How do I calculate someone's age in C#?

Ask Question

1743

Given a `DateTime` representing a person's birthday, how do I calculate their age in years?

c# .net datetime

421

edited Apr 21 '18 at 17:48

community wiki
29 revs, 25 users 37%
Shaik Raffi

locked by Yvette Colomb ♦ Apr 21 '18 at 17:49

This post has been locked while disputes about its content are being resolved. For more info visit meta.

Read more about locked posts here.

127 what all of the answers so far have missed is that it depends where the person was born and where they are right now. – Yaur May 21 '11 at 7:34

37 @Yaur: Just convert the time of now + birth into GMT/UTC, age is only a relative value, hence timezones are irrelevant. For determining the user's current timezone, you can use GeoLocating. – Stefan Steiger Oct 3 '11 at 10:20

4 If we're taking into consideration @Yaur 's suggestion of cross-timezone calculations, should Day Light Saving Time affect the calculation in any manner? – DDM Jul 11 '15 at 3:42

|

63 Answers

1 2 3 next

1879

✓

An easy to understand and simple solution.

```
// Save today's date.
var today = DateTime.Today;
// Calculate the age.
var age = today.Year - birthdate.Year;
// Go back to the year the person was born in case of a leap year
if (birthdate > today.AddYears(-age)) age--;
```

However, this assumes you are looking for the *western* idea of age and not using *East Asian reckoning*.

edited Jul 5 '18 at 3:51

community wiki
24 revs, 14 users 20%
Mike Polen

223 Just wanted to comment on DateTime.Now performance. If you don't need an accurate time zone value, use DateTime.UtcNow it's much faster. – JAG Jan 22 '09 at 10:29

92 Given we're talking birthdays you can just use DateTime.Today given the time part has no relevance. – Tristan Warner-Smith Jul 24 '09 at 18:04

73 This answer does not work with all locales and all ages. Several countries have skipped dates after the birth of current living people, including Russia (1918), Greece (1924) and Turkey (1926). – Lars D Nov 9 '09 at 22:09

27 Actually, it's still not entirely correct. This code presumes that 'bday' is the date-portion of a DateTime. It's an edge-case (I guess most people will just be passing dates and not date-times), but if you pass in a birthday as a date-and-time where the time is greater than 00:00:00 then you'll run into the bug Danvil pointed out. Setting bday = bday.Date fixes this. – Øyvind Nov 16 '10 at 15:37

100 The last line made me think too much. Instead how about: if (bday.AddYears(age) > now) age--; This seems to be a more intuitive expression. – cdiggins Jul 16 '11 at 17:53

946

This is a strange way to do it, but if you format the date to `yyyymmdd` and subtract the date of birth from the current date then drop the last 4 digits you've got the age :)

I don't know C#, but I believe this will work in any language.

```
20080814 - 19800703 = 280111
```

Drop the last 4 digits = `28` .

C# Code:

```
int now = int.Parse(DateTime.Now.ToString("yyyyMMdd"));
int dob = int.Parse(dateOfBirth.ToString("yyyyMMdd"));
int age = (now - dob) / 10000;
```

Or alternatively without all the type conversion in the form of an extension method. Error checking omitted:

```
public static Int32 GetAge(this DateTime dateOfBirth)
{
    var today = DateTime.Today;

    var a = (today.Year * 100 + today.Month) * 100 + today.Day;
    var b = (dateOfBirth.Year * 100 + dateOfBirth.Month) * 100 + dat

    return (a - b) / 10000;
}
```

edited Jan 1 '17 at 1:58

community wiki
12 revs, 8 users 38%
ScArcher2

50 Horray for `yyyymmdd` . The 1st database product I used on a **P**ersonal **C**omputer (circa 1981) stored dates in this format. Date manipulation of any kind was so much easier. – radarbob Nov 19 '14 at 19:43

2 Actually this is great for usage on MS-SQL with datetime-fields (total days since 01-011900) – Patrik Eckebrecht Jul 3 '15 at 12:01

2 @numerek Please post your suggested modifications as their own answer. For what it's worth, the current year times 10000 is nowhere near an integer overflow, by two orders of magnitude. 20,150,000 vs 2,147,483,648 – GalacticCowboy Sep 3 '15 at 20:23

2 @LongChalk `20180101 - 20171231 = 8870` . Drop the last 4 digits and you have (an implied) `0` for the age. How did you get `1` ? – Rufus L Jun 14 '18 at 20:36

1 @flindeberg Yeah, that's what I was saying to LongChalk... – Rufus L Jul 3 '18 at 19:32

361

I don't know how the wrong solution can be accepted. The correct C# snippet was written by Michael Stum

Here is a test snippet:

```
DateTime bDay = new DateTime(2000, 2, 29);
DateTime now = new DateTime(2009, 2, 28);
MessageBox.Show(string.Format("Test {0} {1} {2}",
                CalculateAgeWrong1(bDay, now),     // outputs 9
                CalculateAgeWrong2(bDay, now),     // outputs 9
                CalculateAgeCorrect(bDay, now)));  // outputs 8
```

Here you have the methods:

```
public int CalculateAgeWrong1(DateTime birthDate, DateTime now)
{
    return new DateTime(now.Subtract(birthDate).Ticks).Year - 1;
}

public int CalculateAgeWrong2(DateTime birthDate, DateTime now)
{
    int age = now.Year - birthDate.Year;

    if (now < birthDate.AddYears(age))
        age--;

    return age;
}

public int CalculateAgeCorrect(DateTime birthDate, DateTime now)
{
    int age = now.Year - birthDate.Year;

    if (now.Month < birthDate.Month || (now.Month == birthDate.Month
birthDate.Day))
        age--;
    return age;
}
```

edited Feb 7 '16 at 0:04

community wiki
4 revs, 4 users 78%
RMA

8 Output was -Test 9 9 8 – MoXplod Nov 29 '11 at 20:52

28 While this code works, it asserts that a person born on a leap day attains the next year of age on March 1st on non-leap years, rather than on February 28th. In reality, *either option may be correct*. Wikipedia has something to say about this. So while your code is not "wrong", neither is the accepted solution. – Matt Johnson Aug 17 '14 at 5:44

