**Chapter 3 Repetition - using FOR Loops**

**Repetition** is a very important aspect of programming. It lets us use the big advantages of a computer – which are its **speed** and the fact that it doesn't get bored or tired on repetitive tasks. Repetition is also known (more formally) as **iteration** and (less formally) as **looping**.

// An initial look at a FOR loop

// Chap0301.cpp

#include <iostream>

**using namespace std;**

**Start by doing this**

**Continue as long as**

**Do this each time**

void main()

{

int i;

cout << "For loop demonstration" << endl << endl;

for (i = 1; i <= 10; i = i + 1)

**This statement is not part of the loop**

cout << "It's on its way ..." << endl;

cout << "It's arrived!" << endl;

**system("pause");**

}

The FOR loop is one of three loop structures in C++. Like the others it allows us to execute a number of statements (called the **body** of the loop) a pre-determined number of times. In this program there is only a single statement in the body of the loop:-

**cout** **<<** "**It's on its** **way** ..." **<< endl;**

It is conventional practice to indent the statement(s) in the loop body. This has no effect on the execution of the program - it just helps us to see clearly what comprises the loop body. This is a **convention** to improve the **readability** of your code.

In the example above:-

* a loop variable '**i**' is given a starting value (**1** in this case),
* is incremented by **1** just after the body executes each time around **(**'**i** **=** **i** **+** **1**'),
* and determines the conditions under which looping can continue (as long as '**i** **<=** **10**' where ***<=*** means *less than or equal to*).

So '**i**', in this case, starts at **1**. It then becomes **2**,**3**,**4** etc each time after the loop body is executed. And finally when '**i**' reaches **11** the loop ends - because '**i**' is no longer less than or equal to **10**.

Note that the name '**i**' is by common convention used in C and C++ programs as a FOR control variable like this. Unfortunately C++ programmers have a liking for very short names which don’t describe what the data items are being used for. We'll follow the convention and use **'i'** in FOR loop control – but don't use such short data names anywhere else!

Try the following:-

1. Vary the following (one at a time - restoring each value before changing the next) and observe the results:-

(a) the starting value for '**i**'

(b) the ending value for '**i**'

(c) the increment value (e.g. **i** **=** **i** **+** **2**)

3. What happens if you remove the '**i** **=** **i** **+** **1**' part of the loop control?

3. What happens if you remove the '**i** **<=** **10**' part of the loop control?

4. With the program in its original state, try changing the '**i** **<=** **10**' to '**i** **<** **10**'.

// Using the loop variable in the loop.

// Chap0303.cpp

**#include <iostream>**

**using namespace std;**

**void main()**

**{**

**int i;**

**cout << "Another For loop demonstration" << endl << endl;**

**cout << "It's on its way" << endl;**

**for (i = 2; i <= 10; i++)**

**cout << "It's on its way - now I've said that "**

**<< i << " times" << endl;**

**cout << "It's arrived" << endl;**

**system("pause");**

}

Note a bit of C++ jargon here. Instead of writing '**i** **=** **i** **+** **1**' we can write '**i++**'. The '**++**' operator causes its operand to increment by 1. The inventors of the language were so pleased with this little bit of shorthand that they used it as the name of the language. They are suggesting that C++ is an incremented or improved version of the older C language.

More importantly you can use the value of the loop variable within the loop - in this case it is used as a 'counter' to show how many times the loop has executed. Why does '**i**' start at **2**?

Notice that there is only one statement inside this FOR loop – a **cout** statement. The **cout** is actually too long to fit on this page so it has been split over two lines in this example. This demonstrates that the layout and spacing of your program code is mostly immaterial to the C++ compiler. But neat layout is very important to the human reader.

// A program to print a temperature chart (from 20°C to 40°C).

// Note the use of a compound statement in the FOR loop body.

// Chap0303.cpp

**#include <iostream>**

**using namespace std;**

**void main()**

**{**

**int cent;**

**float fahr;**

**cout << "Temperature conversion" << endl << endl;**

**cout.precision(3);**

// Print the table heading

**cout << "\tTemp C" << "\tTemp F" << endl;**

// Print the table body

**for (cent = 20; cent <= 40; cent++)**

**{**

**fahr = 32 + 1.8 \* cent;**

**cout << endl << "\t" << cent << "\t" << fahr;**

**}**

**cout << endl;**

**system("pause");**

}

In this program we need two statements to be executed each time round the loop:-

1. To work out the fahrenheit version of the temperature

3. To display the next line in the table

Whenever the loop body needs to contain more than one statement, we have to bracket the statements together using braces - **{** and **}**. The body thus becomes a 'compound statement' (which is, as far as C++ is concerned, a single statement).

Forgetting to bracket the statements together like this is a very common error amongst novice programmers – and even amongst experienced programmers! It is a hard mistake to trace because the compiler doesn’t object. The tell-tale symptom is that the first line is repeated the required number of times but that the other lines occur only once. This is because the compiler regards the semicolon at the end of the first line as closing the loop.

Remember that indenting means nothing to the compiler so it can’t guess from the fact that you’ve indented two lines that you intend them both to be part of the loop. You must include them within the braces.

But for the sake of the human reader (including yourself) you should not forget to indent the statements in the loop body.

Edit the previous program so that it looks like the program below:-

// This program displays the chart from high values to low.

