//AUTHOR: Dan Crisp

//DATE: 27OCT2016

//FILE: parseSet.cpp

/\* Readin and writing data from a text file. \*/

/\* Works well for AllAlarmChannels.txt as input, but doens't handle saveset.txt header very well \*/

/\* Skip spaces. This would only work for spaces at beginning of line. strtok and delimiter below do a much better job.

Removed from get\_stuff()

while(fin.peek()== ' '){ //skip spaces

fin.get(character);

cout << "\nFOUND SPACE: '" << character << "'\n";

}\*/

#include<iostream>

#include<fstream>

#include<cstring>

#include<cstdlib>

#include<cmath>

using namespace std;

const int MAX\_CHARS\_PER\_LINE = 100;

const int MAX\_LINES\_PER\_FILE = 200;

const char\* DELIMITER = " #";

void get\_stuff( istream& , string\* , string\* , string\* , string\* , int );

void put\_stuff( ostream& , string\* , string\* , string\* , string\* , int );

int main()

{

system("cls");

string name\_set[MAX\_LINES\_PER\_FILE] = {""};

string value\_set[MAX\_LINES\_PER\_FILE] = {""};

string name\_read[MAX\_LINES\_PER\_FILE] = {""};

string value\_read[MAX\_LINES\_PER\_FILE] = {""};

char inpfile[40] = {""};

char outfile[40] = {""};

char sortOption;

// Request Filenames From User

cout << "Enter filename for input relative to current location:\n";

cin >> inpfile;

cout << "Now enter filename for output:\n";

cin >> outfile;

//fin.open("AllAlarmChannels.txt");

ifstream fin;

fin.open(inpfile);

if (!fin.good())

return 1;

cout << "\nGETTING INPUT:\n";

for (int i = 0; i<MAX\_LINES\_PER\_FILE; i++) {

cout << i;

get\_stuff(fin, name\_set, value\_set, name\_read, value\_read, i);

cout << "\n";

}

fin.close();

ofstream fout;

fout.open(outfile);

if(!fout.good())

return 1;

cout << "\nWRITING OUTPUT:\n";

for (int i = 0; i < MAX\_LINES\_PER\_FILE; i++) {

put\_stuff(fout, name\_set, value\_set, name\_read, value\_read, i);

}

fout.close();

//cout << "\nWould you like to sort these lines by RD value?\n";

//cin >> sortOption;

return 0;

}

void get\_stuff(istream& fin, string\* name\_set, string\* value\_set, string\* name\_read, string\* value\_read, int i)

{

char character;

char\* thing = new char[MAX\_CHARS\_PER\_LINE];

char \* splitThing;

int count = 0;

// Skip lines starting with '#'

while (fin.peek()=='#') {

fin.getline(thing,MAX\_CHARS\_PER\_LINE);

}

//getline(c-string, numchars )

fin.getline(thing,MAX\_CHARS\_PER\_LINE);

//strtok().. if a token is found, a pointer to the beginning of token, otherwise a null pointer. Calling with NULL instead of c-str causes function to continue scanning where a previous successful call to the function ended.

splitThing = strtok (thing,DELIMITER);

while (splitThing != NULL) {

switch(count) {

case 0:

name\_set[i] = splitThing;

splitThing = strtok (NULL, DELIMITER);

case 1:

value\_set[i]= splitThing;

splitThing = strtok (NULL, DELIMITER);

case 2:

name\_read[i] = splitThing;

splitThing = strtok (NULL, DELIMITER);

case 3:

value\_read[i] = splitThing;

splitThing = strtok (NULL, DELIMITER);

}

count++;

}

}

void put\_stuff(ostream& fout, string\* name\_set, string\* value\_set, string\* name\_read, string\* value\_read, int i)

{

char \*namestr = new char[name\_set[i].length() + 1];

char \*valuestr = new char[value\_set[i].length() + 1];

char \*namestr2 = new char[name\_read[i].length() + 1];

char \*valuestr2 = new char[value\_read[i].length() + 1];

strcpy(namestr, name\_set[i].c\_str());

strcpy(valuestr, value\_set[i].c\_str());

strcpy(namestr2, name\_read[i].c\_str());

strcpy(valuestr2, value\_read[i].c\_str());

// Display output to fout and screen

// can't cout type string\*, had to copy into char first.

fout << i << " (" << namestr << ", " << valuestr << ", " << namestr2 << ", " << valuestr2 << ")\n";

cout << i << " (" << namestr << ", " << valuestr << ", " << namestr2 << ", " << valuestr2 << ")\n";

}