1. **Program Statement**

This program gets user’s birthdate in a specific format and uses it to call function that does exception handling. It uses a nested try-catch blocks to print appropriate error message for the error in the internal try-catch block(s) and prints out the converted date at the end of program.

1. **Requirements**
   1. **Assumptions**
      1. User enters “0” before the value if month and day is less than the number 10
      2. User enters dashes once in the middle of day, month, and year
      3. User only enters integer values
      4. User knows that some months do not have 31 days
   2. **Specifications**
      1. Get string input from the user that stores the full birth date in the following format
         1. Mm-dd-yyyy
      2. Convert and store values for month, day, and year
      3. Exception handling using try-block, must be done in a class
         1. Day can only be between 1 and 31
         2. Month can only be between 1 and 12
         3. Year can only be between 1915 and 2017
      4. Nested try-block also checks the same exceptions as above
2. **Decomposition Diagram**

|  |  |  |
| --- | --- | --- |
| **Main** | | |
| **Input** | **Process** | **Output** |
| String value for the date with dashes in the middle: mm-dd-yyyy | Convert each value to int and store it in specific variables.  Call functions from other class and handle exceptions | Invalid exception if invalid input and print date at the end if all correct values |
| Month | Convert it to int and store it in its variable | Print it out at the end of the program |
| Day | Convert it to int and store it in its variable | Print it out at the end of the program |
| Year | Convert it to int and store it in its variable | Print it out at the end of the program |

1. **Test Strategy**
   1. **Valid Data**
   2. **Invalid Data**
2. **Test Plan Version 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | Month between 0 and 12 |  |  |  |  |
| Valid | 2 | Day between 0 and 31 |  |  |  |  |
| Valid | 3 | Year between 1915 and 2017 |  |  |  |  |
| Invalid | 1 | Month < 0 |  |  |  |  |
| Invalid | 2 | Month > 12 |  |  |  |  |
| Invalid | 3 | Day < 0 |  |  |  |  |
| Invalid | 4 | Day > 31 |  |  |  |  |
| Invalid | 5 | Year < 1915 |  |  |  |  |
| Invalid | 6 | Year > 2017 |  |  |  |  |

1. **Initial Algorithm**
   1. In main
      1. Welcome message
      2. Ask user for a date and store it in a variable *thisDate*
         1. Take each part of the date, which is month, day, and year, and convert them to int using built in function stoi() and store each value in their respective variables
      3. Make an instance of class *exceptions*
      4. Call each function from the class and pass in appropriate parameter
         1. If the function returns false, print out thank you message and assume the input is correct
      5. Print out the full birthdate using dashes, but using concatenation and the integer variables rather than the user input string
      6. Thank you message
   2. In *exceptions* class
      1. Keep constructor empty
      2. In *tryDay* function
         1. Make a Boolean variable to store whether the input value is correct or not
         2. Try: if day is not between 1 and 31, set the Boolean value to false, then try: Ask user to the day again and store it in passed by reference variable
         3. Catch the day and print out cannot have that many days in a month
         4. Throw again to 1st try block
         5. Catch the day and say invalid input
         6. Return result
      3. In *tryMonth* function
         1. Make a Boolean variable to store whether the input value is correct or not
         2. Try: if month is not between 1 and 12, set the Boolean value to false, then try: Ask user to the month again and store it in passed by reference variable
         3. Catch the month and print out cannot have that many months in a year
         4. Throw again to 1st try block
         5. Catch the month and say invalid input
         6. Return result
      4. In *tryYear* function
         1. Make a Boolean variable to store whether the input value is correct or not
         2. Try: if year is not between 1915 and 2017, set the Boolean value to false, then try: Ask user to enter the year again and store it in passed by reference variable
         3. Catch the year and print out cannot be born in that year
         4. Throw again to 1st try block
         5. Catch the year and say invalid input
         6. Return result
2. **Test Plan Version 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | Month between 1 and 12, Day between 0 and 31, Year between 1915 and 2017 | 05-30-1999 | Prints the converted date |  |  |
| Invalid | 1 | Month < 1 | -1-30-1999 | “Incorrect Month” and asks for month again |  |  |
| Invalid | 2 | Month > 12 | 50-30-1999 | “Invalid month: 50” |  |  |
| Invalid | 3 | Day < 1 | 05—5-2017 | “Invalid day: -5” |  |  |
| Invalid | 4 | Day > 31 | 05-50-2017 | “Invalid day: 50” |  |  |
| Invalid | 5 | Year < 1915 | 05-30-1910 | “Invalid year: 1910” |  |  |
| Invalid | 6 | Year > 2017 | 05-30-2018 | “Invalid year: 2018” |  |  |

1. **Code**

**Main**

// CIS200\_Lab\_10.2.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <iostream>

#include <string>

#include "exceptions.h"

using namespace std;

int main()

