

### Objectives

#### Program Purpose

- Display information about the first 1000 integers. Display whether they are:
  - prime or composite;
  - abundant, deficient or perfect.
- Display the first 1000 triangle, square, oblong, pentagonal and hexagonal numbers.
- Allow a minimum and maximum to be selected for the range of numbers.
- Improve the information provided to the program user.

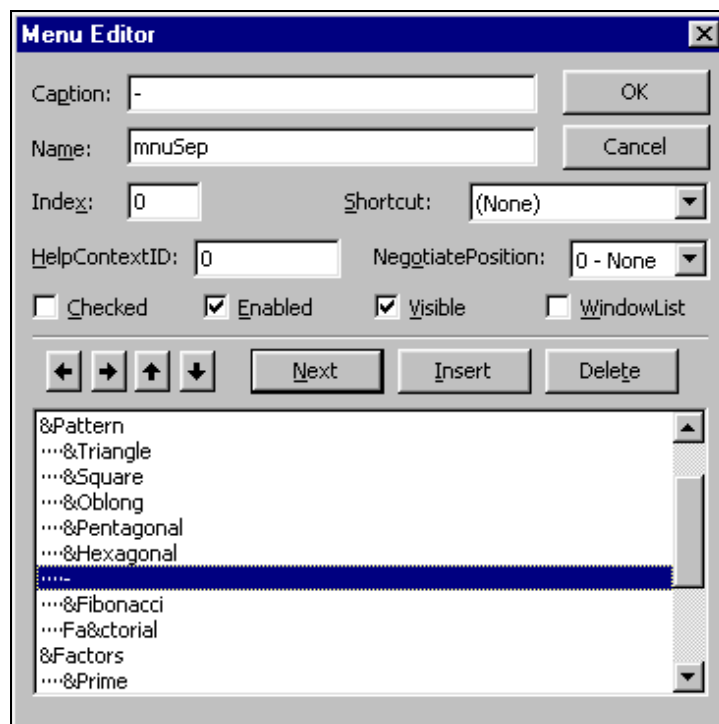
#### Learning Goals

- Algorithmic Thinking applied to number patterns and series.
- Reuse the module from Program Example 25.
- Simplifying code.
- Improving the interface and customising Program 25.
- User defined TYPE...END TYPE declarations.
- Use records and arrays together.
- Error handling of variable overflows.
- Extended use of the menu editor.

### Design Notes

This project is an extension and follow up to **Program Example 25 – Number Series 1**. We will attempt to improve the interface and useability of the program. We will also rationalise and simplify the code.

To activate the **Menu Editor**, hit **Ctrl+E** while in **Design** mode or go to: **Tools/Menu Editor** while the form is active in **Design Mode**. It is fairly straightforward to use. Set a caption and name for each menu item. Use the arrow keys to indent menu items. Experiment with the settings to arrive at your desired menu. To create a separator bar type the character '-' in the caption text box.



Add the module from the previous program example. To add a Module go to **Project/Add Module**. Two changes are necessary in the code.

