**1. INTRODUCTION**

**1.1 Problem Statement:**

1. Providing a wide range of calendar viewing options.

2. Scheduling the events using add note feature.

3. Day of mentioned date month and year is displayed.

**1.2 Scope:**

1. Future scope of application

This application can be easily implemented under various situations.we can add new features as and when we require. Reusability is possible as and when require in this application. There is flexibility in all the modules.

2. Software scope

a) Reusability

Reusability is possible as and when required in this application .We can update this application. Reusable software reduces design ,coding and testing cost by amortizing effort over several designs .Reducing the amount of code also implifies understanding ,which increases the likelyhood that the code is correct.

**1.3 Objectives:**

1. To create a calendar management tool which allows users to manage their events and customize the calendars appearance using windows properties.

2. It provides a very simple interface and displays days, dates, months , and years based on the input given by the user.

3. To apply file handling concept.

**2. LITERATURE SURVEY**

**2.1 Operations:**

Basically three operations can be done in this calendar application. To find out the day corresponding to a given date, the date, month and year are asked. You can list the days and dates of any month of any year. For example, entering 04 2014 (April 2014) will give you an output as shown in the screenshot in this post.

You can navigate the months using arrow keys, or press ‘n’ and ‘p’ keys to view the next and previous months respectively. The third feature of this C mini project on Calendar application utilizes file handling. With this feature, you can add important notes with corresponding dates.

The functions used in the source code are simple and easy to understand. The ones listed below have been used to produce background with colour effects. They are described in the source code with comments.

* void SetColor(intForgC)
* void ClearConsoleToColors(intForgC, intBackC)
* void SetColorAndBackground(intForgC, intBackC)

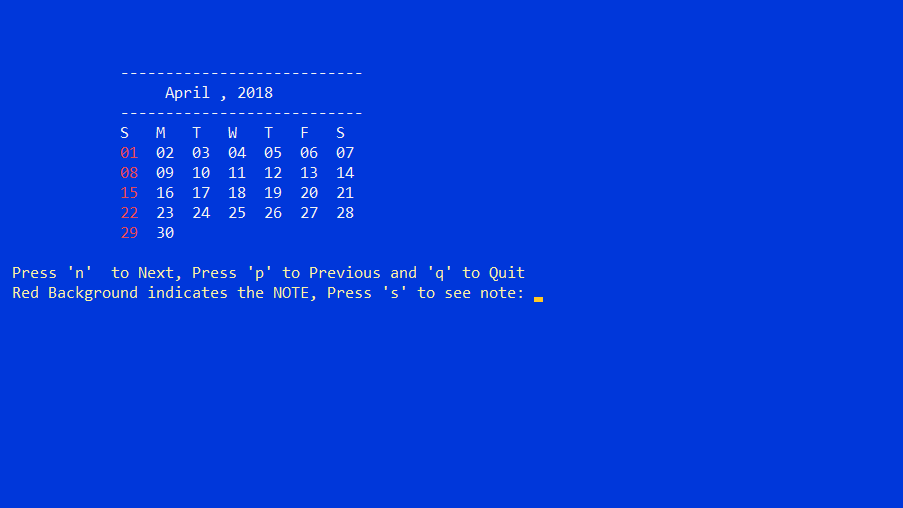
**voidgotoxy (int x, int y)** – You need to understand this function as it is an important one used in this Calendar in C language. You can find this function used in many C projects. This function allows you to print text in any place of screen. Using this function in Code::Blocks requires coding, but it can be directly used in Turbo C. Here is a code for this function in Code::Blocks.

Code for gotoxy (Mini Project in C Calendar Application)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | COORD coord={0,0};  // sets coordinates to (0,0) as global variables  voidgotoxy(intx,inty)  {          coord.X=x;coord.Y=y;// X and Y are the coordinates          SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE),coord);  } |

### Output Screenshot:

Days and Dates of a Month



**Fig-1 Displaying Calendar in output**

**2.2 Basics of File Handling in C:**

So far the operations using C program are done on a prompt / terminal which is not stored anywhere. But in the software industry, most of the programs are written to store the information fetched from the program. One such way is to store the fetched information in a file. Different operations that can be performed on a file are:

Creation of a new file (fopen with attributes as “a” or “a+” or “w” or “w++”)

Opening an existing file (fopen)

Reading from file (fscanf or fgetc)

Writing to a file (filePointerrintf or filePointeruts)

Moving to a specific location in a file (fseek, rewind)

Closing a file (fclose)

The text in the brackets denotes the functions used for performing those operations.

