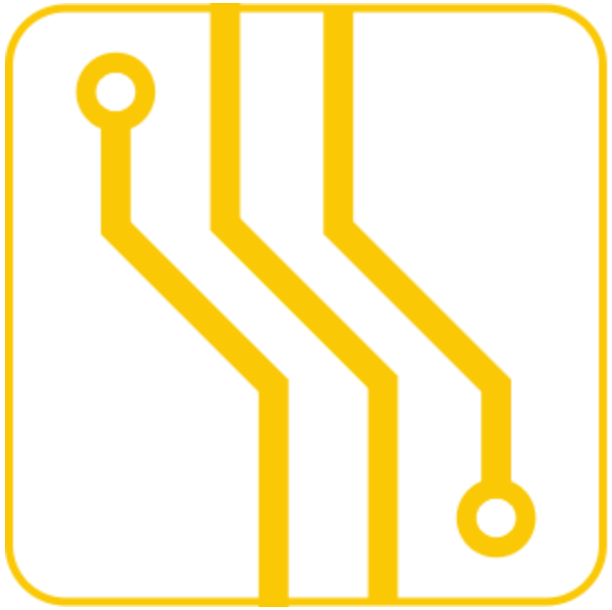
Introduction to programming in C

Who are we?



EmLab

Who are you?

- What are you interested in?
- Is there anything you would like to see in this course?

Course overview

1. Intro to programming in C
2. Basic microcontroller programming, GPIO
3. Timers, PWM, and LED intensity controlling
4. Analog signal
5. Display
6. Serial communication (UART)
7. Sensors, and advanced peripherals
8. Programming self-driving car, and project selection
9. Working on the project
10. Project finalization, what now

Hardware

- 8-bit PIC controllers
- Really low level programming (working with registers)
- Some seminars might include 32-bit ARM controllers for more advanced stuff
- Sensors used within this course might be changed based on your needs

Basic C program (basic_program.c)

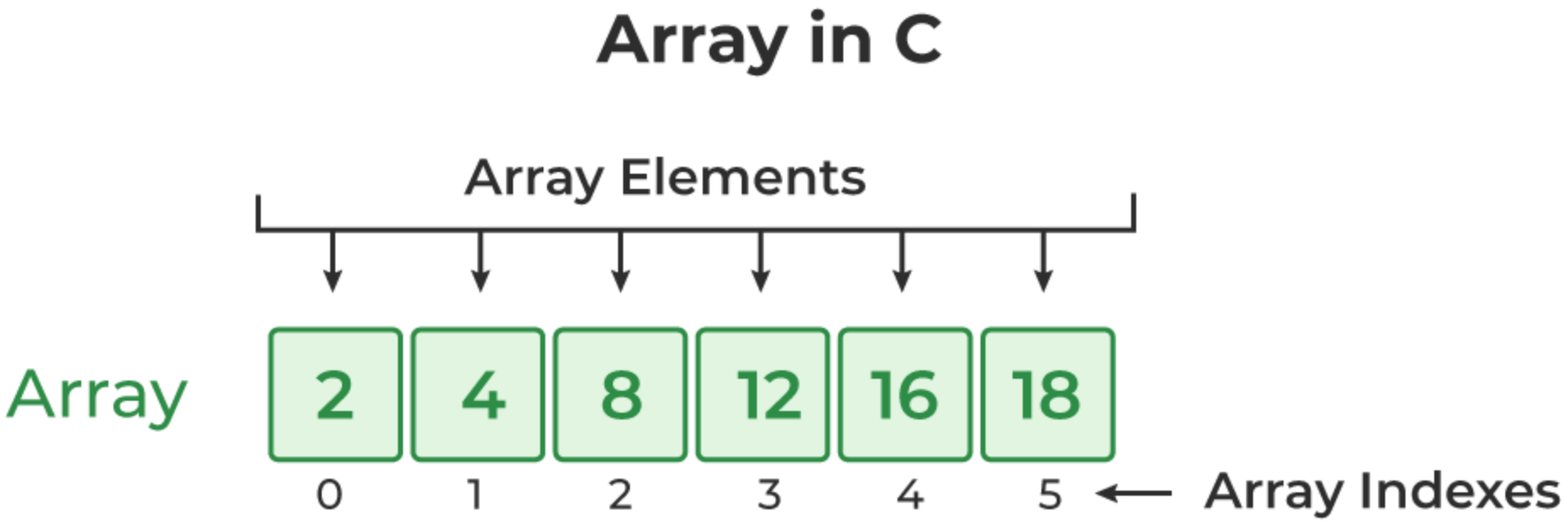
- Include standard library to use print.
- Main function.

Basic data types in C (data_types.c)

- Representation of numbers and characters in computer via bytes
- Signed (two-complements) vs. unsigned numeric types
- Strings of characters are stored in ASCII format
- Integers: `int`, `unsigned int`, `long`, `unsigned long`, ...
- Fixed size integers: `int8_t`, `uint8_t`, ... (see stdint.h)
- Floating point numbers: `float`, and `double`
- Characters: `char`, and `char*`

Aggregate data types (aggregate_data_types.c)

- Structs can compose multiple different data types
- Arrays allow us to store multiple data of the same type



Operators (operators.c)

- Arithmetic: `+`, `-`, `*`, `/`, and `%`
- Logic: `==`, `!=`, `!`, `&&`, and `||`
- Bit: `&`, `|`, `^`, and `~`
- Increment and decrement: `++x`, `x++`, `--x`, and `x--`
- shorthand `a = a + b` is same as `a += b`

Building blocks (building_blocks.c)

- Loops: `for`, `while`, and `do ... while`
- Skip loop: `continue`, and `break`
- Conditionals: `if`, `else`, and ternary
- `switch` with `case`, `break`
- Functions
- Macros
- Jumps

Memory (memory.c)

- Stack (static) vs. heap (dynamic)
- Referencing variables to obtain their addresses within memory.
- Dereference addresses to get values from the given memory location.
- Sometimes we want to use keyword `volatile` to ensure, that the value is read correctly

Code separation

- Code can be separated into multiple functions
- Each function can be stored in a different file
- Use header files to link functions from different file via `#include`
- Include from predefined directories vs. from project files

Let's finally code something (tasks.c)

1. Copy the file with tasks to your computer
2. Read the comments carefully
3. Implement the desired functionality marked with `TODO` comments
4. Check the outcome of asserts
5. If any problems occur, go back to the third step

Additional sources

- Course PV198 materials
- C refference
- Ben Eater (nice videos about how processors work
- Crash Course: Computer Science


That's all Folks!