DATE CLASS

Design Document

Abstract

The Date Class includes the following overloaded operators: <<, >>, < , ++, --, and -.

Mickie Blair

Table of Contents

Table of Figures

Month Class

Description:

The program demonstrates overloaded functions of the Date Class. The Date Class consists three integer variables to hold the month number, the day and the year. The class constructors provide different ways of instantiating a Date class object. Prefix and postfix ++ and – operators are overloaded to demonstrated incrementing and decrementing the day. The -operator is overloaded to calculate the number of days between two dates. The class uses friends to overload the stream extraction and stream insertion operators.

Date Class:

The Date Class creates a date object.

Private Member Variables

Variable	Description
month	An integer that hold the month number
day	An integer that holds the day number
year	An integer that holds the year

Table 1 - Private Member Variables

Public Member Functions

Function	Description
class InvalidInput	Exception Class for Invalid Input
Constructor – No Args	Takes no arguments and sets the default date to 01/01/1900
Constructor – All Values	Takes 3 integer arguments to set the month, day, and year
Constructor – Date	Takes a pointer to a Date Object, and sets the month, day and year equal to the values in the Date object referenced by the Date pointer
determineMonthName (static)	A static method that takes an integer argument to determine the month name using a switch statement. The month name is returned to the calling function.
setMonth	Takes one integer argument to set the month number.
setDay	Takes one integer argument to set the day.
setYear	Takes one integer argument to set the year.
getMonth	Returns an integer of the month number.
getDay	Returns an integer of the day number.
getYear	Returns an integer of the year.
operator <	Takes a constant Date reference object and returns a Boolean indicating if the calling object is less than constant Date reference object passed as a parameter.

operator++ (prefix)	Takes no arguments and returns a pointer to date object that has incremented the Date object day using ++ prefix
operator++ (postfix)	Takes an int argument used as a dummy parameter and returns a date object that has incremented the Date object day using postfix++.
operator - - (prefix)	Takes no arguments and returns a pointer date object that has decremented the Date object day using - - prefix
operator - - (postfix)	Takes an int argument used as a dummy parameter and returns a date object that has decremented the Date object day using postfix--.
operator -	Takes a constant Date reference object and returns an integer indicating the number of days between the calling object and the constant Date reference object passed as a parameter.

Table 2 - Public Member Functions

Friend Functions of the Month Class

Function	Description
operator<<	Takes an ostream reference object and constant Date reference object and returns a reference to the month name determined using the month number, the day and the year in the specified format.
operator>>	Takes an istream reference object and Date reference object and returns a reference to the month number, the day and the year created by the user.

Table 3 - Friend Functions of Month Class

The Class Declaration

The class specification file (Date.h) contains the declaration statements for the variables and functions that are members are of the class.

```
#ifndef DATE_H
#define DATE_H

#include <iostream>
#include <string>

using namespace std;

//date class declaration
class Date
{
private:
	int month;			//integer variable for month
	int day;			//integer variable for day
	int year;			//integer variable for year

public:
	//Exception Class
	class InvalidInput{};

	//constructors
	Date();
	Date(int, int, int);
	Date(Date*);


	//determine month name static method
	static string determineMonthName(int);
```

```
        //setters
        void setMonth(int);
        void setDay(int);
        void setYear(int);

        //getters
        int getMonth();
        int getDay();
        int getYear();

        //operator overloads
        bool operator < (const Date&);
        Date operator++();
        Date operator++(int);
        Date operator--();
        Date operator--(int);
        int operator-(const Date&);

        //friends
        friend ostream& operator<<(ostream&, const Date&);
        friend istream& operator>>(istream&, Date&);
};

#endif
```

