Flow of Control

- LOOPS -

The `while` Repetition Structure

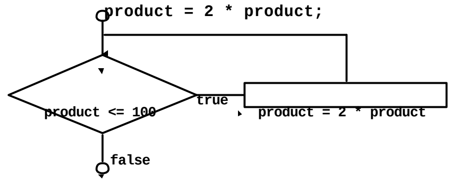
- Repetition structure
  - Programmer specifies an action to be repeated while some condition remains **`true`**
  - e.g.:

    While there are more items on my shopping list
    Purchase next item and cross it off my list
  - **`while`** loop repeated until condition becomes **`false`**

The `while` Repetition Structure

- Example:

```
int product = 2;
while ( product <= 20 )
    product = 2 * product;
```



Example: Counter-Controlled Repetition

- A class of 10 students took a quiz. The grades (integers in the range 0 to 100) for this quiz are available to you. Determine the class average on the quiz
- The algorithm

```
Set total to zero
Set grade counter to one
While grade counter is less than or equal to 10
   Input the next grade
   Add the grade into the total
   Add one to the grade counter
Set the class average to the total divided by ten
   Print the class average
```

```c
/* Class average program with counter-controlled repetition */
#include <stdio.h>

int main()
{
   int counter, grade, total, average;

   /* initialization phase */
   total = 0;
   counter = 1;

   /* processing phase */
   while ( counter <= 10 ) {
        printf( "Enter grade: " );
        scanf( "%d", &grade );
        total = total + grade;
        counter = counter + 1;
    }

   /* termination phase */
   average = total / 10.0;
   printf( "Class average is %d\n", average );

   return 0;   /* indicate program ended successfully */
}
```

```
Enter grade: 98
Enter grade: 76
Enter grade: 71
Enter grade: 87
Enter grade: 83
Enter grade: 90
Enter grade: 57
Enter grade: 79
Enter grade: 82
Enter grade: 94
Class average is 81
```

A Similar Problem

- Problem becomes:
  Develop a class-averaging program that will process an arbitrary number of grades each time the program is run.
  - Unknown number of students
  - How will the program know to end?
- Use sentinel value
  - Also called signal value, dummy value, or flag value
  - Indicates “end of data entry.”
  - Loop ends when user inputs the sentinel value
  - Sentinel value chosen so it cannot be confused with a regular input (such as **`-1`** in this case)

```c
/* Class average program with sentinel-controlled repetition */
#include <stdio.h>
int main()
{
   float average;
   int counter, grade, total;

   /* initialization phase */
   total = 0;
   counter = 0;

   /* processing phase */
   printf( "Enter grade, -1 to end: " );
   scanf( "%d", &grade );
   while ( grade != -1 ) {
        total = total + grade;
        counter = counter + 1;
        printf( "Enter grade, -1 to end: " );
        scanf( "%d", &grade );
   }
   /* termination phase */
   if ( counter != 0 ) {
        average = ( float ) total / counter;
        printf( "Class average is %.2f", average ); }
   else
        printf( "No grades were entered\n" );
   return 0;   /* indicate program ended successfully */
}
```

```
Enter grade, -1 to end: 75
Enter grade, -1 to end: 94
Enter grade, -1 to end: 97
Enter grade, -1 to end: 88
Enter grade, -1 to end: 70
Enter grade, -1 to end: 64
Enter grade, -1 to end: 83
Enter grade, -1 to end: 89
Enter grade, -1 to end: -1
Class average is 82.50
```

The `for` Repetition Structure

- Format when using **`for`** loops
  `for` (*initialization* ; *loopContinuationTest* ; *increment*)
  statement
- Example:
  ```c
  for(counter = 1; counter <= 10; counter++ )
      printf( "%d\n", counter );
  ```
  – Prints the integers from one to ten

No semicolon (;) after last expression

The `for` Repetition Structure

- For loops can usually be rewritten as while loops:

```
initialization;
while ( loopContinuationTest ) {
  statement;
  increment;
}
```

