

Počítačové systémy

Adam Šmelko

smelko@ksi.mff.cuni.cz

NSWI170 - cvičení

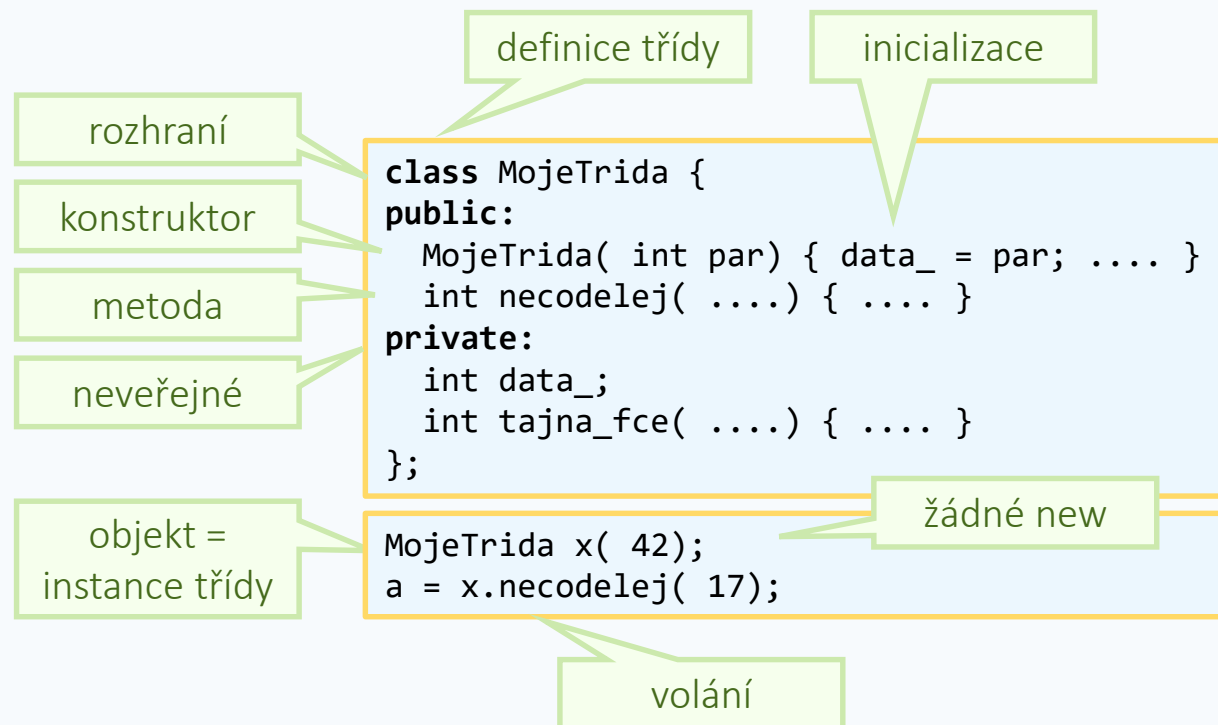
LS 2020/2021

Język C

- struct/class

Třída - encapsulace / zapouzdření

- třída
 - metody nad daty
 - typ
- public
 - veřejné **rozhraní**
- private
 - přístupné pouze z metod třídy
 - slouží k implementaci
 - nikomu do toho nic není
- objekt
 - instance třídy
 - proměnná
- encapsulace
 - data jsou přístupná pouze metodám třídy
 - lokalita přístupu → lokalita výskytu chyb
 - udržitelnost



- inheritance
- polymorfismus

• toto není cvičení z OOP

Encapsulace intervalu

- globální proměnná
 - přístupná komukoliv
 - není zřejmé, kdo za ni má zodpovědnost
 - kdokoliv ji může zničit ☠
- co když jich chci víc?
- encapsulace
 - nezveřejňujte nic, co není **rozhraní**

globální

```
unsigned long last_time = 0;
```

```
void loop_delay( unsigned long interval) {  
    auto cur_time = millis();  
    if( cur_time >= last_time + interval) {  
        last_time = cur_time;  
        x22_semafor();  
        ....  
    }  
}
```

jediná funkce

```
class LoopDelay {  
public:  
    LoopDelay() {}  
    bool delay( unsigned long interval) { .... }  
private:  
    unsigned long last_time_ = 0;  
};
```

zapouzdření

do tohole nikomu nic není

```
LoopDelay leva, prava;
```

```
int x22a_semafor( unsigned long rl, unsigned long rp) {  
    if( leva.delay( rl))  
        ....  
    if( prava.delay( rp))  
        ....  
}
```

nezávislé

Arduino

Tlačítka

Tlačítka

- inicializace

- **pinMode**(button1_pin, **INPUT**);

setup

```
// my funshield library  
....  
constexpr int button[] { ....};
```

- detekce

- bool **digitalRead**(button1_pin);

dekompozice, abstrakce

-  inverzní logika

vyrobte si vlastní funkci

button_pressed(int tlacitko)

