**INTEGRATED PROJECT REPORT**

**On**

**CALENDAR USING GRAPHICS IN C**

**Submitted in partial fulfillment of the requirement for the**

**Course Integrated Project III (CSP2208) of**

**COMPUTER SCIENCE AND ENGINEERING**

**B.E. Batch-2015**

**in**

**May-2017**

****

**Submitted By: (Group G-219)**

Name- Prashant Jaswal

Roll. No. 1510991449

Name- Pulkit Lamba

Roll. No.1510991472

Name- Puneet Chopra

Roll. No. 151099175

Name- Puneet Singh

Roll. No. 1510991478

**DEPARTMENT OF COMPUTER SCIENCES**

**CHITKARA UNIVERSITY**

**PUNJAB**

**CERTIFICATE**

This is to be certified that the project entitled “Calendar using C graphics” has been submitted for the Bachelor of Computer Science Engineering at Chitkara University, Punjab during the academic semester January 2017- May-2017 is a bonafide piece of project work carried out by “Prashant(1510991449),Pulkit(1510991472), Puneet(1510991475), Puneet(1510991478) ” students of group(G-219) towards the partial fulfillment for the award of the course Integrated Project (CSP2208) under the guidance and supervision of “Mrs. Priyanka Gupta”

Sign. of Group Incharge :

Mrs. Priyanka Gupta

(Designation & Department)

**CANDIDATE’S DECLARATION**

We, “Prashant (1510991449), Pulkit (1510991472), Puneet (1510991475), Puneet (1510991478) students of group(G-219)”, B.E.-2015 of the Chitkara University, Punjab hereby declare that the Integrated Project Report entitled “Calendar using C graphics” is an original work and data provided in the study is authentic to the best of our knowledge. This report has not been submitted to any other Institute for the award of any other course.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sign. of Student 1 | Sign. of Student 2 | Sign. of Student 3 | Sign. of Student 4 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Prashant Jaswal | Pulkit Lamba | Puneet Chopra | Puneet Singh |  |  |  |  |  |  |  |  |  |  |
| ID No-1510991449 | ID No-1510991472 | ID No-1510991475 | ID No-1510991478 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Place:

Date:

**ACKNOWLEDGEMENT**

It is our pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior and acts during the course of study.

We express our sincere gratitude to all for providing me an opportunity to undergo Integrated Project as the part of the curriculum.

We are thankful to “Mrs. Priyanka Gupta” for her support, cooperation, and motivation provided to us during the training for constant inspiration, presence and blessings.

We also extend our sincere appreciation to *“*Mrs. Priyanka Gupta” who provided her valuable suggestions and precious time in accomplishing our Integrated project report.

Lastly, We would like to thank the almighty and our parents for their moral support and friends with whom we shared our day-to day experience and received lots of suggestions that improve our quality of work.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Prashant Jaswal | Pulkit Lamba | Puneet Chopra | Puneet Singh |  |  |  |  |  |  |  |  |  |  |
| ID No-1510991449 | ID No-1510991472 | ID No-1510991475 | ID No-1510991478 |  |  |  |  |  |  |  |  |  |  |

**ABSTRACT**

This project is concerned with building and programming a Calendar Tool so that users can view and manage their appointments quickly, reliably and electronically. This program can exchange important data about events so that people can schedule meetings with anyone who is using a similar kind of program.

The Calendar Application presented here is a very simple console application developed using C programming language. It is built using graphics properties, and utilizes many windows properties to give the application a colour look and feel. This application contains customized colour for different events like holidays, reminder etc. Any user can pull out the information from the application and is very easy to use. Furthermore, this application will only visualize reminders for the user events.

This Project utilizes various aspects of C language such as graphics, functions, arrays, pointers, file handling, and data structure. With this simple application users can perform all basic calendar related operations and features.

**INTRODUCTION**

Like many, I also started computer programming with **C language** which is one of the most widely used programming languages of all time.

**Introduction to C programming language:**

The C programming language is a computer programming language that was developed to do system programming for the operating system UNIX and is an imperative programming language. C was developed in the early 1970 by Ken Thompson and Dennis Ritchie. It is a procedural language, which means that people can write their programs as a series of step-by-step instructions. C is a compiled language.

Because the ideas behind C are kept close to the design of the computer, the compiler (program builder) can generate machine code/native code for the computer. Programs built in machine code are very fast. This makes C a good language for writing operating systems. Many operating systems, including Linux and UNIX, are programmed using this language. The language itself has very few keywords, and most things are done using libraries, which are collections of code for them to be reused.

