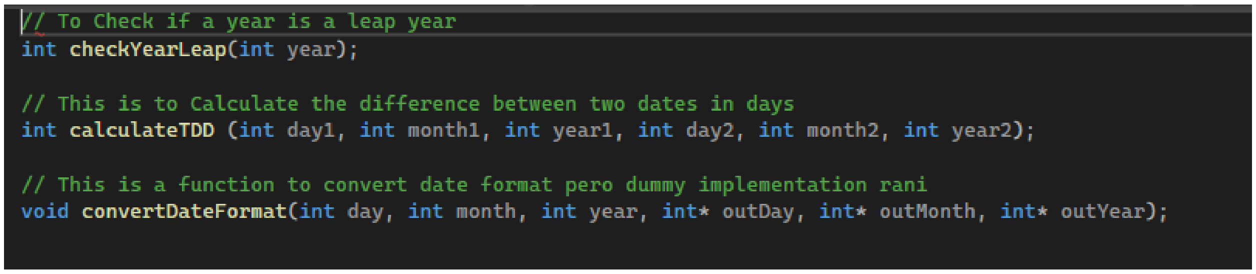
Hello! Idk how to start documentation, so I will start it like this.

Here is my header file, I named it abad_datetime.h as sir instructed



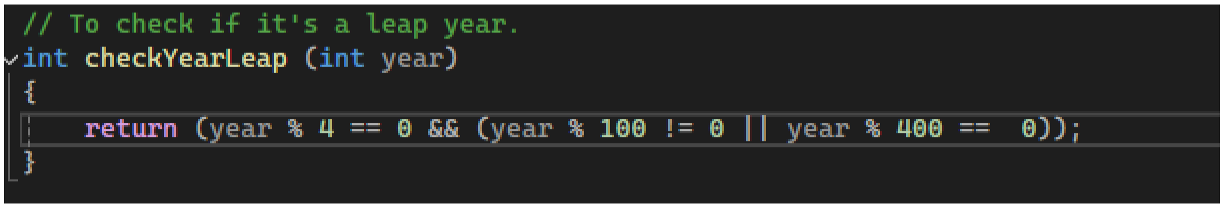
```c
// To Check if a year is a leap year
int checkYearLeap(int year);

// This is to Calculate the difference between two dates in days
int calculateTDD (int day1, int month1, int year1, int day2, int month2, int year2);

// This is a function to convert date format pero dummy implementation rani
void convertDateFormat(int day, int month, int year, int* outDay, int* outMonth, int* outYear);
```

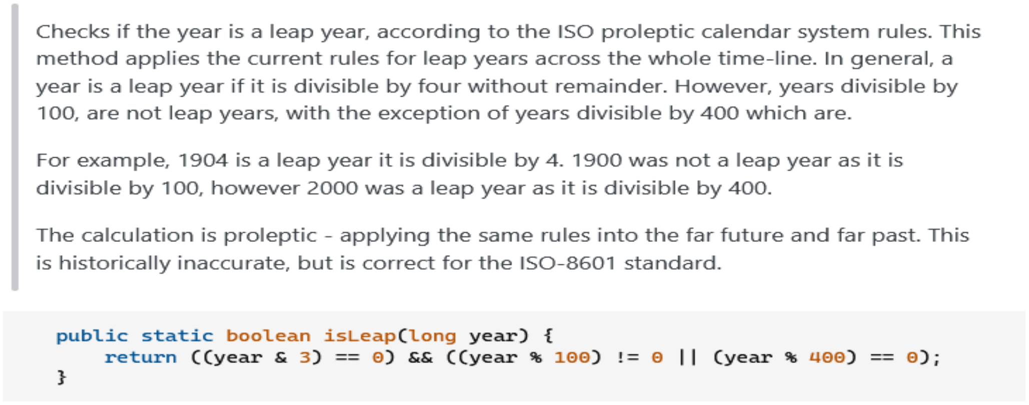
So in my header file there are three declarations checkYearLeap which is to check if a year is a leap year, calculateTDD which is a shortcut name of Calculate Two difference Dates, then concertDateFormat

So in the implementation sulod sa header file, checking the year leap is okay, welp with the power of the internet you can always find solution on something you dont know then I found out the formula to calculate the leap year which is you jus have to divide it by 4 here is my leap year



```c
// To check if it's a leap year.
int checkYearLeap (int year)
{
    return (year % 4 == 0 && (year % 100 != 0 || year % 400 ==  0));
}
```

