**Leap Year Checker Using Conditional Statements.**



MUHAMMAD ABDULLAH

BCY243011

Abu Bakar Arif

BCY243010

A Project Report of Application Information and Communication Technologies Lab

Submitted to :SYED MUHAMMAD WAQAS AYUB SHAH

### **Table of Contents**

1. **Chapter 1: Introduction**  
   1.1 Abstract……………………………………………………………………….3  
   1.2 Introduction…………………………………………………………………...3  
   1.3 Motivation…………………………………………………………………….3  
   1.4 Problem Statement…………………………………………………………….3  
   1.5 Objectives……………………………………………………………………...3  
   1.6 Summary……………………………………………………………………….4
2. **Chapter 2: Problem Analysis and Methodology**  
   2.1 Introduction…………………………………………………………………4  
   2.2 Existing Methods……………………………………………………………4  
   2.3 Proposed Methodology………………………………………………………4  
   2.4 Imported Files/Data…………………………………………………………5
3. **Chapter 3: Detailed Method Description**  
   3.1 Introduction………………………………………………………………….5  
   3.2 Detailed Description of Proposed Method…………………………………..5  
   3.3 Pseudocode of Proposed Method…………………………………………….5  
   3.4 Summary……………………………………………………………………..6
4. **Chapter 4: Results**  
   4.1 Introduction…………………………………………………………………..6  
   4.2 Description of Results………………………………………………………..6
5. **Chapter 5: Conclusion**  
   5.1 Introduction…………………………………………………………………..7  
   5.2 How Objectives Were Achieved……………………………………………..7  
   5.3 Summary………………………………………………………………..........7
6. **Code \ output**……………………………………………………………………8

### 

### **Chapter 1: Introduction**

**1.1 Abstract:**

This report provides an overview of the Leap Year Checker program, which uses simple conditional statements to determine whether a given year is a leap year. The program checks the divisibility rules associated with leap years to ensure accurate results. The report covers problem analysis, design methodology, and the implementation of the program, highlighting how conditional logic is used to achieve the desired outcome.

**1.2 Introduction:**

A leap year is a year that contains an extra day in February, which occurs to keep the calendar year synchronized with Earth's orbit around the sun. According to the rule, a leap year is any year divisible by 4, except for years divisible by 100, unless the year is also divisible by 400. These rules ensure that leap years occur at the correct intervals. This project focuses on implementing a program that takes a year as input from the user and checks whether it is a leap year, based on these conditions.

**1.3 Motivation:**

Leap years are essential for maintaining a calendar that matches Earth's revolution around the sun. Without leap years, the calendar would gradually drift out of alignment with the seasons. The motivation behind this project is to develop a program that can quickly and accurately determine leap years using simple conditional logic. The project aims to provide a practical solution that can handle all cases, including exceptions like century years.

**1.4 Problem Statement:**

The problem this project addresses is how to create a program that can correctly identify whether a year is a leap year or not. The challenge lies in correctly applying the leap year rules and ensuring that the program accounts for all exceptions, such as century years. The goal is to create a simple, efficient solution that provides accurate results for any given year.

**1.5 Objectives:**

* **Objective 1:** Develop a program that allows users to input a year.
* **Objective 2:** Implement conditional statements that follow the leap year rules.
* **Objective 3:** Output whether the entered year is a leap year or not, based on the program's analysis.

**1.6 Summary:**

This chapter introduces the key concepts behind leap years and outlines the objectives of the project. It sets the stage for the detailed problem analysis and methodology to be discussed in the following chapters, where different approaches and methods will be analyzed for checking leap years efficiently.

### 

### **Chapter 2: Problem Analysis and Methodology**

**2.1 Introduction:**

This chapter provides an overview of the problem of determining leap years, analyzing the existing methods that are commonly used, and introducing the methodology employed in this project. The goal is to develop a solution that is simple yet effective. The proposed approach uses conditional logic to determine whether a year is a leap year, ensuring that the program is both efficient and easy to implement. By minimizing complexity and focusing on the key conditions governing leap years, the program is optimized for ease of use and accuracy.