13 @MattJohnson I think that's actually correct. If my bday was Feb 29, then Feb 28 my bday hasn't passed, and I should still be the same age as on Feb 27. On March 1, however, we have passed my bday and I should be the next age. In the US, a business that sells alcohol will have a sign that says something like "If you were born after this day in YYYY, you can't purchase alcohol" (where YYYY changes every year). That means that someone born on Feb 29 cannot buy alcohol on Feb 28 in the year they turn 21 (most places), and lends support to the idea that they are not a year older until March 1. – jfren484 Jul 12 '16 at 17:18

2 @jfren484 - read the Wikipedia article. It varies considerably across jurisdictions. – Matt Johnson Jul 12 '16 at 19:26

5 @jfren484 Your claim has absolutely nothing to do with philosophy; but everything to do with ***your own personal feeling***. When a person born on 29 Feb "ages" is largely unimportant unless the age forms a 'legal age boundary' (e.g. Can buy alcohol, vote, get pension, join army, get driving license). Consider US drinking age (21 years): For most people that's 7670 days. It's 7671 days if born before 29 Feb in leap year or from 1 Mar before leap year. If born on 29 Feb: 28 Feb is 7670 days and 1 Mar is 7671 days. ***The choice is arbitrary*** it can go either way. – Disillusioned Mar 4 '17 at 10:06

125

I don't think any of the answers so far provide for cultures that calculate age differently. See, for example, East Asian Age Reckoning versus that in the West.

Any *real* answer has to include localization. The Strategy Pattern would probably be in order in this example.

edited Jul 5 '18 at 3:52

community wiki
2 revs, 2 users 86%
James A. Rosen

24 From the wikipedia article that you provided: "In China and Japan it is used for traditional fortune-telling or religion, and it is disappearing in daily life between peoples in the city." – some Dec 28 '08 at 9:15

7 @some -- Koreans still use this system primarily. – Justin L. Jun 25 '10 at 9:15

21 Actually this concept can be pretty important - people don't like being told their personal information incorrectly. As an example, half of my family lives in Malaysia and half in the UK. Right now my age is considered two years higher when I'm with one side of my family than with the other. – Phil Gan Aug 5 '10 at 8:16

6 Not only us this system used primarily in Korea, but as a tourist discussing ages with locals, locals will politely refer to yourself an each other by their birth year. I'm not 25, I'm 87. I like this approach better. more of an 'international birthdatetime format' – Dean Rather Nov 12 '12 at 5:56

2 In Vietnam, in daily life even when Western system is used, age is calculated merely by subtracting the years. Date is excluded – phuclv Jul 11 '15 at 0:56

103

The simple answer to this is to apply `AddYears` as shown below because this is the only native method to add years to the 29th of Feb. of leap years and obtain the correct result of the 28th of Feb. for common years.

Some feel that 1th of Mar. is the birthday of leaplings but neither .Net nor any official rule supports this, nor does common logic explain why some born in February should have 75% of their birthdays in another month.

Further, an Age method lends itself to be added as an extension to `DateTime`. By this you can obtain the age in the simplest possible way:

1. List item

int age = birthDate.Age();

```
public static class DateTimeExtensions
{
    /// <summary>
    /// Calculates the age in years of the current System.DateTime ol
    /// </summary>
    /// <param name="birthDate">The date of birth</param>
    /// <returns>Age in years today. 0 is returned for a future date
    public static int Age(this DateTime birthDate)
    {
        return Age(birthDate, DateTime.Today);
    }

    /// <summary>
    /// Calculates the age in years of the current System.DateTime ol
date.
    /// </summary>
    /// <param name="birthDate">The date of birth</param>
    /// <param name="laterDate">The date on which to calculate the a
    /// <returns>Age in years on a later day. 0 is returned as minim
    public static int Age(this DateTime birthDate, DateTime laterDat
    {
        int age;
        age = laterDate.Year - birthDate.Year;
```

```
            if (age > 0)
            {
                age -= Convert.ToInt32(laterDate.Date < birthDate.Date.A
            }
            else
            {
                age = 0;
            }

            return age;
        }
    }
```

Now, run this test:

```
    class Program
    {
        static void Main(string[] args)
        {
            RunTest();
        }

        private static void RunTest()
        {
            DateTime birthDate = new DateTime(2000, 2, 28);
            DateTime laterDate = new DateTime(2011, 2, 27);
            string iso = "yyyy-MM-dd";

            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    Console.WriteLine("Birth date: " + birthDate.AddDays
    "  Later date: " + laterDate.AddDays(j).ToString(iso) + "  Age: " +
    birthDate.AddDays(i).Age(laterDate.AddDays(j)).ToString());
                }
            }

            Console.ReadKey();
        }
    }
```

The critical date example is this:

Birth date: 2000-02-29 Later date: 2011-02-28 Age: 11

Output:

```
{
    Birth date: 2000-02-28  Later date: 2011-02-27  Age: 10
    Birth date: 2000-02-28  Later date: 2011-02-28  Age: 11
    Birth date: 2000-02-28  Later date: 2011-03-01  Age: 11
    Birth date: 2000-02-29  Later date: 2011-02-27  Age: 10
    Birth date: 2000-02-29  Later date: 2011-02-28  Age: 11
    Birth date: 2000-02-29  Later date: 2011-03-01  Age: 11
    Birth date: 2000-03-01  Later date: 2011-02-27  Age: 10
    Birth date: 2000-03-01  Later date: 2011-02-28  Age: 10
    Birth date: 2000-03-01  Later date: 2011-03-01  Age: 11
}
```

