**

                    now **=** now**->**next**;**

**}**

**}**

            //Find max number

**if(**count **>** max**)** **{**

                max **=** count**;**

**}**

**}**

**}**

**return** max**;**

**}**

//This function tests whether 'check' is a delimiter

bool List**::**CheckDelimiter**(**string check**)** **{**

**if(!(**check**).**compare**(**","**)** **||** **!(**check**).**compare**(**";"**))** **{**

**return** **true;**

**}**

**return** **false;**

**}**

//This function returns whether "check" is currently in the list

bool List**::**CheckExist**(**string check**)** **{**

    bool exist **=** **false;**

    now **=** start**;**

**while(**now **!=** **NULL)** **{**

        //Checking if the data referenced in the node is equivalent to the 'check' program

**if((**now**->**data**).**compare**(**check**)** **==** 0**)** **{**

            exist **=** **true;**

**}**

        //Traversing through the array

        now **=** now**->**next**;**

**}**

    now **=** start**;**

**return** exist**;** //returns true if check is in the Linked List

**}**

//This function tests whether 'check' is a keyword

bool List**::**CheckKeyword**(**string check**)**   **{**

**if(!(**now**->**data**).**compare**(**"BEGIN"**)** **||**

**!(**now**->**data**).**compare**(**"END"**)** **||**

**!(**now**->**data**).**compare**(**"FOR"**))** **{**

**return** **true;**

**}**

**return** **false;**

**}**

//This function tests whether 'space' is a character string with characters from 'a-z'

bool List**::**CheckLetter**(**string check**)**    **{**

    char c **=** '0'**;**

    int num **=** 0**;**

    int length **=** check**.**length**();**

    bool n **=** **false;**

**for(**int i **=** 0**;** i **<** length**;** i**++)** **{**

        c **=** check**[**i**];**

        num **=** **(**int**)**c**;**

**if(**num **<** 123 **&&** num **>** 96**)**   **{**

            n **=** **true;**

**}**

**else{**

**return** **false;**

**}**

**}**

**return** n**;**

**}**

//This function tests whether 'space' is a numerical value

bool List**::**CheckNumber**(**string check**)**    **{**

    char c **=** '0'**;**

    int num **=** 0**;**

    int length **=** check**.**length**();**

    bool n **=** **false;**

**for(**int i **=** 0**;** i **<** length**;** i**++)** **{**

        c **=** check**[**i**];**

        num **=** **(**int**)**c**;**

**if(**num **<** 59 **&&** num **>** 47**)**   **{**

            n **=** **true;**

**}**

**else{**

**return** **false;**

**}**

**}**

**return** n**;**

**}**

//This function tests whether 'space' is an operator

bool List**::**CheckOperator**(**string check**)** **{**

    char c **=** '0'**;**

    int num **=** 0**;**

    int length **=** check**.**length**();**

    bool n **=** **false;**

**if(**length **==** 1**)** **{**

        c **=** check**[**0**];**

        num **=** **(**int**)**c**;**

**if(**num **==** 42 **||** num **==** 43 **||** num **==** 45 **||** num **==** 47 **||** num **==** 61**)** **{**

            n **=** **true;**

**}**

**else**    **{**

**return** **false;**

**}**

**}**

**else** **if(**length **==** 2**)**    **{**

**if(**check**[**0**]==**check**[**1**])**   **{**

**for(**int i **=** 0**;** i **<** length**;** i**++)** **{**

                c **=** check**[**i**];**

                num **=** **(**int**)**c**;**

**if(**num **==** 61 **||** num **==** 43 **||** num **==** 45**)**   **{**

                    n **=** **true;**

**}**

**else{**

**return** **false;**

**}**

**}**

**}**

**else**    **{**

**return** **false;**

**}**

**}**

**return** n**;**

**}**

//This function tests whether 'space' is all Uppercase

bool List**::**CheckUpper**(**string check**)** **{**

    char c **=** '0'**;**

    int num **=** 0**;**

    int length **=** check**.**length**();**

    bool n **=** **false;**

**for(**int i **=** 0**;** i **<** length**;** i**++)** **{**

        c **=** check**[**i**];**

        num **=** **(**int**)**c**;**

**if(**num **<** 91 **&&** num **>** 64**)**   **{**

            n **=** **true;**

**}**

**else{**

**return** **false;**

**}**

**}**

**return** n**;**

**}**

//This function pops the information from the stack

void List**::**Pop**()**    **{**

    //follows (Last In First Out) stack structure, so the last inputted item is the first to exit

    start **=** start**->**next**;**

**}**

//This function prints a list

string List**::**PrintList**()**   **{**

    string output **=** ""**;**

    //Loops through the Linked List

    now **=** start**;**

**while(**now **!=** **NULL)** **{**

        output **+=** now**->**data **+** " "**;**

        now **=** now**->**next**;**

**}**

**return** output**;**

**}**

//This method inserts the 'space' at the top of the stack

void List**::**Push**(**string space**)** **{**

    //Initialize node pointer, pointer to the next node, and the node's data

    node**\*** nod **=** **new** node**;**

    nod**->**next **=** **NULL;** //The Linked List should always end in NULL

    nod**->**data **=** space**;**

    temp **=** start**;**

    start **=** nod**;**

    nod**->**next **=** temp**;**

**}**

//This method takes the main stack and separates it into smaller more specific lists

