INTRODUCTION

Calendar utility basically helps users to note the important events, navigate among months and viewing date or day or month or year .

Requirements for calendar :

**HIGH LEVEL REQIREMENTS**

* We can add important notes with corresponding dates.
* Finding and listing out the day corresponding to a given date, the date, month and year needed can be done.
* Calendar application utilizes file handling.
* Navigation can be done .
* These are the features done in this utility.

**LOW LEVEL REQUIREMENTS**

* We should be aware of the data we want with respect to the date ,day, month and year .
* We can list the days and dates of any month of any year. For example, entering 03 2019 (March 2019) will give an output
* We can navigate the months using arrow keys, or press ‘n’ and ‘p’ keys to view the next and previous months respectively.
* With the file handling feature, we can add important notes with corresponding dates.

**SWOT ANALYSIS**

STRENGTH -we can find ,list, add or navigate among the date ,month and year in the calendar.

Marking important dates is also possible .

WEEKNESS- The proper data to be accessed in the calendar should be known or else it will result in error.

OPPURTUNITIES -By marking important dates and having a routine check on calendars will help us in discharging our duties on date.

THREAT -If we don’t access the features correctly it will lead to error hence creating a collapsed environment and not enabling us to deliver our duties on date .

5W & 1H

WHAT- This is a project which allow us to have a note on important dates , finding the dates or months or year.

WHY - This is to help the users with all the calendar activities and maintaing a scheduled routine.

WHO - people who work continuously with different or same tasks that have to be done on regular basis are benefited from Calendar.

WHOM -Students ,people working in corporate ,business, product based etc can be benefited

WHEN -Calendar is more useful when a deal or event or tasks are scheduled

HOW -By using all the features we can handle tasks in orderly manner ,missing the activites will be less hence achieving better results.

**FUNCTION USED IN CODE**

* void gotoxy (int x, int y)

**SOURCE CODE**

#include<stdio.h>

#include<conio.h>

struct Date{

int d;

int m;

int y;

};

struct Date date;

struct Remainder{

int d;

int m;

char note[50];

};

struct Remainder R;

int check\_leapYear(int year){

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

return 1;

return 0;

}

void increase\_month(int \*m, int \*y){

++\*m;

if(\*m > 12){

++\*y;

\*m = \*m - 12;

}

}

void decrease\_month(int \*m, int \*y){

--\*m;

if(\*m < 1){

--\*y;

if(\*y<1600){

printf("No record available");

return;

}

\*m = \*m + 12;

}

}

int getNumberOfDays(int month,int year){

switch(month){

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){

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");

}

}

void print\_date(int m, int y){

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

switch(m){

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", y);

printf("---");

}

int getDayNumber(int day,int mon,int year){

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(int d,int m,int y){

int day;

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

return("Invalid month value");

}

if(!(d>=1 && d<=getNumberOfDays(m,y))){

return("Invalid date");

}

if(y>=1600){

day = getDayNumber(d,m,y);

day = day%7;

return(getName(day));

}else{

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

}

}

int checkNote(int d, int m){

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.d == d && R.m == m){

fclose(fp);

return 1;

}

}

fclose(fp);

return 0;

}

void printMonth(int mon,int year,int x,int y){

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;

}

print\_date(mon,year);

y += 3;

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

y++;

nod = getNumberOfDays(mon,year);

day = getDayNumber(d,mon,year);

switch(day){

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;

}

}

void AddNote(){

FILE \*fp;

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

printf("Enter the date(D/M): ");

scanf("%d%d",&R.d, &R.m);

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

fflush(stdin);

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

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

puts("Note is saved sucessfully");

fclose(fp);

}else{

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

}

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

getch();

fclose(fp);

}

void showNote(int m){

FILE \*fp;

int i = 0, isFound = 0;

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

if(fp == NULL){

printf("Error in opening the file");

}

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

if(R.m == m){

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

isFound = 1;

i++;

}

}

if(isFound == 0){

printf("This Month contains no note");

}

printf("Press p to go to previous ");

printf("Press n to go to next");

getch();

}

int main(){

int choice;

char ch = 'a';

while(1){

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);

switch(choice){

case 1:

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

scanf("%d %d %d",&date.d,&date.m,&date.y);

printf("Day is : %s",getDay(date.d,date.m,date.y));

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

getch();

break;

case 2 :

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

scanf("%d %d",&date.m,&date.y);

while(ch!='q'){

printMonth(date.m,date.y,20,5);

ch = getch();

if(ch == 'n'){

increase\_month(&date.m,&date.y);

printMonth(date.m,date.y,20,5);

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

decrease\_month(&date.m,&date.y);

printMonth(date.m,date.y,20,5);

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

showNote(date.m);

}

}

break;

case 3:

AddNote();

break;

}

}

return 0;

}

**Best methods followed**

* Used functions to decrease dependency on main function.
* Used structures and arrays and sorting algorithms to accept the inputs from user.
* Printf statements have been placed only wherever necessary to avoid confusions.
* Created header file so that the fuctions can be used else where ever required without any difficulty.
* Unit testing is done to avoid any computational errors.