```
Script started on 2024-12-05 12:17:10-06:00 [TERM="xterm-256color" TTY="/dev/pts/8'
e_prevost@ares:~/Project_1$ pwd
/home/students/e_prevost/Project_1
e_prevost@ares:~/Project_1$ cat box_frame.info
****************

Robert Prevost

CSC 122 W01



box frame project



Takes text from a user and produces a box frame around it



Base Level: Level 5

Total Level: Level 5

****************e_prevost@ares:~/Project_1$ show-code main.cpp


main.cpp:
```

```cpp
#include "BoxFrame.h"
#include <fstream>
#include <limits>
#include <sstream>

void clearInputBuffer() {
    std::cin.clear();
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
}

char getFrameChar() {
    char c;
    while (true) {
        std::cout << "Enter frame character: ";
        std::cin >> c;
        clearInputBuffer();

        if (std::isprint(c)) {
            return c;
        }
        std::cout << "Please enter a printable character.\n";
    }
}

std::string getAlignment() {
    int choice;
    while (true) {
        std::cout << "Select alignment:\n"
                  << "1. Left\n"
                  << "2. Center\n"
                  << "3. Right\n"
                  << "Choice: ";

        std::cin >> choice;
        clearInputBuffer();

        switch (choice) {
            case 1: return "left";
            case 2: return "center";
            case 3: return "right";
            default:
                std::cout << "Invalid choice. Please try again.\n";
        }
    }
}

bool getIOChoice(const std::string& prompt) {
    char choice;
    while (true) {
        std::cout << prompt << " (f for file, k for keyboard/screen): ";
        std::cin >> choice;
        clearInputBuffer();

        if (choice == 'f' || choice == 'F') return true;
        if (choice == 'k' || choice == 'K') return false;
        std::cout << "Invalid choice. Please try again.\n";
    }
}

int main() {
    BoxFrame box;
    std::istream* in;
    std::ostream* out;
    std::ifstream inFile;
    std::ofstream outFile;

    box.setFrameChar(getFrameChar());
    box.setAlignment(getAlignment());

    bool useInputFile = getIOChoice("Select input source");
    bool useOutputFile = getIOChoice("Select output destination");

    if (useInputFile) {
        std::string filename;
        std::cout << "Enter input filename: ";
        std::getline(std::cin, filename);
        inFile.open(filename);
        if (!inFile) {
            std::cerr << "Error opening input file\n";
            return 1;
        }
        in = &inFile;
    } else {
        in = &std::cin;
    }

    if (useOutputFile) {
        std::string filename;
        std::cout << "Enter output filename: ";
        std::getline(std::cin, filename);
        outFile.open(filename);
        if (!outFile) {
            std::cerr << "Error opening output file\n";
            return 1;
        }
        out = &outFile;
    } else {
        out = &std::cout;
    }

    if (!useInputFile) {
        std::cout << "Enter your phrases (type 'quit' to end):\n";
    }

    std::string line;
    while (std::getline(*in, line)) {
        if (!useInputFile && line == "quit") {
            break;
        }
        box.setPhrase(line);
        *out << box << "\n\n";
    }

    return 0;
}
e_prevost@ares:~/Project_1$ show-code BoxFrame.h


BoxFrame.h:


#ifndef BOX_FRAME_H
#define BOX_FRAME_H

#include <string>
#include <vector>
#include <iostream>

class BoxFrame {
private:
    std::vector<std::string> words;
    char frameChar;
    std::string alignment;
    int maxWordLength;

    void splitIntoWords(const std::string& phrase);
    std::string createPaddedLine(const std::string& word) const;
    std::string createBorderLine() const;

public:
    BoxFrame(char frameChar = '*');

    void setFrameChar(char c);
