KIET Group of Institutions, Ghaziabad

COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

PROJECT BASED LEARNING

on

CALENDAR APPLICATION

SUBJECT: DATA STRUCTURE USING C LAB

(KCS-351)

Submitted By:

ABHINAV SHARMA - 2100290110004 (CSIT 3A)
GYANVI ASHISH CHAUDHARY- 2100290110057 (CSIT 3A)

ACKNOWLEDGMENT

I’ve got this golden opportunity to express my kind gratitude and sincere thanks to my subject faculty “Mr. Vinay Kumar”, Computer Science and Information Technology Department, KIET GROUP OF INSTITUTIONS for their kind support and necessary counselling in the preparation of this project report. I’m also indebted to each and every person responsible for the making up of this project directly or indirectly. I must also acknowledge or deep debt of gratitude each one of my colleague who led this project come out in the way it is. It’s my hard work and untiring sincere efforts and mutual cooperation to bring out the project work. Last but not the least, I would like to thank my parents for their sound counselling and cheerful support. They have always inspired us and kept our spirit up.

PROJECT ABSTRACT

The calendar application presented here is a very simple console application developed using C programming language.

It is built without using graphics properties; instead it utilizes many windows properties to give the application a colorful look and feel

It is compiled in online compiler.

INDEX

1. INTRODUCTION
2. REQUIREMENT ANALYSIS
3. DESIGN
4. CODE IMPLEMENTATION
5. RESULT/OUTPUT
6. REFERENCES

INTRODUCTION

PROBLEM STATEMENT:-

TO IMPLEMENT CALENDAR APPLICATION USING C LANGUAGE

A calendar is a system of organizing days. This is done by giving names to periods of time, typically days, weeks, months and years. A date is the designation of a single and specific day within such a system. A calendar is also a physical record (often paper) of such a system.

Calendar, any system for dividing time over extended periods, such as days, months, or years, and arranging such divisions in a definite order. A calendar is convenient for regulating civil life and religious observances and for historical and scientific purposes. The word is derived from the Latin calendarium, meaning “interest register” or “account book,” itself a derivation from calendae , the first day of the month in the Roman Republican calendar the day on which future market days, feasts, and other occasions were proclaimed.

REQUIREMENT ANALYSIS

THIS PROJECT WILL REQUIRE:-

- C COMPILER
- DATA STRUCTURE(ARRAY)