**2.3 Functions in File Operations:**

Opening or creating file:

For opening a file, fopen function is used with the required access modes. Some of the commonly used file access modes are mentioned below.

**2.4 File opening modes in C:**

“r” – Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. If the file cannot be opened fopen( ) returns NULL

“w” – Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.

“a” – Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer that points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.

“r+” – Searches file. If is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. Returns NULL, if unable to open the file.

“w+” – Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist a new file is created. Returns NULL, if unable to open file.

“a+” – Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.

As given above, if you want to perform operations on a binary file, then you have to append ‘b’ at the last. For example, instead of “w”, you have to use “wb”, instead of “a+” you have to use “a+b”. For performing the operations on the file, a special pointer called File pointer is used which is declared as

FILE \*filePointer;

So, the file can be opened as

filePointer = fopen(“fileName.txt”, “w”)

The second parameter can be changed to contain all the attributes listed in the above table.

**2.5 Reading from a file –**

The file read operations can be performed using functions fscanf or fgets. Both the functions performed the same operations as that of printf and gets but with an additional parameter, the file pointer. So, it depends on you if you want to read the file line by line or character by character.

And the code snippet for reading a file is as:

FILE \* filePointer;

filePointer = fopen(“fileName.txt”, “r”);

fscanf(filePointer, "%s %s %s %d", str1, str2, str3, &year);

**2.6 Writing a file –:**

The file write operations can be performed by the functions filePointerrintf and filePointeruts with similarities to read operations. The snippet for writing to a file is as :

FILE \*filePointer ;

filePointer = fopen(“fileName.txt”, “w”);

filePointerrintf(filePointer, "%s %s %s %d", "We", "are", "in", 2012);

**2.7Closing a file –:**

After every successful fie operations, you must always close a file. For closing a file, you have to use fclose function. The snippet for closing a file is given as :

FILE \*filePointer ;

filePointer= fopen(“fileName.txt”, “w”);