And for the later date 2012-02-28:

```
{
    Birth date: 2000-02-28  Later date: 2012-02-28  Age: 12
    Birth date: 2000-02-28  Later date: 2012-02-29  Age: 12
    Birth date: 2000-02-28  Later date: 2012-03-01  Age: 12
    Birth date: 2000-02-29  Later date: 2012-02-28  Age: 11
    Birth date: 2000-02-29  Later date: 2012-02-29  Age: 12
    Birth date: 2000-02-29  Later date: 2012-03-01  Age: 12
    Birth date: 2000-03-01  Later date: 2012-02-28  Age: 11
    Birth date: 2000-03-01  Later date: 2012-02-29  Age: 11
    Birth date: 2000-03-01  Later date: 2012-03-01  Age: 12
}
```

edited Feb 7 '16 at 0:05

community wiki
6 revs, 6 users 86%
camelCasus

3 A comment regarding having the 29th Feb birthday on 1st March, technically, having it on the 28th is too early (1 day early in fact). On the 1st is one day too late. But since the birthday is between, using the 1st to

calculate the age in non-leap years makes more sense to me, since that person is indeed that old on March 1st (and 2nd and 3rd) every year, but not on Feb 28th. – CyberClaw Jul 24 '18 at 16:31

79

My suggestion

```
int age = (int) ((DateTime.Now - bday).TotalDays/365.242199);
```

That seems to have the year changing on the right date. (I spot tested up to age 107)

edited Aug 1 '13 at 13:20

community wiki
2 revs, 2 users 89%
James Curran

4 Where does 365.255 come from? I don't think this will work in general. – dreeves Jan 18 '09 at 3:15

13 365 for the days in a year. +0.25 for leap years. +0.005 for other corrections – James Curran Jan 20 '09 at 16:04

22 I don't think Harry Patch would have appreciated your spot-testing methodology: latimes.com/news/obituaries/... – MusiGenesis Aug 1 '09 at 16:03

10 The average length of a year in the Gregorian Calendar is 365.2425 days. – dan04 Oct 6 '10 at 2:01

11 ^^ Because sometimes it's important. In my testing this fails on the persons birthday, it reports them younger than they are. – ChadT Mar 25 '11 at 5:27

69

Another function, not by me but found on the web and refined it a bit:

```
public static int GetAge(DateTime birthDate)
{
    DateTime n = DateTime.Now; // To avoid a race condition around m
    int age = n.Year - birthDate.Year;

    if (n.Month < birthDate.Month || (n.Month == birthDate.Month && 
birthDate.Day))
        age--;

    return age;
}
```

Just two things that come into my mind: What about people from countries that do not use the gregorian calendar? DateTime.Now is in the server-specific culture i think. I have absolutely 0 knowledge about actually working with Asian calendars and I do not know if there is an easy way to convert dates between calendars, but just in case you're wondering about those chinese guys from the year 4660 :-)

edited Jul 25 '17 at 17:11

community wiki
5 revs, 3 users 93%
Michael Stum

2 Main problems to solve are:

1. Calculate Exact age - in years, months, days, etc.

47

2. Calculate Generally perceived age - people usually do not care how old they exactly are, they just care when their birthday in the current year is.

Solution for **1** is obvious:

```
DateTime birth = DateTime.Parse("1.1.2000");
DateTime today = DateTime.Today;     //we usually don't care about b
TimeSpan age = today - birth;        //.NET FCL should guarantee thi
double ageInDays = age.TotalDays;    //total number of days ... also
double daysInYear = 365.2425;        //statistical value for 400 yea
double ageInYears = ageInDays / daysInYear;  //can be shifted ... no
```

Solution for **2** is the one which is not so precise in determing total age, but is perceived as precise by people. People also usually use it, when they calculate their age "manually":

```
DateTime birth = DateTime.Parse("1.1.2000");
DateTime today = DateTime.Today;
int age = today.Year - birth.Year;    //people perceive their age in

if (today.Month < birth.Month ||
   ((today.Month == birth.Month) && (today.Day < birth.Day)))
{
  age--;  //birthday in current year not yet reached, we are 1 year
          //+ no birthday for 29.2. guys ... sorry, just wrong date
}
```

Notes to 2.:

- This is my preferred solution
- We cannot use DateTime.DayOfYear or TimeSpans, as they shift number of days in leap years
- I have put there little more lines for readability

Just one more note ... I would create 2 static overloaded methods for it, one for universal usage, second for usage-friendliness:

```
public static int GetAge(DateTime bithDay, DateTime today)
{
  //chosen solution method body
}

public static int GetAge(DateTime birthDay)
{
  return GetAge(birthDay, DateTime.Now);
}
```

edited Feb 7 '16 at 0:06

community wiki
5 revs, 2 users 96%
Thetam

45

I am late to the party, but here's a one-liner:

```
int age = new DateTime(DateTime.Now.Subtract(birthday).Ticks).Year-1
```

edited Dec 12 '09 at 13:04

community wiki
2 revs, 2 users 80%
SillyMonkey

22 This is broken. Made testable: public static int CalculateAge(DateTime dateOfBirth, DateTime dateToCalculateAge) { return new DateTime(dateToCalculateAge.Subtract(dateOfBirth).Ticks).Year - 1; } ...Gives age 14 when I input 1990-06-01 and calculate the age on the day BEFORE his 14th birthday (1990-05-31). – Kjensen Feb 5 '11 at 21:42

36

This is the version we use here. It works, and it's fairly simple. It's the same idea as Jeff's but I think it's a little clearer because it separates out the logic for subtracting one, so it's a little easier to understand.

```
public static int GetAge(this DateTime dateOfBirth, DateTime dateAsA
{
    return dateAsAt.Year - dateOfBirth.Year - (dateOfBirth.DayOfYear
dateAsAt.DayOfYear ? 0 : 1);
}
```

You could expand the ternary operator to make it even clearer, if you think that sort of thing is unclear.