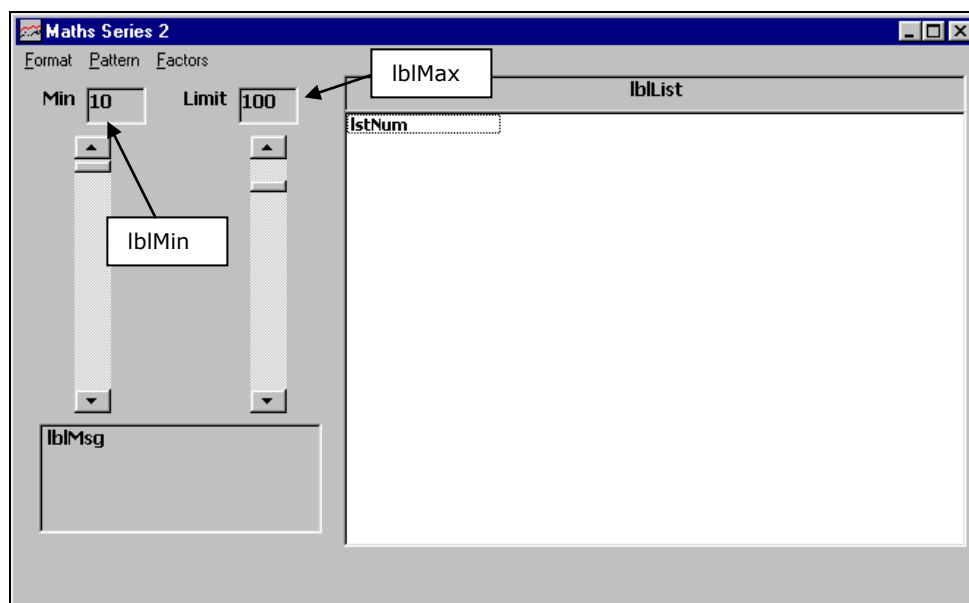
1. Change **Const MaxNum = 1000** (from 50)
2. Sub BuildNumTable is refined to avoid testing multiples of 2, 3, 5 and 7 for primeness.

## Interface

The quickest way to create this interface is to make a copy of the last project and rename the project and form files. You can then modify the interface by deleting the command buttons, adding the different labels and vertical scroll bars and then creating the new menu items. Finally, move the code from each of the old command button events to the new **menu\_Click** events, e.g. **mnuFactorial\_Click**.

Create the interface as shown:

Use 3 dropdown menus (containing 2, 7 and 6 items respectively), 6 labels, 2 vertical scroll bars and 1 listbox.



## Names of Objects

Type of Object	Number	Names of Objects	Simple Initial Properties of Objects
Form	1	Form1	Caption – "Maths Series 2"
Listbox	1	IstNum	Font – Tahoma, Bold, 8
Vertical Scroll Bars	2	vsbMax, vsbMin	
Labels	6	label1, label2, IblMax, IblMin, IblMsg, IblList	Font – Tahoma, Bold, 8
Menu	1	mnuFormat	
		Items	mnuStandard, mnuScientific
			Indented 1 arrow click in Menu Editor Captions - S&tandard, &Scientific mnuStandard – Checked – True
Menu	2	mnuPattern	
		Items	mnuTriangle, mnuSquare mnuOblong mnuPentagonal mnuHexagonal mnuSep mnuFactorial mnuFibonacci
			Indented 1 arrow click in Menu Editor Captions - &Triangle, &Square, &Oblong, &Pentagonal, &Hexagonal, "-", Fa&ctorial, &Fibonacci
Menu	3	mnuFactors	
		Items	mnuPrime mnuComposite mnuAbundant mnuDeficient mnuPerfect mnuSep mnuCheck
			Indented 1 arrow click in Menu Editor Captions - &Prime, &Composite, &Abundant, &Deficient, &Perfect, "-", &Check

## Further Initial Properties of Objects

Object	Property	Initial Value
Form1	Startup Position	2 - Center Screen
	Icon	Any
vsbMax	Min	2
	Max	1000
	Value	10
	LargeChange	25
vsbMin	Min	2
	Max	50
	Value	10
	LargeChange	25
IstNum	Columns	6
		Font -Tahoma, 8
Label1	Caption	Limit
Label2	Caption	Min
LblMax	Caption	""
	BorderStyle	Fixed Single
lblMin	Caption	""
	BorderStyle	Fixed Single
lblMsg	Caption	""
	BorderStyle	Fixed Single
lblList	Caption	""
	BorderStyle	Fixed Single
	Text Alignment	Center

## Code

### MODULE - NUMINFO2.BAS

*'has some changes from NumInfo1.bas (in Program Example 25)*

*'This module is for number processing and storing of information regarding:  
' PRIME/COMPOSITENESS - NUMBER OF FACTORS and SUM OF FACTORS*

