Green University of Bangladesh

Department of EEE (Day) || Spring, 2023

Calendar Manager

Course Title: Computer Programming Lab ||

Code: CSE 102 || **Section**: 232 D2

Students Details

Name	Id
Naimul Hassan Sourov	**232001055**
Md. Rabbi khan	**232001055**

Submission Date: 14 June 2024

Course Teacher's Name: MD SAKLAIN MORSHED

Lab Project Status
Marks: Signature: Comments: Date:

Table of Contents

1. Introduction

1.1 Title of the Project

Calendar Manager

1.2 Introduction

This Calendar Manager project offers functionalities such as printing a calendar for any year, determining the day of the week for a specific date, calculating the day number within a year, computing the difference between two dates, and adding days to a specific date. With an easy-to-use menu-driven interface, Calendar Manager makes handling dates straightforward and efficient.

1.3 Why Choose Calendar Manager

- Comprehensive date functionalities
- Efficient and accurate calculations
- User-friendly, menu-driven interface
- Modular, expandable codebase

1.4 Problem Statement

The project is designed to address the need for managing and interacting with dates effectively, providing functionalities like viewing calendars, calculating date differences, and modifying dates.

1.5 Objectives

- Offer a range of functions for viewing, calculating, and managing dates.
- Ensure precise and fast date calculations using reliable algorithms.
- Provide an easy-to-use interface suitable for all user levels.
- Serve as a resource for learning date/time management and programming concepts.
- Design the code for easy updates and extensions by future developers.

2. Implementation of the Project

2.1 Tools & Technologies

- Hp laptop
- Code: Blocks

2.3 Concepts Used

- Conditional Logic
- Loops
- Algorithmic Logic
- Arrays
- Modular Programming

2.4 Learning Topics

2.4.1 Conditional Logic

Used to determine whether a year is a leap year and how many days are in each month.

2.4.2 Loops

For iterating over days, months, and years, and for controlling the flow of the application based on user input.

2.4.3 Algorithmic Logic

Implementing algorithms like Sakamoto's method for calculating the day of the week demonstrates proficiency in logical problem-solving.

2.4.4 Arrays

Employed to store and manage names of months and days of the week, facilitating easy access and manipulation.

2.4.5 Modular Programming

The project is divided into functions, each handling specific tasks such as printing a calendar, determining leap years, or handling date differences, enhancing code readability and maintainability.

2.5 Code Implementation

The Calendar Manager project uses C programming language to implement several features that revolve around calendar and date management. Below are key aspects of the code implementation:

2.5.1 Main Function and Menu Display

The main function drives the menu-based interface, allowing users to select different operations. The function prints the available options to the user.

```c
#include <stdio.h>
#include <stdlib.h>

// Function prototypes
int isLeapYear(int year);
int getDaysInMonth(int month, int year);
int getDayOfWeek(int day, int month, int year);
void printMonth(int month, int year);
void printYear(int year);
void printDayOfWeek(int day, int month, int year);
int dayOfYear(int day, int month, int year);
void dateDifference(int day1, int month1, int year1, int day2, int month2, int year2);
void addDays(int day, int month, int year, int daysToAdd);
void displayMenu();

int main() {
    int choice;
    do {
        displayMenu();
        scanf("%d", &choice);

        switch (choice) {
            case 1: {
                int year;
                printf("Enter year: ");
                scanf("%d", &year);
                printYear(year);
                break;
            }
            case 2: {
                int day, month, year;
                printf("Enter day: ");
                scanf("%d", &day);
                printf("Enter month: ");
                scanf("%d", &month);
                printf("Enter year: ");
                scanf("%d", &year);
                printDayOfWeek(day, month, year);
                break;
            }
            case 3: {
                int day, month, year;
                printf("Enter day: ");
                scanf("%d", &day);
                printf("Enter month: ");
                scanf("%d", &month);
                printf("Enter year: ");
                scanf("%d", &year);
                printf("Day of year: %d\n", dayOfYear(day, month, year));
                break;
            }
            case 4: {
                int day1, month1, year1, day2, month2, year2;
                printf("Enter first date (day month year): ");
                scanf("%d %d %d", &day1, &month1, &year1);
                printf("Enter second date (day month year): ");
                scanf("%d %d %d", &day2, &month2, &year2);
                dateDifference(day1, month1, year1, day2, month2, year2);
                break;
            }
            case 5: {
                int day, month, year, daysToAdd;
                printf("Enter day: ");
                scanf("%d", &day);
                printf("Enter month: ");
                scanf("%d", &month);
                printf("Enter year: ");
                scanf("%d", &year);
                printf("Enter number of days to add: ");
                scanf("%d", &daysToAdd);
                addDays(day, month, year, daysToAdd);
                break;
            }
            case 0:
                printf("Exiting program...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 0);

    return 0;
}

void displayMenu() {
    printf("\nMenu:\n");
    printf("1. Print calendar for a year\n");
    printf("2. Find day of the week for a specific date\n");
    printf("3. Find day number within the year\n");
    printf("4. Calculate difference between two dates\n");
    printf("5. Add days to a date\n");
    printf("0. Exit\n");
    printf("Enter your choice: ");
}
```

