Subject #11: Functions

- A *function* is a separate piece of the program which performs a certain operation.
- We have been using some functions that are built-in in the C programming language, for example:

```
y = sqrt(x) ;
```

The function sqrt(x) calculates the square root of x, and returns a value of type *double*. printf() is also a function, that performs the operation of printing some text.

- Type the following program in a file named degrees.c, then compile and run it:

```
#include <stdio.h>

#define STOP 1000

/* convert from fahrenheit to celsius */

float get_fahr()
{
    float input;

    printf("Enter a degree in Fahrenheit,\n");
    printf("Enter %d to stop:\n", STOP);
    scanf("%f", &input);

    return(input);
}


float fahr2cels(float f)
{
    return (f - 32.0)*5.0/9.0;
}


main()
{
    float fahr, cels;

    while ((fahr = get_fahr()) != STOP) {
        cels = fahr2cels(fahr);
        printf("%f degrees Fahrenheit = %f degrees Celsius\n", fahr, cels);
    }
}
```

- The definition of a function specifies its arguments and their type, and the type that the function returns. It defines also, of course, the way calculations are to be done. The above program consists of three functions:
  - main() is executed when you run the program. Every program must have exactly one function named main().
  - fahr2cels() is a function that has one argument of type *float* and returns a value of type *float*.
  - get_fahr() has no arguments and returns a value of type *float*.
- A function calling other functions must know the types of arguments they get and return. Therefore, they should be declared, similarly to the declaration of variables. The declarations are usually placed at the beginning of the program after the `#include` statements and are of the same form as the function header line. (*Note*, if function declarations are used, they must match the function definitions. Otherwise, the program may behave unexpectedly.) For library functions, those declarations are provided by the *include* lines. Also, when the called function is defined before the calling one in the same file, declaration is optional. The function main calls other functions and is never called by any function. Therefore, it is convenient to put it at the end of the program.
- The statement:

  return *value* ;

  returns the execution flow to the calling function, and returns the value *value*.
- A function must be called with the correct number of arguments, of the correct types.
- Functions may call other functions (including themselves).
- It is not necessary to use the value that is returned by the function. For example, the function *printf()* returns the number of characters that were printed, so we can write:

  int n;
  n = printf(...) ;

  However, we always used printf() without using its returned value.
- Functions are important for the following reasons:
  1. They allow to define an operation that can then be performed in several places in the program with different parameters.
  2. They allow us to organize the program by separating it into small pieces. A big task can be separated into smaller tasks that are easier to program and to understand.
  3. A function that performs a specific task can be used in more than one program.

Classwork: Printing a table of powers.

- Write a function `int power(int a, int n)` computing the n-th power of a.
- Write a program printing a reasonably sized table of powers, in a nice format.

Subject #12: Local and global variables

- Look at the following program:

```
#include <stdio.h>

int a = 0;

void f(int i)
{
    int k = 0;

    printf("in f   : a = %d, i = %d, k = %d\n", a, i, k);
    a++;
    i++;
    k += 10;
}

main()
{
    int i, j;

    i = 0;
    j = 3;
    printf("in main: a = %d, i = %d, j = %d\n", a, i, j);
    f(i);
    printf("in main: a = %d, i = %d, j = %d\n", a, i, j);
    i = 5;
    f(j);
    printf("in main: a = %d, i = %d, j = %d\n", a, i, j);
}
```

- This program produces the following output:

```
in main: a = 0, i = 0, j = 3
in f   : a = 0, i = 0, k = 0
in main: a = 1, i = 0, j = 3
in f   : a = 1, i = 3, k = 0
in main: a = 2, i = 5, j = 3
```

- The variables that are declared inside a function are called its *local variables*.
  Functions are *not aware* of local variables of other functions. Even if two functions have variables with the same name, there is no connection between them.

- Local variables, like `k`, are created each time the function is called, and destroyed when the function returns.

- The variable `i` in `f` is also a local variable. It is created when `f` is called, and is assigned the value of the expression in the function call. Since it is a copy of the parameter passed to the function, any changes made to it are *not* reflected in the value of the expression passed to the function.
- The variable `a` is a *global variable*. All the functions can access it and change it.
- Excessive use of global variables makes the program hard to understand. Make a variable global only if necessary.