Integers vs. Doubles

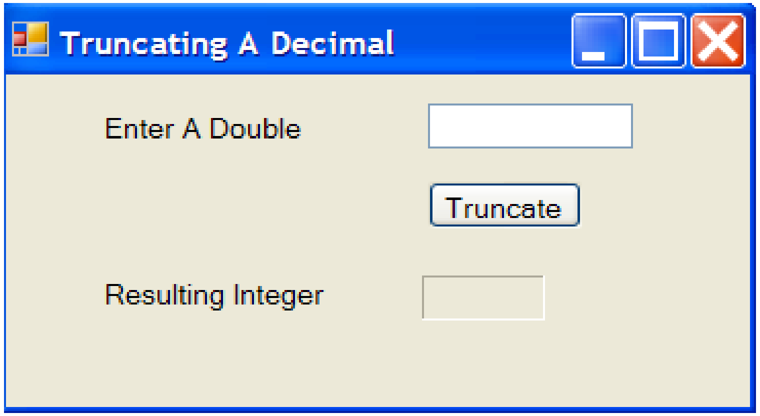
An integer variable can hold a numeric value between -1.2 billion and +1.2 billion. These values must be whole numbers, however. No decimal places can be held.

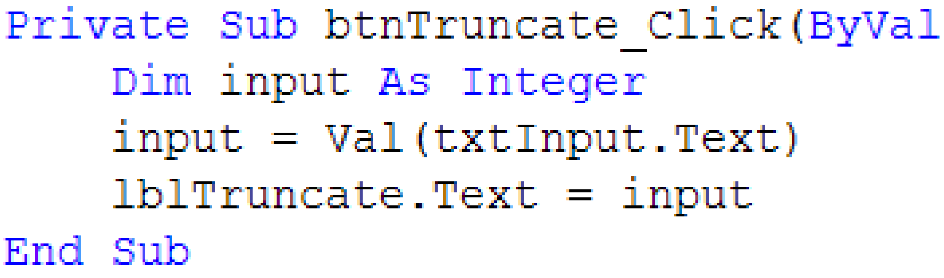
If you try and put a decimal value into an integer it ignores everything after the decimal place. This is known as truncating which means shortening or cutting off.

Decimal Input	Resulting Integer
2.4	2
2.9	2
2.1	2
2.0	2
-4.2	-4
109.3	109

Truncating should not be confused with rounding off. Truncating involves ignoring everything after the decimal place.
Even a number asclose to 3 as 2.99999999999999 simply becomes 2 if assigned to an integer variable.

Integers Do Not Hold Decimal Places





```
Private Sub btnTruncate_Click(ByVal
    Dim input As Integer
    input = Val(txtInput.Text)
    lblTruncate.Text = input
End Sub
```

Write this program up and call it IntegerTruncate.
Test some decimal values.

Integer Division

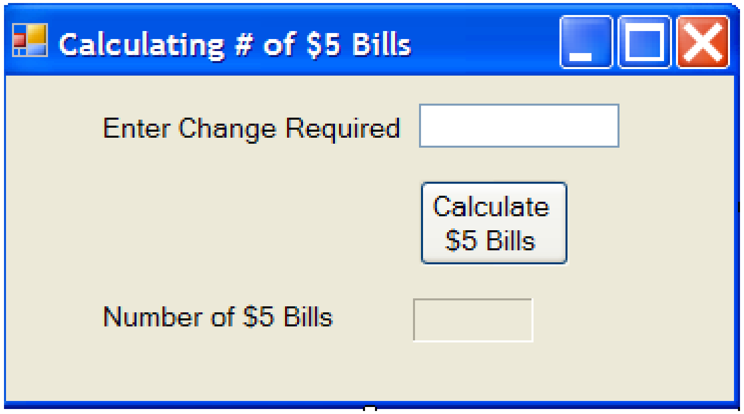
The fact that integers do not allow decimal places can be used in some interesting ways particularly in cases where an even number is required.

Consider a scenario in which a cashier is giving change in the amount $62.35. If the largest bill he has in the till is $5 bills, how many of these will he need to count out.

Straight division of 62.35/5 is of limited use. The answer of 12.47 suggests that we should chop a bill in sections to make up .47 of a bill. This, of course, is not advisable (actually, I think it may even be illegal).

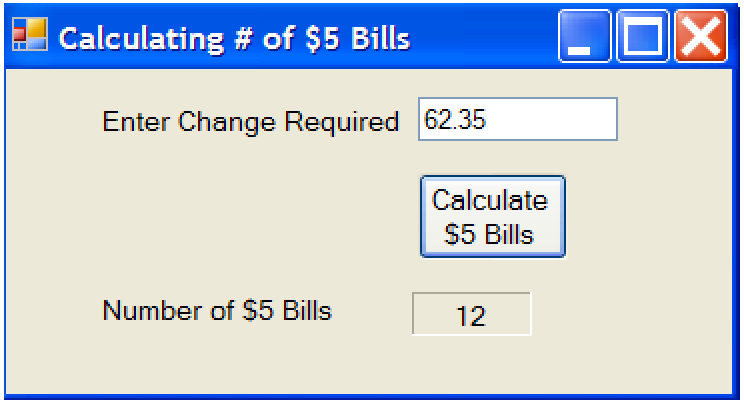
Lets, instead create an integer variable called numberOfFives to reflect the number of five dollar bills that will be given as change.

