```
Script started on Sun 04 Nov 2012 11:42:50 AM CST
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ p
wd
/home/georgia/cplusplus
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ C
PP --versi\033[K033[K033[Kion
This is CPP version 1.219 executing under perl v5.12.4 and compiling with:

g++ (Ubuntu/Linaro 4.6.1-9ubuntu3) 4.6.1
Copyright (C) 2011 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.


\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at filecopy.in\033[Kfo
Name: Jakob Hansen
Class: CSC122 - Evening Section
Lab: data=value

Levels attempted:

4 - For just getting the basic instructions down.


Program Description:

A program that will take a text file with a labeled value format that is messy
and formatted strangely, and copy the pertinent, appropriately formatted
data to another file!
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at student.h
#ifndef STUDENT_H_INC
#define STUDENT_H_INC

#include<string>
#include<fstream>

using namespace std;

class student
{
    string name;
    long id;
    double gpa;
    char gender;

public:

    student(void) : name(""), id(0), gpa(0.0), gender('\0') { }

    student(const student & s) : name(s.name), id(s.id),
                                 gpa(s.gpa), gender(s.gender) { }

    student(const string newname, const long newid, const double newgpa,
            const char newgender) : name(""), id(0), gpa(0.0), gender('\0')
    {
        set_name(newname);
        set_id(newid);
        set_gpa(newgpa);
        set_gender(newgender);
    }

    bool print(ofstream & output) const;
    bool read(ifstream & input);

    string get_name(void) const {return name;}
    long get_id(void) const {return id;}
    double get_gpa(void) const {return gpa;}
    char get_gender(void) const {return gender;}

    void set_name(string newname)
    {
        name = newname;
        return;
    }
    void set_id(long newid)
    {
        id = newid;
        return;
    }
    void set_gpa(double newgpa)
    {
        gpa = newgpa;
        return;
    }
    void set_gender(char newgender)
    {
        gender = newgender;
        return;
    }
};

long strtolong(string & str);

double strtodub(string & str);

short char_to_ones(const char & c);


# endif
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at student.cpp
#include "student.h"
#include <string>
#include <cctype>
#include <iostream>
#include <fstream>
#include <climits>

using namespace std;

bool student::print(ofstream & output) const
{
    if (output.is_open() && output.good())
    {
        output << "\nname = " << name
               << "\nID = " << id
               << "\nGPA = " << gpa
               << "\ngender = " << gender << "\n";
    }

    return output.good();
}

bool student::read(ifstream & input)
{
    string line, label, value;
    string::iterator vit;
    string::size_type pos, i;
    bool labels[4] = {false, false, false, false};
    bool endblock;
    streampos filepos;

    while (!input.eof() && !endblock)
    {
        filepos = input.tellg();
```

```
        getline(input, line);

        pos = line.find('=');
        label = string(line, 0, pos);
        value = string(line, pos+1, line.length()-1);
        vit = value.begin();

        for (i=0; i<label.length(); i++)
        {
            label[i] = toupper(label[i]);
        }
        while(isspace(*vit))
        {
            value.erase(vit);
            vit = value.begin();
        }

        if (label.find("NAME") != string::npos)
        {
            if (!labels[0])
            {
                set_name(value);
                labels[0] = true;
            }
            else
            {
                endblock = true;
                input.seekg(filepos);
            }
        }
        else if (label.find("ID") != string::npos)
        {
            if (!labels[1])
            {
                set_id(strtolong(value));
                labels[1] = true;
            }
            else
            {
                endblock = true;
                input.seekg(filepos);
            }
        }
        else if (label.find("GPA") != string::npos)
        {
            if (!labels[2])
            {
                set_gpa(strtodub(value));
                labels[2] = true;
            }
            else
            {
                endblock = true;
                input.seekg(filepos);
            }
        }
        else if (label.find("GENDER") != string::npos)
        {
            if (!labels[3])
            {
                set_gender(value[0]);
                labels[3] = true;
            }
            else
            {
                endblock = true;
                input.seekg(filepos);
            }
        }
        else if (label.find('#') != string::npos)
        {
            // do nothing?
        }

    }


    if (!labels[0])
    {
        value = "";
        set_name(value);
    }
    if (!labels[1])
    {
        set_id(0);
    }
    if (!labels[2])
    {
        set_gpa(0.0);
    }
    if (!labels[3])
    {
        value = "";
        set_gender(value[0]);
    }

    return endblock;
}

long strtolong(string & str)
{
    long answer=0, t;
    bool neg = false;
    string::iterator it = str.begin();

    if (*it == '-' || *it == '+')
    {
        if (*it == '-')
        {
            neg = true;
        }
        str.erase(it);
    }


    for (it = str.end()-1, t=1; it >= str.begin() && t < LONG_MAX; it--,t*=10)
    {
        answer += (char_to_ones(*it) * t);
