```
Script started on 2022-12-08 20:53:30-06:00 [TERM="xterm" TTY="/dev/pts/12" COLUMN
j_pec2@ares:~/JPMainDir/CSC122/Port3/frameProject$ pwd
/home/students/j_pec2/JPMainDir/CSC122/Port3/frameProject
j_pec2@ares:~/JPMainDir/CSC122/Port3/frameProject$ cat FrameInfo.txt
Jack Pec

CSC122-001

Frame Project

Base level: 5

Add 1.5 for choices of frame types
Add 1 for option configuration thats saved and loaded

Overall level 7.5


Desc:

It's the "A BOX Upon Ye" Projectj_pec2@ares:~/JPMainDir/CSC122/Port3/frameProject$
frame.h  FrameInfo.txt  frameMe.txt  frameOut.txt  FrameProjectDriver.cpp  saveOp.
j_pec2@ares:~/JPMainDir/CSC122/Port3/frameProject$ show-code FrameProjectDriver.cp


FrameProjectDriver.cpp:
```

```
#include <iostream>
#include <fstream>

#include "frame.h"

using namespace std;


ofstream openOutputfile();
void closeOutputFile(ofstream & file);
void mainframe();


int main(void)
{

    bool run = true;
    char in;

    cout << "Frame Project:\n";

    do
    {
        mainframe();

        cout << "\nPress q to quit, any other char to continue\n";
        cin >> in;

        if(in == 'q')
        {
            run = false;
        }



    }while(run);

    return 0;
}

void mainframe()
{
    frame source;
    //frame mainFrame;

    char framechoice = '1';
    char indentchoice = '4';
    char choice = '1';
    char choice2 = '0';
    char borderChoice = '*'; //default

    string savename = "saveOp.txt";

    ifstream save_fileIn;


    ofstream outputfile;




    save_fileIn.open(savename);

    if (!save_fileIn)
    {
        save_fileIn.close();
        save_fileIn.clear();

        cerr << "Could not read in save file, A new save file will be\n"
             "made after a getting settings for a frame";

    }
    else
    {

        //save_fileIn >> ws;

        save_fileIn >> framechoice;
        save_fileIn >> indentchoice;
        save_fileIn >> choice;
        save_fileIn >> choice2;
        save_fileIn >> borderChoice;

        //save_fileIn >> ws;

        /*
        cout << framechoice << "\n";
        cout << indentchoice<< "\n";
        cout << choice<< "\n";
        cout << choice2<< "\n";
        cout << borderChoice<< "\n";
        */

        save_fileIn.close();
        save_fileIn.clear();

    }


    char y;
    cout << "Edit/create options executing program?\n"
         "Press y, otherwise press any key\n"
         "to run program based on previous settings\n";
    cin >> y;
    if(y == 'y')
    {
        cout << "Press 1 for single line frame\n"
             "Press 2 double line frame\n"
             "Press 3 for shaded line frame\n"
             "Press 4 for custom frame from one char\n";


        cin >> framechoice;

        if(framechoice == '4')
        {
            cout << "Enter a char for it's border: ";
            cin >> borderChoice;
        }

        cout << "Press 1 for Left indentation\n"
             "Press 2 for Right indentation\n"
             "Press 3 for Centered indentation\n";


        cin >> indentchoice;

        cout << "Press 1 to enter phrases from file,"
             << " any other number to input manually: ";

        cin >> choice;

        cout << "Press 1 to enter print into a file,"
             << " any other number to print onto screen: ";

        cin >> choice2;
    }

    switch(framechoice)
    {
    case '1':
        source.thisBorder = frame::ChoiceBorder::singleL;
        break;

    case '2':
        source.thisBorder = frame::ChoiceBorder::doubleL;
        break;

    case '3':
        source.thisBorder = frame::ChoiceBorder::shaded;
        break;
    case '4':

        source.thisBorder = frame::ChoiceBorder::defaultB;
        source.set_border(borderChoice);

        break;


    }

    switch(indentchoice)
    {
    case '1':
        source.thisChoice = frame::ChoiceOri::left;
        break;

    case '2':
        source.thisChoice = frame::ChoiceOri::right;
        break;

    case '3':
        source.thisChoice = frame::ChoiceOri::center;
        break;

    default:
        source.thisChoice = frame::ChoiceOri::left;
        break;
    }

    if(choice2 == '1')
    {
        outputfile = openOutputfile();
    }


    if(choice == '1')
    {
        ifstream data_file;
        string fname = "";

        cout << "what's the name of input file?  ";
        cin >> fname;
        // }
        data_file.open(fname);

        while (!data_file)
        {
            data_file.close();
            data_file.clear();
            cerr << "\n\aCould not open file '" << fname << "'!\n\n";
            cout << "what's the name of input file?  ";
            getline(cin, fname);
            data_file.open(fname);
        }

        /*
        if(choice2 == '1')
        {
            outputfile = openOutputfile();
        }
        */

        data_file >> ws;

        while(!data_file.eof())
        {

            frame tempy = source;

            if(choice2 == '1')
            {
                data_file >> tempy;

                outputfile << tempy;

                data_file >> ws;

            }
            else
            {
                data_file >> tempy;

                cout << tempy;

                data_file >> ws;
            }

        }

        //closeOutputFile(outputfile);
        data_file.clear();
        data_file.close();
    }
    else if(choice != '1')
    {

        if(choice2 == '1')
        {
            cout << "Enter Phrase : ";
            frame k = source;

            cin >> k;

            outputfile << k;
        }
        else
        {
            cout << "Enter Phrase: ";
            frame k = source;

            cin >> k;

            cout << k;

        }

    }

    closeOutputFile(outputfile);

    ofstream save_fileOut;

    save_fileOut.open(savename,std::ofstream::trunc);