The Class Implementation

The class implementation file (Date.cpp) contains the definitions for the public functions of the Date class.

```
#include "Date.h"
#include <iostream>
#include <string>

using namespace std;

//default no args constructor
Date::Date()
{
        month = 1;
        day = 1;
        year = 1900;
}

//constructor
Date::Date(int m, int d, int y)
{
        month = m;
        day = d;
        year = y;
}

//copy constructor
Date::Date( Date* obj)

{
        month = obj->month;
        day = obj->day;
        year = obj->year;
```

```
}

//static function to determine Month Name
string Date::determineMonthName(int num)
{
	string name;   //variable to hold month name

	//switch for month name using int month
	switch (num)
	{
		case 1:
		{
			name = "January";
			break;
		}
		case 2:
		{
			name = "February";
			break;
		}
		case 3:
		{
			name = "March";
			break;
		}
		case 4:
		{
			name = "April";
			break;
		}
		case 5:
		{
			name = "May";
			break;
		}
		case 6:
		{
			name = "June";
			break;
		}
		case 7:
		{
			name = "July";
			break;
		}
		case 8:
		{
			name = "August";
			break;
		}



		case 9:
		{
			name = "September";
			break;
		}

		 case 10:
		{
			name = "October";
			break;
```

```
                }
                case 11:
                {
                        name = "November";
                        break;
                }
                case 12:
                {
                        name = "December";
                        break;
                }
        }

        return name;
}

//setters which throw invalid input if value out of range
void Date::setMonth(int m)
{
        if (m < 1 || m > 12)
                {
                        cout << "\nMonth is invalid.\n" << endl;
                        throw InvalidInput();
                }
        else
                {
                        month = m;
                }
}

void Date::setDay(int d)
{
        switch (month)
                {
                case 1:
                case 3:
                case 5:
                case 7:
                case 8:
                case 10:
                case 12:
                {
                        if (d < 1 || d > 31)
                        {
                                cout << "\nDay must be between 1 and 31 for " <<
                                determineMonthName(month)<< ".\n" << endl;

                               throw InvalidInput();
                        }

                        else
                        {
                                day = d;
                        }
                break;
                }

                case 4:
                case 6:
                case 9:
                case 11:
                {
                        if (d < 1 || d > 30)
```

```
                        {
                                cout << "\nDay must be between 1 and 30 for " <<
                                determineMonthName(month) << ".\n" << endl;

                             throw InvalidInput();
                        }

                        else
                        {
                                day = d;
                        }
                break;
                }

                case 2:
                {
                        if (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0))
                        {
                                if (d < 1 || d > 29)
                                {
                                        cout << "\nFor a Leap Year, February's days must be
                                        between 1 and 29.\n" << endl;

                                     throw InvalidInput();
                                }

                                else
                                {
                                        day = d;
                                }
                        }

                        else
                        {
                                if (d < 1 || d > 28)
                                {
                                         cout << "\nFor a Non-Leap Year, February's days must be
                                         between 1 and 28.\n" << endl;

                                    throw InvalidInput();
                                }

                                else
                                {
                                        day = d;
                                }
                        }
                break;
                }
        }
}

void Date::setYear(int y)
{
        if (y < 0)
                {
                        cout << "\nYear must only contain numbers.\n\n";
                        throw InvalidInput();
                }
        else
                {
                        year = y;
                }
```

```
}

//getters
int Date::getMonth()
{
	return month;
}

int Date::getDay()
{
	return day;
}

int Date::getYear()
{
	return year;
}

//less than operator overload for date
bool Date::operator<(const Date& right)
{
	bool isLessThan = false;

	if (year < right.year)
		{
			isLessThan = true;
		}
	else if (year == right.year)
		{
			if (month< right.month)
				{
					isLessThan = true;
				}

			else if (month == right.month)
			{
				if (day< right.day)
				{
					isLessThan = true;
				}
			}
		}

	return isLessThan;
}

//Overloaded prefix ++ operator
Date Date::operator++()
{
	++day;

	switch (month)
	{
		case 1:
		case 3:
		case 5:
		case 7:
		case 8:
		case 10:
			{
				if (day == 32)
				{
					day = 1;
```

```
                                        month++;
                                }
                        break;
                        }

                case 4:
                case 6:
                case 9:
                case 11:
                        {
                                if (day == 31)
                                {
                                        day = 1;
                                        month++;
                                }
                        break;
                        }

                case 12:
                        {
                                if (day == 32)
                                {
                                        day = 1;
                                        month= 1;
                                        year++;
                                }
                        break;
                        }

                case 2:
                        {
                        if (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0))
                                {
                                        if (day == 30)
                                                {
                                                        day = 1;
                                                        month++;
                                                }
                                }

                        else
                                {
                                        if (day == 29)
                                                {
                                                        day = 1;
                                                        month++;
                                                }
                                }
                        break;
                        }
        }

        return *this;
}

//Overloaded postfix ++ operator
Date Date::operator++(int)
{
        Date temp(month, day, year);

        day++;

        switch (month)
```

```
	{
		case 1:
		case 3:
		case 5:
		case 7:
		case 8:
		case 10:
		{
			if (day == 32)
			{
				day = 1;
				month++;
			}
		break;
		}

		case 4:
		case 6:
		case 9:
		case 11:

		{
			if (day == 31)
			{
				day = 1;
				month++;
			}
		break;
		}

		case 12:
		{
			if (day == 32)
			{
				day = 1;
				month = 1;
				year++;
			}
		break;
		}

		case 2:
		{
			if (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0))
