|  |  |
| --- | --- |
| **Author Identification Block** | |
| **Author:** | Mr. Chris Graff |
| **Student ID:** | \*20274911 |
| **E-Mail:** | serathano@gmail.com |
| **Course:** | CMSC 2123 – Discrete Structures |
| **CRN:** | 21599, Autumn, 2012 |
| **Project:** | p02 |
| **Due:** | October 10, 2012 |
| **Account:** | tt009 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Scoring Block** | | | |
| **Component** | **Available** | **Earned** | **Explanation** |
| Compilation |  |  |  |
| Submission Instructions | 4 | 4 |  |
| Author Identification | 2 | 2 |  |
| Modularity | 6 | 6 |  |
| Command Line | 6 | 6 |  |
| Input file | 6 | 6 |  |
| Output file | 6 | 6 |  |
| Execution | 20 | 20 |  |
| **Total** | **50** | **50** |  |

//Author: Chris Graff

//StudentID#: \*20274911

//Email: cgraff@uco.edu

//Course: CMSC 2123-Discrete Structures

//CRN: 12599, Autumn,2012

//Project: p02

//Due: October 10th, 2012

//Account: tt009

#include<iostream>

#include<string.h>

#include<iomanip>

#include<fstream>

#include"List02.h"

using namespace std;

struct CommandLineException

{

CommandLineException (int max, int actual)

{

cout <<endl <<"Too many command line arguements." <<endl;

cout <<"A maximum of " <<max <<" arguements are permitted." <<endl;

cout <<actual <<" arguements were entered." <<endl;

}

};

struct FileException

{

FileException (string filename)

{

cout <<endl <<"File " <<filename <<" could not be opened or doesn't exist" <<endl;

}

};

string Pair(string x, string y)

{

return "(" + x + "," + y + ")";

}

void Cartesianpro(List& L1, List& L2, List& C)

{

for(L1.First();!L1.IsEol();L1.Next())

{

for(L2.First();!L2.IsEol();L2.Next())

{

C.Insert(Pair(L1.Member(), L2.Member()));

}

}

}

void CartesianMgr(istream& i1, istream& i2, ostream& o)

{

List L1(i1);

L1.print(o, "L1");

List L2(i2);

L2.print(o, "L2");

List C;

Cartesianpro(L1, L2, C);

C.print(o, "C");

}

int main (int argc, char\* argv[])

{

try

{

char iFileName1[255], iFileName2[255], oFileName[255];

switch (argc)

{

case 1:

cout <<"Enter the first input file name:";

cin >> iFileName1;

cout <<"Enter the second input file name:";

cin >> iFileName2;

cout <<"Enter the output file name:";

cin >> oFileName;

break;

case 2:

strcpy(iFileName1, argv[1]);

cout <<"Enter the second input file name:";

cin >> iFileName2;

cout <<"Enter the output file name:";

cin >> oFileName;

break;

case 3:

strcpy(iFileName1, argv[1]);

strcpy(iFileName2, argv[2]);

cout <<"Enter the output file name:";

cin >> oFileName;

break;

case 4:

strcpy(iFileName1, argv[1]);

strcpy(iFileName2, argv[2]);

strcpy(oFileName, argv[3]);

// default:

// throw CommandLineException (3, argc-1);

// break;

}

ifstream i1(iFileName1);

if (!i1)

throw FileException(iFileName1);

ifstream i2(iFileName2);

if (!i2)

throw FileException(iFileName2);

ofstream o(oFileName);

if (!o)

throw FileException(oFileName);

CartesianMgr(i1, i2, o);

i1.close();

i2.close();

o.close();

}

catch (...)

{

cout <<"Program terminated." <<endl;

}

}

#ifndef List02\_h

#define List02\_h 1

// -----------------------------------------------------------------------

//Author: Chris Graff

//StudentID#: \*20274911

//Email: cgraff@uco.edu

//Course: CMSC 2123-Discrete Structures

//CRN: 12599, Autumn,2012

//Project: p02

//Due: October 10th, 2012

//Account: tt009

// ----------------------------------------------------------------------

//C++ include files and namespace std.

// -----------------------------------------------------------------------

#include <cstdlib>

#include <cstring>

#include <iostream>

#include <iomanip>

#include <string>

#include <fstream>

using namespace std;

// -----------------------------------------------------------------------

struct ListException

{

ListException(string m)

{

cout << endl;

cout << "I am the List and i am " << m << ".";

cout << endl;

}

};

// -----------------------------------------------------------------------

class List {

int size ; //Number of elements available in the list

int count; //Actual number of elements occupied.

//Index of the next available element

string\* L ; //A pointer to a list of strings.

int cursor ; //Index of the current element

public:

List(int sz =100): size (sz), count(0), cursor (-1)

{L=new string[size];}

List(istream& i,int sz =100): size ( sz ), count(0), cursor ( -1)

{L =new string[ size ]; Scan ( i);}

~ List(){if (L) delete[] L ;}

bool isfull(void){return count>=size;}

void Insert (string v)

{

if(isfull())

throw ListException("full");

L[count++]=v;

}

void Scan (istream& i)

{

for(;;)

{

string v;

i>>v;

if(i.eof()) break;

Insert(v);

}

}

void print (ostream& o,string title )

{

o << endl;

o << title << "={";

for(int a=0;a<count;a++)

{

if(a!=0) o<<",";

o<<L[a];

}

o<<"}";

}

void First (void){ cursor =0;}

void Next (void){if ( cursor <count) cursor ++;}

bool IsEol (void)

{

if (cursor <0|| cursor > count)

throw ListException( "cursor & count");

return cursor >= count;

}

string Member(void)

{

if (cursor <0|| cursor >= count)

throw ListException( "cursor & count");

return L [ cursor ];

}

};

#endif