    /*
           cout << framechoice << "\n";
           cout << indentchoice<< "\n";
           cout << choice<< "\n";
           cout << choice2<< "\n";
           cout << borderChoice<< "\n";
     */

    save_fileOut << framechoice;
    save_fileOut << indentchoice;
    save_fileOut << choice;
    save_fileOut << choice2;
    save_fileOut << borderChoice;

    save_fileOut.close();
    save_fileOut.clear();



}

```

```

ofstream openOutputfile()
{
    ofstream file;
    string fname;
    cout << "What should the Output file name be?  ";
    //getline(cin, fname);
    cin >> fname;
    file.open(fname);


    while (!file)
    {
        file.close();
        file.clear();
        cout << "\nCannot write to '" << fname
             << "'!!\a\n\nPlease choose "
             "another name (and/or location):  ";
        //getline(cin, fname);
        cin >> fname;
        file.open(fname);
    }

    return file;


}

void closeOutputFile(ofstream & file)
{
    file.close();
    file.clear();

}
```

```
j_pec2@ares:~/JPMainDir/CSC122/Port3/frameProject$ ls
frame.h  FrameInfo.txt  frameMe.txt  frameOut.txt  FrameProjectDriver.cpp  saveOp.
j_pec2@ares:~/JPMainDir/CSC122/Port3/frameProject$ show-code frame.h


frame.h:

```

```
/*

Frame Lib


*/
#include <string>
#include <sstream>
#include <iomanip>

#pragma once


class frame
{
    std::string val;
    char borderChar;


public:

    enum class ChoiceOri {left,right,center};

    ChoiceOri thisChoice = ChoiceOri::left; //okay its public

    enum class ChoiceBorder {defaultB,singleL,doubleL,shaded};

    ChoiceBorder thisBorder = ChoiceBorder::defaultB;

    frame(std::string valIn = "", char borderIn = '*')
        :
        val(valIn),
        borderChar(borderIn)
    {

    }

    void set_border(char b)
    {
        borderChar = b;
    }

    char get_border()
    {
        return borderChar;
    }








    friend std::istream & operator>>(std::istream & in, frame & r)
    {

        //takes care of reading in whitespace
        if (in.peek() == '\n')
        {
            in.ignore();
        }


        std::getline(in,r.val);

        return in;
    }

    friend std::ostream & operator<<(std::ostream & out,
                                     const frame & r)
    {
        std::string a = "";
        std::string largestWord = "";

        std::string main = r.val;

        std::istringstream iss;

        // std::string borderStringSide = r.borderChar;
        std::string borderStringSide(1, r.borderChar);

        iss.str(main);

        do //first pass
        {
            iss >> a;

            if(a.length() > largestWord.length())
            {
                largestWord = a;

            }

        }
        while (!iss.eof());

        iss.clear();
        iss.str (main);

        std::string borderString;

        switch(r.thisBorder)
        {

        case ChoiceBorder::defaultB:

            //out << r.borderChar <<" ";
            borderString =
                std::string(largestWord.length()+4, r.borderChar);
            out << borderString << '\n';

            break;

        case ChoiceBorder::singleL:
            borderString = "+"
                           + std::string(largestWord.length()+2, '-') + "+'
            out << borderString << '\n';
            borderStringSide = "|";


            break;

        case ChoiceBorder::doubleL:
            borderString = "+"
                           + std::string(largestWord.length()+4, '=') + "+'
            out << borderString << '\n';
            borderStringSide = "||";

            break;

        case ChoiceBorder::shaded:
            borderString = "+"
                           + std::string(largestWord.length()+2, '-') + "+'
            out << borderString << '\n';
            borderStringSide = "|";

            break;
        }

        do
        {
            iss >> a;

            out << borderStringSide <<" ";

            switch(r.thisChoice)
            {

            case ChoiceOri::left:

                out << std::left
                    << std::setw(static_cast<int>(largestWord.length()))
                    << a;
                break;

            case ChoiceOri::right:

                out  << std::right
                     << std::setw(static_cast<int>(largestWord.length()))
                     << a;
                break;

            case ChoiceOri::center:
                out << std::setw(static_cast<int>(
                                     (largestWord.length()-a.length())/2))
                    << "" << a
                    << std::setw(static_cast<int>(
                                     largestWord.length()-a.length()
                                     -(largestWord.length()-a.length())/2)
                    << "";
                break;
            }
            if( r.thisBorder == ChoiceBorder::shaded)
            {
                out << " "<< borderStringSide << "*" << '\n';
            }
            else
            {
                out << " "<< borderStringSide  << '\n';
            }


        }
        while (!iss.eof());

        if( r.thisBorder == ChoiceBorder::shaded)
        {
            out << borderString << "*" << '\n';
            out << std::string(largestWord.length()+4, '*') << "*" << '\n'
        }
        else
        {
            out << borderString <<'\n';
        }



        return out;

    }

};
j_pec2@ares:~/JPMainDir/CSC122/Port3/frameProject$ ls
frame.h  FrameInfo.txt  frameMe.txt  frameOut.txt  FrameProjectDriver.cpp  saveOp.t
j_pec2@ares:~/JPMainDir/CSC122/Port3/frameProject$ CPP FrameProjectDriver.cpp frame
FrameProjectDriver.cpp***


j_pec2@ares:~/JPMainDir/CSC122/Port3/frameProject$ ./FrameProjectDriver.out
Frame Project:
Edit/create options executing program?