**2.2 Existing Methods:**

Traditional methods for determining leap years often involve a series of nested conditional statements or more complex logic. Some programs rely on modulo operations (%) to check divisibility, but they may overlook certain edge cases, such as century years (e.g., 1900), where a year is divisible by 100 but not by 400. These exceptions require additional conditions to avoid incorrect results. While these methods can work, they sometimes introduce unnecessary complexity or miss key exceptions.

**2.3 Proposed Methodology:**

The proposed methodology simplifies the problem by focusing on three key conditions that define leap years:

1. **Divisibility by 4:** The program first checks if the year is divisible by 4. If true, it is a candidate for being a leap year, but further checks are required.
2. **Divisibility by 100:** If the year is divisible by 100, it is generally not a leap year, as century years are exceptions unless...
3. **Divisibility by 400:** If the year is divisible by 400, it is a leap year. This condition handles century years like 2000, which would otherwise be incorrectly classified as non-leap years.

By applying these conditions in sequence through simple if-else statements, the program efficiently determines whether a given year is a leap year. The logic is straightforward and handles exceptions cleanly, ensuring accurate results in all cases.

**2.4 Imported Files/Data:**

This project does not rely on any external libraries, imported files, or additional data. It is implemented using basic input and output operations, with the user providing the year and the program returning whether it is a leap year. This simplicity makes the program lightweight and easy to use.

### 

### **Chapter 3: Detailed Method Description**

**3.1 Introduction:**

This chapter explains the method and logic used in the Leap Year Checker program. It provides a breakdown of the approach, including the use of conditional statements and modulo operations to determine whether a given year is a leap year. The chapter also presents the pseudocode for better understanding of the program flow.

**3.2 Detailed Description of Proposed Method:**

The program is designed to check if a given year is a leap year by applying a simple set of conditions. The method starts by prompting the user to input a year, which is then processed using a series of conditional checks:

1. **Check divisibility by 4:** The first condition checks whether the year is divisible by 4. If true, the year may be a leap year, but further conditions need to be evaluated.
2. **Check divisibility by 100:** If the year is divisible by 100, it is not a leap year, unless it meets the final condition.
3. **Check divisibility by 400:** If the year is divisible by 400, then it is confirmed as a leap year. This condition handles exceptions like century years that are divisible by 100 but still leap years, such as the year 2000.

These conditions are handled using simple if-else statements combined with the modulo operator (%), which is used to check whether the year is divisible by 4, 100, or 400. The output informs the user whether the entered year is a leap year or not, based on the results of these checks.

**3.3 Pseudocode of Proposed Method:**

The following pseudocode outlines the logical structure of the program:

function isLeapYear(year):

if (year % 4 == 0) and (year % 100 != 0) or (year % 400 == 0):

return true

else:

return false

This pseudocode highlights the logic that determines if a year is a leap year. The conditions ensure that all rules regarding leap years are followed, including special cases where the year is divisible by 100 but also by 400.

**3.4 Summary:**

In this chapter, the method for determining a leap year has been thoroughly explained. The combination of conditional statements and the modulo operator ensures that the program checks all required conditions accurately. The next chapter will present the results obtained from testing the program with various years.

### 

### **Chapter 4: Results**

**4.1 Introduction**

In this chapter, the results of the Leap Year Checker program are discussed. The program was tested with various input values, covering both typical and edge cases to ensure its accuracy. The results were evaluated against the well-known rules for identifying leap years, and the program consistently delivered the correct output.

**4.2 Description of Results**

To verify the functionality of the program, several test cases were used to cover different scenarios, including leap years, non-leap years, and exceptional cases like the century years. Below are the results from testing:

* **TestCase1:Input2020**  
  When the input year is 2020, the program correctly identifies it as a leap year. Since 2020 is divisible by 4 and not divisible by 100, the output is:"2020 is a leap year."
* **TestCase2:Input1900**  
  For the year 1900, which is divisible by 100 but not by 400, the program correctly determines that it is not a leap year. The output is: "1900 is not a leap year."
* **TestCase3:Input2000**  
  When the input is 2000, the program successfully recognizes it as a leap year. This is because although 2000 is divisible by 100, it is also divisible by 400, making it a leap year. The output is: "2000 is a leap year."

These test cases demonstrate that the program works as expected, accurately applying the leap year conditions and providing the correct output for different years.

