Files

Johdatus ohjelmointiin / Meija Lohiniva, OAMK 2023

File Handling

- The basic file handling operations are:
  - Writing
  - Reading
- The file can be a
  - text file
  - binary file
- For file handling, we need pointers.
- To open a file, we use `fopen()` function.

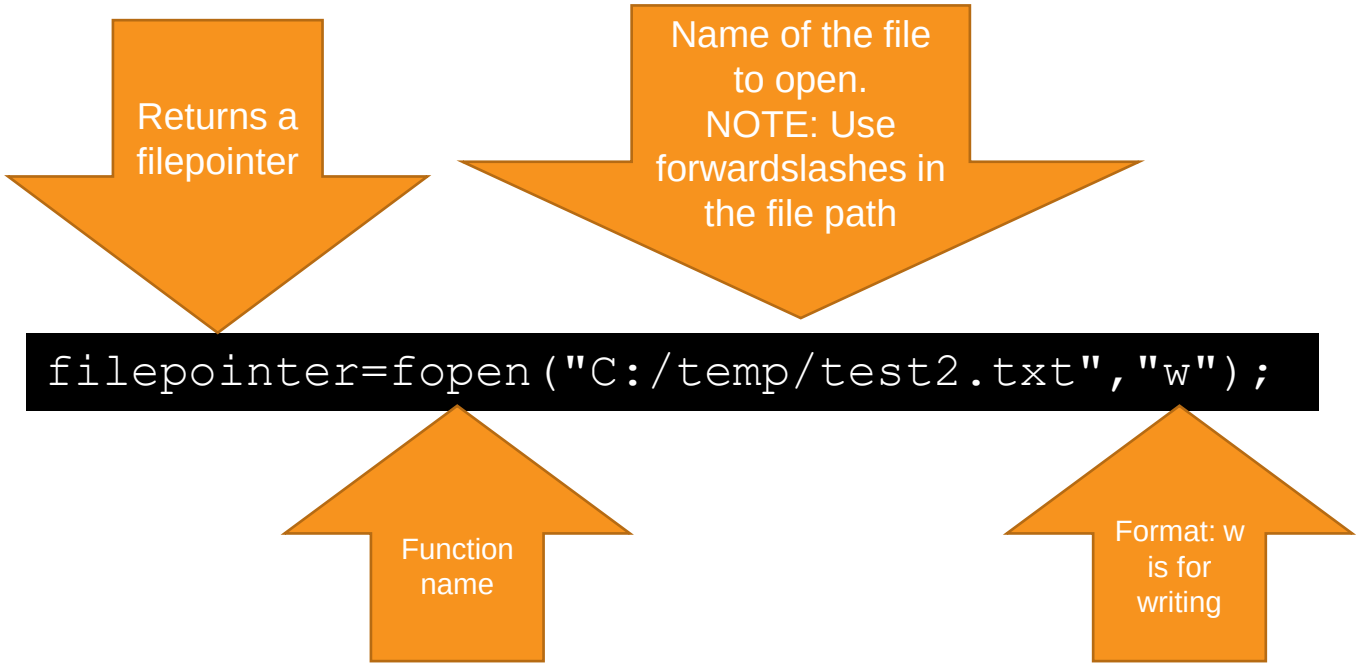
```
FILE *filepointer;
filepointer=fopen("C:/temp/test2.txt","w");
```