1. **SYSTEM REQUIRMENTS**

**3.1 Software Used:**

Compiler: GNU GCC Compiler

Command Prompt

Windows operating System

Code Blocks

**3.2 Hardware Used:**

Processor : INTEL® CORE™ i7-8750H

CPU Speed : 2.20 GHZ

RAM : 16 GB

Hard Disk : 512 GB

OS : Windows 10 Pro 64-Bit

Sound Card : No

Mouse : No

Keyboard : Yes

1. **SYSTEM ARCHTECTURE**

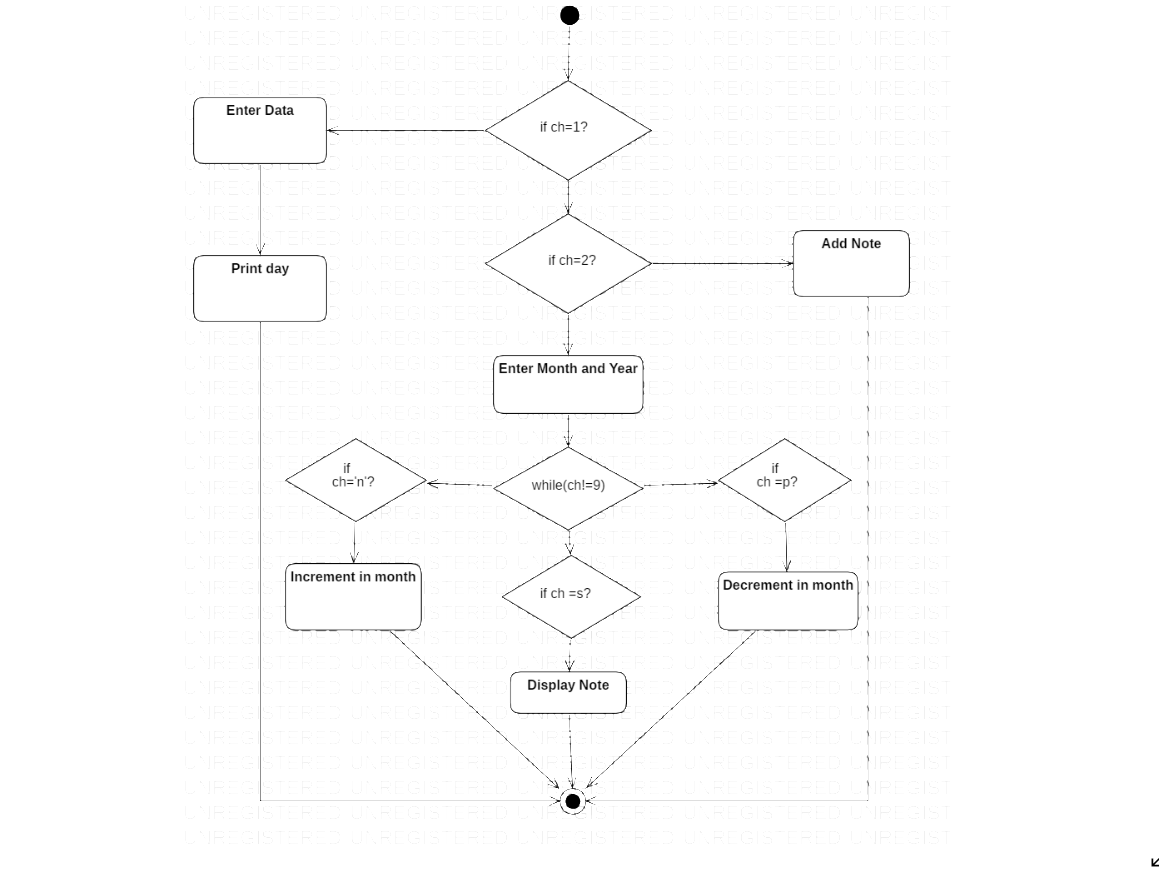
****

Figure 1- FLOW CHART

1. **IMPLEMENTATION**

**5.1 Source code:**

#include<stdio.h>

#include<conio.h>

#include<windows.h>

struct Date{

intdd;

int mm;

intyy;

};

struct Date date;

struct Remainder{

intdd;

int mm;

char note[50];

};

struct Remainder R;

COORD xy = {0, 0};

voidgotoxy (int x, int y)

{

xy.X = x; xy.Y = y; // X and Y coordinates

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), xy);

}

//This will set the forgroundcolor for printing in a console window.

voidSetColor(intForgC)

{

WORD wColor;

//We will need this handle to get the current background attribute

HANDLE hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_SCREEN\_BUFFER\_INFO csbi;

//We use csbi for the wAttributes word.

if(GetConsoleScreenBufferInfo(hStdOut, &csbi))

{

//Mask out all but the background attribute, and add in the forgourndcolor

wColor = (csbi.wAttributes& 0xF0) + (ForgC& 0x0F);

SetConsoleTextAttribute(hStdOut, wColor);

}

return;

}

voidClearColor(){

SetColor(15);

}

voidClearConsoleToColors(intForgC, intBackC)

{

WORD wColor = ((BackC& 0x0F) << 4) + (ForgC& 0x0F);

//Get the handle to the current output buffer...

HANDLE hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

//This is used to reset the carat/cursor to the top left.

COORD coord = {0, 0};

//A return value... indicating how many chars were written

// not used but we need to capture this since it will be

// written anyway (passing NULL causes an access violation).

DWORD count;

//This is a structure containing all of the console info

// it is used here to find the size of the console.

CONSOLE\_SCREEN\_BUFFER\_INFO csbi;

//Here we will set the current color

SetConsoleTextAttribute(hStdOut, wColor);

if(GetConsoleScreenBufferInfo(hStdOut, &csbi))

{

//This fills the buffer with a given character (in this case 32=space).