Obviously this is done as an extension method on `DateTime` , but clearly you can grab that one line of code that does the work and put it anywhere. Here we have another overload of the Extension method that passes in `DateTime.Now` , just for completeness.

edited May 23 '10 at 10:19

community wiki
2 revs, 2 users 86%
Ch00k

5 I think this can be off by one day when exactly one of dateOfBirth or dateAsAt falls in a leap year. Consider the age of a person born on March 1, 2003 on February 29, 2004. To rectify this, you need to do a lexicographic comparison of (Month, DayOfMonth) pairs and use that for the conditional. – Doug McClean Dec 23 '08 at 15:36

32

The best way that I know of because of leap years and everything is:

```
DateTime birthDate = new DateTime(2000,3,1);
int age = (int)Math.Floor((DateTime.Now - birthDate).TotalDays / 365
```

Hope this helps.

edited Aug 1 '08 at 15:26

community wiki
3 revs
Nick Berardi

30

I use this:

```
public static class DateTimeExtensions
{
    public static int Age(this DateTime birthDate)
    {
        return Age(birthDate, DateTime.Now);
    }

    public static int Age(this DateTime birthDate, DateTime offsetDa
    {
        int result=0;
        result = offsetDate.Year - birthDate.Year;

        if (offsetDate.DayOfYear < birthDate.DayOfYear)
        {
              result--;
        }

        return result;
    }
}
```

edited Jun 3 '15 at 11:10

community wiki
4 revs, 4 users 78%
Elmer

29

+50

This gives "more detail" to this question. Maybe this is what you're looking for

```
DateTime birth = new DateTime(1974, 8, 29);
DateTime today = DateTime.Now;
TimeSpan span = today - birth;
DateTime age = DateTime.MinValue + span;

// Make adjustment due to MinValue equalling 1/1/1
int years = age.Year - 1;
int months = age.Month - 1;
int days = age.Day - 1;

// Print out not only how many years old they are but give months and
Console.Write("{0} years, {1} months, {2} days", years, months, days
```

edited Oct 15 '13 at 12:16

community wiki
2 revs, 2 users 70%
Jacqueline Loriault

1 timespan itself automatically takes into account leap years between 2 dates so I'm not sure what your getting on about. I have asked on microsoft forums and microsoft has confirmed it takes into account leap years between 2 dates. – Jacqueline Loriault Oct 23 '13 at 19:29

1 Consider the following TWO senarios. 1st DateTime.Now is 1/1/2001 and a child is born on 1/1/2000. 2000 is a leap year and the result will be 1years, 0 months and 1 days. In the second senarion DateTime.Now is 1/1/2002 and the child is born on 1/1/2001. In this case the result will be 1 years, 0 months and 0 days. That will happen because you are adding the timespan on a non-leap year. If DateTime.MinValue was a leap year then the results would be 1 year at the first and 0 years 11 months and 30 days. (Try it in your code). – Athanasios Kataras Oct 24 '13 at 10:31

3 You are quite right it's not. But IF it was that would be the result. Why does it matter? It doesn't. In either case leap or not then there are examples where this does not work. That was what I wanted to show. The DIFF is correct. Span takes into account leap years. But ADDING to a base date is not. Try the examples in code and you will see I'm right. – Athanasios Kataras Mar 18 '15 at 18:20

24

I have created a SQL Server User Defined Function to calculate someone's age, given their birthdate. This is useful when you need it as part of a query:

```
using System;
using System.Data;
using System.Data.Sql;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;

public partial class UserDefinedFunctions
{
    [SqlFunction(DataAccess = DataAccessKind.Read)]
    public static SqlInt32 CalculateAge(string strBirthDate)
    {
        DateTime dtBirthDate = new DateTime();
        dtBirthDate = Convert.ToDateTime(strBirthDate);
        DateTime dtToday = DateTime.Now;

        // get the difference in years
        int years = dtToday.Year - dtBirthDate.Year;

        // subtract another year if we're before the
        // birth day in the current year
        if (dtToday.Month < dtBirthDate.Month || (dtToday.Month == d
dtToday.Day < dtBirthDate.Day))
            years=years-1;

        int intCustomerAge = years;
        return intCustomerAge;
    }
};
```

edited Feb 7 '16 at 0:07

community wiki
3 revs, 3 users 95%
user2601

22

I've spent some time working on this and came up with this to calculate someone's age in years, months and days. I've tested against the Feb 29th problem and leap years and it seems to work, I'd appreciate any feedback:

```
public void LoopAge(DateTime myDOB, DateTime FutureDate)
{
    int years = 0;
    int months = 0;
    int days = 0;

    DateTime tmpMyDOB = new DateTime(myDOB.Year, myDOB.Month, 1);

    DateTime tmpFutureDate = new DateTime(FutureDate.Year, FutureDat

    while (tmpMyDOB.AddYears(years).AddMonths(months) < tmpFutureDat
    {
        months++;

        if (months > 12)
        {
            years++;
            months = months - 12;
        }
    }

    if (FutureDate.Day >= myDOB.Day)
    {
        days = days + FutureDate.Day - myDOB.Day;
    }
    else
    {
        months--;

        if (months < 0)
        {
            years--;
            months = months + 12;
        }

        days +=
            DateTime.DaysInMonth(
                FutureDate.AddMonths(-1).Year, FutureDate.AddMonths(
            ) + FutureDate.Day - myDOB.Day;

    }

    //add an extra day if the dob is a leap day
    if (DateTime.IsLeapYear(myDOB.Year) && myDOB.Month == 2 && myDOB
    {
        //but only if the future date is less than 1st March
        if (FutureDate >= new DateTime(FutureDate.Year, 3, 1))
            days++;
    }

}
```

edited Feb 7 '16 at 0:07

community wiki
3 revs, 3 users 69%
Jon

22

Here's yet another answer:

```
public static int AgeInYears(DateTime birthday, DateTime today)
{
    return ((today.Year - birthday.Year) * 372 + (today.Month - birtl
(today.Day - birthday.Day)) / 372;
}
```

This has been extensively unit-tested. It does look a bit "magic". The number 372 is the number of days there would be in a year if every month had 31 days.