C is available for many different types of computers. This is why C is called a "portable" language. A program that is written in C and that respects certain limitations can be compiled for many different platforms.

The syntax of C has also influenced many other programming languages, such as C++, C#, and Java, and many more programming languages we use nowadays.

**Calendar:**

A Calendar is a tool used to decipher the time of an event and for planning things that have not happened yet. This is done using days, called calendar dates. The dates are usually made based on how things in the sky seem to move. The year and month are based on motions of the Sun and moon. By knowing what day something happened or will happen, people have an easier life.

Calendar systems have a beginning time or era. The era is often a religion-based event, such as the birth of Jesus, but it may be a political event such as a coronation or the founding of a state. Dates that are counted from the coronation are called "regnal".

The calendar most used today is the Gregorian. The Jewish and Islamic calendars and other calendar systems from many different parts of the world are also used.

A full calendar system has a different calendar date for every day. Thus, the week cycle is by itself not a full calendar system; neither is a system to name the days within a year without a system for identifying the years.

The simplest calendar system just counts time periods from a reference date. This applies for the Julian day or Unix Time. Virtually the only possible variation is using a different reference date, in particular one less distant in the past to make the numbers smaller. Computations in these systems are just a matter of addition and subtraction.

**Determination of the day of the week:**

There are various methods to calculate the day of the week for any particular date in the past or future. These methods ultimately rely on algorithms to determine the day of the week for any given date, including those based solely on tables as found in perpetual calendars that require no calculations to be performed by the user. A typical application is to calculate the day of the week on which someone was born or any other specific event occurred.

To determine the day of the week from numerical operations, Sunday through Saturday are represented as numbers which may, corresponding to Sunday being the first day of the week, be 1 to 7 (or 0, rejecting whole sevens) respectively, which is equivalent to ISO 8601's alternative usage of 1 = Monday to 7 = Sunday). This is achieved with arithmetic modulo 7. Modulo 7 is an operation that calculates the remainder of a number being divided by 7. Thus, the number 7 is treated as 0, 8 as 1, 9 as 2, 18 as 4 and so on; the interpretation of this being that if Sunday is signified as day 1, then 7 days later (i.e. day 8) is also a Sunday, and day 18 will be the same as day 4, which is a Wednesday since this falls three days after Sunday.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Standard | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday | Usage examples |
| ISO 8601 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | %\_ISODOWI%, %@ISODOWI[]% ([4DOS](https://en.wikipedia.org/wiki/4DOS)); DAYOFWEEK() ([HP Prime](https://en.wikipedia.org/wiki/HP_Prime)) |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |  |
|  | 2 | 3 | 4 | 5 | 6 | 7 | 1 | [%NDAY OF WEEK%](https://en.wikipedia.org/wiki/%25NDAY_OF_WEEK%25) ([NetWare](https://en.wikipedia.org/wiki/NetWare), [DR-DOS](https://en.wikipedia.org/wiki/DR-DOS)); %\_DOWI%, %@DOWI[]% ([4DOS](https://en.wikipedia.org/wiki/4DOS)) |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 0 | HP financial calculators |

The basic approach of nearly all of the methods to calculate the day of the week begins by starting from an ‘anchor date’: a known pair (such as January 1, 1800 as a Wednesday), determining the number of days between the known day and the day that you are trying to determine, and using arithmetic modulo 7 to find a new numerical day of the week.

One standard approach is to look up (or calculate, using a known rule) the value of the first day of the week of a given century, look up (or calculate, using a method of congruence) an adjustment for the month, calculate the number of leap years since the start of the century, and then add these together along with the number of years since the start of the century, and the day number of the month. Eventually, one ends up with a day-count to which one applies modulo 7 to determine the day of the week of the date.

Some methods do all the additions first and then cast out sevens, whereas others cast them out at each step, as in Lewis Carroll's method. Either way is quite viable: the former is easier for calculators and computer programs; the latter for mental calculation (it is quite possible to do all the calculations in one's head with a little practice). None of the methods given here perform range checks, so unreasonable dates will produce erroneous results.

In this project, we used Zeller’s algorithm to determine the day of the week.