			{
				if (day == 30)
				{
					day = 1;
					month++;
				}
			}

			else
			{
				if (day == 29)
				{
					day = 1;
					month++;
				}
			}
		break;
		}
```

```
        }

        return temp;
}

//Overloaded prefix -- operator
Date Date::operator--()
{
        --day;

        switch (month)
        {
                case 1:
                        {
                          if (day == 0)
                          {
                                day = 31;
                                month = 12;
                                year--;
                           }
                        break;
                        }

                case 3:
                        {
                          if (day == 0)
                          {
                             month--;

                             if (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0))
                               {
                                  day = 29;
                               }

                             else
                               {
                                  day = 28;
                               }
                           }
                        break;
                        }

                case 5:
                case 7:
                case 10:
                case 12:
                        {
                                if (day == 0)
                                {
                                        day = 30;
                                        month--;
                                }
                        break;
                        }

                case 2:
                case 8:
                case 4:
                case 6:
                case 9:
                case 11:
                        {
                                if (day == 0)
```

```
                                {
                                        day = 31;
                                        month--;
                                }
                        break;
                        }
        }

        return *this;
}

//Overloaded postfix -- operator
Date Date::operator--(int)
{
        Date temp(month, day, year);

        --day;

        switch (month)
        {
                case 1:
                {
                        if (day == 0)
                        {
                                day = 31;
                                month = 12;
                                year--;
                        }
                break;
                }

                case 3:
                {
                        if (day == 0)
                        {
                           month--;

                           if (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0))
                                {

                                        day = 29;
                                }

                                else
                                {
                                        day = 28;
                                }
                        }
                break;
                }

                case 5:
                case 7:
                case 10:
                case 12:
                {
                        if (day == 0)
                        {
                                day = 30;
                                month--;
                        }
                break;
                }
```

```
                case 2:
                case 8:
                case 4:
                case 6:
                case 9:
                case 11:
                {
                        if (day == 0)
                        {
                                day = 31;
                                month--;
                        }
                break;
                }

        }

        return temp;
}

//overloaded - operator
int Date::operator-(const Date& secondDate)
{
        //declared start and end date
        Date start;
        Date end;

        //declare days difference accumulator
        int daysDiff = 0;

        //set the start and end date
        if (*this < secondDate)
                {
                        start = *this;
                        end = secondDate;
                }
        else
                {
                        start = secondDate;
                        end = *this;
                }

        //increment the day and the days difference while the start day
        // is less than the end day

        while (start < end) {
                start++;
                daysDiff++;
        }

        //return days difference
        return daysDiff;
}

//overload the cout's <<
ostream& operator<<(ostream& strm, const Date& obj)
{
        //create strm of month (using static determineMonthName), day and year
        strm << Date::determineMonthName(obj.month) << " "
                 <<  obj.day << ", " << obj.year;

        return strm;
```

```
}

//overload the cin's >>
istream& operator >> (istream& strm, Date& obj)
{
	string input;				//variable for string input from user
	bool isValid = false;		//Boolean for valid input from user
	string mm;					//string of month only
	string dd;					//string of day only
	string yyyy;				//string of year only
	int m;						//month after converting to integer
	int d;						//day after converting to integer
	int y;						//year after converting to integer

	//input validation
	while (!isValid)
	{
		cout << "Enter date (ex. 01/01/2018): ";

		getline(cin, input);

		try
		{
		 //check format of input
		 if (input.length() != 10 || input[2] != '/' && (input[5] != '/'))

		    {
		       //if not valid display message and throw invalid input
		        cout << "\nDate Format is Invalid. (Valid Format: mm/dd/yyyy)\n"
		        << endl;
		       throw Date::InvalidInput();
		     }

		else
		{
		  //assign substrings of input to individual string variables
		       mm.assign(input, 0, 2);
		       dd.assign(input, 3, 2);
		       yyyy.assign(input, 6, 4);

		  //check for integers only
		       for (int index = 0; index < int(mm.length()); index++)
		        {
		            if (!isdigit(mm[index]))
		              {
		                 cout << "\nMonth must be numbers only.\n" << endl;

		                 throw Date::InvalidInput();
		              }
		         }

		       for (int index = 0; index < int(dd.length()); index++)
		         {
		              if (!isdigit(dd[index]))
		                {
		                   cout << "\nDay must be numbers only.\n" << endl;

		                   throw Date::InvalidInput();
		                }
		          }


		        for (int index = 0; index < int(yyyy.length()); index++)
```

```
                        {
                            if (!isdigit(yyyy[index]))
                              {
                                cout << "\nYear must be numbers only.\n" << endl;

                                throw Date::InvalidInput();
                               }
                         }

                 //convert to int can throw an illegal_argument exception
                   y = (stoi(yyyy));
                   obj.setYear(y);

                   m = (stoi(mm));
                   obj.setMonth(m);

                   d = (stoi(dd));
                   obj.setDay(d);

                   isValid = true;
                 }
                }

                catch (invalid_argument)
                {
                  //if caught change isValid to false and display message
                   isValid = false;
                    cout << "\nInvalid Date Format. Date must be numbers separated by /.\n" <<
                    endl;
                }

                catch (Date::InvalidInput)
                {
                //if caught change isValid to false(specific message supplied by setters)
                isValid = false;
                }

        }

        return strm;
}
```

