```
Script started on 2024-02-19 21:00:52-06:00 [TERM="xterm" TTY="/dev/pts/3" COLUMNS=
ee43254@ares:~$ pwd
/home/students/ee43254
ee43254@ares:~$ cat input_prot.info
Name: Kyle Enkhzul

Class: CSC122-W01



Activity: Oops...shall we try again?

Level: 3, 2 (base program), 1 (as few overloaded functions)



Description:



This lab requires the coding of libraries in order to validate input and

arrays. It also helps practice function overloading with the usage of templates.

This library file allows general reusability and is meant to be an all purpose

input file for validating user input when related to bounds or choices.

ee43254@ares:~$ show-code input_prot.cpp


input_prot.cpp:
```

```cpp
#include <iostream>
#include <string>

#include "input_prot.h"

using namespace std;

int main(void) {
        // Testing getBounds function
        short num1, num2;

        cout << "\nWhat are your lower and upper bounds? ";
        cin >> num1 >> num2;

        input_protect(num1, num2, "\nWhat is your lower bound? ",
        "\nWhat is your upper bound? ", "\nPlease enter a correct bound! "
        // Testing getLowerBound function
        double num3, num4;
        cout << "\nWhat is your old and new lower bound? ";
        cin >> num3 >> num4;

        getLowerBound(num3, num4, "What is your new lower bound? ",
                "Enter a new lower bound that is less than your old lower b


        // Testing getUpperBound function
        long num5, num6;
        cout << "\nWhat is your old and new upper bound? ";
        cin >> num5 >> num6;

        getUpperBound(num5, num6, "What is your new upper bound? ",
                "Enter a new upper bound that is greater than your old uppe

        // Testing choices function

        char ans;
        cout << "\nWould you like to continue? ";
        cin >> ans;

        input_protect(ans, "Would you like to continue? ", "\nPlease enter
                {'Y', 'N', 'n', 'y'});

        return 0;
}
```

```
ee43254@ares:~$ show-code input_prot.h


input_prot.h:
```

```cpp
#ifndef INPUT_PROT_H_INC
#define INPUT_PROT_H_INC

#include <iostream>
#include <string>
#include <ios>

template < typename InputT >
void input_protect(InputT & lower, InputT & upper, const std::string & lowe
        const std::string & upperPrompt, const std::string & error) {

        std::cout << "Validating bounds...\n";

        while(lower > upper) {
                std::cerr << error;
                std::cout << lowerPrompt;
                std::cin >> lower;
        }
        while(upper < lower) {
                std::cerr << error;
                std::cout << upperPrompt;
                std::cin >> upper;
```

```
23          }
24
25          std::cout << "Your lower bound is " << lower << " and your upper b
26                  << upper << "\n";
27          }
28
29  template < typename InputT >
30  void getLowerBound(InputT & oldLowerBound, InputT & newLowerBound,
31          const std::string & promptLower, const std::string & error) {
32
33          std::cout << "Validating lower bounds...\n";
34
35          while(newLowerBound > oldLowerBound) {
36                  std::cerr << error;
37                  std::cout << promptLower;
38                  std::cin >> newLowerBound;
39          }
40
41          std::cout << "Your new lower bound is now " << newLowerBound << "\
42  }
43
44  template <typename InputT>
45  void getUpperBound(InputT & oldUpperBound, InputT & newUpperBound,
46          const std::string & promptUpper, const std::string & error) {
47
48          std::cout << "Validating upper bounds...\n";
49
50          while(newUpperBound < oldUpperBound) {
51                  std::cerr << error;
52                  std::cout << promptUpper;
53                  std::cin >> newUpperBound;
54          }
55
56          std::cout << "Your new upper bound is now " << newUpperBound << "\
57  }
58
59  template <typename InputT>
60  void input_protect(InputT & value, const std::string & prompt, const std::
61          const std::initializer_list<InputT> & myList) {
62
63          bool valueFound = false;
64          std::cout << "Validating choice...\n";
65
66          while(!valueFound) {
67                  for(auto i = myList.begin(); i != myList.end(); ++i) {
68                          if(*i == value) {
69                                  std::cout << "Value found! \n";
70                                  valueFound = true;
71                                  break;
72                          }
73                  }
74                  if(!valueFound) {
75                  std::cout << errormsg;
76                  std::cout << prompt;
77                          std::cin >> value;
78                          }
79          }
80          std::cout << value;
81  }
82
83  #endif
```

```
ee43254@ares:~$ CPP input_prot
input_prot.cpp***


ee43254@ares:~$ ./input_prot.out

What are your lower and upper bounds? 10 5
Validating bounds...

Please enter a correct bound!
What is your lower bound? 3
Your lower bound is 3 and your upper bound is 5

What is your old and new lower bound? 9.0 4.0
Validating lower bounds...
Your new lower bound is now 4

What is your old and new upper bound? 500 1000
Validating upper bounds...
Your new upper bound is now 1000

Would you like to continue? K
Validating choice...

Please enter Y or N only!
Would you like to continue? G

Please enter Y or N only!
Would you like to continue? Q

Please enter Y or N only!
Would you like to continue? Y
Value found!
Yee43254@ares:~$ cat input_prot.tpq
1. How can you pass a prompt or error message to a function as an argument?

        One can pass a prompt or an error message to a function by having the

        template function in the library take a parameter of a string reference.



2. How do you pass a string to a function? Will the strings need to be changed

here? What care do you need to take for these arguments, then?

        You can pass a string to a function by reference. If you want to change
```

the string you can simply change it in the function arguments. You need to make sure that the string is const and std:: and passed by reference.

3. How do you pass a list of values to a function? Will the elements need to be changed here? What care do you need to take for these arguments, then?

You can pass a list of values to a function by providing an initialized lis of chars. The elements will need to be changed there and you need to make sure you use curly brackets because it is an initialized list.

4. What other arguments does each function take? Are they changed? What special care should you take with them?

Since I overloaded my functions, any of the arguments can take any major data types of double, long, or char. One just needs to make sure that the code they are working with is static_casted especially with char variables

5. What value is returned by your functions? What type is it and what does it represent?

The value returned by the functions are the typename variables and it represents any data that it needs to. It is sort of an umbrella term.

6. What care does a caller of your functions have to take with this return value?

The functions must validate the user input and if it is not valid data, then the functions must ask the user again until it has valid input.

7. How does the compiler distinguish which of your functions is being used for a particular call?

The compiler distinguishes which of the functions is being used by the different arguments it takes. Thus, a function can be overloaded but still perform different tasks depending on data given.

8. How do you protect your library from being circularly included?

By including guards such as #ifndef, #define, and #endif.

9. What changes are needed in your main application to get it to work with the library? What about the compiling process?

To get the library to work in the main application one must do #inc the library file.

10. How many files does your libraryconsist of? What are they? Which one(s) do you #include?

My library consists of three files. They are ios, iostream, and str I #include'ed all of them.

ee43254@ares:~$ exit
exit

Script done on 2024-02-19 21:02:39-06:00 [COMMAND_EXIT_CODE="0"]