Fall Semester 2019

Library Publication Storage and Retrieval Application

V0.8 (Milestone 1) - draft

When Books and other publications arrive in a library, they should be tagged and put on shelves, so they are easily retrievable to be lent out to those who need it.

Your task is to design an application that receives the publications and stores them into the system with the information needed for their retrieval.
Later, each publication can be lent out to members of the library with a due date for return.

Before we start developing the application, we need to have few classes developed to help us with the dates in the system and also the user interface of the application.

PROJECT DEVELOPMENT NOTES

- When developing the classes in this project, note that you may add additional member variables, member functions to any class if you find them necessary or helpful.
  Make sure these additional variables and functions are well documented.
- An extra empty module is provided with the project in case you would like to add any helper functions of your own or additional classed to add to your project.
  This module has two files: Utils.h and Utils.cpp.
  Utils.h will be included an all the tester files and Utills.cpp will be added to the compile line of all submissions.
- You may reuse and copy any code your professor provided for your workshops or functions you may have from previous work in this subject or other subjects and place it in the Utils module.

- If you choose not to reuse any of your code, just leave these file empty but have them present when submitting your code.

OVERVIEW OF THE CLASSES TO BE DEVELOPED FOR MILESTONE 1

All the code developed in this project should be under the namespace sdds;

Date Class

A class the encapsulates year, and month and day values for Date stamp, comparison and Date IO purposes.

MenuItem Class

A class that hold a text Item; (an option or title to be displayed) in a menu to be selected by the user. This is a fully private class that is only accessible by Menu (see next class)

Menu Class

A class that has several **MenuItem**s to be displayed so the user can select one of them for an action to be executed in the program

THE DATE CLASS

The Date class was partially implemented by another program that left the company and your responsibly is to reuse the parts she developed to complete the implementation:

The date class incapsulates the following values:

- Year; an integer between the year 1500 till today
- Month, an integer between 1 and 12
- Day, an integer between 1 and the number of days in the month.
- Error code; an integer which holds the code that corresponds to an error that recently happened or ZERO if the date object is valid and ready to be used.
- Current year; an integer that is automatically initialized to the current date of the system for validation purposes when a Date object is instantiated.

The Date module in files Date.h and Date.cpp is well documented and is placed in the project directory.

Study it and learn what each constant, variable and member function does and then using those function and your knowledge of iosteam, cin an cout add the following member functions to the Date class:

```
std::istream& read(std::istream& is = std::cin);
```

This function reads a date from console in following format YYYY/MM/DD as follows:

- Clear the error code by setting it NO_ERROR
- Read the year, the month and the day member variables using istream and ignore a single character after the year and the month values to bypass the Slashes

    Note that the separators do not have to be Slash characters "/" but any separator that is not an integer number.

- Check and see if istream has failed. If it did fail, set the error code to CIN_FAILED and clear the istream.
  If not, validate the values entered.
- Flush the keyboard
- Return the istream object

```cpp
std::ostream& write(std::ostream& os = std::cout)const;
```

If the Date object is in a "bad" state or (it is invalid) print the "**dateStatus()**".
otherwise the function should write the date in the following format using the ostream object:

- Prints the year
- Prints a Slash "/"
- Prints the month in two spaces, padding the left digit with zero, if the month is a single digit number
- Prints a Slash "/"
- Prints the day in two spaces, padding the left digit with zero, if the day is a single digit number
- Makes sure the padding is set back to spaces from zero
- Returns the ostream object.

Operator overloads: (do not use friend)

Overload the following comparison operators to compare two dates.

Use the return value of the **daysSince0001_1_1** member function to compare two dates:
```cpp
bool operator==
bool operator!=
bool operator>=
bool operator<=
bool operator<
bool operator>
```

Use the return value of the **daysSince0001_1_1** member function to overload the
```cpp
int operator-
```
to return the difference in number of days between two dates.
Example:
```cpp
Date
     D1(2019, 12, 02),
     D2(2019, 11, 11);
int days = D1 - D2;
```

"days: in the above code snippet will hold the value 21.

Bool cast overload:

Overload the Boolean cast so if a date is casted to bool, it will return true if the date is valid and false if it is not.

Helper operator overloads:

Overload the following helper operator overloads to have the Date class compatible with cin and cout, input and output operations:

```
operator<<  (for cout)
operator>>  (for cin)
```

DATE FUNCTIONS THAT ARE ALREADY IMPLEMENTED:

Private functions:

```
int daysSince0001_1_1()const; // returns number of days passed since the date 0001/1/1
bool validate();              /* validates the date setting the error code and then
                                 returning the result that is true, if valid, and
                                 false if invalid. */
void errCode(int theErrorCode); // sets the error code
int systemYear()const;          // returns the current system year
bool bad()const;             // return true if
int mdays()const;            // returns the number of days in current month
void setToToday();           // sets the date to the current date (system date)
```

Public Functions and constructors:

```
Date();                          // creates a date with current date
Date(int year, int mon, int day); /* creates a date with assigned values
                                     then validates the date and sets the
                                     error code accordingly */
int errCode()const;          // returns the error code or zero if date is valid
const char* dateStatus()const;  /* returns a string corresponding to the current status
                                   of the date */
int currentYear()const;         // returns the m_CUR_YEAR value;
```

DATE TESTER PROGRAM AND EXECUTION SAMPLE

Write your own tester or use **dateTester.cpp** to make sure your Date Module works correctly.