The explanation of why it works (lifted from here) is:

> Let's set `Yn = DateTime.Now.Year, Yb = birthday.Year, Mn = DateTime.Now.Month, Mb = birthday.Month, Dn = DateTime.Now.Day, Db = birthday.Day`
>
> ```
> age = Yn - Yb + (31*(Mn - Mb) + (Dn - Db)) / 372
> ```
>
> We know that what we need is either `Yn-Yb` if the date has already been reached, `Yn-Yb-1` if it has not.
>
> a) If `Mn<Mb`, we have `-341 <= 31*(Mn-Mb) <= -31 and -30 <= Dn-Db <= 30`
>
> ```
> -371 <= 31*(Mn - Mb) + (Dn - Db) <= -1
> ```
>
> With integer division
>
> ```
> (31*(Mn - Mb) + (Dn - Db)) / 372 = -1
> ```
>
> b) If `Mn=Mb` and `Dn<Db`, we have `31*(Mn - Mb) = 0 and -30 <= Dn-Db <= -1`
>
> With integer division, again

```
(31*(Mn - Mb) + (Dn - Db)) / 372 = -1
```

c) If `Mn>Mb` , we have `31 <= 31*(Mn-Mb) <= 341 and -30 <= Dn-Db <= 30`

```
1 <= 31*(Mn - Mb) + (Dn - Db) <= 371
```

With integer division

```
(31*(Mn - Mb) + (Dn - Db)) / 372 = 0
```

d) If `Mn=Mb` and `Dn>Db` , we have `31*(Mn - Mb) = 0 and 1 <= Dn-Db <= 3` 0

With integer division, again

```
(31*(Mn - Mb) + (Dn - Db)) / 372 = 0
```

e) If `Mn=Mb` and `Dn=Db` , we have `31*(Mn - Mb) + Dn-Db = 0`

and therefore `(31*(Mn - Mb) + (Dn - Db)) / 372 = 0`

edited Jul 5 '18 at 3:54

community wiki
3 revs, 3 users 83%
Matthew Watson

1 I stumbled into this long and annoying discussion, and your solution is a really nice and small approach. Thanks for keeping it simple – nabuchodonossor May 29 '18 at 9:43

Do we need to consider people who is smaller than 1 year? as Chinese culture, we describe small babies' age as 2 months or 4

19

weeks.

Below is my implementation, it is not as simple as what I imagined, especially to deal with date like 2/28.

```
public static string HowOld(DateTime birthday, DateTime now)
{
    if (now < birthday)
        throw new ArgumentOutOfRangeException("birthday must be less

    TimeSpan diff = now - birthday;
    int diffDays = (int)diff.TotalDays;

    if (diffDays > 7)//year, month and week
    {
        int age = now.Year - birthday.Year;

        if (birthday > now.AddYears(-age))
            age--;

        if (age > 0)
        {
            return age + (age > 1 ? " years" : " year");
        }
        else
        {// month and week
            DateTime d = birthday;
            int diffMonth = 1;

            while (d.AddMonths(diffMonth) <= now)
            {
                diffMonth++;
            }

            age = diffMonth-1;

            if (age == 1 && d.Day > now.Day)
                age--;

            if (age > 0)
            {
                return age + (age > 1 ? " months" : " month");
            }
            else
            {
                age = diffDays / 7;
                return age + (age > 1 ? " weeks" : " week");
            }
        }
    }
    else if (diffDays > 0)
    {
        int age = diffDays;
        return age + (age > 1 ? " days" : " day");
    }
    else
    {
        int age = diffDays;
        return "just born";
    }
}
```

This implementation has passed below test cases.

```
[TestMethod]
public void TestAge()
{
    string age = HowOld(new DateTime(2011, 1, 1), new DateTime(2012,
    Assert.AreEqual("1 year", age);

    age = HowOld(new DateTime(2011, 11, 30), new DateTime(2012, 11,
    Assert.AreEqual("1 year", age);

    age = HowOld(new DateTime(2001, 1, 1), new DateTime(2012, 11, 30
    Assert.AreEqual("11 years", age);

    age = HowOld(new DateTime(2012, 1, 1), new DateTime(2012, 11, 30
    Assert.AreEqual("10 months", age);

    age = HowOld(new DateTime(2011, 12, 1), new DateTime(2012, 11, 3
    Assert.AreEqual("11 months", age);

    age = HowOld(new DateTime(2012, 10, 1), new DateTime(2012, 11, 3
    Assert.AreEqual("1 month", age);

    age = HowOld(new DateTime(2008, 2, 28), new DateTime(2009, 2, 28
    Assert.AreEqual("1 year", age);

    age = HowOld(new DateTime(2008, 3, 28), new DateTime(2009, 2, 28
    Assert.AreEqual("11 months", age);

    age = HowOld(new DateTime(2008, 3, 28), new DateTime(2009, 3, 28
    Assert.AreEqual("1 year", age);

    age = HowOld(new DateTime(2009, 1, 28), new DateTime(2009, 2, 28
    Assert.AreEqual("1 month", age);

    age = HowOld(new DateTime(2009, 2, 1), new DateTime(2009, 3, 1))
    Assert.AreEqual("1 month", age);

    // NOTE.
    // new DateTime(2008, 1, 31).AddMonths(1) == new DateTime(2009,
    // new DateTime(2008, 1, 28).AddMonths(1) == new DateTime(2009,
    age = HowOld(new DateTime(2009, 1, 31), new DateTime(2009, 2, 28
    Assert.AreEqual("4 weeks", age);

    age = HowOld(new DateTime(2009, 2, 1), new DateTime(2009, 2, 28)
    Assert.AreEqual("3 weeks", age);

    age = HowOld(new DateTime(2009, 2, 1), new DateTime(2009, 3, 1))
    Assert.AreEqual("1 month", age);

    age = HowOld(new DateTime(2012, 11, 5), new DateTime(2012, 11, 3
    Assert.AreEqual("3 weeks", age);

    age = HowOld(new DateTime(2012, 11, 1), new DateTime(2012, 11, 3
    Assert.AreEqual("4 weeks", age);

    age = HowOld(new DateTime(2012, 11, 20), new DateTime(2012, 11,
    Assert.AreEqual("1 week", age);

    age = HowOld(new DateTime(2012, 11, 25), new DateTime(2012, 11,
    Assert.AreEqual("5 days", age);

    age = HowOld(new DateTime(2012, 11, 29), new DateTime(2012, 11,
    Assert.AreEqual("1 day", age);

    age = HowOld(new DateTime(2012, 11, 30), new DateTime(2012, 11,
    Assert.AreEqual("just born", age);

    age = HowOld(new DateTime(2000, 2, 29), new DateTime(2009, 2, 28
    Assert.AreEqual("8 years", age);

    age = HowOld(new DateTime(2000, 2, 29), new DateTime(2009, 3, 1)
    Assert.AreEqual("9 years", age);

    Exception e = null;

    try
```

```
        {
            age = HowOld(new DateTime(2012, 12, 1), new DateTime(2012, 1
        }
        catch (ArgumentOutOfRangeException ex)
        {
            e = ex;
        }

        Assert.IsTrue(e != null);
    }
```