Public Const MaxNum = 1000 *'how large do we need the array*

Type NumRec *'the type of data we need in the array*  
 Prime As Boolean *'is the number prime - true or false?*  
 NumFactors As Long *'how many factors does the number have?*  
 SumFactors As Long *'what do all the factors sum to?*  
 End Type

Public NumTable(1 To MaxNum) As NumRec *'declare an array to hold 1000 records about the numbers*

### Sub BuildNumTable()

```
Dim i As Long
'initialises array 'NumTable' with data about primes etc

For i = 1 To MaxNum

    'if i is a multiple of 2, 3, 5 or 7 then its not prime
    If i Mod 2 = 0 And i <> 2 Then
        NumTable(i).Prime = False
    ElseIf i Mod 3 = 0 And i <> 3 Then
        NumTable(i).Prime = False
    ElseIf i Mod 5 = 0 And i <> 5 Then
        NumTable(i).Prime = False
    ElseIf i Mod 7 = 0 And i <> 7 Then
        NumTable(i).Prime = False

    'else check if its prime
    ElseIf Prime(i) Then
        NumTable(i).Prime = True
    Else
        NumTable(i).Prime = False
    End If
    Factors (i) 'adds the information regarding "i's" factors

Next i
```

**End Sub**

### Function Square(x As Long) As Boolean

```
If Sqr(x) = Int(Sqr(x)) Then
    Square = True
Else
    Square = False
End If
```

**End Function**

### Function Prime(n As Long) As Boolean

```
'Determines if a number is prime or not
Dim i As Integer

Prime = True 'Assume Prime

i = 2
While Prime And i <= Sqr(n) 'Check all elements of array
    If n Mod i <> 0 Then
        If Not Square(n) Then
            'Hasn't got that prime number as a factor
            'and isn't a square number
            i = i + 1 'Could be prime
        Else
            Prime = False
        End If
    Else
        Prime = False 'Isn't prime
    End If
Wend
```

**End Function**

### Sub Factors(n As Long)

```
Dim i As Integer

NumTable(n).NumFactors = 0
NumTable(n).SumFactors = 0

For i = 1 To n
    If n Mod i = 0 Then 'its a factor
        NumTable(n).NumFactors = NumTable(n).NumFactors + 1
        NumTable(n).SumFactors = NumTable(n).SumFactors + i
    End If
Next i
```

**End Sub**

## GENERAL SECTION

Option Explicit

*'messages for information display*

```
Const msgTriangle = "Triangular Numbers can be arranged in the shape of a Triangle"
Const msgSquare = "Square Numbers can be arranged in the shape of a Square"
Const msgOblong = "Oblong Numbers can be arranged in the shape of an Oblong"
Const msgPentagonal = "Pentagonal Numbers can be arranged in the shape of a Pentagon"
Const msgHexagonal = "Hexagonal Numbers can be arranged in the shape of a Hexagon"
Const msgFibonacci = "Fibonacci Numbers are derived from adding the previous 2 numbers in the series together. The first two numbers are 0,1"
Const msgFactorial = "A Factorial Number is the number multiplied by all integers from itself down to 1 eg 6! (read as 6 Factorial) = 6x5x4x3x2x1= 720"
Const msgPrime = "A Prime numbers has only 2 factors - 1 and itself"
Const msgComposite = "Composite Numbers have at least 3 factors"
Const msgAbundant = "An Abundant Number is a number where the sum of its factors is more than twice the integer"
Const msgDeficient = "A Deficient Number is a number where the sum of its factors is less than twice the integer"
Const msgPerfect = "A Perfect Number is a number where the sum of its factors is exactly twice the integer"
Const msgCheck = "The data in the array: NumTable; regarding Primeness, Number of factors and Sum of factors for verification purposes"
```