Calculating Number of $5 Bills



What would happen to the calculations if this was declared as a Double instead.

```
Private Sub btnTruncate_Click(ByVal sender
    Dim numberOfFives As Integer
    Dim change As Decimal
    change = Val(txtInput.Text)
    numberOfFives = change / 5
    lblTruncate.Text = numberOfFives
End Sub
```



Enter Change Required 62.35

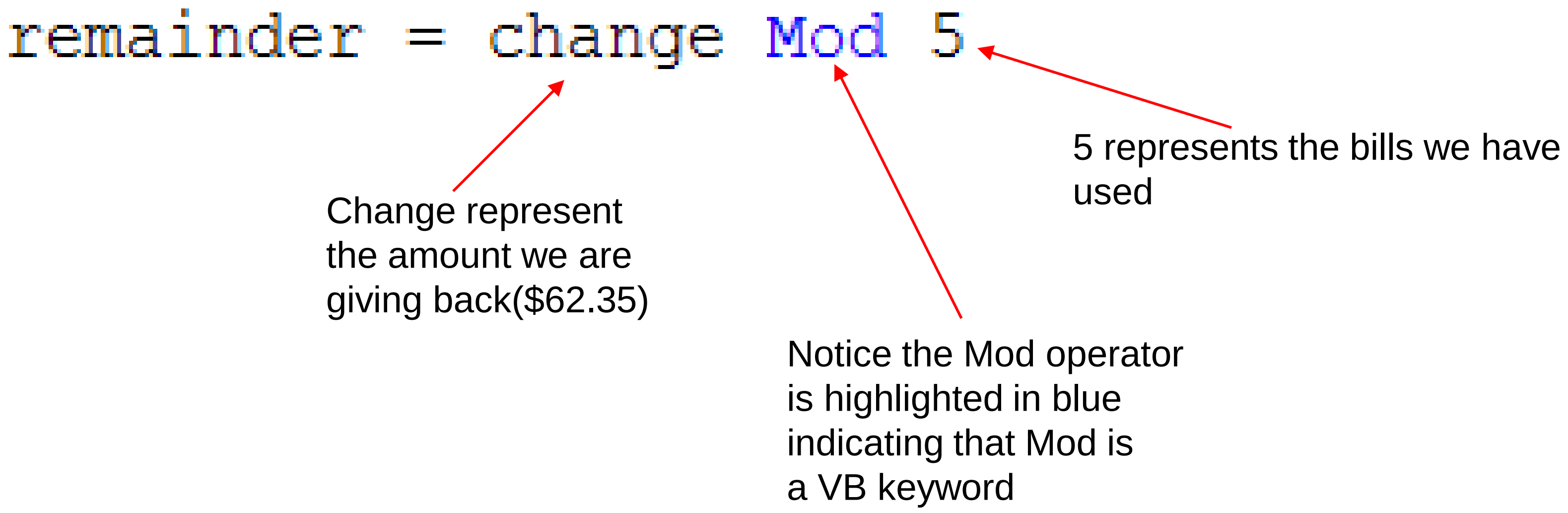
Number of $5 Bills 12

Remainder

After our cashier has given out 12 $5 bills, the next item of concern will be, how much more change do we need to give back. In this example, it should be obvious that the outstanding amount will be $2.35 . For more complex examples there is a math operator in Visual Basic that allows us easily calculate this.

It is called the modulus or sometimes the remainder operator and it is simply the word Mod.

To calculate the change required for $ 62.35 after the five dollar bills have been distributed we simple write:



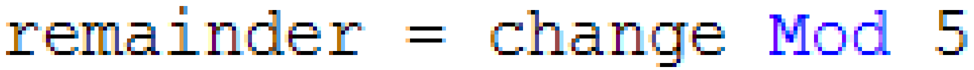
```
remainder = change Mod 5
```

Change represent the amount we are giving back($62.35)

Notice the Mod operator is highlighted in blue indicating that Mod is a VB keyword

5 represents the bills we have used

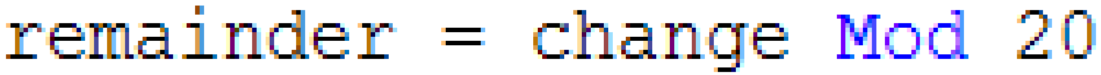
The Mod Operator



```
remainder = change Mod 5
```

Another way of translating this statement is;
After 5 goes into 62.35 as many times as it can
evenly, what will be left over.

