Author: Michael LaPan
Assignment: program 3
class: adv C++ CST 280

Table of contents:

headers:

gradDataClass.h

StringTokenizer.h

CPP's:

gradDataClass.cpp

Main.cpp

StringTokenizer.cpp

Pictures of output

HEADERS:
gradDataClass.h

```
#ifndef GRAD_DATA_CLASS_H
#define GRAD_DATA_CLASS_H
```

```
#include<string>
using namespace std;
/*******************************
*class gradDataClass deff      *
*this class hold information   *
*about a diploma               *
*******************************/
class gradDataClass
{
private:
        string college;
        string LName;
        string FName;
        string middleA;
        string degree;
        string date;
        string month;
        string day;
        string year;
public:
        //gets, returns the vairable the user needs
        string getCollege();
        string getLName();
        string getFName();
        string getMiddleA();
        string getDegree();
        string getDate();
        string getMonth();
        string getDay();
        string getYear();

        //sets, sets the varible to a user specified varible.
        void setCollege(string college);
        void setLName(string LName);
        void setFName(string FName);
        void setMiddleA(string middleA);
        void setDegree(string degree);
        void setDate(string date);
        void setMonth(string month);
        void setDay(string day);
        void setYear(string year);
};

#endif
```

StringTokenizer.h

```
#ifndef STRING_TOKENIZER_H
#define STRING_TOKENIZER_H
// This class is designed to manage a line of "tokenized" data
// (comma-separated files).  The primary class data member is a
// string object.  It includes the ability to reset the delimiter
// from a comma to a different character.

// Strategy: programmer must "reset" the token "pointer" to begin.
// Then, by calling one of the "get" function, they can return the
// next data token separated by the current and next comma (or endline).

// Precondition:  Class user has knowledge of number and types
// of tokens in data.

#include<string>
using namespace std;

class StringTokenizer
{
  private:
    string theLine;       // String containing raw data
    int currIndex;        // Index of current token in focus
    string tempString;    // Temporary holding string for token work
    char delimiterChar;   // Character that is delimiter (default: comma)

    void get_a_Token();   // Utility function to extract string tokens

  public:
    // Default constructor - accepts comma-delimited string to parse
    StringTokenizer(string);

    // Parameterized constructor - accepts a character other than a comma
    // as delimiter.
    // Precondition: separator is a single character (non-space)
    StringTokenizer(char);
    // Parameterized constructor - accepts an initial string to parse and
    // a character other than a comma as delimiter.
    // Precondition: separator is a single character (non-space)
    StringTokenizer(string, char);
    // "Set" and "Get" functhions
    void setLine(string);
    string getLine();
    // Iterator to reset token sequence to first token
    void reset();
    // Returns total number of tokens in line (number of commas + 1)
    int numberTokens();
    // Observes if pointer to current token is at end of string
    bool atEnd();
    // Functions to return data tokens based on type
    string getStringToken(); // Returns a string token being referenced
    char getCharToken();    // Returns an character value being referenced
    double getDoubleToken(); // Returns a double value being referenced
    int getIntToken();      // Returns an integer value being referenced
```

```
};

#endif
```

CPP'S:

gradDataClass.cpp

```
#include "gradDataClass.h"

#include<string>
using namespace std;


	//gets, returns the vairable the user needs
	string gradDataClass::getCollege()
	{
		return college;
	}
	string gradDataClass::getLName()
	{
		return LName;
	}
	string gradDataClass::getFName()
	{
		return FName;
	}
	string gradDataClass::getMiddleA()
	{
		return middleA;
	}
	string gradDataClass::getDegree()
	{
		return degree;
	}
	string gradDataClass::getDate()
	{
		return date;
	}
	string gradDataClass::getMonth()
	{
		return month;
	}
	string gradDataClass::getDay()
	{
		return day;
	}
	string gradDataClass::getYear()
	{
		return year;
	}
	//sets, sets the varible to a user specified varible.
	void gradDataClass::setCollege(string collegeTemp)
	{
```

```
		college = collegeTemp;
	}
	void gradDataClass::setLName(string LNameTemp)
	{
		LName = LNameTemp;
	}
	void gradDataClass::setFName(string FNameTemp)
	{
		FName = FNameTemp;
	}
	void gradDataClass::setMiddleA(string middleATemp)
	{
		middleA = middleATemp;
	}
	void gradDataClass::setDegree(string degreeTemp)
	{
		degree = degreeTemp;
	}
	void gradDataClass::setDate(string dateTemp)
	{
		date = dateTemp;
	}
	void gradDataClass::setMonth(string monthTemp)
	{
		month = monthTemp;
	}
	void gradDataClass::setDay(string dayTemp)
	{
		day = dayTemp;
	}
	void gradDataClass::setYear(string yearTemp)
	{
		year = yearTemp;
	}
```