2.5.2 Determining Leap Years:

The function determines if a year is a leap year. It uses logical conditions to check if the year is divisible by 4, not divisible by 100 unless it's also divisible by 400.

```c
// Function to check if a year is a leap year
int isLeapYear(int year) {
    return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
}
```

2.5.3 Days in Month Calculation

The function returns the number of days in each month, considering leap years.

```c
// Function to get the number of days in a month
int getDaysInMonth(int month, int year) {
    switch(month) {
        case 1: return 31;
        case 2: return isLeapYear(year) ? 29 : 28;
        case 3: return 31;
        case 4: return 30;
        case 5: return 31;
        case 6: return 30;
        case 7: return 31;
        case 8: return 31;
        case 9: return 30;
        case 10: return 31;
        case 11: return 30;
        case 12: return 31;
        default: return 0;
    }
}
```

2.5.4 Day of the Week Calculation

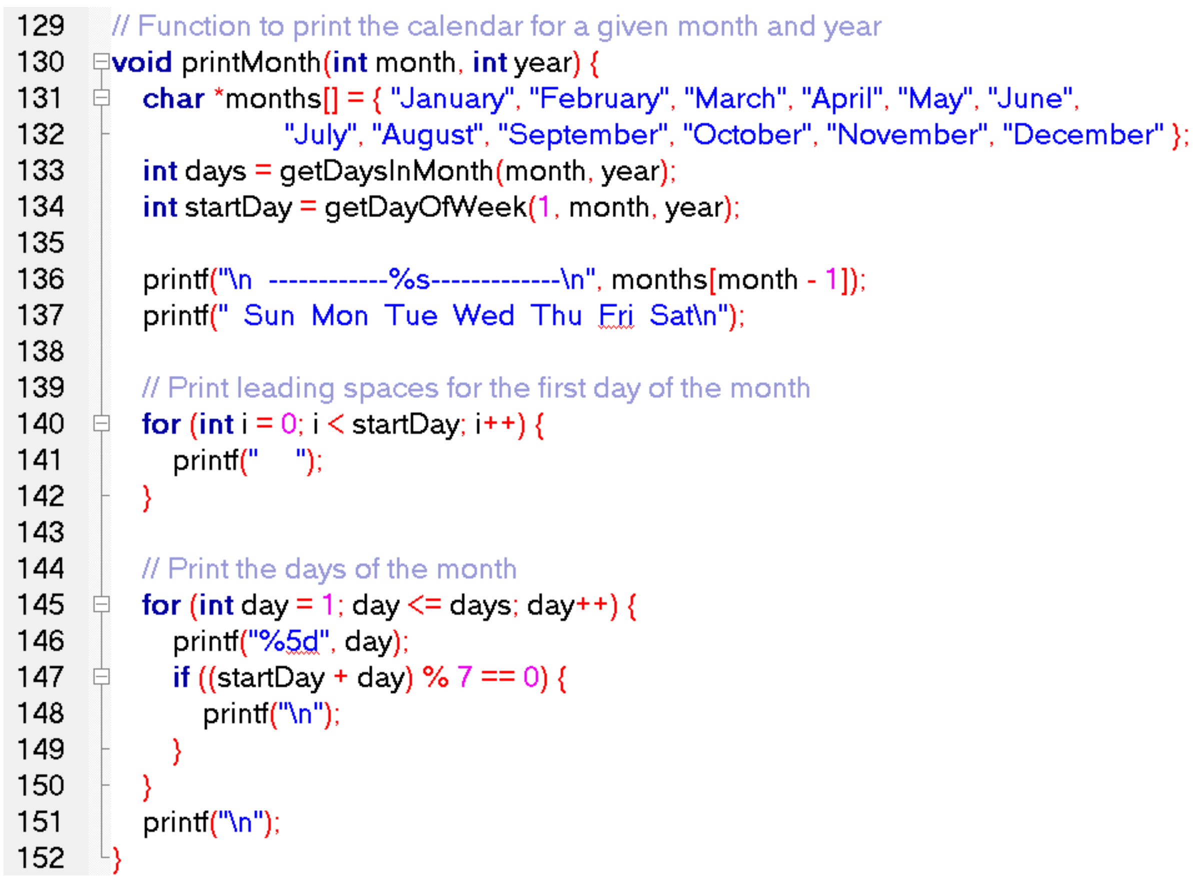
The function uses Sakamoto's method to calculate the day of the week for a given date.

```c
// Function to get the day of the week for a given date using Sakamoto's method
int getDayOfWeek(int day, int month, int year) {
    static int t[] = { 0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4 };
    if (month < 3) {
        year -= 1;
    }
    return (year + year/4 - year/100 + year/400 + t[month-1] + day) % 7;
}
```