{

exceptions date;

cout << "Welcome human, let's get rolling!" << endl;

system("pause");

system("cls");

string thisDate;

int day, month, year;

cout << "Give me a date with dashes in the middle: ";

cin >> thisDate;

month = stoi(thisDate.substr(0, 2));

if (!date.tryMonth(month))

{

cout << "Thanks for letting me know your month of birth assuming it's correct" << endl;

}

day = stoi(thisDate.substr(3, 2));

if (!date.tryDay(day))

{

cout << "Thanks for letting me know your day of birth assuming it's correct" << endl;

}

year = stoi(thisDate.substr(6, 4));

if (!date.tryYear(year))

{

cout << "Thanks for letting me know your year of birth assuming it's correct" << endl;

}

switch (month)

{

case 1:

cout << "Your birth date is January " << day << ", " << year << endl;

break;

case 2:

cout << "Your birth date is February " << day << ", " << year << endl;

break;

case 3:

cout << "Your birth date is March " << day << ", " << year << endl;

break;

case 4:

cout << "Your birth date is April " << day << ", " << year << endl;

break;

case 5:

cout << "Your birth date is May " << day << ", " << year << endl;

break;

case 6:

cout << "Your birth date is June " << day << ", " << year << endl;

break;

case 7:

cout << "Your birth date is July " << day << ", " << year << endl;

break;

case 8:

cout << "Your birth date is August " << day << ", " << year << endl;

break;

case 9:

cout << "Your birth date is September " << day << ", " << year << endl;

break;

case 10:

cout << "Your birth date is October " << day << ", " << year << endl;

break;

case 11:

cout << "Your birth date is November " << day << ", " << year << endl;

break;

case 12:

cout << "Your birth date is December " << day << ", " << year << endl;

break;

default:

cout << "Invalid month" << endl;

break;

}

cout << "Thanks for using me, human. See ya next time!" << endl;

return 0;

}

**Exceptions.h**

#pragma once

#ifndef EXCEPTIONS\_H

#define EXCEPTIONS\_H

#include <iostream>

#include <string>

using namespace std;

class exceptions

{

public:

exceptions();

bool tryDay(int &day);

bool tryMonth(int &month);

bool tryYear(int &year);

private:

};

exceptions::exceptions()

{

; //empty

}

bool exceptions::tryDay(int &day)

{

bool result = true;

try {

if (day < 1 || day > 31)

{

result = false;

try {

cout << "Incorrect day, what is the correct day: ";

cin >> day;

if (day < 1 || day > 31) {

result = false;

throw day;

}

result = true;

}

catch (int day)

{

cout << "You cannot have " << day << " days in a month" << endl;

throw;

}

}

}

catch(int day) {

cout << "Invalid input: " << day << endl;

}

return result;

}

bool exceptions::tryMonth(int &month)

{

bool result = true;

try {

if (month < 1 || month > 12)

{

result = false;

try {

cout << "Incorrect Month, what is the correct month: ";

cin >> month;

if (month < 1 || month > 12) {

result = false;

throw month;

}

result = true;

}

catch (int month)

{

cout << "You cannot have " << month << " months in a year" << endl;

throw;

}

}

}

catch (int month) {

cout << "Invalid month: " << month << endl;

}

return result;

}

bool exceptions::tryYear(int &year)

{

bool result = true;

try {

if (year < 1915 || year > 2017)

{

result = false;

try {

cout << "Incorrect year, what is the correct year: ";

cin >> year;

if (year < 1915 || year > 2017) {

result = false;

throw year;

}

result = true;

}

catch (int year)

{

cout << "You cannot be born in the year " << year << endl;

throw;

}

}

}

catch (int month) {

cout << "Invalid year: " << month << endl;

}

return result;

}

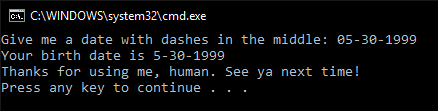
#endif // !EXCEPTIONS\_H

1. **Updated Algorithm**
   1. In main
      1. Welcome message
      2. Ask user for a date and store it in a variable *thisDate*
         1. Take each part of the date, which is month, day, and year, and convert them to int using built in function stoi() and store each value in their respective variables
      3. Make an instance of class *exceptions*
      4. Call each function from the class and pass in appropriate parameter
         1. If the function returns false, print out thank you message and assume the input is correct
      5. Print out the full birthdate ~~using dashes~~ in the format (month day, year), but using concatenation and the integer variables rather than the user input string
      6. Thank you message
   2. In *exceptions* class
      1. Keep constructor empty
      2. In *tryDay* function
         1. Make a Boolean variable to store whether the input value is correct or not
         2. Try: if day is not between 1 and 31, set the Boolean value to false, then try: Ask user to the day again and store it in passed by reference variable
            1. If day is not between 1 and 31, set result to false and throw day
            2. Set result to true
         3. Catch the day and print out cannot have that many days in a month
         4. Throw again to 1st try block
         5. Catch the day and say invalid input
         6. Return result
      3. In *tryMonth* function
         1. Make a Boolean variable to store whether the input value is correct or not
         2. Try: if month is not between 1 and 12, set the Boolean value to false, then try: Ask user to the month again and store it in passed by reference variable
            1. If month is not between 1 and 12, set result to false and throw month
            2. Set result to true
         3. Catch the month and print out cannot have that many months in a year
         4. Throw again to 1st try block
         5. Catch the month and say invalid input
         6. Return result
      4. In *tryYear* function
         1. Make a Boolean variable to store whether the input value is correct or not
         2. Try: if year is not between 1915 and 2017, set the Boolean value to false, then try: Ask user to enter the year again and store it in passed by reference variable
            1. If year is not between 1915 and 2017, set result to false and throw year
            2. Set result to true
         3. Catch the year and print out cannot be born in that year
         4. Throw again to 1st try block
         5. Catch the year and say invalid input
         6. Return result
2. **Test Plan Version 3**