OAMK

Open/Close

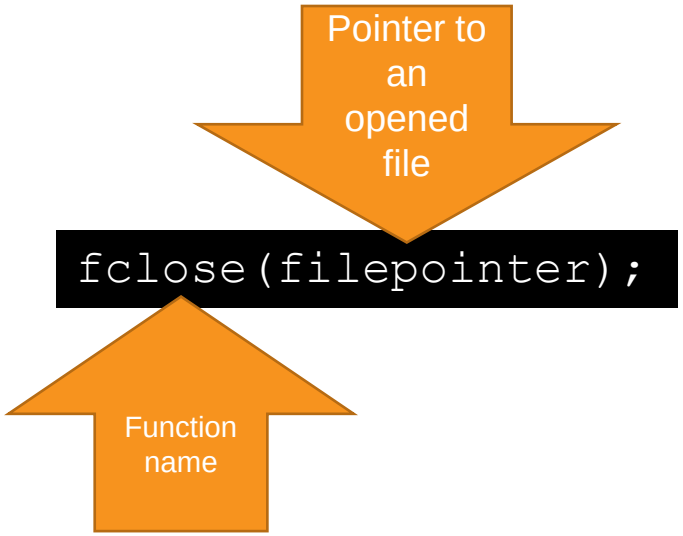
fopen()

- Parameters
  - filename
  - format
    - r: read
    - w: write
    - a: add
- Returns
  - a file pointer
- More info: https://www.tutorialspoint.com/c_standard_library/c_function_fopen.htm



fclose()

- Remember always to close the file with `fclose()`.
- More info: https://www.tutorialspoint.com/c_standard_library/c_function_fclose.htm