2.5.5 Printing Calendar for Month and Year

- The `printMonth` function prints the calendar for a specific month and year.

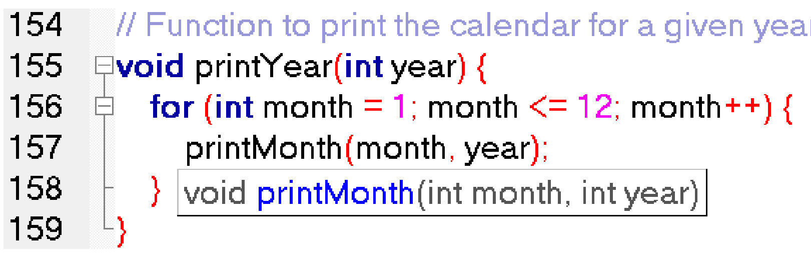
- The `printYear` function prints the calendar for all months each year by calling `printMonth`.



```c
// Function to print the calendar for a given month and year
void printMonth(int month, int year) {
    char *months[] = { "January", "February", "March", "April", "May", "June",
                       "July", "August", "September", "October", "November", "December" };
    int days = getDaysInMonth(month, year);
    int startDay = getDayOfWeek(1, month, year);

    printf("\n  ------------%s-------------\n", months[month - 1]);
    printf("  Sun  Mon  Tue  Wed  Thu  Fri  Sat\n");

    // Print leading spaces for the first day of the month
    for (int i = 0; i < startDay; i++) {
        printf("     ");
    }

    // Print the days of the month
    for (int day = 1; day <= days; day++) {
        printf("%5d", day);
        if ((startDay + day) % 7 == 0) {
            printf("\n");
        }
    }
    printf("\n");
}
```

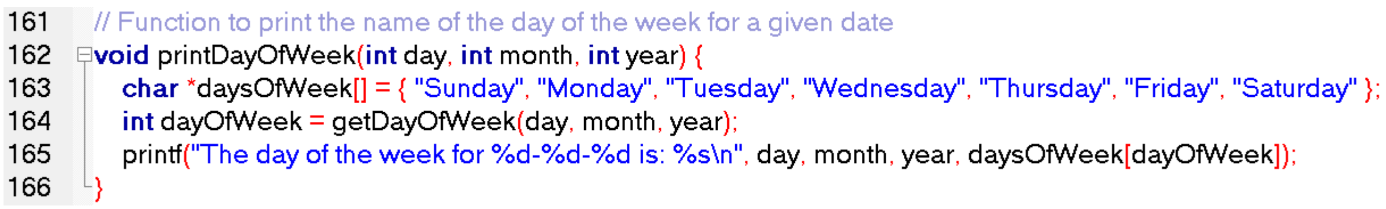
2.5.6 Date Operations

Function to print the calendar for a given year



```c
// Function to print the calendar for a given year
void printYear(int year) {
    for (int month = 1; month <= 12; month++) {
        printMonth(month, year);
    }
}
```

Function to print the name of the day of the week for a given date



```c
// Function to print the name of the day of the week for a given date
void printDayOfWeek(int day, int month, int year) {
    char *daysOfWeek[] = { "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday" };
    int dayOfWeek = getDayOfWeek(day, month, year);
    printf("The day of the week for %d-%d-%d is: %s\n", day, month, year, daysOfWeek[dayOfWeek]);
}
```