**About the project:**

Before 1980s, the personal computers are not usually used by people. People have to record information about events on paper or remember them in mind, and if they want to schedule a meeting, they have to compare the timetables of attendees or even have to make a phone call to all attendees to check suitable meeting time. So that wastes not only labor power and also material resources. The efficiency of working was decreasing a lot. Advancement in computer engineering and information technology has reached a state in the past decade that leads to the widespread ownership of personal computers worldwide, with an ever-increasing population of internet users, carrying out numerous types of transactions online. Thus, some calendar tools produced are used to help people ensure that information can be quickly and reliably stored and organized. What’s more, people can exchange calendaring and scheduling information in an easy and automated manner. It is the objective of this project to developing an all-in-one software that is able to store information about events and handle information in many data formats. Program can exchange important data about events so that users can schedule meetings with anyone else who has a same date format aware program.

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 05 2017 (May 2017) will give you an output as shown in the screenshot in chart. You can navigate the months using arrow keys, and national and public holidays of each month is marked respectively. Also, you can press ‘R’ key to change the color of date i.e. to mark that date an important date. The third feature of this project on Calendar application utilizes file handling. With this feature, you can add text file to print the current calendar.

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

**SOFTWARE AND HARDWARE REQUIREMENT SPECIFICATION**

Before you can start programming in C, you will need a C compiler. A compiler is a program that converts C code into executable machine code.

The minimum software requirements to program in C is a text editor, as opposed to a word processor. A plain text Notepad Editor can be used but it does not offer any advanced capabilities such as code completion or debugging. There are many text editors (see List of Text Editors), among the most popular are Notepad++ for Windows, Atom, Sublime Text, Vim and Emacs are also available cross-platform. These text editors come with syntax highlighting and line numbers, which makes code easier to read at a glance, and to spot syntax errors.

Though not absolutely needed, many programmers prefer and recommend using an Integrated development environment (IDE) instead of a text editor. An IDE is a suite of programs that developers need, combined into one convenient package, usually with a graphical user interface. These programs include a text editor, linker, project management and sometimes bundled with a compiler. They also typically include a debugger, a tool that will preserve your C source code after compilation and enable you to do such things as step through it manually, or alter data as an aid to finding and correcting programming errors.

For beginners, it is recommended not to use an IDE, since it hides most of what is going on. Using the command line builds up familiarity with the toolchain. An IDE may be useful to somebody with programming experience but knows how the IDE works. So, as a general guideline: Do not use an IDE unless you know what the IDE does!

IDE Used by us :

**CodeBlocks:**

Code::Blocks is a [free](https://en.wikipedia.org/wiki/Free_software), [open-source](https://en.wikipedia.org/wiki/Open-source_software) [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [IDE](https://en.wikipedia.org/wiki/Integrated_development_environment) that supports multiple [compilers](https://en.wikipedia.org/wiki/Compilers) including [GCC](https://en.wikipedia.org/wiki/GNU_Compiler_Collection), [Clang](https://en.wikipedia.org/wiki/Clang) and [Visual C++](https://en.wikipedia.org/wiki/Visual_C%2B%2B). It is developed in [C++](https://en.wikipedia.org/wiki/C%2B%2B) using [wxWidgets](https://en.wikipedia.org/wiki/WxWidgets) as the [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface) toolkit. Using a plugin architecture, its capabilities and features are defined by the provided plugins. Currently, Code::Blocks is oriented towards [C](https://en.wikipedia.org/wiki/C_(programming_language)), [C++](https://en.wikipedia.org/wiki/C%2B%2B), and [Fortran](https://en.wikipedia.org/wiki/Fortran). It has a custom [build system](https://en.wikipedia.org/wiki/Build_automation) and optional [Make](https://en.wikipedia.org/wiki/Make_(software)) support.

Code::Blocks is being developed for [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [Linux](https://en.wikipedia.org/wiki/Linux), and [macOS](https://en.wikipedia.org/wiki/MacOS) and has been ported to [FreeBSD](https://en.wikipedia.org/wiki/FreeBSD), [OpenBSD](https://en.wikipedia.org/wiki/OpenBSD) and [Solaris](https://en.wikipedia.org/wiki/Solaris_(operating_system)).