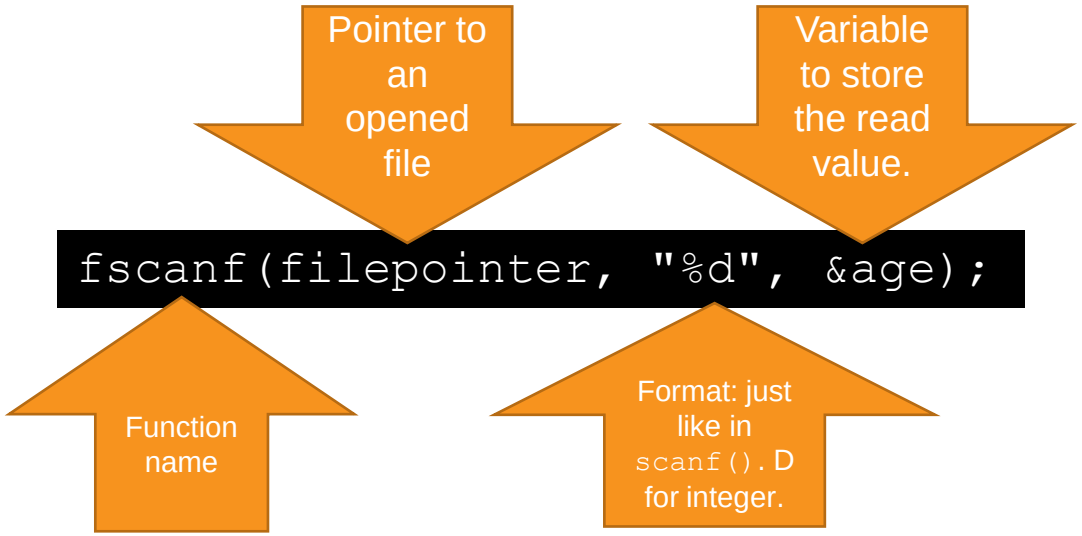


OAMK

Read

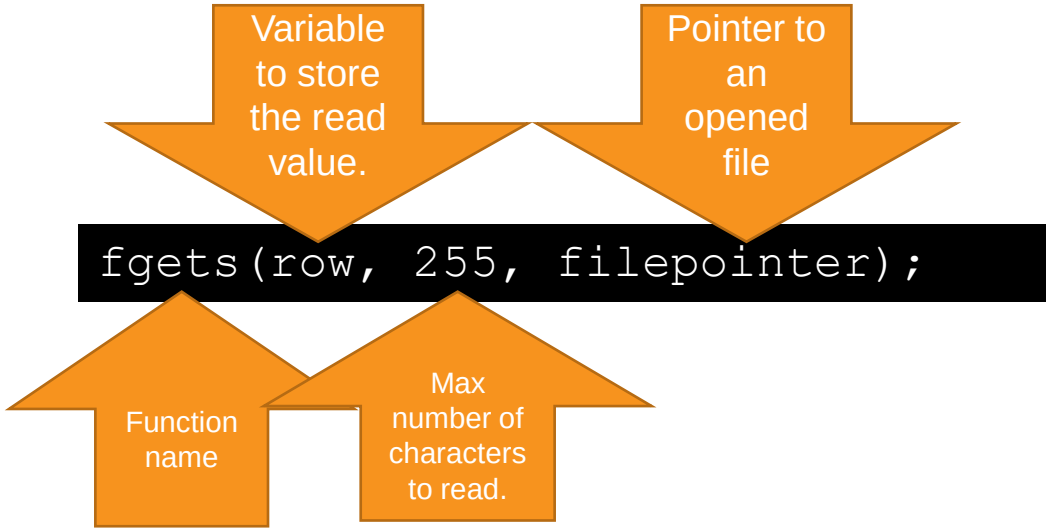
`fscanf()`

- Reads characters from a file till it faces *a space*.
- Very similar to `scanf()`.
- More info: https://www.tutorialspoint.com/c_standard_library/c_function_fscanf.htm



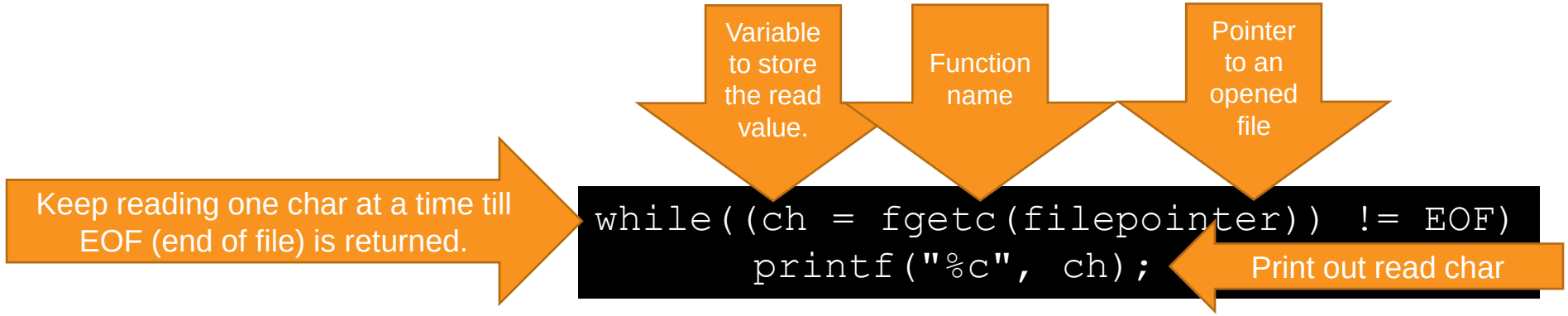
fgets()

- Reads characters from a file till it faces *a new line*.
- Parameters:
  - str: a pointer to a char array to store the value.
  - n: max number of characters to read.
  - stream: Pointer to a file to be read.



- More info: https://www.tutorialspoint.com/c_standard_library/c_function_fgets.htm

fgetc()



- Reads *one character* from a file.
- Parameters:
  - stream: Pointer to a file to be read.
- Retuns:
  - A character read OR
  - EOF if end of file is reached OR
  - an error
- More info: https://www.tutorialspoint.com/c_standard_library/c_function_fgetc.htm

Task 1: First File Read

- Create a simple textfile.
- Write a piece of code with the following functionality:
  1. Open the file.
  2. Read the contents of the file into a string variable.
  3. Print the contents on the screen.
  4. Close the file.

