1. **Program Statement**

This program uses try-block exception handling for a string input of user’s birthdate in a given format.

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
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. 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
      2. Make an instance of class *exceptions*
      3. Call each function from the class and pass in appropriate parameter
         1. If the function returns false, ask user again to type in correct value for the data value that returns false
         2. Else, print out the data value is correct
      4. Print out the full birthdate using dashes
   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 0 and 31, set the Boolean value to false, then throw the passed in variable
         3. Set Boolean value to true, if passed in variable was never thrown
         4. Catch the integer thrown before and print out the error message
         5. Return the Boolean variable
      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 0 and 12, set the Boolean value to false, then throw the passed in variable
         3. Set Boolean value to true, if passed in variable was never thrown
         4. Catch the integer thrown before and print out the error message
         5. Return the Boolean variable
      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 throw the passed in variable
         3. Set Boolean value to true, if passed in variable was never thrown
         4. Catch the integer thrown before and print out the error message
         5. Return the Boolean variable
2. **Test Plan Version 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | Month between 0 and 12, Day between 0 and 31, Year between 1915 and 2017 | 05-30-1999 | “Month is correct  Day is correct  Year is correct” |  |  |
| Invalid | 1 | Month < 0 | -1-30-1999 | “Invalid month: -1” |  |  |
| Invalid | 2 | Month > 12 | 50-30-1999 | “Invalid month: 50” |  |  |
| Invalid | 3 | Day < 0 | 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**

**Source.cpp**

#include "exceptionsFirst.h"

#include <iostream>

#include <string>

using namespace std;

int main()

{

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

system("pause");

system("cls");

string thisDate;

int day, month, year;

exceptions date;

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

cin >> thisDate;

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

if (!date.tryMonth(month))

{

cout << "What is the correct month: ";

cin >> month;

}

else {

cout << "Month is correct" << endl;

}

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

if (!date.tryDay(day))

{

cout << "What is the correct day: ";

cin >> day;

}

else {

cout << "Day is correct" << endl;

}

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

if (!date.tryYear(year))

{

cout << "What is the correct year: ";

cin >> year;

}

else {

cout << "Year is 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;

}

**ExceptionsFirst.h**

#pragma once

#ifndef EXCEPTIONSFIRST

#define EXCEPTIONSFIRST

#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;

try {

if (day < 1 || day > 31)

{

result = false;

throw day;

}

result = true;

}

catch (int day)

{

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

}

return result;

}

bool exceptions::tryMonth(int month)

{

bool result;

try {

if (month < 1 || month > 12)

{

result = false;

throw month;

}

result = true;

}

catch (int month)

{

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

}

return result;

}

bool exceptions::tryYear(int year)

{

bool result;

try {

if (year < 1915 || year > 2017)

{

result = false;

throw year;

}

result = true;

}

catch (...) {

cout << "Invalid year" << endl;

}

return result;

}

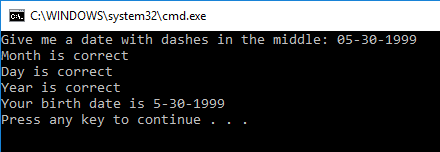
#endif // !EXCEPTIONSFIRST\_H

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, ask user again to type in correct value for the data value that returns false
         2. Else, print out the data value is correct
      5. Print out the full birthdate ~~using dashes~~ in the format (month day, year), but using concatenation and the int variables rather than the user input string
      6. Thank you message
   2. In *exceptionsFirst* 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 0 and 31, set the Boolean value to false, then throw the passed in variable
         3. Set Boolean value to true, if passed in variable was never thrown
         4. Catch the integer thrown before and print out the error message
         5. Return the Boolean variable
      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 0 and 12, set the Boolean value to false, then throw the passed in variable
         3. Set Boolean value to true, if passed in variable was never thrown
         4. Catch the integer thrown before and print out the error message
         5. Return the Boolean variable
      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 throw the passed in variable
         3. Set Boolean value to true, if passed in variable was never thrown
         4. Catch the integer thrown before and print out the error message
         5. Return the Boolean variable
2. **Test Plan Version 3**

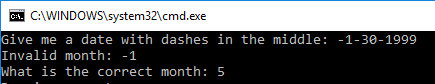
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | Month between 0 and 12, Day between 0 and 31, Year between 1915 and 2017 | 05-30-1999 | “Month is correct  Day is correct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

1. **Screenshots**

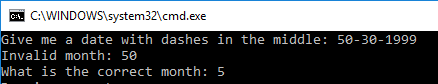
Valid Test Case 1:



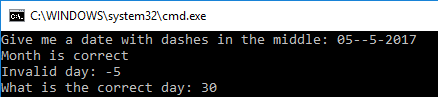
Invalid Test Case 1:



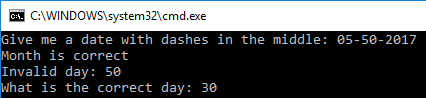
Invalid Test Case 2:



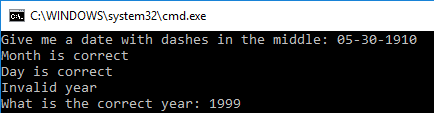
Invalid Test Case 3:



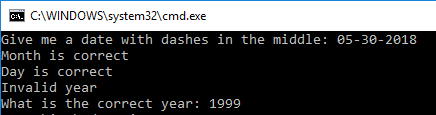
Invalid Test Case 4:



Invalid Test Case 5:



Invalid Test Case 6:



1. **Status**

Program works perfectly with assumptions in mind