**BASIC COMPILATION STEPS**

**There are different kinds of file :**

Compiling C programs requires you to work with four kinds of files:

1. Regular **source code** files. These files contain function definitions, and have names which end in ".c" by convention.
2. **Header** files. These files contain function declarations (also known as function prototypes) and various preprocessor statements (see below). They are used to allow source code files to access externally-defined functions. Header files end in ".h" by convention.
3. **Object** files. These files are produced as the output of the compiler. They consist of function definitions in binary form, but they are not executable by themselves. Object files end in ".o" by convention, although on some operating systems (*e.g.* Windows, MS-DOS), they often end in ".obj".
4. **Binary executables**. These are produced as the output of a program called a "linker". The linker links together a number of object files to produce a binary file which can be directly executed. Binary executables have no special suffix on Unix operating systems, although they generally end in ".exe" on Windows.

There are other kinds of files as well, notably libraries (".a" files) and shared libraries (".so" files), but you won't normally need to deal with them directly.

**The Preprocessor**

Before the C compiler starts compiling a source code file, the file is processed by a preprocessor. This is in reality a separate program (normally called "cpp", for "C preprocessor"), but it is invoked automatically by the compiler before compilation proper begins. What the preprocessor does is convert the source code file you write into another source code file (you can think of it as a "modified" or "expanded" source code file). That modified file may exist as a real file in the file system, or it may only be stored in memory for a short time before being sent to the compiler. Either way, you don't have to worry about it, but you do have to know what the preprocessor commands do.

Preprocessor commands start with the pound sign ("#"). There are several preprocessor commands; two of the most important are:

**#define**. This is mainly used to define constants. For instance,

#define BIGNUM 1000000

specifies that wherever the character string BIGNUM is found in the rest of the program, 1000000 should be substituted for it. For instance, the statement:

int a = BIGNUM;

becomes

int a = 1000000;

#define is used in this way so as to avoid having to explicitly write out some constant value in many different places in a source code file. This is important in case you need to change the constant value later on; it's much less bug-prone to change it once, in the #define, than to have to change it in multiple places scattered all over the code.

**#include**. This is used to access function definitions defined outside of a source code file. For instance:

#include <stdio.h>

causes the preprocessor to paste the contents of <stdio.h> into the source code file at the location of the #include statement before it gets compiled. #include is almost always used to include header files, which are files which mainly contain function declarations and #define statements. In this case, we use #include in order to be able to use functions such as printf and scanf, whose declarations are located in the file stdio.h. C compilers do not allow you to use a function unless it has previously been declared or defined in that file; #include statements are thus the way to re-use previously-written code in your C programs.

There are a number of other preprocessor commands as well, but we will deal with them as we need them.

**Making the object file: The Compiler**

After the C preprocessor has included all the header files and expanded out all the #define and #include statements (as well as any other preprocessor commands that may be in the original file), the compiler can compile the program. It does this by turning the C source code into an object code file, which is a file ending in ".o" which contains the binary version of the source code. Object code is not directly executable, though. In order to make an executable, you also have to add code for all of the library functions that were #included into the file (this is not the same as including the declarations, which is what #include does). This is the job of the linker (see the next section).

In general, the compiler is invoked as follows:

% gcc -c foo.c

where % is the unix prompt. This tells the compiler to run the preprocessor on the file foo.c and then compile it into the object code file foo.o. The -c option means to compile the source code file into an object file but not to invoke the linker. If your entire program is in one source code file, you can instead do this:

% gcc foo.c -o foo

This tells the compiler to run the preprocessor on foo.c, compile it and then link it to create an executable called foo. The -o option states that the next word on the line is the name of the binary executable file (program). If you don't specify the -o, i.e. if you just type gcc foo.c, the executable will be named a.out for silly historical reasons.

Note also that the name of the compiler we are using is gcc, which stands for "GNU C compiler" or "GNU compiler collection" depending on who you listen to. Other C compilers exist; many of them have the name cc, for "C compiler". On Linux systems cc is an alias for gcc.

**Putting it all together: The Linker**

The job of the linker is to link together a bunch of object files (.o files) into a binary executable. This includes both the object files that the compiler created from your source code files as well as object files that have been pre-compiled for you and collected into library files. These files have names which end in .a or .so, and you normally don't need to know about them, as the linker knows where most of them are located and will link them in automatically as needed.

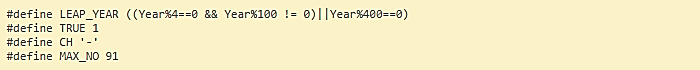
Like the preprocessor, the linker is a separate program called ld. Also like the preprocessor, the linker is invoked automatically for you when you use the compiler. The normal way of using the linker is as follows:

% gcc foo.o bar.o baz.o -o myprog

This line tells the compiler to link together three object files (foo.o, bar.o, and baz.o) into a binary executable file named myprog. Now you have a file called myprog that you can run and which will hopefully do something cool and/or useful.