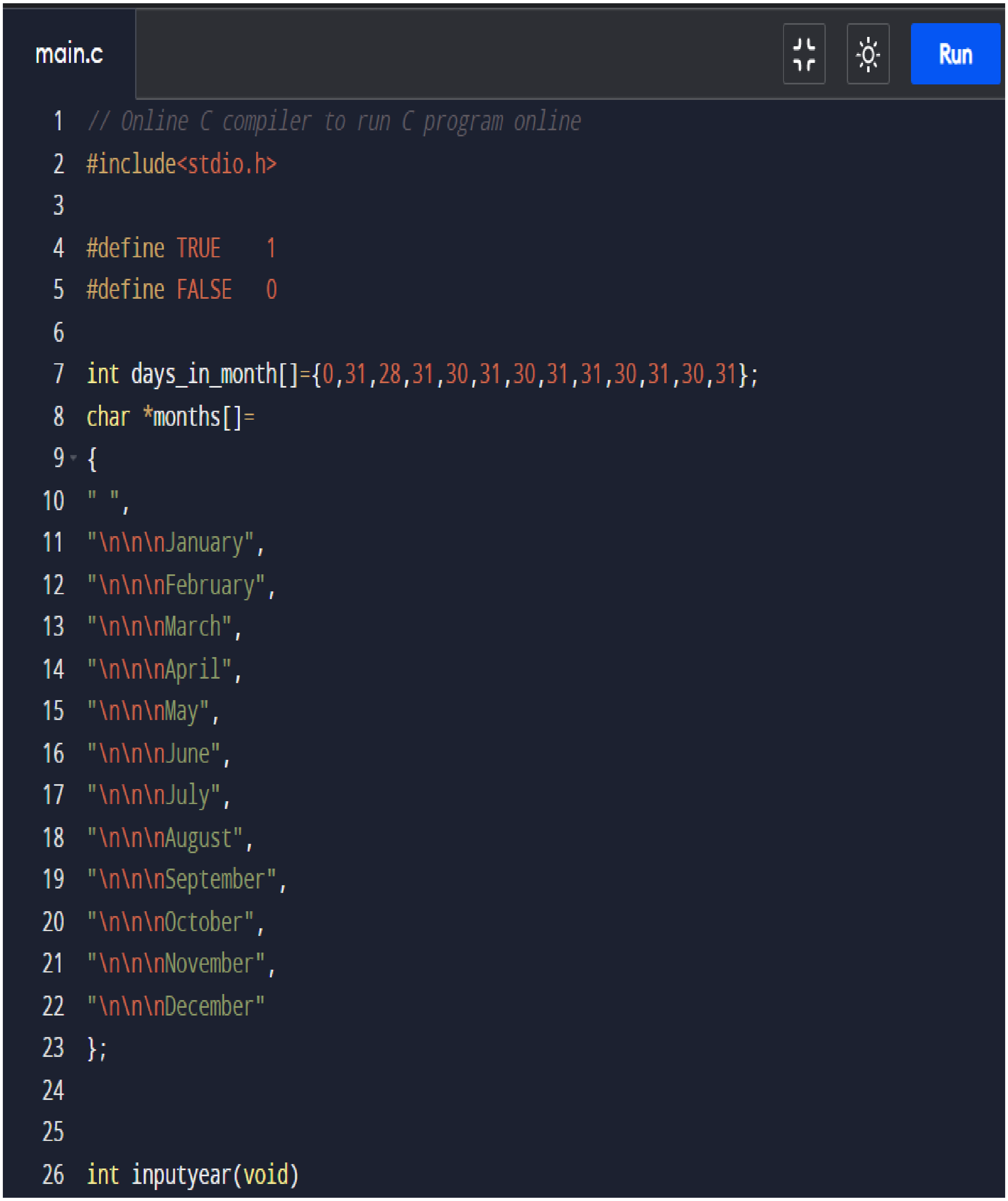
DESIGN

```
#include<stdio.h>

#define TRUE    1
#define FALSE   0

int days_in_month[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
char *months[]=
{
        " ",
        "\n\n\nJanuary",
        "\n\n\nFebruary",
        "\n\n\nMarch",
        "\n\n\nApril",
        "\n\n\nMay",
        "\n\n\nJune",
        "\n\n\nJuly",
        "\n\n\nAugust",
        "\n\n\nSeptember",
        "\n\n\nOctober",
        "\n\n\nNovember",
        "\n\n\nDecember"
};


int inputyear(void)
{
        int year;
        printf("Please enter a year (example: 1999) : ");
        scanf("%d", &year);
        return year;
}

int determinedaycode(int year)
{
        int daycode;
        int d1, d2, d3;

        d1 = (year - 1.)/ 4.0;
        d2 = (year - 1.)/ 100.;
        d3 = (year - 1.)/ 400.;
        daycode = (year + d1 - d2 + d3) %7;
        return daycode;
}


int determineleapyear(int year)
{
        if(year% 4 == FALSE && year%100 != FALSE || year%400 == FALSE)
        {
                days_in_month[2] = 29;
                return TRUE;
        }
        else
        {
                days_in_month[2] = 28;
                return FALSE;
        }
}
```

