C programming essentials

Basics

reference page: Creference

Compiling

command: gcc -g -Wall -Wextra -std=c99 -o compiledName fileName.c
- g for debugging
- Wall for all warnings
- Wextra for more warnings
- std=c99 for the c standard
- o compinedName for the name of the output file

Generalities

- Commenting: can be done with `/*` and `*/`, or rest of line with `//`
- Adding a library: #include <stdio.h>
- Adding a math library: #include <math.h> and add -lm to compiler gcc

Creating functions

```
int multiply(int first, int second){
  int res = first * second;
  return res;
}
```

stdio library

IO

printf command is the basic print function.
- first argument is a string to be printed.
- Remember to use `\n` since it is not included in printf - next arguments are variables to be substituted to locations by marked by % - [%6.1f] sets 6 spaces for the number and uses one decimal. Brackets not needed.

sign	variable
%d	signed 16 bit int

sign	variable
%f	float 32 bits
%lf	double 64 bit
%c	characters

Flow control

- and is &&, or is ||

If

```
if (ret > 0) {
  doSomething
} else  if (ret < 0) {
  somethingElse;
} else {
  lastSomething;
}
```

switch

```
switch(a) {
  case 'a':
    doSomething;
    break; // remember to break, otherwise it continues directly
  case 'b':
  case 'c': // Note that this takes in consideration both a and b
    doSomethingElse;
    break;
  default:
    doDefault;
    break;
}
```

while

```
while (a < 10) {
  a++;
}
// OR
do {
  a ++;
} while (a < 10)
```

for loop

```
for (int a = 0; a < 10; a++) {
  doSomething;
}
```

For comparison between the two methods, look at the following:

```
alustus;
while (ehto) {
  lause;
  toimenpide;
}

for (alustus; ehto; toimenpide){
  lause;
}
```

Pointers and arrays

The two basic operators of pointers are `&` and `*`.
- `*` is used to create a pointer and as a deferencing or indirection operator
- `&` is called the address-of operator and it yields the address of a variable

The story is briefly the following:

```
#include <stdio.h>
main(void){
  int var = 5; // var is a direct reference to 5, i.e., the variable name
  int `*`p_var = &var; // creates a pointer p_var to the value of var. & yields its address
  printf("Get the value of var through pointer: %d", `*`p_var);
  // at this point p_var is a pointer, &p_var yields its address, &var is an address, and v
}
```