Function to get the day number within the year for a given date

```c
// Function to get the day number within the year for a given date
int dayOfYear(int day, int month, int year) {
    int days = day;
    for (int i = 1; i < month; i++) {
        days += getDaysInMonth(i, year);
    }
    return days;
}
```

Function to get the day number within the year for a given date

```c
// Function to calculate the difference between two dates
void dateDifference(int day1, int month1, int year1, int day2, int month2, int year2) {
    int n1 = year1 * 365 + day1;
    for (int i = 0; i < month1 - 1; i++) {
        n1 += getDaysInMonth(i + 1, year1);
    }
    n1 += year1 / 4 - year1 / 100 + year1 / 400;

    int n2 = year2 * 365 + day2;
    for (int i = 0; i < month2 - 1; i++) {
        n2 += getDaysInMonth(i + 1, year2);
    }
    n2 += year2 / 4 - year2 / 100 + year2 / 400;

    printf("Difference: %d days\n", abs(n2 - n1));
}
```

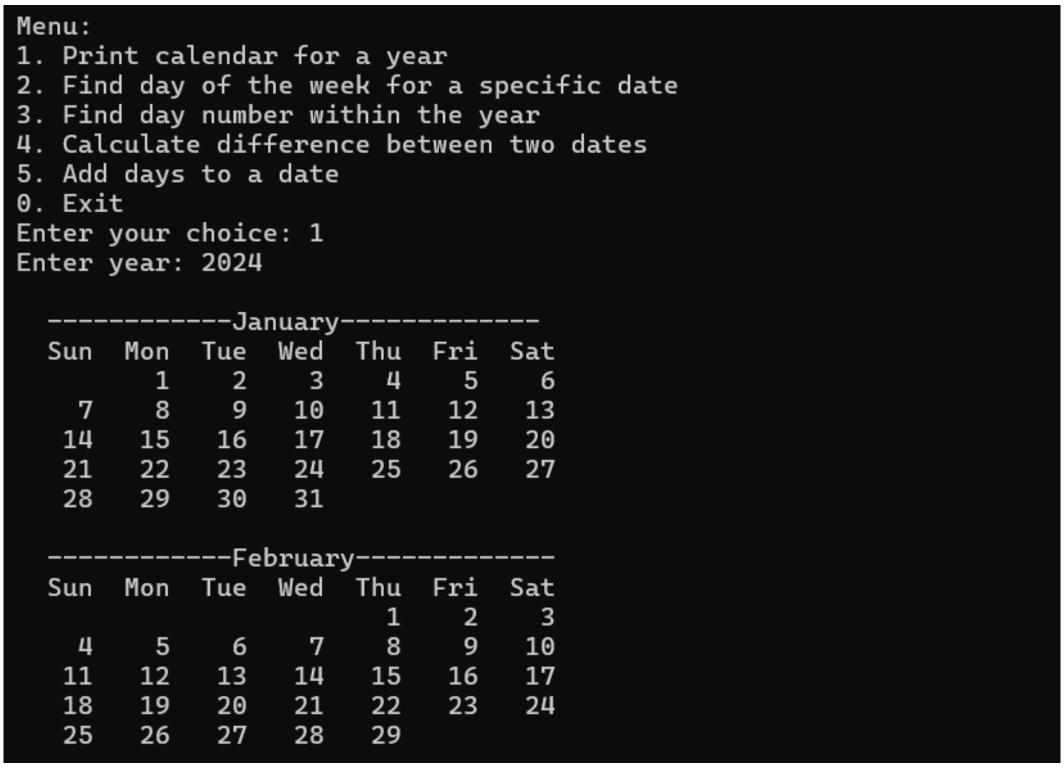
Function to add days to a date and print the new date

```c
// Function to add days to a date and print the new date
void addDays(int day, int month, int year, int daysToAdd) {
    while (daysToAdd > 0) {
        int daysInMonth = getDaysInMonth(month, year);
        if (day + daysToAdd <= daysInMonth) {
            day += daysToAdd;
            daysToAdd = 0;
        } else {
            daysToAdd -= (daysInMonth - day + 1);
            day = 1;
            if (++month > 12) {
                month = 1;
                year++;
            }
        }
    }
    printf("New date: %d-%d-%d\n", day, month, year);
}
```