MAIN.cpp

```
// This program demonstrates tokenizing comma-delimited data
#include <fstream>
#include <iostream>
#include <string>
#include <sstream>
#include <iomanip>
#include "gradDataClass.h"
#include "StringTokenizer.h"

using namespace std;


void processLine(string inputLine, gradDataClass objArray[], int place);
```

```
void parsedDegree(gradDataClass objArray[], int place);


void parseMonth(gradDataClass objArray[], int place);


void parseDay(gradDataClass objArray[], int place);


void parseYear(gradDataClass objArray[], int place);

void delimDate(char dlim, gradDataClass objArray[], int place);


int stringToInt(string toConvert);

void displayDiploma(gradDataClass objArray[], int place);

//enum for each type of degree
enum degree_code
{
	Associate_of_Arts,
	Associate_of_Science,
	Associate_of_Applied_Science,
	Bachelor_of_Arts,
	Bachelor_of_Science
};

degree_code enumDegree(string const& inString);



const int MAX = 50;

int main()
{
	string inputLine;
	int size = 0;

	gradDataClass objArray[MAX];

	ifstream inFile("gradData.txt");

	if (inFile.fail())	// Test for file existence
	{
		cout << "Problem opening file";
		exit(-1);
	}

	//priming read
```

```
	getline(inFile, inputLine);
	while (!inFile.eof())
	{
		processLine(inputLine, objArray, size);
		getline(inFile, inputLine); //continuation read
		size++;
	}
	//loops through each in size and gets all in for each
	//diploma, and then displays it
	for (int i = 0; i < size; i++)
	{
		delimDate('/', objArray, i);
		parseMonth(objArray, i);
		parseDay(objArray, i);
		parseYear(objArray, i);
		displayDiploma(objArray, i);
		system("pause");
	}

	return 0;
}


/*******************************************************
*		displayDiploma deff                           *
*displayDiploma take an array of objects and the place  *
*you want to access in the array. it then outputs       *
*a formatted diploma.. had some trouble with this       *
* so a few dont display right.                          *
*******************************************************/
void displayDiploma(gradDataClass objArray[], int place)
{
	cout << setw(40) << objArray[place].getCollege() << endl;
	cout << setw(35) << fixed << "has conferred upon" << endl;
	cout << setw(26) << objArray[place].getFName() << " "
	     << objArray[place].getMiddleA()
	     << " " << objArray[place].getLName() << endl;
	cout << setw(33) << "the degree of" << endl;
	cout << setw(28) << objArray[place].getDegree() << endl;
	cout << setw(50) << "with all right and privilages thereto pertaining" << endl;
	cout << setw(18) << "on this " << objArray[place].getDay() << " of "
	     << objArray[place].getMonth() << endl;
	cout << setw(20) << "in the year " << objArray[place].getYear()
	      << endl << endl << endl;
}


/*******************************************************
*                    parseMonth                        *
*parse month takes an array of objects, and the place   *
*in the array you want to be parsed. it then converts   *
*the month in number for into a month in word form      *
*******************************************************/
```

```
void parseMonth(gradDataClass objArray[], int place)
{
	int tempM = stringToInt(objArray[place].getMonth());
	switch (tempM)
	{

	case 1:
		objArray[place].setMonth("January");
		break;
	case 2:
		objArray[place].setMonth("February");
		break;
	case 3:
		objArray[place].setMonth("March");
		break;
	case 4:
		objArray[place].setMonth("April");
		break;
	case 5:
		objArray[place].setMonth("May");
		break;
	case 6:
		objArray[place].setMonth("June");
		break;
	case 7:
		objArray[place].setMonth("July");
		break;
	case 8:
		objArray[place].setMonth("January");
		break;
	case 9:
		objArray[place].setMonth("August");
		break;
	case 10:
		objArray[place].setMonth("September");
		break;
	case 11:
		objArray[place].setMonth("October");
		break;
	case 12:
		objArray[place].setMonth("December");
		break;
	default:
		break;
	}
}
/*******************************************************
*                                parseday              *
*parse day takes an array of objects, and the place     *
*in the array you want to be parsed. it then converts the*
*day in number form into a day in word form             *
*after thinking about it loading all days from a file   *
*would have been better, just not enought time to       *
```

```
*implement                                                    *
*********************************************************/

void parseDay(gradDataClass objArray[], int place)
{
        string lThenTwnty[19] = { "first", "second", "third", "fourth", "fifth", "sixth", "seventh",
                "eighth", "ninth", "tenth", "eleventh", "twelfth", "thirteenth",
                "fourteenth", "fifteenth", "sixteenth", "seventeenth", "eighteenth",
                "nineteenth" };
        string gThenTwnty[4] = { "twentieth", "twenty-", "thirtieth", "thirty-" };