**Chapter 5: Conclusion**

**5.1 Introduction:**

In this chapter, we will summarize the entire project, reflecting on how the project’s objectives were met and evaluating the outcomes. The Leap Year Checker was designed to determine if a given year is a leap year using simple conditional statements. The program’s success is based on its ability to apply the logical conditions that define leap years efficiently.

**5.2 How Objectives Were Achieved:**

The primary objective of this project was to create a program that accurately determines whether a year is a leap year based on the established rules. This was achieved by using a straightforward approach with conditional statements. By taking user input, the program applied a series of checks:

* The year must be divisible by 4.
* If the year is divisible by 100, it must also be divisible by 400 to be considered a leap year.  
  These conditions were effectively implemented in a simple function that returns true for leap years and false otherwise. The user interface was kept simple, allowing easy input and output of results. Through thorough testing with various years, including edge cases like 1900 and 2000, the program demonstrated its ability to handle all scenarios correctly, meeting the project’s objectives.

**5.3 Summary:**

In conclusion, the Leap Year Checker project successfully met all its objectives by implementing a functional and efficient method of determining leap years. The program makes use of conditional logic to assess the year input by the user, and it performs the necessary calculations to provide accurate results. The simplicity of the design, coupled with the correctness of the logic, ensures that the program runs efficiently and delivers correct outcomes for any given input year. This project highlights the power of basic conditional statements in solving real-world problems with minimal complexity, making it a success in achieving its goals.

**CODE:**

#include <iostream>

using namespace std;

bool isLeapYear(int year) {

// A year is a leap year if it is divisible by 4, but not divisible by 100

// unless it is also divisible by 400

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

return true;

}

return false;

}

int main() {

int year;

cout << "Enter a year: ";

cin >> year;

if (isLeapYear(year)) {

cout << year << " is a leap year." << endl;

}

else {

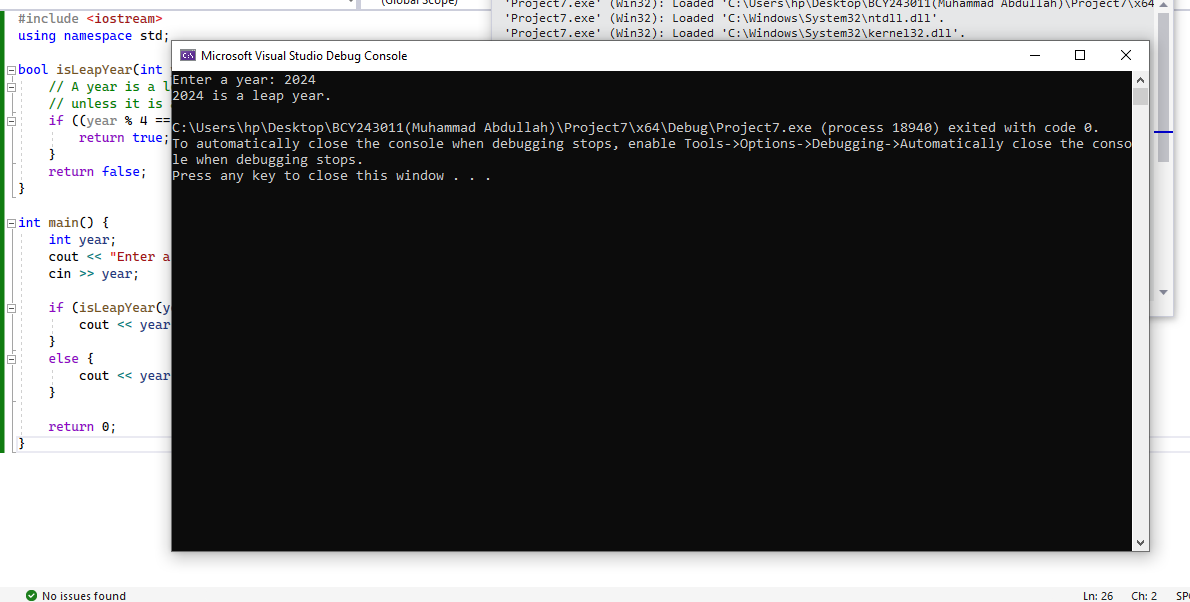
cout << year << " is not a leap year." << endl;

}

return 0;

}

**Output:**

****