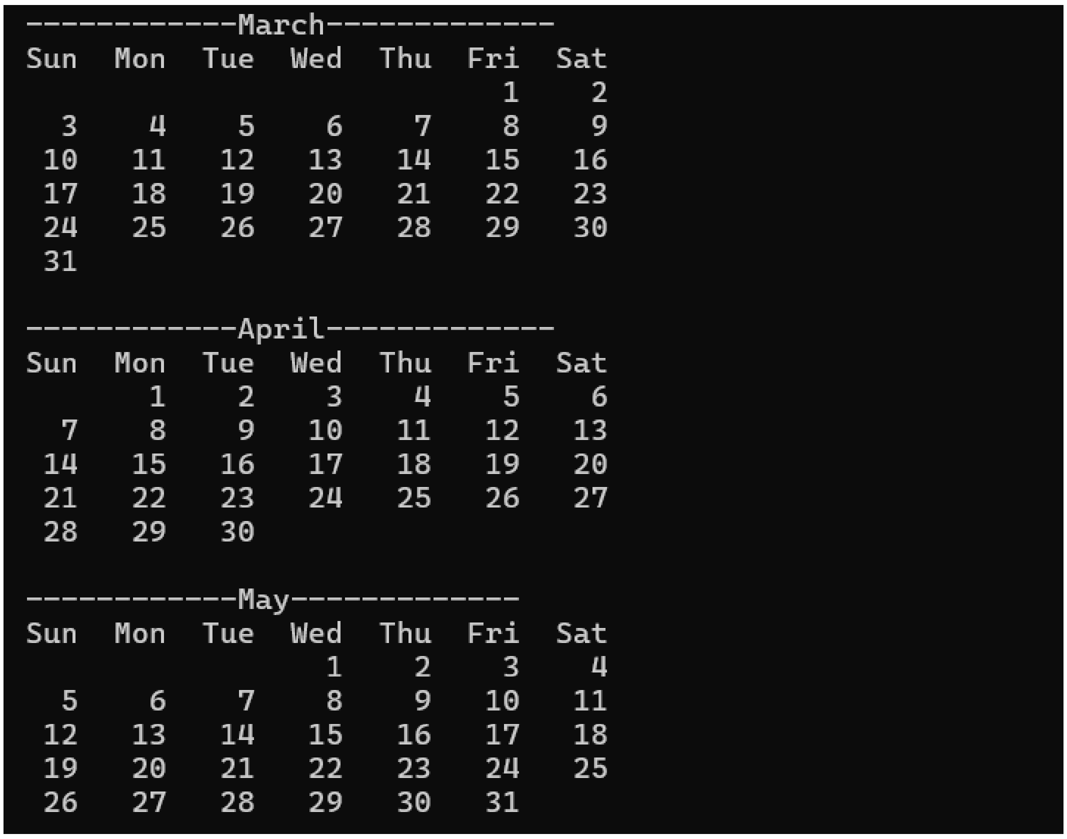
3.Performance Evaluation

3.1 Test Result / Output

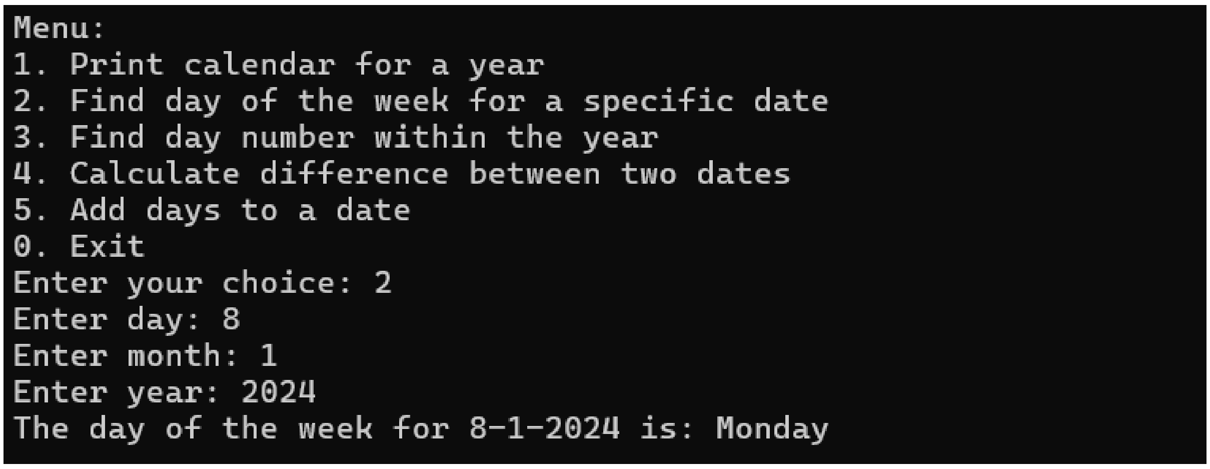
Here's an example of what the interaction might look like:



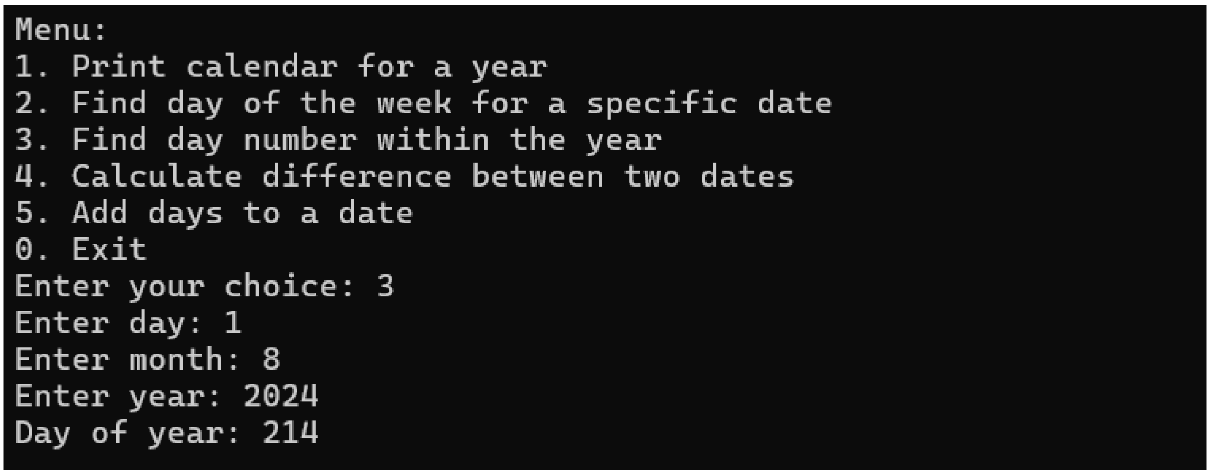
```
Menu:
1. Print calendar for a year
2. Find day of the week for a specific date
3. Find day number within the year
4. Calculate difference between two dates
5. Add days to a date
0. Exit
Enter your choice: 1
Enter year: 2024

  ------------January-------------
  Sun  Mon  Tue  Wed  Thu  Fri  Sat
         1    2    3    4    5    6
    7    8    9   10   11   12   13
   14   15   16   17   18   19   20
   21   22   23   24   25   26   27
   28   29   30   31

  ------------February-------------
  Sun  Mon  Tue  Wed  Thu  Fri  Sat
                        1    2    3
    4    5    6    7    8    9   10
   11   12   13   14   15   16   17
   18   19   20   21   22   23   24
   25   26   27   28   29
```