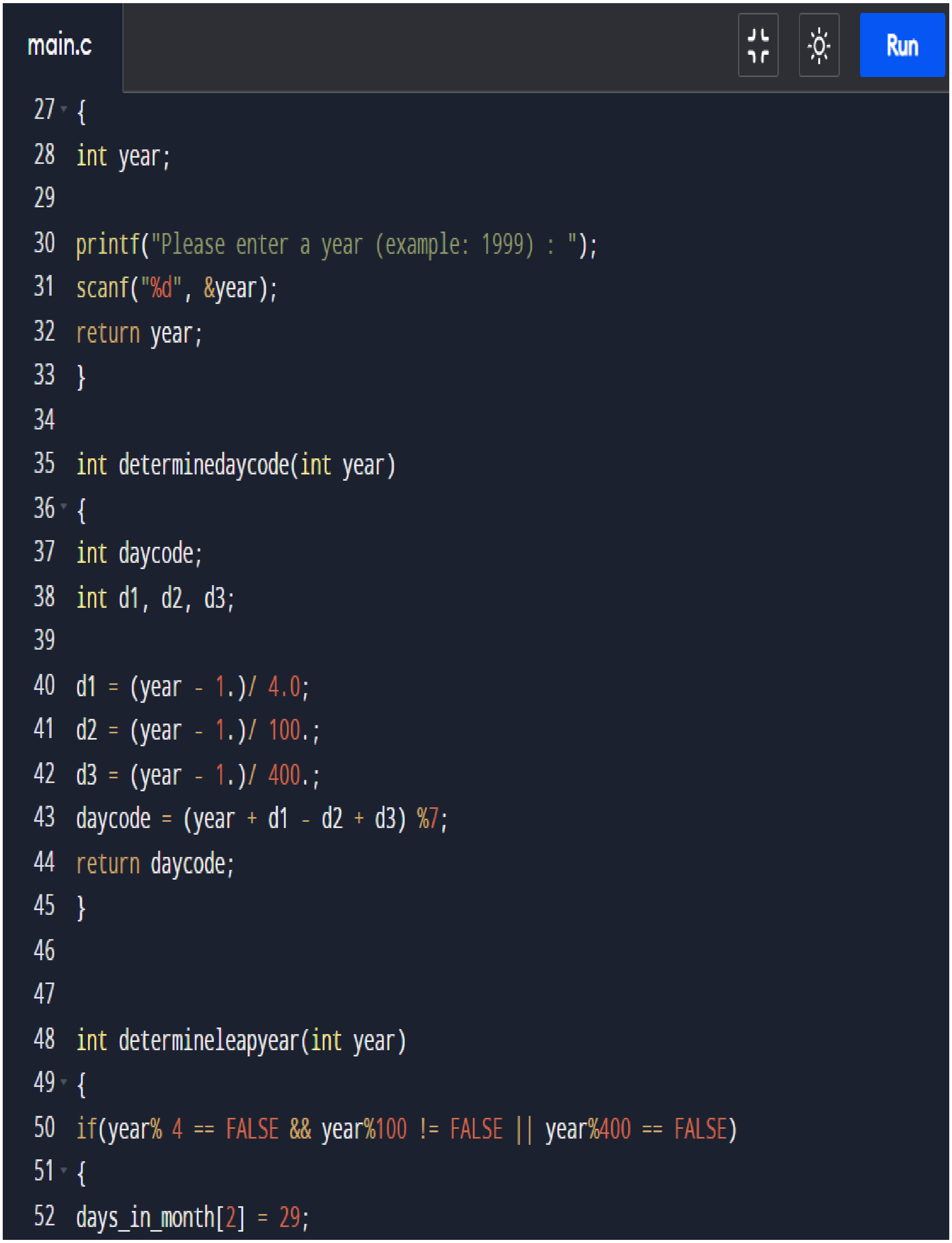
```
void calendar(int year, int daycode)
{
    int month, day;
    for ( month = 1; month <= 12; month++ )
    {
        printf("%s", months[month]);
        printf("\n\nSun  Mon  Tue  Wed  Thu  Fri  Sat\n" );

        // Correct the position for the first date
        for ( day = 1; day <= 1 + daycode * 5; day++ )
        {
            printf(" ");
        }

        // Print all the dates for one month
        for ( day = 1; day <= days_in_month[month]; day++ )
        {
            printf("%2d", day );

            // Is day before Sat? Else start next line Sun.
            if ( ( day + daycode ) % 7 > 0 )
                printf("   " );
            else
                printf("\n " );
        }
            // Set position for next month
            daycode = ( daycode + days_in_month[month] ) % 7;
    }
}
```

```
int main(void)
{
        int year, daycode, leapyear;

        year = inputyear();
        daycode = determinedaycode(year);
        determineleapyear(year);
        calendar(year, daycode);
        printf("\n");
}
```