If you ask, why is there a 100 or 400?
So if a year is divisible by 100 it would normally be a not leap year
Else
If a year is divisible by 400 then it is a leap year. How did I know that? Welp:
https://stackoverflow.com/questions/73009576/for-a-leap-year-check-why-check-if-the-year-is-not-divisible-by-100 if you don't have internet here's the screenshot

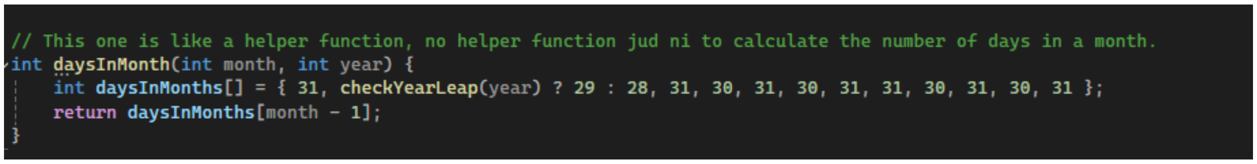


> Checks if the year is a leap year, according to the ISO proleptic calendar system rules. This method applies the current rules for leap years across the whole time-line. In general, a year is a leap year if it is divisible by four without remainder. However, years divisible by 100, are not leap years, with the exception of years divisible by 400 which are.
>
> For example, 1904 is a leap year it is divisible by 4. 1900 was not a leap year as it is divisible by 100, however 2000 was a leap year as it is divisible by 400.
>
> The calculation is proleptic - applying the same rules into the far future and far past. This is historically inaccurate, but is correct for the ISO-8601 standard.

```java
public static boolean isLeap(long year) {
    return ((year & 3) == 0) && ((year % 100) != 0 || (year % 400) == 0);
}
```

Yes this is java but what i only followed is the return, not the public static chuchu idk what that is but I think mura rag int *labada ulo aning java uy*

So next! To calculate the difference I need a helper function, diri jud ko ga lisod because I got stuck at leap year chuchu so here’s my helper function



```c
// This one is like a helper function, no helper function jud ni to calculate the number of days in a month.
int daysInMonth(int month, int year) {
    int daysInMonths[] = { 31, checkYearLeap(year) ? 29 : 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
    return daysInMonths[month - 1];
}
```

So while making this helper function, while searching for ways I found out there’s an easy way to write if else just by using “Ternary Conditional Operation” which is kanang ‘?’ and ‘:’ mura ra siyag shortcut

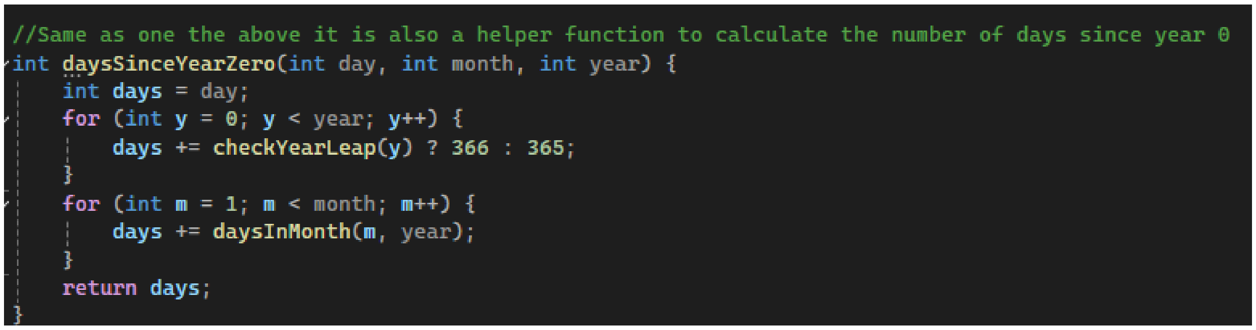
? = if
: = else
but yeah for more information just read this:
https://www.naukri.com/code360/library/conditional-operator-in-c

So back to my helper function on days in months, so this helper function it is just to calculate the number of days in a months.

So for february *which is i got stuck here for many days* is it uses the ternary operator checkYearLeap(year) ? 29 : 28 to set the number of days based on whether the year is a leap year or not.

If checkYearLeap(year) returns true (non-zero), February has 29 days.
Else, February has 28 days.
Then ang return statement niya kay daysInMonths[month - 1] this accesses the array using month - 1 because array in C are 0-based man, while month calendar numbers are 1-based. Cause if wala ni siya ig ka count niyas mga numbers mag start 0, but wa may start 0 sa calendar.

