Program Looping

Review

- What is the range of numbers that can be expressed by 16-bit **int** ?
- What is the range of numbers that can be expressed by 2-byte unsigned **int** ?
- What is the purpose of the **%** operator?

- One of the greatest power of computers is their ability to perform repeated calculations
- C program has constructs specifically designed to handle situations where the same code is needed to be used repeatedly
  - The for statement
  - The while statement
  - The break statement
  - The continue statement

Triangular numbers

- The number of dots is takes to form a triangle containing n rows
- If your triangle has one row, number of dots required is 1
    - with two rows the number of required dots is (1 + 2) = 3
    - with four rows the number of dots required is (1 + 2 + 3 + 4) =10

The *for* statement

```
/* Program to calculate the 200th triangular number
   Introduction of the for statement                */

#include <stdio.h>

int main ()
{
    int  n, triangularNumber;

    triangularNumber = 0;

    for ( n = 1;  n <= 200;  n = n + 1 )
        triangularNumber = triangularNumber + n;

    printf ("The 200th triangular number is %i\n", triangularNumber);

    return 0;
}
```

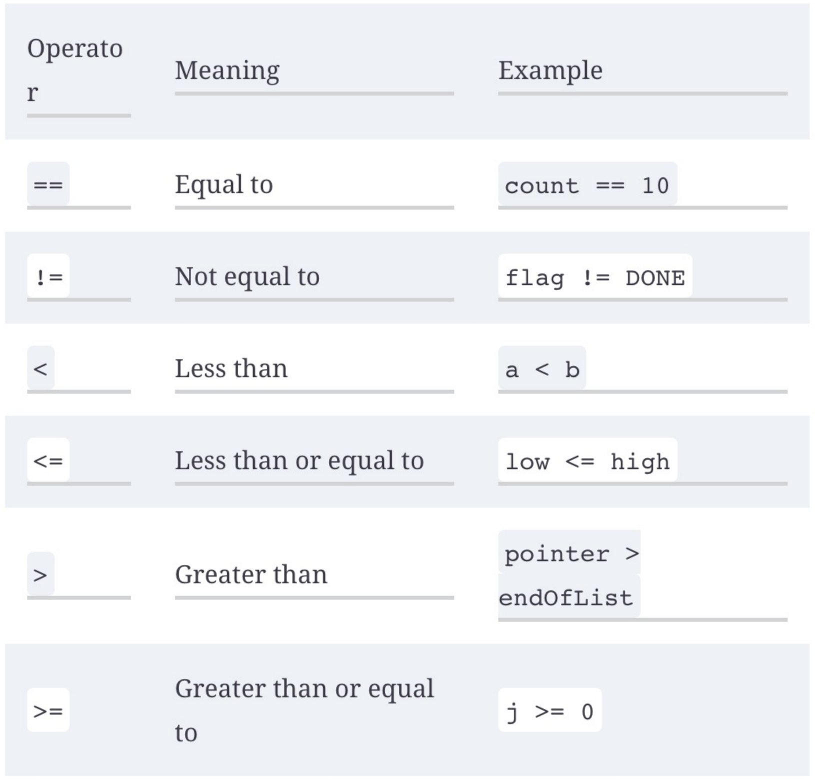
Syntax

```
/* Program to calculate the 200th triangular number
   Introduction of the for statement              */

#include <stdio.h>

int main ()
{
    int  n, triangularNumber;

    triangularNumber = 0;

    for ( n = 1;  n <= 200;  n = n + 1 )
        triangularNumber = triangularNumber + n;

    printf ("The 200th triangular number is %i\n",
triangularNumber);

    return 0;
}
```

- for (*init_expression; loop_condition; loop_expression*)
  program statement (or statements)
- *init_expression* is used to set the initial values *before* the loop begins
  - generally referred to as an *index* variable
- loop continues as long as the *loop_condition* is satisfied
  - loop_condition is specified by relational expression: n <= 200
  - can be read as “n less than or equal to 200.”

Relational Operators

Table 4.1 **Relational Operators**



Operator	Meaning	Example
`==`	Equal to	`count == 10`
`!=`	Not equal to	`flag != DONE`
`<`	Less than	`a < b`
`<=`	Less than or equal to	`low <= high`
`>`	Greater than	`pointer > endOfList`
`>=`	Greater than or equal to	`j >= 0`

Relational Operators

- They have lower precedence than all arithmetic operators
- a < b + c is evaluated as a < (b + c)
- == is the “is equal to” operator
- assignment is done by =

The *while* statement - Syntax

- while (*expression*)

      program statement (or statements)
- The expression specified inside the parentheses is evaluated.
- If the result of the expression evaluation is TRUE, the program statement that immediately follows is executed.

```c
// Program to introduce the while statement

#include <stdio.h>

int main ()
{
     int  count = 1;

     while ( count <= 5 ) {
          printf ("%i\n", count);
          ++count;
     }

    return 0;
}
```

The *for* Loop

for (
init_expression; *loop_condition*;
loop_expression)
    program statement (or statements)

The *while* Loop

- *init_expression*;
  while (*loop_condition*) {
      program statement (or statements)
      loop_expression;
  }