# INI files

- INI files contain data in basic text format
- The structure of INI file is the following:

```
[section1]
key1 = value1
key2 = value2
key3 = value3

; comment
[section2]
key4 = value4
key5 = value5
key6 = value6
```

- There are:
    - Sections with names between brackets
    - Pairs of keys and values
    - Empty lines
    - Comments

# Project

- Write a C program to parse INI files using only C standard library
- You can work in pairs
- Send the source codes via eKursy
- Hints:
    - Compile with `-Wall -Wextra` flags and fix all warnings
    - Run against Valgrind to make sure there are no memory errors/leaks

## Requirements for 3.0

- The program should accept two command line parameters in the format:

```
$ ./program PATH-TO-INI-FILE.ini section.key
```

For example:

```
$ ./program test.ini data.velocity
```

- Program must print out the value of `key` in the given `section`
- For the example above, the program has to print out value under `velocity` key in `[data]` section
- You can assume the following limits:
    - Number of sections < 30
    - Length of section name < 30
    - Length of key < 30
    - Length of value < 30

## Useful resources

- Main function, command line parameter handling
- File input/output
- `fopen()` reference and man page
- `fclose()` reference and man page
- `fgets()` reference and man page
- `fscanf()` reference and man page

## Requirements for 3.5

- Program must detect a missing section and print out dedicated message e.g. `Failed to find section [data]`
- Program must detect a missing key in a present section e.g. `Failed to find key "velocity" in section [data]`

## Requirements for 4.0

- Parse the data into well defined structures e.g. `struct section { ... };`
- Program must detect invalid identifiers in INI file (section names and keys) i.e. those which contain characters other than letters, digits and dashes (the `-` character)

## Useful resources

- `struct` declarations
- `isalnum()` reference and man page

## Requirements for 4.5

- The limits from the 3.0 requirements are no longer valid
- Therefore, program must accept arbitrary INI files with section name, key or value of any length (without compile-time limit)

### Useful resources

- Dynamic memory management

## Requirements for 5.0

- Program must distinguish types of values between strings and numbers
- Program must be able to understand simple expressions given by the following command line parameters:

```
$ ./program PATH-TO-INI-FILE.ini expression "distance.velocity * travel.time"
```

- The rules for expression evaluation are the following:
  - For numbers, support addition (`+` operator), subtraction (`-` operator), multiplication (`*` operator) and division (`/` operator)
  - For strings, support concatenation (`+` operator)
  - Expression involving operands of different types is invalid (detect it and print out a message) e.g. `distance.velocity + text.message` is invalid
  - Usage of operators `-`, `*` and `/` for string type operands is also invalid e.g. `text.hello * text.world`

### Useful resources

- `strcmp()` reference and man page
- `strtok()` reference and man page
- `strcpy()` reference and man page
- `strcat()` reference and man page

## Example files

- example-3.0.ini
- example-4.5.ini

# Exercises to practice

1. Write a `cat`-like utility, which opens a file and prints its content to the terminal

```
$ cat input.txt
Intelligence forged from circuits and code,
A mind of silicon, a new kind of ode.

$ ./program input.txt
Intelligence forged from circuits and code,
A mind of silicon, a new kind of ode.
```

2. Write a simple `grep`-like utility, which opens a file and prints lines containing a given word

```
$ cat input.txt
Infinite knowledge at its core,
Capable of so much more,
A creation to marvel and admire,
AI's greatness shall never expire.

$ ./program input.txt admire
A creation to marvel and admire,
```

3. Write a program, that will read numeric content and calculate sums of each row

```
$ cat input.txt
58  32  49  22
93  76  68  85
14  25  17  35
42  19  11  70
56  81  37  98

$ ./program input.txt
161
322
91
142
272
```