    }

    if(neg)
    {
        answer = -answer;
    }

    return answer;
}

double strtodub(string & str)
{
    double pre = 0.0, post = 0.0, t = 0.0;
    string::size_type pos;
    string::iterator it = str.begin();
    bool neg = false;

    if (*it == '-' || *it == '+')
    {
        if (*it == '-')
```

```
        {
            neg = true;
        }
        str.erase(it);
    }

    pos = str.find('.');

    for (it = str.end()-1, t=1.0; it > str.begin() + pos; it--, t*=10.0)
    {
        post += (static_cast<double>(char_to_ones(*it)) * t);
    }

    post = post/t;

    for (it = str.begin()+pos-1, t=1.0; it >= str.begin(); it--, t*=10.0)
    {
        pre += (static_cast<double>(char_to_ones(*it)) * t);
    }

    pre += post;

    if(neg)
    {
        pre = -pre;
    }

    return pre;
}

short char_to_ones(const char & c)
{
    short answer;
    switch (c)
    {
    case '0':
        answer = 0;
        break;
    case '1':
        answer = 1;
        break;
    case '2':
        answer = 2;
        break;
    case '3':
        answer = 3;
        break;
    case '4':
        answer = 4;
        break;
    case '5':
        answer = 5;
        break;
    case '6':
        answer = 6;
        break;
    case '7':
        answer = 7;
        break;
    case '8':
        answer = 8;
        break;
    case '9':
        answer = 9;
        break;
    default:
        answer = 0;
    }
    return answer;
}


\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at filecopy.cpp
#include<iostream>
#include<string>
#include<fstream>
#include<vector>
#include<iomanip>
#include "student.h"

using namespace std;

int main(void)
{
    string filename;
    ifstream infile;
    ofstream outfile;
    student singlestu;
    vector<student> stuvec;
    vector<student>::iterator it;


    cout << "\n\n\tWelcome to the file copier!\n"
            "Input file name: ";
    getline(cin, filename);

    infile.open(filename.c_str());
    while (!infile)
    {
        infile.close();
        infile.clear();

        cout << "File Not Found!" << endl;

        cout << "Input file name:  ";
        getline(cin, filename);

        infile.open(filename.c_str());
    }


    infile.peek();
    while ( !infile.eof() )
    {
        singlestu.read(infile);
        stuvec.push_back(singlestu);
        infile.peek();
    }
    infile.close();
    infile.clear();



    cout << "\nFile read successfully.\nEnter the output file name:  ";
    getline(cin, filename);
    outfile.open(filename.c_str());
    while (!outfile)
    {
        outfile.close();
        outfile.clear();
        cout << "Invalid output file name!" << endl;

        cout << "Enter the output file name:  ";
        getline(cin, filename);

        outfile.open(filename.c_str());
    }
```

```
    for (it = stuvec.begin(); it < stuvec.end(); it++)
    {
        it->print(outfile);
    }

    outfile.close();
    outfile.clear();
    cout << '\n';



    return 0;
}
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ C
PP filecopy student
filecopy.cpp***
student.cpp...
student.cpp: In member function â\200\230bool
student::read(std::ifstream&)â\200\231:
student.cpp:45:40: warning: conversion to â\200\230charâ\200\231 from â\200\230intâ
\200\231
may alter its value [-Wconversion]


\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ .
/filecopy.out


       Welcome to the file copier!
Input file name: some file
File Not Found!
Input file name:  input.dat

File read successfully.
Enter the output file name:  /can't write??
Invalid output file name!
Enter the output file name:  output.dat

\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
atinput\033[K033[K033[K033[K033[K033[K input.dat


nAMe = student student
iD =   29358

# this is a comment
GpA =   4.5
gender=male

name = another student
Id =32597
# this is another comment
Gpa =  2.7
gender=  f

name=  third stude
ID= 23539
GPA =3.8
gender =  m
ID=32523
name=Fourth Student
gender=f
GPA=4.2

town=Williamsburg
gender=male
GPA = 3.1
name= Fifth Student
name=YET another
GPA=1.1


ID=29572
name= last one
GPA=2.2
gender=f
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at outpudt\033[K033[K033[Kt.dat

name = student student
ID = 29358
GPA = 4.5
gender = m

name = another student
ID = 32597
GPA = 2.7
gender = f

name = third stude
ID = 23539
GPA = 3.8
gender = m

name = Fourth Student
ID = 32523
GPA = 4.2
gender = f

name = Fifth Student
ID = 0
GPA = 3.1
gender = m

name = YET another
ID = 29572
GPA = 1.1
gender = \000

name = last one
ID = 0
GPA = 2.2
gender = f
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at filecopy.tpq
What does your main look like? Aside from user interfacing code, is it over 10
lines long? (It shouldn't be...)

    There is not much going on there besides prompting and stuff.

How much data does an object read from the file?

    The object reads data from the file until...well...until there is no more
    file.

How do you recognize comments in the data file?

    Actually the whole concept of labeled data format precludes the necessity of
    comment-preceding-characters because any line of data that is not labeled
    with an appropriate label and '=' character are by the nature of the
    class ignored.

How do you recognize labeled data lines in the file?

    By using the "find" function of the string class (or you could do a good ol'
    linear search) on the line of data that's "got". If the '=' (or any other
```