// Chap0304.cpp

**#include <iostream>**

**using namespace std;**

**void main()**

**{**

**int cent;**

**float fahr;**

**cout << "Temperature conversion" << endl << endl;**

**cout.precision(3);**

// Print the table heading

**cout << "\tTemp C\tTemp F " << endl;**

// Print the table body

**for (cent = 40; cent >= 20; cent = cent - 1)**

**{**

**fahr = 32 + 1.8 \* cent;**

**cout << endl << "\t" << cent << "\t" << fahr;**

**}**

**cout << endl;**

**system("pause");**

**}**

As we can see, we can have a loop where we decrement the value of the loop variable each time round. This loop was written using '**cent** **=** **cent** **-** **1**', but we can also write it as '**cent--**'. Try it.

Also, see if you can use defined constants instead of the literal values 20 and 40 in this program. Work out what the 20 and 40 represent and replace them with constants you have named suitably and set to the appropriate values.

*// Program for user to stipulate the number of repetitions in a loop.*

*// Chap0305.cpp*

#include <iostream>

**using namespace std;**

void main()

{

int i, numStars;

cout << "Line of stars" << endl << endl;

*// Get the number of repetitions*

cout << "Enter number of stars to display:- ";

cin >> numStars;

*// Display the stars horizontally*

for (i = 1; i <= numStars; i++)

cout << "\*";

cout << endl;

**system("pause");**

}

Note that in the middle section of the FOR loop header, the value of a variable is used to stipulate the limit to the number of loops performed. Previously we have used literal values (**2**, **10**, etc) to fix the upper limit.

Can you make a very simple change which would cause the program to display the stars vertically instead of horizontally?

Now go back to the temperature program and modify it so that instead of always displaying a temperature range of **20** to **40** degrees, it asks the user what should be the starting and ending temperatures of the range.

*If you have time…*

Modify the stars program so that it also asks how many rows of stars to draw, and then draws the required number of rows each containing the desired number of stars.

*Hint* To achieve this you will need two FOR loops – one to control the number of rows and one to control the number of stars. And you need to put the stars loop inside the rows loop – we refer to this as **nested looping**.

(Chap0306.cpp)

Then modify it further so that you ask the user a third question – “How many blank rows do you require between each row of stars?”.

*Hint* Clearly you will now need a third FOR loop – an extra one to print a blank line. But you don’t need three loops fully nested. The trick is to work out the correct position for your new FOR inside the row control FOR.

**FOR practice exercises Test each program thoroughly** before moving on.

3.1 Design and write a program to output the four times table (no input!):-

**1 times 4 is 4**

**2 times 4 is 8**

**…**

**10 times 4 is 40**

*XXFOR01.CPP*

3.2 As problem 3.1 but give the user the choice of which table to see.

*XXFOR03.CPP*

3.3 Design and write a program to prompt for 5 scores and output the total.

*XXFOR03.CPP*

3.4 Design and write a program to output the average of 5 input scores.

*XXFOR04.CPP*

3.5 Design and write a program which asks the user how many scores will be input and then prompts for and totals that many scores.

*XXFOR05.CPP*

3.6 Design and write a program to ask the user how many coins he/she possesses. Prompt for the value of each coin in turn. Output the total value of all the coins.

*XXFOR06.CPP*

3.7 Design and write a program to ask the user for a number as an 'upper limit'. Output all the even numbers up to the limit figure given.

*XXFOR07.CPP*

3.8 Design and write a program to ask the user for a number as an 'upper limit'. Output all the numbers from that limit figure down to zero.

*XXFOR08.CPP*

3.9 Design and write a program to ask the user for a number. Decide whether the input number is a prime number or not and show an appropriate message.

Hint: Run your FOR loop from 3. Each time divide the input number by the loop counter and if you get an exact division (i.e. no remainder) then the input number isn’t prime. If you reach half the input number with no exact divisor then the number is prime.

*XXFOR09.CPP*

3.10 Design and write a program to display a tables grid:-

**1 2 3 4 5 6 7 8 9 10**

**2 4 6 8 10 12 14 16 18 20**

**...**

**10 20 30 40 50 60 70 80 90 100**

Hint: Use one FOR inside another FOR – i.e. nested FOR loops. *XXFOR10.CPP*

**Currency conversion**

The following **FOR** problems all involve converting sums of money from one currency to another - from pounds sterling (£) to euros (€):-

3.11 Design and write a program to ask for five conversions. In each case ask the user for the exchange rate and for the sum (in pounds) to be converted. Show the user the equivalent number of euros each time.

*XXFOR11.CPP*

3.12 As in problem 3.11 above but this time show the user the total converted value (in euros) of all five conversions.

*XXFOR13.CPP*

3.13 As in problem 3.12 above but this time show the user the total converted value in both pounds and euros.

*XXFOR13.CPP*

3.14 As in problem 3.13 above but this time only ask for the exchange rate once and use that same exchange rate for each of the five conversions.

*XXFOR14.CPP*

3.15 As in problem 3.14 above but this time allow the user to make as many conversions as required until 1000 euros or more have been converted.

*XXFOR15.CPP*

3.16 As in problem 3.15 above but this time ask the user to set a maximum euro limit on conversions – instead of assuming a limit of 1000 euros.

*XXFOR16.CPP*

3.17 As in problem 3.16 above but this time also show the user how many conversions were made before the limit was reached.

*XXFOR17.CPP*

3.18 As in problem 3.17 above but this time also show the user the average value of all conversions made before the limit was reached.

*XXFOR18.CPP*

3.19 As in problem 3.18 above but this time also show the user the highest value conversion made before the limit was reached.

*XXFOR19.CPP*

3.20 As in problem 3.19 above but this time also show the user the lowest value conversion made before the limit was reached.

*XXFOR20.CPP*

*These problems are recommended – concentrate on this sequence at first*

*These problems are not like the examples in the notes –leave them until later*

*These problems need IFs as well as FOR loops*