```
Script started on 2024-04-11 17:44:23-05:00 [TERM="xterm" TTY="/dev/pts/5" COLUMNS=
ee43254@ares:~$ pwd
/home/students/ee43254
ee43254@ares:~$ cat month.info
Name: Kyle Enkhzul

Class: CSC122-W01



Activity: Beware the Ides of March!

Level: 6, 4 (base program), 2 (inlining + const)



Description:



This lab requires the coding and design of a class and accompanying library

to represent a single month of the year. The program is intended to take the

input of a the number of the month, the first three letters of the month, or

the full name of the month depending on user needs. The program can then

calculate the months in advance, the months before, and whether two months

in comparison are the same or not.

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


month.cpp:
```

```
 1  #include <iostream>
 2  #include <string>
 3  #include "month.h"
 4
 5  using namespace std;
 6
 7                  Month::Month(void)
 8                          : monthNumber(1) { }
 9
10                  Month::Month(string monthInput)
11                          : monthNumber(getMonthNumberFromName(monthInput)) {
12
13
14                  Month::Month(unsigned short monthInput)
15                          : monthNumber(monthInput)               { }
16
17                  Month::Month(const Month & m)
18                          : monthNumber(m.monthNumber) { }
19
20                  const string monthNames[] = {
21                          "January", "February", "March", "April", "May", "Ju
22                          "August", "September", "October", "November", "Dece
23
24                  unsigned short Month::get_month(void) {
25                          return monthNumber;
26                  }
27
28                  void Month::set_month(string monthInput) {
29                          monthNumber = getMonthNumberFromName(monthInput);
30                  }
31
32                  void Month::set_month(unsigned short monthInput) {
33                          monthNumber = monthInput;
34                  }
35
36                  void Month::output(void) {
37                          cout << "Month: " << monthNames[monthNumber - 1] <<
38                          monthNumber << ")\n";
39                  }
40
41                  void Month::input(void) {
42                          string userInput;
43                          cin >> userInput;
44
45                          if(userInput.length() < 3 && isdigit(userInput[0]))
46                                  monthNumber = static_cast<unsigned short>(s
47                          }
48                          else if(userInput.length() >= 3) {
49                                  monthNumber = getMonthNumberFromName(userIn
50                          }
51                          else {
52                                  cerr << "Invalid input. Setting month to Ja
53                                  monthNumber = 1;
54                          }
55                  }
56
57                  inline void Month::advance(unsigned short months) {
58                          monthNumber += months;
59                  monthNumber = static_cast<unsigned short>((monthNumber - 1)
60                  }
61
62                  void Month::before(string monthInput) {
63                          monthNumber -= getMonthNumberFromName(monthInput);
64                          if(monthNumber <= 0) {
65                                  monthNumber = 12 + monthNumber;
66                          }
67                  }
68                  inline void Month::before(unsigned short months) {
69                          monthNumber -= months;
70                          if(monthNumber <= 0) {
```

```
                                monthNumber = 12 + monthNumber;
                        }
                }

                inline bool Month::same(const Month & other) const {
                        return monthNumber == other.monthNumber;
                }

int main() {
        Month myMonth;

        cout << "Enter the month (number or name or 3-letter abbreviation)
        myMonth.input();

        myMonth.output();

        // After
        short advanceMonths;
        cout << "Enter the number of months to advance: ";
        cin >> advanceMonths;
        myMonth.advance(advanceMonths);
        myMonth.output();

        // Before
        short beforeMonths;
        cout << "Enter the number of months to go back: ";
        cin >> beforeMonths;
        myMonth.before(beforeMonths);
        myMonth.output();


        // Same
        Month anotherMonth;
        cout << "Enter the month (number or name or 3-letter abbreviation)
        anotherMonth.input();

        if(myMonth.same(anotherMonth)) {
                cout << "Both months are the same." << endl;
        }
        else {
                cout << "The months are different." << endl;
        }

        return 0;
}
ee43254@ares:~$ show-code month.h


month.h:


#ifndef MONTH_H_INC
#define MONTH_H_INC

#include <iostream>
#include <cctype>
#include <string>

class Month {
        private:

                // Private variable
                unsigned short monthNumber;

                // Member function to convert string to number
                unsigned short getMonthNumberFromName(std::string name){
                        std::string months[12] = {"jan", "feb", "mar", "apr
                        "jun", "jul", "aug", "sep", "oct", "nov",
                        "dec"};
                        for(char &s : name) {
                                s = static_cast<char>(tolower(static_cast<
                        }
                        for(unsigned short i= 0;i < 12; i++){
                                if(static_cast<std::string>(name.substr(0,3
                                        return static_cast<unsigned short>(
                                }
                                else{
                                }
                        }
                        return 1;
                }

        public:

                // Constructors
                Month(void);
                Month(std::string monthInput);
                Month(unsigned short monthNumber);
                Month(const Month & m);

                Month & operator = (const Month &) = default;

                // Getters and Setters
                unsigned short get_month(void);
                void set_month(unsigned short monthInput);
                void set_month(std::string monthInput);

                // Input and Output
                void output(void);
                void input(void);

                // Methods to traverse months
                inline void advance(unsigned short months);
                void before(std::string monthInput);
                inline void before(unsigned short months);

                // Comparison of months
                inline bool same(const Month & other) const;
```

```
    58
    59  };
    60
    61  #endif
ee43254@ares:~$ CPP month
month.cpp***


ee43254@ares:~$ ./month.out
Enter the month (number or name or 3-letter abbreviation) 5
Month: May (5)
Enter the number of months to advance: 10
Month: March (3)
Enter the number of months to go back: 2
Month: January (1)
Enter the month (number or name or 3-letter abbreviation) 1
Both months are the same.
ee43254@ares:~$ ./month.out
Enter the month (number or name or 3-letter abbreviation) November
Month: November (11)
Enter the number of months to advance: 12
Month: November (11)
Enter the number of months to go back: 6
Month: May (5)
Enter the month (number or name or 3-letter abbreviation) 5
Both months are the same.
ee43254@ares:~$ ./month.out
Enter the month (number or name or 3-letter abbreviation) December
Month: December (12)
Enter the number of months to advance: 6
Month: June (6)
Enter the number of months to go back: 6
Month: December (12)
Enter the month (number or name or 3-letter abbreviation) Dec
Both months are the same.
ee43254@ares:~$ cat month.tpq
1. Do you have any private methods? Do they help support the translation

between number and letter/word name representations which occur at several

places in this class' methods? Why would such functions be private?

Doesn't the user of the class need to call them?


I have one private method that allows the program to convert a string to a

number which then corresponds to a month. They are private because it is only

needed by the program to do the work and would be redundant if the user had to

type in a function in order to calculate what month they want. It is meant to

compartmentalize the process and streamline the user experience.


2. How can you have two methods called set_month? How can your class have

four constructors?! This is lunacy! (Hint: O__r_o_di_g.)


There can be multiple methods and constructors all with the same name because

of overloading with different parameters.


3. Why didn't you need two versions of your method to advance to the next

month? (Hint: What data type is returned?)


We did not need two versions of the advance method because the data type

returned is a short, not a string.


4. Does your input method prompt the user? Why shouldn't it?


The input method does not prompt the user. It should not in order to promote

reusability, reduce redundancy, and allow more flexible testing. If an input

method constantly had prompted the user, it would need to change every time

the programmer needed it for something else.


5. Does your output method(s) print anything besides the month number or

name/abbrev. (as requested) (even an endl)? Why shouldn't it?


It does not due to reasons mentioned in number four. Not having other printed

stuff allows for greater flexibility and reusability.
```