OAMK

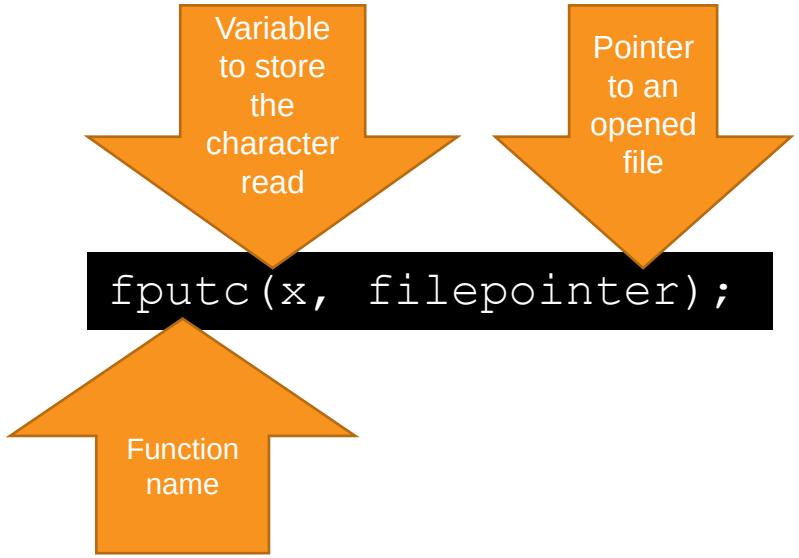
Write

fprintf()

- Writes characters into a file.
- Very similar to `printf()`.
- More info: https://www.tutorialspoint.com/c_standard_library/c_function_fprintf.htm

fputc()

- Writes *one character* into a file.
- Parameters:
  - char: character to write
  - stream: filepointer to an opened file



- More info: https://www.tutorialspoint.com/c_standard_library/c_function_fputc.htm

Task 2: First File Write

- Create a simple textfile.
- Write a piece of code with the following functionality:
  1. Open the file.
  2. Write something in the file.
  3. Close the file.
  4. Open the file again.
  5. Read the contents.
  6. Print the contents on the screen.
  7. Close the file.

Error Handling

Error Handling

- A variety of errors may occur when handling files.
- File handling functions have different mechanisms to notify about errors:
  - `fopen()` returns `NULL` and sets a global variable `errno`.
  - `fgetc()` returns an error.
  - `fputc()` returns `EOF` and sets a global variable `errno`.
- Make sure to handle errors!

```
if (filepointer == NULL) {
   perror("Error while opening the file.\n");
exit(EXIT_FAILURE); }
```

Task 3: Error Handling

- Experiment with the error handling. Try at least the following:
  - Try to open a file that doesn't exist. Show an error on the screen.
  - Try to read a file that is not open. How can you handle this error?
  - Try to write in a file that has been opened for reading, what happens?

Task 4: Other File Handling Functions

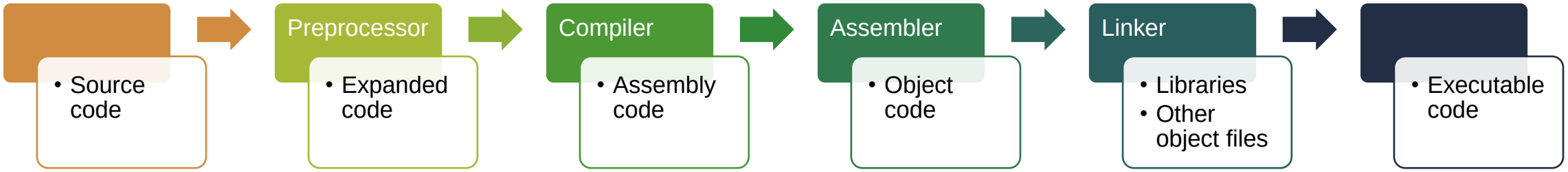
- Familiarize yourself with at least the following file handling functions and create a small piece of code that utilizes them:
  - `feof()`
  - `fputw()`
  - `fgetw()`
  - `fseek()`

https://www.javatpoint.com/file-handling-in-c

C Projects & Files

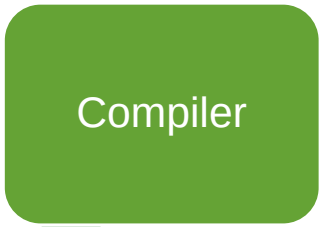
Compilation Process

- C is an executable language.
  - Code is compiled = code is converted into executable code.
- C compilation process can be divided into four steps.