        int tempD = stringToInt(objArray[place].getDay());
        switch (tempD)
        {

        case 1:
                objArray[place].setDay(lThenTwnty[0]);
                break;
        case 2:
                objArray[place].setDay(lThenTwnty[1]);
                break;
        case 3:
                objArray[place].setDay(lThenTwnty[2]);
                break;
        case 4:
                objArray[place].setDay(lThenTwnty[3]);
                break;
        case 5:
                objArray[place].setDay(lThenTwnty[4]);
                break;
        case 6:
                objArray[place].setDay(lThenTwnty[5]);
                break;
        case 7:
                objArray[place].setDay(lThenTwnty[6]);
                break;
        case 8:
                objArray[place].setDay(lThenTwnty[7]);
                break;
        case 9:
                objArray[place].setDay(lThenTwnty[8]);
                break;
        case 10:
                objArray[place].setDay(lThenTwnty[9]);
                break;
        case 11:
                objArray[place].setDay(lThenTwnty[10]);
                break;
        case 12:
                objArray[place].setDay(lThenTwnty[11]);
                break;
        case 13:
                objArray[place].setDay(lThenTwnty[12]);
                break;
```

```
case 14:
        objArray[place].setDay(lThenTwnty[13]);
        break;
case 15:
        objArray[place].setDay(lThenTwnty[14]);
        break;
case 16:
        objArray[place].setDay(lThenTwnty[15]);
        break;
case 17:
        objArray[place].setDay(lThenTwnty[16]);
        break;
case 18:
        objArray[place].setDay(lThenTwnty[17]);
        break;
case 19:
        objArray[place].setDay(lThenTwnty[18]);
        break;
case 20:
        objArray[place].setDay(gThenTwnty[0]);
        break;
case 21:
        objArray[place].setDay(gThenTwnty[1] + lThenTwnty[0]);
        break;
case 22:
        objArray[place].setDay(gThenTwnty[1] + lThenTwnty[1]);
        break;
case 23:
        objArray[place].setDay(gThenTwnty[1] + lThenTwnty[2]);
        break;
case 24:
        objArray[place].setDay(gThenTwnty[1] + lThenTwnty[3]);
        break;
case 25:
        objArray[place].setDay(gThenTwnty[1] + lThenTwnty[4]);
        break;
case 26:
        objArray[place].setDay(gThenTwnty[1] + lThenTwnty[5]);
        break;
case 27:
        objArray[place].setDay(gThenTwnty[1] + lThenTwnty[6]);
        break;
case 28:
        objArray[place].setDay(gThenTwnty[1] + lThenTwnty[7]);
        break;
case 29:
        objArray[place].setDay(gThenTwnty[1] + lThenTwnty[8]);
        break;
case 30:
        objArray[place].setDay(gThenTwnty[2]);
        break;
case 31:
        objArray[place].setDay(gThenTwnty[3] + lThenTwnty[0]);
```

```
		break;
	default:
		break;
	}
}

/*******************************************************
*                                      parseyear       *
*parse year takes an array of objects, and the place    *
*in the array you want to be parsed. it then converts   *
*the year in number form into a year in word form       *
*after thinking about it loading all days from a file   *
*would have been better, just not enought time to       *
*implement                                              *
*******************************************************/
void parseYear(gradDataClass objArray[], int place)
{
	string lThenTwnty[19] = { "one", "two", "three", "four", "five", "six", "seven",
		"eight", "nine", " ten", " eleven", " twelve", " thirteen", " fourteen", " fifteen",
		" sixteen", " seventeen", " eighteen", "nineteen" };
	string gThenTwnty[8] = { "two-thousand", "twenty", "thirty", "forty", "fifty", };
	int tempY = stringToInt(objArray[place].getYear());

	switch (tempY)
	{

	case 1:
		objArray[place].setYear(gThenTwnty[0] + lThenTwnty[0]);
		break;
	case 2:
		objArray[place].setYear(gThenTwnty[0] + lThenTwnty[1]);
		break;
	case 3:
		objArray[place].setYear(gThenTwnty[0] + lThenTwnty[2]);
		break;
	case 4:
		objArray[place].setYear(gThenTwnty[0] + lThenTwnty[3]);
		break;
	case 5:
		objArray[place].setYear(gThenTwnty[0] + lThenTwnty[4]);
		break;
	case 6:
		objArray[place].setYear(gThenTwnty[0] + lThenTwnty[5]);
		break;
	case 7:
		objArray[place].setYear(gThenTwnty[0] + lThenTwnty[6]);
		break;
	case 8:
		objArray[place].setYear(gThenTwnty[0] + lThenTwnty[7]);
		break;
	case 9:
		objArray[place].setYear(gThenTwnty[0] + lThenTwnty[8]);
		break;
```

```
case 10:
        objArray[place].setYear(gThenTwnty[0] + lThenTwnty[9]);
        break;
case 11:
        objArray[place].setYear(gThenTwnty[0] + lThenTwnty[10]);
        break;
case 12:
        objArray[place].setYear(gThenTwnty[0] + lThenTwnty[11]);
        break;
case 13:
        objArray[place].setYear(gThenTwnty[0] + lThenTwnty[12]);
        break;
case 14:
        objArray[place].setYear(gThenTwnty[0] + lThenTwnty[13]);
        break;
case 15:
        objArray[place].setYear(gThenTwnty[0] + lThenTwnty[14]);