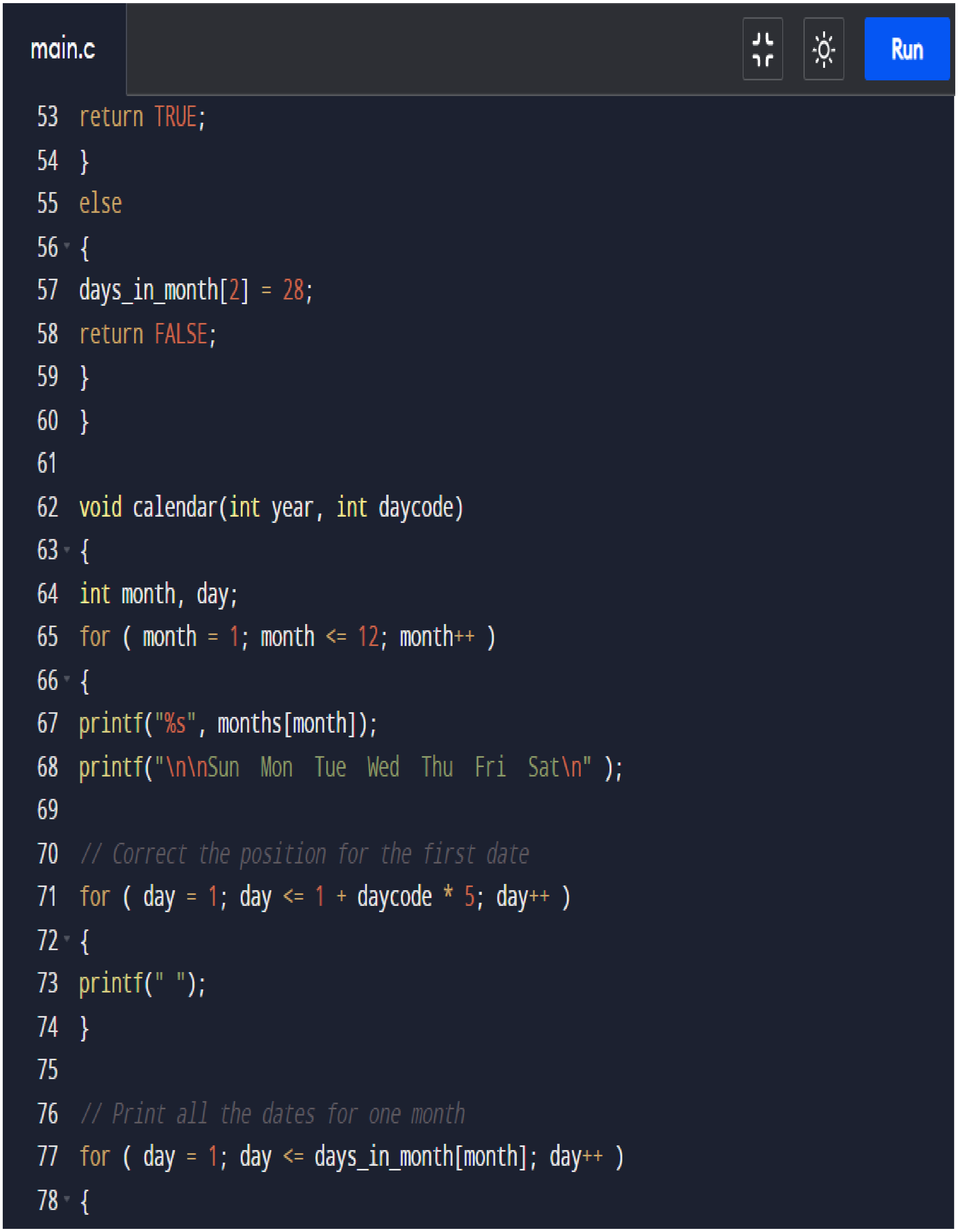
CODE IMPLEMENTATION



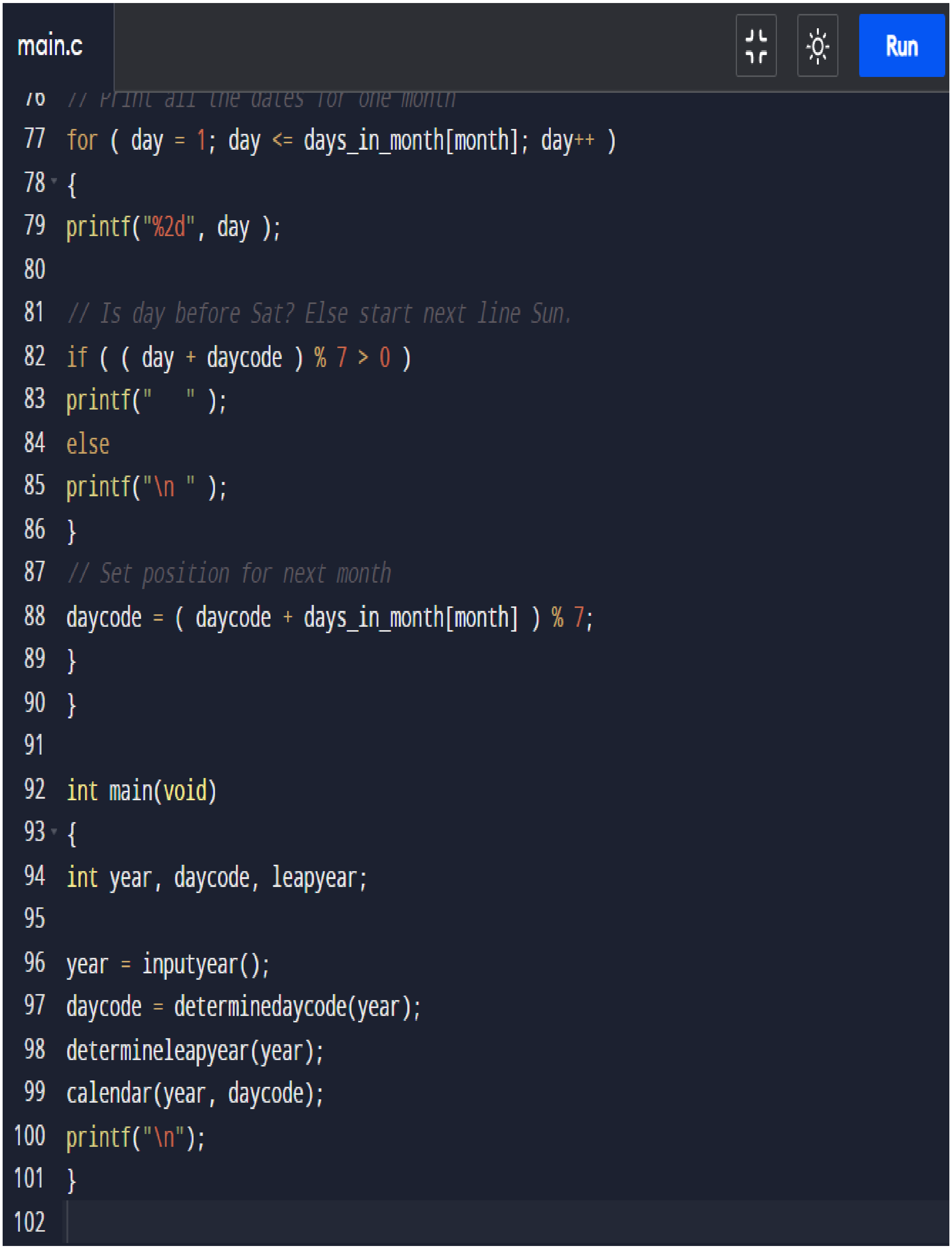
```c
// Online C compiler to run C program online
#include<stdio.h>

#define TRUE    1
#define FALSE   0

int days_in_month[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
char *months[]=
{
" ",
"\n\n\nJanuary",
"\n\n\nFebruary",
"\n\n\nMarch",
"\n\n\nApril",
"\n\n\nMay",
"\n\n\nJune",
"\n\n\nJuly",
"\n\n\nAugust",
"\n\n\nSeptember",
"\n\n\nOctober",
"\n\n\nNovember",
"\n\n\nDecember"
};


int inputyear(void)
```



```c
{
int year;

printf("Please enter a year (example: 1999) : ");
scanf("%d", &year);
return year;
}

int determinedaycode(int year)
{
int daycode;
int d1, d2, d3;

d1 = (year - 1.)/ 4.0;
d2 = (year - 1.)/ 100.;
d3 = (year - 1.)/ 400.;
daycode = (year + d1 - d2 + d3) %7;
return daycode;
}


int determineleapyear(int year)
{
if(year% 4 == FALSE && year%100 != FALSE || year%400 == FALSE)
{
days_in_month[2] = 29;
```

