What have we seen so far?

- Hello World!
- Simple types:
  - int \%d
  - double \%lf
  - char \%c
  - void ??
- Declaration of variables and assignment of value
- Arithmetic operations
- Type casting
- Precedence of operators in arithmetic operations

In [1]:
```
#include <stdio.h>

int main()
{
    printf("Hello!\n");
}
```

```
Hello!
```

- Increment/decrement operators
- Compound Assignment Operators += -= *= /=

See the Lecture 2 examples

printf()

We will nor shed some light on the *printf* function that we have been using already for a while. This function is defined in the *stdio.h* header and its implementation (that we do not ever look at) can be system specific. It allows us to send formated output to be displayed on the standard output device (*stdout*) in our case this is the screen. (It does not need to be the screen, this could be a file, a communication device such as a printer or a network protocol).

int printf(const char *f, ...)
the definition of *printf*. Returns the number of written characters.

f - text to be written, might contain formating tags (\%specifier) to be replaced by values provided in arguments (not *data safe*).

In [2]:
```
#include <stdio.h> // this is where printf is declared

int main(){
    int a = printf("Hello!\n"); // <- this is 7 charachters
    printf("%d\n", a);
}
```

```
Hello!
7
```

Format tags

Allows to pass more detailed instructions for the way value is to be printed.
\%[flags][width][.prec]spc

- Some Flags: - left justify, +force sign
- width - minimum number of characters to be printed
- .prec - precision
  - for ints: the minimum number of digits to be written
  - floats and doubles:
    - e, E, f: the number of digits to be printed after the decimal point.
    - g, G: maximum number of significant digits to be printed

In [3]:

```
#include <stdio.h> // this is where printf is declared

int main(){
    int a = 10;
    printf("%+d\n", a); // force the sign
    printf("%d\n", a-90); // force the sign

    printf("%5d\n", a+10000); // the width is set to 5

    // with width I can nicly print data!
    printf("%5d\n", 1);
    printf("%5d\n", 10);
    printf("%5d\n", 100);
    printf("%5d\n", 1000);

    // add precision to output
    printf("%5.3d\n", 1);
    printf("%5.3d\n", 10);
    printf("%5.3d\n", 100);
    printf("%5.3d\n", 1000);
}
```

```
+10
-80
10010
    1
   10
  100
 1000
  001
  010
  100
 1000
```

In [4]:

```
#include <stdio.h> // this is where printf is declared

int main(){
    double a = 10;
    printf("%+lf\n", a); // force the sign
    printf("%lf\n", a-90); // force the sign

    printf("%5lf\n", a); // the width is set to 5

    printf("%10.3lf\n", 1.9994);
    printf("%10.2lf\n", 10.55568);
    printf("%10.3lf\n", 100.5555);
    printf("%10.4lf\n", 1000.8888);
}
```

```
+10.000000
-80.000000
10.000000
     1.999
     10.56
   100.555
 1000.8888
```

The cursor control

Controls the position of the cursor

- \n - starts a new line

In [5]:
```
#include <stdio.h> // this is where printf is declared

int main(){
    printf("This is a sentence?\n and this is a new line\n");
    printf("This\b is\b a\b sentence?\b\n");
    printf("This\t is\t a\t sentence\t?\n");
    printf("This\v is\v a\v sentence\v?\naaa\n"); // ??

    printf("bbbbbbbbbbbb\r aaa\n"); // ??

    printf("To print a \\ use  \\\\ \n");
    printf(" \" \n");
}
```

```
This is a sentence?
 and this is a new line
This is a sentence?
This     is      a       sentence        ?
This is a sentence?
aaa
 aaa
To print a \ use  \\
 "
```

Reading input from standard input

scanf please run the examples on a system with a terminal access.

In [6]:
```
#include <stdio.h>

int main()
{
    int a = 0;
    scanf("%d", &a);
    printf("The value of a has been read a=%d\n", a);
}
```

```
The value of a has been read a=0
```

In [7]:
```
#include <stdio.h>

int main(){
  int a;
  printf("Give me an int!\n");
  scanf("%d", &a); //mind the &!!
  printf("a=%d\n", a);

  double b;
  printf(" Give me an double!\n");
  scanf("%lf", &b);

  printf("a=%d b=%lf\n", a, b);
}
```

```
Give me an int!
a=2131140324
 Give me an double!
a=2131140324 b=0.000000
```

Mathematical library: math.h

In [8]:

```
//%cflags:-lm

#include <stdio.h>
#include <math.h>

int main(){
        double a = 6.8;
        printf("%lf\n", a);

        printf("%lf\n", sin(a));
    printf("%lf\n", cos(a));
    printf("%lf\n", tan(a));
}
```

```
6.800000
0.494113
0.869397
0.568340
```

pow calculates power of a value

In [9]:

```
//%cflags:-lm

#include <stdio.h>
#include <math.h>

int main(){
    int a = 5;
    int b = 3;
    //a^b
    printf("%lf\n", pow(a,b));
}
```

```
125.000000
```

In [10]:

```
//%cflags:-lm

#include <stdio.h>
#include <math.h>

int main(){
    double a = 5;
    int b = 3;
    //a^b
    printf("%lf\n", pow(a,b));
}
```

```
125.000000
```

exp(x) is equivalent to e^x

e^{x+y}

In [11]:

```
//%cflags:-lm

#include <stdio.h>
#include <math.h>


int main()
{
    double x=1, y=3;
    scanf("%lf", &x);
    scanf("%lf", &y);

    printf("%lf\n", exp(x+y));
}
```

```
54.598150
```

Do not do this:

pow(exp(1.), x+y)

//%cflags:-lm

include <stdio.h>

include <math.h>

int main(){ double a = 6.8; printf("%lf\n", a);

```
// e^x
printf("%lf\n", exp(1.0));
printf("%lf\n", exp(a));

//natural logarithm ln()
printf("natural log of e^5 is %lf\n", log(exp(5.0)));
//base 10 log: log10
printf("base 10 log of 100 is %lf\n", log10(100.0));

//pow
printf("%lf\n", pow(10.0, 2.4));

//sqrt of a number
printf("sqrt(100) is %.2lf\n", sqrt(100));
```

}

A practice task:

1. A program that prints your name
2. Modify it so it stores your student ID
3. Modify the program so the ID is read from keyboard
4. Write a new program that uses variables of type double to perform mathematical operations
    - Read the values from the keyboard
    - Perform basic arithmetic operations (+,-,/,*)
    - Use functions from math.h to perform more complex operations
        - sqrt, pow, log10, ln ... (look at http://www.cplusplus.com/reference/cmath/pow/)