FillConsoleOutputCharacter(hStdOut, (TCHAR) 32, csbi.dwSize.X \* csbi.dwSize.Y, coord, &count);

FillConsoleOutputAttribute(hStdOut, csbi.wAttributes, csbi.dwSize.X \* csbi.dwSize.Y, coord, &count );

//This will set our cursor position for the next print statement.

SetConsoleCursorPosition(hStdOut, coord);

}

return;

}

voidSetColorAndBackground(intForgC, intBackC)

{

WORD wColor = ((BackC& 0x0F) << 4) + (ForgC& 0x0F);;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), wColor);

return;

}

intcheck\_leapYear(int year){ //checks whether the year passed is leap year or not

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

return 1;

return 0;

}

voidincrease\_month(int \*mm, int \*yy){ //increase the month by one

++\*mm;

if(\*mm > 12){

++\*yy;

\*mm = \*mm - 12;

}

}

voiddecrease\_month(int \*mm, int \*yy){ //decrease the month by one

--\*mm;

if(\*mm < 1){

--\*yy;

if(\*yy<1600){

printf("No record available");

return;

}

\*mm = \*mm + 12;

}

}

intgetNumberOfDays(intmonth,int year){ //returns the number of days in given month

switch(month){ //and year

case 1 : return(31);

case 2 : if(check\_leapYear(year)==1)

return(29);

else

return(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(-1);

}

}

char \*getName(int day){ //returns the name of the day

switch(day){

case 0 :return("Sunday");

case 1 :return("Monday");

case 2 :return("Tuesday");

case 3 :return("Wednesday");

case 4 :return("Thursday");

case 5 :return("Friday");

case 6 :return("Saturday");

default:return("Error in getName() module.Invalid argument passed");

}

}

voidprint\_date(int mm, intyy){ //prints the name of month and year

printf("---------------------------\n");

gotoxy(25,6);

switch(mm){

case 1: printf("January"); break;

case 2: printf("February"); break;

case 3: printf("March"); break;

case 4: printf("April"); break;

case 5: printf("May"); break;

case 6: printf("June"); break;

case 7: printf("July"); break;

case 8: printf("August"); break;

case 9: printf("September"); break;

case 10: printf("October"); break;

case 11: printf("November"); break;

case 12: printf("December"); break;

}

printf(" , %d", yy);

gotoxy(20,7);

printf("---------------------------");

}

intgetDayNumber(intday,intmon,int year){ //retuns the day number

int res = 0, t1, t2, y = year;

year = year - 1600;

while(year >= 100){

res = res + 5;

year = year - 100;

}

res = (res % 7);

t1 = ((year - 1) / 4);

t2 = (year-1)-t1;

t1 = (t1\*2)+t2;

t1 = (t1%7);

res = res + t1;

res = res%7;

t2 = 0;

for(t1 = 1;t1 <mon; t1++){

t2 += getNumberOfDays(t1,y);

}

t2 = t2 + day;

t2 = t2 % 7;

res = res + t2;

res = res % 7;

if(y > 2000)

res = res + 1;

res = res % 7;

return res;

}

char \*getDay(intdd,intmm,intyy){

int day;

if(!(mm>=1 && mm<=12)){

return("Invalid month value");

}

if(!(dd>=1 &&dd<=getNumberOfDays(mm,yy))){

return("Invalid date");

}

if(yy>=1600){

day = getDayNumber(dd,mm,yy);

day = day%7;

return(getName(day));

}else{

return("Please give year more than 1600");

}

}

intcheckNote(intdd, int mm){

FILE \*fp;

fp = fopen("note.dat","rb");

if(fp == NULL){

printf("Error in Opening the file");

}

while(fread(&R,sizeof(R),1,fp) == 1){

if(R.dd == dd&& R.mm == mm){

fclose(fp);

return 1;

}

}

fclose(fp);

return 0;

}

voidprintMonth(intmon,intyear,intx,int y){ //prints the month with all days

int nod, day, cnt, d = 1, x1 = x, y1 = y, isNote = 0;

if(!(mon>=1 &&mon<=12)){

printf("INVALID MONTH");

getch();

return;

}

if(!(year>=1600)){

printf("INVALID YEAR");

getch();

return;