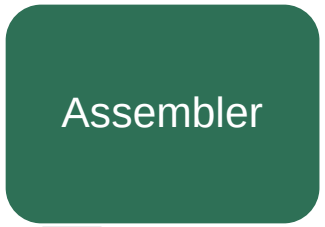


Compilation Process

- Expands source code.
- Handles anything starting with a #.



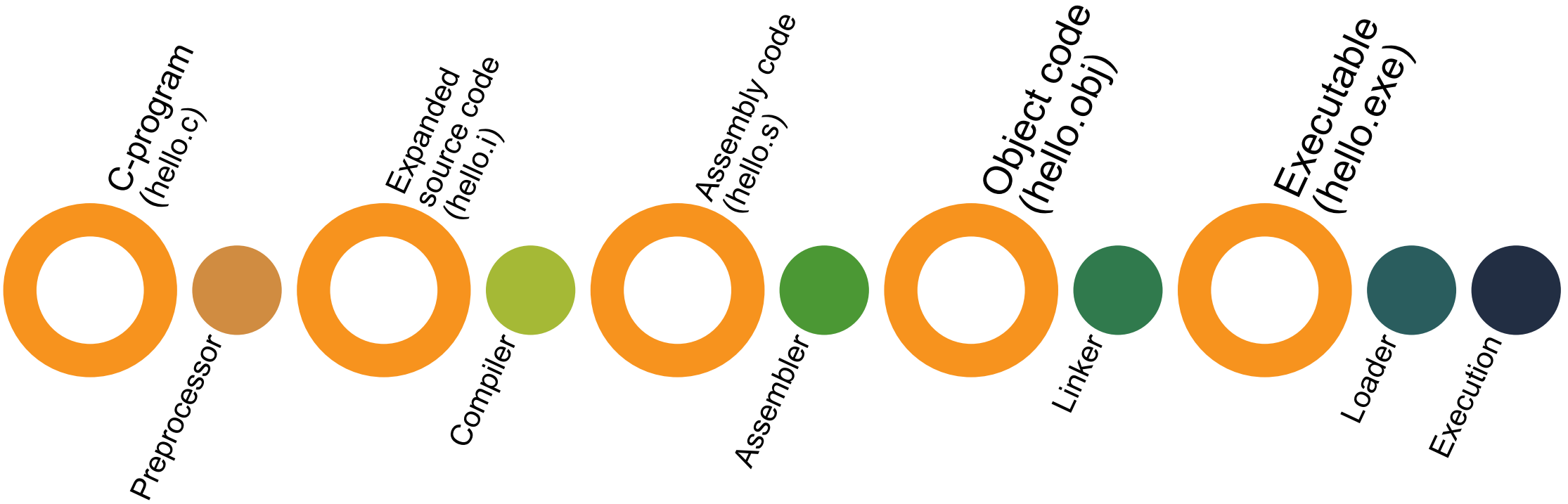
- Converts expanded code into assembly code.



- Converts assembly code into object code.
- File generated has the same name as source code but file extension is .o or .obj.

