Gareth Jones (JLB) – Computing (SAH)

Six times n

1. The task

Problem
Write a for loop that will generate the following table of values.

N	6*N	12*N	18*N	24*N
1	6	12	18	24
2	12	24	36	48
3	18	36	54	72
4	24	48	72	96
5	30	60	90	120

Extension
Generalise the previous program so that the user can choose the multipliers and the number of entries in the table.

2. Design and Development

Initial design:

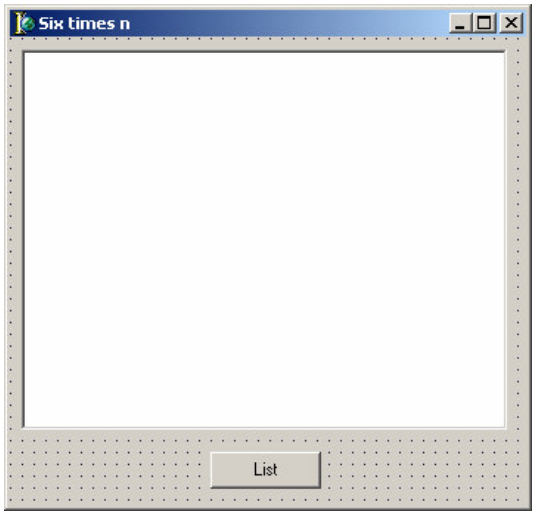
A single button and memo box, clicking the button generates the table of values in the memo field.

Developing the design:

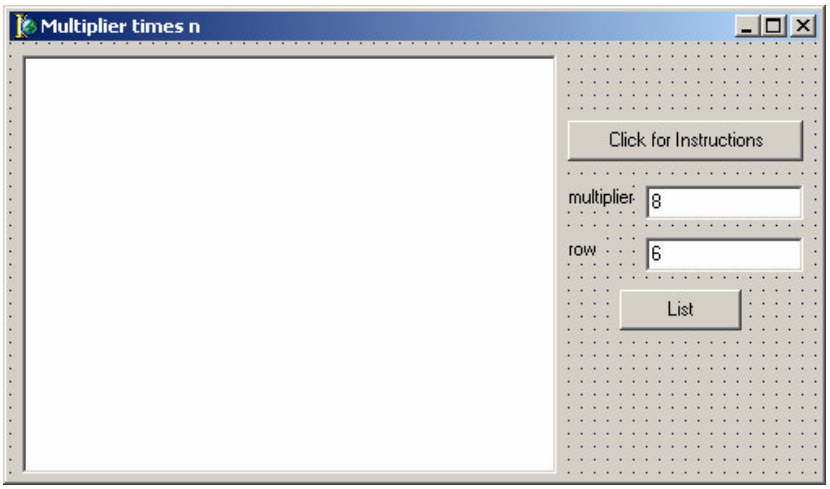
To achieve the extension I added two text boxes to the interface.

Final design:

The final design has two text boxes which are labelled ‘multiplier’ and ‘rows’, two buttons and a text box. I added the second button to display instructions for the user.



Left: Initial design



Right: Final design

3. Problems

Initially I created the tables with spaces rather than tabs, which meant that the lines did not align correctly. I resolved this problem by using a tab, ASCII code 9.

I had also neglected to enter a "=" character after the identifier in each of the for loops, and so the program would not compile first time.

I also had not entered any validation routines. The number of rows must be positive, or no rows are displayed. I therefore entered an if statement to generate an error if the row number was not valid.

4. Unit Listing

Here is the Pascal source code copied from Delphi for the first version:

```
[Code generated by Delphi not included]
procedure TfrmMain.btnListClick(Sender: TObject);

{Local Declarations}

var
i: Integer;

begin
  mmoDisplay.Clear;
//
// Print header line
//
  mmoDisplay.Lines.Add('N' + chr(9) + '6' + 'N' + chr(9) + '12' + 'N' +
    chr(9) + '18' + 'N' + chr(9) + '24' + 'N');
//
// For each row multiply by 1, 2, 3 and 4. Print, then proceed to next row.
//
  for i:= 1 to 5 do
    begin
      mmoDisplay.Lines.Add(IntToStr(i) + chr(9) + IntToStr(6*i) + chr(9) +
        IntToStr(12*i) + chr(9) + IntToStr(18*i) + chr(9) + IntToStr(24*i));
    end;
end;

end.
```