This is all you need to know to begin compiling your own C programs.

**DATABASE ANALYZING, DESIGN AND IMPLEMENTATION**

* **#define Directive** is used to define a constant or creating a macro in C programming language. The first statement is defining a macro which accepts Year as an input parameter and returns TRUE/FALSE. If the year is a Leap Year, it returns TRUE otherwise FALSE.

To define preprocessor macros we can use #define. Its syntax is:

**#define identifier** replacement

When the preprocessor encounters this directive, it replaces any occurrence of identifier in the rest of the code by replacement. This replacement can be an expression, a statement, a block or simply anything. The preprocessor does not understand C++ proper, it simply replaces any occurrence of identifier by replacement.

|  |  |  |
| --- | --- | --- |
| 1 2 3 | #define TABLE\_SIZE 100  int table1[TABLE\_SIZE];  int table2[TABLE\_SIZE]; |  |

After the preprocessor has replaced TABLE\_SIZE, the code becomes equivalent to:

|  |  |  |
| --- | --- | --- |
| 1 2 | int table1[100];  int table2[100]; |  |

#define can work also with parameters to define function macros:

|  |  |
| --- | --- |
|  | #define getmax(a,b) a>b?a:b |

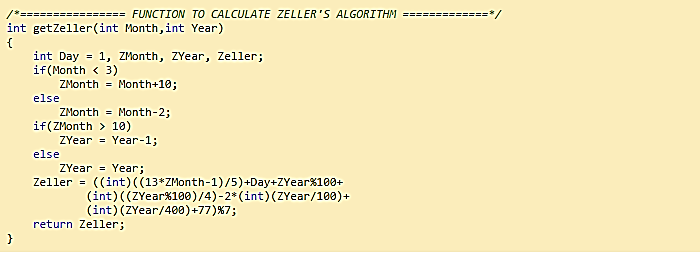
Defined macros are not affected by block structure. A macro lasts until it is undefined with the #undef preprocessor directive:

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 | #define TABLE\_SIZE 100  int table1[TABLE\_SIZE];  #undef TABLE\_SIZE  #define TABLE\_SIZE 200  int table2[TABLE\_SIZE]; |  |

This would generate the same code as:

|  |  |
| --- | --- |
| 1 2 | int table1[100];  int table2[200]; |

* **Zeller's Algorithm** can be used to determine the day of the week for any date in the past, present or future for any dates between 1582 and 4902. We're using this function to get the weekday of the 1st day of given month.



In **Zeller’s algorithm**, the months are numbered from 1 for March to 12 for February. The year is assumed to begin in March; this means, for example, that January 1995 is to be treated as month 11 of 1994. The formula for the Gregorian calendar is

{\displaystyle w=\left(d+\left\lfloor {\frac {13(m+1)}{5}}\right\rfloor +y+\left\lfloor {\frac {y}{4}}\right\rfloor +\left\lfloor {\frac {c}{4}}\right\rfloor -2c\right){\bmod {7}},}

Where

* *Y* is the year minus 1 for January or February, and the year for any other month
* *y* is the last 2 digits of *Y*
* *c* is the first 2 digits of *Y*
* *d* is the day of the month (1 to 31)
* *m* is the shifted month (March=1, February=12)
* *w* is the day of week (1=Sunday,0=Saturday)

The only difference is one between Zeller’s algorithm (*Z*) and the Gaussian algorithm (*G*), that is *Z* − *G* = 1 = Sunday.

{\displaystyle (d+\lfloor (m+1)2.6\rfloor +y+\lfloor y/4\rfloor +\lfloor c/4\rfloor -2c){\bmod {7}}-(d+\lfloor 2.6m-0.2\rfloor +y+\lfloor y/4\rfloor +\lfloor c/4\rfloor -2c){\bmod {7}}}

{\displaystyle =(\lfloor (m+2+1)2.6-(2.6m-0.2)\rfloor ){\bmod {7}}}{\displaystyle =(\lfloor 2.6m+7.8-2.6m+0.2\rfloor ){\bmod {7}}}

{\displaystyle =8{\bmod {7}}=1}

So, we can get the values of months from those for the Gaussian algorithm by adding one:

