CS 201 Homework 03
Computation

Sarah Carter

September 29, 2019

Source Code Link: `https://github.com/SarahFDAK/CS201.git`
This homework took approximately 4-5 hours to complete.

1 Design

I wrote out ideas for my programs in a notebook this time, so I already had some things outlined before I started writing the actual program. I think that helped make the process a little neater.

2 Post Mortem

Initially, I forgot to finish the box and include the text in the boxPrint program - I successfully wrote a program that would print the top portion of the box. I added in extra functions in boxer.cpp to clean up some code redundancy. A friend gave me handy advice to use do-while loops rather than just while loops. I had tried it before and probably was battling syntax errors somewhere; they make the whole program considerably more concise, which makes me happy.

3 Answers to Questions

- Four types of errors are compile-time, link-time, run-time, and logic. Compile-time errors are found by the compiler in the form of the program, such as syntax or type errors. Link-time errors are found by the linker when it is trying to combine object files into an executable program. Run-time errors are found by checks in a program when it is running. Logic errors are found by the programmer when editing and debugging code to find the cause of incorrect results.
- We really shouldn't ignore any errors in student programs. Even the most inconsequential-seeming error can snowball into major problems with a program. If you want to ignore an error, I guess a misspelling like "Hello wrlodl" in a hello world program could happen, but WHY.
- Every completed project should guarantee that the program will produce the desired and described results for any legal input; it should give an understandable error for illegal inputs, which explains what went wrong; users should not have to worry about misbehaving hardware or software; and the project is allowed to terminate and does not loop into oblivion upon finding an error.
- I supposed debugging is unpopular because it is relatively interminable. It also tends to indicate that the program is not working as desired, which isn't encouraging.
- The steps in debugging are: get the program to compile, get the program to link to functions and libraries, and get the program to work as intended.
- Commenting helps when debugging because it can clarify what something is meant to do (vs what it actually does). It also explains what changes were made and what was corrected.

- Debugging is the process of going through the code to find errors. Testing is when you actually run the program to see if it compiles, links, and actually works as expected. If the test fails on one or more fronts, you go back to debugging the problem.

4 Sample Output

Listing 1: Sample Program Output

```
Please enter a word and a positive integer (enter a 0
   to exit):
wonder 1
**********
*        *
* wonder *
*        *
**********
Please enter a word and a positive integer (enter a 0
   to exit):
hello 5
******************
******************
******************
******************
******************
*****        *****
***** hello *****
*****        *****
******************
******************
******************
******************
******************
Please enter a word and a positive integer (enter a 0
   to exit):
you 2
```

```
*********
*********
**      **
** you **
**      **
*********
*********
Please enter a word and a positive integer (enter a 0
   to exit):
dog-pony-ride 3
**********************
**********************
**********************
***                  ***
*** dog-pony-ride ***
***                  ***
**********************
**********************
**********************
Please enter a word and a positive integer (enter a 0
   to exit):
yo 0
Thank you for your participation!
```

5 My Program

```
//
//  homework 3 - boxPrint.cpp
//  Sarah Carter
//  9/23/19
//  This program will request a word and a positive integer from the user, then will

#include <iostream>
#include "boxer.hpp"

int main(int argc, const char * argv[]) {
    //Get a word and an integer from the user
    std::string userWord;
    int userInt;

    //If they enter a positive integer, print the box, then ask for new input.
```

```
    do {
        std::cout << "Please enter a word and a positive integer (enter 'a 0' to exi
        std::cin >> userWord >> userInt;
        //Check if they entered a positive number. If not, ask for one. Keep asking
        if(userInt < 0){
            std::cin.clear();
            continue;
        }
        boxer(userWord, userInt);
        std::cin.clear();
    } while(userInt > 0);
    //When they have entered 0, thank them and exit the program.
    std::cout << "Thank you for your participation!\n";
    return 0;
}
```