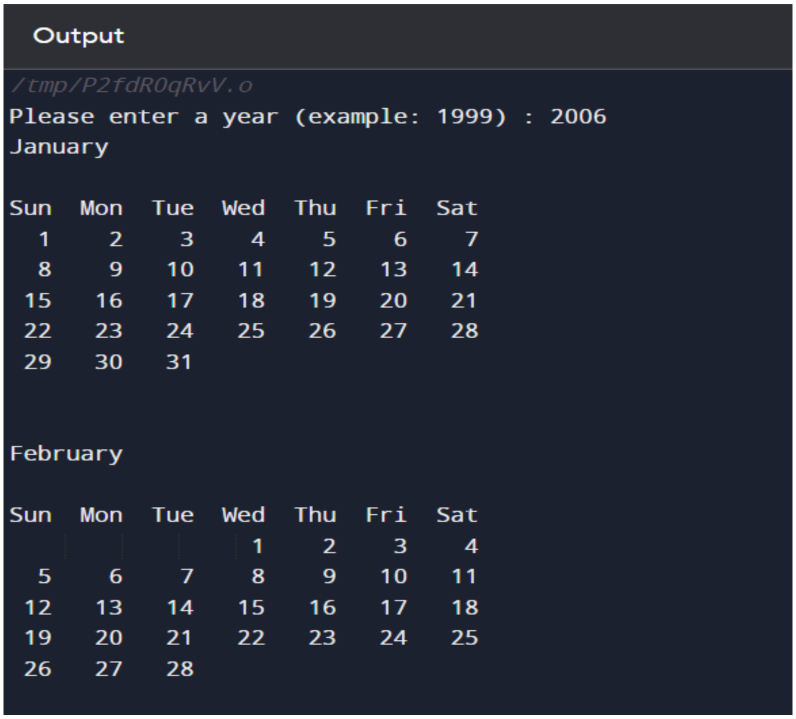


```c
return TRUE;
}
else
{
days_in_month[2] = 28;
return FALSE;
}
}

void calendar(int year, int daycode)
{
int month, day;
for ( month = 1; month <= 12; month++ )
{
printf("%s", months[month]);
printf("\n\nSun  Mon  Tue  Wed  Thu  Fri  Sat\n" );

// Correct the position for the first date
for ( day = 1; day <= 1 + daycode * 5; day++ )
{
printf(" ");
}

// Print all the dates for one month
for ( day = 1; day <= days_in_month[month]; day++ )
{
```

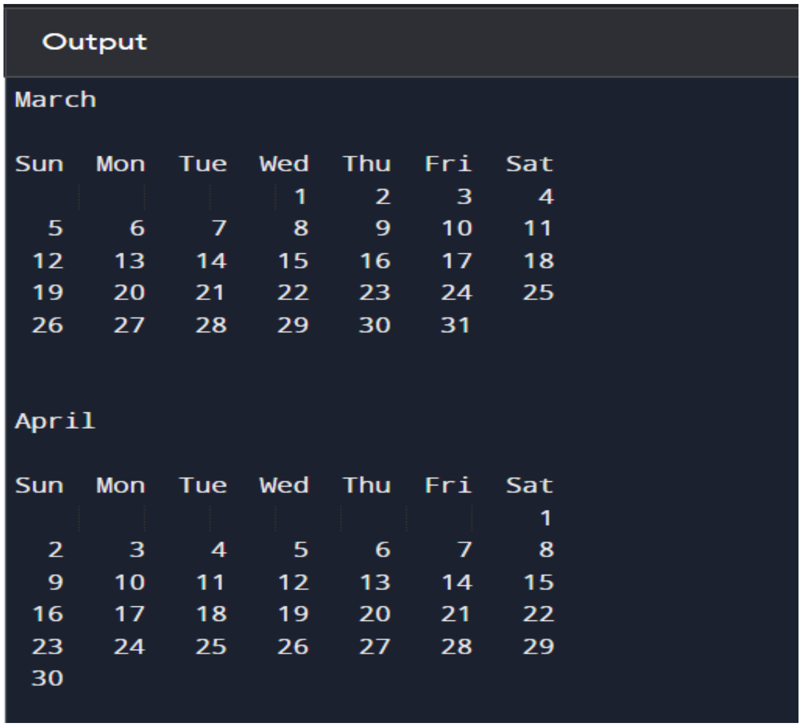


```c
// Print all the dates for one month
for ( day = 1; day <= days_in_month[month]; day++ )
{
printf("%2d", day );

// Is day before Sat? Else start next line Sun.
if ( ( day + daycode ) % 7 > 0 )
printf("   " );
else
printf("\n " );
}
// Set position for next month
daycode = ( daycode + days_in_month[month] ) % 7;
}
}

int main(void)
{
int year, daycode, leapyear;

year = inputyear();
daycode = determinedaycode(year);
determineleapyear(year);
calendar(year, daycode);
printf("\n");
}