        break;
case 16:
        objArray[place].setYear(gThenTwnty[0] + lThenTwnty[15]);
        break;
case 17:
        objArray[place].setYear(gThenTwnty[0] + lThenTwnty[16]);
        break;
case 18:
        objArray[place].setYear(gThenTwnty[0] + lThenTwnty[17]);
        break;
case 19:
        objArray[place].setYear(gThenTwnty[0] + lThenTwnty[18]);
        break;
case 20:
        objArray[place].setYear(gThenTwnty[0] + gThenTwnty[0]);
        break;
case 21:
        objArray[place].setYear(gThenTwnty[0] + gThenTwnty[1] + lThenTwnty[0]);
        break;
case 22:
        objArray[place].setYear(gThenTwnty[0] + gThenTwnty[1] + lThenTwnty[1]);
        break;
case 23:
        objArray[place].setYear(gThenTwnty[0] + gThenTwnty[1] + lThenTwnty[2]);
        break;
case 24:
        objArray[place].setYear(gThenTwnty[0] + gThenTwnty[1] + lThenTwnty[3]);
        break;
case 25:
        objArray[place].setYear(gThenTwnty[0] + gThenTwnty[1] + lThenTwnty[4]);
        break;
case 26:
        objArray[place].setYear(gThenTwnty[0] + gThenTwnty[1] + lThenTwnty[5]);
        break;
case 27:
        objArray[place].setYear(gThenTwnty[0] + gThenTwnty[1] + lThenTwnty[6]);
```

```
            break;
case 28:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[1] + lThenTwnty[7]);
            break;
case 29:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[1] + lThenTwnty[8]);
            break;
case 30:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[2]);
            break;
case 31:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[2] + lThenTwnty[0]);
            break;
case 32:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[2] + lThenTwnty[1]);
            break;
case 33:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[2] + lThenTwnty[2]);
            break;
case 34:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[2] + lThenTwnty[3]);
            break;
case 35:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[2] + lThenTwnty[4]);
            break;
case 36:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[2] + lThenTwnty[5]);
            break;
case 37:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[2] + lThenTwnty[6]);
            break;
case 38:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[2] + lThenTwnty[7]);
            break;
case 39:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[2] + lThenTwnty[8]);
            break;
case 40:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[3]);
            break;
case 41:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[3] + lThenTwnty[0]);
            break;
case 42:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[3] + lThenTwnty[1]);
            break;
case 43:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[3] + lThenTwnty[2]);
            break;
case 44:
            objArray[place].setYear(gThenTwnty[0] + gThenTwnty[3] + lThenTwnty[3]);
            break;
case 45:
```

```
                objArray[place].setYear(gThenTwnty[0] + gThenTwnty[3] + lThenTwnty[4]);
                break;
        case 46:
                objArray[place].setYear(gThenTwnty[0] + gThenTwnty[3] + lThenTwnty[5]);
                break;
        case 47:
                objArray[place].setYear(gThenTwnty[0] + gThenTwnty[3] + lThenTwnty[6]);
                break;
        case 48:
                objArray[place].setYear(gThenTwnty[0] + gThenTwnty[3] + lThenTwnty[7]);
                break;
        case 49:
                objArray[place].setYear(gThenTwnty[0] + gThenTwnty[3] + lThenTwnty[8]);
                break;
        case 50:
                objArray[place].setYear(gThenTwnty[0] + gThenTwnty[4]);
                break;

        default:
                break;
        }
}

/*******************************************************
*                                    parseDegree                                *
*parse degree takes an array of objects and the place                    *
*in the array you want to be parsed. it then convers                     *
*the acrocrymn to the full name                                                  *
*******************************************************/

void parsedDegree(gradDataClass objArray[], int place)
{

        string temp = objArray[place].getDegree();
        switch (enumDegree(temp)){
        case Associate_of_Arts:
                objArray[place].setDegree("Associate of Arts");
                break;
        case Associate_of_Science:
                objArray[place].setDegree("Associate of Science");

                break;
        case Associate_of_Applied_Science:
                objArray[place].setDegree("Associate of Applied Science");

                break;
        case Bachelor_of_Arts:
                objArray[place].setDegree("Bachelor of Arts");

                break;
        case Bachelor_of_Science:
                objArray[place].setDegree("Bachelor of Science");
                break;
```

```
	default:
		break;
	}
}

/*******************************************************
*                                  enumDegree          *
*converts the input string (a degree) into the assoatied *
*enum to be put into the switch                         *
*******************************************************/
degree_code enumDegree(string const& inString)
{
	if (inString == "A.A") return Associate_of_Arts;
	if (inString == "A.S") return Associate_of_Science;
	if (inString == "A.A.S") return Associate_of_Applied_Science;
	if (inString == "B.A") return Bachelor_of_Arts;
	if (inString == "B.S") return Bachelor_of_Science;
}

/*******************************************************
*                                  stringToInt         *
*changes a string to an integer be useing a string stream *
*******************************************************/
int stringToInt(string toConvert)
{
	int sTi = 0;
	stringstream tempsstring;
	tempsstring << toConvert;
	tempsstring >> sTi;
	return sTi;
}

/*******************************************************
*                                  delimDate           *
*takes the delimiter to look for, the array of objects  *
* and the place to look at. it then splits out the month *
*, day and, year fomr a date                            *
*******************************************************/
void delimDate(char dlim, gradDataClass objArray[], int place)
{
	string stringToDelim;
	StringTokenizer tokenizer(dlim);
	tokenizer.setLine(objArray[place].getDate());

	stringToDelim = tokenizer.getStringToken();
	objArray[place].setMonth(stringToDelim);
	stringToDelim = tokenizer.getStringToken();
	objArray[place].setDay(stringToDelim);
	stringToDelim = tokenizer.getStringToken();
	objArray[place].setYear(stringToDelim);
}
```

```
/*******************************************************
*                                processLine                                  *
*this function parses a coma seperated line, and places                      *
*each item in to the class. did this the long way by not                     *
*useing the stringTokenizer class provided in example,                       *
*was haveing some trouble getting the class to work                          *
*untill today                                                                *
*******************************************************/

void processLine(string inputLine, gradDataClass objArray[], int place)
{

        string college, lastName, firstName, middleInital, degree, date;

        // Variables to mark string positions
        int  len, start, nextComma;

        // Get first string
        nextComma = inputLine.find(',');          // Mark 1st comma location
        len = nextComma - 0;                    // Calculate length of substring
        college = inputLine.substr(0, len);       // Get zip code as string
        start = nextComma + 1;                  // Mark position after this comma

        // Get second
        nextComma = inputLine.find(',', start);    // Mark 2nd comma location
        len = nextComma - start;                // Calculate length of substring
        lastName = inputLine.substr(start, len);  // Get numerical info as string
        start = nextComma + 1;                  // Mark position after this comma

        // Get third string
        nextComma = inputLine.find(',', start);    // Mark next comma location
        len = nextComma - start;                // Calculate length of substring
        firstName = inputLine.substr(start, len);    // Get string
        start = nextComma + 1;                  // Mark position after this comma

        // Get fourth string
        nextComma = inputLine.find(',', start);    // Mark next comma location
        len = nextComma - start;                // Calculate length of substring
        middleInital = inputLine.substr(start, len);    // Get string
        start = nextComma + 1;                  // Mark position after this comma

        // Get fifth string
        nextComma = inputLine.find(',', start);    // Mark next comma location
        len = nextComma - start;                // Calculate length of substring
        degree = inputLine.substr(start, len);    // Get string
        start = nextComma + 1;                  // Mark position after this comma

        // Get sixth string
        nextComma = inputLine.find(',', start);    // Mark next comma location
        len = nextComma - start;                // Calculate length of substring
        date = inputLine.substr(start, len);    // Get string
        start = nextComma + 1;                  // Mark position after this comma
```

```
            objArray[place].setCollege(college);
            objArray[place].setDate(date);
            objArray[place].setDegree(degree);
            objArray[place].setFName(firstName);
            objArray[place].setLName(lastName);
            objArray[place].setMiddleA(middleInital);
}
```