Imagine that our cashier suddenly discovered he has some $20 bills. He knows that he can now pay out 3 $20 bills evenly, so now what is the change he still has to pay out.



```
remainder = change Mod 20
```

Well as it turns out, the answer is still $2.35 because 3 X 20 is 60 and the remainder is still $2.35

Answer the following

10 Mod 3	1
10 Mod 4	2
10 Mod 6	4
10 Mod 5	0
27 Mod 4	3
27 Mod 3	0
31 Mod 8	7
115 Mod 2	1
115 Mod 10	5
115 Mod 5	0

Soldiers Transport

287 soldiers need to get to their camp. They have troop transporters which can carry 17 men each. Assuming that the transporter carry a full load each whenever possible, how many men will be in the last transport.

? = 287 Mod 17
Answer is 15

Soldiers Transport

288 soldiers need to get to their camp. They have troop transporters which can now carry 16 men each. Assuming that the transporter carry a full load each whenever possible, how many men will be in the last transport.

? = 288 Mod 16
Answer is 0

Since a troop transport would never travel with 0, the answer is 16. The last full load of 16 men would have brought in the last of the men. 16 goes into 288 an even number of times…Tricky, that Mr. Wright!!

Commenting Your Code

Programs generally are not written and then left unchanged. Usually programs are being constantly updated and improved.
In many cases the person who writes the original code is not the same person or people who do the modifications, updates and bug fixes.
It can be very difficult to look at code someone else has written and always know the reason for the use of a particular piece of code.

In fact, even if the same person who wrote the original code is re-visiting their code, it can be difficult to remember all the details of how and why some code was written.

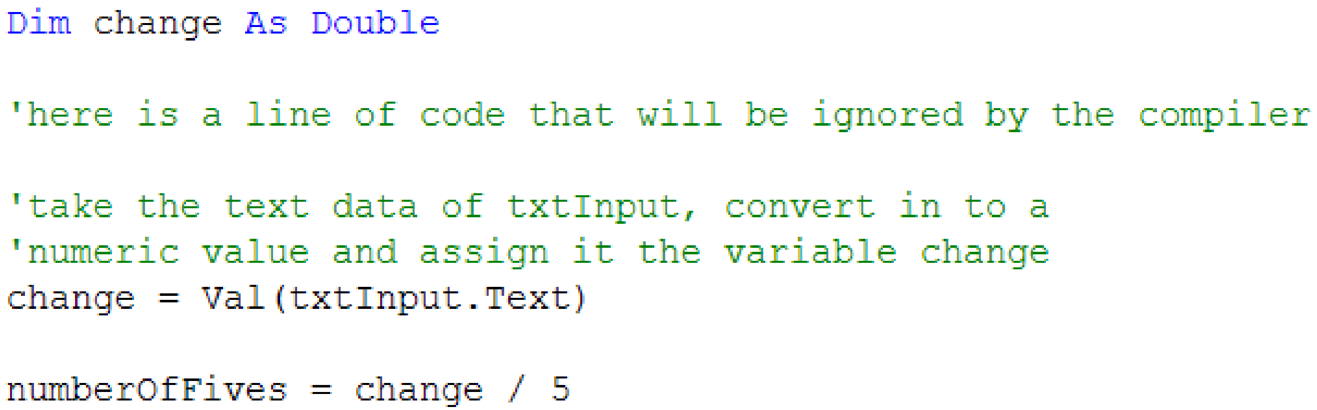
To help with this problem, it is useful and advisable to write notes describing the purpose of the code. This process is called commenting the code.

Since the compiler does not read code the humans do, it is necessary to inicate to the compiler what sections are for the computer to run and what sections are just for the programmers to read.

Code Commenting

By putting an apostrophe at the beginning of a line, a programmer can indicate to the compiler that it should ignore the line.

In this code snippet, there are 3 lines of comments.
The compiler see the apostrophe and ignores those lines
The lines show up in green font which makes it easier for programmers to separate it from the regular code.



```
Dim change As Double

'here is a line of code that will be ignored by the compiler

'take the text data of txtInput, convert in to a
'numeric value and assign it the variable change
change = Val(txtInput.Text)

numberOfFives = change / 5
```

As shown here, the comments can be used to desribe and and explain the function of the code for future understanding.

Comment, Comment, Comment

Most programming projects are worked on by teams of programmers.
It is necessary for each programmer to understand the code written by other programmers on the team so that the code can be fitted together.

Code can be complex. Even if you are the only working on a program, it is easy to forget your own purpose in writing a piece of code when you re-visit a year, a month or even a day later!

These programs are your study material. Writing comments into your code is extremely useful for study puposes.

Comment, Comment, Comment!!!!