```

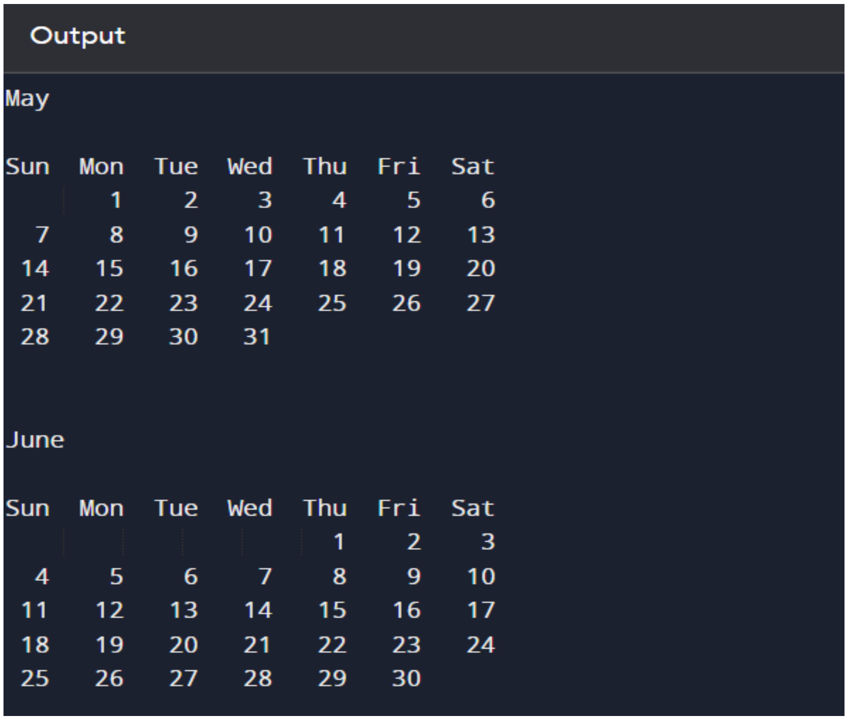
RESULT/OUTPUT



```
/tmp/P2fdR0qRvV.o
Please enter a year (example: 1999) : 2006
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
```

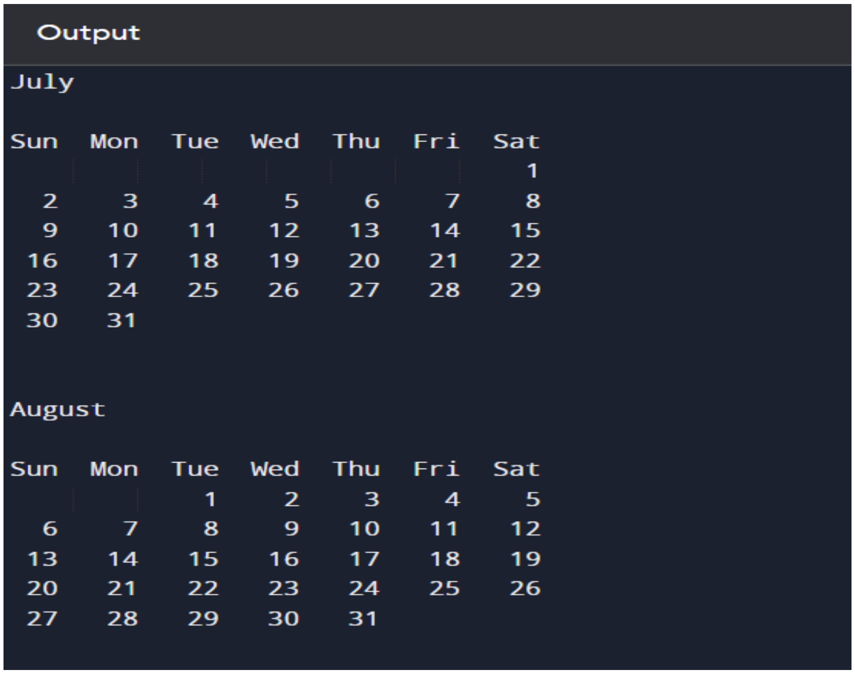


```
March

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
```