5.1 Boxer Header

```
//
//  boxer.hpp
//  boxerPrint
//
//  Created by Sarah Carter on 9/23/19.
//  Copyright © 2019 Sarah Carter. All rights reserved.
//

#ifndef boxer_hpp
#define boxer_hpp

#include <stdio.h>
#include <iostream>

void boxer(std::string userWord, int userInt);

#endif /* boxer_hpp */
```

5.2 Boxer Source

```
//
//  boxer.cpp
//  boxerPrint
//
//  Created by Sarah Carter on 9/23/19.
//  Copyright © 2019 Sarah Carter. All rights reserved.
//

#include "boxer.hpp"
```

```cpp
#include <iostream>

//Declare function prototypes
void printMiddle(int userInt, int length);
void printAllAsterisks(int userInt, int length);
void printSideStars(int userInt, int length);

void boxer(std::string userWord, int userInt){
    //Calculate the length of the word entered by the user
    int length = userWord.size();
    //Call various print functions to create the asterisk box around the word.
    printAllAsterisks(userInt, length);
    printMiddle(userInt, length);
    printSideStars(userInt, length);
    std::cout << " " << userWord << " ";
    printSideStars(userInt, length);
    std::cout << std::endl;
    printMiddle(userInt, length);
    printAllAsterisks(userInt, length);
}
//Prints the line above and below the word
void printMiddle(int userInt, int length){
    printSideStars(userInt, length);
    for (int l = 0; l < (length + 2); l++){
        std::cout << " ";
    }
    printSideStars(userInt, length);
    std::cout << std::endl;
}
//Prints the blocks of asterisks at top and bottom of the box
void printAllAsterisks(int userInt, int length){
    for (int i = 0; i < userInt; i++){
        for (int j = 0; j < (2*userInt)+length +2; j++){
            std::cout << "*";
        }
        std::cout << std::endl;
    }
}
//Prints the asterisks to the left and right of the word
void printSideStars(int userInt, int length){
    for (int k = 0; k < userInt; k++){
        std::cout << "*";
    }
}
```

5.3 Collatz Sequence

```cpp
//
//  collatz.cpp
//  Sarah Carter
//  9/24/19
//  This program asks the user for a positive integer, checks to make sure it is pos
//

#include <iostream>

int main(int argc, const char * argv[]) {
    int c = 0;

    do {
        //Request integer
        std::cout << "Please enter a postive integer (enter 0 to exit): \n";
        std::cin >> c;
        std::cout << c << " ";
        //Check if number is positive. If not, request a new number.
        if(c < 0){
            std::cin.clear();
            continue;
        }
        //Print Collatz sequence, stopping when 1 is reached
        while(c > 1){
            //Check if number is even or odd
            if(c%2 != 0){
                c = 3*c +1;
            }
            else{
                c = c/2;
            }
            std::cout << c << " ";
        }
        std::cout << std::endl;

    } while(c != 0);
    return 0;
}
```

5.4 Kelvin Conversion w/Correction

```cpp
//
//  kelvin.cpp
//  Sarah Carter
//  9/25/19
//  This program asks the user for a temperature in degrees celcius, then returns th
//

```

```cpp
#include <iostream>

double ctok(double c){
    double k = c + 273.15;
    return k;
}

int main(){
    double c = 0;
    std::cout << "Enter a celcius temperature to convert to kelvin: \n";
    std::cin >> c;
    while(c < -273.15){
        std::cout << "Please enter a temperature that can exist (is greater than abs
        std::cin.clear();
        std::cin >> c;
    }
    double k = ctok(c);
    std::cout << k << "K\n";
    return 0;
}
//
//double ctok(double c){
//    int k = c + 273.15; //k need to be a double
//    return int; //Should return k, which is a double, not an int
//}
//int main(int argc, const char * argv[]) {
//    double c = 0;
//    cin >> d; //std library not declared, does not request a number, variable d is
//    double k = ctok("c"); //passes a string, not a double, to ctok
//    Cout << k << '/n'; //std library is not included, "Cout" should not be capital
//    //no return statement - maybe not necessary?
//}
```
