CS-114 Fundamentals of Programming

Project Report

Creation of a Feature Rich Calendar in C++ Using Programming Fundamentals

Supervised by Asst Prof Jahan Zeb

Abstract

The development of calendars has always been a fascinating aspect of human history, serving as essential tools for organizing and tracking time. In this study, we delve into the analysis of calendars spanning multiple years, with the goal of uncovering recurring patterns that can be manipulated to create our own custom calendar.

By closely examining the behavior of the first day of each month over time, we discovered a consistent trend that allowed us to predict its movement. Additionally, we encountered the unique challenges presented by leap years, which required special consideration due to the introduction of an extra day.

Through careful analysis and problem-solving, we tackled these issues and gained valuable insights for our calendar implementation project. This paper provides an overview of our findings and the strategies employed to construct a comprehensive calendar application using fundamental principles in the C++ programming language.

Contents

Chapter 1

Introduction

1.1 Libraries Used

The following libraries were used:

- iostream
- fstream

1.2 Description

After thoroughly analyzing the calendars of several years, we dedicated our efforts to uncovering any discernible patterns that could be manipulated to create our very own calendar system.Much to our delight, we discovered a remarkable trend that proved consistent throughout the years: the first day of each month progressively shifted forward by one day, with a slight variation occurring after a leap year.

To illustrate this pattern, let's consider the example of January 1st, 2001, which happened to fall on a Monday. The subsequent year, January 1st, 2002, fell on a Tuesday, following the one-day forward progression.Continuing this trend, January 1st of 2003 fell on a Wednesday, and in 2004, it landed on a Thursday. However, in 2004, being a leap year, an additional day, February 29th, was introduced, which momentarily disrupted our established flow.

The presence of the extra day in a leap year posed a unique challenge. In the subsequent year, 2005, January 1st did not fall on a Friday as expected; instead, it was pushed forward to Saturday. This adjustment accounted for the additional day in the leap year, ensuring that the pattern remained intact. We applied similar considerations for the other months, diligently accommodating leap years' periodic disruptions to maintain consistency.

Although the intricacies of leap years added complexity to our calendar design endeavor, we swiftly addressed these issues with careful calculations. By adhering to the observed pattern, which involved a one-day shift for most years and a two-day shift after leap years, we were able to establish a calendar system that embodied our desired characteristics.

Our analysis and understanding of the calendars spanning several years laid the foundation for the creation of our custom calendar, meticulously accounting for the progression of days, the impact of leap years, and the continuous flow of time.

Chapter 2

Limitations

Initially, we chose the year 2001 as our reference point for developing our calendar system due to its convenience and ease of handling. It served as an ideal starting point for building our code, as it exhibited the desired characteristics we sought, namely having January 1st fall on a Monday. This choice allowed us to establish arrays and references aligned with the year 2001, forming the backbone of our program's functionality.

However, as we progressed in our development and witnessed the successful behavior and desired responses of our code, we recognized the need to expand our horizons and explore other years that shared similar characteristics to 2001. We aimed to find alternative reference years to ensure the versatility and applicability of our calendar system across a broader range of time.

After exhaustive searches, meticulous evaluations, and thorough testing, we ultimately arrived at the year 1973 as our finalized reference year. This carefully selected year possessed the same fundamental qualities as 2001, with January 1st falling on a Monday. By transitioning our code's foundation to align with 1973, we ensured that our calendar program could generate accurate calendars for any year post-1973.

Nevertheless, it is important to acknowledge the limitations of our calendar system. Due to the reliance on the characteristics of the year 1973 as the basis for our code, the program cannot accurately print calendars for years prior to 1973. This constraint arises from the specific patterns and behaviors observed in the calendars we analyzed, which were most applicable and consistent from 1973 onwards.

While the calendar system we devised serves as a robust and reliable tool for generating calendars within the post-1973 timeframe, it is crucial to exercise caution and awareness of its limitations when attempting to utilize it for earlier years. The effectiveness and accuracy of the calendar program are optimized for years following 1973, ensuring a seamless experience for users seeking calendars within this specified range.

Chapter 3

Conclusion

Throughout this project and the course as a whole, we have gained valuable insights into problem-solving using the principles of simplicity and effectiveness while exploring the vast realm of programming. This project, in particular, presented us with a significant challenge: to devise a functioning calendar system that met our specific requirements. By applying our newfound knowledge and skills, we successfully tackled this task, ultimately achieving our desired outcome.

The journey of developing our calendar system taught us the importance of breaking down complex problems into manageable components and seeking elegant solutions. We learned to analyze patterns, identify trends, and devise strategies to accommodate unique circumstances, such as leap years, within our programming logic. The process involved meticulous testing, careful consideration of data structures, and the implementation of algorithms that allowed us to generate accurate calendars for any year post-1973.

While we take pride in our accomplishment, it is crucial to acknowledge that our solution has inherent limitations. Due to its reliance on the specific characteristics observed in the year 1973, our calendar system cannot generate accurate calendars for years preceding that date. Additionally, there may be unforeseen edge cases or scenarios where our system may not function optimally.

Nevertheless, this project served as a stepping stone in our journey as programmers, showcasing the power of logical reasoning, problem-solving skills, and attention to detail. We gained a deeper appreciation for the intricacies and possibilities offered by programming languages and the immense potential they hold for solving real-world challenges.

As we move forward, we will continue to refine our skills, expanding our knowledge and exploring new techniques to develop more robust and versatile solutions. The lessons learned from this project, along with our experiences in the course, will undoubtedly shape our future endeavors, equipping us with the tools necessary to overcome obstacles, think creatively, and make meaningful contributions in the realm of programming and beyond.