label-signifying-)character is present, it means you should pay attention to what comes before and after it, if not, then you don't really care about what's in that line of data.

How can you tell a line is neither (comment nor labeled data)?

Well if it's not a comment or a line of labeled data then...we don't really care do we? We only care if something is a DATUM. If it's anything else, it's for the humans...

How can you detect that you are done with a block of data?

If we have read in every one of the available valid labels. If we reach a label that we have seen before without having read the others, we assume we have some missing data and proceeed as such.

What happens if you hit the end of the file while in the object's input method? (Hint: There are two possibilities!)

Well, if eof flag were not a terminal condition of the loop in the read method, you might run into an infinite loop if some labels were missing. Or, if it is a terminal condition, then the method just returns whatever data it has (and, hopefully, reasonable "empty" data for missing values)

How do you fill in default values when you run out of data in your block?

with a conditional statement at the end of the read loop, although now that you bring it up, one could probably assign those values at the beginning of the process and overwrite them later if necessary, but... I guess it's sort of a "six of one, half-dozen of the other" type of situation.

How do you split a line into the label part and the value part?

I used the parameterized constructor of the string class to create strings out of parts of other strings but...in general you don't really care about the label beyond the fact of whether or not it is a valid label-but for the data, you want to take whatever is past the label and it's signifying character and trim the whitespace one way or another (I used the delete method of the string class...awkward, maybe?) and then store that in your member variables.

What happens if the label separator is part of a data item? (Maybe there's a student named =) at the school..?)

We only care about the FIRST such character (given the nature of a linear search), so if there's a student with an '=' in his/her name, then well that's just fine!

Does spacing around the label or around the separator matter to your program?

Not if you take care to strip any whitespace around valuable data...

(In your design) Can a single data item take up multiple lines? Can multiple data items be on a single line? Why/Why not?

In my design data cannot take up multiple lines. Any data that overflowed lines would get ignored since it would be taken to have no label.

Can you use a function which translates a string into an integer to help you translate floating point values?

Yes, you can translate the parts of a number that lie to the left and to the right of the decimal point as integers, and then just add them?! cool!

How would do you translate a string into bool data? char data? string data?

To translate string to bool would require a helper bool and a conditional statement, string to char * could just use the handy c_str() function, string to string may use the assignment operator!

Why don't the handy and input libraries associated with the string library and its test program have .C files?

Because all the functions are INLINED!
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ exit
exit

Script done on Sun 04 Nov 2012 11:44:15 AM CST