Public Functions of the Month Class

InvalidInput Class

The InvalidInput creates an instance of the InvalidInput Exception.

Function call example.

```
throw InvalidInput();
```

Constructor-No Args Function

The Constructor function creates a Date object with no arguments received from main. The month number is set to 1, the day is set to 1, and the year is set to 1900.

Function call example.

```
Date date1;
```

Constructor-All Values Function

The Constructor function creates a Month object with 3 arguments from the calling function.

Function call example.

```
Date* date = new Date(3, 3, 2000);
```

Constructor-Date Function

The Constructor function creates a Date object with a pointer to a Date Object. It sets the month, day and year equal to the values in the Date object referenced by the Date pointer.

Function call example.

```
Date end;
end = secondDate;
```

Pseudocode

```
Date end month = secondDate month;
Date end day = secondDate day;
Date end year = secondDate year;
```

determineMonthName Function

A static method that takes an integer argument to determine the month name using a switch statement. The month name is returned to the calling function.

Function call example.

```
strm << Date::determineMonthName(obj.month) << " " <<  obj.day << ", " << obj.year;
```

Pseudocode

```
switch (obj.month)

   obj.month 1 : name = “January”

   obj.month 2 : name = “February”

   obj.month 3 : name = “March”

   obj.month 4 : name = “April”

   obj.month 5 : name = “May”

   obj.month 6 : name = “June”

   obj.month 7 : name = “July”

   obj.month 8 : name = “August”

   obj.month 9 : name = “September”

   obj.month 10: name = “October”

   obj.month 11: name = “November”

   obj.month 12: name = “December”
```

setMonth Member Function

Takes one integer to set the month number. The function can throw an InvalidInput Exception which indicates to the calling function that the input received from the user is invalid.

Function call example.

```
obj.setMonth(m);
```

Pseudocode

```
if month < 1 or month > 12
    throw exception
else
    month = m;
```

```
Enter date (ex. 01/01/2018): 15/30/2000

Month is invalid.
```

setDay Member Function

Takes one integer to set the day. The function can throw an InvalidInput Exception which indicates to the calling function that the input received from the user is invalid.

Function call example.

```
obj.setDay(d);
```

Pseudocode

```
if month is 1, 2, 5, 7, 8, 10 or 12
      if day < 1 or day > 31
          throw InvalidInput
      else
          day = d
if month is 4, 6, 9, or 11
      if day < 1 or day > 30
          throw InvalidInput
      else
          day = d

if month is 2
      if year is a leap year
          if day < 1 or day > 29
             throw InvalidInput
          else
             day = d

          throw InvalidInput
      else
          if day < 1 or day > 28
             throw InvalidInput
          else
             day = d
```

```
Enter date (ex. 01/01/2018): 01/45/2000

Day must be between 1 and 31 for January.
```

setYear Member Function

Takes one integer to set the year. The function can throw an InvalidInput Exception which indicates to the calling function that the input received from the user is invalid.

Function call example.

```
obj.setYear(y);
```

Pseudocode

```
if year < 0
   throw InvalidInput
else
   year = y
```

```
Enter date (ex. 01/01/2018): 12/21/00/0

Year must be numbers only.
```

getMonth Member Function

Returns an integer of the month.

getDay Member Function

Returns an integer of the day.

getYear Member Function

Returns an integer of the year.

operator < Overloaded Function for is equal to

Takes a constant Date reference object and returns a Boolean indicating if the calling object is less than constant Date reference object passed as a parameter.

Function call example.

```
while (start < end)
       {
              start++;
              daysDiff++;
       }
```

Pseudocode

```
if start year < end year
     then start Date is less than end Date

else if start year and end year are equal
     if start month is less than end month
        then start Date is less than end Date

     else if start month is equal to end month
          if start day is less than end day
              then start Date is less than end Date
```

operator++ Overloaded Function for Prefix

Takes no arguments and returns a date object that has incremented the Date object day using ++ prefix It also increments the month when the last day of the month is reached. At the end of the year, the year is also incremented. It returns a pointer to the incremented date object.

Pseudocode

```
add one to the day

if day increase past the last day possible in the month
    make day 1 and add one to month

    if adding one to month will make the month 12
    make month 1 and increment the year

return a pointer to the object
```

operator++ Overloaded Function for Postfix

Takes an int argument used as a dummy parameter and returns a date object that has incremented the Date object day using postfix++. It also increments the month when the last day of the month is reached. At the end of the year, the year is also incremented.

Pseudocode

```
create temporary date object with the month number, day, and year of the calling object

add one to the day

if day increase past the last day possible in the month
    make day 1 and add one to month

    if adding one to month will make the month 12
    make month 1 and increment the year

return the temporary month object
```

operator- - Overloaded Function for Prefix

Takes no arguments and returns a date object that has decremented the Date object day using -- prefix It also decrements the month when the first day of the month is reached. At the start of the year, the year is also decremented. It returns a pointer to the decremented date.

Pseudocode

```
subtract one from the day

if month is 1 and day now equals 0
    make month 12 and set day to 31

if month is 3 and day now equals 0
    if leap year
       subtract one from month and make day 29
    else
        subtract one from month and make day 29

if month is 5, 7, 10, or 12
   subtract one from month and make day 30
```

```
if month is 2, 8, 4, 6, 9 or 11
   subtract one from month and make day 31

return a pointer to the object
```

operator- - Overloaded Function for Postfix

Takes no arguments and returns a date object that has decremented the Date object day using postfix -- It also decrements the month when the first day of the month is reached. At the start of the year, the year is also decremented. It returns a pointer to the decremented date.

Pseudocode

```
create temporary date object with the month number, day, and year of the calling object

subtract one from the day

if month is 1 and day now equals 0
    make month 12 and set day to 31

if month is 3 and day now equals 0
    if leap year
       subtract one from month and make day 29
    else
        subtract one from month and make day 29

if month is 5, 7, 10, or 12
   subtract one from month and make day 30

if month is 2, 8, 4, 6, 9 or 11
   subtract one from month and make day 31

return the temporary date object
```

operator- Overloaded Function for Date Subtraction

Takes a constant Date reference object and returns an integer indicating the number of days between the calling object and the constant Date reference object passed as a parameter.