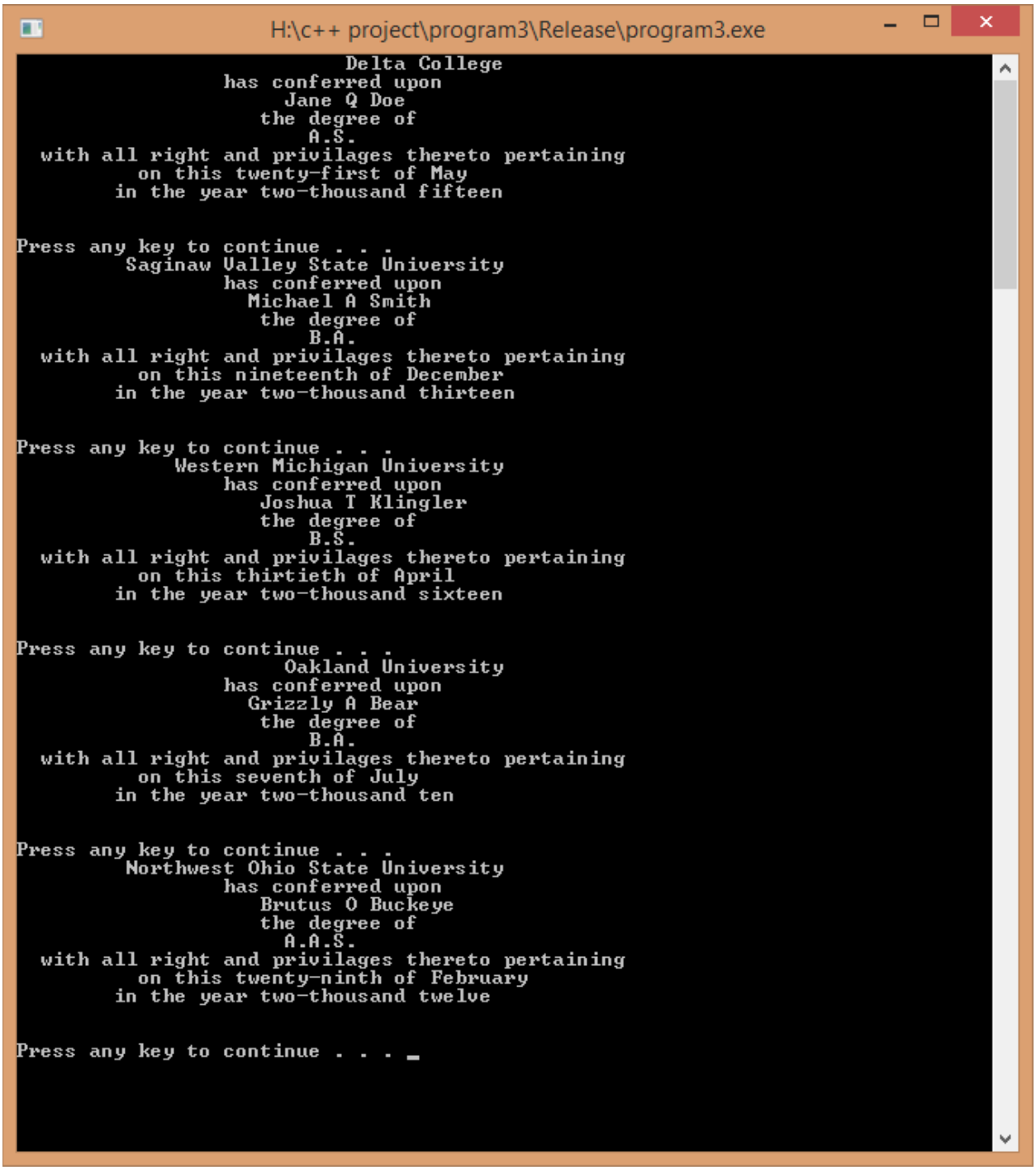
STRINGTOKENIZER.cpp:

```
// Class to manage "tokenized" data (comma-separated files)

#include<iostream>
#include<string>
using namespace std;

#include "StringTokenizer.h"

// Constructor
// Receive line of comma-delimited data and set index of current
// token to zero
StringTokenizer::StringTokenizer(string line)
{
  theLine = line;
  currIndex = 0;
  delimiterChar = ',';    // Default delimiter is comma
}

// Parameterized constructor - accepts a character other than a comma
// as delimiter.
// Precondition: separator is a single character (non-space)
// Postcondtion: data string NOT initialized
StringTokenizer::StringTokenizer(char newDelimiter)
{
  currIndex = 0;
  delimiterChar = newDelimiter;    // Set delimiter to programmer choice
}

// Parameterized constructor - accepts an initial string to parse and
// a character other than a comma as delimiter.
// Precondition: separator is a single character (non-space)
StringTokenizer::StringTokenizer(string line, char newDelimiter)
{
  theLine = line;
  currIndex = 0;
  delimiterChar = newDelimiter;   // Set delimiter to programmer choice
}

// "Set" and "Get" functhions
void StringTokenizer::setLine(string line)
```

```
{
  theLine = line;
}

string StringTokenizer:: getLine()
{
  return theLine;
}

// Reset index of current token to zero
void StringTokenizer::reset()
{
  currIndex = 0;
}

// Return number of tokens in current string
int StringTokenizer::numberTokens()
{
  int lineLength = theLine.length();
  int count = 0;

  for (int i = 0; i < lineLength; i++)
    if (theLine.at(i) == delimiterChar)
      count++;

  return count + 1;    // Return number of tokens as number of commas + 1
}

// To observe if token pointer is at end of string
bool StringTokenizer::atEnd()
{
  if (currIndex == theLine.length())
    return true;
  else
    return false;
}

// Return next token as a string object
// Precondition: Token pointer is not at end of string
string StringTokenizer::getStringToken()
{
  get_a_Token();
  return tempString;
}

// Return next token as a character object
// Precondition: Token pointer is not at end of string
char StringTokenizer::getCharToken()
{
  char outChar;

  get_a_Token();
  outChar = tempString[0];              // Extract one char from string