}

gotoxy(20,y);

print\_date(mon,year);

y += 3;

gotoxy(x,y);

printf("S M T W T F S ");

y++;

nod = getNumberOfDays(mon,year);

day = getDayNumber(d,mon,year);

switch(day){ //locates the starting day in calender

case 0 :

x=x;

cnt=1;

break;

case 1 :

x=x+4;

cnt=2;

break;

case 2 :

x=x+8;

cnt=3;

break;

case 3 :

x=x+12;

cnt=4;

break;

case 4 :

x=x+16;

cnt=5;

break;

case 5 :

x=x+20;

cnt=6;

break;

case 6 :

x=x+24;

cnt=7;

break;

default :

printf("INVALID DATA FROM THE getOddNumber()MODULE");

return;

}

gotoxy(x,y);

if(cnt == 1){

SetColor(12);

}

if(checkNote(d,mon)==1){

SetColorAndBackground(15,12);

}

printf("%02d",d);

SetColorAndBackground(15,1);

for(d=2;d<=nod;d++){

if(cnt%7==0){

y++;

cnt=0;

x=x1-4;

}

x = x+4;

cnt++;

gotoxy(x,y);

if(cnt==1){

SetColor(12);

}else{

ClearColor();

}

if(checkNote(d,mon)==1){

SetColorAndBackground(15,12);

}

printf("%02d",d);

SetColorAndBackground(15,1);

}

gotoxy(8, y+2);

SetColor(14);

printf("Press 'n' to Next, Press 'p' to Previous and 'q' to Quit");

gotoxy(8,y+3);

printf("Red Background indicates the NOTE, Press 's' to see note: ");

ClearColor();

}

voidAddNote(){

FILE \*fp;

fp = fopen("note.dat","ab+");

system("cls");

gotoxy(5,7);

printf("Enter the date(DD/MM): ");

scanf("%d%d",&R.dd, &R.mm);

gotoxy(5,8);

printf("Enter the Note(50 character max): ");

fflush(stdin);

scanf("%[^\n]",R.note);

if(fwrite(&R,sizeof(R),1,fp)){

gotoxy(5,12);

puts("Note is saved sucessfully");

fclose(fp);

}else{

gotoxy(5,12);

SetColor(12);

puts("\aFail to save!!\a");

ClearColor();

}

gotoxy(5,15);

printf("Press any key............");

getch();

fclose(fp);

}

voidshowNote(int mm){

FILE \*fp;

int i = 0, isFound = 0;

system("cls");

fp = fopen("note.dat","rb");

if(fp == NULL){

printf("Error in opening the file");

}

while(fread(&R,sizeof(R),1,fp) == 1){

if(R.mm == mm){

gotoxy(10,5+i);

printf("Note %d Day = %d: %s", i+1, R.dd, R.note);

isFound = 1;

i++;

}

}

if(isFound == 0){

gotoxy(10,5);

printf("This Month contains no note");

}

gotoxy(10,7+i);

printf("Press any key to back.......");

getch();

}

int main(){

ClearConsoleToColors(15, 1);

SetConsoleTitle("Calender Project - Programming-technique.blogspot.com");

int choice;

charch = 'a';

while(1){

system("cls");

printf("1. Find Out the Day\n");

printf("2. Print all the day of month\n");

printf("3. Add Note\n");

printf("4. EXIT\n");

printf("ENTER YOUR CHOICE : ");

scanf("%d",&choice);

system("cls");

switch(choice){

case 1:

printf("Enter date (DD MM YYYY) : ");

scanf("%d %d %d",&date.dd,&date.mm,&date.yy);

printf("Day is : %s",getDay(date.dd,date.mm,date.yy));

printf("\nPress any key to continue......");

getch();

break;

case 2 :

printf("Enter month and year (MM YYYY) : ");

scanf("%d %d",&date.mm,&date.yy);

system("cls");

while(ch!='q'){

printMonth(date.mm,date.yy,20,5);

ch = getch();

if(ch == 'n'){

increase\_month(&date.mm,&date.yy);

system("cls");

printMonth(date.mm,date.yy,20,5);