Then after I completed the extension:

```
[Code generated by Delphi not included]
procedure TfrmMain.btnListClick(Sender: TObject);

{Local Declarations}

var
i, Row, RowMax, Number: Integer;

begin
  RowMax:= StrToInt(txtRowMax.Text);
  Number:= StrToInt(txtNumber.Text);
  mmoDisplay.Clear;
//
// Print header line
//
  mmoDisplay.Lines.Add('N' + chr(9) + IntToStr(Number) + 'N' + chr(9) + IntToStr(Number*2) + 'N' +
    chr(9) + IntToStr(Number*3) + 'N' + chr(9) + IntToStr(Number*4) + 'N');
//
// For each row multiply by 1, 2, 3 and 4. Print, then proceed to next row.
//
  for i:= 1 to RowMax do
    begin
```

```
      mmoDisplay.Lines.Add(IntToStr(i) + chr(9) + IntToStr(Number*i) + chr(9) +
        IntToStr(Number*2*i) + chr(9) + IntToStr(Number*3*i) + chr(9) +
        IntToStr(Number*4*i));
    end;
end;

procedure TfrmMain.btnInstructionsClick(Sender: TObject);
begin
  mmoDisplay.Clear;
  mmoDisplay.Lines.Add('Enter the multiplier in the multiplier box, and the number of rows to generate in');
  mmoDisplay.Lines.Add(' the box labelled n. Click List to generate the table.');
end;

end.
```

Then, after adding a validation routine for the row number:

```
Code generated by Delphi not included]
procedure TfrmMain.btnListClick(Sender: TObject);

{Local Declarations}

var
i, Row, RowMax, Number: Integer;

begin
  RowMax:= StrToInt(txtRowMax.Text);
  Number:= StrToInt(txtNumber.Text);
  If (RowMax) < 1 then
    ShowMessage('A valid row number must be entered. This must be a positive integer.')
else
  mmoDisplay.Clear;
//
// Print header line
//
  mmoDisplay.Lines.Add('N' + chr(9) + IntToStr(Number) + 'N' + chr(9) + IntToStr(Number*2) + 'N' +
    chr(9) + IntToStr(Number*3) + 'N' + chr(9) + IntToStr(Number*4) + 'N');
//
// For each row multiply by 1, 2, 3 and 4. Print, then proceed to next row.
//
  for i:= 1 to RowMax do
    begin
      mmoDisplay.Lines.Add(IntToStr(i) + chr(9) + IntToStr(Number*i) + chr(9) +
        IntToStr(Number*2*i) + chr(9) + IntToStr(Number*3*i) + chr(9) +
        IntToStr(Number*4*i));
    end;
end;

procedure TfrmMain.btnInstructionsClick(Sender: TObject);
begin
  mmoDisplay.Clear;
  mmoDisplay.Lines.Add('Enter the multiplier in the multiplier box, and the number of rows to generate in');
  mmoDisplay.Lines.Add(' the box labelled n. Click List to generate the table.');
end;

end.
```

5. Testing

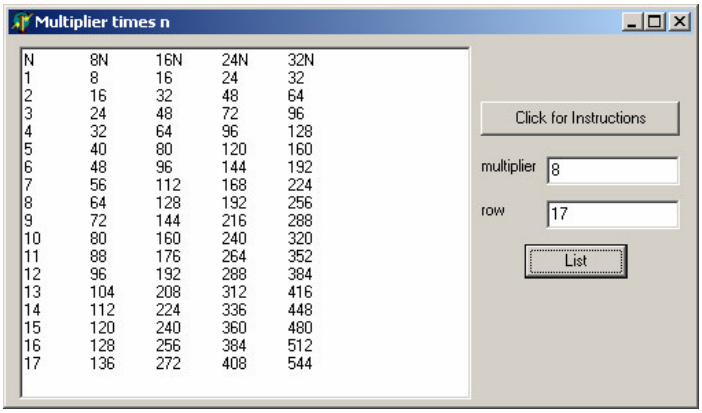
Test	Why testing?	Expected outcome					Actual outcome
Multiplier = 1 Row = 1	Check positive values	N 1	1N 1	2N 2	3N 3	4N 4	As expected
Multiplier = 0 Row = 0	Check invalid data entry	N	0N	0N	0N	0N	As expected
Multiplier = -1 Row = -1	Check invalid data entry	N	-1N	-2N	-3N	-4N	As expected

However, this is not logical. The program cannot cope with negative values or 0, so I added a validation check to make sure any row number entered was positive. I then need to test this.

Test	Why testing?	Expected outcome	Actual outcome
Multiplier = -1 Row = -1	Check validation routine	Error message "A valid row number must be entered. This must be a positive integer."	As expected
Multiplier = 0 Row = 0			

6. Sample run

A screenshot of the program running.



7. Appraisal

I have met the requirements of this task and extended the task further by adding a validation routine.

The user can choose the increment of multiplication for the columns, perhaps a further extension that I could add to this program would be to allow the user to choose the increment and also starting values, not just the maximum value. This could be achieved with little extra code.

In making this program I have practised using loops and become more confident in the Delphi environment. Initially I had used many brackets in a part of code where multiplying three numbers, tried without the brackets and the program still functioned correctly. This should help me write code that is more neatly formatted in future.