Another helper function



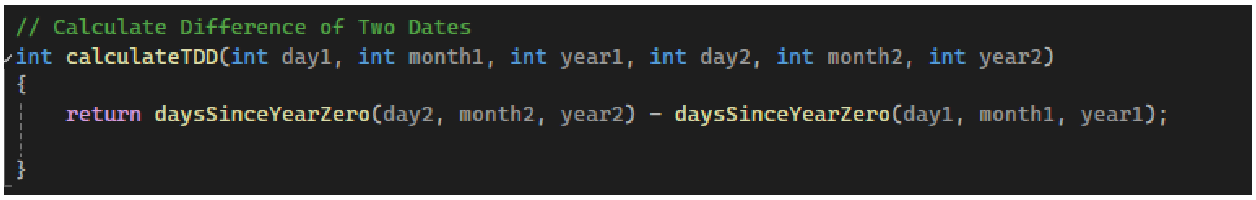
```c
//Same as one the above it is also a helper function to calculate the number of days since year 0
int daysSinceYearZero(int day, int month, int year) {
    int days = day;
    for (int y = 0; y < year; y++) {
        days += checkYearLeap(y) ? 366 : 365;
    }
    for (int m = 1; m < month; m++) {
        days += daysInMonth(m, year);
    }
    return days;
}
```

So kani siya mo calculate the number of days from year 0
Mag initialize siya kay start siya sa day then after that it goes through the for year loop
for (int y = 0; y < year; y++) iterates through each year from 0 up to the given year.
Days += checkYearLeap(y) ? 366 : 365; adds the number of days in each year. 366 is a leap year and 355 is not.

So the first loop, which is year, add days for complete years while ang month adds days for complete months in a given year.

Sa month same ra sila sa pag iterate sa for loop year pero it adds number of days in total
Days + daysinMonth = chuchu

So! Nahumana man ang helper functions, these helper functions supports calculate the difference of two dates this is the calculate the difference of two dates.



```
// Calculate Difference of Two Dates
int calculateTDD(int day1, int month1, int year1, int day2, int month2, int year2)
{
    return daysSinceYearZero(day2, month2, year2) - daysSinceYearZero(day1, month1, year1);

}
```

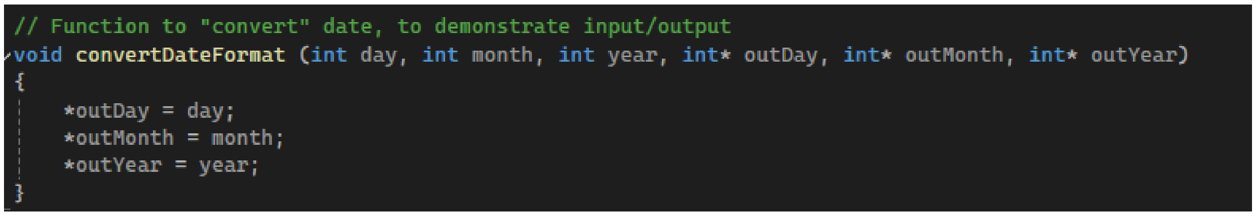
As you can see gi tawag ang daysSinceYearZero

The daysSinceYearZero(day1, month1, year1) This function calculates the total number of days from year 0 up to the first date (day1, month1, year1).
The same goes for day2, month2, year2 calculate total number of days from year 0 but it's up to the second date

If you ask why gi una ang second date? By subtracting the first date's total days from the second date's total days, the function accurately calculates the difference in days.

So if ako e una ang first date mo negative ang gawas plus ma minusan og 1 example 12/15/2005 then 12/25/2005 ang output kay -9 even tho the output should be 10 das why gi una ang second date to subtract the first date aron murag mas accurate siya.

So naa natas convert function date actually mao ning pina ka sayun kay lahus rajud ni printf pero naa lay churva tungod sa headerfile



```
// Function to "convert" date, to demonstrate input/output
void convertDateFormat (int day, int month, int year, int* outDay, int* outMonth, int* outYear)
{
    *outDay = day;
    *outMonth = month;
    *outYear = year;
}
```

Purpose? Demonstrates how to “convert” or output a date in a different format.
So it takes input day, month, year and assigns them to the pointers outDay, outMonth, outYear.
Essentially copies the input values to the specified output pointers, which can be used to retrieve or display the date in another part of the program.

SUMMARY!!!
So the purpose of this header file is the usage of

checkYearLeap: Determines leap years.
calculateTDD: Computes the difference in days between two dates. With the help of helper function name **daysInMonth** which retrieves days in a specific month. And **daysSinceYearZero** that calculates total days from year 0 to a specified date.
convertDateFormat: Illustrates date output handling.