}else if(ch == 'p'){

decrease\_month(&date.mm,&date.yy);

system("cls");

printMonth(date.mm,date.yy,20,5);

}else if(ch == 's'){

showNote(date.mm);

system("cls");

}

}

break;

case 3:

AddNote();

break;

case 4 :

exit(0);

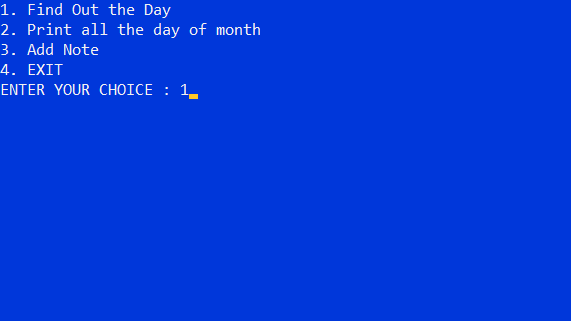
}

}

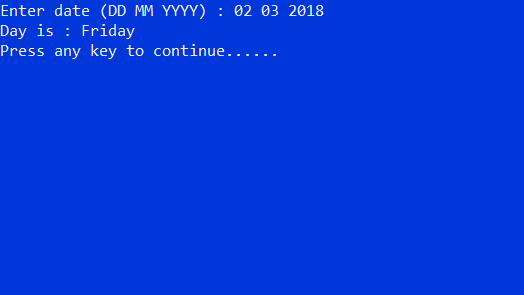
return 0;