6. How do you know what display method the programmer desires when your output method is called (terrible pun/vocabulary clash, isn't it?)? (Hint: Is a bool enough? Or do you need an enumeration?)

In order to get past this, I just printed both the number of the month and the full month name to seamline user experience.

7. What about the input method? How can it detect what kind of form the user is using and adapt to it?

The input method can take a string and then static_cast it into whatever data type is needed in order to produce the code needed to print.

8. Does your driver program do one test per run or does it allow multiple tests of the class' features during a single run? Which seems more convenient for you/the end programmer?

Allowing multiple tests runs of a classes feature allows the end programmer to fully test and realize the faults and successes of the program. Having to go back and forth to test simple features can be time-consuming and annoying.

9. Are the tests in your driver program hard-coded/literal or are they adaptable to the needs of the programmer running the tests? Which would be more useful?

The tests in the driver are adapted to the needs of the programmer. Adaptability to the needs of the program is always going to be a more useful program than hard-coding.

10. Are the tests your driver can run specifically ordered in some way or can the programmer doing the testing choose what s/he is going to test first, next, ... last? Which would be more convenient/useful?

The tests in the driver are not specifically ordered but rather done in a random order to showcase all the features. The programmer can choose to test any singular feature at any point. The latter is obviously more useful than specifically ordering some way.

ee43254@ares:~$ exit
exit

Script done on 2024-04-11 17:46:11-05:00 [COMMAND_EXIT_CODE="0"]