*'Ten interesting Number facts to add to make the program more interesting*

```
Const msgInteresting1 = "No-one has ever created a formula for generating prime numbers"
Const msgInteresting2 = "31, 331, 3331, 33331, 333331, are all prime numbers"
Const msgInteresting3 = "All square numbers are the sum of 2 consecutive Triangular numbers"
Const msgInteresting4 = "36 is both a Square number and a Triangular number"
Const msgInteresting5 = "Fibonacci numbers were named after the wealthy Italian merchant Leonardo Fibonacci (1170-1240) of Pisa"
Const msgInteresting6 = "Emirp numbers are prime numbers whose digits reversed are also prime numbers - eg 13"
Const msgInteresting7 = "A Googol is 10 to the power of 100 - A Leviathan Number is (10 ^666) !"
Const msgInteresting8 = "Factorions are the sum of the factorials of their digits - eg 145! = 1! + 4! + 5!"
Const msgInteresting9 = "Narcissistic Numbers are the sums of the cubes of their digits - eg 153= 1^3 + 5^3 + 3^3"
Const msgInteresting10 = "To date only 30 Perfect numbers have been found - they are all even"
```

Dim i As Long 'used for loops and some calculations

```
'Dim Scientific As Boolean 'used to check which format is
'Dim Standard As Boolean 'required for display of factorial numbers
Dim FormatNum As String
```

```
Const StandardInteger = "#,###"
Const StandardDecimal = "###.##"
Const Scientific = "Scientific"
```

**Private Sub cmdComposite\_Click()**

```
lblMsg.Caption = msgComposite
lblList.Caption = "Composite Numbers from " & vsbMin.Value _
    & " to " & vsbMax.Value
```

```
ListReset
For i = 1 To Val(vsbMax.Value)
    If NumTable(i).Prime = False Then
        lstNum.AddItem i
    End If
Next i
```

**End Sub****Private Sub Form\_Load()**

```
lblMax.Caption = vsbMax.Value
lblMin.Caption = vsbMin.Value
```

```
FormatNum = StandardInteger
BuildNumTable
```

**End Sub****Private Sub Form\_Unload(Cancel As Integer)**

```
RandomMessage
```

**End Sub****Sub RandomMessage()**

```
'randomly displays an interesting number fact
Dim i As Integer, randomMsg As String
```

```
Randomize
i = Int(Rnd() * 10) + 1
Select Case i
    Case 1
        MsgBox msgInteresting1, , "Interesting Fact " & i
    Case 2
        MsgBox msgInteresting2, , "Interesting Fact " & i
    Case 3
        MsgBox msgInteresting3, , "Interesting Fact " & i
    Case 4
        MsgBox msgInteresting4, , "Interesting Fact " & i
    Case 5
        MsgBox msgInteresting5, , "Interesting Fact " & i
    Case 6
        MsgBox msgInteresting6, , "Interesting Fact " & i
    Case 7
        MsgBox msgInteresting7, , "Interesting Fact " & i
    Case 8
        MsgBox msgInteresting8, , "Interesting Fact " & i
    Case 9
        MsgBox msgInteresting9, , "Interesting Fact " & i
    Case 10
        MsgBox msgInteresting10, , "Interesting Fact " & i
End Select
```

**End Sub****Private Sub mnuAbundant\_Click()**

```
lblMsg.Caption = msgAbundant
lblList.Caption = "Abundant Numbers from " & vsbMin.Value _
    & " to " & vsbMax.Value
```

```
ListReset
For i = Val(vsbMin.Value) To Val(vsbMax.Value)
    If NumTable(i).SumFactors > 2 * i Then
        lstNum.AddItem i
    End If
Next i
```

**End Sub****Private Sub mnuCheck\_Click()**

```
lblMsg.Caption = msgCheck
lblList.Caption = "No. Prime? Number of Factors & Sum of Factors"
```

```
lstNum.Clear
lstNum.Columns = 3
lstNum.FontSize = 8
For i = vsbMin.Value To vsbMax.Value
    lstNum.AddItem i & " " & NumTable(i).Prime & " " & _
        NumTable(i).NumFactors & " " & NumTable(i).SumFactors
Next i
```

