Computing: HexToDec

1. The task

A decimal number (base 10) can be written as a sum of powers of ten, for example, 3241 can be represented as

$3 \times 10^3 + 2 \times 10^2 + 4 \times 10^1 + 1 \times 10^0$.

A base 16 (hexadecimal) number has the following digits

0 1 2 3 4 5 6 7 8 9 A B C D E F (where A=10, B=11 …)

Similarly $2FA_{\text{base } 16}$ can be represented as

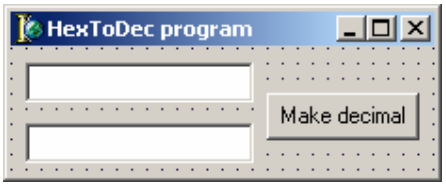
$2 \times 16^2 + 15 \times 16^1 + 10 \times 16^0 = 762_{\text{base } 10}$

Write a function to convert hexadecimal numbers to decimal numbers.

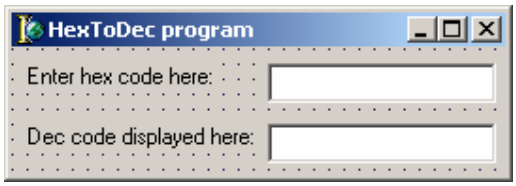
Hints:
Test it using the built-in procedure IntToHex.

Extension
Show how to place it (a) where it can be accessed by one event handler only, and (b) where it can be accessed by any event handler in the current unit.

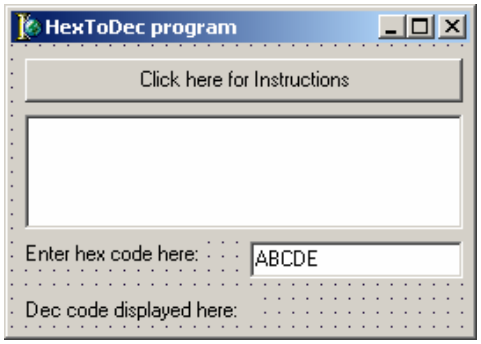
2. Design and development



Initial design: My first design consisted of two text boxes and a button. The user entered the hexadecimal value into the first textbox, pressed the convert button and the decimal value was returned.



Design development: Using a button made the interface slower to use. Instead I changed the time when the procedure was executed, instead of on the click of the button I assigned this to the OnKeyUp event in the text box. I added labels to the program to aid the user. I included the functionality to change the string all to upper case (i.e. a to A, b to B…) before processing so the user can enter either upper case, lower case or a combination of characters.



Final design: I added a button that displays instructions into a memo box that instruct the user on how to use the program to complete my design. I changed the default value of the hexadecimal entry to a sample entry so that if the user did not change the text the program would compile correctly. I also added a validation routine to check that the user had entered a valid string into the textbox. As text entered in the decimal box serves no functionality I changed the design of my program to display the result in a label component instead of a textbox.

Pseudo-code:

Defining the function HexToDec:

```
DefineFunction HexToDec(In:String,Str;Out:Integer,Result)
DefineVariables
  i, j, TermPower areof Integer
start
```

```
  TermPower ← 1
  for j ← 1 to Length(Str) - 1
    TermPower ← TermPower * 16
  end
  Result ← 0
  Str ← MakeAllUpperCase(Str)
  for I ← 1 to Length(Str)
    case Str[i]
      1..9 → Result ← Result + Str[i] * TermPower
      A..F → Result ← Result + (ASCII(Str[i])-55) * TermPower
      * → ShowErrorMessage(Character Str[i] is invalid hexadecimal)
    TermPower ← TermPower divideby 16
end
```

OnKeyUp Procedure for txtEntry, the textbox in which the hexadecimal value is entered:

```
DefineProcedure txtEntry:OnKeyUp
start
    lblResult ← HexToDec(txtEntry)
end
```

OnClick Procedure for btnInstructions

```
DefineProcedure btnInstructions.OnClick
start
  memobox.addlines(Enter a hexadecimal value into the first text box and the
                   decimal value will appear in the second text box.)
end
```

Data dictionary

Identifier	**Type**	**Formatting and reasoning**
i, j	Integer	Loop counter
TermPower	Integer	Power of 16 that the term is to be multiplied by. It would be possible to create the loop that assigns the value to TermPower initially in its own function, however I did not see this as necessary as the function is only used once.
Str	String (of Hex. values)	String given to function, read from the txtEntry textbox.
Result	String (of Dec. values)	Resultant value printed to the txtResult textbox.

3. Problems

I encountered a variety of problems whilst creating the program, which I have documented below.

Most of the difficulties I had were in my initial approach to the problem. The pseudo-code I initially created for the function HexToDec did not take into account that a numeric value meant a two-digit multiplier would need to be created. I then re-wrote the pseudo-code using a case statement as can be seen above.

If a string longer than 8 characters were entered into the textbox the displayed value would be 0 as the value exceeds the range of the integer type. Therefore I limited the number of characters that can be entered into the text box to 8.

When the hexadecimal value were entered in lower case the program would not process the characters correctly, as the computer makes a distinction between the upper and lower case letters. Using the AnsiUpperCase() function the string was internally manipulated before processing to all upper case characters. I also added a validation routine to check that the characters were valid hexadecimal as the program regarded any invalid character as a 0 entry.

