

# Projekt 1

Odevzdání 1. projektu: nejpozději do pondělního rána

**30. 10. 2023, 8:00**

**StudIS** <https://www.vut.cz/studis/>

***IZP / Odevzdání 1. projektu***

Do informačního systému vložte soubor `keyfilter.c`

# Struktury

V C je **struct** označení součinnového datového typu. Struktura obsahuje konečný a pevně daný počet položek. Každá položka má svůj typ.

Definice struktury / typu a deklarace proměnných:

<pre>struct person {     char name[35];     char address[30]; };</pre>		<pre>typedef struct {     char name[35];     char address[30]; } Person;</pre>
<pre>struct person p1, p2;</pre>	nebo	<pre>Person p1, p2;</pre>

V prvním případě je **person** tzv. *jmenovka* struktury, která se musí uvádět i s klíčovým slovem **struct**.

Ve druhém případě je **Person** nové jméno pro typ struktury.

# Validace kalendářního data

Je dána struktura popisující kalendářní datum:

```
typedef struct {  
    int year;  
    int month;  
    int day;  
} Date;
```

Chceme, aby jen některé trojice celých čísel byly přípustné.

Napište funkci `int validateDate(Date date)`, která ověří, zda její parametr `date` vyjadřuje platné datum mezi roky 1583 a 2500.

- Je-li neplatný už rok (mimo interval [1583, 2500]), funkce vrátí 3.
- Je-li rok v pořádku, ale měsíc mimo interval [1, 12] funkce vrátí 2.
- Při neplatném dnu funkce vrátí číslo 1.
- Pro platná data funkce vrací nulu.

# Validace kalendářního data

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int year;
    int month;
    int day;
} Date;

int validateDate (Date d) { /* DOPLŇTE */ }

int main (int argc, char* argv[]) {
    Date date; int v;
    if (argc != 4) { printf("%s year month day\n", *argv); return 4; }
    date.year = atoi(argv[1]);
    date.month = atoi(argv[2]);
    date.day = atoi(argv[3]);
    printf("%04d-%02d-%02d ", date.year, date.month, date.day);
    v = validateDate(date);
    switch (v) {
        case 0: printf("is valid\n"); break;
        case 1: printf("has invalid day\n"); break;
        case 2: printf("has invalid month\n"); break;
        case 3: printf("has invalid year\n");
    }
    return v;
}
```

# Porovnání dvou kalendářních dat

Máme stejnou strukturu popisující datum:

```
typedef struct {  
    int year;  
    int month;  
    int day;  
} Date;
```

Napište funkci `cmpDate` pro porovnání dvou dat. Funkce

- vrátí `-1`, když je první datum dříve než druhé,
- vrátí `0`, jsou-li obě data stejná,
- vrátí `1`, když je první datum později než druhé.

```
int cmpDate (Date d1, Date d2) { /* DOPLŇTE */ }
```

# Operace s maticemi

## *Jednotková matice*

Napište proceduru `void unit(int n, int a[n][n])`, která nastaví čtvercovou matici řádu  $n$  na jednotkovou.

## *Transpozice matice*

Napište proceduru `void transpose(int n, int a[n][n])`, která transponuje čtvercovou matici na místě.

## *Součet matic*

Napište proceduru

`void add(int m, int n, int a[m][n], int b[m][n], int c[m][n])`, která sečte dvě matice stejného typu a součet uloží do třetí matice.

## *Součin matic*

Napište proceduru `void multiply(int m, int n, int p, int a[m][n], int b[n][p], int c[n][p])`, která znásobí první matici druhou maticí a součin uloží do třetí matice.