Function call example.

```
cout << "\nDays between " << date1 << " and " << date2
                << ":  " << date1 - date2 << endl;
```

Pseudocode

```
create date object named start
create date object named end
create days difference = 0;

if calling object is less than the constant Date reference object parameter
   set start date to calling date
   set end date to constant Date reference object
else
   set start date to constant Date reference object
   set end date to calling date

while start date is less than end day
```

```
     add increment start
     add one to days difference

return days difference
```

Friend Functions of the Month Class

operator << Overloaded Function

Takes an ostream reference object and constant Date reference object and returns a reference to the month name determined using the month number, the day and the year in the specified format. This allows cout to display the private member variables values without using the dot operator.

Format of output

```
January 1, 2000
```

Function call example.

```
cout << date1 << endl;
```

Pseudocode

```
output is
determineMonthName(using date1 month), date1 day, date1 year
```

operator >> Overloaded Function

Takes an istream reference object and Date reference object and returns a reference to the month number, the day and the year created by the user. This allows cin to set the values of the private member functions without using the dot operator. Inside the function the input is validated by the set member functions which can throw and InvalidInput Exception. The members of the object are not return until all members are valid.

Function call example.

```
cin >> date1;
```

Pseudocode

```
declare string for user input;

declare and initialize Boolean isValid = false;

declare string variables to hold month, day and year;

declare int variables to hold month, day and year;

while isValid is false
{
  Ask user for a date
  get input line from user

  try
  {
     if length is not 10 or if string does not have / at position 2 and 5
         throw InvalidInput exception
     else
```

```
        assign first two characters to month string
        assign characters 4 and 5 to day string
        assign characters 7, 8, 9 and 10 to year string

              check month string, day string, and year string
                   if strings contain anything other than digits
                       throw InvalidInput

     convert year string to integer
            if not successful throw illegal_argument exception
            else setYear
     convert month string to integer
            if not successful throw illegal_argument exception
            else setMonth
     convert day string to integer
            if not successful throw illegal_argument exception
            else setDay

      set isValid = true;

  }

  catch exceptions
  {
      set isValid = false;
  }

}

if isValid is true

return stream object;
```

```
Enter date (ex. 01/01/2018): aaaaaaa

Date Format is Invalid. (Valid Format: mm/dd/yyyy)

Enter date (ex. 01/01/2018): aa/aa/aaaa

Month must be numbers only.

Enter date (ex. 01/01/2018): 01/aa/aaaa

Day must be numbers only.

Enter date (ex. 01/01/2018): 01/01/aaaa

Year must be numbers only.
```

Programming Strategy:

In this program, the Date class functions will be demonstrated using input from the user. The program is broken down int several functions accessed by a menu utilizing a switch statement. The functions to demonstrate the program are called by functions outside of the main function. The Date object members are set using the overloaded cin operator(<<). The results of each test are displayed using the overloaded cout operator(<<). The program will allow the user to choose multiple menu options during the program run. The program will also allow the user to run the program again if they choose. The Program includes the header files Date.h and Prototypes.h.

Program Functions

Function	Description
main	The program's main function. It calls the programs other functions from inside a try block to catch insufficient memory exceptions.
repeatProgram	Asks the user if they would like to run the program again. User input is validated. Calls the showIntro and repeatMenu Functions .
showIntro	Displays an introduction to the program. Takes no arguments.
repeatMenu	Asks the user if they would like to choose another menu option. User input is validated.
dateClass	Contains a switch statement to execute menu functions after getMenuOption function has been called.
getMenuOption	Asks the user for a menu option to execute. User input is validated. Returns the menu Option to the calling function.
calculateDays	Calculates the number of days between two dates
demoIncrementOperator	Demonstrates prefix and postfix ++
demoDecrementOperator	Demonstrates prefix and postfix - -
getNumberOfTimes	Function to get the number of times the user would like to increment or decrement the date. Can throw InvalidInput, out_of_range, and invalid_argument exceptions

Table 4 - Program Functions

main Function

The main function will call the function to repeat the program if sufficient memory exists.

Pseudocode for main

```
try
	{
		repeating the program

	}

catch (bad_alloc)
	{
		error insufficient memory
	}
```

repeatProgram Function

The function uses a do while loop to ask the user if they would like to run the program again. The user input is validated. Inside the do while loop the showIntro function and repeatMenu functions are called. The only variable is a string to hold the user input to continue or exit. It takes no arguments.

.Pseudocode

```
do
{
    showIntro

    repeatMenu

    input do again
     {
        validate do again
     }
}
while user want to continue
```

```
Would you like to run the program again? (Enter Y or N) k

Invalid entry:
Would you like to run the program again? (Enter Y or N) Y
```

showIntro Function

The showIntro Function displays a brief description of the program.

```
Date Class with Modifications

The user will be asked to enter a date using the
overloaded cin stream extraction operator (>>).

The program will demonstrate the following:

 - overloaded subtraction (-) operator for calculating the
   the number of days between two dates.
 - overloaded increment operator (++) prefix and postfix
 - overloaded decrement operator (--) prefix and postfix.

The results will be displayed using an overloaded
cout stream insertion operator (<<).

The program will implement exceptions and input validation.
```

repeatMenu Function

The function uses a do while loop to ask the user if they would like to choose another option from the menu. The user input is validated. Inside the do while loop the dateClass function is called. The only variable is a string to hold the user input to continue or exit. It takes no arguments.