**End Sub****Private Sub mnuComposite\_Click()**

```
lblMsg.Caption = msgComposite
lblList.Caption = "Composite Numbers from " & vsbMin.Value & " _
    to " & vsbMax.Value
ListReset
```

```
For i = Val(vsbMin.Value) To Val(vsbMax.Value)
    If NumTable(i).Prime = False Then
        lstNum.AddItem i
    End If
Next i
```

**End Sub****Private Sub mnuDeficient\_Click()**

```
lblMsg.Caption = msgDeficient
lblList.Caption = "Deficient Numbers from " & vsbMin.Value _
    & " to " & vsbMax.Value
```

```
ListReset
For i = Val(vsbMin.Value) To Val(vsbMax.Value)
    If NumTable(i).SumFactors < 2 * i Then
        lstNum.AddItem i
    End If
Next i
```

**End Sub****Private Sub mnuFactorial\_Click()**

```
Const msgOverflow = "Maximum Factorial Size is _
    170! (7.26x10 ^ 326)"
Dim Factorial As Double, j As Integer
```

```
On Error GoTo Err_Factorial
```

```
lblMsg.Caption = msgFactorial
lblList.Caption = "Factorial Numbers from " & vsbMin.Value _
    & " to " & vsbMax.Value
```

```
lstNum.Clear
lstNum.FontSize = 6
```

```
For i = Val(vsbMin.Value) To Val(vsbMax.Value)
    Factorial = i
    For j = i - 1 To 1 Step -1
        Factorial = Factorial * j
    Next j
```

```
'Display number according to Format
```

```
If FormatNum = StandardInteger Then
    lstNum.Columns = 1
    lstNum.AddItem i & "! = " & Format(Factorial, StandardDecimal)
Else
    lstNum.AddItem i & "! = " & Format(Factorial, Scientific)
    lstNum.Columns = 3
End If
```

```
Next i
Exit Sub
Err_Factorial:
```

```
MsgBox "Error Number: " & Err.Number & " " & Err.Description _
    & Chr(13) & msgOverflow
```

**End**

**Private Sub mnuFibonacci\_Click()**

```
Dim FirstNum As Double
Dim SecondNum As Double
Dim NextNum As Double
```

```
lblMsg.Caption = msgFibonacci
FirstNum = 1
SecondNum = 1
```

```
lblList.Caption = "Fibonacci Numbers to " & vsbMax.Value
```

```
lstNum.Clear
lstNum.Columns = 2
lstNum.FontSize = 7
```

```
lstNum.AddItem FirstNum
lstNum.AddItem SecondNum
```

```
If vsbMax.Value > 2 Then
```

```
    For i = 3 To vsbMax.Value
        NextNum = FirstNum + SecondNum
        lstNum.AddItem Format(NextNum, StandardInteger)
        FirstNum = SecondNum
        SecondNum = NextNum
    Next i
End If
```

**End Sub****Private Sub mnuHexagonal\_Click()**

*'the nth hex number = the nth square number plus twice the (n-1)th triangle number*

```
lblMsg.Caption = msgHexagonal
lblList.Caption = "Hexagonal Numbers from " & vsbMin.Value
_ & " to " & vsbMax.Value
```

```
ListReset
```

```
For i = Val(vsbMin.Value) To Val(vsbMax.Value)
```

```
    lstNum.AddItem i ^ 2 + (i * (i - 1))
```

```
Next i
```

**End Sub****Private Sub mnuOblong\_Click()**

```
lblMsg.Caption = msgOblong
lblList.Caption = "Oblong Numbers from " & vsbMin.Value _
_ & " to " & vsbMax.Value
```

```
ListReset
```

```
For i = Val(vsbMin.Value) To Val(vsbMax.Value)
```

```
    lstNum.AddItem i * (i + 1)
```

```
Next i
```