Press y, otherwise press any key
to run program based on previous settings
y
Press 1 for single line frame
Press 2 double line frame
Press 3 for shaded line frame
Press 4 for custom frame from one char
1
Press 1 for Left indentation
Press 2 for Right indentation
Press 3 for Centered indentation
1
Press 1 to enter phrases from file, any other number to input manually: 1
Press 1 to enter print into a file, any other number to print onto screen: 1
```

```
What should the Output file name be?  frameOut.txt
what's the name of input file?  frameMe.txt

Press q to quit, any other char to continue
h
Edit/create options executing program?
Press y, otherwise press any key
to run program based on previous settings
y
Press 1 for single line frame
Press 2 double line frame
Press 3 for shaded line frame
Press 4 for custom frame from one char
2
Press 1 for Left indentation
Press 2 for Right indentation
Press 3 for Centered indentation
2
Press 1 to enter phrases from file, any other number to input manually: 1
Press 1 to enter print into a file, any other number to print onto screen: g
what's the name of input file?  frameMe.txt
+=========+
|| hello ||
|| world ||
+=========+
+===========+
||     the ||
||     sky ||
||      is ||
|| falling ||
+===========+
+=======+
||   O ||
||   | ||
|| -+- ||
||   | ||
|| /-\ ||
+=======+

Press q to quit, any other char to continue
h
Edit/create options executing program?
Press y, otherwise press any key
to run program based on previous settings
h
what's the name of input file?  frameMe.txt
+=========+
|| hello ||
|| world ||
+=========+
+===========+
||     the ||
||     sky ||
||      is ||
|| falling ||
+===========+
+=======+
||   O ||
||   | ||
|| -+- ||
||   | ||
|| /-\ ||
+=======+

Press q to quit, any other char to continue
h
Edit/create options executing program?
Press y, otherwise press any key
to run program based on previous settings
y
Press 1 for single line frame
Press 2 double line frame
Press 3 for shaded line frame
Press 4 for custom frame from one char
3
Press 1 for Left indentation
Press 2 for Right indentation
Press 3 for Centered indentation
3
Press 1 to enter phrases from file, any other number to input manually: 1
Press 1 to enter print into a file, any other number to print onto screen: g
what's the name of input file?  frameMe.txt
+-------+
| hello |*
| world |*
+-------+*
**********
+---------+
|   the   |*
|   sky   |*
|   is    |*
| falling |*
+---------+*
************
+-----+
|  O  |*
|  |  |*
| -+- |*
|  |  |*
| /-\ |*
+-----+*
********

Press q to quit, any other char to continue
h
Edit/create options executing program?
Press y, otherwise press any key
to run program based on previous settings
```

```
y
Press 1 for single line frame
Press 2 double line frame
Press 3 for shaded line frame
Press 4 for custom frame from one char
4
Enter a char for it's border: %
Press 1 for Left indentation
Press 2 for Right indentation
Press 3 for Centered indentation
3
Press 1 to enter phrases from file, any other number to input manually: g
Press 1 to enter print into a file, any other number to print onto screen: g
Enter Phrase: Don't Frame Me!
%%%%%%%%%
% Don't %
% Frame %
%  Me!  %
%%%%%%%%%

Press q to quit, any other char to continue
h
Edit/create options executing program?
Press y, otherwise press any key
to run program based on previous settings
y
Press 1 for single line frame
Press 2 double line frame
Press 3 for shaded line frame
Press 4 for custom frame from one char
1
Press 1 for Left indentation
Press 2 for Right indentation
Press 3 for Centered indentation
1
Press 1 to enter phrases from file, any other number to input manually: 1
Press 1 to enter print into a file, any other number to print onto screen: g
what's the name of input file?  frameMe.txt
+-------+
| hello |
| world |
+-------+
+---------+
| the     |
| sky     |
| is      |
| falling |
+---------+
+-----+
| O   |
| |   |
| -+- |
| |   |
| /-\ |
+-----+

Press q to quit, any other char to continue
q
j_pec2@ares:~/JPMainDir/CSC122/Port3/frameProject$ exit
exit

Script done on 2022-12-08 20:57:03-06:00 [COMMAND_EXIT_CODE="0"]
```