.Pseudocode

```
do
{
    dateClass

    input do again
     {
        validate do again
     }
}
while user want to continue
```

```
Would you like to run the program again? (Enter Y or N) k

Invalid entry:
Would you like to run the program again? (Enter Y or N) Y
```

dateClass Function

Contains a switch statement to execute menu functions after getMenuOption function has been called.

Pseudocode

```
get menu option

option 1 – calculate days

option 2 – demonstrate ++ operator

option 2 – demonstrate -- operator
```

getMenuOption Function

Asks the user for a menu option to execute. User input is validated. Returns the menu Option to the calling function.

Pseudocode

```
get menu option

option 1 – calculate days

option 2 – demonstrate ++ operator

option 2 – demonstrate – operator

while option is not valid ask again

return option
```

```
Menu Options

1. Calculate Days Between Dates (- operator)
2. Demonstrate Prefix and PostFix Increment Operator (++)
3. Demonstrate Prefix and PostFix Decrement Operator (--)

Please Enter a Menu Option: 0

Invalid entry:

1. Calculate Days Between Dates (- operator)
2. Demonstrate Prefix and PostFix Increment Operator (++)
3. Demonstrate Prefix and PostFix Decrement Operator (--)

Please Enter a Menu Option: (Enter 1, 2, 3)
```

calculateDays Function

The calculateDays function calculates the number of days between two dates. The dates are entered using the overloaded cin which validates the data before the object is created. The number of days between the two date is calculated using the overloaded operator (-). Inside the overloaded minus operator, the overloaded less than (<) operator is utilized.

Pseudocode

```
Ask user for two dates

Display the result of date1 – date2
```

```
Days Between Dates

Please enter two dates to calculate the days between.

Enter date (ex. 01/01/2018): 03/09/1977

Enter date (ex. 01/01/2018): 10/13/2019

Days between March 9, 1977 and October 13, 2019:  15558
```

demoIncrementOperator Function

The function demonstrates the increment operator (++) for both prefix and postfix. The program asks the user for the number of times they would like to increment the Date object using the getNumberOfTimes to validate the input. The date is entered using the overloaded cin which validates the data before the object is created. A for loop is utilized to display the results.

Pseudocode

```
create Date Object.

ask user the number of times they would like to increment Date

ask user for the date

set date to a prefix Date object
set date to a postfix Date object

loop through prefix Date object displaying results
loop through postfix Date object displaying results
```

```
Demonstration of Prefix And Postfix Increment Operator (++)

How many times would you like to increment the date?

Enter a whole positive number: 5

Enter date (ex. 01/01/2018): 07/18/2015

The date July 18, 2015 will be incremented 5 times using ++date.
July 19, 2015
July 20, 2015
July 21, 2015
July 22, 2015
July 23, 2015

The date July 18, 2015 will be incremented 5 times using date++.
July 18, 2015
July 19, 2015
July 20, 2015
July 21, 2015
July 22, 2015
```

demoDecrementOperator Function

The function demonstrates the decrement operator (--) for both prefix and postfix. The program asks the user for the number of times they would like to increment the Date object using the getNumberOfTimes to validate the input. The date is entered using the overloaded cin which validates the data before the object is created. A for loop is utilized to display the results.

Pseudocode

```
create Date Object.

ask user the number of times they would like to decrement Date

ask user for the date

set date to a prefix Date object
set date to a postfix Date object

loop through prefix Date object displaying results
loop through postfix Date object displaying results
```

```
Demonstration of Prefix And Postfix Increment Operator (--)

How many times would you like to decrement the date?

Enter a whole positive number: 6

Enter date (ex. 01/01/2018): 03/02/2000

The date March 2, 2000 will be decremented 6 times using --date.
March 1, 2000
February 29, 2000
February 28, 2000
February 27, 2000
February 26, 2000
February 25, 2000

The date March 2, 2000 will be decremented 6 times using date--.
March 2, 2000
March 1, 2000
February 29, 2000
February 28, 2000
February 27, 2000
February 26, 2000
```

getNumberOfTimes Function

The function gets the number of times the user would like to increment or decrement the date. The function can throw InvalidInput, out_of_range, and invalid_argument exceptions which are handled using a try/catch block and a Boolean isValid.

Pseudocode

```
string input;
int num;
Boolean isValid = false;

while isValid is false
{
   Ask user to enter a whole number

   try
    {
      loop through input if any character is not a digit
         throw InvalidInput

      convert string input to int number
         if not successful throw invalid argument

      check if number is zero or greater

      set isValid to true
    }

   catch exceptions
    {
      display message
      set isValid false;
    }

return number
```

```
How many times would you like to increment the date?

Enter a whole positive number: aaaaaa

Input must be whole numbers only and greater than 0.

Enter a whole positive number: 1.1

Input must be whole numbers only and greater than 0.

Enter a whole positive number: 54654654654654564

Number is too large, please enter a smaller number.

Enter a whole positive number: -000

Input must be whole numbers only and greater than 0.

Enter a whole positive number:
```