    void setAlignment(const std::string& align);
    void setPhrase(const std::string& phrase);

    friend std::istream& operator>>(std::istream& is, BoxFrame& box);
    friend std::ostream& operator<<(std::ostream& os, const BoxFrame& box)
};

#endif
e_prevost@ares:~/Project_1$ show-code BoxFrame.cpp


BoxFrame.cpp:


#include "BoxFrame.h"
#include <sstream>
#include <cctype>

BoxFrame::BoxFrame(char frameChar) : frameChar(frameChar), alignment("left'

void BoxFrame::setFrameChar(char c) {
    if (std::isprint(c)) {
        frameChar = c;
    }
}

void BoxFrame::setAlignment(const std::string& align) {
    alignment = align;
}

void BoxFrame::setPhrase(const std::string& phrase) {
    splitIntoWords(phrase);
}

void BoxFrame::splitIntoWords(const std::string& phrase) {
    std::istringstream iss(phrase);
    std::string word;
    words.clear();
    maxWordLength = 0;

    while (iss >> word) {
        words.push_back(word);
        if (word.length() > maxWordLength) {
```

```cpp
            maxWordLength = word.length();
        }
    }
}

std::string BoxFrame::createPaddedLine(const std::string& word) const {
    std::string result;
    int padding = maxWordLength - word.length();

    result += frameChar;
    result += ' ';

    if (alignment == "left") {
        result += word + std::string(padding, ' ');
    }
    else if (alignment == "right") {
        result += std::string(padding, ' ') + word;
    }
    else if (alignment == "center") {
        int leftPad = padding / 2;
        int rightPad = padding - leftPad;
        result += std::string(leftPad, ' ') + word + std::string(rightPad,
    }

    result += ' ';
    result += frameChar;

    return result;
}

std::string BoxFrame::createBorderLine() const {
    return std::string(maxWordLength + 4, frameChar);
}

std::istream& operator>>(std::istream& is, BoxFrame& box) {
    std::string line;
    std::getline(is, line);
    box.splitIntoWords(line);
    return is;
}

std::ostream& operator<<(std::ostream& os, const BoxFrame& box) {
    if (box.words.empty()) {
        return os;
    }

    os << box.createBorderLine() << '\n';

    for (const auto& word : box.words) {
        os << box.createPaddedLine(word) << '\n';
    }

    os << box.createBorderLine();

    return os;
}
```

```
e_prevost@ares:~/Project_1$ CPP main BoxFrame
BoxFrame.cpp...
main.cpp***
,.BoxFrame.cpp: In constructor
‘BoxFrame::BoxFrame(char)’:
BoxFrame.cpp:5:25: warning:
declaration of ‘frameChar’ shadows a member
of ‘BoxFrame’ [-Wshadow]
    5 | BoxFrame::BoxFrame(char frameChar) :
    frameChar(frameChar), alignment("left"), maxWordLength(0) {}
      |                    ~~~~~^~~~~~~~~
In file included from BoxFrame.cpp:1:
BoxFrame.h:11:10: note: shadowed
declaration is here
   11 |     char frameChar;
      |          ^~~~~~~~~
BoxFrame.cpp:5:1: warning:
‘BoxFrame::words’ should be initialized in
the member initialization list [-Weffc++]
    5 | BoxFrame::BoxFrame(char frameChar) :
    frameChar(frameChar), alignment("left"), maxWordLength(0) {}
      | ^~~~~~~~
BoxFrame.cpp: In constructor
‘BoxFrame::BoxFrame(char)’:
BoxFrame.cpp:5:25: warning:
declaration of ‘frameChar’ shadows a member
of ‘BoxFrame’ [-Wshadow]
    5 | BoxFrame::BoxFrame(char frameChar) :
    frameChar(frameChar), alignment("left"), maxWordLength(0) {}
      |                    ~~~~~^~~~~~~~~
In file included from BoxFrame.cpp:1:
BoxFrame.h:11:10: note: shadowed
declaration is here
   11 |     char frameChar;
      |          ^~~~~~~~~
BoxFrame.cpp: In constructor
‘BoxFrame::BoxFrame(char)’:
BoxFrame.cpp:5:25: warning:
declaration of ‘frameChar’ shadows a member
of ‘BoxFrame’ [-Wshadow]
    5 | BoxFrame::BoxFrame(char frameChar) :
    frameChar(frameChar), alignment("left"), maxWordLength(0) {}