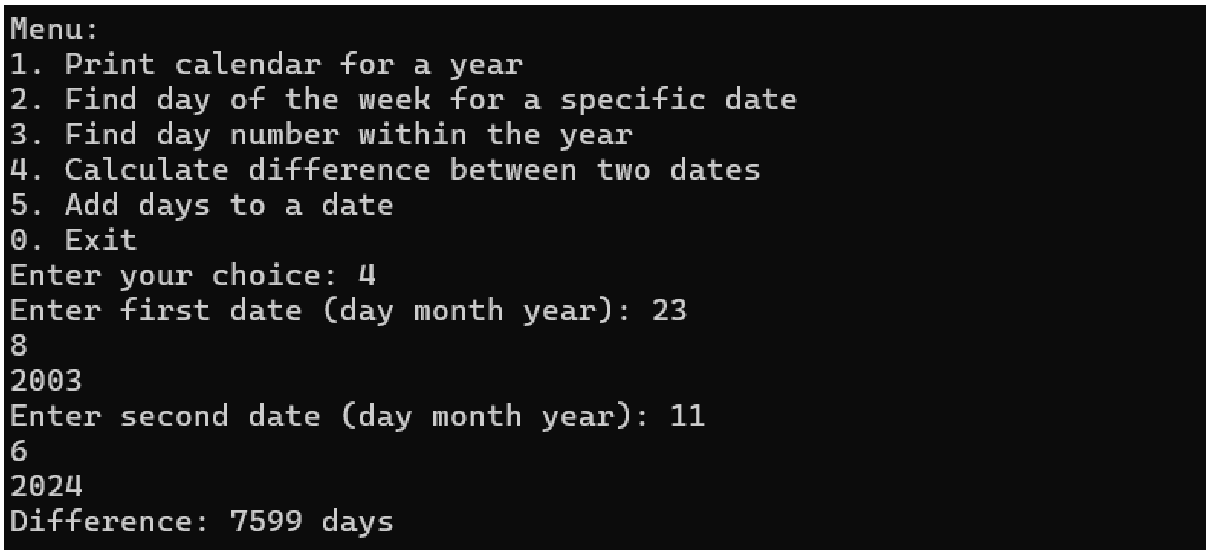
```
------------March-------------
Sun  Mon  Tue  Wed  Thu  Fri  Sat
                         1    2
  3    4    5    6    7    8    9
 10   11   12   13   14   15   16
 17   18   19   20   21   22   23
 24   25   26   27   28   29   30
 31

------------April-------------
Sun  Mon  Tue  Wed  Thu  Fri  Sat
       1    2    3    4    5    6
  7    8    9   10   11   12   13
 14   15   16   17   18   19   20
 21   22   23   24   25   26   27
 28   29   30

------------May-------------
Sun  Mon  Tue  Wed  Thu  Fri  Sat
                 1    2    3    4
  5    6    7    8    9   10   11
 12   13   14   15   16   17   18
 19   20   21   22   23   24   25
 26   27   28   29   30   31
```



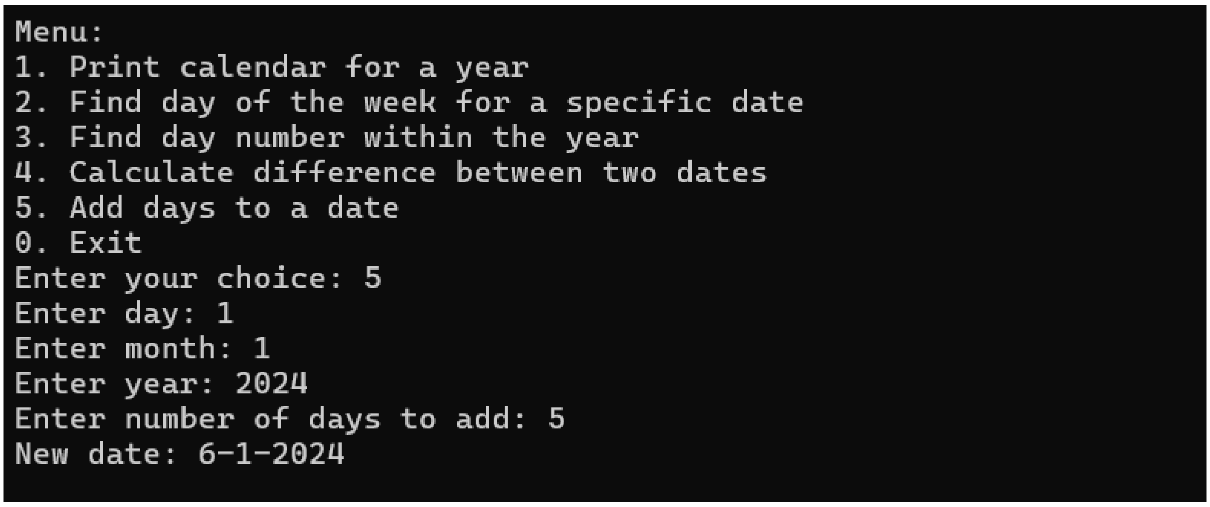
```
Menu:
1. Print calendar for a year
2. Find day of the week for a specific date
3. Find day number within the year
4. Calculate difference between two dates
5. Add days to a date
0. Exit
Enter your choice: 2
Enter day: 8
Enter month: 1
Enter year: 2024
The day of the week for 8-1-2024 is: Monday
```