4. Unit listing

Here is the source code copied from *Delphi* (*Automatically generated code omitted*):

```
//
// Define HexToDec function
//
function HexToDec(Str: string): Integer;
var
  i, j, TermPower: Integer;
begin
  TermPower:=1;
//
// Calculate highest power of 16 required for given string length
//
  for j:=1 to (Length(Str)-1) do begin
    TermPower:=TermPower * 16;
  end;
  Result:=0;
//
// Make string all upper case
//
  Str:=UpperCase(Str);
//
// Analyse each character in the string, performing one of three operations
//
  for i:=1 to Length(Str) do begin
    case Str[i] of
//
//  For numerical values multiply by the appropriate power of 16
//
      '1'..'9': Result:=Result+(StrToInt(Str[i]))*TermPower;
//
//  For alphabetical values convert to numerical value first then multiply by
//  appropriate power of 16
//
      'A'..'F': Result:=Result+(Ord(Str[i])-55)*TermPower;
//
//  For values that are neither numerical or valid alphabetical characters
//  display an error message
//
      else ShowMessage('Character ' + Str[i] + ' is invalid hexadecimal.');
    end;
//
//  At the end of each cycle of the iteration divide the highest power of 16
//  necessary for the string by 16. For the last character in the string this
//  multiplier will be 1
//
    TermPower:=TermPower div 16;
  end;
end;

procedure TfrmMain.txtEntryKeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
//
// OnKeyUp in the txtEntry textbox the HexToDec function is executed
//
begin
    lblResult.Caption:=IntToStr(HexToDec(txtEntry.Text));
end;

end.
```

5. Testing

Test	**Why testing**	**Expected outcome**	**Actual outcome (T/F)**
Individual hex characters, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.	Check characters are correctly converted.	Characters are converted to decimal correctly.	As expected – T.
Lowercase alphabetical individual characters, a, b, c,	Check lowercase characters can be entered.	Lowercase characters are not recognised.	As expected - T.

d, e, f.			

However, this feature would be useful in my program so I searched the Delphi help to find a function to turn a string into all upper case values. The Delphi help suggested the *UpperCase()* function, which I implemented in my program. On re-testing the lowercase values were accepted by my program.

Test	**Why testing**	**Expected outcome**	**Actual outcome (T/F)**
Enter invalid characters	Check validation routine works correctly	Validation routine generates error that the character is invalid hexadecimal	As expected, see *Figure 1* – T.
Error message disappears on mouse click or keyboard stroke	Check error message is only called once	Error message is called once.	Error message is repeatedly called if the message box is closed with a keystroke – F.

Figure 1.

Choosing to close the error message with a key press will not work as the program is set to execute the procedure OnKeyPress when the cursor is in the txtEntry textbox. Therefore when the key is pressed on the error message the procedure is executed again and so the error message is displayed again. To solve this problem I could choose to execute the procedure on a different event, a suitable alternative would be the OnExit event. However this would compromise the ease of use of the program.

A useful alternative would be to have got the error message to print to the lblResult label, however I could not achieve this from inside the procedure, and so have decided to leave this bug in the program.

Test	**Why testing**	**Expected outcome**	**Actual outcome (T/F)**
Check multiplication of appropriate powers of 16 is correct by testing consecutive values of numerical values ('12'), consecutive alphabetical values ('AB') and then a mixture ('B4,' '9E').	Check correct conversion to base 16. If the test works for two consecutive figures it will work for 3, 4 and so on due to the logic of the program.	Correct results are generated (Tested using DecToHex function in Delphi).	As expected - T. *See Figure 2.*
Length of txtEntry textbox limited to 8 characters.	Check limit has been applied correctly.	No more than 8 characters can be entered.	As expected – T.

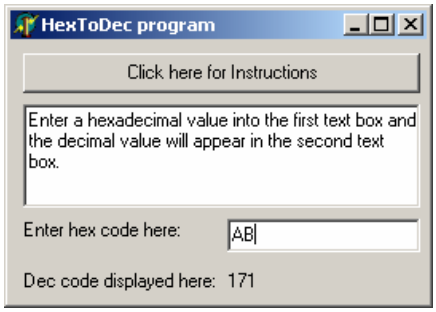
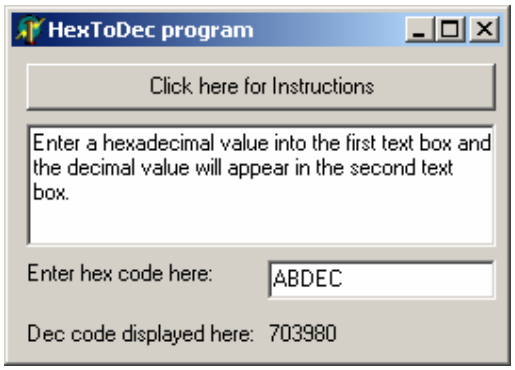


Figure 2.

6. Sample run

Here is a screenshot showing a sample run of my program.



7. Appraisal

The program that I have created meets the brief, the function HexToDec successfully converts an entry in hexadecimal to an output in decimal. To meet the extension; the code is currently positioned in a place in the unit where it is accessible by all event handlers. To make it visible only to the current event handler the function must be defined within the procedure, after the variable declarations (var) and before the beginning of the body of the program (begin).

I extended the brief further by using the UpperCase function which I found using the *Delphi* help, by adding an instructions button and memo box and a validation routine.

A known issue with the program is that the error message ("Invalid character…") must be clicked with the mouse. Choosing to close the error message with a key press will not work as the program is set to execute the procedure OnKeyPress when the cursor is in the txtEntry textbox. Therefore when the key is pressed on the error message the procedure is executed again and so the error message is displayed again. To cure this error I would need to assign the event to a different event, OnExit would be a suitable alternative. A better approach would be to use an 'if' statement where if the key pressed was enter then the procedure would not be called.

I found writing the pseudo-code for this task very helpful. I had initially found the task complicated and had tried a variety of approaches, using arrays for example. Using a string however allowed for a variable length input and made the code easier.

During the creation of this program I have gained a sound understanding of functions and improved my approach to tasks.