Hope it's helpful.

edited Feb 7 '16 at 0:18

community wiki
2 revs, 2 users 73%
rockXrock

18

Keeping it simple (and possibly stupid:)).

```
DateTime birth = new DateTime(1975, 09, 27, 01, 00, 00, 00);
TimeSpan ts = DateTime.Now - birth;
Console.WriteLine("You are approximately " + ts.TotalSeconds.ToStrin
old.");
```

answered Aug 18 '10 at 14:29

community wiki
user181261

17

```
TimeSpan diff = DateTime.Now - birthdayDateTime;
string age = String.Format("{0:%y} years, {0:%M} months, {0:%d}, day
```

I'm not sure how exactly you'd like it returned to you, so I just made a readable string.

answered Sep 19 '13 at 15:18

community wiki
Dakotah Hicock

17

The simplest way I've ever found is this. It works correctly for the US and western europe locales. Can't speak to other locales, especially places like China. 4 extra compares, at most, following the initial computation of age.

```
public int AgeInYears(DateTime birthDate, DateTime referenceDate)
{
  Debug.Assert(referenceDate >= birthDate,
               "birth date must be on or prior to the reference date

  DateTime birth = birthDate.Date;
  DateTime reference = referenceDate.Date;
  int years = (reference.Year - birth.Year);

  //
  // an offset of -1 is applied if the birth date has
  // not yet occurred in the current year.
  //
  if (reference.Month > birth.Month);
  else if (reference.Month < birth.Month)
    --years;
  else // in birth month
  {
    if (reference.Day < birth.Day)
      --years;
  }

  return years ;
}
```

I was looking over the answers to this and noticed that nobody has made reference to regulatory/legal implications of leap day births. For instance, per Wikipedia, if you're born on February 29th in various jurisdictions, you're non-leap year birthday varies:

- In the United Kingdom and Hong Kong: it's the ordinal day of the year, so the next day, March 1st is your birthday.
- In New Zealand: it's the previous day, February 28th for the purposes of driver licencing, and March 1st for other purposes.
- Taiwan: it's February 28th.

And as near as I can tell, in the US, the statutes are silent on the matter, leaving it up to the common law and to how various regulatory bodies define things in their regulations.

To that end, an improvement:

```
public enum LeapDayRule
{
  OrdinalDay     = 1 ,
  LastDayOfMonth = 2 ,
}

static int ComputeAgeInYears(DateTime birth, DateTime reference, Lea
ruleInEffect)
{
  bool isLeapYearBirthday = CultureInfo.CurrentCulture.Calendar.IsLe
birth.Month, birth.Day);
  DateTime cutoff;

  if (isLeapYearBirthday && !DateTime.IsLeapYear(reference.Year))
  {
    switch (ruleInEffect)
    {
      case LeapDayRule.OrdinalDay:
        cutoff = new DateTime(reference.Year, 1, 1)
                             .AddDays(birth.DayOfYear - 1);
        break;

      case LeapDayRule.LastDayOfMonth:
        cutoff = new DateTime(reference.Year, birth.Month, 1)
                             .AddMonths(1)
                             .AddDays(-1);
        break;

      default:
        throw new InvalidOperationException();
    }
  }
  else
  {
    cutoff = new DateTime(reference.Year, birth.Month, birth.Day);
  }

  int age = (reference.Year - birth.Year) + (reference >= cutoff ? 0
  return age < 0 ? 0 : age;
}
```

It should be noted that this code assumes:

- A western (European) reckoning of age, and
- A calendar, like the Gregorian calendar that inserts a single leap day at the end of a month.

edited Jul 5 '18 at 3:53

community wiki
8 revs, 5 users 66%
Nicholas Carey

This is not a direct answer, but more of a philosophical reasoning

16

about the problem at hand from a quasi-scientific point of view.

I would argue that the question does not specify the unit nor culture in which to measure age, most answers seem to assume an integer annual representation. The SI-unit for time is `second`, ergo the correct generic answer should be (of course assuming normalized `DateTime` and taking no regard whatsoever to relativistic effects):

```
var lifeInSeconds = (DateTime.Now.Ticks - then.Ticks)/TickFactor;
```

In the Christian way of calculating age in years:

```
var then = ... // Then, in this case the birthday
var now = DateTime.UtcNow;
int age = now.Year - then.Year;
if (now.AddYears(-age) < then) age--;
```

In finance there is a similar problem when calculating something often referred to as the *Day Count Fraction*, which roughly is a number of years for a given period. And the age issue is really a time measuring issue.