- false $\approx$ tlačítko je právě stisknuto

- ➡ 3.1 detekce stisknutého tlačítka

- stisknuté tlačítko i $\Rightarrow$ svítí LED i; nestisknuté tlačítko i $\Rightarrow$ nesvítí LED i

- ➡ 3.2 zapínání / vypínání LED stiskem tlačítka

- detekce "*právě jsem stiskl*"

- evidence předchozího stavu
- detekce změny, nový stav

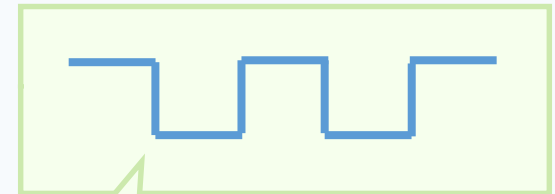
- ➡ 3.3 bitové zobrazení čítače stisků

- tlačítka

- b1 = increment
- b2 = decrement
- b3 = reset

- dekrementace 0 $\Rightarrow$ 15; inkrementace 15 $\Rightarrow$ 0

- bitové operátory: **&**(AND), **|**(OR), **~**(NEGACE), **<<**(LEFT SHIFT), **>>**(RIGHT SHIFT)



jak
zdetekovat ?

5 = **0101**b =



11 = **1011**b =



Tlačítka - blikání

- ➔ 3.4 vypínač s blikáním

- zapnout / vypnout z 3.2
- při delším držení tlačítka se LED rozbliká
 - bliká i po uvolnění tlačítka až do jeho dalšího stisku
- parametry: délka stisku potřebná pro rozblikání, frekvence blikání
- rozmyslet, jaké udržovat stavy
 - tlačítka, LED, blikání
- timeout
 - kdy rozblikat
 - frekvence blikání
- dekompozice, encapsulace

nepoužívejte
delay()

- výčtový typ (enum)

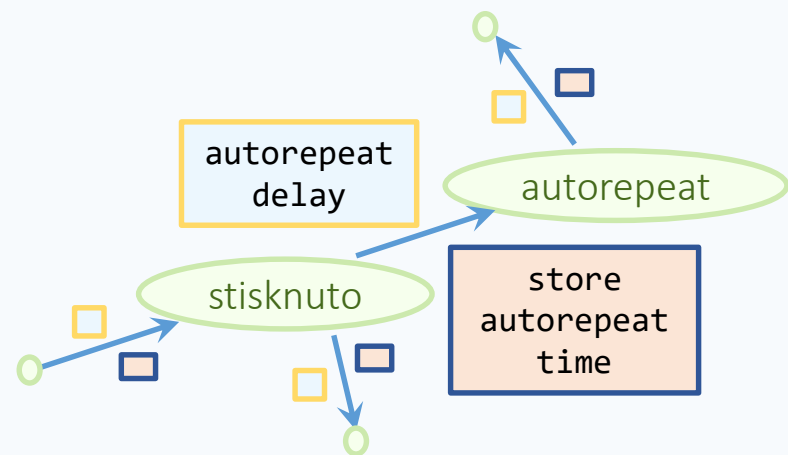
- uživatelsky pojmenované symbolické hodnoty
- může se hodit pro evidenci stavů
- mnohem lepší než stavy 1/2/3
 - také nepoužíváte názvy proměnných p1, p2, p3
 - ... teda ... neměli byste
- lze použít v konstrukci switch

výčtový typ

```
enum helicopter { h_off, h_rotating, h_air };  
helicopter state = h_off;
```

Tlačítka - autorepeat

- ➔ 3.5 automatický čítač
 - čítač stisků z 3.3
 - při delším držení b1/b2 přičítá/odečítá samo (autorepeat)
 - až do uvolnění tlačítka
 - parametry
 - prodleva pro autorepeat, rychlost autorepeatu
 - neprohloupíte, když si namalujete
 - stavy
 - přechody mezi stavy
 - události, které přechody vyvolávají
 - timeout, stisk tlačítka, ...
 - akce na přechodech
 - inkrementace, nastavení času, ...



- ♥ odfiltrování zákmitů
 - tlačítka jsou čínská ➔ nedokonalá
 - můžou někdy samovolně generovat zákmity (jitters)
 - odfiltrovat zákmit kratší než <pojmenovaná_konstanta>
 - např. 80 ms - vyzkoušejte