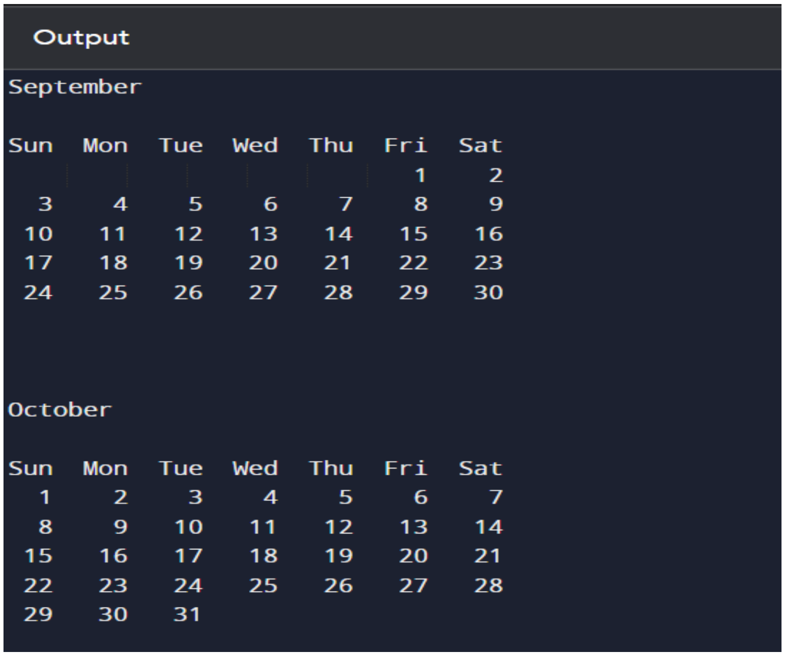
Output

```
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


June

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
```



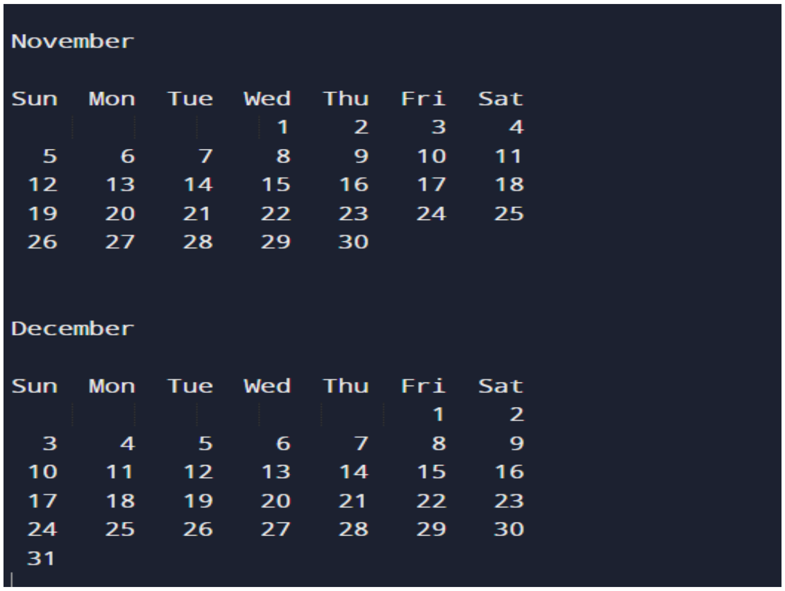
Output

```
July

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


August

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
```



Output

```
September

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



October

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
```



```
November

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


December

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
```

REFERENCES

1. https://flutterbyexample.com/lesson/calendar-app-introduction

2. https://www.geeksforgeeks.org/what-is-array