}

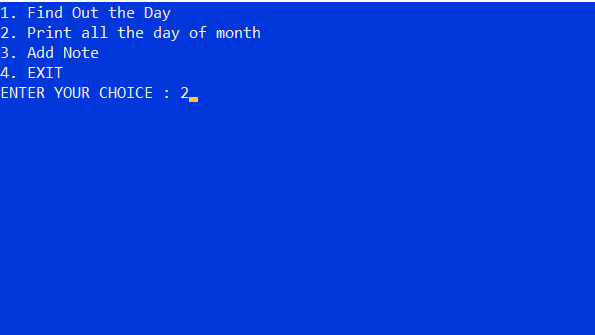
1. **TESTING**

****

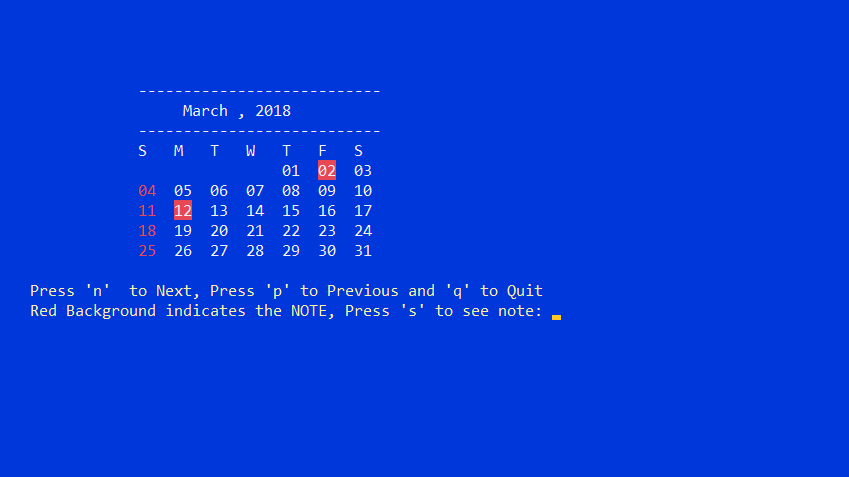
**Fig-3 Calendar Menu**

****

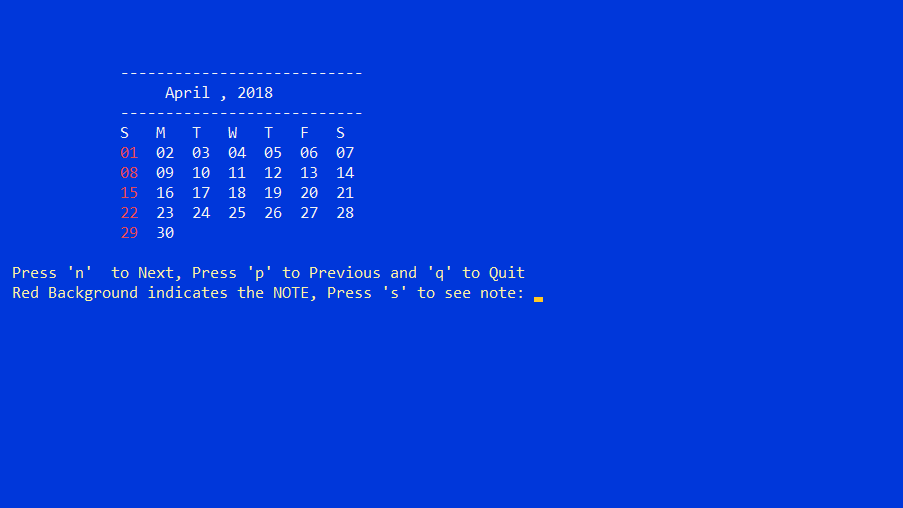
**Fig-4 Dispalying Day of the given Date**

****

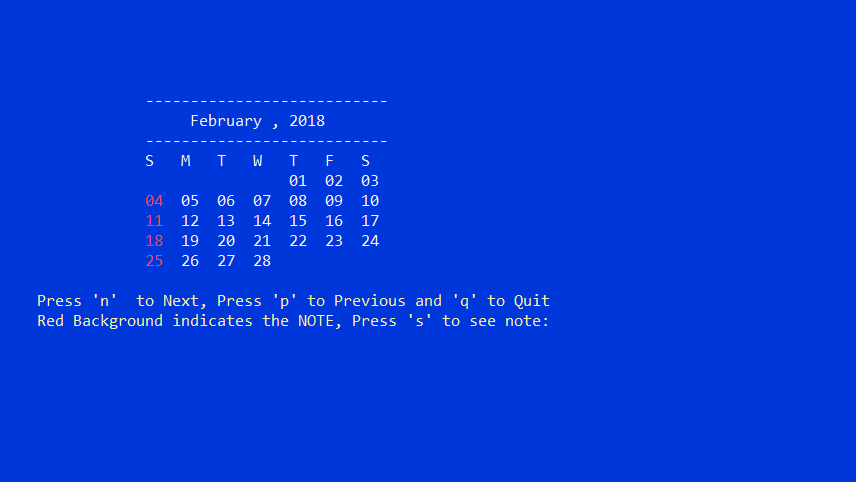
**Fig-5 Selecting Choice from Menu**

****

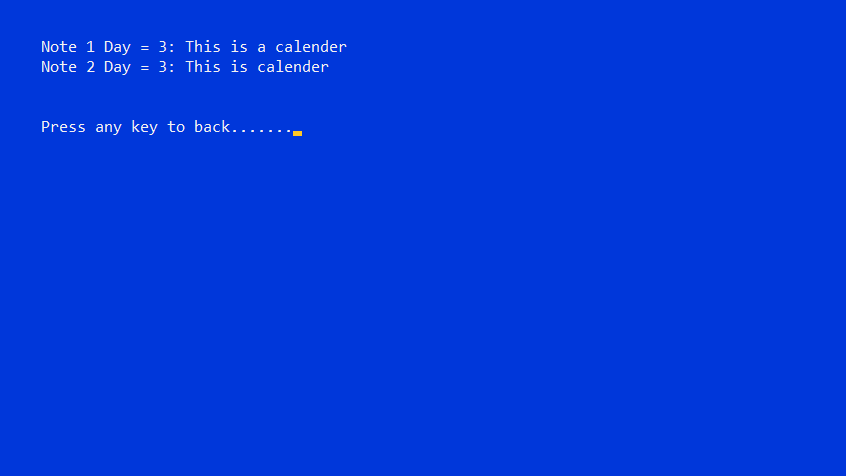
**Fig-6 Dispalying Calendar of given Month**