```

```
  return outChar;
}

// Return next token as a double value
// Precondition: Token pointer is not at end of string
double StringTokenizer::getDoubleToken()
{
  double outValue;

  get_a_Token();

  outValue = atof(tempString.data());     // Convert to number
  return outValue;
}

// Returns an integer value being referenced
int StringTokenizer::getIntToken()
{
  int outInt;

  get_a_Token();

  outInt = atoi(tempString.data());       // Convert to number
  return outInt;
}

// This utility function retrieves the next token and returns it to
// be correctly typed before the final return from the object function call
void StringTokenizer::get_a_Token()
{
  int newIndex = theLine.find(delimiterChar,currIndex);

  if (newIndex >= 0)   // If not last token
  {
    tempString = theLine.substr(currIndex,newIndex - currIndex);
    currIndex = newIndex + 1;       // Advance to position after next comma
  }
  else    // If last token
  {
    tempString =  theLine.substr(currIndex,theLine.length() - currIndex);
    currIndex = theLine.length();    // Set current index to end
  }
}
```

Pictures of output:


H:\c++ project\program3\Release\program3.exe
Delta College
has conferred upon
Jane Q Doe
the degree of
A.S.
with all right and privilages thereto pertaining
on this twenty-first of May
in the year two-thousand fifteen
Press any key to continue . . .
Saginaw Valley State University
has conferred upon
Michael A Smith
the degree of
B.A.
with all right and privilages thereto pertaining
on this nineteenth of December
in the year two-thousand thirteen
Press any key to continue . . .
Western Michigan University
has conferred upon
Joshua T Klingler
the degree of
B.S.
with all right and privilages thereto pertaining
on this thirtieth of April
in the year two-thousand sixteen
Press any key to continue . . .
Oakland University
has conferred upon
Grizzly A Bear
the degree of
B.A.
with all right and privilages thereto pertaining
on this seventh of July
in the year two-thousand ten
Press any key to continue . . .
Northwest Ohio State University
has conferred upon
Brutus O Buckeye
the degree of
A.A.S.
with all right and privilages thereto pertaining
on this twenty-ninth of February
in the year two-thousand twelve
Press any key to continue . . .