**End Sub****Private Sub mnuPentagonal\_Click()**

*'the nth pent number = the nth square number plus the (n-1)th triangle number*

```
lblMsg.Caption = msgPentagonal
lblList.Caption = "Pentagonal Numbers from " & vsbMin.Value
_ & " to " & vsbMax.Value
```

```
ListReset
```

```
For i = Val(vsbMin.Value) To Val(vsbMax.Value)
```

```
    lstNum.AddItem i ^ 2 + (i * (i - 1) / 2)
```

```
Next i
```

**End Sub****Private Sub mnuPerfect\_Click()**

```
lblMsg.Caption = msgPerfect
lblList.Caption = "Perfect Numbers from " & vsbMin.Value _
_ & " to " & vsbMax.Value
```

```
ListReset
```

```
For I = Val(vsbMin.Value) To Val(vsbMax.Value)
```

```
    If NumTable(i).SumFactors = 2 * I Then
        lstNum.AddItem i
    End If
Next i
```

**End Sub****Private Sub mnuPrime\_Click()**

```
lblMsg.Caption = msgPrime
lblList.Caption = "Prime Numbers from " & vsbMin.Value
_ & " to " & vsbMax.Value
```

```
ListReset
```

```
For I = Val(vsbMin.Value) To Val(vsbMax.Value)
```

```
    If NumTable(i).Prime = True Then
        lstNum.AddItem i
    End If
Next i
```

**End Sub****Private Sub mnuScientific\_Click()**

```
FormatNum = Scientific
```

```
mnuScientific.Checked = True
mnuStandard.Checked = False
```

**End Sub****Private Sub mnuSquare\_Click()**

```
lblMsg.Caption = msgSquare
lblList.Caption = "Square Numbers from " & vsbMin.Value _
_ & " to " & vsbMax.Value
```

```
ListReset
```

```
For I = Val(vsbMin.Value) To Val(vsbMax.Value)
```

```
    lstNum.AddItem I * i
```

```
Next i
```

**End Sub****Private Sub mnuStandard\_Click()**

```
FormatNum = StandardInteger
```

```
mnuStandard.Checked = True
mnuScientific.Checked = False
```

**End Sub****Private Sub mnuTriangle\_Click()**

```
lblMsg.Caption = msgTriangle
lblList.Caption = "Triangle Numbers from " & vsbMin.Value _
_ & " to " & vsbMax.Value
```

```
ListReset
```

```
For I = Val(vsbMin.Value) To Val(vsbMax.Value)
```

```
    lstNum.AddItem I * (I + 1) / 2
```

```
Next i
```

**End Sub**

#### **Private Sub vsbMax\_Change()**

*'ensure Min value is always less than or equal to Max value*

```
If vsbMin.Value > vsbMax.Value Then
    vsbMin.Value = vsbMax.Value
End If
```

```
lbMin.Caption = vsbMin.Value
lbMax.Caption = vsbMax.Value
```

**End Sub**

#### **Private Sub vsbMin\_Change()**

*'ensure Min value is always less than or equal to Max value*

```
If vsbMin.Value > vsbMax.Value Then
    vsbMax.Value = vsbMin.Value
End If
```

```
lbMin.Caption = vsbMin.Value
lbMax.Caption = vsbMax.Value
```

**End Sub**

#### **Sub ListReset()**

```
lstNum.Clear
lstNum.Columns = 6
lstNum.FontSize = 8
```

**End Sub**

### **Possible Extensions to the Program**

1. The sub 'mnuFactorial\_Click' still creates problems with the size of the numbers. Add code to increase the width and size of the form and listbox to fit these larger numbers in. Also modify the sub 'ListReset' to resize the form and listbox back to normal. **Hint:** Look at Program Example 3 if you have it available.
2. Have the listbox automatically redisplay the appropriate range every time the scroll bars are moved.
3. Create a second listbox so that two series can be displayed adjacent to each other. This will help in searching for patterns.
4. Modify the program to flag (highlight or bring to attention) double perfect numbers or super abundant numbers.

### **Notes**