Months m

January 1

February 4

March 3

April 6

May 1

June 4

July 6

August 2

September 5

October 0

November 3

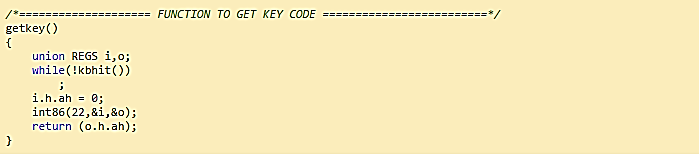
December 5

* **khbit()**

kbhit function is used to determine if a key has been pressed or not. To use kbhit function in your program you should include the header file "conio.h". If a key has been pressed then it returns a non-zero value otherwise returns zero.

Declaration: int kbhit();

As long as in the above program user doesn't presses a key kbhit() return zero and (!0) i.e. 1 the condition in while loop is true and "You haven't pressed a key." will be printed again and again. As a key is pressed from the keyboard the condition in while loop become false as now kbhit() will return a non-zero value and ( !(non-zero) = 0), so the control will come out of the while loop.

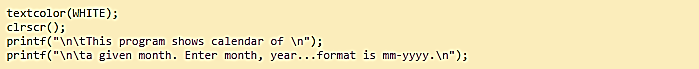


* **textcolor() function** is used to change the color of drawing text in c programs.

Where color is an integer variable. For example, 0 means BLACK color, 1 means BLUE, 2 means GREEN and soon. You can also use write appropriate color instead of integer. For example, you can write textcolor(YELLOW); to change text color to YELLOW. But use colors in capital letters only.

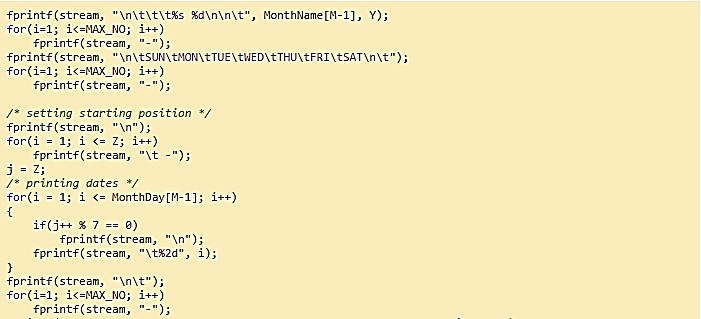
Declaration: void textcolor(int color);

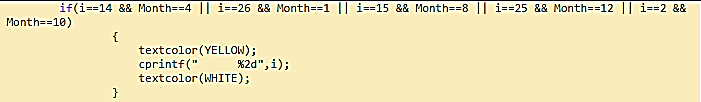
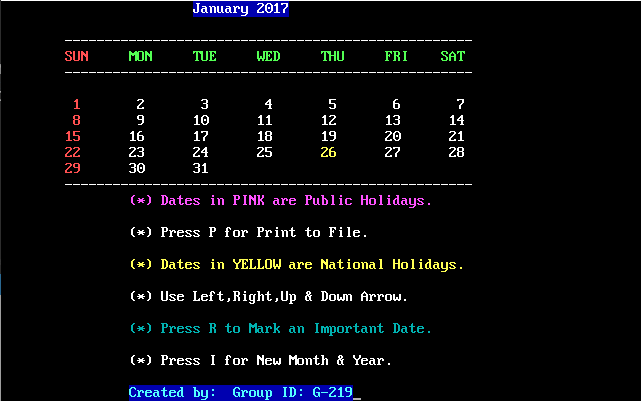
* **clrscr()** function is used to clear screen.



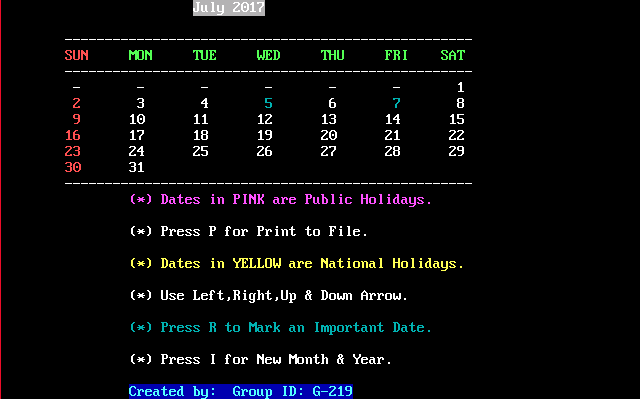
* **textbackground()** - This function is used to change current background colour in text mode.
* **fflush(stdin);** - This function is called to clear the input buffer.
* **printchar()**- It accepts a character and prints the same 51 times on a single line. This function will be used to do some formatting the output.
* **getch();** - this function is used to get a character/key hit from the console input i.e. keyboard. Program waits until the user hits any key on the keyboard.
* **exit(1);** - will exit from the program.
* Once everything is written to the file, **fclose()** function is called to close the file properly.
* Adjusting February in case of Leap Year, here we are calculating the days in February month. If it's a leap year, consider 29 days else 28 days.