OUTPUT

```
Date Class with Modifications

The user will be asked to enter a date using the
overloaded cin stream extraction operator (>>).

The program will demonstrate the following:

 - overloaded subtraction (-) operator for calculating the
   the number of days between two dates.
 - overloaded increment operator (++) prefix and postfix
 - overloaded decrement operator (--) prefix and postfix.

The results will be displayed using an overloaded
cout stream insertion operator (<<).

The program will implement exceptions and input validation.

Menu Options

1. Calculate Days Between Dates (- operator)
2. Demonstrate Prefix and PostFix Increment Operator (++)
3. Demonstrate Prefix and PostFix Decrement Operator (--)

Please Enter a Menu Option:
```

```
Days Between Dates

Please enter two dates to calculate the days between.

Enter date (ex. 01/01/2018): 05/31/1995

Enter date (ex. 01/01/2018): 10/13/2019

Days between May 31, 1995 and October 13, 2019:  8901

Would you like to choose from the menu again? (Enter Y or N)
```

```
Demonstration of Prefix And Postfix Increment Operator (++)

How many times would you like to increment the date?

Enter a whole positive number: 8

Enter date (ex. 01/01/2018): 02/21/2000

The date February 21, 2000 will be incremented 8 times using ++date.
February 22, 2000
February 23, 2000
February 24, 2000
February 25, 2000
February 26, 2000
February 27, 2000
February 28, 2000
February 29, 2000

The date February 21, 2000 will be incremented 8 times using date++.
February 21, 2000
February 22, 2000
February 23, 2000
February 24, 2000
February 25, 2000
February 26, 2000
February 27, 2000
February 28, 2000

Would you like to choose from the menu again? (Enter Y or N)
```

```
Demonstration of Prefix And Postfix Increment Operator (--)

How many times would you like to decrement the date?

Enter a whole positive number: 6

Enter date (ex. 01/01/2018): 11/10/2015

The date November 10, 2015 will be decremented 6 times using --date.
November 9, 2015
November 8, 2015
November 7, 2015
November 6, 2015
November 5, 2015
November 4, 2015

The date November 10, 2015 will be decremented 6 times using date--.
November 10, 2015
November 9, 2015
November 8, 2015
November 7, 2015
November 6, 2015
November 5, 2015

Would you like to choose from the menu again? (Enter Y or N)
```

```
Would you like to run the program again? (Enter Y or N)
```

```
C:\Users\blair\source\repos\MidtermDateClassModification\Debug\MidtermDateClassModification.
exe (process 16536) exited with code 0.
Press any key to close this window . . .
```

demonstrate

The Program Prototypes Header File

```
//Prototypes Header file
#ifndef PROTOTYPES_H
#define PROTOTYPES_H

void showIntro();
void repeatProgram();
void repeatMenu();
void dateClass();
int getMenuOption();
void calculateDays();
void demoIncrementOperator();
void demoDecrementOperator();
int getNumberOfTimes();

//Exception Class
class InvalidInput {};

#endif
```