- Initialization and increment
  - Can be comma-separated lists
  - Example:

```
for (i = 0, j = 0;  j + i <= 10; j++, i++)
   printf( "%d\n", j + i );
```

```
/*Summation with for */
#include <stdio.h>

int main()
{
   int sum = 0, number;
   for ( number = 2; number <= 100; number += 2 )
      sum += number;
   printf( "Sum is %d\n", sum );
   return 0;
}
```

Program Output:

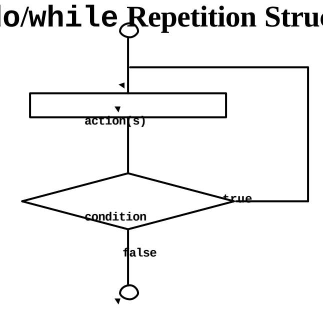
```
Sum is 2550
```

The `do`/`while` Repetition Structure

- The **`do`/`while`** repetition structure
  - Similar to the **`while`** structure
  - Condition for repetition tested after the body of the loop is performed
    - All actions are performed at least once
  - Format:

```
do {
   statement;
} while ( condition );
```

The `do`/`while` Repetition Structure



```
/*Using the do/while repetition structure */

#include <stdio.h>
int main()
{
  int counter = 1;

  do {
      printf( "%d  ", counter );
      counter = counter + 1;
  } while ( counter <= 10 );

  return 0;
}
```

Program Output:

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

Nested Loops

- When a loop body includes another loop construct this is called a *nested loop*.
- In a nested loop structure the inner loop is executed from the beginning every time the body of the outer loop is executed.
- **Example 1**:

```
value = 0;
for (i=1; i<=10; i=i+1)
    for (j=1; j<=5; j=j+1)
          value = value + 1;
```

- How many times the inner loop is executed?

Nested Loops (cont.)

- **Example 2:**

```
value = 0;
for (i=1; i<=10; i=i+1)
    for (j=1; j<=i; j=j+1)
          value = value + 1;
```

How many times the inner loop is executed?

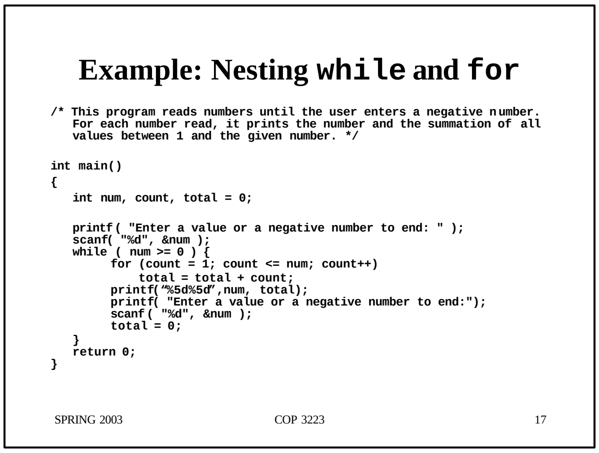
Printing a triangle

- Write a program to draw a triangle like the following: (The number of lines is an input)

```
*
**
***
****
*****
```

- We can use a nested for-loop:

```
for (i=1; i<=num_lines; ++i){
   for (j=1; j<=i; ++j)
        printf("*");
   printf("\n");
}
```



Example: Nesting `while` and `for`

```
/* This program reads numbers until the user enters a negative number.
   For each number read, it prints the number and the summation of all
   values between 1 and the given number. */

int main()
{
   int num, count, total = 0;

   printf( "Enter a value or a negative number to end: " );
   scanf( "%d", &num );
   while ( num >= 0 ) {
        for (count = 1; count <= num; count++)
            total = total + count;
        printf(“%5d%5d”,num, total);
        printf( "Enter a value or a negative number to end:");
        scanf( "%d", &num );
        total = 0;
   }
   return 0;
}
```