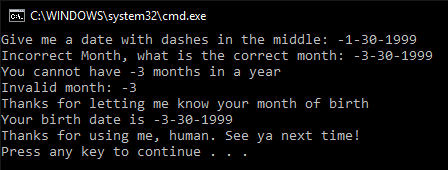
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | Month between 1 and 12, Day between 0 and 31, Year between 1915 and 2017 | 05-30-1999 | Prints the converted date | Prints the converted date | Pass |
| Invalid | 1 | Month < 1 | -1-30-1999 | “Incorrect Month” and asks for month again | “Incorrect month” and asks for month again | Pass |
| Invalid | 1.1 | 2nd try, Month < 1 | -3-30-1999 | “You cannot have -3 months in a year” | “Your cannot have -3 months in a year” | Pass |
| Invalid | 2 | Month > 12 | 50-30-1999 | “Incorrect Month” and asks for month again | “Incorrect Month” and asks for month again | Pass |
| Invalid | 2.1 | 2nd try, Month > 12 | 90-30-1999 | “You cannot have 90 months in a year” | “You cannot have 90 months in a year” | Pass |
| Invalid | 3 | Day < 1 | 05—5-2017 | “Incorrect day” and asks for day again | “Incorrect day” and asks for day again | Pass |
| Invalid | 3.1 | 2nd try, Day < 1 | 05—1-2017 | “You cannot have -1 days in a month” | “You cannot have -1 days in a month” | Pass |
| Invalid | 4 | Day > 31 | 05-50-2017 | “Incorrect day” and asks for day again | “Incorrect day” and asks for day again | Pass |
| Invalid | 4.1 | 2nd try, Day > 31 | 05-70-2017 | “You cannot have 70 days in a month” | “You cannot have 70 days in a month” | Pass |
| Invalid | 5 | Year < 1915 | 05-30-1910 | “Incorrect year” and asks for year again | “Incorrect year” and asks for year again | Pass |
| Invalid | 5.1 | Year < 1915 | 05-30-1900 | “You cannot be born in the year 1900” | “You cannot be born in the year 1900” | Pass |
| Invalid | 6 | Year > 2017 | 05-30-2018 | “Incorrect year” and asks for year again | “Incorrect year” and asks for year again | Pass |
| Invalid | 6.1 | Year > 2017 | 05-30-2020 | “You cannot be born in the year 2020” | “You cannot be born in the year 2020” | Pass |

1. **Screenshots**

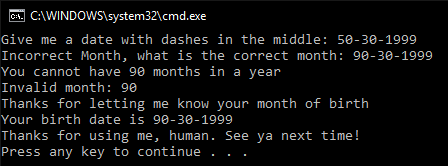
Valid Test Case 1:



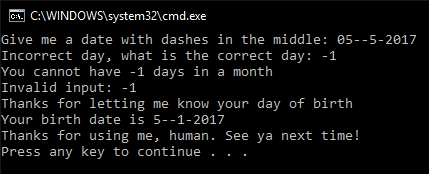
Invalid Test Case 1 and 1.1:



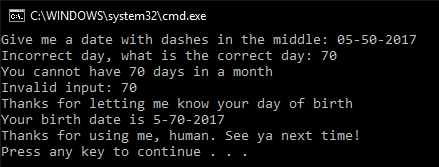
Invalid Test Case 2 and 2.1:



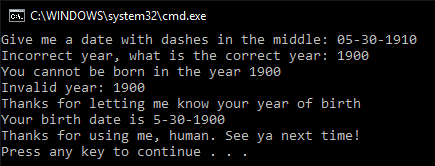
Invalid Test Case 3 and 3.1:



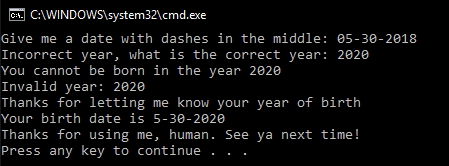
Invalid Test Case 4:



Invalid Test Case 5:



Invalid Test Case 6:



1. **Status**

Program works perfectly with assumptions in mind