Subject #6: *if* Statement, Relational and Logical Operations

The *if* statement is used to execute a statement or a group of statements only if a certain condition is met.

- Example:

```
if (i == j)
  printf("i and j are equal\n") ;
```

- Instead of a single statement, a whole block can be executed when the condition is fulfilled. Proper indentation should be employed, so that the nesting would be clearly visible.
- An *else* statement is optional after the *if* statement. Multiple options can be achieved by the *else if* idiom. For example:

```
if (i < j) {
    printf("i is smaller than j\n") ;
    i = j ;
} else if (i > j) {
    printf("i is greater than j\n") ;
    j = i ;
} else
    printf("i and j are equal\n") ;
```

- An *else* associates with the closest *else*-less *if* preceding it.
- List of relational operators:

`a == b`	a is equal to b
`a != b`	a is not equal to b
`a > b`	a is greater than b
`a < b`	a is smaller than b
`a >= b`	a is greater than or equal to b
`a <= b`	a is smaller than or equal to b

- Those operators return an *int* value, which is 1 when true, 0 when false. C has no special Boolean type. In fact, the statement inside the *if* is performed exactly when the value inside the brackets is non-zero.
- Beware: the most common bug in C is confusing `=` with `==`.
- The expression `i % j` (for integer `i` and `j`) is equal to the remainder obtained when `i` is divided by `j`. The arithmetical binary operator `%` is called the *modulus* operator. Therefore, the condition `i%2 == 0` checks whether `i` is even.
- Logical operations are needed in order to build combinations of several conditions. For example, in order to check whether an integer variable `i` is positive **and** even, the combined condition is `(i > 0) && (i%2 == 0)` .

- List of logical operations:

`c1 && c2`	c1 And c2
`c1 || c2`	c1 Or c2
`!c`	Not c

- Those operators also return either 1 ("true") or 0 ("false"). They treat a non-zero operand as "true", and a zero one as "false". For example, `0 || 3` is 1.
- A logical expression is evaluated from left to right, and the evaluation stops when the outcome is known. This might be important. For example, the condition (`n > 1 && m/n > 2`) is safe (no division in zero can occur), whereas reversing the order of the checks is not safe.
- Beware: `&` and `|` are *other* legal operators in C. confusing them with `&&` and `||` is a gross bug.
- We give the table of precedence and associativity of all the operations encountered so far:

Operators	*Associativity*
`! ++ -- + - (type)`	right to left
`* / %`	left to right
`+ -`	left to right
`< <= > >=`	left to right
`== !=`	left to right
`&&`	left to right
`||`	left to right
`= += -= *= /= %=`	right to left

- Class exercise: Write a C program that asks for an integer number and checks wheter the following conditions are met:
  - The number has an integer root.
  - The number is even.
  - The number is smaller than 100.
- Change your program to check whether *at least* 2 of the conditions are met.
- The function calculating square roots in *C* is **sqrt()**. In order to use it, and many other mathematical functions, the line:

```
#include <math.h>
```

  should be inserted at the beginning of the program.