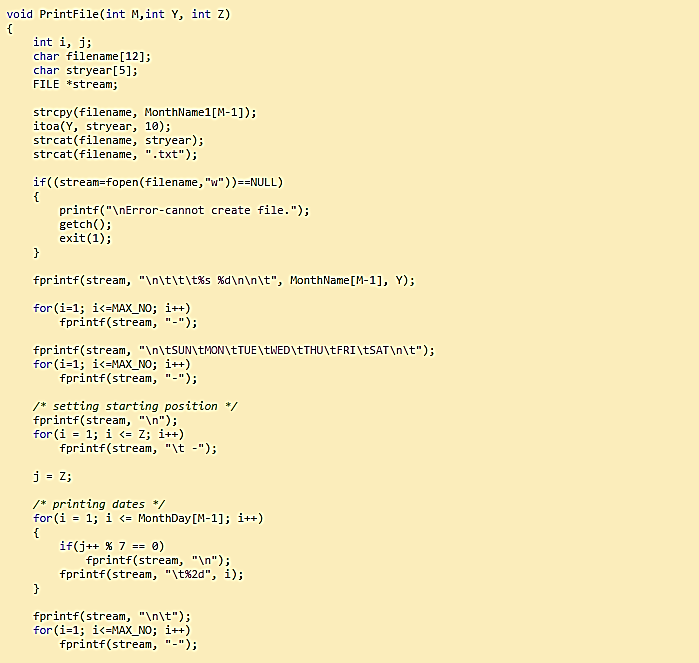
* The lines below are used to print the dates of the month with proper formatting.
* **National Holidays:** Added National Holidays, ‘i’ represents the date here whereas Month is already specified. Specified ‘yellow’ colour has been allotted to them.  
  + 26th January : Republic Day
  + 14th April : Ambedkar Jayanthi
  + 15th August : Independence Day
  + 2nd October : Gandhi Jayanthi
  + 25th December : Christmas .



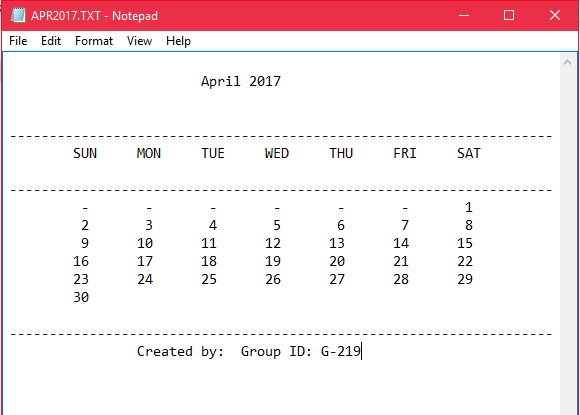
* **Public Holidays**: Similarly, we have added some public holidays. And those holidays are visible in marked with ‘pink’ colour. As in image below, 13th March is considered as HOLI(A public holiday in some parts of India).
* **Marking Important Date:** Also, there is an option to mark an important date. By pressing R key, you’ll go to the screen where you can enter the date, month & year to be marked as important date. The colour of the important dates have been kept as different so that they are easily visible in the calendar. Marked 5th and 7th of July as Important.



* **PrintFile();** - function will be used to print the output in a text file and save in the disk.  
  + **strcpy(des,src);** - this function is used to copy src string to the des string. Here month name will be copied to filename variable. The month name will be used to create the text file name.
  + **itoa();** - is used to convert an integer to a string. We're converting given year to string so that it can be concatenated with the month name and form the output text file name.
  + **strcat(str1, str2);** - this function is used to concatenate str2 with str1. Initially we copied the month name in filename variable. Now we concatenate year with the month name to make the filename as follows: "MONTHYEAR.txt".