The Entire Program

```
//Mickie Blair
//Midterm Project
//Date Class Demo Program

#include "Prototypes.h"
#include "Date.h"

#include <iostream>
#include <string>
#include <iomanip>
#include <cstring>
#include <new>

using namespace std;

//main function
int main() {
	try
	{
		//repeat the program
		repeatProgram();

		//display complete message
		cout << "Program run complete.\n";

	}
	catch (bad_alloc)
	{
		cout << "Insufficient Memory.\n";
	}


	return 0;
```

```
}

//show Intro
void showIntro() {
	string key;
	//display introduction
	cout << "Date Class with Modifications\n" << endl;
	cout << "The user will be asked to enter a date using the\n"
		<< "overloaded cin stream extraction operator (>>).\n\n"
		<< "The program will demonstrate the following:\n\n"
		<< " - overloaded subtraction (-) operator for calculating the\n"
		<< "   the number of days between two dates.\n"
		<< " - overloaded increment operator (++) prefix and postfix\n"
		<< " - overloaded decrement operator (--) prefix and postfix.\n\n"
		<< "The results will be displayed using an overloaded\n"
		<< "cout stream insertion operator (<<).\n\n"
		<< "The program will implement exceptions and input validation.\n" << endl;

}

//repeat Program
void repeatProgram() {

	string again;				//variable for again loop

	//do while loop
	do
	{
		//display introduction
		showIntro();

		//repeat menu
		repeatMenu();

		//ask user if they would like to go again
		cout << "Would you like to run the program again? (Enter Y or N) ";
		getline(cin, again);

		//validate entry
		while (again != "Y" && again != "N" && again != "y" && again != "n")
		{
			cout << "\nInvalid entry:" << endl;
			cout << "Would you like to run the program again? (Enter Y or N) ";
			getline(cin, again);
		}

		cout << endl;

		//clear
		system("CLS");
	}

	while (again == "Y" || again == "y");
}

//repeat Menu
void repeatMenu() {

	string again;				//variable for again loop
```

```
	//do while loop
	do
	{
		//demonstrate the month class with overloads
		dateClass();

		//ask user if they would like to choose from the menu again
		cout << "\nWould you like to choose from the menu again? (Enter Y or N) ";
		getline(cin, again);

		//validate entry
		while (again != "Y" && again != "N" && again != "y" && again != "n")
		{
		   cout << "\nInvalid entry:" << endl;
		   cout << "Would you like to choose from the menu again? (Enter Y or N) ";
		   getline(cin, again);
		}

		//clear the screen
		system("CLS");
	}

	while (again == "Y" || again == "y");



}

void dateClass() {

	int option;		//char for option

		//get menu option
		option = getMenuOption();

		//switch menu
		switch (option)
		{
			case 1:
				{
					//calculate days
					calculateDays();

					break;
				}

			case 2:
				{
					//increment operator
					demoIncrementOperator();
					break;
				}

			case 3:
				{
					//demonstrate decrement
					demoDecrementOperator();
```

```
                                        break;
                                }
                        }
        }

//get menu options with input validation
int getMenuOption()
{
        string input;        // string for input

        cout << "Menu Options\n\n"
                << "1. Calculate Days Between Dates (- operator)\n"
                << "2. Demonstrate Prefix and PostFix Increment Operator (++)\n"
                << "3. Demonstrate Prefix and PostFix Decrement Operator (--)\n"
                << endl;

        cout << "Please Enter a Menu Option: ";
        getline(cin, input);

        while (input != "1" && input != "2" && input != "3")
        {
                cout << "\nInvalid entry:\n" << endl;
                cout << "1. Calculate Days Between Dates (- operator)\n"
                        << "2. Demonstrate Prefix and PostFix Increment Operator (++)\n"
                        << "3. Demonstrate Prefix and PostFix Decrement Operator (--)\n"
                        << endl;
                cout << "Please Enter a Menu Option: (Enter 1, 2, 3) ";
                getline(cin, input);
        }

        //clear the screen
        system("CLS");

        //return choices as an integer
        return stoi(input);
}

//calculate days
void calculateDays() {
        Date date1;          //date object to hold date 1
        Date date2;          //date object to hold date 2

        //header
        cout << "Days Between Dates\n" << endl;
        //ask user for two dates
        cout << "Please enter two dates to calculate the days between.\n\n";
        cin >> date1;

        cout << endl;

        cin >> date2;

        //display results
        cout << "\nDays between " << date1 << " and " << date2
                << ":  " << date1 - date2 << endl;

}

//demonstrate ++ operator (prefix and post fix)
```

```
void demoIncrementOperator() {
	int times;		// integer for times to increment

    //create default month object
	Date overloadedCin;

	//header
	cout << "Demonstration of Prefix And Postfix Increment Operator (++)\n\n";

	//ask user for a number
	cout << "How many times would you like to increment the date?\n\n";
	times = getNumberOfTimes();

	cout << endl;
	//ask user for a date
	cin >> overloadedCin;

	//set dates to new variables
	Date preInc = overloadedCin;
	Date postInc = overloadedCin;

	//preincrement
	cout << "\nThe date " << preInc << " will be incremented " << times << " times
	using ++date." << endl;

	//loop through number of times
	for (int index = 0; index < times; index++)
	{
		cout  << ++preInc << endl;
	}

	//post increment
	cout << "\nThe date " << postInc << " will be incremented " << times << " times
	using date++." << endl;

	//loop through number of times
	for (int index = 0; index < times; index++)
	{
		cout << postInc++ << endl;
	}
}

//demonstrate ++ operator (prefix and post fix)
void demoDecrementOperator() {
	int times;		// integer for times to increment

	//create default month object
	Date overloadedCin;

	//header
	cout << "Demonstration of Prefix And Postfix Increment Operator (--)\n\n";

	//ask user for a number
	cout << "How many times would you like to decrement the date?\n\n";
	times = getNumberOfTimes();

	cout << endl;
	//ask user for a date
	cin >> overloadedCin;
```

```
	//set dates to new variables
	Date preDe = overloadedCin;
	Date postDe = overloadedCin;

	//preincrement
	cout << "\nThe date " << preDe << " will be decremented " << times << " times
	using --date." << endl;

	//loop through number of times
	for (int index = 0; index < times; index++)
	{
		cout << --preDe << endl;
	}

	//post increment
	cout << "\nThe date " << postDe << " will be decremented " << times << " times
	using date--." << endl;

	//loop through number of times
	for (int index = 0; index < times; index++)
	{
		cout << postDe-- << endl;
	}
}

int getNumberOfTimes() {
	string input;		//variable for string input from user
	int num;		//variable for times
	bool isValid = false;	//Boolean for valid input from user

	//input validation
	while (!isValid)
	{

		cout << "Enter a whole positive number: ";

		getline(cin, input);

		//try catch - catches illegal argument, out of range, and Invalid input
		try
		{
			for (int index = 0; index < int(input.length()); index++)
			{
				if (!isdigit(input[index]))
					throw InvalidInput();
			}

			//try to convert to integer
			num = (stoi(input));

			//check for positive and greater than one
			if (num <= 0)
			{
				throw InvalidInput();
			}

			isValid = true;
		}
```

```
            catch (invalid_argument)
            {
             isValid = false;
             cout << "\nInvalid Entry. Input must be whole numbers only.\n" << endl;
            }

            catch (out_of_range)
            {
             isValid = false;
             cout << "\nNumber is too large, please enter a smaller number.\n" << endl;
            }


            catch (InvalidInput)
            {
            isValid = false;
            cout << "\nInput must be whole numbers only and greater than 0.\n" << endl;
            }

        }

        //return number
        return num;
}
```