void List**::**PushBack**()** **{**

    now **=** start**;**

**while(**now **!=** **NULL)** **{**

**if(**CheckKeyword**(**now**->**data**))** **{**

**if(!**keywords**.**CheckExist**(**now**->**data**))** **{**

                keywords**.**Push**(**now**->**data**);**

**}**

**}**

**else** **if(**list**.**CheckLetter**(**now**->**data**))**   **{**

**if(!**identifiers**.**CheckExist**(**now**->**data**))** **{**

                identifiers**.**Push**(**now**->**data**);**

**}**

**}**

**else** **if(**list**.**CheckNumber**(**now**->**data**))**   **{**

**if(!**constants**.**CheckExist**(**now**->**data**))** **{**

                constants**.**Push**(**now**->**data**);**

**}**

**}**

**else** **if(**CheckDelimiter**(**now**->**data**))**   **{**

**if(!**delimiters**.**CheckExist**(**now**->**data**))** **{**

                delimiters**.**Push**(**now**->**data**);**

**}**

**}**

**else** **if(**list**.**CheckOperator**(**now**->**data**))**   **{**

**if(!**operators**.**CheckExist**(**now**->**data**))** **{**

                operators**.**Push**(**now**->**data**);**

**}**

**}**

**else**    **{**

**if(!**errors**.**CheckExist**(**now**->**data**))** **{**

                errors**.**Push**(**now**->**data**);**

**}**

**}**

        //removes item from list

        list**.**Pop**();**

        now **=** start**;**

**}**

**}**

//This function deletes a user-specified program from the list

void List**::**RemoveCharacters**(**string line**)**   **{**

    int length **=** line**.**length**();**

    int i **=** 0**;**

**for(**i **=** 0**;** i **<** length**;** i**++)**   **{**

**if(**line**.**at**(**i**)** **==** **(**')'**)** **||** line**.**at**(**i**)** **==** **(**'('**))** **{**

            line **=** line**.**erase**(**i**,** 1**);** //erase parentheses

            i**--;**

**}**

**else** **if(**line**.**at**(**i**)** **==** **(**';'**)** **||** line**.**at**(**i**)** **==** **(**','**)** **||** line**.**at**(**i**)** **==** **(**'='**))** **{**

            string newLine **=** ""**;**

**for(**int j **=** 0**;** j **<** i**;** j**++)** **{**

                newLine **+=** line**.**at**(**j**);**

**}**

            list**.**Push**(**newLine**);** //Push string before the above characters, so they aren't connected

            string c **=** ""**;**

            c **+=** line**.**at**(**i**);**

            list**.**Push**(**c**);** //Push one of the characters from above

            newLine **=** ""**;**

**for(**int k **=** i**+**1**;** k **<** length**;** k**++)** **{**

                newLine **+=** line**.**at**(**k**);**

**}**

            line **=** newLine**;**

            i**++;**

**}**

        length **=** line**.**length**();** //Reassigns line length

**}**

**if(**line **!=** ""**)** **{**

        list**.**Push**(**line**);** //Pushes the remaining part of the line

**}**

**}**

pa3.hpp

#ifndef pa3\_hpp

#define pa3\_hpp

#include <stdio.h>

#include <iostream>

#include <fstream>

#include <string>

#include <cstring>

#include <cstdlib>

#include <cstddef>

**using** **namespace** std**;**

/\*

\* This class deals with initializing the variables dealing with traversing, replacing, and creating nodes.

\*/

class node **{**

public**:**

    string data**;**

    node**\*** next**;**

**};**

/\*

\* This class deals with initializing the nodes in the Linked List and the functions that alter the data within it.

\*/

class List **{**

private**:**

    node**\*** start**;**

    node**\*** now**;**

    node**\*** temp**;**

public**:**

    List**();**

    //This function tests how many nested loops exist

    int CheckDepth**();**

    //This function tests whether 'check' is a delimiter

    bool CheckDelimiter**(**string check**);**

    //This function returns whether "check" is currently in the list

    bool CheckExist**(**string check**);**

    //This function tests whether 'check' is a keyword

    bool CheckKeyword**(**string check**);**

    //This function tests whether 'space' is a character string with characters from 'a-z'

    bool CheckLetter**(**string space**);**

    //This function tests whether 'space' is a numerical value

    bool CheckNumber**(**string space**);**

    //This function tests whether 'space' is an operator

    bool CheckOperator**(**string check**);**

    //This function tests whether 'space' is all Uppercase

    bool CheckUpper**(**string space**);**

    //This function pops the information from the stack

    void Pop**();**

    //This function prints a list

    string PrintList**();**

    //This method inserts the 'space' at the top of the stack

    void Push**(**string space**);**

    //This method takes the main stack and separates it into smaller more specific lists

    void PushBack**();**

    //This function deletes a user-specified program from the list

    void RemoveCharacters**(**string line**);**

**};**

#endif /\* pa3\_hpp \*/