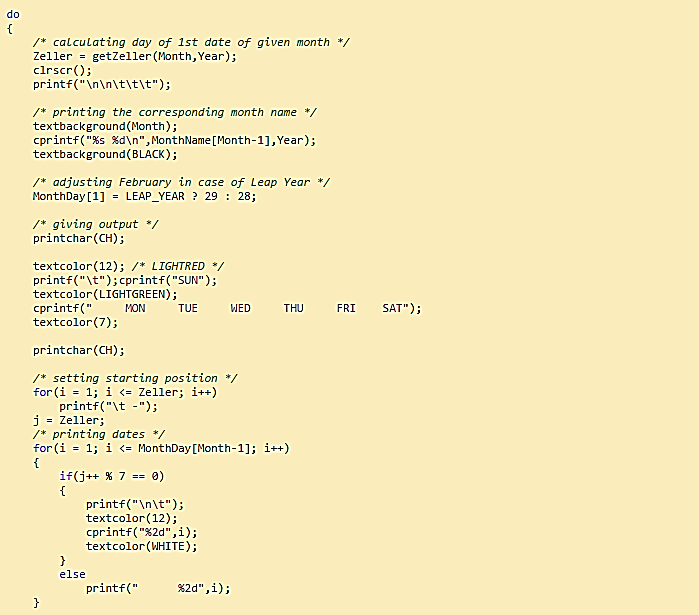


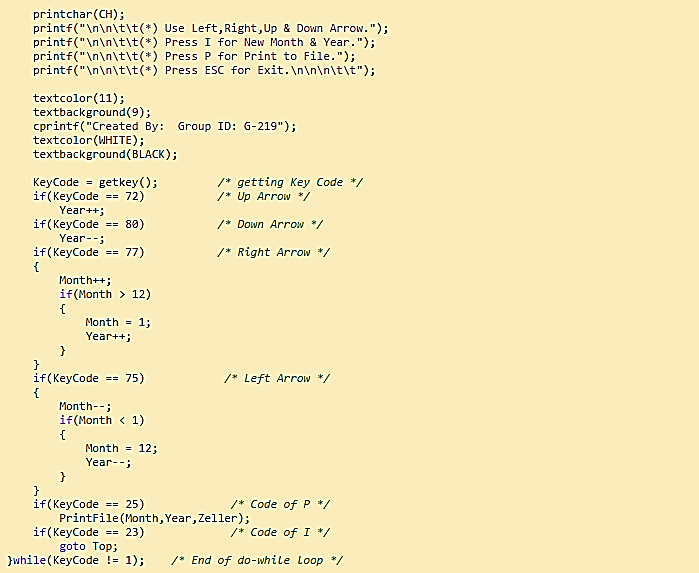
* Pressing P key will print the calendar to a text file where the program is saved. Output is as follows:

****

**As we're done with all the required functions. Let's complete the logic in main method.**

Nothing special in the main part that hasn’t been discussed. Used only to give output and to check the key press inputs.

****

****

**LIMITATIONS**

* Dates Marked as Important can’t be unmarked. (Increase in size every time the date is marked important)
* There is no link to the current date or time in any timezone in the world.
* There is no alarm type notification for the date marked important. It is just a visual interface.
* No link with database to record the events.

**FUTURE SCOPE**

* Accounts can be implemented for each person to store its events and add a note.  
  That can be accessed only by the other users having password with a 2 step verification.
* Merging/Unmerging Important dates on one or more parameters can be done also, transferring them to new date can be added.
* We can attract more users to use this application by providing better interface and advanced functionalities.
* Moreover, this application can be used to fetch events from not only Google calendar but also from other available calendars as well.
* Even if we make it to link with the current date and time (GMT), using device’s location services, then we can add an alarm or a type of notification on users device to check what important has to be done on that day.

**REFRENCES**

* **WEBSITES**
* <https://en.wikipedia.org/wiki/Determination_of_the_day_of_the_week>
* <http://www.cprogramming.com/fod/kbhit.html>
* <http://www.irietools.com/iriepascal/progref298.html>
* <https://en.wikipedia.org/wiki/Zeller%27s_congruence>
* <http://stackoverflow.com/questions/725098/leap-year-calculation>
* <https://www.programiz.com/c-programming/c-goto-statement>
* <https://msdn.microsoft.com/en-us/library/teas0593.aspx>
* **BOOKS**
* Introduction to C Programming by Reema Thareja
* C Programming Book by Sumita Arora