Example for the actual/actual (counting all days "correctly") convention:

```
DateTime start, end = .... // Whatever, assume start is before end

double startYearContribution = 1 - (double) start.DayOfYear / (doubl
(DateTime.IsLeapYear(start.Year) ? 366 : 365);
double endYearContribution = (double)end.DayOfYear / (double)
(DateTime.IsLeapYear(end.Year) ? 366 : 365);
double middleContribution = (double) (end.Year - start.Year - 1);

double DCF = startYearContribution + endYearContribution + middleCon
```

Another quite common way to measure time generally is by "serializing" (the dude who named this date convention must seriously have been trippin'):

```
DateTime start, end = .... // Whatever, assume start is before end
int days = (end - start).Days;
```

I wonder how long we have to go before a relativistic age in seconds becomes more useful than the rough approximation of earth-around-sun-cycles during one's lifetime so far :) Or in other words, when a period must be given a location or a function representing motion for itself to be valid :)

edited Oct 31 '17 at 11:32

community wiki
2 revs, 2 users 97%
flindeberg

16

Here is a solution.

```
DateTime dateOfBirth = new DateTime(2000, 4, 18);
DateTime currentDate = DateTime.Now;

int ageInYears = 0;
int ageInMonths = 0;
int ageInDays = 0;

ageInDays = currentDate.Day - dateOfBirth.Day;
ageInMonths = currentDate.Month - dateOfBirth.Month;
ageInYears = currentDate.Year - dateOfBirth.Year;

if (ageInDays < 0)
{
    ageInDays += DateTime.DaysInMonth(currentDate.Year, currentDate.I
    ageInMonths = ageInMonths--;

    if (ageInMonths < 0)
    {
        ageInMonths += 12;
```

```
            ageInYears--;
        }
    }

    if (ageInMonths < 0)
    {
        ageInMonths += 12;
        ageInYears--;
    }

    Console.WriteLine("{0}, {1}, {2}", ageInYears, ageInMonths, ageInDay
```

edited Feb 22 '18 at 16:36

community wiki
2 revs, 2 users 71%
Rajeshwaran S P

15

This is one of the most accurate answer that is able to resolve the birthday of 29th of Feb compare to any year of 28th Feb.

```
public int GetAge(DateTime birthDate)
{
    int age = DateTime.Now.Year - birthDate.Year;

    if (birthDate.DayOfYear > DateTime.Now.DayOfYear)
        age--;

    return age;
}
```

edited Feb 7 '16 at 0:07

community wiki
2 revs, 2 users 89%
mjb

14

I have a customized method to calculate age, plus a bonus validation message just in case it helps:

```
public void GetAge(DateTime dob, DateTime now, out int years, out in
days)
{
    years = 0;
    months = 0;
    days = 0;

    DateTime tmpdob = new DateTime(dob.Year, dob.Month, 1);
    DateTime tmpnow = new DateTime(now.Year, now.Month, 1);

    while (tmpdob.AddYears(years).AddMonths(months) < tmpnow)
    {
        months++;
        if (months > 12)
        {
            years++;
            months = months - 12;
        }
    }

    if (now.Day >= dob.Day)
        days = days + now.Day - dob.Day;
    else
    {
        months--;
        if (months < 0)
        {
            years--;
            months = months + 12;
        }
        days += DateTime.DaysInMonth(now.AddMonths(-1).Year, now.AddM
now.Day - dob.Day;
    }
    if (DateTime.IsLeapYear(dob.Year) && dob.Month == 2 && dob.Day =
DateTime(now.Year, 3, 1))
        days++;

}

private string ValidateDate(DateTime dob) //This method will validat
{
    int Years = 0; int Months = 0; int Days = 0;

    GetAge(dob, DateTime.Now, out Years, out Months, out Days);

    if (Years < 18)
        message =  Years + " is too young. Please try again on your
    else if (Years >= 65)
        message = Years + " is too old. Date of Birth must not be 65
    else
        return null; //Denotes validation passed
}
```

Method call here and pass out datetime value (MM/dd/yyyy if server set to USA locale). Replace this with anything a messagebox or any container to display:

```
DateTime dob = DateTime.Parse("03/10/1982");

string message = ValidateDate(dob);

lbldatemessage.Visible = !StringIsNullOrWhitespace(message);
lbldatemessage.Text = message ?? ""; //Ternary if message is null th
string
```

Remember you can format the message any way you like.

edited Jan 2 '18 at 9:45

community wiki
7 revs, 3 users 49%
Knickerless-Noggins

13

How about this solution?

```
static string CalcAge(DateTime birthDay)
{
    DateTime currentDate = DateTime.Now;
    int approximateAge = currentDate.Year - birthDay.Year;
    int daysToNextBirthDay = (birthDay.Month * 30 + birthDay.Day) -
        (currentDate.Month * 30 + currentDate.Day) ;

    if (approximateAge == 0 || approximateAge == 1)
    {
        int month =  Math.Abs(daysToNextBirthDay / 30);
        int days = Math.Abs(daysToNextBirthDay % 30);

        if (month == 0)
            return "Your age is: " + daysToNextBirthDay + " days";

        return "Your age is: " + month + " months and " + days + " d
    }

    if (daysToNextBirthDay > 0)
        return "Your age is: " + --approximateAge + " Years";

    return "Your age is: " + approximateAge + " Years"; ;
}
```

edited Feb 7 '16 at 0:14

community wiki
4 revs, 4 users 66%
Doron

```
private int GetAge(int _year, int _month, int _day
{
```

11

```
    DateTime yourBirthDate= new DateTime(_year, _month, _day);

    DateTime todaysDateTime = DateTime.Today;
    int noOfYears = todaysDateTime.Year - yourBirthDate.Year;

    if (DateTime.Now.Month < yourBirthDate.Month ||
        (DateTime.Now.Month == yourBirthDate.Month && DateTime.Now.Da
yourBirthDate.Day))
    {
        noOfYears--;
    }

    return  noOfYears;
}
```