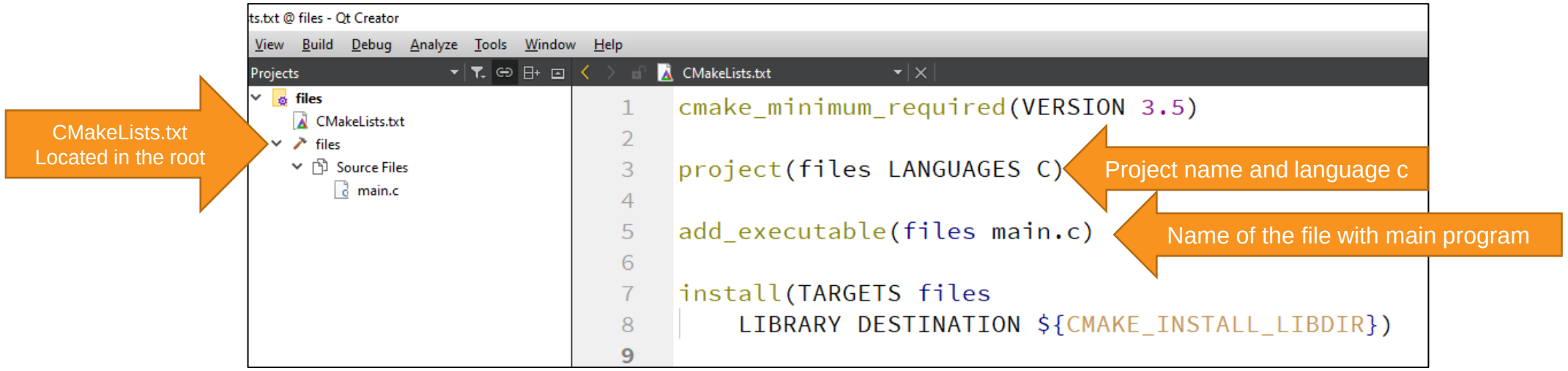
- Combines the object code of library files with the object code of our program.
- Creates an executable file.

Compilation Process



CMake

- CMake is a tool you can use for the compilation process.
- CMake is integrated in Qt Creator but can also be used on command line.
- `CMakeLists.txt` is the main file of the build.



Header Files

- Typically c source code is written in *header* and *source* files.
- Header files – The interface
  - File extension .h.
  - Index of functionality
  - Declarations
  - User defined types
- Source files – The implementation
  - File extension .c.
  - Logic and algorithms

Header Files

- You can write your own libraries.
- Header files are saved with an extension `.h`.
- Header file names are all lowercase.
- You can include your own header files with `#include "myfunctions.h"`.

myfunctions.h

```
int multiply(int, int );
```

Function prototype

myfunctions.c

```
#include "myfunctions.h"

int multiply(int a, int b)
{
    return a*b;
}
```

Function body

main.c

```
#include <stdio.h>
#include "myfunctions.h"

int main()
{
    int calc=multiply(5,6);
    printf("The result is %d\n",calc);
    return 0;
}
```

Note: quotes, not angle brackets

Call to a function in `myfunctions`

Header Files

- Qt will not automatically add new files into CMakeLists.txt -> they will not appear in your project.
- You can modify CMakeLists.txt file by adding the source files in `add_executable()`.
- You can also define variables for source folders in order to not to have to list all of the files separately.

CMakeLists.txt

```
add_executable(files main.c filehandling.c
filehandling.h)
```

Task 5: Headers for File Handling

- Refactor your code from today so that all file handling functionality is in a library `filehandling`.

New Types of Variables

Booleans

- *Boolean* variables are variables that can only get a value of `TRUE` or `FALSE`.
- To use boolean variables in C, you need to include `stdbool` library.
- You can also use integers to achieve similar functionality.
- Boolean values are returned as integers:
  - 1 (or any other number that is not 0) = `TRUE`
  - 0 = `FALSE`

```
#include <stdbool.h>
bool withinWord = false;
if ( withinWord )
{
    wordCount++;
}
```

Constants

- You should avoid "magic numbers" and use *constants* instead.
- Constants are variables that have values that cannot be changed.
- Constants can be integers, doubles, characters, strings etc
- Constants are defined with the keyword `const`.
- Constant names are in CAPITALS.

```
const int LENGTH = 10;
const int WIDTH = 5;
const char NEWLINE = '\n';
const char LOREM_IPSUM_TEXT[] = "dolor sit amet";
```

Task 6: Constants

- Refactor code for Weektask 5.3 Advanced Car (https://github.com/meijastiina/johdatus_ohjelmointiin/tree/main/wk5/5_3_AdvancedCar) to use a constant for a number of cars.
  - What does this do to readability and maintainability?

Task 7: Viikkotehtävät 6

- Start working on the Viikkotehtävät 6.

By Now, Make Sure You Understand These

- How to read a file.
- How to write into a file.

OULUN AMMATTIKORKEAKOULU