****

**Fig-7 Dispalying Calendar next to the given Month**

****

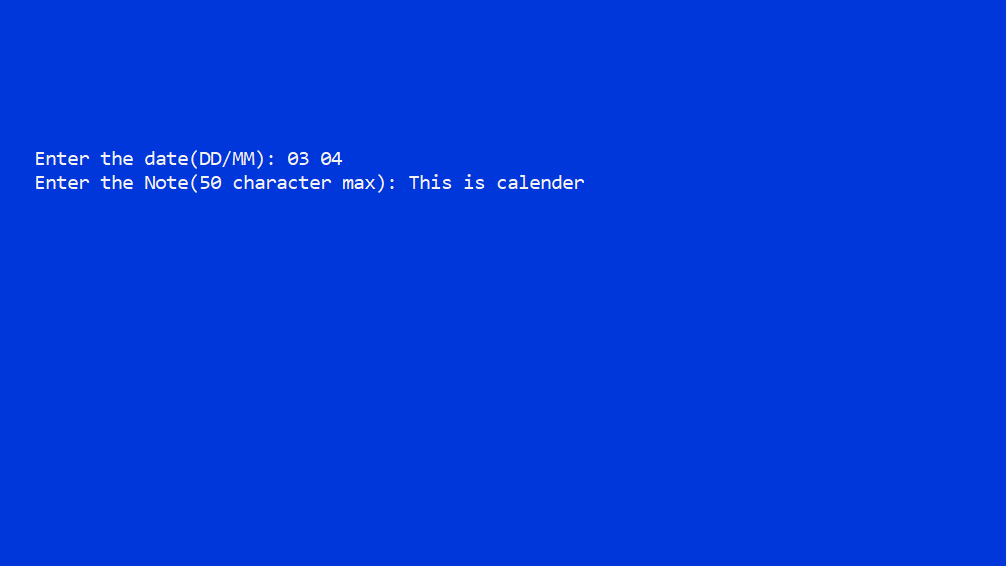
**Fig-8 Dispalying calendar previous to given Month**

****

**Fig-9 Saving Note**

****

**Fig-10 Adding a Note to be Saved**

****

**Fig-11 Dispalying the Saved Note**

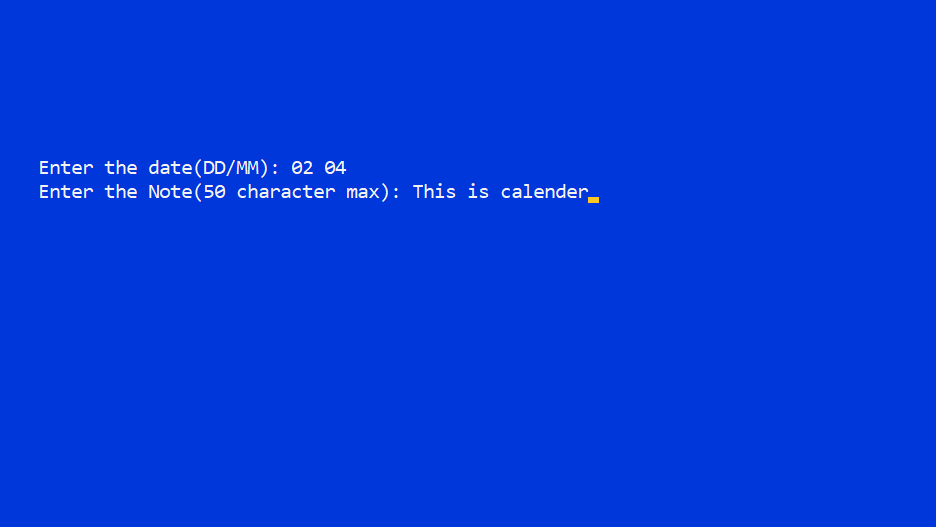
****

Fig-12 Displaying the Saved Note 1

1. **CONCLUSION**

This application can be implemented under various situations .we can add new features as and when required.re usability is possible as when required in this application .this flexible in all modules.

This was an effort to develop a simple calendar application which may be useful to find date day, month and year and is useful in adding Notes.

**REFERENCES:**

**1.** <https://www.geeksforgeeks.org/basics-file-handling-c/>

**2**. C programming and data structures-BEHROUZ A FOROUZAN