Compile your **Date** module with either **dateTester.cpp** or **dateSubmissionTester.cpp** for pre-submission test.

You should complete the coding for the Date module in two days.

For execution sample run any of the following commands on matrix:
~fardad.soleimanloo/244/ms1_date/dateTester
~fardad.soleimanloo/244/ms1_date/submissionTester

THE MENU MODULE

Create a module called Menu (in files Menu.cpp and Menu.h) this module will hold both MenuItem and Menu Classes' implementation code.

Forward declare the class Menu in the headerfile.

THE MENUITEM CLASS

Create a class Called MenuItem. This class is to hold only one C style string of characters for the description in the menu item. The length of the description is unknown.

This class should be fully private (no public members what so ever!).
Make the "Menu" class a friend of this class (which makes MenuItem class only accessible by the Menu class).

The description of the MenuItem is only to be set to a value at the moment of instantiation (or initialization) and is not changeable after the MenuItem is created.

If no value is provided for the description at the moment of creation, the MenuItem should be set as empty (with no description).

A MenuItem object can not be copied from or assigned to another MenuItem object.

When a MenuItem is casted to "bool" it should return true, if it is not empty and it should return false if it is empty.

Display the description of the MenuItem using a function that receives an ostream reference argument and returns it when printing is done. If no value is passed as argument to this function it should pass the "cout" object instead by default.

Make sure there is no memory leak after MenuItem goes out of scope.

THE MENU CLASS

Create a class called Menu.
A Menu Object can not be copied or assigned to another Menu Object.
This class has minimum of three member variables.
  1- A MenuItem to possibly hold the title of the Menu.
  2- An array of MenuItem pointers. The size of this array is set by a constant unsinged integer defined in the Menu header file; called `MAX_MENU_ITEMS`. Have the `MAX_MENU_ITEMS` integer initialized to 20.
     This array will keep potential MenuItems added to the Menu. Each individual element of this array will hold the address of a dynamically allocated MenuItem as they are added to the Menu. (See insertion operator overload for Menu)
  3- An integer to keep track of how many MenuItem pointers are pointing to allocated memories (obviously the value of this variable is always be between 0 and `MAX_MENU_ITEMS`).

CONSTRUCTORS, MEMBER VARIABLES AND OPERATOR OVERLOADS

The following are the list of Constructors, member function and operator overloads.

Make them private or public based on your own judgement.

Also make sure those member functions and operator overloads that do not change the Menu class are constant.

- A Menu is always created empty; with no MenuItems, with or without a title. Example:
  ```
  Menu A;
  Menu B("Lunch Menu");
  ```
- Create a function to display the title of the Menu.
- Create a function to display the entire Menu.
  This function first displays the title (if it is not empty) followed by a ":" and a newline, then it will display all the MenuItems one by one; adding a row number in front of each. The row numbers are printed in two spaces, right justified followed by a "dash" and a "space".
  After printing all MenuItems it should print `" 0- Exit"` and new line and `"> "`.
  Example:

  ```
  Lunch Menu:
   1- Omelet
   2- Tuna Sandwich
   3- California Rolls
   0- Exit
   >
  ```

- Overload the insertion operator (operator<<) to add a MenuItem to the Menu.
  This operator receives a C Style string containing the description of the MenuItem and return the reference of the Menu object itself.
  To accomplish this, check if a spot for a MenuItem is available in the array of MenuItem pointers. If it is, dynamically create a MenuItem out of the description received through the function argument and then store the address in the available spot and finally add to the number of allocated MenuItem pointers.
  If no spot is available, (that is; if number of allocated MenuItem pointers is equal to `MAX_MENU_ITEMS`) this function silently ignores the action.
  At the end, return the reference of the Menu object. Usage example:
  ```
  Menu M;
  M << "Omelet" << "Tuna Sandwich" << "California Rolls";
  ```

- Create a member function called `getSelection`. This function displays the Menu and gets the user selection (this function should be completely foolproof)
  The function receives nothing and returns an unsinged integer (That is the user's selection).
  After displaying the Menu, ask for an integer and make sure the value of the integer is between 0 and the number of the menu items. If the user enters anything incorrect, print:

"Invalid Selection, try again: "
and get the integer again until a valid selection is made.

- Overload `operator~` to do exactly what `getSelection` does.
- Casting the Menu to an integer or an unsigned integer should return the number of MenuItems allocated in the MenuItem array of pointers.
- Casting the Menu to "bool" returns true if the Menu has one or more menu items, otherwise it returns false.
- Overload the insertion operator to print the title of the Menu using cout.
- Example for last three overloads:

```
Menu M ("Lunch Menu");
M << "Omelet" << "Tuna Sandwich" << "California Rolls";
if (M) {
    cout << "The " << m1 << " is not empty and has "
         << unsigned int(M) << " menu items." << endl;
}
```

The above code snippet will print the following:

```
The Lunch Menu is not empty and has 3 menu items.
```

MENU TESTER PROGRAM AND EXECUTION SAMPLE

To be posted.

DUE DATE FOR MILESTONE 1

Monday November 18th, 2019

SUBMISSION INSTRUCTIONS

TBA