# Statistics 243: *class notes*

William J. De Meo

September 12, 1997

**Topics**

## 1 C Programming

**TIP:** If you want to comment out a bunch of code while debugging, but that bunch of code has comments within it, surround the block with an `if` statement that will never be executed, e.g. `if(0)`.

### 1.1 Loops

A very common mistake is to put a semicolon immediately after the while expression; e.g.
`while(n < 100); n++;`
creates an infinite loop.

The *do while* loop:

```
do
        statements
while(expression);
```

Remember the semicolon after *this* while expression!
The listing

```
for(expr1; expr2; expr3)
      statements
```

Is the same as

```
expr1;
while(expr2){
     statements
     expr3;
}
```

A good use for `for` loops is array and vector processing:
`for(sum = 0,i = 0; i < n && sum < 1000; sum += x[i++]);`

## 1.2 The switch Statement

```
char opt;
    switch(opt)
          case 'a':
                statements
                break;
          case 'b':
                statements
                break;
          default:
                statements
                break;
```

## 1.3 Flow of Control

`break` provides an early exit from `for, while, do while, and switch`. It does not break you out of an if statement. You would need a `goto` statement for that.

`continue` causes the next iteration of a `while` or `do while` loop. In a `for` loop, it executes the third expression, and looping continues.

## 1.4 Getting the Program to Run

An *object file* is a file containing machine instructions usually produced by a compiler. In contains statements involving function calls, but the function calls are not always in that object file. So you must link your object files, to create an *executable file*. An executable file contains machine instructions including the associated functions. It is a file that can actually be run. The *linker* combines object files, searches libraries and creates an executable. You often need to tell the linker which libraries you will use:

```
cc prog.c
```

produces a file called `a.out`. Don't let it call it that. Use instead:

```
cc -o prog prog.c
```

To use the math library, use:

```
cc -o prog prog.c -lm
```

Often you will want to compile different parts of your program into different object files called, say, `part1.o, part1.o`, etc. To compile a program without linking, use:

```
cc -c part1.c
```

Once `part1.c` works, you compile `part2.c` with:

```
cc -o prog part1.o part2.c -lm
```