edited Feb 7 '16 at 0:13

community wiki
3 revs, 3 users 67%
AEMLoviji

9

The following approach (extract from Time Period Library for .NET class *DateDiff*) considers the calendar of the culture info:

```
// ----------------------------------------------------------------
private static int YearDiff( DateTime date1, DateTime date2 )
{
  return YearDiff( date1, date2, DateTimeFormatInfo.CurrentInfo.Cale
} // YearDiff

// ----------------------------------------------------------------
private static int YearDiff( DateTime date1, DateTime date2, Calenda
{
  if ( date1.Equals( date2 ) )
  {
    return 0;
  }
  int year1 = calendar.GetYear( date1 );
  int month1 = calendar.GetMonth( date1 );
  int year2 = calendar.GetYear( date2 );
  int month2 = calendar.GetMonth( date2 );

  // find the the day to compare
  int compareDay = date2.Day;
  int compareDaysPerMonth = calendar.GetDaysInMonth( year1, month1 )
  if ( compareDay > compareDaysPerMonth )
  {
    compareDay = compareDaysPerMonth;
  }

  // build the compare date
  DateTime compareDate = new DateTime( year1, month2, compareDay,
    date2.Hour, date2.Minute, date2.Second, date2.Millisecond );
  if ( date2 > date1 )
  {
    if ( compareDate < date1 )
    {
      compareDate = compareDate.AddYears( 1 );
    }
  }
  else
  {
    if ( compareDate > date1 )
    {
      compareDate = compareDate.AddYears( -1 );
    }
  }
  return year2 - calendar.GetYear( compareDate );
} // YearDiff
```

Usage:

```
// ----------------------------------------------------------------
public void CalculateAgeSamples()
{
  PrintAge( new DateTime( 2000, 02, 29 ), new DateTime( 2009, 02, 28
  // > Birthdate=29.02.2000, Age at 28.02.2009 is 8 years
  PrintAge( new DateTime( 2000, 02, 29 ), new DateTime( 2012, 02, 28
  // > Birthdate=29.02.2000, Age at 28.02.2012 is 11 years
} // CalculateAgeSamples

// ----------------------------------------------------------------
```

```
public void PrintAge( DateTime birthDate, DateTime moment )
{
  Console.WriteLine( "Birthdate={0:d}, Age at {1:d} is {2} years", b
YearDiff( birthDate, moment ) );
} // PrintAge
```

edited Jul 5 '18 at 3:54

community wiki
2 revs, 2 users 99%
user687474

8

I used ScArcher2's solution for an accurate Year calculation of a persons age but I needed to take it further and calculate their Months and Days along with the Years.

```
    public static Dictionary<string,int> CurrentAgeInYearsMonthsDays
ndtBirthDate, DateTime? ndtReferralDate)
    {
        //------------------------------------------------------------
        // Can't determine age if we don't have a dates.
        //------------------------------------------------------------
        if (ndtBirthDate == null) return null;
        if (ndtReferralDate == null) return null;

        DateTime dtBirthDate = Convert.ToDateTime(ndtBirthDate);
        DateTime dtReferralDate = Convert.ToDateTime(ndtReferralDate

        //------------------------------------------------------------
        // Create our Variables
        //------------------------------------------------------------
        Dictionary<string, int> dYMD = new Dictionary<string,int>();
        int iNowDate, iBirthDate, iYears, iMonths, iDays;
        string sDif = "";

        //------------------------------------------------------------
        // Store off current date/time and DOB into local variables
        //------------------------------------------------------------
        iNowDate = int.Parse(dtReferralDate.ToString("yyyyMMdd"));
        iBirthDate = int.Parse(dtBirthDate.ToString("yyyyMMdd"));

        //------------------------------------------------------------
        // Calculate Years
        //------------------------------------------------------------
        sDif = (iNowDate - iBirthDate).ToString();
        iYears = int.Parse(sDif.Substring(0, sDif.Length - 4));

        //------------------------------------------------------------
        // Store Years in Return Value
        //------------------------------------------------------------
        dYMD.Add("Years", iYears);

        //------------------------------------------------------------
        // Calculate Months
        //------------------------------------------------------------
        if (dtBirthDate.Month > dtReferralDate.Month)
            iMonths = 12 - dtBirthDate.Month + dtReferralDate.Month
        else
            iMonths = dtBirthDate.Month - dtReferralDate.Month;

        //------------------------------------------------------------
        // Store Months in Return Value
        //------------------------------------------------------------
        dYMD.Add("Months", iMonths);

        //------------------------------------------------------------
        // Calculate Remaining Days
        //------------------------------------------------------------
        if (dtBirthDate.Day > dtReferralDate.Day)
            //Logic: Figure out the days in month previous to the cu
admitted month.
            //       Subtract the birthday from the total days which
many days the person has lived since their birthdate day the previou
            //       then take the referral date and simply add the
person has lived this month.

            //If referral date is january, we need to go back to the
December to get the days in that month.
            if (dtReferralDate.Month == 1)
                iDays = DateTime.DaysInMonth(dtReferralDate.Year - 1
dtBirthDate.Day + dtReferralDate.Day;
            else
                iDays = DateTime.DaysInMonth(dtReferralDate.Year, dt
1) - dtBirthDate.Day + dtReferralDate.Day;
```

```
            else
                iDays = dtReferralDate.Day - dtBirthDate.Day;

            //------------------------------------------------------------
            // Store Days in Return Value
            //------------------------------------------------------------
            dYMD.Add("Days", iDays);

            return dYMD;
    }
```

edited Oct 14 '12 at 12:05

community wiki
3 revs, 2 users 99%
Dylan Hayes

8

SQL version:

```
declare @dd smalldatetime = '1980-04-01'
declare @age int = YEAR(GETDATE())-YEAR(@dd)
if (@dd> DATEADD(YYYY, -@age, GETDATE())) set @age = @age -1

print @age
```

answered Jun 30 '16 at 11:24

community wiki
xenedia

7

I've made one small change to Mark Soen's answer: I've rewriten the third line so that the expression can be parsed a bit more easily.

```
public int AgeInYears(DateTime bday)
{
    DateTime now = DateTime.Today;
    int age = now.Year - bday.Year;
    if (bday.AddYears(age) > now)
        age--;
    return age;
}
```

I've also made it into a function for the sake of clarity.

edited Feb 22 '18 at 16:38

community wiki
3 revs, 2 users 74%
cdiggins

1 2 3 next