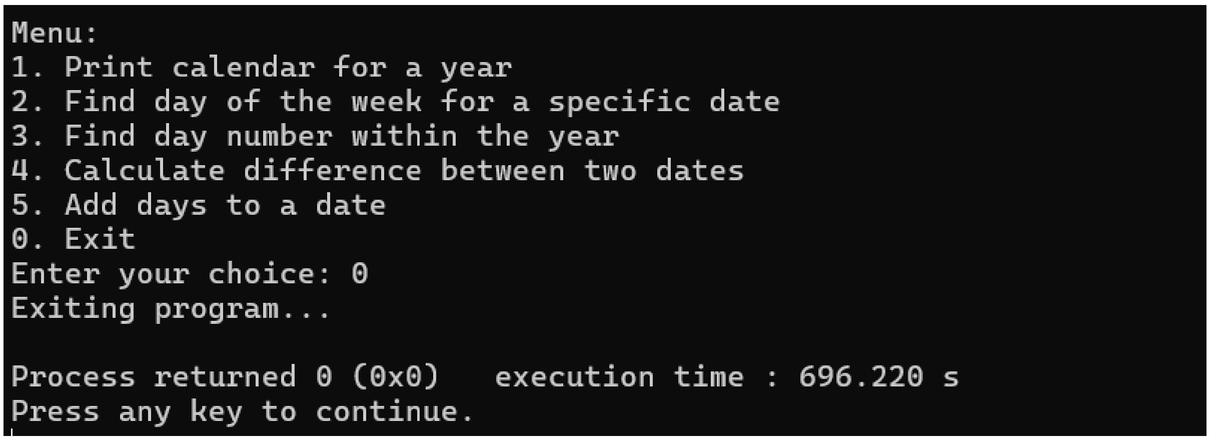
```
Menu:
1. Print calendar for a year
2. Find day of the week for a specific date
3. Find day number within the year
4. Calculate difference between two dates
5. Add days to a date
0. Exit
Enter your choice: 3
Enter day: 1
Enter month: 8
Enter year: 2024
Day of year: 214
```



```
Menu:
1. Print calendar for a year
2. Find day of the week for a specific date
3. Find day number within the year
4. Calculate difference between two dates
5. Add days to a date
0. Exit
Enter your choice: 4
Enter first date (day month year): 23
8
2003
Enter second date (day month year): 11
6
2024
Difference: 7599 days
```



```
Menu:
1. Print calendar for a year
2. Find day of the week for a specific date
3. Find day number within the year
4. Calculate difference between two dates
5. Add days to a date
0. Exit
Enter your choice: 5
Enter day: 1
Enter month: 1
Enter year: 2024
Enter number of days to add: 5
New date: 6-1-2024
```



```
Menu:
1. Print calendar for a year
2. Find day of the week for a specific date
3. Find day number within the year
4. Calculate difference between two dates
5. Add days to a date
0. Exit
Enter your choice: 0
Exiting program...

Process returned 0 (0x0)   execution time : 696.220 s
Press any key to continue.
```

3.2 Achievements

It features an efficient, user-friendly, menu-driven interface, employing robust algorithms like Sakamoto's method for quick and precise results. The modular design and basic error handling enhance readability, maintainability, and robustness, making it an excellent educational tool for learning C programming concepts.

3.3 Challenges Faced

Several challenges were encountered during development, particularly in implementing efficient algorithms like Sakamoto's method and maintaining a modular code structure for seamless functionality posed significant challenges.

4. Future Work

4.1 Additional Features

Additional features could include displaying holidays for the year and providing a graphical interface for easier interaction. Another enhancement could be adding support for different calendar systems, such as lunar or Julian calendars.

5. Conclusion

The Calendar Manager project successfully implements a range of date-related functionalities with accurate calculations and a user-friendly interface. It serves as an effective educational tool for learning C programming concepts and can be further enhanced with additional features and improvements.

Discussion

It features a menu-driven interface for user interaction and employs efficient algorithms for accurate date computations. The project aims to offer a practical tool for date-related tasks while serving as an educational resource for learning C programming concepts.

References

1. Textbook:

- Teach Yourself C – By Herbert Schildt; Publisher: McGraw-Hill Osborne Media; 3rd Edition (April 1, 1997)

2. Online Tutorials or Documentation:

- how to find the week-day for any date?

3. Lecture Notes or Slides:

- Md Saklain Morshed, Lecture Slides on Structure Programming Lab, Green University of Bangladesh.