      |                    ~~~~~^~~~~~~~~
In file included from BoxFrame.cpp:1:
BoxFrame.h:11:10: note: shadowed
declaration is here
   11 |     char frameChar;
      |          ^~~~~~~~~
BoxFrame.cpp: In member function ‘void
BoxFrame::splitIntoWords(const string&)’:
BoxFrame.cpp:29:27: warning:
comparison of integer expressions of different signedness:
```

```
‘std::__cxx11::basic_string<char>::size_type’ {aka
‘long unsigned int’} and ‘int’
[-Wsign-compare]
   29 |         if (word.length() > maxWordLength) {
      |             ~~~~~~~~~~~~~~^~~~~~~~~~~~~~~
BoxFrame.cpp:30:40: warning: conversion
from ‘std::__cxx11::basic_string<char>::size_type’
{aka ‘long unsigned int’} to ‘int’
may change value [-Wconversion]
   30 |             maxWordLength = word.length();
      |                             ~~~~~~~~~~~^~
BoxFrame.cpp: In member function ‘std::string
BoxFrame::createPaddedLine(const string&) const’:
BoxFrame.cpp:37:33: warning: conversion
from ‘std::__cxx11::basic_string<char>::size_type’
{aka ‘long unsigned int’} to ‘int’
may change value [-Wconversion]
   37 |     int padding = maxWordLength -
   word.length();
      |
      ~~~~~~~~~~~~~~^~~~~~~~~~~~~~~


e_prevost@ares:~/Project_1$ ./main.out
Enter frame character: *
Select alignment:
1. Left
2. Center
3. Right
Choice: 2
Select input source (f for file, k for keyboard/screen): f
Select output destination (f for file, k for keyboard/screen): k
Enter input filename: data.txt
*********
* Frame *
* this  *
*********

**********
* HAHAHA *
**********

******
* Hi *
* HI *
* HI *
******

**********
* Potato *
**********

e_prevost@ares:~/Project_1$ ./main.out
Enter frame character: $
```

```
Select alignment:
1. Left
2. Center
3. Right
Choice: 3
Select input source (f for file, k for keyboard/screen): f
Select output destination (f for file, k for keyboard/screen): f
Enter input filename: data.txt
Enter output filename: output.txt
e_prevost@ares:~/Project_1$ cat output.txt
$$$$$$$$$
$ Frame $
$  this $
$$$$$$$$$

$$$$$$$$$$
$ HAHAHA $
$$$$$$$$$$

$$$$$$
$ Hi $
$ HI $
$ HI $
$$$$$$

$$$$$$$$$$
$ Potato $
$$$$$$$$$$

e_prevost@ares:~/Project_1$ ./main.out
Enter frame character: #
Select alignment:
1. Left
2. Center
3. Right
Choice: 1
Select input source (f for file, k for keyboard/screen): k
Select output destination (f for file, k for keyboard/screen): k
Enter your phrases (type 'quit' to end):
Its time to party everybody put your hands in the air like you just dont care woot
#############
# Its       #
# time      #
# to        #
# party     #
# everybody #
# put       #
# your      #
# hands     #
# in        #
# the       #
# air       #
# like      #
# you       #
```

```
# just      #
# dont      #
# care      #
# woot      #
# woot      #
#############

Im so coool
#########
# Im    #
# so    #
# coool #
#########

Hello people of the owrld
##########
# Hello  #
# people #
# of     #
# the    #
# owrld  #
##########

quit
e_prevost@ares:~/Project_1$ ./main.out
Enter frame character: L
Select alignment:
1. Left
2. Center
3. Right
Choice: 2
Select input source (f for file, k for keyboard/screen): k
Select output destination (f for file, k for keyboard/screen): f
Enter output filename: output.txt
Enter your phrases (type 'quit' to end):
Its time to get this party started
fortnite fortnite fortnite
Give me an A Professor
quit
e_prevost@ares:~/Project_1$ cat output.txt
LLLLLLLLLLL
L   Its   L
L  time   L
L   to    L
L   get   L
L  this   L
L  party  L
L started L
LLLLLLLLLLL

LLLLLLLLLLLL
L fortnite L
L fortnite L
L fortnite L
LLLLLLLLLLLL

LLLLLLLLLLLLL
L   Give    L
L    me     L
L    an     L
L     A     L
L Professor L
LLLLLLLLLLLLL

e_prevost@ares:~/Project_1$ exit
exit

Script done on 2024-12-05 12:20:26-06:00 [COMMAND_EXIT_CODE="0"]
```